Assiut Univ. J. of Mathematics and Computer Science *Printed ISSN 1687-4897*
48(1), pp58-71 (2019)

SOLVING TWO-CLASS CLASSIFICATION PROBLEM USING MEMETIC PROGRAMMING

Amal A. Farhat, I.E. El-Semman, Emad Mabrouk

Department of Mathematics, Faculty of Science, Assiut University, Assiut 71516, Egypt

Received: 10/12/2019 **Accepted:** 9/4/2019 **Available Online:** 7/7/2019

The main target of the two-class classification problem is to design a classifier that discriminates between two objects from a seen dataset, then use this classifier to predict the object's class for unseen instances. Different methods have been used to solve the two-class classification problem, such as Genetic algorithm (GA) and Genetic Programming (GP). However, there is still a need to design new methods that can overcome some limitations in evolutionary algorithms, e.g., the high disruption of the breading operations; mutation and crossover. Recently, the Memetic Programming (MP) algorithm was proposed as an improvement to the GP algorithm. In this paper, we adapt the MP algorithm to produce a new classifier algorithm called the Memetic Programming Classifier (MPC) algorithm to solve the two-class classification problem. The performance of MPC is validated through different datasets from the UCI database and the accuracy is compared along with different methods. As a result, the proposed MPC algorithm shows a competitive performance compared with 179 classifiers in the literature.

Keywords: *Classifier; Classification Problem, Genetic Programming; Local Search; Memetic Programming;*

1 INTRODUCTION

Machine learning (ML) is one of the most interesting branches of artificial intelligence [6, 11]. ML can be defined as the task of programming computers that is dependable of building a learning model from past experiences or from training datasets. Therefore, this model can be used to gain information from the given data, make predictions in the future, or both. Recently, several ML techniques enabled computers to outperform human-level execution at image classification [10], to teach mobile robots the visual perception in forest paths [8], to beat people in complex games [2, 19]. Moreover, several ML techniques used to give deep speech to text applications in popular mobile phones [9].

Indeed, one of the most important applications of ML appears in the classification problems.

Classification plays a significant role in real life applications, e.g. computational biology and text and sound categorization [1, 7]. In such problems, a set of patterns or examples are given, where each example has some features. These features represent the inputs and one label refers to the output class of this example. The desired classifier attempts to build a simple mathematical model that can identify the label for seen examples with higher accuracy. Then, this classifier can be used to identify the label of unseen examples. Genetic Programing (GP) is introduced as one of Artificial Intelligence techniques [12], which considered as a developed version of the well-known Genetic Algorithm (GA). Moreover, GA as an evolutionary algorithm is evolved based on Darwin's theory of survival of the fittest [17]. Utilization of GP in the field of classification received considerable attention in the recent years. Mabrouk et al. [16] produced a new algorithm called the Memetic Programming (MP) algorithm by extending the GP algorithm using a set of local search techniques to improve its performance. The aim of the present paper is to build a suitable classifier for two-class classification problem using a modified version of the MP algorithm.

The paper is organized out as follows: The proposed Memetic Programming Classifier (MPC) algorithm will be introduced in the next section. In Section 3, we report some results of the MPC algorithm for different benchmark problems. Finally, the conclusion will be presented in Section 4.

2. METHODS

2.1. Two-class classification problem definition

A classifier is a function F that assigns a class label y to a feature vector x [3]. If we have a training set $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, where $x_i \in \mathbb{X}^q \; and \; y_i \in \mathbb{B}^2$. In addition, $\mathbb{X}^q$ is a q-dimensional space that represents the feature vector and $\mathbb{B}^2$ is the two-dimensional binary space that represents the label vector of a given training set. Therefore,

the classifier F is defined as $F : \mathbb{X}^q \rightarrow N_2$, where $N2$ is the set of label vectors for two classes and defined as:

$$N_2 = \left\{ \boldsymbol{u} \in \mathbb{B}^2 : u_i \in \{0,1\}, \sum_{j=1}^{2} u_j \right\} \quad (1)$$

where $\boldsymbol{u}$ is a vector in $\mathbb{B}^2$, u_1 represents the first class, while u_2 represents the second class. The classification method finds the classifier F that minimize $|\boldsymbol{F}(\boldsymbol{x}) - \boldsymbol{y}|$. After building the classifier, the quality of this classifier can be emphasized using its result for the testing set.

2.2. Classifier design

The MP algorithm searches the proposed solution space to find the best solution. This solution represents the best individual found through the search process. Each individual in the MP algorithm is a tree in which external nodes are terminals and internal nodes are functions, see Figure (1a). The domains of terminals and functions are problem dependent. Mainly, the algorithm starts with a population of individuals generated randomly then iterates three steps many times, Figure (1b). In the first step, the algorithm selects a pool of parents according to their performance to generate new individuals using the diversification and intensification strategies. In the diversification step, the algorithm uses cross-over and mutation operators as in GP to guarantee the diversity in the new population. However, in the intensification step, a local search algorithm is used to intensify elite programs from the current population. The MP algorithm repeats these steps until reaching a predefined termination condition [16]. In this application, we represent the classifier as a large tree, T, with a specific structure, see Figure (1a). This tree is encoded in MPC as a code representation as in Figure (1a). After evaluating the value of the tree, we can use this value to create a corresponding rule as in Figure (1a). According to the resulting value of T, a pattern x can be classified into one of the available classes.

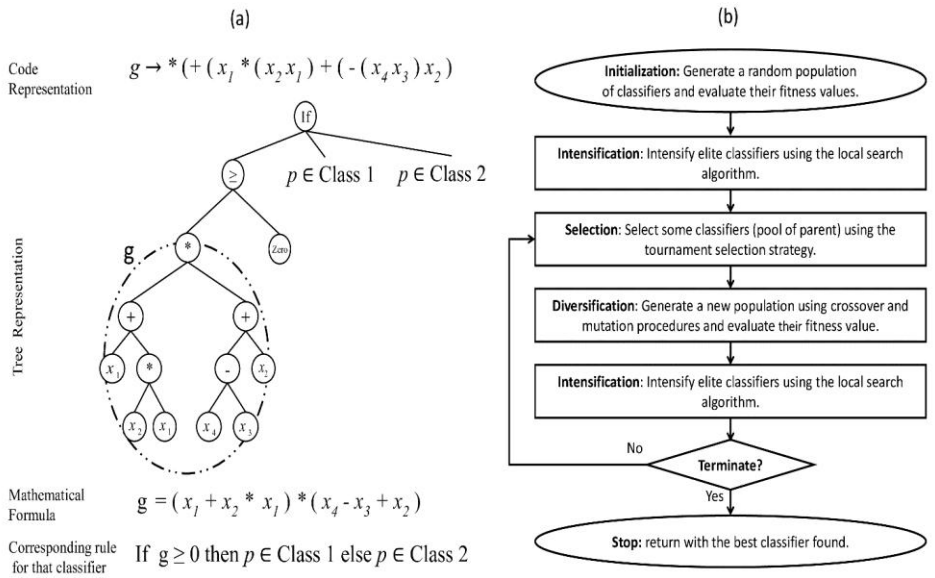

Figure 1: (a) A tree representation that illustrates how the code representation is converted to a two-class classification

During the search process, the MP evaluates the fitness value for each classifier using a predefined training dataset. Then, the algorithm performs the breading operations on these classifiers to improve their performance. Finally, the best classifier (with higher accuracy) found in all generations will be considered as the output of the algorithm. Therefore, the testing dataset can be used to compute the accuracy of the resulting classifier. The following steps summarize how to generate the best classifier.

Step 1: The tree structure

Each tree in the MPC algorithm consists of some internal nodes generated from the set of functions, and some leave nodes generated from the set of terminals. In this study, the function set FS contains a combination of arithmetic and Boolean functions to extract the desired

mathematical rules. Specifically, $FS = \{+, -, *, \%, sqrt, IF, AND, OR\}$ where $a \% b = 1$ if $b = 0$; otherwise, $a \% b = a / b$. AND and OR are Boolean functions of two parameters, where $AND(a, b)$ returns 1 if $a > 0$ and $b > 0$, otherwise, it returns zero. However, $OR(a; b)$ returns zero if $a < 0$ and $b < 0$, otherwise, it returns 1. Moreover, IF is a Boolean function of three arguments and $IF(a, b, c)$ returns b if $a > 0$, otherwise, it returns c. In addition, the terminal set $TS = \{feature\_variable, LC\}$, where the *feature_variables* is the list of all attributes of the given problem and LC is a list of numbers, e.g. $LC = \{1, 2, 3, 5, 7\}$.

Suppose that p is a pattern taken from a dataset with two classes with four attributes. Let $FS = \{+, -, *\}$ and $TS = \{a_1, a_2, a_3, a_4\}$ then Figure (1a) explains how the generated classifier classifies the pattern p.

Step 2: Fitness function

A set of samples $X_{tr} = \{x_1, x_2, \dots, x_N\}$ are used as the training dataset during the search process of the MPC algorithm to evolve the generated classifiers. The MPC algorithm estimates the performance of each classifier using the fitness function. The fitness function takes the classifier with the training dataset, then the value g of the classifier will be evaluated for each pattern (raw) in the training dataset. If $g > 0$ for a pattern p, then it will be classified as Class 1, otherwise p will be classified as Class 2. Finally, the fitness value of that classifier can be specified using the following equation:

$$fitness = \frac{no.of\ patterns\ classified\ correctly}{no.\ of\ patterns\ in\ the\ training\ dataset\ (N)}. \qquad (2)$$

Accordingly, a classifier with fitness value one will be considered as the optimal solution for the given problem since all patterns are classified correctly.

Step 3: Crossover and mutation

Crossover and mutation are essential operators in the evolution of the MPC algorithm. Both operators are applied on some classifiers chosen using the c-way tournament selection method, where c classifier is chosen randomly and the fittest one will be the winner [21]. The crossover operator is applied for two classifiers where the algorithm chooses two nodes randomly, one from each tree, and interchanges the two sub-trees rooted at these nodes to result new two classifiers. On the other hand, the mutation operator is applied for one classifier, selected using the tournament selection. Then, the new classifier is generated by replacing a sub-tree chosen randomly by a new one which created randomly.

Step 4: Local search procedures

Several local search procedures are used to generate new classifiers in a neighborhood of the current one. The shaking procedure is used to alter classifier nodes without changing its structure. On the other hand, expanding terminal nodes or cutting sub-trees of the original classifier are used to change the structure of the classifier using the grafting and pruning procedures. The local search procedures are applied at each generation on some promising classifiers to generate new classifiers in the neighborhood of the selected classifier. For more details about shaking, grafting and pruning procedures and their applications see [16].

Step 5: Termination of the MPC algorithm

There are two termination conditions to terminate the algorithm:

1. Finding a classifier that can classify all training samples correctly, i.e. its fitness value is 1.

2. Reaching to the maximum number of fitness evaluations.

In other words, if an individual correctly classifying all training samples then the algorithm will terminate and produce that individual as the output classifier. Otherwise, if the algorithm accesses the maximum

number of fitness evaluations, the output classifier will be the individual with the highest fitness value.

3. Numerical Experiments

To demonstrate the ability of the proposed algorithm in producing high-efficiency rules for classification problems, we applied the MPC algorithm for 4 benchmark datasets from the UCI repository [13]. Additionally, the MPC algorithm maximizes the fitness function in Equation 2, so it returns with the highest accuracy classifier. The next subsection shows the details and properties of all datasets under consideration. The proposed settings for the MPC algorithm are introduced in Subsection 3.2. In Subsection 3.3, the numerical results of the MPC algorithm are introduced along with different results for set of algorithms in the literatures.

3.1. Datasets

In this study, four different problems are used: The Hill-Valley problem and three different versions of the Monks problems. Table 1 exhibits properties of these datasets in terms of the number of patterns (No.cases), the number of attributes (No.attr.), attributes type (Attr.type) and the number of classes in each dataset (No.classes). More details of the datasets can be found in [13].

To prepare the datasets for being used by the MPC algorithm, some data preprocessing approaches are applied. Specifically, the binarization approach is used to enumerate the nominal attributes for the Monks datasets, while for the Hill-Valley dataset, the normalization approach is used to make all attributes having the same interval.

Table 1: Properties of the benchmark datasets

DataSets	No.cases	No.attr.	Attr.type	No.classes
Hill-Valley	606	100	continuous	2
Monk1	432	6	nominal	2

Monks2	432	6	nominal	2
Monks3	432	6	nominal	2

3.2. MPC settings

To run the MPC algorithm, according to the previous sections, the following set of parameters must be determined before calling the algorithm:

- nPop: Population size.
- nGnrs: Maximum number of generations.
- iDepth: Depth of trees in the initial population.
- mDepth: Maximum depth for each tree during the search process.
- nLsp: Number of trees using in the local search algorithm at each generation.
- nTrial: Number of trial trees produced from the current tree.
- nShaking: Number of nodes that will be changed in the shaking search.
- nBranch: Number of branches that will be changed in the grafting/pruning search.
- bDepth: Depth of branches that will be changed in the grafting/pruning search.
- nFail: Maximum number of failures (non-improvements) in the local search algorithm.

In evolutionary algorithms, a set of values for each parameter is tested through several independent runs of the algorithm. Therefore, the parameter value that produces the highest fitness values can be used as the best value for that parameter [4, 14, 18, 20]. The best values of the MPC parameters are shown in Table 2. These values are selected as shown in [15, 16] and based on a lot of pilot experiments of the proposed MPC algorithm.

Table 2: Common parameters for all datasets

Parameter	Value	Parameter	Value
nPop	1000	nTrial	3
nGnrs	1000	nShaking	2
iDepth	3	nBranch	2
mDepth	5	bDepth	1
nLsp	5%	nFail	1

3.3. MPC results

To assess the performance of the MPC algorithm, we used the parameter values in Table 2. Moreover, 1000 individuals are used as the population size and 1000 generations are used as the maximum number of iterations. For each dataset, the algorithm stops when the maximum fitness reaches 1, i.e. at least one individual can classify all patterns in the training dataset correctly.

Nevertheless, whenever the maximum fitness is less than 1, the algorithm continue the search process for 1000 iterations at most. The function set used for both problems is $FS = \{+, -, *, IF, AND, OR\}$ and the terminal set $TS = \{all\_feature\_variable, 1, 2, 3\}$.

During the experimental results of this paper, the 4-fold cross-validation technique is used to assess the ability and stability of the MPC algorithm to compare the results with Fernández-Delgado et al. [5]. The 4-fold cross-validation technique divides the given dataset into two complementary subsets, the training set to train the algorithm and the testing set used to validate the stability and evaluate the accuracy of the resulting classifier. In 4-fold cross-validation, the entire dataset is randomly split into 4 folds, with 3 folds are used as the training dataset, and the remaining fold is retained as the testing dataset. The classifier will be evolved using the training dataset and the accuracy will be

estimated using the testing dataset. This process is then repeated 4 times until each of the 4 folds is used exactly once as the testing dataset. The average of the resulting 4 recorded accuracies, called the cross-validation accuracies, will be considered as the accuracy of the resulting classifier.

Fernández-Delgado et al. [5] studied 179 classifiers on various types of datasets and they concluded that the parRF_t, rf_t and svm_C algorithms are the best classifiers for these datasets. We performed 50 independent runs and the best rule found was considered. In Table 3, we show results of MPC with results of the three classifiers parRF_t, rf_t, svm_C and the maximum accuracies (Max) founded through 179 classifier, see [5].

Table 3: Results of the MPC algorithm and different algorithms in the literature.

Problem	MPC	parRF_t	rf_t	svm_C	Max
Hill-Valley	75.5776	55.3	54.1	53.6	74.3
Monk1	91.6667	61.1	61.1	51.9	79.9
Monk2	81.0185	65.7	65.7	65.3	67.8
Monk3	88.8889	53.7	53.7	53.2	77.3

From the obtained results of Table 3, we can conclude that our classifier rules achieved the highest accuracy for all problem under consideration. Moreover, there is 5% misclassification in the Monk3 problem due to the noise in the training dataset. After 50 independent runs on each one of the Monks problems, the following three rules are the best rules founded by the MPC algorithm in terms of the fitness value and the accuracy.

Monk1 Problem:

$$IF(AND(OR(A_{11}, A_{13}), A_{22}) - A_{51} + AND(A_{11}, A_{22}) * A_{53} - \left(A_{22} + (A_{23} * A_{13})\right) \geq 0 \ \ then\ Class\ 1$$

$$Else\ Class\ 2$$

Monk2 Problem:

$$IF\big(IF\big(OR(A_{31},A_{41}),(A_{31}*A_{41}),(A_{31}-A_{51})\big)+(A_{51}-A_{62})*OR(A_{41},A_{31})+A_{11} - \big(OR(OR(A_{51},A_{31}),A_{41})\big)+A_{21}\big)\geq 0\ then\ Class\ 1$$

$$Else\ Class\ 2$$

Monk3 Problem:

$$IF\left((A_{23}-A_{51})-\big(AND\big((A_{22}*A_{53}),(3-A_{51})\big)+A_{51}\big)\right)\geq 0\ then\ Class\ 1$$

$$Else\ Class\ 2$$

4. CONCLUSION AND FUTURE WORK

In this paper, we introduced a new evolutionary algorithm for the classification problems. The proposed MPC can be used to classify the 2□class classification problems efficiently. Moreover, the proposed algorithm succeeded in providing only one rule to classify patterns of the given problem without overlaps.

We compared the resulting classifiers generated by the MPC algorithm with 179 other classifier published by Fernández-Delgado et al. [5] and our algorithm outperformed all of these algorithms. Based on this comparison, we may conclude that the MPC algorithm shows a great achievement for considered datasets.

An area of future work is to apply the proposed algorithm for more datasets in the literature and applied problems. Various modifications, experiments and parameter settings have been left for the future due to lack of time, since the experiments with large datasets take time and require days to finish a single run of the algorithm.

Another area of future work is to introduce a modified version of the MPC algorithm to solve the c-class classification problems, where c >= 2. Further, we aim to produce a parallel version of the MPC algorithm and use the cloud computing machine, which will reflect significant improvements to the results.

REFERENCES

[1] Angermueller, C., P¨arnamaa, T., Parts, L., Stegle, O., 2016. Deep learning for computational biology. Molecular Systems Biology 12 (7), 878.

[2] Baier, H., Winands, M., 2011. Active opening book application for monte-carlo tree search in 19_ 19 go. In: Benelux Conference on Artificial Intelligence. pp. 3–10.

[3] Chaudhari, N. S., Purohit, A., Tiwari, A., 2009. Genetic programming for classification. International Journal of Computer and Electronics Engineering, IJCEE 1, 69–76.

[4] Eiben, A. E., Smit, S. K., 2011. Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm and Evolutionary Computation 1 (1), 19–31.

[5] Fern´andez-Delgado, M., Cernadas, E., Barro, S., Amorim, D., 2014. Do we need hundreds of classifiers to solve real world classification problems. J. Mach. Learn. Res 15 (1), 3133–3181.

[6] Ghahramani, Z., 2015. Probabilistic machine learning and artificial intelligence. Nature 521 (7553), 452.

[7] Gibaja, E., Ventura, S., 2015. A tutorial on multilabel learning. ACM Computing Surveys (CSUR) 47 (3), 52.

[8] Giusti, A., Guzzi, J., Cires¸an, D. C., He, F.-L., Rodr´ıguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Di Caro, G., 2016. A machine learning approach to visual perception of forest trails for mobile robots. IEEE Robotics and Automation Letters 1 (2), 661–667.

[9] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., Others, 2014. Deep speech: Scaling up end-to-end speech recognition. arXiv preprint

arXiv:1412.5567.

[10] He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet

classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026-1034.

[11] Kononenko, I., 2001. Machine learning for medical diagnosis: history, state of the art and perspective. Artificial Intelligence in medicine 23 (1), 89–109.

[12] Koza, J. R., 1992. Genetic programming: on the programming of computers by means of natural selection. Vol. 1. MIT press.

[13] Lichman, M., 2013. UCI machine learning repository. URL http://archive.ics.uci.edu/ml

[14] Lobo, F., Lima, C. F., Michalewicz, Z., 2007. Parameter setting in evolutionary algorithms. Vol. 54. Springer Science & Business Media.

[15] Mabrouk, E., Hedar, A., Fukushima, M., 2010. Memetic programming algorithm with automatically defined functions. Tech. rep., Technical Report 2010-015, Department of Applied Mathematics and Physics, Kyoto University, Japan.

[16] Mabrouk, E., Hedar, A.-R., Fukushima, M., 2008. Memetic programming with adaptive local search using tree data structures. In: Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology. ACM, pp. 258–264.

[17] Morse, H., 2018. Where Do We Come From? Is Darwin Correct?: A Philosophical and Critical Study of Darwin's Theory of Natural Selection. Routledge.

[18] Rojas, I., Gonz´alez, J., Pomares, H., Merelo, J., Castillo, P., Romero, G., 2002. Statistical analysis of the main parameters involved in the design of a genetic algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 32 (1), 31–37.

[19] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., 2016. Mastering the game of go with deep neural networks and tree search. nature 529 (7587), 484–489.

[20] Smit, S. K., Eiben, A. E., 2009. Comparing parameter tuning methods for evolutionary algorithms. In: 2009 IEEE congress on evolutionary computation. IEEE, pp. 399–406.

[21] Xie, H., 2009. An analysis of selection in genetic programming. PhD thesis, Victoria University of Wellington, New Zealand.

حل مشكلة التصنيف ثنائية الفئه بإستخدام البرمجة الجينية المتطورة

الهدف الرئيسي لحل مشاكل التصنيف ذي الفئتين هو تصميم مصنف يستطيع أن يميز بين الفئتين من خلال مجموعة أمثلة مخصصة للتدريب واستخدام المصنف الناتج للتنبؤ بفئة أمثلة غير مرئية. يوجد العديد من الطرق التي استخدمت لحل مشاكل التصنيف ذي الفئتين مثل البرمجة الجينية (GP) . ولكن تظل الحاجة لتصميم طرق جديدة تستطيع التغلب على بعض القيود في الخوارزميات التطورية مثل الاضطراب العالي لعمليات الفرز، الطفرة و التزاوج. خوارزمية البرمجة الجينية المتطورة (GP) . في هذه اقترحت مؤخرا كتحسين لخوارزمية (MP)الورقة البحثية، قمنا بتمديد خوارزمية MP لإنتاج خوارزمية جديدة تسمى خوارزمية مصنف البرمجة الجينية المتطورة (MPC)خوارزمية لحل مشاكل التصنيف ذي الفئتين. تم التحقق من أداء MPC من خلال مجموعة بيانات مختلفة من قاعدة UCIبيانات تظهر أداء MPC وتمت مقارنة الدقة بطرق مختلفة. وجدنا أن تنافسي مقارنة بـ ١٧٩ مصنف في الأدبيات.