ISSN 2007-9737

Evolutive Layers in M3GP Basis for Symbolic Deep Learning Models

Luis Muñoz Delgado*

Instituto Tecnológico de Tijuana, Tijuana,
Mexico

ing.mecanica@tectijuana.edu.mx

Abstract. This paper introduces a novel approach to constructive induction in genetic programming through the Multidimensional Multiclass Genetic Programming with Multidimensional populations also known as M3GP algorithm. M3GP is leveraged to create new features that either augment the original dataset or transform it into a refined version, resulting in improved performance for learning algorithms. With this premise the primary contribution of this work is the integration of an evolutive layer structure within M3GP, where the n best-performing features generated in the previous iterations are reused to continuously enhance the algorithm's performance. This approach parallels the concept of layers in neural networks, establishing a pathway for symbolic construction methods, such as genetic programming, to incorporate layered learning. The second contribution is defining the structure of operation that can be applied in any constructive induction method to connect the improvement of symbolic models, and sampling key points of improvement for configuration options. The findings underscore the potential of evolutionary layering to improve feature generation and model accuracy, marking an advancement in the constructive induction field into a deep learning process. The result shows a higher tendency of fitness improvement against non-layered networks for regression problems and a lower improvement in classification problems, opening the possibilities for a new niche for deep evolutive networks.

Keywords. Genetic programming, M3GP, evolutive layers, classification, regression machine learning, constructive induction, deep learning, symbolic.

1 Introduction

One of the most advanced constructive induction methods in recent years is Multidimensional Multiclass Genetic Programming with Multidimensional populations (M3GP), which leverages multidimensional populations to address the inherent complexity of feature engineering in machine learning. M3GP aims to generate new features that can transform the dataset to ease the learning process for classification or regression models. By constructing multidimensional features, M3GP enhances the representational capacity of the data, making it more suitable for various learning algorithms [2].

In recent years, deep learning has emerged as a powerful paradigm by employing layered architectures that automatically extract hierarchical representations from data. However, these models often operate as black box [12], offering limited interpretability despite their high performance. In contrast, M3GP provides a symbolic approach to feature construction, yielding transparent models that encapsulate valuable insights about complex patterns in the data. By drawing parallels with deep learning's layered structure, the incorporation of evolutive layers within M3GP not only enhances the quality of the generated features but also bridges the gap between the interpretability of symbolic methods and the hierarchical learning capabilities characteristic of deep networks.

These newly generated features contribute to improving solution quality and provide an enriched basis for predictive modeling. Consequently, M3GP shows great promise for enhancing both the performance and interpretability of machine learning models, especially in challenging scenarios where conventional feature engineering approaches and standard deep learning architectures may fall short.

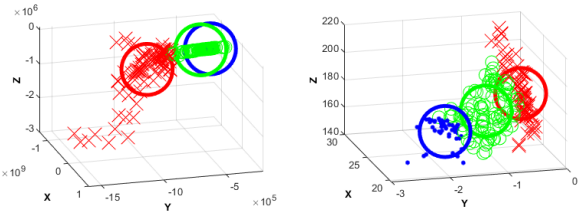

Fig. 1. Classification clusters for a 3-class problem, where the circles represent the class centroids. On the left, a low-fitness M3GP individual; on the right, a high-fitness individual

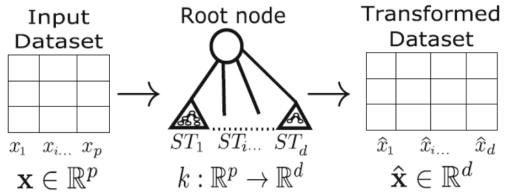

Fig. 2. Special M3GP tree with d dimensions from the root node. Each node contains a standard GP tree, where the input data changes from p to d dimensions, based on the number of nodes in the root node

2 Features in a Dataset

When analyzing a dataset, each column represents a variable that defines features describing the problem, while each row is a sample that provides insight into the desired behavior of the solution. Often, datasets lack a structure conducive to learning, either due to natural complexities or data collection methods. Improving a dataset refers here to enhancing the feature content that describes the problem effectively.

Feature enhancement can occur through either compilation or constructive induction (CI). Compilation rewrites original features in a more compact, logically equivalent form. In contrast, CI improves feature accuracy, sometimes at the cost of increasing dimensionality, by constructing, modifying, or deleting features [15].

This study focuses on data-driven CI, which explores relationships among raw features to transform them into a new, more effective space. CI aims to simplify the learning process by transforming original features into a representation better suited for specific algorithms [22, 3]. This transformation can also enhance interpretability, as a simplified model can emerge when features are sufficiently descriptive.

In addition to the conventional approaches to feature enhancement, deep learning architectures exemplify how layered transformations can effectively reshape raw data into highly informative representations. In deep learning, each layer acts as a feature extractor that progressively abstracts the input data capturing complex, non-linear relationships while filtering out irrelevant information. This hierarchical transformation not only simplifies the data space but also enhances the model's ability to generalize, leading to improved performance in various tasks [1, 5]. Such a mechanism demonstrates that a structured, layered approach can yield representations that are both robust and efficient, providing a compelling parallel to data-driven constructive induction. In this context, the principles underlying deep learning further motivate the exploration of multi-layer strategies such as the evolutive layers proposed in this work to transform and refine the feature space, ultimately contributing to more effective and interpretable solutions.

3 M3GP

M3GP is a tool based on genetic programming (GP) [10] to improve the search space in terms of locally maximizing the distance between classes and thus increasing the performance in classification and for regression the space is changed to a more favorable representation for the learning algorithm. This method is a variant of tree-based GP called Multidimensional Multiclass GP with Multidimensional Populations (M3GP) [17], which is a wrapper approach for supervised classification and regression [18]. M3GP evolves a transformation of the form $k : R^p \rightarrow R^d$ with $p, d \in N$ using a special tree representation see Fig 1, in essence mapping the p input features of the problem to a new feature space of size d. Afterward, M3GP applies the Mahalanobis distance classifier [23] to measure the quality of each transformation based on classification

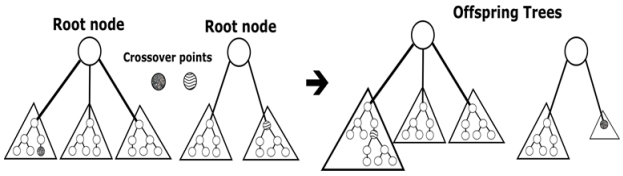

Fig. 3. Standard crossover

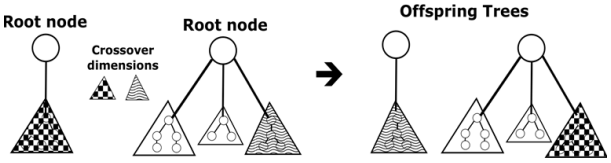

Fig. 4. Swapping of dimensions crossover

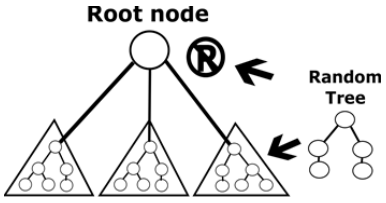

Fig. 5. Standard sub tree mutation, the only node that cannot be mutated is the root node

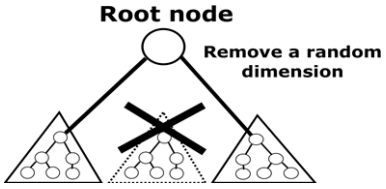

Fig. 6. Remove dimension mutation, randomly removes a leaf from the root node

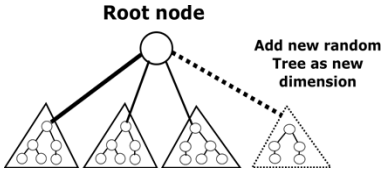

Fig. 7. Add new dimension, adding a new leaf to the root node with a random generated tree

accuracy see Fig 1, for regression multiple linear is applied to generate de prediction model.

In other words, M3GP is a wrapper-based GP performing a feature treatment of the input data, obtaining new data features that improve solution quality. This section describes the key steps in the M3GP evolutionary process:

Initialization of Multidimensional population: M3GP begins by initializing a one dimension population of individuals, where each individual represents a transformation of the original features, as is common in traditional GP. Each feature of the root node can be viewed as a dimension in a multidimensional space, see Fig 2. This multidimensional representation provides M3GP with a more flexible basis for generating solutions that capture complex patterns in data, which is particularly valuable for feature construction.

Fitness Evaluation: Each individual in the population is evaluated based on its ability to improve classification or regression performance. Fitness is typically measured using classification accuracy for multiclass tasks or an error metric for regression, computed by applying the newly generated features to a learning algorithm, such as Mahalanobis distance for classification or multiple linear regression for regression. The fitness evaluation helps identify the most promising individuals to evolve and create new features that better separate or group the target classes or improve prediction for regression.

Selection: M3GP uses a selection mechanism (tournament selection [14]) to choose high-fitness individuals for reproduction. Individuals with higher fitness have a better chance of being selected, ensuring that promising feature representations are preserved and expanded in future generations.

Genetic Operators Crossover: Selected pairs of individuals undergo crossover, where subsets of their features are exchanged to produce offspring. This operator enables the combination of useful features from different individuals, enhancing the diversity of feature space. Two possible crossovers with equal probability are possible standard crossover see Fig 3 and swapping of dimensions crossover see Fig 4.

Genetic Operators Mutation: Mutation introduces slight modifications to the feature set of an individual. In M3GP, mutation can add see Fig 7, modify see Fig 5, or remove see Fig 6 dimensions within the feature set, encouraging exploration

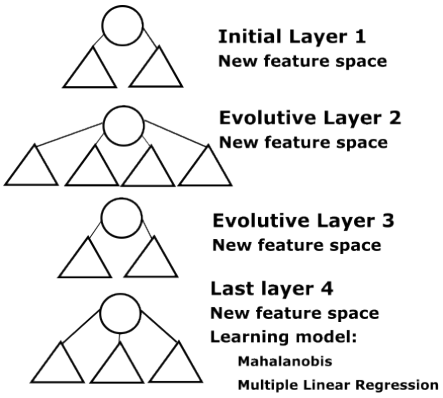

Fig. 8. Four layer sample of evolutive layers, each layer fully runs M3GP to evolving the population to grow in dimensions to improve the feature content, the best n dimensions are chosen to be added to the next layer as data features; because evolution drives the growth development of the tree structure, some layers can have more dimensions than others; all layers have a learning model to guide the search; last layer is define to be the final learning model

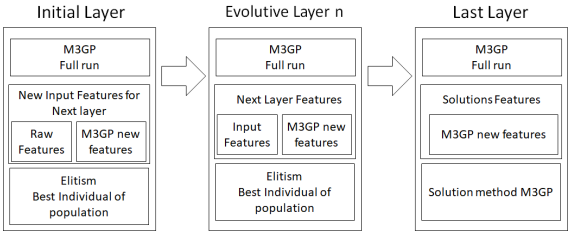

Fig. 9. Evolutive layer structure: the first layer starts with raw features; the best features are added to the next layer to start a new M3GP run. This process repeats for n layers until the last layer is reached

of the feature space and preventing premature convergence; all mutations have equal probability.

Feature Transformation and Constructive Induction: M3GP's evolutionary process is designed to not only improve individual solutions but also to construct new, derived features that may provide higher-level abstractions of the data. Through iterative transformations, M3GP refines these features, encapsulating meaningful patterns that represent the problem's solution space. This constructive induction process distinguishes M3GP from traditional GP, as it allows for cumulative learning, where each generation builds upon the previous ones.

Termination: The algorithm continues evolving the population until a stopping criterion is met, such as a maximum number of generations or satisfactory fitness level is reach. The final solution consists of a set of multidimensional features that optimally represent the problem, facilitating classification or regression tasks for external models in this work Mahalanobis distance and multiple linear regression respectively.

4 Evolutive Layer Concept in M3GP

The evolutive layer is an innovative enhancement added to the M3GP algorithm, inspired by the layered architecture of neural networks, where each layer feeds into the next one. In this approach, after each complete run of the M3GP algorithm, a selection of the top-performing features (for instance, the n best features) is retained and fed into a subsequent execution of a new M3GP run (raw features + best features).

The feature selection method is RELIEFF [8, 9] based on the speed and resources required to run on high feature datasets; however, it is important to emphasize that the feature selection process is a critical step in this framework and can be carried out using any alternative method deemed appropriate. This initial selection constitutes the first and foundational step in the proposed deep learning framework, allowing the evolutionary layering process to build upon the progressively refined feature space and enhancing the overall performance of the algorithm, see Fig 8.

By iteratively carrying forward the most useful constructed inducted features, the evolutive layer allows M3GP to refine its representation across runs, effectively "learning in layers". This resembles how neural networks refine their understanding through layers of neurons, where each layer captures increasingly abstract patterns from the data for a graphic approach see Fig 9.

This layer-based evolution approach allows M3GP to create a cumulative learning structure, where the features from one layer provide a higher-level abstraction, making each subsequent execution potentially more effective at capturing

Table 1. Setting for M3GP with evolutive layer

Settings	Value
Evolutive Layers	50
Population	20 and 50
Generations	50
Initial Dimensions	1
Initialization	6 depth full initialization
Data	70% Training : 30% Testing
Genetic operators	Crossover 0.5, Mutation 0.5
Function set	+,-,*,/ protected as in [10]
Terminal set	Ephemeral random constants [0,1]
Bloat control	17 depth limit
Selection	Lexicographic tournament of size 5
Elitism	Keep best individual in each run.
Elitism for layers	Not keeping the best individual or keeping best individual through layers
New features for layer	n <=5

complex relationships within the data. This not only improves classification or regression accuracy but also introduces a structural hierarchy to the feature construction process, enabling M3GP to handle more challenging problems with greater interpretability and efficiency. Having structural hierarchy allows than any layer can become the final solution, remember that is generating symbolic model, and if any layer provides the expected solution is no necessary to keep adding layers.

Another key mechanism is the elitism between layers. Although it is possible to initiate a subsequent layer without transferring the best individual from the previous layer relying solely on the raw data plus the new features to guide the search, the inclusion of elitism has proven to be a crucial adjustment. By carrying forward the best individual from one layer to the next, the algorithm preserves the quality of the transformations achieved in earlier layers. This continuity not only reinforces the learning process but also ensures that high-performing features are not lost during the transition between layers. In essence, this elitism mechanism facilitates robust inter-layer communication, contributing significantly to the cumulative improvement and overall effectiveness of the deep learning framework proposed in this work.

5 Experiments

This work uses matlab with a variant of GPLAB [19] to run M3GP, all settings can be seen in Table 1. Four dataset of real world problems see Table 2. The main objective of using real world problems is to define a baseline based on previous results [17, 18], with this the objective is to improve over classic M3GP. Each problem executed 30 times.

The MCD3 dataset is a land classification problem based on satellite data, the second classification problem is to be able to detect a heart disease [7]. Yacht [4] is a regression problem to predict hydrodynamic performance of sailing based on different feature adjustments and finally the tower problem [21] predict the output of the distillation process.

The configuration of the layered network can significantly affect solution quality. One key factor is the allocation of computational resources for each run, to balance the use of resources between basic M3GP and new implementation with evolutive layer the following adjustment were done.

One configuration employs a population of 20, 50 generations and 50 layers to achieve the same number of evaluations as the basic M3GP [17, 18] were 100 generations and 500 individual are used (100x500 = 50000). Most deep learning model escalate with more resources to test this possibility a second configuration of evolutive layer of M3GP is tested with a population of 50 individual, increasing the evaluations to 125 000 (50 layers x 50 population x 50 generations).

Table 2. Datasets for experiments

Type	Problem	Information
Classification	MCD3 [13]	Features 6 x 322 Samples 3 Classes
Classification	Heart [7]	Features 13 x 270 Samples 2 Classes
Regression	Yacht [4]	Features 6 x 308 Samples
Regression	Tower [21]	Features 25 x 4999 Samples

Table 3. Overall results for MCD3, the best median solutions between M3GP variants. The p values are based on the Friedman test

Method	Training	p value	Testing	p value	Features	Nodes
M3GP	99.5	N/A	95.3	N/A	5	66
M3GP NE SR	97.7	2e-7	95.3	0.96	4	8
M3GP NE MR	99.1	0.93	95/8	0.07	4	6
M3GP SR	99.1	0.29	95.8	0.16	5	97
M3GP MR	99.1	0.94	96.9	0.00	6	93

Another aspect is the elitism strategy used to maintain the best solution across different layers, which is essential for keeping fitness stable. However, maintaining a high degree of elitism can lead to bloat [11, 20].

To address this, a second configuration was implemented that utilizes only the raw features plus the best n features generated in the previous layer. This small but crucial adjustment leaves the hard task to the genetic operator to find a new best individual to improve the features, in other word the populations restarts each layer, balancing the population tree size back to one dimension, preserving the best features and bloat control by lowering the elitist pressure.

In the end, all the experiments are conformed by five M3GP variants, the first is classic M3GP without any layers or changes, the second method uses the full method of M3GP with evolutive layer and elitism between layers, but runs with the same number of evaluation that classic M3GP which reflect in a lower population of 20 individuals to compensate the extra evaluation by each layer, from now on named M3GP SR (Same Resources), the third variant of the method is M3GP with evolutive layers running with a higher population to increase the search resources, from now on named M3GP MR (More Resources) and finally the fourth and fifth method is M3GP SR and M3GP MR but without elitism between layers, this means that the best individual of the population is not transfer to the new layer, to let evolution start over from scratch with the new features of the previous layer, from now on named M3GP NE (No Elitism) SR and MR.

6 Results

6.1 Classification

The MCD3 classification results are shown in Table 3, showing the best median values for training, testing, features, and nodes, and the p value of the Friedman test to validate the distribution of training and testing against classic M3GP.

The overall training fitness of classic M3GP was not surpassed and there were no statistically significant difference between the best results except with the M3GP NE SR that got a worst result over all other methods. For testing there is statistical significance improvement over classic M3GP by the variant M3GP MR that uses evolutive layers and elitism; Fig 10 illustrates the distribution of the solutions using boxplots to represent quartile ranges.

To evaluate statistical significance among all methods, the Nemenyi test was applied. The results show no significant difference between the elitist variant and classic M3GP for training see

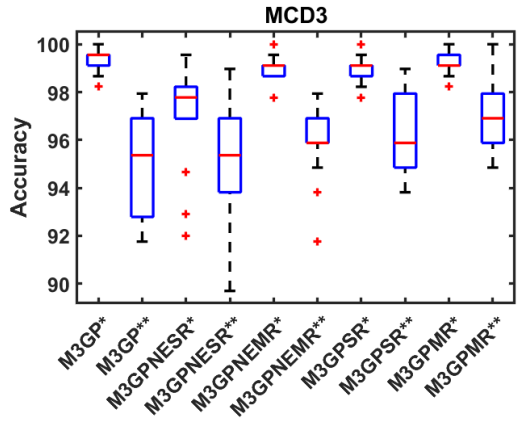

Fig. 10. Training (*) and Testing (**) fitness reached by the classification problem MCD3 at the end of each run

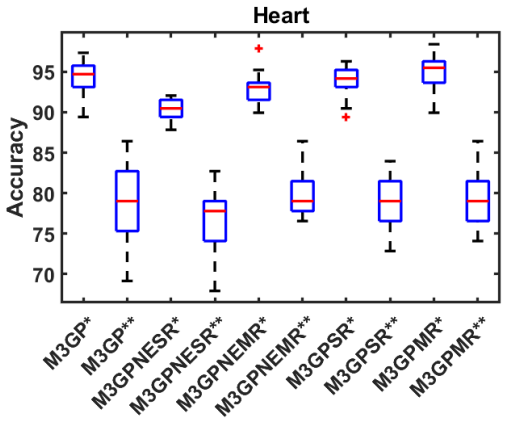

Fig. 11. Training (*) and Testing (**) fitness reached by the classification problem Heart at the end of each run

Table 4. Overall results for Heart, the best median solutions between M3GP variants. The p values are based on the Friedman test

Method	Training	p value	Testing	p value	Features	Nodes
M3GP	94.7	N/A	79	N/A	12	110
M3GP NE SR	90.4	3e-7	77.7	0.61	8	5
M3GP NE MR	93.1	0.16	79	0.48	4	8
M3GP SR	94.1	0.99	79	0.83	12	95
M3GP MR	95.5	0.78	79	0.99	12	142

Table 5. Overall results for Tower, the best median solutions between M3GP variants. The p values are based on the Friedman test

Method	Training	p value	Testing	p value	Features	Nodes
M3GP	19.74	N/A	22.05	N/A	38	354
M3GP NE SR	25.55	0.00	26.89	0.00	15	86
M3GP NE MR	24.36	0.02	25.61	0.00	16	95
M3GP SR	14.77	0.03	17.80	2e-4	207	3718
M3GP MR	14.19	6e-5	18.32	0.01	223	3907

Table 9; however, for testing, the elitist variants were significantly different see Table 10.

The behavior of classification improvement of the evolutive layer for MCD3, can be seen in Fig 14 and Fig 15 where the figures show the maximum, median and minimum of the beast individual in each layer overall the 30 runs, for MCD3 training there is a continual improvement until the 100 accuracy is reach, for testing there is a tendency of improvement on the elitist variants of evolutive layers.

For Heart classification problem the results are shown in Table 4, for training the best result was obtained by M3GP MR but without a statistical significance based on the overall distribution, the worst result was obtained by M3GP NE SR that has lower distribution solution range. For testing fitness the solution was the same on all methods sharing the same distribution with no significant statistical difference, the most compact distribution was obtained by M3GP NE MR, the boxplot distribution are shown in Fig 11.

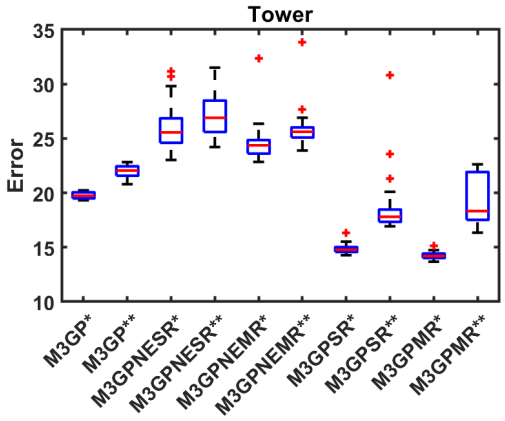

Fig. 12. Training (*) and Testing (**) fitness reached by the regression problem Tower at the end of each run

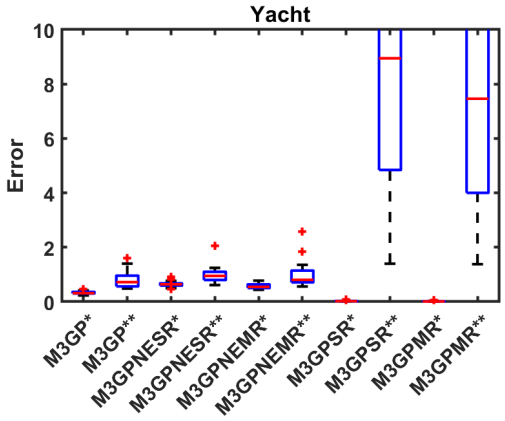

Fig. 13. Training (*) and Testing (**) fitness reached by the regression problem Yacht at the end of each run

Table 6. Overall results for Yatch, the best median solutions between M3GP variants. The p values are based on the Friedman test

Method	Training	p value	Testing	p value	Features	Nodes
M3GP	0.3126	N/A	0.7206	N/A	29	955
M3GP NE SR	0.6395	0.00	0.9479	0.32	14	48
M3GP NE MR	0.5455	0.01	0.8038	0.82	16	71
M3GP SR	0.0194	0.00	8.9462	1e-11	198	4771
M3GP MR	0.0105	0.00	7.4593	5e-12	216	5575

The Nemenyi test results show no significant difference between the elitist variant and classic M3GP for training or testing see Table 7 and Table 8.

The behavior of classification improvement of the evolutive layer in the heart problem, are shown in Fig 16 and Fig 17, it shows a continual improvement over all variants with elitism, on non-elitist variants show spikes of lost and improvement over each layer, generates by not keeping the best individual between layers.

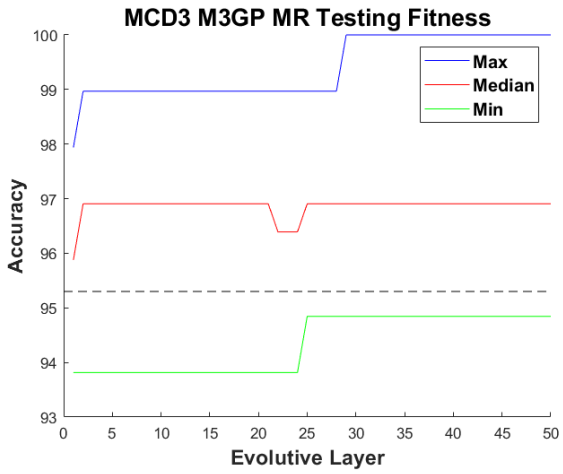

Fig. 14. MCD3 problem with More Resources that classic M3GP, testing fitness reached by the end of each evolutive layer in range of max-median-min, dotted line represents the best median of classic M3GP

6.2 Regression

In the regression problem, Tower results shown in Table 5, showing the best median values for training, testing, features, and nodes and the p value of the Friedman test to validate the distribution of training and testing against classic M3GP. The worst results were obtained by the no elitism variants (regression problems are evaluated minimizing the root mean square error). The best results were obtained by both variants with

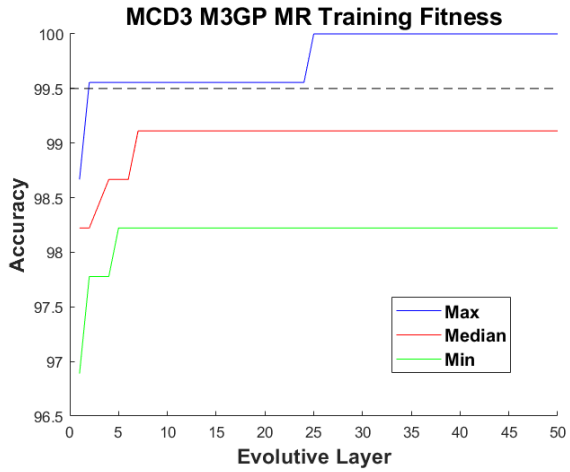

Fig. 15. MCD3 problem with More Resources that classic M3GP, training fitness reached by the end of each evolutive layer in range of max-median-min, dotted line represents the best median of classic M3GP

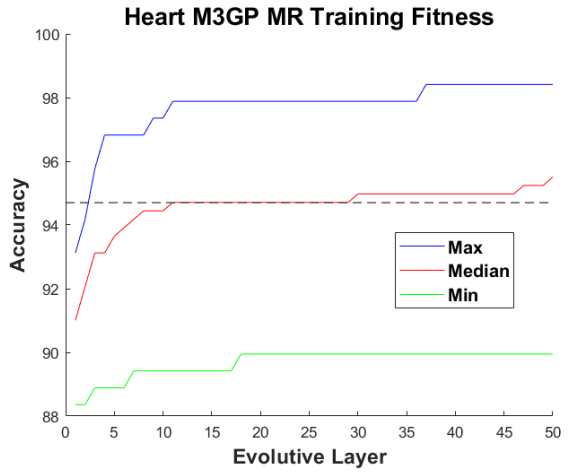

Fig. 16. Heart problem with More Resources that classic M3GP, training fitness reached by the end of each evolutive layer in range of max-median-min, dotted line represents the best median of classic M3GP

evolutive layers with elitism see Fig 12, the improvement beats classic M3GP solution quality by 5.5 in training and 4.25 in testing, and also the range of solution is statistically significant p value of 0.01 or less.

To evaluate statistical significance among all methods for tower, the Nemenyi test was applied. The results show significant difference between the elitist variant and classic M3GP for training see Table 11; however, for testing, the elitist variants are not significantly different see Table 12.

The training fitness improves through all the layers in a stable manner see Fig 18, where each layer has a tendency to lower the root mean square error. But for testing fitness just the maximum shows overfitting but after some layers a tendency to recover alignment with the median and minimum see Fig 19.

For last the regression problem Yacht, the results are shown in Table 6, were the best median values for training, testing, features, and nodes and the p value of the Friedman test to validate the distribution of training and testing against classic M3GP. The worst results were obtained by the no elitism variants, which is expected by not keeping in the best solution to the next layer it forces the layer to start over the search, keeping a low bloat solution with fewer features and nodes, also a low overfitting between training and testing fitness.

The best training results were obtained by M3GP with evolutive layers and elitism, the improvement is significant by a whole digit but the improvement came with a high bloated solution, overfitting the testing solution quality, best testing values were obtained by classic M3GP.

The quality of solutions are statistically different with a p value of 0.00 of the Friedman test for training, where each variant has a statistically significant separation from classic M3GP, it is also different from M3GP NE and M3GP evolutive layers, M3GP NE is different from M3GP evolutive layers. On testing there's only two groups that are statistically significant M3GP and M3GP NE are equal, but different from M3GP SR and M3GP MR evolutive layers. The overall range of solution quality shown in Fig 13. The Nemenyi test show significant difference between the elitist variant and classic M3GP for training see Table 13 and testing see Table 14.

From Fig 21 we can see behavior of fitness through each evolutive layer, on each figure the classic M3GP median appears as a dotted line (fitness to beat). The no elitism variants show the erratic behavior produced by not applying elitism

ISSN 2007-9737

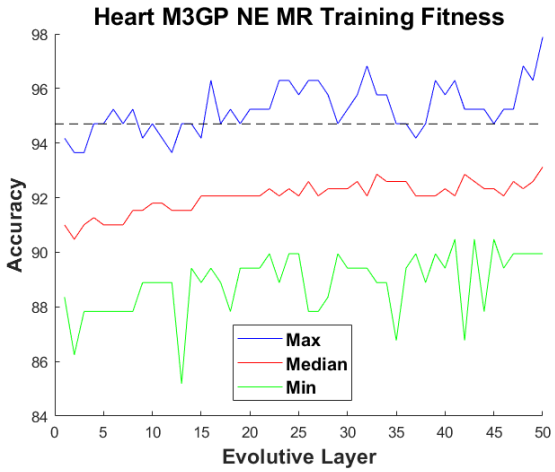

Fig. 17. Heart problem with No Elitism and More Resources that classic M3GP, training fitness reached by the end of each evolutive layer in range of max-median-min, dotted line represents the best median of classic M3GP

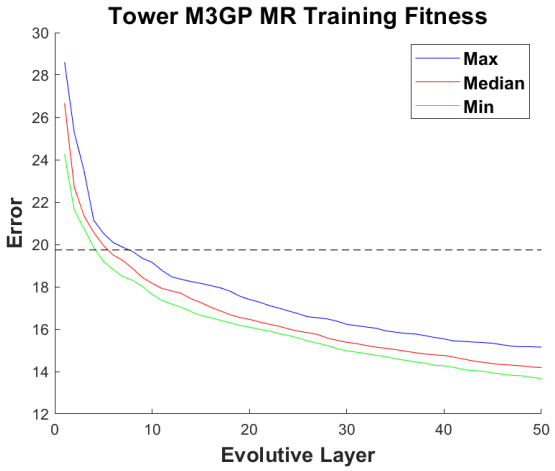

Fig. 18. Tower problem with More Resources that classic M3GP, training fitness reached by the end of each evolutive layer in range of max-median-min, dotted line represents the best median of classic M3GP

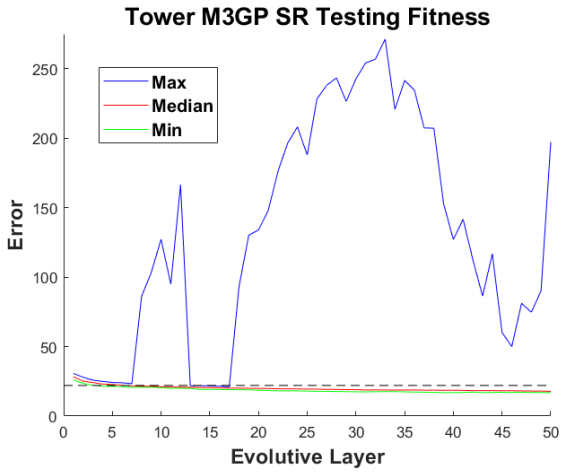

Fig. 19. Tower problem with Same Resources that classic M3GP, testing fitness reached by the end of each evolutive layer in range of max-median-min, dotted line represents the best median of classic M3GP

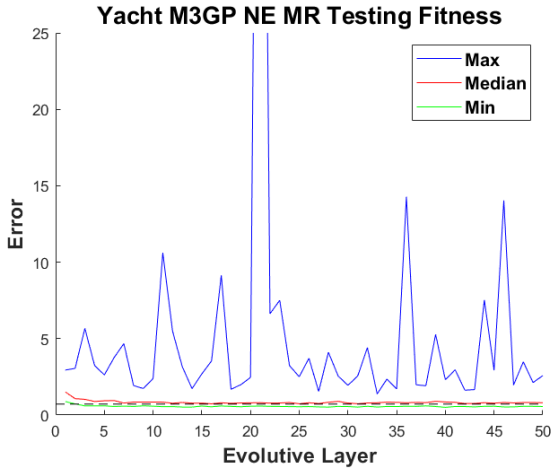

Fig. 20. Yacht problem with More Resources than classic M3GP but without elitism between layers. Testing fitness reached by the end of each evolutive layer in range of max-median-min, dotted line represents the best median of classic M3GP

where layer restarts most of the time with a worse fitness, on the other hand elitism provides stable improvement of fitness see Fig 20, improving over the median even the max, median and min values are below the goal (remembering that for regression lower values is better). The high overfitting can be seen in Fig 22, a part of the max values were cut off to make visible the median and min values.

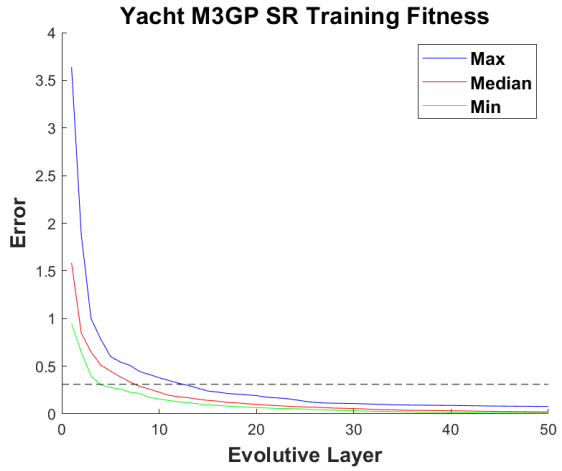

Fig. 21. Yacht problem with Same Resources than classic M3GP but with elitism between layers. Testing fitness reached by the end of each evolutive layer in range of max-median-min, dotted line represents the best median of classic M3GP

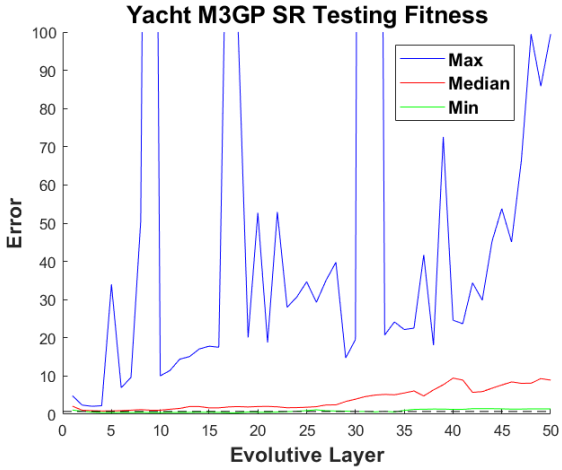

Fig. 22. Yacht problem with Same Resources as classic M3GP, with elitism between layers, testing fitness reached by the end of each evolutive layer in range of max-median-min, dotted line represents the best median of classic M3GP

7 Conclusions

7.1 Summary

The presented approach offers a way to improve classification and regression accuracy without changing any settings of existing methods; the improvement comes from introducing an evolutive layer for feature construction, providing a new learning structure where the induction of features built upon each layer creates further opportunities for improvement.

The new structure can operate with the same resources as classic M3GP to achieve a good solution; however, allocating more resources increases both solution quality and complexity.

The variant without elitism shows a loss of fitness at each layer change by starting a new layer population with improved features from the

Table 7. Nemenyi statistical test for training on the heart problem: "Equal" means that the distributions of solutions are not significantly different (with an alpha of 0.05); "Not" indicates that the distributions are significantly different; and "Not Applicable" (NA)

Train Heart	M3GP	M3GP NE SR	M3GP NE MR	M3GP SR	M3GP MR
M3GP	NA	Not	Equal	Equal	Equal
M3GP NE SR		NA	Not	Not	Not
M3GP NE MR			NA	Equal	Not
M3GP SR				NA	Equal
M3GP MR					NA

Table 8. Nemenyi statistical test for testing on the heart problem: "Equal" means that the distributions of solutions are not significantly different (with an alpha of 0.05); "Not" indicates that the distributions are significantly different; and "Not Applicable" (NA)

Test Heart	M3GP	M3GP NE SR	M3GP NE MR	M3GP SR	M3GP MR
M3GP	NA	Not	Equal	Equal	Equal
M3GP NE SR		NA	Not	Not	Not
M3GP NE MR			NA	Equal	Not
M3GP SR				NA	Equal
M3GP MR					NA

Table 9. Nemenyi statistical test for training on the MCD3 problem: ”Equal” means that the distributions of solutions are not significantly different (with an alpha of 0.05); ”Not” indicates that the distributions are significantly different; and ”Not Applicable” (NA)

Train MCD3	M3GP	M3GP NE SR	M3GP NE MR	M3GP SR	M3GP MR
M3GP	NA	Not	Equal	Equal	Equal
M3GP NE SR		NA	Not	Not	Not
M3GP NE MR			NA	Equal	Equal
M3GP SR				NA	Equal
M3GP MR					NA

Table 10. Nemenyi statistical test for testing on the MCD3 problem: ”Equal” means that the distributions of solutions are not significantly different (with an alpha of 0.05); ”Not” indicates that the distributions are significantly different; and ”Not Applicable” (NA)

Test MCD3	M3GP	M3GP NE SR	M3GP NE MR	M3GP SR	M3GP MR
M3GP	NA	Equal	Equal	Equal	Not
M3GP NE SR		NA	Equal	Equal	Not
M3GP NE MR			NA	Equal	Equal
M3GP SR				NA	Equal
M3GP MR					NA

Table 11. Nemenyi statistical test for training on the Tower problem: ”Equal” means that the distributions of solutions are not significantly different (with an alpha of 0.05); ”Not” indicates that the distributions are significantly different; and ”Not Applicable” (NA)

Train Tower	M3GP	M3GP NE SR	M3GP NE MR	M3GP SR	M3GP MR
M3GP	NA	Not	Not	Not	Not
M3GP NE SR		NA	Equal	Not	Not
M3GP NE MR			NA	Not	Not
M3GP SR				NA	Equal
M3GP MR					NA

Table 12. Nemenyi statistical test for testing on the Tower problem: ”Equal” means that the distributions of solutions are not significantly different (with an alpha of 0.05); ”Not” indicates that the distributions are significantly different; and ”Not Applicable” (NA)

Test Tower	M3GP	M3GP NE SR	M3GP NE MR	M3GP SR	M3GP MR
M3GP	NA	Not	Not	Equal	Equal
M3GP NE SR		NA	Equal	Not	Not
M3GP NE MR			NA	Not	Not
M3GP SR				NA	Equal
M3GP MR					NA

previous layer, causing the loss of the best individual between layers.

The benefits of this configuration allow the search for new features to restart, reducing solution complexity and resource usage—in other words, minimizing the bloating effect through lower elitism pressure. Overall fitness is lower, however, without improvement compared to classic M3GP.

For the classification problems the results present small improvement overall, but still significant by reaching higher results (MCD3 in training and Heart in testing) than running M3GP without evolutive layers. Another detail is the number of features is between 5 and 12, it might look small considering that 50 layers of building features were needed to reach that small number, in the end the hidden layers take many more feature to build upon to reach the last layer, but in the end any layer can be transformed and optimize the search for a better solution. The node count is significantly higher which impacts readability and increases the overall resources needed for improvement.

In the regression problems, training fitness of the best solution (maximum, median and minimum) shows a tendency of decreasing the error over higher layer bypassing the classic M3GP fitness (goal to beat), the testing result shows a continual tendency of improvement until reaching a point of

Table 13. Nemenyi statistical test for training on the Yacht problem: ”Equal” means that the distributions of solutions are not significantly different (with an alpha of 0.05); ”Not” indicates that the distributions are significantly different; and ”Not Applicable” (NA)

Train Yacht	M3GP	M3GP NE SR	M3GP NE MR	M3GP SR	M3GP MR
M3GP	NA	Not	Not	Not	Not
M3GP NE SR		NA	Equal	Not	Not
M3GP NE MR			NA	Not	Not
M3GP SR				NA	Equal
M3GP MR					NA

Table 14. Nemenyi statistical test for testing on the Yacht problem: ”Equal” means that the distributions of solutions are not significantly different (with an alpha of 0.05); ”Not” indicates that the distributions are significantly different; and ”Not Applicable” (NA)

Test Yacht	M3GP	M3GP NE SR	M3GP NE MR	M3GP SR	M3GP MR
M3GP	NA	Equal	Equal	Not	Not
M3GP NE SR		NA	Equal	Not	Not
M3GP NE MR			NA	Not	Not
M3GP SR				NA	Equal
M3GP MR					NA

over fitting. Let’s remember that no elitist variant don’t present this effect, so a deeper layer provides improvement in regression along with over fitting.

Finally, the presented real world problems in this article are from different domains, M3GP is able to find a transformation to ease learning load on the classification and regression methods, making the domain not a factor for improvement in the evolutive layer for feature improvement search of more favorable space.

7.2 Practical Implications

Overall, one remark needs to be addressed for Tower: classic M3GP is unable to improve training and testing, and the same occurs with other state-of-the-art algorithms [18]. This remarkable improvement raises a question about problem difficulty; for example, the classification problems are almost solved by reaching accuracies between 95 and 99, pushing elitist variants towards overfitting when nearing 100 accuracy. Something to keep in mind is that each layer alone constitutes a solution, and we can utilize any of these layers individually before overfitting begins to appear.

If resources or computational time present a problem, the layer configuration can be predefined by the user, utilizing the previous version of M3GP, which is M2GP [6], where the number of features at the root node is defined at the beginning of the evolutionary process. In this way, predefined evolutionary layers can be utilized, limiting the complexity of the symbolic model.

Another recommendation is based on Nemenyi’s statistical analysis, which shows no significant difference between running M3GP with evolutionary layers using the same resources as classic M3GP and increasing the resources to perform more evaluations.

7.3 Future Work

For classification, a different method for class distance must be implemented, Mahalanobis works great with classic M3GP but fall short in improving the new variants, or combine different methods in each layer (as activation function in neuronal networks) to improve evolutive layer effect.

Another development will be feature analysis and tree pruning to reduce the incremental impact of feature size on deeper layers. Also utilizing different learning methods in predefined evolutive layers to find a synergy between methodologies.

Lastly the implantation of transfer learning between layers [16] to allow shortcuts in the learning process.

References

1. **Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., Farhan, L. (2021).** Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. Journal of Big Data, Vol. 8, No. 1, pp. 53. DOI: 10.1186/s40537-021-00444-8.
2. **Bengio, Y., Courville, A., Vincent, P. (2013).** Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 35, No. 8, pp. 1798–1828. DOI: 10.1109/TPAMI.2013.50.
3. **Bensusan, H., Kuscu, I. (1996).** Constructive induction using genetic programming. International Conference on Machine Learning.
4. **Gerritsma, J., Onnink, R., Versluis, A. (1981).** Geometry, resistance and stability of the delft systematic yacht hull series1. International Shipbuilding Progress, Vol. 28, No. 328, pp. 276–297. DOI: 10.3233/ISP-1981-2832801.
5. **Indolia, S., Goswami, A. K., Mishra, S., Asopa, P. (2018).** Conceptual understanding of convolutional neural network- a deep learning approach. Procedia Computer Science, Vol. 132, pp. 679–688. DOI: https://doi.org/10.1016/j.procs.2018.05.069. International Conference on Computational Intelligence and Data Science.
6. **Ingalalli, V., Silva, S., Castelli, M., Vanneschi, L. (2014).** A multi-dimensional genetic programming approach for multi-class classification problems. **Nicolau, M., Krawiec, K., Heywood, M. I., Castelli, M., García-Sánchez, P., Merelo, J. J., Rivas Santos, V. M., Sim, K.**, editors, Genetic Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 48–60.
7. **Janosi, S. W. P. M., Andras, Detrano, R. (1989).** Heart Disease. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C52P4X.
8. **Kira, K., Rendell, L. A. (1992).** The feature selection problem: traditional methods and a new algorithm. Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI Press, pp. 129–134.
9. **Kononenko, I., Šimec, E., Robnik-Šikonja, M. (1997).** Overcoming the myopia of inductive learning algorithms with relieff. Applied Intelligence, Vol. 7, No. 1, pp. 39–55. DOI: 10.1023/A:1008280620621.
10. **Koza, J. R. (1992).** Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge, MA, USA.
11. **Langdon, W. B., Poli, R. (1998).** Fitness causes bloat. **Chawdhry, P. K., Roy, R., Pant, R. K.**, editors, Soft Computing in Engineering Design and Manufacturing, Springer London, London, pp. 13–22.
12. **LeCun, Y., Bengio, Y., Hinton, G. (2015).** Deep learning. Nature, Vol. 521, No. 7553, pp. 436–444. DOI: 10.1038/nature14539.
13. **Lichman, M. (2013).** UCI machine learning repository.
14. **Luke, S., Panait, L. (2002).** Lexicographic parsimony pressure. Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 829–836.
15. **Matheus, C. J. (1989).** A constructive induction framework. In **Segre, A. M.**, editor, Proceedings of the Sixth International Workshop on Machine Learning. Morgan Kaufmann, San Francisco (CA), pp. 474–475. DOI: https://doi.org/10.1016/B978-1-55860-036-2.50121-1.
16. **Muñoz, L., Trujillo, L., Silva, S. (2020).** Transfer learning in constructive induction with genetic programming. Genetic Programming and Evolvable Machines, Vol. 21, No. 4, pp. 529–569. DOI: 10.1007/s10710-019-09368-y.

17. **Muñoz, L., Silva, S., Trujillo, L. (2015).** M3gp – multiclass classification with gp. **Machado, P., Heywood, M. I., McDermott, J., Castelli, M., García-Sánchez, P., Burelli, P., Risi, S., Sim, K.**, editors, Genetic Programming, Springer International Publishing, Cham, pp. 78–91.

18. **Muñoz, L., Trujillo, L., Silva, S., Castelli, M., Vanneschi, L. (2019).** Evolving multidimensional transformations for symbolic regression with m3gp. Memetic Computing, Vol. 11, No. 2, pp. 111–126. DOI: 10.1007/s12293-018-0274-5.

19. **Silva, S., Almeida, J. (2008).** Gplab-a genetic programming toolbox for matlab. Proceedings of the Nordic MATLAB Conference.

20. **Trujillo, L., Muñoz, L., Galván-López, E., Silva, S. (2016).** neat genetic programming: Controlling bloat naturally. Information Sciences, Vol. 333, pp. 21–43. DOI: https://doi.org/10.1016/j.ins.2015.11.010.

21. **Vladislavleva, E. J., Smits, G. F., den Hertog, D. (2009).** Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. IEEE Transactions on Evolutionary Computation, Vol. 13, No. 2, pp. 333–349. DOI: 10.1109/TEVC.2008.926486.

22. **Wnek, J., Michalski, R. S. (1994).** Hypothesis-driven constructive induction in aq17-hci: A method and experiments. Machine Learning, Vol. 14, No. 2, pp. 139–168. DOI: 10.1023/A:1022622132310.

23. **Xiang, S., Nie, F., Zhang, C. (2008).** Learning a mahalanobis distance metric for data clustering and classification. Pattern Recognition, Vol. 41, No. 12, pp. 3600–3612. DOI: https://doi.org/10.1016/j.patcog.2008.05.018.

Article received on 07/02/2025; accepted on 18/06/2025.
**Corresponding author is Luis Muñoz Delgado.*