Evolving Multi-label Classification Rules with Gene Expression Programming: A Preliminary Study

José Luis Ávila-Jiménez, Eva Gibaja, and Sebastián Ventura

Department of Computer Sciences and Numerical Analysis, University of Córdoba

Abstract. The present work expounds a preliminary work of a genetic programming algorithm to deal with multi-label classification problems. The algorithm uses Gene Expression Programming and codifies a classification rule into each individual. A niching technique assures diversity in the population. The final classifier is made up by a set of rules for each label that determines if a pattern belongs or not to the label. The proposal have been tested over several domains and compared with other multi-label algorithms and the results shows that it is specially suitable to handle with nominal data sets.

1 Introduction

Most of classification problems associate only one label per pattern, l_i, from a set of disjoint labels, L. Nevertheless, this is not the only possible scenario, because there are many problems of increasing actuality, like text and sound categorization, protein and gene classification or semantic scene classification, where a pattern can have associated not just one, but a set of class labels, $Y \subseteq L$. To deal with this kind of situation, an automatic learning paradigm called multi-label classification has emerged, which allows the *one label per pattern* restriction to be avoided. Thus, the main characteristic of multi-label classification is that labels are not mutually excluding, allowing patterns with more than one associated label.

Multi-label classification problems have been basically dealt with from two points of view [1]. On one hand some studies describe several *transformation methods*, also called preprocessing techniques, which transform an original multi-label problem into several single label problems allowing the use of a classical classification algorithm.

On the other hand, many *algorithm adaptation techniques* have been developed in order to adapt a classical algorithm to directly work with multi-label data. For instance, some SVM approaches have been developed [2]. With reference to other techniques, in [3] an adaptation of the K-nearest neighbor algorithm, called *ML-KNN* has been proposed. The use of syntactic trees for hierarchical multi-label classification has been studied in [4] and an adaptation of the C4.5 algorithm to multi-label classification is presented in [5]. In spite

E.S. Corchado Rodriguez et al. (Eds.): HAIS 2010, Part II, LNAI 6077, pp. 9–16, 2010.

of their wide use in classical classification, bio-inspired approaches have rarely been applied to solve multi-label problems. It can be highlighted the *MULAM* algorithm based on ant colonies [6] and the work of Vallim et al. [7] where a genetic algorithm is proposed. In addition the authors have developed a genetic programming model that uses discriminant functions to learn multilabel classifiers [8].

However, there are few multilabel approach that allows to build understandable models the user can interpret as useful knowledge. Thats why this research proposes a multi-label classification algorithm designed for finding rules, which are more useful in certain domains. The proposed technique is based in GEP [9], a genetic programming paradigm successfully applied in classification problems. The global results point out that our approach is able to obtain results that are better or comparable to other classical multi-label approaches.

The paper is organized as follows: first we describe our proposal, the experiments carried out and the metrics used to measure the performance of the multi-label algorithms studied. Finally, the results are shown with conclusions about our study and future research lines.

2 GC-ML Algorithm

This section describes the most relevant features of the proposed algorithm, called *GC-ML* (*GEPCLASS Multi-label*). It is a Genetic Programming algorithm that maintains a population of individuals each of them will potentially be part of a multilabel classifier. The population will evolve to better classifier by the application of genetic operators like selection, mutation and crossover.

Firstly, a description of the individual representation and evaluation is showed and after that, the overall operation of the algorithm will be described.

2.1 Individual Representation

The proposed algorithm tries to learn a set of rules for each label in the problem in order to build a multi-label classifier. Each rule will be in the form *if* $\boldsymbol{A}$ *then* $\boldsymbol{L}$ where A is the clause that must be satisfied by the pattern to be associated with a label L, and it is composed by a set of conditions joined by logical operators (*and*, *or* or *not*). Each condition is a combination of an attribute and a constant value joint by a relational operator ($=$, $\neq$, $>$ or $<$).

In our algorithm, as in GEP algorithms, individuals have a dual encoding, that is, they have both genotype and phenotype. Genotype is a lineal string that consists of several genes, whose number and length is fixed and predetermined for each problem. Genes are linked by a connection function which is a parameter of the algorithm. Each gene is divided into two sections, *head* and *tail*. The former contains terminal and non-terminal elements, whereas the latter can just contain terminal elements. Non-terminal elements are functions but terminal elements are constants and the input elements associated with patterns's attributes.

The head size is a parameter of the algorithm while the tail size is calculated with the following expression: $t = int((h \times (n-1)+1)/2)$, where $int()$ returns the integer part of the argument and n is the largest arity found in the function set. Expression trees (ETs) are the phenotypical representation of the chromosome. Selection will operate over ETs while reproduction will operate over the chromosomes.

In order to use rules some modifications have been made to the original GEP individual representation, that are similar to Weiner and Lopes proposal [10]. The main modification is related to the non-terminal set, which has been divided in two groups, the logical function set and the relational function set. Logical functions can have as offspring in the expression tree other logical or relational function but not a terminal element. However, a relational function only can have as offspring an attribute and a constant value, both terminal elements. This modification is necessary to distinguish between them at phenotypical level.

A second modification in the structure of the genes has been made to assure that each relational function has just one constant and one attribute as offspring. Thus, the terminal set will contain a collection of *constant/value* pairs.

In Figure 1 it is shown an individual with two genes of different length, the expression tree that generates as phenotype (considering AND as connection function) and the rule that it represents. The consequent of the rule is not stored neither in the genotype or in the phenotype, but it is assigned during the evaluation phase as it will be showed below.

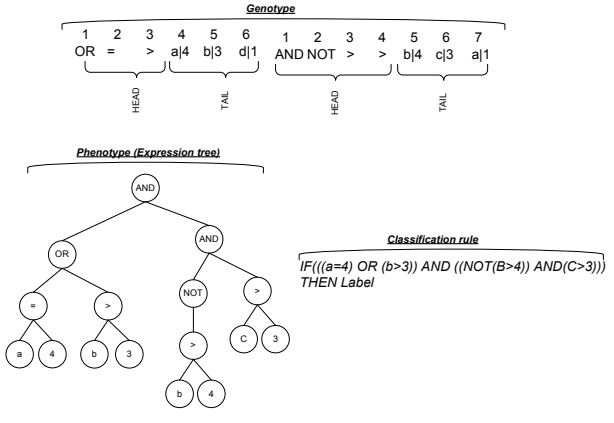

Fig. 1. Example of GC-ML individual

The final classifier will be composed by a set of rules for each class. An so, whenever any of these rules determines that the pattern belongs to the class, the classifier will consider the patter belonging to the class regardless of the results yielded by the rest.

2.2 Individual Evaluation

GC-ML has a population of individuals with the above mentioned features. Due to the multi-label nature of the problem, during the evaluation stage each of these individuals is evaluated for each label in the training set by means of a *fitness* function. Thus, instead an unique fitness value, a fitness vector is stored for each individual in order to store the fitness value for each label. It is necessary to point out that during the application of the selection operators, only the greatest fitness value in the vector will be considered.

The fitness function used is the harmonic mean between precision and recall, also known as the *F-score*. Precision for a label is the number of items correctly labeled as belonging to the class divided by the total number of patterns belonging to the label, and recall is defined as the number of items correctly labeled divided by the total number of patterns that the classifier has considered belonging to the label.

The expressions of precision, recall and F-Score are the following:

$$precision = \frac{tp}{tp + fn} \tag{1}$$

$$recall = \frac{tp}{tp + fp} \tag{2}$$

$$F - score = \frac{2 \times precision \times recall}{precision + recall} \tag{3}$$

2.3 Evolutionary Algorithm

The algorithm developed is similar to that proposed with GEP[9], but it has to calculate n fitness values for each individual, and it also must do the multi-label token competition to correct the fitness of individuals after evaluation.

The algorithm starts generating the initial population,and, while the algorithm does not peak the maximum number of generations, the following actions are performed for each of them:

1. For each label present in the problem, all individuals are evaluated, and the fitness array is reached.
2. After the evaluation of the individuals, the algorithm applies the *Token Competition* [11] technique to correct the fitness values calculated. This approach has been widely used in other genetic algorithms applied to classification problems [12].
   GC-ML carries out a token competition for each label where a token is played for each positive pattern associated with the label. This token is won by the individual with the highest fitness that correctly classifies it. After token distribution, algorithm proceeds to correct the label fitness of each individual using the following expression:

$$new\_fitness = \frac{original\_fitness \times tokens\_won}{total\_tokens} \tag{4}$$

3. After the token competition the best individuals are selected to constitute the next generation. To make this selection the best fitness of each individual is used. Moreover the genetic operators are applied over the population.

When the algorithm finishes, it is easy to find the individuals that must be in the learned classifier. Only those individuals that have won some tokens are relevant for the classifier and the rest can be rejected.

3 Experimental Section

The implementation of the algorithm was carried out using the *JCLEC* library [13]. JCLEC is a framework to develop evolutionary computation applications implemented in Java. JCLEC provides a set of generic classes that abstract the main features presented in evolutionary algorithms.

The objective of experiments is to determine the performance of the proposed algorithm and then to compare it with other multi-label proposals, both transformation methods and pure multi-label methods. Our algorithm has been compared to three other methods namely, Binary Relevance (BR), Label Powerset (LP) and the ML-KNN method. Both BR and LP are problem transformation methods[14] while ML-KNN is an implementation of the k-nearest neighbor method specifically designed for multi-label data.

The configuration parameters of the algorithm have been obtained testing it previously to the main experiments. A population of 5000 individuals have been used, each of them with 6 genes with a head length of 35. The maximum number of generations is 60 and the crossover probability is 0.8. In addition, the mutation and transposition probability is 0.2. The selection method is tournament of 2 individuals size.

The four data sets used to perform the experiments have been *scene* [15], *yeast* [16], *genbase* [17] and *medical* [18]. These data sets belong to a wide variety of application domains and they have been used in other multi-label studies covering several paradigms. Table 1 shows the main characteristics of these data sets, including label cardinality that is the average number of labels per example, and label density, which is the same number divided by the number of labels in the problem, $|L|$.

Table 1. Characteristics of datasets

Dataset	#Patterns	#Labels	Cardinality	Density
Scene	2407	6	1.061	0.176
Genbase	662	27	1.252	0.046
Yeast	2417	14	4.228	0.302
Medical	978	45	1.245	0.028

4 Results and Discussion

The measures of accuracy, precision and recall have been used in order to compare GC-ML with BR, LP and ML-KNN methods. Accuracy is the percent of correct answers of the classifier divided by the total answers (Equation 5) and precision and recall are the same measures used to calculate fitness function, previously showed in Equations 1 and 2. A 10 fold cross validation has been made for each data set and algorithm. Table 2 shows the average metrics.

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{5}$$

```
IF

(PARAM_48 > 0.5)
 OR
(((PARAM_19 = 0.4) AND (PARAM_71 != 0.1)) OR (PARAM_13 >0.0))
OR
((NOT (PARAM_60 <0.5) OR (PARAM_29 = 0.3)) OR (PARAM_25 != 0.6))
OR
((PARAM_16 >0.8) OR ((PARAM_36 = 0.1) AND (PARAM_11 = 0.4) ) OR (PARAM_46 != -6.0))
OR
((NOT (PARAM_54 >0.3) OR (PARAM_11 = 0.2)) AND PARAM_14 = 1.6))
OR
((PARAM_42 >3.1) AND ((PARAM_16 = 0.3) OR (PARAM_11 = 0.4) ) OR (PARAM_12 < -1.0))

THEN

LABEL_1
```

Fig. 2. Example of a discovered rule

Table 2. Experimental results

Algorithm	**Bin. Rel.**	**Label Pow.**	**ML-Knn**	***GC-ML***
Dataset	Accuracy values			
Scene	0.434	0.577	0.629	0.571
Genbase	0.273	0.684	0.638	0.778
Yeast	0.421	0.398	0.492	0.432
Medical	0.592	0.617	0.560	0.649
	Precision values			
Scene	0.443	0.602	0.660	0.554
Genbase	0.276	0.694	0.674	0.753
Yeast	0.527	0.528	0.732	0.672
Medical	0.651	0.678	0.573	0.702
	Recall values			
Scene	0.815	0.591	0.678	0.690
Genbase	0.273	0.654	0.638	0.683
Yeast	0.619	0.528	0.549	0.572
Medical	0.619	0.650	0.568	0.699

The proposed algorithm obtains better results, in general, than the other algorithms for each data set and measure. These results are comparable with those obtained by the other algorithms. Nevertheless, our algorithm obtains better results with categorical datasets. For example, if we see the results of scene and yeast, which are numeric data sets, GC-ML has the best results in accuracy, but it is similar to those obtained by the rest. However, with respect to the medical and genbase results, the proposed algorithm obtains better a accuracy. This behavior can be observed with the other metrics and it is reasonable because classification rules are more suitable to deal with nominal attributes than with numerical ones. Despite the other results, it is worth noting that, regardless of the kind of data set, our algorithm obtains always better values than the transformation methods (LP and BR) for every measure and dataset.

5 Conclusions

The present work shows the GC-ML algorithm, an algorithm for multi-label classification. This is an evolutionary proposal, based on GEP, where individuals encode classification rules to determine whether a pattern belongs or not to a particular label in a multi-label context. The final classifier is built as the combination of multiple rules present in the population. The algorithm implements a token competition technique specifically designed to deal with multi-label patterns whose aim is to ensure that there are individuals in the population associated with all the classes present in the problem.

Studies developed to verify the performance of the GC-ML algorithm with respect to other alternatives indicate that it gets better results than problem transformation proposals like BR and LP. Besides, it obtains similar results, or better in most of cases, than the multi-label implementation of the KNN algorithm. The experiments also show that the proposed algorithm is well indicated to be used with nominal data sets.

Regarding to future research, the algorithm is being tested with other datasets and also compared with other approaches for multi-label classification.

Acknowledges

This work has been financed in part by the TIN2008-06681-C06-03 project of the Spanish Inter-Ministerial Commission of Science and Technology (CICYT), the P08-TIC-3720 project of the Andalusian Science and Technology Department and FEDER funds.

References

1. Tsoumakas, G., Katakis, I., Vlahavas, I.: A review of multi-label classification methods. In: Proceedings of the 2nd ADBIS Workshop on Data Mining and Knowledge Discovery (ADMKD 2006), pp. 99–109 (2006)

2. Wan, S.P., Xu, J.H.: A multi-label classification algorithm based on triple class support vector machine. In: Proc. 2007 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR 2007), Beijing, China (November 2007)
3. Zhang, M.-L., Zhou, Z.-H.: A k-nearest neighbor based algorithm for multi-label classification, vol. 2, pp. 718–721. The IEEE Computational Intelligence Society (2005)
4. Blockeel, H., Schietgat, L., Struyf, J., Dzeroski, S., Clare, A.: Decision trees for hierarchical multilabel classification: A case study in functional genomics. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI, LNB), vol. 4213, pp. 18–29. Springer, Heidelberg (2006)
5. Clare, A., King, R.D.: Knowledge discovery in multi-label phenotype data. In: Siebes, A., De Raedt, L. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, p. 42. Springer, Heidelberg (2001)
6. Chan, A., Freitas, A.A.: A new ant colony algorithm for multi-label classification with applications in bioinfomatics. In: GECCO 2006: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 27–34. ACM Press, New York (2006)
7. A new approach for multi-label classification based on default hierarchies and organizational learning (2008)
8. Ávila, J.L., Galindo, E.L.G., Zafra, A., Ventura, S.: A niching algorithm to learn discriminant functions with multi-label patterns. In: Corchado, E., Yin, H. (eds.) IDEAL 2009. LNCS, vol. 5788, pp. 570–577. Springer, Heidelberg (2009)
9. Ferreira, C.: Gene expression programming: a new adaptative algorithm for solving problems. Complex Systems 13(2), 87–129 (2001)
10. Weinert, W.R., Lopes, H.S.: GEPCLASS: A classification rule discovery tool using gene expression programming. In: Li, X., Zaïane, O.R., Li, Z.-h. (eds.) ADMA 2006. LNCS (LNAI), vol. 4093, pp. 871–880. Springer, Heidelberg (2006)
11. Wong, M.L., Leung, K.S.: Data Mining Using Grammar-Based Genetic Programming and Applications. Kluwer Academic Publishers, Norwell (2000)
12. Lu, W., Traore, I.: Detecting new forms of network intrusion using genetic programming. In: The 2003 Congress on Evolutionary Computation, CEC 2003, vol. 3, pp. 2165–2172 (2003)
13. Ventura, S., Romero, C., Zafra, A., Delgado, J.A., Hervás, C.: JCLEC: A Java framework for evolutionary computation. Soft Computing 12(4), 381–392 (2008)
14. Tsoumakas, G., Vlahavas, I.: Random k-labelsets: An ensemble method for multilabel classification. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI, LNB), vol. 4701, pp. 406–417. Springer, Heidelberg (2007)
15. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classification. Pattern Recognition 37(9), 1757–1771 (2004)
16. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Advances in Neural Information Processing Systems, vol. 14 (2001)
17. Diplaris, S., Tsoumakas, G., Mitkas, P., Vlahavas, I.: Protein classification with multiple algorithms. In: Advances in Informatics, pp. 448–456 (2005)
18. Diesner, J., Frantz, T.L., Carley, K.M.: Communication networks from the enron email corpus it's always about the people. enron is no different. Comput. Math. Organ. Theory 11(3), 201–228 (2005)