Multi-label Classification with Gene Expression Programming

J.L. Ávila, E.L. Gibaja, and S. Ventura

Department of Computer Science and Numerical Analysis, University of Córdoba

Abstract. In this paper, we introduce a Gene Expression Programming algorithm for multi label classification. This algorithm encodes each individual into a discriminant function that shows whether a pattern belongs to a given class or not. The algorithm also applies a niching technique to guarantee that the population includes functions for each existing class. In order to evaluate the quality of our algorithm, its performance is compared to that of four recently published algorithms. The results show that our proposal is the best in terms of accuracy, precision and recall.

1 Introduction

Classification is one of the most studied tasks in the machine learning and data mining fields. This task basically consists of finding a function which is able to identify the set of an object's attributes (predictive variables) with a label or class identification (categorical variable). In the simplest case, each learning example has only one associated label, l_i, of a set of labels, L, which has been previously defined. This label l_i defines a set of patterns that do not share any element with subsets which are defined by the other labels l_j ($\forall j \neq i$). Nevertheless, this is not the only possible hypothesis, because numerous problems can be found where a given pattern can be simultaneously mapped to more than one class label. Typical examples are semantic scene classification [1], text and sound categorization [2,3], protein and gene classification [4] or medical diagnosis [5]. All these problems, which involve assigning all possible proper labels to a given example from a set of prediction variables, are multi-label classification problems [6].

The problem of multi-label classification has been tackled from numerous points of view. On the one hand, some papers describe a pre-processing of the data set which transforms a multi-label problem into one with a single-label which admits the application of supervised learning algorithms [1,7]. On the other hand, a specifically designed approach for multi-label data can be carried out [8,9]. With regard to the techniques that have been used, it is worth highlighting decision trees [8,10], Bayesian classifiers [11], artificial neural networks [12,13] and support vector machines [1,3,4]. Techniques of lazy learning have also been used, particularly a multi-label version of the well-known K-nearest neighbor algorithm called ML-KNN [9], and associative classification methods [14].

Finally, it is worthwhile mentioning the emerging interest in applying ensemble methods to multi-label classification in order to improve predictions [7,15,16].

As has been shown, there is a great variety of approaches to solve this kind of problems. Nevertheless, it seems that multi-label evolutionary approaches have not been applied, despite that the fact they have solved successfully numerous problems in traditional classification. Therefore, the goal of this paper has been the application and the analysis of this type of algorithms in multi-label problems. We have focused specifically on the Gene Expression Programming [17], a paradigm that has been successfully applied to other classification problems [18]. The algorithm developed, called GEP-MLC, encodes a discriminant function in each individual and uses a niching algorithm to guarantee diversity in the solutions. As we will see later, its results are quite satisfactory in terms of accuracy, precision and recall, improving results obtained by other recent algorithms.

The paper is organized as follows. The next section introduces the proposed algorithm. Then, the experiments carried out will be described, as will the results of the experiments along a set of conclusions and proposals for future research.

2 Algorithm Description

In this section we specify different aspects which have been taken into account in the design of the GEP-MLC algorithm, such as individual representation, genetic operators, fitness function and evolutionary process.

2.1 Individual Representation

As mentioned above, the GEP-MLC learns discriminant functions. A discriminant function is a function which is applied to the input features of a pattern (predictive variables) and produces a numerical value associated with the class that the pattern belongs to. To establish this correspondence, a set of thresholds are defined, and intervals of values in the output space are mapped to classification labels. The simplest example is that of the binary classifier, where only one threshold is defined (usually zero). Values to the right of this threshold are associated with patterns belonging to the class, while values to the left will be associated with non-membership in the class.

$$if(f(\mathbf{X}) > 0)\ then\ \mathbf{X} \in class\ else\ \mathbf{X} \notin class \tag{1}$$

In the case of multi-class problems (the number of classes $N > 2$), there are two approaches to tackle the problem. On the one hand, $N - 1$ thresholds and N intervals can be defined. On the other hand, $N - 1$ functions with only one threshold can be used and deal with the membership of an individual class as a binary classification problem. This last approach is the one that has been used in this study. So, each individual codes in its genotype the mathematical expression corresponding to a discriminant function (binary classifier), and the threshold value of zero has been assigned for all cases. As will be shown later, the class

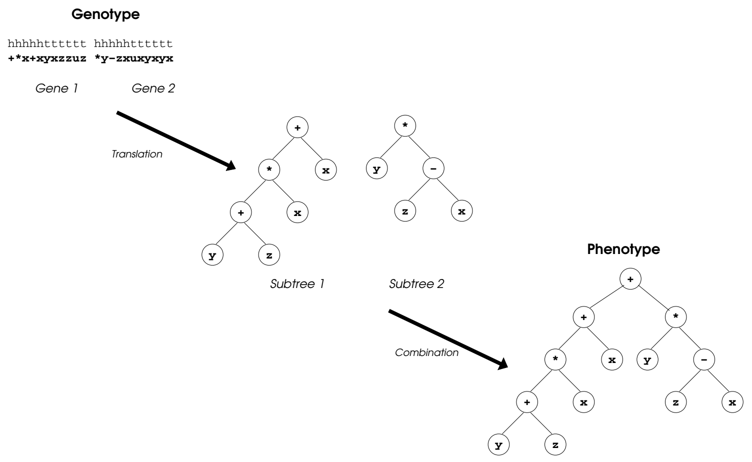

Fig. 1. Conversion from genes to subtrees

associated to each discriminant function is assigned during the evaluation of the individual.

Regarding individual representation, it must be said that in GEP-MLC, 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. Each gene is divided into two sections, *head* and *tail*. The first one contains terminal and non-terminal elements, whereas the second one can just contain terminal elements[1]. Head length is selected *a priori*, but tail size is calculated as

$$t = h(n-1)+1 \tag{2}$$

where t is tail size, h is head length and n the maximum arity (number of arguments) in the non-terminal set.

Phenotype is an expression tree obtained by a mapping process in which (a) each gene is converted into an expression subtree and (b) subtrees are combined by means of a connection function (in our case, the summation operator -see Figure 1 for an example).

2.2 Genetic Operators

GEP-MLC uses all the genetic operators defined in the standard GEP algorithm, that is, crossover, transposition and mutation. In the following their main features will be explained briefly.

Crossover Operators. choose two random parents and swap parts of their genotypes to generate two valid new offspring. GEP has three kinds of crossover

[1] In this context, terminal elements are functions without arguments, and non-terminal ones are functions with one or more arguments.

operators: (a) *One-point recombination* takes two parent chromosomes and splits them at exactly the same point. Afterwards the material downstream of the recombination point is exchanged between the two chromosomes. (b) *Two-point recombination* takes two points at random, and split chromosomes by them. The material between the recombination points is then exchanged between the two chromosomes, forming two new children chromosomes. (c) *Gene recombination* exchanges entire genes between two parent chromosomes, forming two children chromosomes containing genes from both parents.

Mutation Operators. perform a random change in a randomly selected element in the genotype. However, the operator must preserve the internal structure of the genotype and consequently, an element from the gene tail must be exchanged for an element from the terminal set, but an element from the gene head can be swapped either for an element from the terminal set or the non-terminal one.

Transposition Operators. are special mutation operators that select a substring from the genotype and transfer it to another position. GEP paradigm defines three transposition operators: (a)*IS Transposition* randomly selects one substring also with random length, this is inserted in any position in the gene head, other than the first element(the tree root). (b)*RIS Transposition* randomly chooses one fragment in the same way as IS transposition, but the element chosen should start with a function. This fragment is inserted into the first element of the gene head (the tree root). (c) *Gene Transposition* changes the position of two genes in individuals with more than one gene. This transposition randomly chooses two genes that swap their position in the genotype.

2.3 Individual Evaluation

During the evaluation phase, the discriminant function is mapped by a given individual, and its quality as a classifier is calculated for each class defined in the problem. The fitness function used is the F score, that is, the harmonic mean of precision and recall [19]:

$$fitness = \frac{2 \times precision \times recall}{precision + recall} \tag{3}$$

In contrast with single-label algorithms, that assign the label that produces the highest fitness value, the GEP-MLC algorithm calculates a fitness value for each label, storing N raw fitness values (one per label). As will be shown, the fitness value used in the selection phase is obtained by transforming these values during the token competition phase and taking the highest one.

2.4 Evolutionary Algorithm

Listing 1 shows the pseudocode of the GEP-MLC algorithm. As can be seen, its structure is similar to that of the standard GEP algorithm, but also takes the n

Algorithm 1. GEP-MLC pseudocode

```
Generate initial population $P(0)$
$g_{count} \leftarrow 0$
**while** $g_{count} < g_{max}$ **do**
    **for all** $l_i \in L$ (labels set) **do**
        Evaluate all individuals in $P(g_{count})$
    **end for**
    **for all** $l_i \in L$ **do**
        $total\_tokens$ = number of patterns belonging to $l_i$
        **for all** Individual $Ind_j$ in $P(g_{count})$ **do**
            $tokens\_won_{Ind_j} = 0$
        **end for**
        **for all** positive pattern (token) **do**
            Select $Ind_j$ with highest fitness that correctly classifies the pattern
            $tokens\_won_{Ind_j} ++$
        **end for**
        **for all** Individual $Ind_j$ in $P(g_{count})$ **do**
            Update the fitness by using formula 4
        **end for**
    **end for**
    Do parents selection
    Apply genetic operators
    Update population
    $g_{count} ++$
**end while**
Generate classifier
```

fitness calculation into consideration. In addition, the algorithm also implements the *Token Competition* technique to correct individual fitness after its evaluation.

Token competition [20] is used in classification algorithms to emulate the niching effect, as shown in natural ecosystems; when a species finds a convenient place to live, it does not usually evolve abruptly, but tries to adapt to a mild environment and does not allow any other species to settle there. Token Competition tries to achieve a set of specialized individuals to classify patterns sub-sets. So, for each positive pattern, a *token* is a stake which is won by the individual with the highest fitness correctly classifying the pattern. When all the tokens are distributed, the algorithm corrects the fitness of each individual using the expression:

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

Token Competition penalizes individuals that, despite their average fitness, do not contribute to the classifier. On the other hand, they help both the individuals with good fitness that correctly classify many patterns, and individuals specialized in classifying strange patterns, which are not usually correctly classified as the best individuals. In the proposed algorithm, there will be as many token competitions as labels in the training set, only using patterns with labels. Each token is played by every individual with the fitness associated to its label.

When the algorithm finishes, it is easy to find which individuals must be in the learned classifier. Only individuals that win any token are relevant for the classifier.

3 Experimental Section

The goal of the experiments is to establish if the proposed GEP-MLC algorithm is effective in dealing with multi-label classification problems, and its performance is compared to other multi-label classification algorithms. This section explains several details related with these experiments such as data sets and algorithmic details.

3.1 Data Sets

For the experimentation, the algorithm proposed has been tested with three multi-label data sets, *scene*, *emotions* and *yeast*. Scene data set contains a series of patterns about kinds of landscapes (beach, sunset, mountain, field, urban and fall foliage). Emotions data set is concerned with the classification of songs according to the emotions they evoke. Finally, Yeast includes information about protein function. Table 1 shows the main features (number of labels, attributes and patterns) of each data set. Label cardinality (the average number of labels per example) and label density (the same number divided by the total number of labels) are used to explain how much multi-label is a data set [6,7].

All data sets have been randomly split into 10 partitions in order to carry out a 10-fold cross validation. For each test, 5 different runs have been executed and an average value has been calculated in order to measure the performance of the evolutionary algorithm as independently as possible from its randomness.

3.2 Implementation

GEP-MPC implementation was made using the JCLEC library [21]. JCLEC is a software system that provides a framework to develop evolutionary algorithms. Our algorithm is based on the GEP module available at the library. The class responsible for fitness calculation and the evolutionary algorithm had to be implemented (the standard GEP algorithm did not implement the niching scheme).

Table 1. Data sets Features

Data set	#patterns	#train	#test	#attributes	#labels	density	cardinality
Scene	2407	1211	1196	294	6	1.061	0.176
Emotions	593	1500	917	103	6	1.868	0.311
Yeast	2417	391	202	78	14	4.228	0.302

Table 2. GEP-MLC algorithm parameters

Parameter	Value
Number of genes	6
Head size	35
Population size	1000
Max of generations	60
Tournament size	2
Mutation probability	0.2
Crossover probability	0.7
Transposition probability	0.4

A set of tests was made to find the optimal parameters of the algorithm (see Table 2). These parameters have been the ones used in the main experiments.

The rest of the algorithms used in the tests were available in the MULAN library. This is a Java package which contains several problem transformation and algorithm adaptation methods for multi-label classification, an evaluation framework that computes several evaluation measures and a class providing data set statistics. MULAN is built on top of the WEKA data mining tool [22] and is freely available at `http://mlkd.csd.auth.gr/multilabel.html`.

4 Results and Discussion

The performance of the proposed algorithm has been compared to four other methods for multi-label classification, namely, Binary Relevance (BR), Label Powerset (LP), RAKEL [7] and the ML-KNN method [9]. The measures of accuracy, precision and recall have been used to compare these methods. To extend these measures from single-label to multi-label we have used the macro-averaged approach proposed in [23], where precision and recall are first evaluated locally for each category, and then globally by averaging over the results of the different categories.

Table 3 shows the accuracy (acc), precision (prec) and recall (rec) results. As can be observed, the proposed algorithm obtains better results than the other ones for the three measures and data sets, being the difference more significant with the data sets which contain more multi-label patterns.

It can also be observed that the differences between the results obtained with the GEP based algorithm and the rest of the algorithms increase with the higher numbers of multi-label features (more density and cardinality). Thus, in the case of the scene data set, where density is very close to one (nearly a single label problem), the GEP algorithm obtains the best results in accuracy and recall, but these results can be compared to those obtained with RAKEL and ML-KNN. Nevertheless, when the results for emotion are analyzed, GEP is found to obtain the best scores for all measures, and the scores are much more higher than the scores of the algorithms studied. Furthermore, the same result can be observed with the yeast data set, the one with the highest number of labels and values of density and cardinality (the most multi-label data set).

Table 3. Experimental results

	Scene			Emotions			Yeast		
Algorithm	acc	prec	rec	acc	prec	rec	acc	prec	rec
Binary Relevance	0.538	0.630	0.623	0.203	0.280	0.253	0.141	0.192	0.129
Label Powerset	0.587	0.594	0.597	0.290	0.276	0.285	0.131	0.193	0.192
RAKEL	0.619	0.773	0.651	0.253	0.110	0.258	0.119	0.212	0.119
ML-Knn	0.647	0.799	0.675	0.126	0.321	0.029	0.113	0.114	0.113
GEP-MLC	0.709	0.746	0.744	0.903	0.724	0.695	0.738	0.715	0.649

5 Conclusions and Future Work

This study presents the GEP-MLC algorithm, an evolutionary algorithm for multi-label classification. This algorithm, based on GEP, codifies discriminant functions that indicate that a pattern belongs to a certain class in such a way that the final classifier is obtained by combining several individuals from the population. It uses a niching technique (token competition) to ensure that the population will present functions representing all the classes present in a given problem. Studies have been carried out to check the performance of our algorithm and compare it with those of other available algorithms, to verify that GEP-MLC renders the best performance of them all in terms of exactness, precision and recall, and is at the same time much less insensitive to the degree of overlapping in its classes, which is a very positive characteristic in this type of problem. Regarding to future research, the algorithm is being tested in other domains and, besides, the efficiency is being studied in order to be optimized.

Acknowledgment. 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) and FEDER funds.

References

1. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classification. Pattern Recognition 37(9), 1757–1771 (2004)
2. Li, T., Zhang, C., Zhu, S.: Empirical studies on multi-label classification. In: ICTAI 2006: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence, Washington, DC, USA, pp. 86–92. IEEE Computer Society, Los Alamitos (2006)
3. Li, T., Ogihara, M.: Detecting emotion in music. In: Proceedings of the 14th intern. conference on music information retrieval (ISMIR 2003), Baltimore, USA (2003)
4. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. Advances in Neural Information Processing Systems 14, 681–687 (2001)
5. Rak, R., Kurgan, L.A., Reformat, M.: Multilabel associative classification categorization of medline articles into mesh keywords. IEEE Engineering in Medicine and Biology Magazine 26(2), 47–55 (2007)

6. Tsoumakas, G., Katakis, I.: Multi label classification: An overview. International Journal of Data Warehousing and Mining 3(3), 1–13 (2007)
7. Tsoumakas, G., Vlahavas, I.: Random k-labelsets: An ensemble method for multi-label classification. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS, vol. 4701, pp. 406–417. Springer, Heidelberg (2007)
8. Clare, A., King, R.D.: Knowledge discovery in multi-label phenotype data. In: Siebes, A., De Raedt, L. (eds.) PKDD 2001. LNCS, vol. 2168, p. 42. Springer, Heidelberg (2001)
9. 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, Los Alamitos (2005)
10. Noh, H.G., Song, M.S., Park, S.H.: An unbiased method for constructing multilabel classification trees. Computational Statistics & Data Analysis 47(1), 149–164 (2004)
11. Ghamrawi, N., Mccallum, A.: Collective multi-label classification. In: CIKM 2005: Proceedings of the 14th ACM international conference on Information and knowledge management, New York, USA, pp. 195–200. ACM Press, New York (2005)
12. Crammer, K., Singer, Y.: A family of additive online algorithms for category ranking. The Journal of Machine Learning Research 3, 1025–1058 (2003)
13. Zhang, M.L., Zhou, X.H.: Multilabel neural networks with applications to functional genomics and text categorization. IEEE Transactions on Knowledge and Data Engineering 18(10), 1338–1351 (2006)
14. Rak, R., Kurgan, L., Reformat, M.: A tree-projection-based algorithm for multi-label recurrent-item associative-classification rule generation. Data & Knowledge Engineering 64(1), 171–197 (2008)
15. Schapire, R.E., Singer, Y.: Boostexter: A boosting-based system for text categorization. Machine Learning 39(2/3), 135–168 (2000)
16. Johnson, M., Cipolla, R.: Improved image annotation and labelling through multi label boosting. In: Proceedings of the British Machine Vision Conference (16th BMVC), British Machine Vision Association (BMVA), Oxford, U.K (2005)
17. Ferreira, C.: Gene expression programming:a new adaptative algorithm for solving problems. Complex Systems 13(2), 87–129 (2001)
18. Zhou, C., Xiao, W., Tirpak, T.M., Nelson, P.C.: Evolving accurate and compact classification rules with gene expression programming. IEEE Transactions on Evolutionary Computation 7(6), 519–531 (2003)
19. Han, J., Kamber, M.: Data Mining: Methods and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2006)
20. Wong, M.L., Leung, K.S.: Data Mining Using Grammar-Based Genetic Programming and Applications. Genetic Programming Series. Kluwer Academic Publishers, Dordrecht (2002)
21. 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)
22. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
23. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. 34(1), 1–47 (2002)