J. of Mult.-Valued Logic & Soft Computing, Vol. 17, pp. 183–206
Reprints available directly from the publisher
Photocopying permitted by license only

A Gene Expression Programming Algorithm for Multi-Label Classification

J. L. ÁVILA, E. L. GIBAJA, A. ZAFRA AND S. VENTURA

Department of Computer Science and Numerical Analysis. University of Córdoba
E-mail: sventura@uco.es

Received: December 16, 2009. Accepted: April 20, 2010

This paper presents a Gene Expression Programming algorithm for multi-label classification which encodes a discriminant function into each individual to show 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. The algorithm has been compared to other recently published algorithms. The results found on several datasets demonstrate the feasibility of this approach in the tackling of multi-label problems.

Keywords: Multi-label classification, discriminant functions, gene expression programming, machine learning

1 INTRODUCTION

Classification is one of the most studied tasks in the Machine Learning and Data Mining fields [18]. This task basically consists in finding a function whose input is a set of an object's attributes, and it returns a categorical variable that associates the object with a label or class.

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 elements with subsets which are defined by other labels l_j $(\forall j \neq i)$. These problems are also called single-label problems and thus, when $|L| = 2$, it is called a binary classification problem while if $|L| > 2$ it is called a multi-class one.

This classical approach cannot describe several situations where each pattern can be associated with more than one label, for example in text categorization [24, 43], protein and gene classification [12, 43], semantic scene classification [2], medical diagnosis [33] or sound categorization [23].

To deal with this kind of situation, an automatic learning paradigm called multi-label classification *(MLC)* has been developed, which allows the *one label per pattern* restriction to be avoided. The main characteristic of multi-label classification is that labels are not mutually excluding, and it allows patterns with more than one associated label. In multi-label classification, an example is formally assigned a label subset $Y \subseteq L$.

This research proposes a multi-label classification algorithm based on genetic programming. It uses *Gene Expression Programming (GEP)* specifically, a paradigm proposed by Ferreira [13]. *GEP* has been successfully used in classical classification [21, 27, 58], although it has not been previously used in multi-label classification. This paradigm has been chosen because it represents functions easily and makes them evolve to satisfactory solutions. These functions have been used as discriminant functions to build multi-label classifiers as shown below.

In addition to the genetic algorithm, an innovative version of the token competition technique [51] has been improved in order to guarantee the existence of functions for each label present in the problem.

The global results obtained in the statistical study that was carried out point out that discriminant functions with GEP are a better way to handle numerical datasets with many multi-label features than other classical multi-label approaches.

The paper is organized as follows: first we show some aspects of MLC and the state of the art (Section 2). After that, we describe the GEP paradigm (Section 3), our proposal (Section 4), the experiment carried out and the metrics which can be used to measure the performance of a multi-label algorithm (Section 5). Finally, the results are shown with conclusions about our study and future research lines (Section 6).

2 BACKGROUND

Multi-label learning covers a wide variety of topics [16]. MLC problems have been dealt with from two points of view [44]. On the one hand some studies describe several transformation methods, that is, a technique which transforms an original multi-label problem into several single label problems, that allow the use of a classical classification algorithm. On the other hand, there are studies proposing adaptation of classical classification paradigms to do multi-label classification, also called algorithm adaptation methods.

Transformation techniques and studies where they have been used are detailed below (Subsection 2.1), as well as the main algorithm adaptation methods that have been proposed (Subsection 2.2).

2.1 Transformation methods

Problem transformation techniques, also called pre-processing techniques, try to generate one or several single label datasets from one multi-label dataset before applying a classical classification technique.

The simple transformation methods are classified in [45] as *copy* methods, *selection* methods and *ignore* method.

Copy methods transform each multi-label pattern into patterns with just one of the labels from the original pattern. In addition to this method, a *copy-weight* method is also proposed, whose main characteristic is that each pattern has an inverse correlation weight with the number of labels of the original pattern.

Selection methods transform each multi-label pattern into one single label pattern, using some criteria to select the label to be associated with the pattern. Depending on the criteria, we can differentiate between these methods

- *Maximum selection:* where the pattern is associated with the most frequent label in the dataset.
- *Minimum selection:* where the pattern is associated with the least frequent label in the dataset.
- *Random selection:* where the label is selected randomly.

The *ignore* method deletes every multi-label pattern from the dataset, leaving the set with only non-multi-label patterns from the original set.

These transformations have at least two drawbacks: In the first place, they always imply some loss of information, either about relationships between labels, or when some labels are spurned. And in the second place, its computational cost would be too excessive to be applied in some datasets. However, these methods allow the use of classical classification techniques in the dataset, which are more widely tested and accepted than a multi-label equivalent.

In addition to these easy methods and to partially solve some of the drawbacks that they present, [45] also shows two more elaborated transformation methods, the *binary relevance* method (BR) and the *label powerset* method (LP).

The BR method generates binary datasets for each label, which have all the patterns of the original dataset, and where the positive patterns are those that belong to the label and the rest are considered to be negative patterns. After that, the algorithm must learn *n* binary classifiers, one for each dataset, that will determine if a pattern belongs to a certain label. These classifiers will be combined to generate a multi-label classifier.

The LP method also generates binary datasets, but it generates one dataset for each combination of labels in the original dataset. In principle this method could result in excessive computational cost, but in real-world problems the combination of labels present in a dataset is significantly less than the theoretical maximum. The algorithm must learn one binary classifier for each binary dataset generated, and the multi-label classifier is built combining these, as in the BR method.

In addition to all these methods, Read [31] proposes the *pruning transformation* method, similar to LP, but specially designed for problems with a large number of label combinations. This method eliminates the combinations which are less relevant for a given problem, and after that, uses a classical classification method.

The main studies using transformation techniques grouped by the classification paradigm are described below.

SVM have been used with multi-label data applying a preprocessing technique. Boutell's study [2] focuses on determining the advantages and disadvantages of different types of dataset pre-processsing in semantic scene classification prior to the use of an SVM algorithm. In [6, 22] there have also been datasets modified with a problem transformation algorithm prior to the use of SVM. Related to scene classification using preprocessed data is also the work of Li et al.[25].

An algorithm to solve ranking problems is proposed in [14] using *ranking by pairwise* where the classifier determines a ranking value for each label independently of the rest, thus applying a transformation, and the algorithm uses C4.5 as the base classifier. The same authors, in [15], proposed an improvement on this algorithm where ranking can be absolute (calibrated) in relation to a given pattern.

The KNN algorithm has also been used with some of the above-mentioned preprocessing data methods. As an example we can cite [34] where they have used this algorithm with BR preprocessing.

2.2 Algorithm adaptation techniques

The main multi-label adaptation techniques include classical classification techniques like SVM, decision trees, lazy classification, neural networks, probabilistic approaches and also some bioinspired proposals.

The difficulty in using SVM with multi-label is that it has been designed to split two classes. This problem has been dealt with by using transformation techniques and adapting the algorithm from at least two points of view. There is a proposal in [19] of a method that uses SVM to solve classification and ranking problems because a ranking problem can be easily transformed into a multi-label one by setting a threshold that allows us to discriminate relevant and irrelevant labels. Wan's study [50] proposes the so-called double class SVM which considers that patterns with two labels belong to an intermediate class between the two, and the SVM seek two hyperplanes that split the two classes in the common area where the multi-label patterns are.

In [56] Zhang and Zhou propose a modification of the KNN algorithm, called *ML-KNN*, which adapts this algorithm to the use of multi-label patterns.

Neural networks have been widely used in classical classification problems [53]. Several multi-label adjustments have been proposed, two of which are highlighted here. In study [8] several multi-label algorithms are shown, including a generalization of the multilayer perceptron adapted to text classification. The *BP-MLL* algorithm developed in [57] is a backpropagation algorithm that accommodates multi-label information by modifying the error function.

In [55] a probabilistic classifier based on independent component analysis is developed which uses a Laplace distribution to find hidden correlations between different labels. And finally in [36] a technique of *deconvolution* is proposed that tries to find the contribution of each individual source to a multi-label data distribution.

The use of syntactic trees for hierarchical multi-label classification is proposed in [1, 48]. This study compares the use of a single tree to that of learning several trees simultaneously. Clare and King present an adaptation of the C4.5 algorithm in [7] to multi-label classification. This algorithm is one of the most common in classical classification with decision trees. In this algorithm, the way to build decision trees is modified to deal with multi-label data, and the entropy definition used in C4.5 is changed to take all labels into consideration.

Rak et al. [30] proposed an algorithm to generate the multi-label association rules subsequently applied to mining patterns. Associative multi-label multiclass classification approximation *(MMAC)* is proposed in [39, 40], which allows the classification task (both classical and multi-label) to be performed as associative classification. The classifiers obtained performed well and with quite a high degree of scalability. These authors have also used this technique to raise a ranking algorithm called Rule Ranked multi-label *(RMR)* [38, 41]. This algorithm allows frequent relationships between attributes to be detected and determines which rules should be included in the classifier by a method of ranking.

Bio-inspired approaches have rarely been used to solve multi-label problems. We highlight the contribution of Freitas using ant colonies and another proposal based on evolutionary algorithms. Chan and Freitas [3] proposed the *MuLAM* algorithm based on ant colonies to deal with such problems. Each ant tries to find a candidate set of rules to solve the problem, discovering at least one rule in each iteration and at most one rule per class. Each iteration calculates the terms information gain and the matrix of pheromones, and later it creates classification rules. Each time the rules are generated, the patterns affected by the newly discovered rules are removed from the dataset. The algorithm finishes when the number of patterns revealed is less than a default parameter. Second, it is worth noting [47] which genetic algorithm is proposed to solve problems of hierarchical multi-label classification. The algorithm models a known set of rules as population and attemps to discover new rules that are hierarchically organized.

2.3 Ensemble methods

Ensemble methods are methods where multiple classifiers, developed under the same or different approaches, combine their answers [29]. The problem of combining the answers of the classifiers can be broached from two points of view, *bagging* and *boosting* [10]. Bagging techniques combine classifier responses democratically (by vote) where each classifier has an equal weight, while with boosting [32], classifiers try to specialize in classifying subsets of patterns.

The *RAKEL* algorithm proposed in [46] has various classifiers specialized in classifying label subsets, seeking the most relevant sets of labels. These same authors proposed the exploration of the state space before execution in order to find the best classifiers [28].

Combinations of classifiers, regardless of their base classifier, are also studied in [4] by using a neural network to determine which classifiers are the most relevant to be included in the final solution. There are also studies that use SVM as the base classifier. In [37] several similar SVM that differ from their initial parameters are combined.

Other studies combine various techniques, as in [5] which uses an ensemble to classify linguistic structures. Johnson and Cipolla [20] focus on classifying images through the use of ensembles.

3 GENE EXPRESSION PROGRAMMING

One of the main problems of genetic programming is how difficult it is to combine simplicity and expressive power in individual representation [13]. Some types of representations of individuals have been proposed, including syntactic trees, stack-based structures or state machines. However we notice that if the form of representation is simple to manipulate genetically, it loses functional complexity, and it is not suitable for solving certain problems. However if the representation allows great functional complexity, it is difficult to make it evolve toward accurate solutions.

To solve this dilemma several alternatives have been proposed among which Gene Expression Programming [13] stands out as it proposes a representation that attempts to combine simplicity with expressive power, because it proposes a model that is simple to evolve, but which can represent complex structures.

In *GEP* each individual has a dual coding. The genotype is composed of a set of genes that are represented by a simple string of characters, and the phenotype for each string is an expression tree consisting of functions in its nonterminal nodes and constant or input values in its terminal nodes.

3.1 Individual structure

GEP individuals have dual codification. Their genotype is organized as a string and phenotype in an expression tree, although they are composed of the same elements. The items that appear in these individuals are:

- a functions set, which is made up of functions that receive some parameters and can only appear in nonterminal nodes of the syntax tree. The number of parameters that each function receives will determine a certain *arity* value for the corresponding node. It is important to consider the arity of each node since, as discussed below, it is used to build valid trees.
- a terminals set, which is the set of elements that can only appear on the leaves of the tree. This set contains both constant values and input parameters received by the tree.

Each gene that constitutes the genotype is divided into two parts: head and tail. The gene's head will have a size chosen a priori for the problem, but the tail size is determined by the following expression:

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

t is the size of the tail, h the size of the head, and a the maximum arity present in nonterminal nodes. The head can contain both functions and terminal set elements, but tail can just contain terminal set elements.

The purpose of these limitations is to allow any gene to be transformed into a valid syntax tree. The format where a syntax tree is in the genotype string is called *K-Expression,* and determines the phenotype generated from the genotype.

The way to build a valid tree from a K-Expression is to fill the tree level by level. The first element of the gene will be the root of the tree. The following n elements, n being the arity of the first element, correspond to the offspring of the root node, then the algorithm proceeds to the next level, filling each node with the offspring that it needs. The process continues until all the leaves of the tree have terminal elements.

Building the tree according to this procedure ensures that all the trees generated are valid, since even in the worst case there will still be enough terminal elements in the tail to complete the tree. For example, consider as the function and terminal set: a, b, c, d (input parameters), $1, 2, 3, 4$ (constant values) and $+, -, *, /, Q$ (function set, Q being the square root).

The following mathematical expression (represented by the syntax tree of Figure 1):

$$\sqrt{(a + b) \times (c - d)} \tag{2}$$

The K-expressions, with $h = 5$ that represent the tree are like:

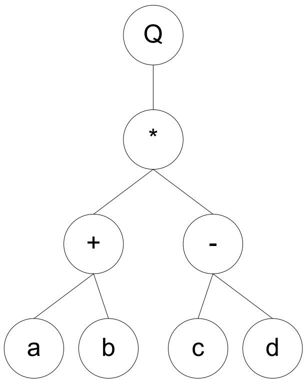

FIGURE 1
Syntax tree of a simple expression

`Q*+-abcd___`

Having in the underscore(_) any element of terminal set. For instance the strings `Q*+-abcd11b` or `Q*-abcd12a` also will be converted into the same syntax tree.

One tree can have more than one associated K-expression because the fact that an item is in the gene does not imply that it appears in the tree (for instance the last three elements in the previous example). These elements, according to Ferreira, allow the genetic diversity of the population to increase without involving a loss of good individuals. Furthermore, these elements can increase the rate of mutation without genetic drift phenomena.

It should also be noted that, in the situation where an individual is represented by several genes, the phenotype will consist of various trees. For certain problems this may be appropriate, but if the objective is for each individual to generate a specific numerical value, these trees should be connected using the so-called *connection function* which receives the same number of parameters as of genes in the genotype

The transformation process in this case occurs in two steps (Figure 2). Firstly each individual gene (a) becomes a tree (b). Then the trees are connected with a connection function (c) to produce a single tree. The connection function can be a default setting of the algorithm, or may be included at the beginning or end of the individuals to evolve along with them.

3.2 Genetic operators

The paradigm of GEP defines a set of genetic operators adapted to the dual representation of individuals, and formulated in a way that respects the structure of the genes. The following paragraphs discuss the three groups of operators defined as: crossover, transposition and mutation.

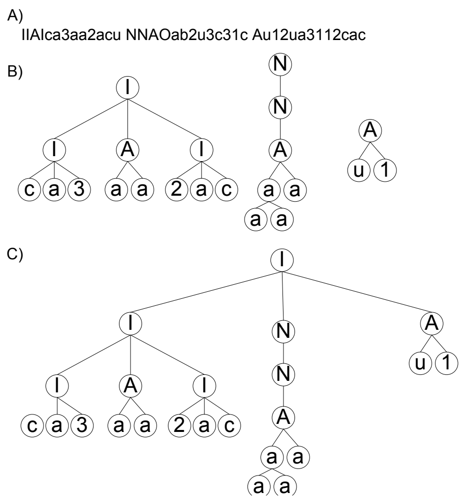

FIGURE 2
Transformation of an individual with some genes. Gene lenght, 13. Individual lenght, 3 genes

In GEP, there are three types of crossover operators: one point recombination, two point recombination and gene recombination. In all cases, two parents are chosen randomly, and part of their genotype is interchanged to generate two new offspring, with valid genes (Figure 3):

- *One point recombination*: the crossing point is randomly chosen at a position in the parents' genotype. The resulting offspring will have the genotype of the first parent up to the chosen point , and thereafter the second parent's genotype.
- *Two point recombination*: in this variant of crossover two points in the genotype are chosen. The genotype of each offspring will have a copy of the chromosome of the first parent up to the first point, a copy of the second parent to the second point, and finally a copy of the first parent.
- *Gene recombination*: It is a special two point recombination where the crossover points are the beginning and the end of one gene. In this operator one gene is randomly chosen in the genotype, and parents exchange it to generate the children.

The mutation operator performs a random change in an element in a random position of the genotype, although it must be taken into account that the internal structure of the genotype must remain intact. That means that an element belonging to the tail of the gene only changes to elements belonging to the terminal set, and an element belonging to the head of the gene may be replaced by any element in the terminal set and function set. Figure 4 shows an example. The mutation of a terminal in a function, or a function in another

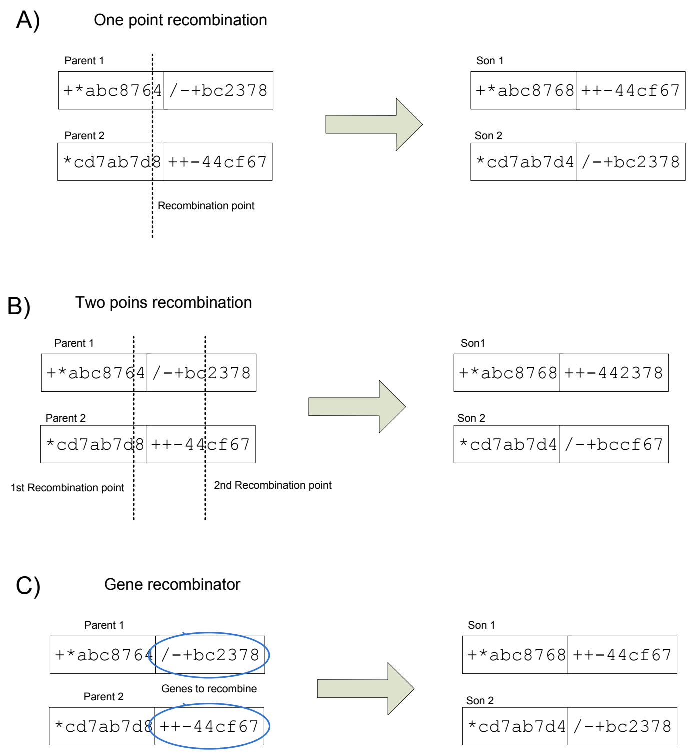

FIGURE 3
Recombination operators

function with higher arity may represent a drastic change in the phenotype of the individual, although it only affects one element. Neutral mutations, that do not alter the individual's genotype, may also occur.

There is a new type of operator in GEP terminology, called transposition operators, which are modified mutation operators that act on genes. Transposition operators select a fragment of the individual's chromosome and transfer it to another location. According to Ferreira [13], transposition operators are necessary because the mutation operator itself does not give enough population diversity to successfully explore the solution space. In GEP three transposition operators are defined (Figure 4):

- *Transposition of insertion sequence*, or *IS* transposition. This randomly selects a random sized fragment of the individual's genotype and inserts it into any part of the head of a gene except the first element of the head (the root element). The original fragment is copied (i.e. is retained in its original position) and the size of the gene does not change.

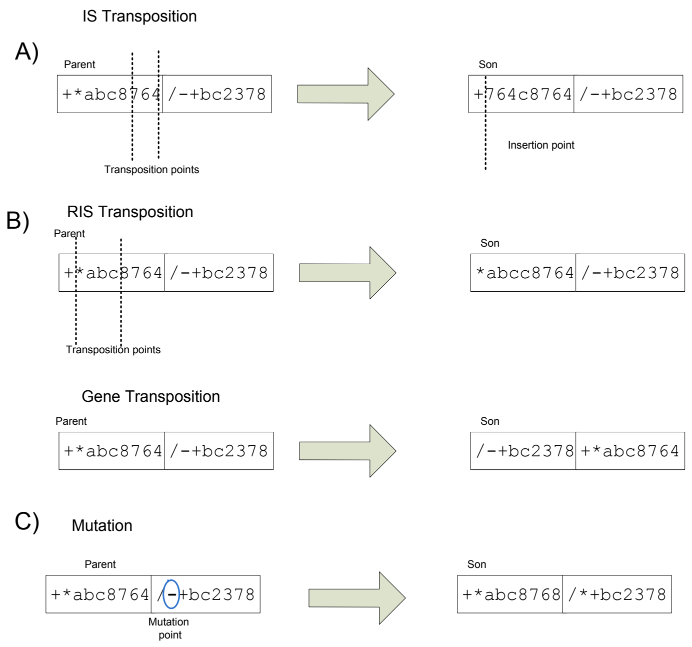

FIGURE 4
Mutation and transposition operators

- *Root transposition of insertion sequence*, or RIS transposition. This randomly selects a random sized fragment of the individual's chromosome, but with the caveat that it must begin with a function, and it is inserted into the first element of the head of a gene (the tree root).
- *Gene transposition* is applied to individuals with more than one gene and consists of changing the position of two genes on the chromosome. In this transposition two randomly chosen genes fully interchange their positions on the chromosome.

4 GEP-MLC ALGORITHM

This section describes the most relevant aspects in the design of the algorithm proposed, which are called *GEP-MLC*. The primary focus will be on the individual's representation, moving on later to discuss how they are evaluated, and finishing with an overall description of algorithm dynamics.

4.1 Individual representation

The algorithm tries to learn a set of discriminant functions for each label in the problem. A discriminant function is a mathematical function that receives the feature vector of each pattern and returns a real value. In binary classification, values above a certain threshold are interpreted as positive responses from the classifier, and values less than the threshold as negative. Usually a 0 threshold value is taken:

$$If(f(\mathbf{X}) > 0)\ then\ \mathbf{X} \in \text{class}\ else\ \mathbf{X} \notin \text{class} \tag{3}$$

For multiclass problems (number of classes $N > 2$) there are two approaches to the problem. On one hand it can be set $N-1$ thresholds defining N intervals. Depending on the range of values returned by the discriminant function, the pattern will be classified in one class or another. On the other hand, $N - 1$ functions can be used with a single threshold and deal with the membership of a class as a binary classification problem. The latter approach is the one that has been used in this study. Therefore, each individual codes the mathematical expression for a discriminant function in its genotype, like a binary classifier, and the threshold of 0 is used for all classes. As shown in Subsection 4.3, the class of each discriminant function is assigned during the evaluation of the individual.

In the algorithm developed, the subsequent evaluation of the syntax tree generated by the genotype of GEP individuals throughout a pattern determines the value of the discriminant function represented by the individual for that pattern.

Each class may have more than one associated discriminant function. Whenever any of them determines that the pattern belongs to the class, the classifier will consider it belongs to the class regardless of the results yielded by the rest. The classifier finds that the pattern does not belong to a label if all the discriminant functions associated with it do not return a positive result, i.e., the classifier consists of a set of rules (each of them associated with a discriminant function) in disjunctive normal form (linked by a logical *OR*).

4.2 Individual evaluation

When the discriminant functions are generated (Figure 2), each of these individuals is evaluated by a *fitness* function for each label of the training set. The fitness function used is the harmonic mean between precision and recall, also known as the *F-score,* because it allows the optimization of both precision and recall, which are opposite metrics. Precision for a label is the number of true positives divided by the total number of elements that belong to the label, and recall is defined as the number of true positives divided by the total number of elements that have been considered to belong to the label. The expression of precision, recall and F-score are:

$$precission = \frac{tp}{tp + fn} \tag{4}$$

$$recall = \frac{tp}{tp + fp} \tag{5}$$

t_p being the number of labels correctly assigned, f_p the incorrectly assigned labels and f_n false negatives.

$$fitness_i = \frac{2 \times precision \times recall}{precision + recall} \tag{6}$$

The fact that each training pattern has associated multiple labels implies that the fitness of each individual depends on the label used to calculate it. In addition it is also necessary for the population to have individuals that generate discriminant functions for each of these labels. Therefore the algorithm calculates one fitness value for each individual and label. For each individual a fitness vector is stored, one per label, but for the application of selection operators, only the greatest of them will be considered.

Following individual evaluation, the algorithm applies the *Token Competition* technique to correct the individual's fitness. Token competition [51] is widely used in genetic algorithms applied to classification problems [54, 26]. This technique is used in traditional classification algorithms to try to emulate the niching effect present in natural ecosystems: when a species finds a comfortable place to live; it does not usually evolve suddenly but it adapts to this benign environment and tries to prevent other species from colonizing it.

Token competition tries to encourage our population to have sets of individuals specialized in classifying subsets of patterns. To do this, each positive training pattern puts a token into play which is won by the individual with the highest fitness that correctly classifies the pattern. Once the algorithm has distributed all its tokens, it proceeds to correct the fitness of each of the individuals using the following expression:

$$new\_fitness = \frac{original\_fitness \times tokens\_won}{total\_tokens} \tag{7}$$

Token competition penalizes individuals that, despite classifying a suitable number of patterns, and being appropriately fit, contribute little or nothing to the classifier because these patterns are also correctly classified by better individuals. On the other hand, token competition favors both individuals with good fitness that classify a lot of patterns, and individuals specialized in classifying strange patterns that might be overlooked by the best individuals.

In our algorithm, a token competition is performed for each label, using only patterns belonging to the label in question. In each competition all individuals play with the fitness that was assigned to the corresponding label.

Algorithm 1 GEP-MLC

```
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 $l_i$
      $tokens\_won_{Ind_j} = 0 \; \forall Ind_j \in P(g_{count})$
      for all positive patterns (tokens) do
         $tokens\_won_{Ind_j}++$ where $Ind_j$ has the highest $fitness_{l_i}$ and correctly
         classify the pattern
      end for
      Update $fitness_{l_i}(\forall Ind_j \in P(g_{count}))$ by using formula 7
   end for
   Do parent selection
   Apply crossover operators
   Apply mutation operator
   Apply transposition operators
   Update population
   $g_{count}++$
end while
Generate classifier
```

4.3 Evolutionary algorithm

The algorithm developed is similar to that proposed with GEP[13], although it has to calculate *n* fitness for each individual, and it also must do the multi-label token competition to correct the fitness of individuals after evaluation. A pseudocode outline of the proposed algorithm can be seen in Algorithm 1.

Once the initial population is generated, as long as 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 under the label, and the fitness array is reached.
2. For each label a token competition is played. In order to do this, the algorithm will perform the following actions:

   (a) The number of tokens won by each individual starts at 0, and the total number of tokens in play is equal to the total number of labeled patterns.

(b) Each token is won by the individual with the best fitness that correctly classifies it.

(c) When all tokens are distributed, the fitness of each individual is updated using expression 7

3. After the token competition n (n being the population size) individuals are selected, using tournament selection, and they constitute the next generation. To make this selection the best fitness of each individual is used.

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.

5 EXPERIMENTAL FRAMEWORK

This section firstly describes the metrics that have been used. Subsequently we describe the initial parameters of the algorithm, the datasets used and some implementation details.

5.1 Metrics

To determine the performance of a multi-label algorithm, the measures most commonly used are the previously defined *precision* and *recall*. When calculating, the average of several tests can opt for the *macro* or *micro* approach depending on how the metric is averaged [43]. Under the macro approach, the average of each pattern is calculated before calculating the metric, and under the micro approach the value of the metric is averaged per label. With D as a dataset, Macro and Micro expressions of precision and recall are:

$$Precission_{macro} = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{tp_i}{tp_i + fn_i} \qquad (8)$$

$$Precission_{micro} = \frac{\sum_{i=1}^{|D|} tp_i}{\sum_{i=1}^{|D|}(tp_i + fn_i)} \qquad (9)$$

$$Recall_{macro} = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{tp_i}{tp_i + fp_i} \qquad (10)$$

$$Recall_{micro} = \frac{\sum_{i=1}^{|D|} tp_i}{\sum_{i=1}^{|D|} (tp_i + fp_i)} \tag{11}$$

In addition with precision and recall the *accuracy* measure has been used to compare multilabel classifiers. The accuracy is the percentage of correct answers divided by the total answers of the classifier. Thats is,

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{12}$$

Note that the results of the macro and micro approaches for accuracy are equal [45].

5.2 Implementation

The implementation of the algorithm was carried out using the *JCLEC* library [49]. JCLEC is a software system that can be used as a framework in the development of the evolutionary computation applications implemented in Java. JCLEC provides a set of generic classes that abstract the main features presented in evolutionary algorithms. It allows fast encoding applications to solve various kinds of problems in all evolutionary paradigms.

Before addressing the whole set of experiments, a series of tests was carried out to determine the optimal parameters of the algorithm. These parameters, along with those used in the rest of the experiments, are set out in detail in Table 1.

The performance of the algorithm proposed has been compared to three other methods for multi-label classification, namely, Binary Relevance (BR), Label Powerset (LP) and the ML-KNN method. BR and LP are problem transformation methods and they use the C4.5 algorithm as the base classifier, as described in [46]. The ML-KNN is a pure multi-label method, specifically a multi-label adaptation of the k-nearest neighbor proposed in [56]. We have

Population	1000 inds
#Gene	6
Head length	35
Max. generations	60
Selection	Tournament size 2
Prob. mutation	0,2
Prob. crossover	0,7
Prob. Transposition	0,4

TABLE 1
Algorithm parameters

used the implementation of these algorithms which are freely available from the *MULAN* library[1].

5.3 Datasets

The six data sets used to perform the experiments are called *scene*, *emotions*, *yeast*, *TMC2007*, *genbase* and *mediamill*. These sets belong to a wide variety of application domains and they have been used in other studies covering several paradigms.

- The dataset *scene* [2] contains a series of patterns of images from different types of landscapes. Each image is assigned to one or more labels depending on the type of landscape that contains, for example beaches, sunsets, mountains, fields, forests and urban landscapes.
- The dataset *emotions* [42] contains data from musical tracks. Each piece of music is multi-labelled depending on the emotion it evokes in the listener in one of six possible emotional categories: surprise, anger, happiness, sadness, calm and relax.
- The dataset *yeast* [12] contains information about genes, and classifies them into 14 groups depending on the functional role of proteins generated by the gene.
- The dataset *genbase* [11] contains similar information, although in this case it is not about genes, but directly concerns proteins and their role.
- *Mediamill* dataset [52] contains information about video files, and these are semantically classified in the same way as the semantic classification of scenes, so that they can be searched.
- Finally the *TMC2007* dataset [35] deals with text classification, namely technical aerospace reports that contain information and are classified according to the anomalies that have subsequently occurred and that may be identified in the following report.

Nominal attributes have been binarized as described in [59] converting them respectively into binary zero and one numeric values respectively. This process is necessary, and common in discriminant functions, because discriminant functions can only accept numerical values as input.

An important factor to take into account when discussing the experimental results of a multi-label algorithm on a dataset is how multi-labeled a dataset is. To do this in [43] are proposed two measures, *label cardinality* and *label density*. Cardinality is the average number of labels associated with each pattern. Letting y_i the number of labels of pattern i, cardinality of a D dataset

[1] at `http://mlkd.csd.auth.gr/multi-label.html`

Dataset	#Patterns	#Labels	Cardinality	Density
Scene	2407	6	1.061	0.176
Genbase	662	27	1.252	0.046
Emotions	593	6	1.868	0.311
TMC2007	28596	22	2.158	0.098
Yeast	2417	14	4.228	0.302
Mediamill	43907	101	4.376	0.043

TABLE 2
Characteristics of datasets

is given by the expression:

$$Cardinality_D = \frac{1}{|D|}\sum_{n \in D} y_i \tag{13}$$

However it is reasonable to consider the total number of labels in a given dataset in order to to avoid the distortion of comparing cardinalities between sets with different number of labels. We define density as the cardinality divided by the total number of labels Y:

$$Density_D = \frac{cardinality_D}{|Y|} \tag{14}$$

Table 2 shows the main characteristics of datasets, including cardinality and density.

A 10 fold cross validation has been done for each dataset and algorithm in order to compare their performance in terms of accuracy precision and recall.

6 RESULTS AND DISCUSSION

In order to compare GEP-MLC with BR, LP and ML-KNN methods, we have used the precision, recall and accuracy measures described below. Table 3 shows the micro approach for each dataset and algorithm in precision, and recall and the accuracy value.

The proposed algorithm obtains in general, better results, for each dataset than the other algorithms. These results are comparable in some datasets, but with others the results are much better than the rest. We can also see that differences between GEP-MLC and other algorithms increase when the observed dataset has more cardinality and density. In other words, when the dataset has more multi-label features (Figure 5).

For example, if we see the results of scene, TMC, genbase and mediamill, whose density is small, GEP-MLC will have the best recall, but the results in

Algorithm Dataset	**Bin. Rel.**	**Label Pow.**	**ML-Knn**	**GEP-MLC**
		Accuracy		
Scene	0.538	0.587	0.647	0.709
Genbase	0.634	0.621	0.585	0.755
Emotions	0.203	0.290	0.126	0.903
TMC2007	0.443	0.613	0.402	0.543
Yeast	0.141	0.131	0.113	0.738
Mediamill	0.582	0.594	0.470	0.703
		Precision		
Scene	0.630	0.594	0.799	0.746
Genbase	0.550	0.535	0.428	0.650
Emotions	0.280	0.276	0.321	0.724
TMC2007	0.582	0.603	0.437	0.618
Yeast	0.192	0.193	0.114	0.715
Mediamill	0.609	0.556	0.419	0.669
		Recall		
Scene	0.623	0.597	0.675	0.744
Genbase	0.596	0.533	0.383	0.582
Emotions	0.597	0.285	0.029	0.695
TMC2007	0.503	0.573	0.483	0.540
Yeast	0.129	0.192	0.113	0.649
Mediamill	0.499	0.521	0.127	0.581

TABLE 3
Experimental results.

accuracy, recall and precision are similar to those obtained by ML-KNN and LP. In fact, they are barely higher than the other two metrics.

However, with respect to yeast and mediamill results, it is clear that the proposed algorithm obtains much better accuracy, precision and recall than the others.

The huge difference between algorithms in highly labeled sets can be explained taking into consideration that BR and LP pre-process data which could lead to a loss in relationships between labels. ML-GEP is also less sensitive to the degree of overlapping between classes than the ML-KNN method.

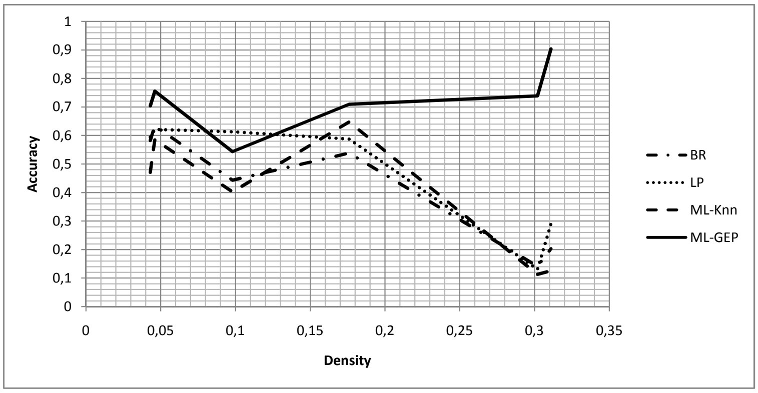

FIGURE 5
Accuracy of algorithms and density of datasets

With respect to the TMC dataset, it does not show the previously mentioned tendency, because ML-GEP does not obtain as good a result as we could hope for when first observing the cardinality and density of this dataset. However it is worth stressing that TMC is, unlike the rest, a Boolean dataset whose data have been turned into numerical values to use discriminant functions. This fact suggests that the proposed algorithm is especially unsuitable for nominal attributes, quite a reasonable hypothesis because discriminant functions can not handle them directly.

Algorithms have been statistically compared using a Friedman test. The Friedman test [9, 17] is a nonparametric test that compares the average ranks of the algorithms, where the algorithm with the highest accuracy in one data set is given a rank of 1 for that dataset, the algorithm with the next highest accuracy value has the rank of 2, and so on. Finally, the average ranks of each algorithm for all data sets are calculated. These ranks let us know which algorithm obtains the best results considering all data sets. In this way, the algorithm with the value closest to 1 indicates the best algorithm on most data sets.

Algorithm	Ranking acc.	Ranking prec.	Ranking rec.
BR	2.833	2.833	2.500
LP	2.333	2.833	2.500
MLKNN	3.666	3.166	3.666
MLGEP	1.166	1.166	1.333

TABLE 4
Average Rankings of the algorithms for accuracy, precision and recall

Algorithm rankings are shown in Table 4. Friedman statistics are 11.8 for accuracy, 8.79 for precision and 9.79 for recall which indicates that there are significant differences between methods. The Critical Difference value for the Bonferroni-Dunn post test considering $p = 0.10$ is 1.58 for all measures which manifests a statistical difference with a probability of 90% between GEP-MLC and BR and ML-KNN.

7 CONCLUSIONS

This study presents the GEP-MLC algorithm, an evolutionary algorithm for multi-label classification. This algorithm, based on GEP, encodes discriminant functions that indicate whether a pattern belongs to a particular class, and it makes the final classifier by combining multiple functions present in the population.

The algorithm uses the multi-label token-competition technique to ensure that there are individuals in the population representing all the classes present in the problem.

Studies carried out to verify the performance of the GEP-MLC algorithm with respect to other alternatives like BR and LP transformations and multi-label KNN, indicate that it gets the best results and that it is also less sensitive to the degree of overlapping between classes.

ACKNOWLEDGMENT

This work has been finnanced 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] H. Blockeel, L. Schietgat, J. Struyf, S. Dzeroski, and A. Clare. (2006). Decision trees for hierarchical multilabel classification: A case study in functional genomics. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4213 LNAI:18–29.

[2] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. (2004). Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771.

[3] A. Chan and A. A. Freitas. (2006). A new ant colony algorithm for multi-label classification with applications in bioinfomatics. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 27–34, New York, NY, USA. ACM Press.

[4] P. Chan, X. Zeng, E. Tsang, D. Yeung, and J. Lee. (2007). Neural network ensemble pruning using sensitivity measure in web applications. In *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, pages 3051–3056.

[5] J. Chen, X. Zhou, and Z. Wu. (2004). A multi-label chinese text categorization system based on boosting algorithm. In *Proceedings of the The Fourth International Conference on Computer and Information Technology*, pages 1153–1158, Washington, DC, USA. IEEE Computer Society.

[6] W. Chen, J. Yan, B. Zhang, Z. Chen, and Q. Yang. (2007). Document transformation for multi-label feature selection in text categorization. In *Proceeding of seventh IEEE International Conference on Data Mining ICDM07*, pages 451–456.

[7] A. Clare and R. D. King. (2001). Knowledge discovery in multi-label phenotype data. *Lecture Notes in Computer Science*, 2168:42–53.

[8] K. Crammer and Y. Singer. (2003). A family of additive online algorithms for category ranking. *The Journal of Machine Learning Research*, 3:1025–1058.

[9] J. Demšar. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

[10] T. Dietterich. (2000). Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15.

[11] S. Diplaris, G. Tsoumakas, P. Mitkas, and I. Vlahavas. (2005). Protein classification with multiple algorithms. *Advances in Informatics*, pages 448–456.

[12] A. Elisseeff and J. Weston. (2001). A kernel method for multi-labelled classification. *Advances in Neural Information Processing Systems*, 14:681–687.

[13] C. Ferreira. (2001). Gene expression programming:a new adaptative algorithm for solving problems. *Complex Systems*, 13(2):87–129.

[14] J. Fürnkranz and E. Hüllermeier. (2003). Pairwise preference learning and ranking. In *In Proceeding ECML-03, Cavtat-Dubrovnik*, pages 145–156.

[15] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker. (2008). Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153.

[16] Tsoumakas G., Zhang M., and Zhou Z., (2009). Learning from multi-label data. ECML PKDD Tutorial. On line at http://www.ecmlpkdd2009.net/wp-content/uploads/2009/08/learning-from-multi-label-data.pdf.

[17] S. Garcia and F. Herrera. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694.

[18] J. Han and M. Kamber. (2006). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 500 Sansome Street, Suite 400, San Francisco, CA 94111, second edition.

[19] A. Jiang, C. Wang, and Y. Zhu. (2008). Calibrated rank-svm for multi-label image categorization. In *(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on Neural Networks IJCNN 08*, pages 1450–1455.

[20] M. Johnson and R. Cipolla. (2005). Improved image annotation and labelling through multi label boosting. In *Proceedings of the British Machine Vision Conference (16th BMVC)*, Oxford, U.K. British Machine Vision Association (BMVA).

[21] J.K. et al Kishore. (2000). Application of genetic programming for multicategory pattern classification. *IEEE Transactions on evolutionary computation.*, 4:87–129.

[22] B. Lauser and A. Hotho. (2003). Automatic multi-label subject indexing in a multilingual environment. Lecture Notes in Computer Science, pages 140–151, Trondheim, Norway. Springer.

[23] T. Li and M. Ogihara. (October 2003). Detecting emotion in music. *Proceedings of the fourth international conference on music information retrieval (ISMIR, 2003)*.

[24] Tao Li, Chengliang Zhang, and Shenghuo Zhu. (2006). Empirical studies on multi-label classification. In *ICTAI '06: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, pages 86–92, Washington, DC, USA. IEEE Computer Society.

[25] X. Li, L. Wang, and E. Sung. (2004). Multilabel svm active learning for image classification. *Image Processing, 2004. ICIP '04. 2004 International Conference on*, 4:2207–2210 Vol. 4.

[26] Wei Lu and I. Traore. (2003). Detecting new forms of network intrusion using genetic programming. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 3, pages 2165–2172 Vol.3.

[27] P. S. Ngan, K. S. Leung, M. L. Wong, and J. C. Y. Cheng. (1998). Using grammar based genetic programming for data mining of medical knowledge. *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 254–259.

[28] Ioannis Partalas, Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. (2006). Ensemble pruning using reinforcement learning. *Advances in Artificial Intelligence*, pages 301–310.

[29] R. Polikar. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45.

[30] Rafal Rak, Lukasz Kurgan, and Marek Reformat. (2008). A tree-projection-based algorithm for multi-label recurrent-item associative-classification rule generation. *Data & Knowledge Engineering*, 64(1):171–197.

[31] J. Read. (2008). A pruned problem transformation method for multi-label classification. In *Proc. 2008 New Zealand Computer Science Research Student Conference*, pages 143–150.

[32] R. Schapire, (2001). The boosting approach to machine learning: An overview.

[33] Shun, Leung, Wai Lam, Kwong S. Leung, and Jack C. Cheng. (1999). Medical data mining using evolutionary computation. *Artificial Intelligence in Medicine*, 16(1):73–96.

[34] E. Spyromitros, G. Tsoumakas, and Ioannis Vlahavas. (2008). An empirical study of lazy multilabel classification algorithms. *Artificial Intelligence: Theories, Models and Applications*, pages 401–406.

[35] A.N. Srivastava and B. Zane-Ulman. (March 2005). Discovering recurring anomalies in text reports regarding complex space systems. In *Aerospace Conference, 2005 IEEE*, pages 3853–3862.

[36] A. s Streich and J. Buhmann. (2008). Classification of multi-labeled data: A generative approach. *Machine Learning and Knowledge Discovery in Databases*, pages 390–405.

[37] B. Sun, X.g Zhang, and R. Wang. (2007). On constructing and pruning svm ensembles. In *Third International IEEE Conference on Signal-Image Technologies and Internet-Based System SITIS '07.*, pages 855–859.

[38] F. Thabtah, P. Cowling, and Y. Peng. (2005). Mcar: multi-class classification based on association rule. In *The 3rd ACS/IEEE International Conference Computer Systems and Applications.*

[39] F. A. Thabtah, P. Cowling, and Y. Peng. (2004). Mmac: A new multi-class, multi-label associative classification approach. Proceedings - Fourth IEEE International Conference on Data Mining, ICDM 2004, pages 217–224, Bradford, United Kingdom.

[40] F. A. Thabtah, P. Cowling, and Y. Peng. (2006). Multiple labels associative classification. *Knowledge and Information Systems*, 9(1):109–129.

[41] F. A. Thabtah and P. I. Cowling. (2007). A greedy classification algorithm based on association rule. *Appl. Soft Comput.*, 7(3):1102–1111.

[42] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. (2008). Multilabel classification of music into emotions. In *Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008), Philadelphia, PA, USA, 2008*, pages 50–65.

[43] G. Tsoumakas and I. Katakis. (2007). Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13.

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

[45] G. Tsoumakas, I. Katakis, and I. Vlahavas. (2009). *Data Mining and Knowledge Discovery Handbook*, chapter Mining Multi-label Data. Springer.

[46] G. Tsoumakas and I. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4701 LNAI:406–417.

[47] R. Vallim, D. Goldberg, X. Llorà, T. Duque, and A. Carvalho. (2008). A new approach for multi-label classification based on default hierarchies and organizational learning. In *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, pages 2017–2022.

[48] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning*, pages 185–214.

[49] S. Ventura, C. Romero, A. Zafra, J. A. Delgado, and C. Hervás. (2008). JCLEC: A Java framework for evolutionary computation. *Soft Computing*, 12(4):381–392.

[50] Shu P. Wan and Jian H. Xu. (2007). A multi-label classification algorithm based on triple class support vector machine. In *Proc. 2007 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR'07)*.

[51] M. L. Wong and K. S. Leung. (2000). *Data Mining Using Grammar-Based Genetic Programming and Applications*. Kluwer Academic Publishers, Norwell, MA, USA.

[52] M. Worring, C. G. M. Snoek, O. de Rooij, G. P. Nguyen, and A. W. M. Smeulders. (2007). The mediamill semantic video search engine. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–1213–IV–1216.

[53] Y. Yang and X. Liu. (1999). A re-examination of text categorization methods. In *22nd Annual International SIGIR*, pages 42–49, Berkley.

[54] Q. Yu, K. Tan, and T. Lee. (2005). Knowledge discovery in data mining via an evolutionary algorithm. *Evolutionary Computation in Data Mining*, pages 101–123.

[55] J. Zhang, Z. Ghahramani, and Y. Yang. (2005). Learning multiple related tasks using latent independent component analysis. In *Advances in Neural Information Processing Systems 18, Neural Information Processing Systems, NIPS 2005*, pages 1585–1592, Cambridge, MA. MIT Press.

[56] M. L. Zhang and Z. H. Zhou. (2005). A k-nearest neighbor based algorithm for multi-label classification. In *IEEE International Conference on Granular Computing*, volume 2, pages 718–721 Vol. 2. The IEEE Computational Intelligence Society.

[57] M. L. Zhang and Z. H. Zhou. (2006). Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351.

[58] C. Zhou, W. Xiao, T. M. Tirpak, and P. C. Nelson. (2003). Envolving accurate and compact classification rules with gene expression programming. *IEEE Transactions on evolutionary computation.*, 7:87–129.

[59] Chi Zhou, Weimin Xiao, Thomas M. Tirpak, and Peter C. Nelson. (2003). Evolving accurate and compact classification rules with gene expression programming. *IEEE Trans. Evolutionary Computation*, 7(6):519–531.