Information Sciences 180 (2010) 4496–4513

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

G3P-MI: A genetic programming algorithm for multiple instance learning

Amelia Zafra, Sebastián Ventura *

Department of Computer Science and Numerical Analysis, University of Cordoba, Spain

ARTICLE INFO

Article history:
Received 27 January 2009
Received in revised form 18 July 2010
Accepted 27 July 2010

Keywords:
Multiple instance learning
Grammar-Guided Genetic Programming
Rule learning

ABSTRACT

This paper introduces a new Grammar-Guided Genetic Programming algorithm for resolving multi-instance learning problems. This algorithm, called G3P-MI, is evaluated and compared to other multi-instance classification techniques in different application domains. Computational experiments show that the G3P-MI often obtains consistently better results than other algorithms in terms of accuracy, sensitivity and specificity. Moreover, it makes the knowledge discovery process clearer and more comprehensible, by expressing information in the form of IF-THEN rules. Our results confirm that evolutionary algorithms are very appropriate for dealing with multi-instance learning problems.

1. Introduction

Multiple instance learning (MIL), introduced by Dietterich et al. [21], is a generalization of traditional supervised learning. In MIL, training patterns called bags are represented as a set of feature vectors called instances. Each bag contains a number of non-repeated instances and each instance usually represents a different view of the training pattern attached to it. There is information about the bags and each one receives a special label, although the labels of instances are unknown. The problem consists of generating a classifier that will correctly classify unseen bags of instances. The key challenge in MIL is to cope with the ambiguity of not knowing which instances in a positive bag are actually positive examples, and which ones are not. In this sense, a multiple instance learning problem can be regarded as a special kind of supervised learning problem with incomplete labeling information.

This learning framework is receiving growing attention in the machine learning community because numerous real-world tasks can be represented very naturally as multiple instance problems. These tasks include text categorization [3], content-based image retrieval [68,43,70], image annotation [67,24], drug activity prediction [40,80], web index page recommendation [77,69] and semantic video retrieval [13]. In order to resolve these problems, the literature abounds in a great number of method proposals that range from algorithms designed specifically to solve multi-instance problems to extensions of classical algorithms adapted to the MIL scenario. The latter include *k*-nearest neighbors, decision trees, rule based systems, support vector machines, neural networks, inductive logic programming and ensembles.

This paper presents a framework for modifying and extending evolutionary algorithms (EAs) to deal with MIL problems. One extension of these algorithms, based on Grammar-Guided Genetic Programming (G3P), is designed and implemented; the resulting algorithm is called G3P-MI. It is evaluated, analyzed and compared to other MIL classification techniques in a collection of multi-instance datasets. Experimental results show that G3P-MI often obtains consistently better results than other algorithms in several applications. Moreover, it adds comprehensibility and clarity to the knowledge discovery process,

* Corresponding author. Address: Department of Computer Science and Numerical Analysis, University of Cordoba, Campus Universitario Rabanales, Edificio Einstein, Tercera Planta, 14071 Cordoba, Spain. Tel.: +34 957212218; fax: +34 957218630.
E-mail addresses: azafra@uco.es (A. Zafra), sventura@uco.es (S. Ventura).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.07.031

expressing information in the form of IF-THEN prediction (classification) rules. These results show that EAs are a promising tool for solving MIL problems.

In summary, the key points of this paper are: to introduce evolutionary algorithms to solve MIL; to study the effectiveness of our algorithm (G3P-MI) in obtaining classification accuracy and a high degree of discovered knowledge (comprehensible rules); to present an empirical comparison between different data sets using different algorithms; and to provide data sets to facilitate future comparisons in this area. The last two points are very relevant to forward real progress in MIL because they allow different methods to be compared under the same conditions. Currently, this is the greatest weakness found in studies on MIL because, although a wide range of problem domains have been researched using a broad spectrum of approaches, extensive comparisons of algorithms are infrequent; most studies empirically compare only a few approaches and use only a few data sets to do so (often only one).

The paper is structured as follows. Section 2 briefly reviews advances in the area of multiple instance learning. Section 3 describes the proposed algorithm. Section 4 evaluates and compares our algorithm to other techniques implemented in the WEKA tool in ten datasets. Finally, Section 5 draws some conclusions and raises several issues for future work.

2. Antecedents

This section gives a definition and a notation of MIL and reviews the most important developments in MIL in recent years.

2.1. Definition and notation of multiple instance learning

MIL is designed to solve the same problems as single-instance learning: learning a concept that correctly classifies training data as well generalizing unseen data. Although the actual learning process is quite similar, the two approaches differ in the class labels provided which are what they learn from. In a traditional machine learning setting, an object m_i is represented by a feature vector v_i, which is associated with a label $f(m_i)$. However, in the multiple instance setting, each object m_i may have V_i various instances denoted $m_{i,1}, m_{i,2}, \ldots, m_{i,v_i}$. Each of these variants will be represented by a (usually) distinct feature vector $V(m_{i,j})$. A complete training example is therefore written as $(\{V(m_{i,1}), V(m_{i,2}), \ldots, V(m_{i,v_i})\}, f(m_i))$.

The goal of learning is to find a good approximation to the function $f(m_i)$, $\hat{f}(m_i)$, analyzing a set of training examples and labeled as $f(m_i)$. To obtain this function Dietterich defines a hypothesis that assumes that if the result observed is *positive*, then at least one of the variant instances must have produced that positive result. Furthermore, if the result observed is *negative*, then none of the variant instances could have produced a positive result. This can be modelled by introducing a second function $g(V(m_{i,j}))$ that takes a single variant instance and produces a result. The externally observed result, $f(m_i)$, can then be defined as follows:

$$f(m_i) = \begin{cases} 1 & \text{if } \exists\, j | g(V(m_{i,j})) = 1, \\ 0, & \text{otherwise.} \end{cases}$$

In the early years of multi-instance learning research, all multi-instance classification work was based on this assumption, that is, that MIL formulations explicitly or implicitly encoded the assumption that a bag was positive if and only if at least one of its instances was positive. This formulation is known as the standard multi-instance hypothesis or Dietterich hypothesis. More recently, generalized multi-instance learning models have been formalized where a bag is qualified to be positive if instances in the bag satisfy some sophisticated constraints other than simply having at least one positive instance. Firstly, Weidmann et al. [59] defined three kinds of generalized multi-instance problems, based on employing different assumptions of how the classification of instances classifications determine their bag label. These definitions are *presence-based MI*, *threshold-based MI*, and *count-based MI*. *Presence-based MI* is defined in terms of the presence of at least one instance of each concept in a bag (note that the standard hypothesis is a special case of this assumption which considers just one underlying concept); *threshold-based MI* requires a certain number of instances of each concept to be present simultaneously and *count-based MI* requires a maximum as well as a minimum number of instances of a certain concept in a bag. Independently, Scott et al. [51] defined another generalized multi-instance learning model in which a bag label is not based on the proximity of one single instance to one single target point. Rather, a bag is positive if and only if it contains a collection of instances, each near one of a set of target points.

Similarly, there has also been an extension of the definition of multi-instance problems which use outputs with discrete values (in particular, binary). The first proposal to use real-valued labels was introduced by Amar et al. [2]. These problems, known as multi-instance regression (MIR) problems, have drawn wide-spread attention. Ray and Page [48] modified regression for MIL by assuming that if an algorithm could identify the best instance in each bag, then straightforward regression could be used to learn a labeling method for each instance (and then for each bag); Ramon and De Raedt [46] adapted internal labeling methods for neural networks to learn a function that used real-valued instances; Dooly et al. [22] presented extensions of *k*-nearest neighbors (*k*-NN), Citation-kNN, and the diverse-density algorithm for the real-valued setting and studied their performance on boolean and real-valued data.

2.2. Literature review of multiple instance learning

The first algorithms for learning from multiple instance examples were developed by Dietterich et al. [21] for drug activity prediction. Their algorithms, named axis-parallel rectangle (APR) methods [21], attempted to find an APR by expanding or shrinking a hyperrectangle in the instance feature space, thus maximizing the number of instances of positive bags enclosed within the rectangle while minimizing the number of instances of negative bags within the rectangle. A bag was classified as positive if at least one of its instances fell within the APR; otherwise, the bag was classified as negative.

Following the basic idea of Dietterich et al.'s work, an extensive number of theoretical studies have appeared that have contributed much to multi-instance learning. Long and Tan [37] initiated the investigation of the PAC-learnability of axis-parallel rectangles within the multi-instance learning framework. They described a high-order polynomial-time theoretical algorithm and showed that if the instances in the bags were independently drawn in product distribution, then the APR was PAC-learnable. Auer et al. [5] showed that if the instances in the bags were not independent, then APR learning under the multi-instance learning framework was NP-hard. Moreover, they also presented a theoretical algorithm that did not require product distribution and that needed less time and sample complexity than Long and Tan's algorithm. Later, this theoretical algorithm was transformed into a practical algorithm named MULTINST [4]. Blum and Kalai [8] reduced the problem of PAC-learning in the multi-instance learning framework to PAC-learning with one-sided or two-sided random classification noise.

From a practical standpoint, there are algorithms designed specifically for MIL, like Maron and Lozano's diverse density (DD) [40] well-known multi-instance learning algorithm proposal. In this algorithm, the diverse density of a point, p, in the feature space is defined as a probabilistic measure contempling how many positive bags are near to an instance p, and how far the negative instances are from p. This algorithm has been extended several times. Zhang and Goldman [75] combined it with the expectation–maximization (EM) algorithm, EM-DD, a general-purpose MI algorithm whose basic idea is to see which instance corresponds to the label of the bag. For that, this information is dealt as a missing value and it is estimated using the EM approach. Pao et al. [43] proposed an EM based learning algorithm to provide a comprehensive procedure to maximize the measurement of DD in given multiple instances. However, since the conception of MIL, most research has dealt with designing multiple instance learning algorithms focused on adapting standard supervised learning techniques to solve multiple-instance problems. Currently, almost all popular machine learning algorithms have been applied to solve multiple instance problems. In the lazy learning context, Wang and Zucker [58] proposed two alternatives, Citation-KNN and Bayesian-KNN, which extend the k-nearest neighbour algorithm (KNN) to the MIL context. These algorithms defined a different calculation of distance, referred to as Hausdorff distance, to measure the separation between bags. With respect to decision trees and learning rules, Chevaleyre and Zucker [17] implemented the ID3-MI and RIPPER-MI, which are multi-instance versions of the decision tree algorithm ID3 and the rule learning algorithm RIPPER, respectively. At that time, Ruffo [50] presented a multi-instance version of the algorithm C4.5, which was called RELIC. Later, Zhou et al. [77] presented the Fretcit-KNN algorithm, which is a variant of the Citation-KNN.

There are also contributions which extend standard neural networks to MIL. Ramon and De Raedt [46] presented a neural network framework for MIL, followed by others who have further improved or extended it. For instance, Zhou and Zhang [79] proposed BP-MIP, a neural network based multi-instance learner derived from the traditional backpropagation neural network. This algorithm was improved by Zhang and Zhou [72] by adopting two different feature selection techniques (feature scaling with diverse density and feature reduction with principal component analysis). After that, Zhang and Zhou [74] designed a neural network multi-instance learning algorithm called RBF-MIP which was derived from the popular radial basis function (RBF) methods. Chai and Yang [11] proposed a neural network algorithm for Multi-Instance Learning based on a normalized RBF network defining a new kernel function for dealing with labeled bags. Another approach that has been adapted to the MIL framework is the support vector machine (SVM). There are numerous proposals for this approach: Gartner et al. [27] adapted kernel methods to work with MIL data by modifying the kernel distance measures to handle sets. Using a similar approach, Chen and Wang [15] and Chen et al. [14] adapted SVMs by modifying the form of the data rather than changing the underlying SVM algorithms, while Andrews et al. [3] adapted the SVM kernels directly to produce one of the best MIL classification systems. Other studies that use this approach include those of Mangasarian and Wild [39] and Gu et al. [30]. Finally, Xu and Frank [66], Zhang and Zhou [73] and Zhou and Zhang [80] show the use of ensembles to enhance multi-instance learners.

Along with the multitude of adaptations carried out, there have also been numerous theoretical studies on the relationship between supervised and multi-instance learning, and on the importance of considering special adaptations to appropriately solve problems based on this learning. Ray and Craven [47] empirically studied the relationship between supervised and multiple instance learning by comparing several MIL methods to their supervised counterparts. They observed that straight-forward supervised learning techniques can outperform some specialized MIL approaches, although neither approach is a clear winner. Zhou [76] showed that multi-instance learners can be derived from supervised learners by shifting their focus from discrimination on the instances to discrimination on the bags. Recently, Zhou and Xu [78] have established a bridge between MIL and semi-supervised learning showing that multi-instance learning can be viewed as a special case of semi-supervised learning.

Finally, it is worth mentioning that MIL has also attracted the attention of the inductive logic programming (ILP) community. De Raedt [45] showed that multi-instance problems could be regarded as a bias in inductive logic programming. He also suggested that the multi-instance paradigm could be the key between propositional and relational representations, being more expressive than the former, and much easier to learn than the latter. Zucker and Ganascia [81] presented REPEAT,

an ILP system based on an ingenious bias which first reformulated relational examples in a multi-instance database, and then induced the final hypothesis with a multi-instance learner. Later, Alphonse and Matwin [1] successfully employed multi-instance learning to help relational learning, where the expressive power of relational representation and the ease of feature selection in propositional representation are combined. This work confirms that multi-instance learning could act as a bridge between propositional and relational learning.

3. G3P-MI

Genetic Programming (GP), introduced by Koza [33], is a learning methodology belonging to the family of evolutionary computation (EC) [7]. Among successful EC implementations, GP retains a significant position due to such valuable characteristics as: its great flexibility for representing solutions, the fact that a priori knowledge is not needed about the statistical distribution of the data (data distribution free), data in their original form can be used to operate on them directly, unknown relationships that exist among data can be detected and expressed as a mathematical expression and finally, the most important discriminative features of a class can be revealed. These characteristics convert these algorithms into a paradigm of growing interest both for obtaining classification rules [36,56], and for other tasks related to prediction, such as feature selection [19,41] and the generation of discriminant functions [18,31]. There are other algorithms that use the GP paradigm to evolve rule sets for different classification problems that are both two-class [52,64], and multiple-class [42,71] showing that GP is a mature field that efficiently achieves low error rates in supervised learning and is still introducing improvements into its methods [32]. These results suggest that it would be interesting to adapt this paradigm to multiple instance learning and check its performance.

Our approach uses an extension of traditional GP systems, called Grammar-Guided Genetic Programming (G3P) [60]. G3P facilitates the efficient automatic discovery of empirical laws and provides a more systematic way of handling typing. More importantly, the G3P can constrain search space so that only grammatically correct individuals are generated. To achieve these goals, G3P employs a context-free grammar which establishes a formal definition of the syntactical restrictions of the problem to be solved and its possible solutions. As will be seen, the implementation of constraints using a grammar can be a very natural way to express the syntax of rules when individual representation is specified. Specifically, our system expresses the information in the form of IF-THEN classification rules which provide a natural extension for knowledge representation. The comprehensibility of the knowledge discovered has been an area of growing interest and this comprehensibility is currently considered to be just as important as obtaining high predictive accuracy. One of its significant characteristics is that the user can understand the system's results and combine them with his/her knowledge to make a well-informed decision, rather than blindly trusting the incomprehensible output of a *black box* system. In this context, our system has the advantage of being able to add comprehensibility and clarity to the knowledge discovery process.

The design of the system will be examined in more detail in continuation with respect to the following aspects: individual representation, genetic operators, the fitness function and the evolutionary process.

3.1. Individual representation

In our approach, an individual represents IF-THEN rules. The IF part of the rule (antecedent) contains a logical combination of conditions about the values of predicting attributes, and the THEN part (consequent) contains the predicted class for the concepts satisfied by the antecedent of the rule. This rule determines if a bag should be considered positive (that is, if it is an instance of the concept to be represented) or negative (if it is not).

If ($cond_B$(bag)) *then*
  bag is an instance of the concept.
Else
  bag is not an instance of the concept.
End-If

where $cond_B$ is a condition that is applied to the bag. Following the Dietterich hypothesis, $cond_B$ can be expressed as:

$$cond_B(bag) = \bigvee_{\forall instance \in bag} cond_I(instance), \tag{1}$$

where $\vee$ is the disjunction operator, and $cond_I$ is a condition that is applied to every instance contained in a given bag.[1]

Given that the only variable part in the last expressions is the condition that is applied to instances (that is, $cond_I$), the individual genotype represents this part, while the phenotype represents the entire rule that is applied to the bags. Fig. 1 shows the grammar used to represent rules. These rules can be applied to a great number of learning problems and are used in the experiments described in Section 4.

[1] This expression is equivalent to the one used to define the concept of multi-instance rule coverage in the RIPPER-MI algorithm [16].

⟨S⟩→⟨cond$_I$⟩

⟨cond$_I$⟩→⟨cmp⟩| **OR** ⟨cmp⟩ ⟨cond$_I$⟩| **AND** ⟨cmp⟩ ⟨cond$_I$⟩

⟨cmp⟩→ ⟨op-num⟩ ⟨variable⟩ ⟨value⟩| ⟨op-cat⟩ ⟨variable⟩ ⟨value⟩

⟨op-cat⟩ → **EQ | NOT EQ**

⟨op-num⟩→ **GT | GE | LT | LE**

⟨variable⟩ → *Any valid attribute in dataset*

⟨value⟩ → *Any valid value*

Fig. 1. Grammar used for representing individuals' genotypes in G3P-MI.

3.2. Initialization

The initialization process consists in generating a group of initial solutions (learning rules). This issue has received surprisingly little attention in genetic programming literature [38]. Very few papers have appeared in reference to this subject [38] since the Grow, Full and Ramped Half-and-half initial algorithms [33], and some posterior methods that select a requested size and guarantee the generation of trees with the same or a smaller size [9,12].

Our algorithm employs a simple and commonly used approach to generate the initial population inspired by that defined by Geyer-Shultz [29]. On the one hand, this process assures the generation of valid individuals because new individuals are generated by executing the production rules of the language defined by context-free grammar. On the other hand, our approach guarantees that the syntax tree generated adopts the size set by the user between a maximum and minimum number of derivations. Its performance consists of two steps: the first step selects a derivation value for the current individual (this value will comprise from a minimum value to a maximum derivation value). The next step consists in deriving the productions to generate an individual with the number of fitted derivations. The process starts with the axiom of the grammar, then chooses one of the available productions for this axiom. The same process continues, and derives the non-terminal symbols of the consequent of the production executed until no more non-terminal symbols are obtained. The choice of the production to generate new individuals is not made totally at random. In order to guarantee that the syntax tree generated is valid and uses the appropriate number of derivations, the system calculates a selection probability for each symbol in the grammar according to a specified number of derivations for that individual. This table of probabilities, although implying some computational input, only has to be calculated once since it is saved with the rest of the structural information about the individuals.

3.3. Genetic operators

G3P-MI uses two genetic operators to generate new individuals in a given generation of the evolutionary algorithm. These operators are based on selective crossover and selective mutation as proposed by Whigham [61], and their basic principles and functioning are briefly described in this section.

3.3.1. Crossover operator

This operator creates new rules by mixing the contents of two parent rules. To do so, a non-terminal symbol is chosen at random with uniform probability from among the available non-terminal symbols in the grammar and two sub-trees (one from each parent) are selected whose roots coincide with the symbol adopted or with a compatible symbol.

The crossover operator presents several configuration parameters. On the one hand, a list of eligible symbols can be defined in order to increase the probability of crossover for certain symbols and to decrease that probability for other symbols. In this case, all non-terminal symbols (excepting the root symbol) have the same probability of being selected. On the other hand, in order to reduce bloating, there is a parameter that defines the largest size possible for offspring generated in a crossover. If one of the new offspring surpasses this size, one of the two parents is randomly selected to pass, without modification, to the next generation. If both offspring surpass this size or at least one of them does not contain a symbol compatible with the chosen symbol, the crossover is aborted and both parents are reproduced, without modification, in the next generation. Fig. 2 shows how this operation is performed.

3.3.2. Mutation operator

Mutation is another of the most influential operators in the evolution process. This operator is thought to be responsible for preventing the loss of genetic diversity in the population, which is highly significant in the genetic convergence process. The mutation operator produces small random changes in an individual to engender a new offspring and continue the search process. This operator randomly selects the node in the tree where the mutation is to take place. If the node is a terminal node, it will be replaced by another compatible terminal symbol. More precisely, two nodes are compatible if they are derivations of the same non-terminal.

When the selected node is a non-terminal symbol, the subtree underneath this node will substitute any other valid derivation subtree as a result. For that reason, a new production of the grammar is derived for this non-terminal symbol. The

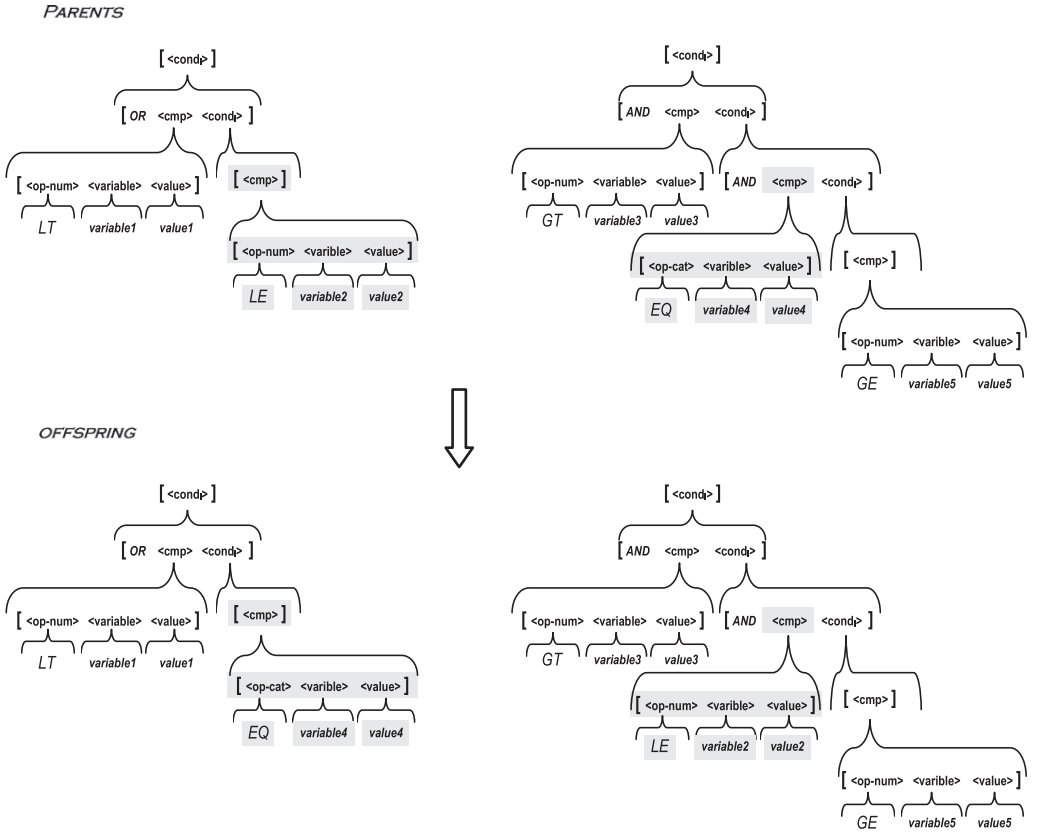


Fig. 2. Example of selective crossover.

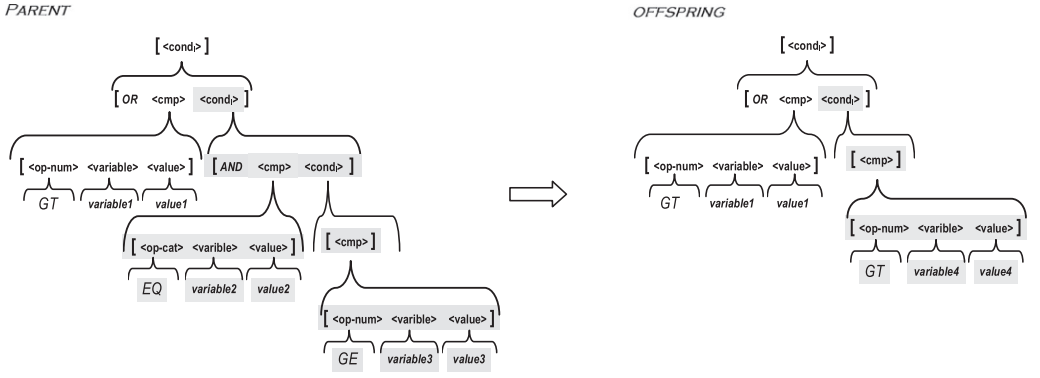


Fig. 3. Example of selective mutation.

procedure used to generate this subtree is the same as the one used to create new individuals and guarantees that the individual does not exceed a maximum size.

As in the case of selective crossover, this operator presents two configuration parameters: on the one hand, the list of eligible symbols as the root of the sub-tree to be mutated (in our case, all non-terminal symbols excepting the root symbol could be chosen with uniform probability) and on the other hand, the largest size which can have the offspring. Fig. 3 shows how this operation is performed.

3.4. Fitness function

The fitness function measures the effectiveness of the classifier. There are several measures to evaluate different components of the classifier and determine the quality of each rule. Our fitness function combines two commonly used indicators, namely sensitivity (Se) and specificity (Sp) [10,54]. Sensitivity is the proportion of cases correctly identified as meeting a certain condition and specificity is the proportion of cases correctly identified as not meeting a certain condition. When working

with MIL, it is necessary to modify a few basic concepts in classification-rule evaluation. Thus, *true positive* (t_p) represents the number of examples where the rule predicts that the bag has a given class and the bag does have that class; *false positive* (f_p) is the number of examples where the rule predicts that the bag has a given class but the bag does not have it; *true negative* (t_n) is the number of examples that the rule predicts where the bag does not have a given class, and indeed the bag does not have it; and finally, *false negative* (f_n) is the number of examples where the rule predicts that the bag does not have a given class but the bag does have it. With these definitions, sensitivity and specificity are explained as follows:

$$sensitivity = \frac{t_n}{t_n + f_p}, \tag{2}$$

$$specificity = \frac{t_p}{t_p + f_n}. \tag{3}$$

The goal of G3P-MI is to maximize both sensitivity and specificity at the same time. These two measurements evaluate different and conflicting characteristics in the classification process. Thus, a sensitivity of 100% means that the test recognizes all positive cases as such. Sensitivity alone does not tell us how well the test predicts other classes (that is, the negative cases). In binary classification, this is the corresponding specificity test, or equivalently, the sensitivity for other classes. A specificity of 100% means that the test recognizes all negative cases as such. Therefore, specificity alone does not clarify how well the test recognizes positive cases. We also need to know the sensitivity of the test to the class, or equivalently, the specificities to other classes. Our evaluation involves a simultaneous optimization of these two conflicting objectives where a value of 1 in both measurements represents perfect classification. We have tried to build a fitness function combining the two measurements, endeavoring to optimize them simultaneously. Of all the combinations tested, the product is that which produces the best results since it weighs both measurements equally, as well as penalizing those individuals with a value of 0 in either of the above measurements. Moreover, this fitness function has the advantages of being simple and returning a meaningful, normalized value in the range [0, 1].

$$fitness = sensitivity \cdot specificity. \tag{4}$$

3.5. Evolutionary algorithm

The main steps of our algorithm are based on a classical generational and elitist evolutionary algorithm. Initially, a population of classification rules is generated following the procedure described in Section 3.2. Once the individuals are evaluated with respect to their ability to solve the problem, the main loop of the algorithm is composed of the following operations. The first step represents the parent selection phase where individuals are selected by means of binary tournaments. Then, the recombination and mutation processes are carried out with a certain degree of probability. Once the offspring are obtained through the previous procedure, they are then evaluated. In the last operation, the population is updated by direct replacement, that is, the resulting offspring replace the current population. To guarantee that the best individual in the population is not lost during the updating process, the algorithm employs elitism.

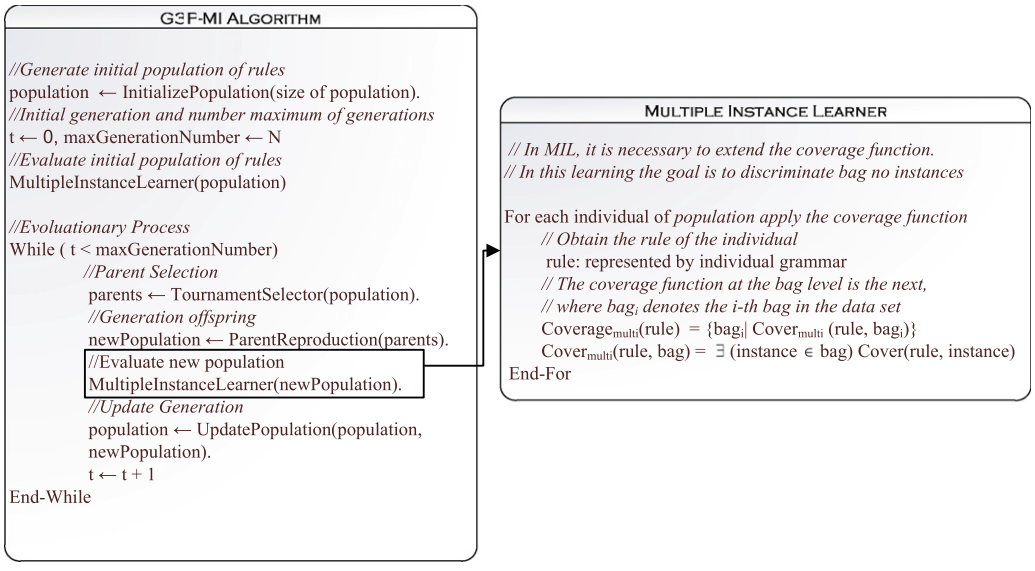


Fig. 4. G3P-MI algorithm.

Finally, there are two conditions for exiting from the main loop. On the one hand, the algorithm ends if the maximum number of generations defined by the user is reached and, on the other hand, it also ends if the best individual in the population achieves the quality objectives indicated by the user. The general outline of our algorithm is shown in Fig. 4. Once the algorithm has finished, the best individual (classification rule) obtained in the previous process is returned.

4. Experimental section

In this section, experiments have been carried out in various problem domains to evaluate the capabilities of the method proposed and compare it to other MIL methods. Results of 16 algorithms on 10 data sets are reported with different parameter configurations are reported. The datasets used in the experiments represent a range of applications and all data sets together with their partitions are available to facilitate future comparisons with previous or new proposals. The algorithms used for comparison are obtained from the WEKA data mining tool [63] and consider some of the most significant proposals in this research field.

First, the application domains and the algorithms used for comparison are presented, followed by a brief description of the experimental methodology. Finally, the groups of algorithms in the data sets are compared and the experimental results are discussed.

4.1. Problem domains used in experiments

Although MIL is a relatively recent learning framework, the MIL setting has produced numerous and interesting applications in different domains which have improved considerably the previous results achieved with other learning frameworks due to greater flexibility in their representation. In our experiments the ten data sets used consider drug activity prediction [40,80], content-based image retrieval [68,67,43] and the well-known East–West challenge [28]. There are brief overviews of the domains considered in our experiments, and of how multiple instance representation has been carried out in each case. Detailed information about these datasets is summarized in Table 1.

4.1.1. Drug activity prediction

The prediction of drug activity was what motivated this application for MIL representation. The problem consists of determining whether a drug molecule will bind strongly to a target protein. Each molecule may adopt a wide range of shapes or conformations. A positive molecule has at least one shape that can bind well (although it is not known which one) and a molecule is negative when none of its shapes can make the molecule bind well. This problem could be represented in a very natural way in a MIL setting: each molecule would be a bag and the conformations it can adopt would be the instances in that bag. Dietterich et al. [21] provided two datasets for the drug activity prediction problem, Musk1 and Musk2. For these two datasets, they showed that MIL approaches significantly outperform normal supervised-learning approaches which ignore the multi-instance nature of MIL problems. Later, these datasets have been extensively used as benchmarks to evaluate and compare MIL methods; for this reason they are considered in our experiments.

The mutagenecity of molecules is another drug-activity prediction problem included in our experiments [53]. The prediction of mutagenecity is considered a typical benchmark for first-order induction tools. The problem consists of predicting the mutagenicity of the molecules, that is, to determine whether a molecule is mutagenic or non-mutagenic. The dataset for mutagenesis [53] consists of 188 molecules, of which 125 are mutagenic (active) and 63 non-mutagenic (inactive). The MIL perspective deals with three different transformations: *mutagenesis-atoms* where a bag contains all the atoms of a compound molecule, *mutagenesis-bonds* where a bag contains all the atom-bond tuples of a compound molecule and *mutagenesis-chains* where a bag contains all the adjacent pairs in bonds of a compound molecule.

Table 1
General information about data sets.

Dataset	ID	Bags			Attributes	Instances	Average Bag Size
		Positive	Negative	Total			
Musk1	Mus1	47	45	92	166	476	5.17
Musk2	Mus2	39	63	102	166	6598	64.69
Mutagenesis-atoms	MutA	125	63	188	10	1618	8.61
Mutagenesis-bonds	MutB	125	63	188	16	3995	21.25
Mutagenesis-chains	MutC	125	63	188	24	5349	28.45
Elephant	ImgE	100	100	200	230	1391	6.96
Tiger	ImgT	100	100	200	230	1220	6.10
Fox	ImgF	100	100	200	230	1320	6.60
EastWest	EastW	10	10	20	24	213	10.65
WestEast	WestE	10	10	20	24	213	10.65

4.1.2. Content-based image retrieval and classification

Content-based image retrieval and classification make up another domain that uses MI representation. This problem involves identifying the intended target object(s) in images. The main difficulty is that one image may contain multiple, possibly heterogeneous objects. Thus, the global description of a whole image is too coarse to achieve good classification and retrieval accuracy. Even when relevant images are provided, it is still a thorny problem in the supervised learning setting to identify which object(s) in the sample image is/are relevant. However, MIL settings are suitable for this type of problem; each image can be treated as a bag of segments which are modeled as instances, and the concept point representing the target object can be learned through MIL algorithms. The experiments presented here consider data sets representing categories of animals [3], specifically elephants, foxes and tigers. Each data set consists of 100 images which contain the animal, and another 100 images which contain other animals. The final goal is to distinguish between the images that do contain the animal in question and those that do not.

4.1.3. East–West challenge

The well-known East–West Challenge [28] is originally an ILP problem inspired by a problem posed by Larson and Milchalski [34]. The problem involves predicting whether a train is going east or west. A train (bag) contains a variable number of cars (instances) that have different shapes and carry different loads (instance-level attributes). Since the standard MI assumption is asymmetric and it is not clear whether an eastbound train or a westbound train can be regarded as a positive example in the MI setting, two MI versions of the data are considered in our experiments. One is called East–West and contains eastbound trains as positive examples, and the other is called West–East and contains westbound trains as positive examples. A more detailed application can be found in [35].

4.2. Learning algorithms used for the experiments

The most relevant proposals presented to date are compared to determine whether our evolutionary algorithm is a more interesting proposal in some domains than other approaches. This section describes the algorithms used in our experiments.

4.2.1. Methods based on diverse density

Diverse density (DD), proposed by Maron and Lozano-Perez [40], is perhaps the best known framework for MI learning. Given a set of positive and negative bags, the idea behind this approach is to learn a concept that is *close* to at least one instance in each positive bag, but *far* from all instances in all the negative bags. Thus, the concept must describe a region of instance space that is not only *dense* in instances from positive bags, but also *diverse* in that it describes every positive bag. The experimentation considers three algorithms based on classical diverse-density algorithms for MI learning.

- *MIDD* is the standard DD algorithm that maximizes bag-level likelihood using the noisy-or model at training time. The class label of a new bag is assigned based on the class that receives maximum probability.
- *MIEMDD* is the expectation–maximization diverse density (EMDD) algorithm [75]. This algorithm combines the DD algorithm with an iterative approach inspired by the expectation–maximization algorithm (EM). In each iteration, the most-likely-cause model [40] is used to find the most likely target points based on the DD model that has been learned.
- *MDD* is a variant of the DD algorithm which computes bag-level class probability as the average of instance-level class probabilities [65]. This probability is used both at training time, when the likelihood is maximized based on the gradient descent, as well as at testing time, when a class label is assigned based on the class with maximum probability.

4.2.2. Methods based on logistic regression

Logistic regression is a popular machine learning method in standard single instance learning [47]. Our experiments consider a Multiple Instance Logistic Regression algorithm *MILR* which adapts standard SI logistic regression to MI learning assuming a standard single instance logistic model at the instance level and using its class probabilities to compute bag-level class probabilities using the noisy-or model employed by DD. Because instance-level class labels are not known, MILR learns the parameters of this MI logistic regression model by maximizing bag-level likelihood.

4.2.3. Methods based on support vector machines

The support vector machine (SVM) is a popular recent development within the machine learning and data mining communities. There is an extensive number of adaptations of this approach to the MIL framework [3,55,75] whose results show good performance in different applications. The experimentation involves the *MISMO* algorithm which replaces the SI kernel function with an MI kernel (i.e. an appropriate similarity function that is based on bags of instances). MISMO uses the SMO algorithm [44] for SVM learning in conjunction with an MI kernel [27].

4.2.4. Distance-based approaches

The *k*-nearest neighbor (k-NN) in a MIL framework was introduced by Wang and Zucker [58]. The main difference between the different proposals for nearest neighbor algorithms lies in the definition of the distance metrics used to measure the distance between bags. Two schemes extensively employed are minimal Hausdorff distance and the Kullback–Leibler distance. The experimentation considers:

Table 2
Parameter of G3P-MI algorithm.

Parameter	Value
Population size	1000
Number of generations	100
Crossover probability	95%
Mutation probability	30%
Elitist probability	5%
Parent selector	*Binary tournament*
Maximum tree depth	50

- *CitationKNN* which is a nearest-neighbor-based approach for MI problems proposed by Wang and Zucker [58]. It measures the distance between bags using Hausdorff distance. In contrast to standard nearest-neighbor learning, where only the nearest neighbors of an example to be classified are taken into account, the classification process with CitationKNN considers those examples in the training set where the example to be classified is the one nearest to both references and citers.
- *MIOptimalBall* is based on the optimal ball method [6] and implements the idea of classification based on the distance to a reference point. More specifically, this method attempts to find a *ball* in instance space so that all instances of all negative bags are outside the ball and at least one instance of each positive bag is inside the ball.

4.2.5. Methods based on supervised learning algorithms

The category based on supervised learning algorithms considers methods involving algorithms that adapt single instances to multiple instances. Three different methods are examined:

- *MIWrapper* is a method that simply assigns the class label of a bag to all its instances and then trains a single instance algorithm on the resulting data [25]. With respect to prediction, estimates of instance-level class probability for the instances in the bag to be classified are obtained from the single instance model; these predictions are then combined (based on the arithmetic or geometric average, or the maximum).
- *SimpleMI* is a method that computes the summary statistics for a bag to get a single instance from it. Depending on the option chosen, it computes the coordinate-wise geometric average, arithmetic average, or minimum and maximum (the latter strategy is also used by the minimax kernel for the MISMO discussed above).
- *Boosting* is a popular method for improving the performance of so-called weak learners [26]. MIBoost [66] is an algorithm inspired by AdaBoost that builds a series of weak classifiers using a single instance learner based on appropriately reweighted versions of the input data (all instances receive their bags' labels). This method considers the geometric mean of later instances inside a bag (arithmetic mean of log-posterior).

4.3. Experimental setting

The WEKA algorithms chosen to compare the effectiveness of our proposal are tested with some configurations accepted by this software and commented upon in Section 5.1. On the other hand, our proposal, G3P-MI, has been implemented in the JCLEC software [57] and its main parameters are shown in Table 2. All experiments are repeated with 10 different seeds and the average results are shown in the results table.

All algorithms are tested using 10-fold stratified cross validation [49,62] on all data sets. Folds are constructed on bags, so that every instance in a given bag appears in the same fold. The partitions of each data set are available at http:www.uco.es/grupos/kdis/mil.

5. Results and discussion

This section discusses our experimental results and compares our method to different algorithms. The first section is dedicated to contrasting different configurations of each algorithm. The combinations of the main parameters that produce the best results are obtained and this configuration is used for the global comparison of all the methods.

5.1. Results with other previously proposed algorithms

This first section of the experimental part studies different configurations of each algorithm considered. Thus, the most significant attributes are customized and finally, the configuration that produces the best results is selected. The optimized version of each algorithm is later used in the overall comparison of all the algorithms. In this way, the comparison is as equitable as possible.

In continuation there is a brief introduction of the variants of each algorithm evaluated. Among other configurations considered, it is important to highlight the use of different types of data preprocessing and various multi-instance scenarios. The

two main hypotheses called standard and collective assumption relate the class labels of the instances in a bag (unknown labels) with the class label of the bag. The former is defined only for two-class learning problems, with a positive and a negative class and it is corresponded with the definition of Dietterich et al. [21]. The underlying model is asymmetric because it is assumed that a bag is labeled positive if and only if it contains at least one positive instance, and negative otherwise. In contrast, in the latter model specified in [65], it is assumed that all individual instances within a bag contribute equally and independently to the bag's class label. All variations are evaluated in the 10-fold partitions previously mentioned. The Friedman test is used to compare different alternatives for each algorithm. The Friedman test [20] 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 are calculated for all data sets. These ranks reveal which algorithm obtains the best results out of all the data sets. In this way, the algorithm with the value closest to 1 indicates the best algorithm in most data sets. According to this test, if the null-hypothesis is accepted, all the algorithms are considered to be equivalent. On the other hand, if the null-hypothesis is rejected, there are significant differences between considered algorithms. Table 3 shows the configurations of each algorithm and the ranking of the accuracy, sensitivity and specificity values in each configuration considering the results on the ten datasets. The values for the Friedman test are shown in Table 4.

In continuation, the best configuration of each algorithm is discussed according to the values observed in the tables previously mentioned.

With respect to the MILR algorithm, two variants of collective assumptions are considered. The Friedman test determines that there are no significant differences among them. However, the best average values are obtained by the use of an arithmetic mean, which is why this configuration is selected. *With respect to the MIOptimalBall algorithm*, three variants with different filter types are considered for transforming the training data. The Friedman test determines that there are no significant differences among them, although the best average values are obtained by the use of normalized training data. *With respect to the MISMO algorithm*, four variants are considered with different combinations of kernels and according to whether the MIMinimax feature space is used or not. Again, the Friedman test determines that there are no significant differences between them. This shows that it is a reliable method independent of the kernel type and the use of minimax. Finally, the RBF kernel without minimax is found to be the option with the best values and therefore this configuration is selected. *With respect to the MIWrapper algorithm*, five significant learning methods are considered for resolving problems in single instance learning, including rule based systems (PART), support vector machine machines, two variants of ensembles and bayesian classifiers. Each method is evaluated by three different prediction methods, arithmetic and geometric averages and maximum instance-level class probabilities. In this case the best prediction method is selected for each learning system. According to the Friedman test, there are significant differences between different prediction methods. Normally, the best methods are arithmetic or geometric averages.

With respect to the MISimple algorithm, the effect of using different single instance learners as base learners is evaluated, including the consideration of AdaBoost and PART methods. Each method is evaluated with three different bag summarization methods using the arithmetic, geometric or minimum and maximum per-bag mean of each attribute. The Friedman test determines that the algorithms do not present significant differences. This means that transformed multi-instance methods are independent. However, the arithmetic average obtains the best results in the two learning methods which is why it is selected. *With respect to the MIBoost algorithm*, the four variants considered have different combinations of base learners and a maximum number of boost iterations. The Friedman test determines that this algorithm is not related to either the classifier or to the increase in the number of iterations. To select one variant, ten is the number of generations selected along with the use of the tree-based classifier because this combination achieves the best average values for all the measurements.

5.2. Comparison of results

This section compares our method to the best configurations of algorithms described in the previous section. Table 5 reports on the average results of accuracy, sensitivity and specificity for all the algorithms in each data set. The Friedman test [20] is done in order to more precisely evaluate the differences between the algorithms. The ranking of each algorithm used by the Friedman test is shown in Table 6.

The best values for accuracy, at a value of 2.20, are obtained by our method while the next best algorithm values show a much greater value. Concretely, the second algorithm that is based on the wrapper method, Bagging & PART, has a value of 5.25. These results confirm that our results are significantly better on average for all the data sets.

The results of Friedman test application are reported in Table 7. The Friedman test indicates that these results are significantly different both in accuracy and sensitivity measurements which prompted the performance of a post hoc test, the Bonferroni–Dunn test [23], to find any significant differences occurring between algorithms. This post-test is not necessary in the case of specificity because the Friedman test directly determines that there are no significant differences between the methods. Fig. 5a shows the application of this test on accuracy. This graphic represents a bar chart, whose values are proportional to the mean rank obtained from each algorithm. If the lowest of these (the best algorithm) is added to the critical difference value (CD value) of this test, a thicker horizontal line is obtained (denoted as *Threshold*) and those values that exceed this line are algorithms with significantly worse results than the control algorithm (in this case the control algorithm is associated to our proposal G3P-MI because it obtains the lowest ranking value). The threshold in this case is fitted to 9.445. Observing this figure, the algorithms that do not exceed the threshold determined by Bonferroni are G3P-MI, some versions of wrapper and

Table 3
Experimental results of the configurations of each algorithm.

Algorithms based on logistic regression					
Algorithm	Configuration characteristics		Ranking[1]		
	MI Assumptions	Mean type for the posteriors	Acc	Se	Sp
MILR (*Type I*)	Collective Assumption	Geometric Mean	1.7499	1.4000	1.7000
MILR (*Type II*) •	Collective Assumption	Arithmetic Mean	1.2500	1.6000	1.3000
Algorithms based on distance					
Algorithm	Configuration characteristics		Ranking[1]		
	Filter type		Acc	Se	Sp
MIOptimalBall (*Type I*) •	Implements no normalization/standardization		1.7500	1.9500	2.0000
MIOptimalBall (*Type II*)	Implements standardize training data		1.9000	2.1000	1.9500
MIOptimalBall (*Type III*)	Implements normalized training data		2.3500	1.9500	2.0500
Algorithms based on support vector machines					
Algorithm	Configuration characteristics		Ranking[1]		
	Kernel	Minimax for feature space	Acc	Se	Sp
MISMO (*Type I*) •	RBF Kernel	Do not make use of MIMinimax feature space	1.7500	2.1999	2.1500
MISMO (*Type II*)	Polynomial Kernel	Do not make use of MIMinimax feature space	2.5000	1.7500	3.2500
MISMO (*Type III*)	RBF Kernel	Make use of MIMinimax feature space	3.3500	3.2500	2.4500
MISMO (*Type IV*)	Polynomial Kernell	Make use of MIMinimax feature space	2.4000	2.8000	2.1500
Algorithms based on supervised learning (Wrapper)					
Algorithm	Configuration characteristics		Ranking[1]		
	Classifier	Method for testing	Acc	Se	Sp
PART (*Type I*)	PART learning	Arithmetic Average of instance-level class probabilities	1.7000	1.4999	2.6500
PART (*Type II*) •	PART learning	Geometric Average of instance-level class probabilities	1.4000	1.5000	2.3500
PART (*Type III*)	PART learning	Maximum probability of positive bag	2.9000	2.9999	0.9999
Bagging& PART (*Type I*)	Bagging with PART learning	Arithmetic Average of instance-level class probabilities	1.5000	1.5499	2.5000
Bagging& PART (*Type II*) •	Bagging with PART learning	Geometric Average of instance-level class probabilities	1.5000	1.4500	2.5000
Bagging & PART (*Type III*)	Bagging with PART learning	Maximum probability of positive bag	2.9999	2.9999	0.9999
AdaBoost & PART (*Type I*) •	AdaBoost with PART learning	Arithmetic Average of instance-level class probabilities	1.3499	1.4500	2.5000
AdaBoost & PART (*Type II*)	AdaBoost with PART learning	Geometric Average of instance-level class probabilities	1.7000	1.5499	2.5000
AdaBoost & PART (*Type III*)	AdaBoost with PART learning	Maximum probability of positive bag	2.9499	2.9999	0.9999
SMO (*Type I*) •	SMO learning with PolyKernel kernel	Arithmetic Average of instance-level class probabilities	1.6500	1.4500	2.5000
SMO (*Type II*)	SMO learning with PolyKernel kernel	Geometric Average of instance-level class probabilities	1.7999	1.7499	2.2000
SMO (*Type III*)	SMO learning with PolyKernel kernel	Maximum probability of positive bag	2.5500	2.8000	1.3000
NaiveBayes (*Type I*)	NaiveBayes learning	Arithmetic Average of instance-level class probabilities	1.9500	1.7000	2.2000
NaiveBayes (*Type II*) •	NaiveBayes learning	Geometric Average of instance-level class probabilities	1.9500	1.4000	2.6999
NaiveBayes (*Type III*)	NaiveBayes learning	Maximum probability of positive bag	2.1000	2.8999	1.0999
Algorithms based on supervised learning (Simple)					
Algorithm	Configuration characteristics		Ranking[1]		
	Classifier	Transform method	Acc	Se	Sp
AdaBoost & PART (*Type I*) •	AdaBoost.M1 with PART learning	Arithmetic per-bag mean of each attribute	1.7500	1.7000	1.9000
AdaBoost & PART (*Type II*)	AdaBoost.M1 with PART learning	Geometric per-bag mean of each attribute	2.5500	2.2500	2.3000
AdaBoost & PART (*Type III*)	AdaBoost.M1 with PART learning	Per-bag minimum and maximum of each attribute	1.7000	2.0500	1.8000
PART (*Type I*) •	PART learning	Arithmetic per-bag mean of each attribute	1.7000	1.8500	1.7500
PART (*Type II*)	PART learning	Geometric per-bag mean of each attribute	2.4500	2.1999	2.2500
PART (*Type III*)	PART learning	Per-bag minimum and maximum of each attribute	1.8499	1.9499	2.0000
Algorithms based on supervised learning (Boost)					
Algorithm	Configuration characteristics		Ranking[1]		
	Classifier	Maximum number of iterations	Acc	Se	Sp
MIBoost (*Type I*)	RepTree learning	Fifty iterations of boost	2.0500	2.5500	2.5000
MIBoost (*Type II*) •	RepTree learning	Ten iterations of boost	2.0000	2.1500	2.8000
MIBoost (*Type III*)	DecisionStump learning	Fifty iterations of boost	2.6500	2.4000	2.3000
MIBoost (*Type IV*)	DecisionStump learning	Ten iterations of boost	3.3000	2.9000	2.4000

[1] Ranking obtaining by the algorithm for executions over all ten data sets.

simple algorithms, the support vector machine (MISMO), a method based on boosting (MIBoost) and diverse density (MIDD). The Bonferroni–Dunn test results consider that the rest of the algorithms are enough far away for there to be significant differences among them. In this study, our proposal is seen to obtain the best results in different data sets. The closest algorithms are the wrapper methods based on different learners.

Table 4
Results of Friedman test ($p = 0.01$).

Algorithm		Friedman value	χ^2 Value	Conclusion
MILR	Acc	2.4999	6.6349	Accept null hypothesis
	Se	0.4000	6.6349	Accept null hypothesis
	Sp	1.6000	6.6349	Accept null hypothesis
MIOptimalBall	Acc	1.9500	9.2104	Accept null hypothesis
	Se	0.1500	9.2104	Accept null hypothesis
	Sp	0.0499	9.2104	Accept null hypothesis
MISMO	Acc	7.7700	11.3449	Accept null hypothesis
	Se	7.8299	11.3449	Accept null hypothesis
	Sp	4.8599	11.3449	Accept null hypothesis
PART (Wrapper)	Acc	12.5999	9.2104	Reject null hypothesis
	Se	15.0499	9.2104	Reject null hypothesis
	Sp	15.4500	9.2104	Reject null hypothesis
Bagging & PART (Wrapper)	Acc	14.9999	9.2104	Reject null hypothesis
	Se	15.0499	9.2104	Reject null hypothesis
	Sp	15.0000	9.2104	Reject null hypothesis
AdaBoost & PART (Wrapper)	Acc	14.1499	9.2104	Reject null hypothesis
	Se	15.0499	9.2104	Reject null hypothesis
	Sp	15.0000	9.2104	Reject null hypothesis
SMO (Wrapper)	Acc	4.6499	9.2104	Accept null hypothesis
	Se	10.049	9.2104	Reject null hypothesis
	Sp	7.8000	9.2104	Accept null hypothesis
NaiveBayes (Wrapper)	Acc	0.1500	9.2104	Accept null hypothesis
	Se	12.5999	9.2104	Reject null hypothesis
	Sp	13.3999	9.2104	Reject null hypothesis
AdaBoost & PART (Simple)	Acc	3.1500	9.2104	Accept null hypothesis
	Se	0.6499	9.2104	Accept null hypothesis
	Sp	1.2500	9.2104	Accept null hypothesis
PART (Simple)	Acc	4.5500	9.2104	Accept null hypothesis
	Se	1.5500	9.2104	Accept null hypothesis
	Sp	1.4000	9.2104	Accept null hypothesis
MIBoost	Acc	6.6900	11.3449	Accept null hypothesis
	Se	1.7700	11.3449	Accept null hypothesis
	Sp	0.8400	11.3449	Accept null hypothesis

Fig. 5b shows the application of the Bonferroni–Dunn posthoc test with respect to the measurement for sensitivity. The threshold in this case is 10.345. Observing this figure, the algorithms that do not exceed the threshold determined by Bonferroni are mainly any wrapper or simple versions, algorithms based on distance (CitationkNN and OptimallBall), MIBoost and diverse density (MIDD, MDD). The Bonferroni–Dunn test results consider the rest of algorithms to be too far away to show significant differences among them. In sensitivity values, again our proposal obtains the best results and there is a considerable difference with respect to the second proposal. We can conclude that our proposal obtains the best values in different measurements, which does not occur with the rest of the techniques previously mentioned. Although some of them considered that there were no significant differences with respect to the control algorithm in accuracy, clearly these algorithms do yield worse results in sensitivity than the control algorithm. This simply highlights that our proposal is a reliable technique, as it is the classifier with the best guarantee for determining correct classifications in any class.

5.3. Comprehensibility in the knowledge discovery process

As previously mentioned, our system generates a learner based on IF-THEN prediction rules which provide a natural extension to knowledge representation. The procedure is made automatically, the input of the algorithm being the data set and other configuration parameters of the algorithm (shown in Table 2) while the output is the best rule achieved in the evolutionary process, whose representation is specified in Section 3.1.

The rules obtained are characterized by several desirable properties: they are simple, intuitive, easy to understand and provide representative information. These qualities represent an advantage with respect to other algorithms that work as black boxes, which provide an incomprehensible output. Thus, our system adds comprehensibility and clarity to the knowledge discovery process, making it possible to classify new patterns quickly by means of the classifier generated. This can be seen in the following examples where one example of each application domain has been selected. These examples show that the rules allow an expert to understand which attributes are relevant and their intervals.

Table 5
Experimental results.

Datasets		MutA	MutB	MutC	Mus1	Mus2	ImgF	ImgT	ImgE	EastW	WestE
MISMO	Acc	0.7000	0.8421	0.7842	0.8778	0.8400	0.5800	0.8250	0.7900	0.7500	0.6500
	Se	0.3833	0.8833	0.7167	0.8250	0.8667	0.4100	0.7800	0.7500	0.7000	0.6000
	Sp	0.8462	0.8231	0.8154	0.9200	0.8000	0.7500	0.8700	0.8300	0.8000	0.7000
MIOptimalBall	Acc	0.7263	0.7421	0.7158	0.8000	0.7700	0.5300	0.6700	0.7750	0.7000	0.3000
	Se	0.5500	0.6833	0.4500	0.9250	0.8333	0.7400	0.6500	0.8800	0.7000	0.3000
	Sp	0.8077	0.7692	0.8385	0.7000	0.8050	0.3200	0.6900	0.6700	0.7000	0.3000
MILR	Acc	0.7000	0.7105	0.7684	0.8778	0.8500	0.5600	0.7700	0.7800	0.6000	0.6000
	Se	0.2000	0.2167	0.4333	0.8500	0.9167	0.4900	0.7400	0.6900	0.5000	0.7000
	Sp	0.9308	0.9385	0.9231	0.9000	0.7500	0.6300	0.8000	0.8700	0.7000	0.5000
PART (Wrapper)	Acc	0.8105	0.7895	0.8684	0.8444	0.8500	0.6500	0.8150	0.8050	0.5000	0.5500
	Se	0.5667	0.7000	0.7500	0.8750	0.8833	0.6100	0.8300	0.7300	0.5000	0.7000
	Sp	0.9231	0.8308	0.9231	0.8200	0.8000	0.6900	0.8000	0.8800	0.5000	0.4000
Bagging & PART (Wrapper)	Acc	0.8105	0.7789	0.8684	0.9000	0.8500	0.6900	0.8100	0.8600	0.6000	0.5500
	Se	0.5833	0.7167	0.7833	0.8500	0.8667	0.6400	0.8000	0.7900	0.5000	0.6000
	Sp	0.9154	0.8077	0.9077	0.9400	0.8250	0.7400	0.8200	0.9300	0.7000	0.5000
AdaBoost & PART (Wrapper)	Acc	0.8053	0.7737	0.8737	0.8778	0.9000	0.6900	0.8000	0.8400	0.5000	0.6000
	Se	0.5333	0.7000	0.7667	0.8750	0.9167	0.6100	0.8000	0.7800	0.4000	0.6000
	Sp	0.9308	0.8077	0.9231	0.8800	0.8750	0.7700	0.8000	0.9000	0.6000	0.6000
SMO (Wrapper)	Acc	0.6842	0.6842	0.6895	0.8444	0.8900	0.6350	0.8000	0.8550	0.5500	0.5000
	Se	0.0000	0.0000	0.0167	0.8250	0.9333	0.4800	0.7700	0.8000	0.6000	0.8000
	Sp	1.0000	1.0000	1.0000	0.8600	0.8250	0.7900	0.8300	0.9100	0.5000	0.2000
NaiveBayes (Wrapper)	Acc	0.6842	0.6263	0.5263	0.8000	0.7700	0.5900	0.7650	0.8400	0.5000	0.6000
	Se	0.0000	0.1500	0.9667	0.8000	0.7667	0.2700	0.8800	0.7700	0.5000	0.6000
	Sp	1.0000	0.8462	0.3231	0.8000	0.7750	0.9100	0.6500	0.9100	0.5000	0.6000
PART (Simple)	Acc	0.7000	0.8263	0.7737	0.8111	0.7400	0.5800	0.7650	0.7800	1.0000	1.0000
	Se	0.4833	0.8333	0.5833	0.8000	0.7833	0.6200	0.8100	0.7600	1.0000	1.0000
	Sp	0.8000	0.8231	0.8615	0.8200	0.6750	0.5400	0.7200	0.8000	1.0000	1.0000
AdaBoost & PART (Simple)	Acc	0.7526	0.8474	0.7947	0.8333	0.7900	0.6250	0.7950	0.8100	0.9500	0.8500
	Se	0.5167	0.8333	0.6667	0.8250	0.8500	0.6200	0.8100	0.8200	0.9000	0.8000
	Sp	0.8615	0.8538	0.8538	0.8400	0.7000	0.6300	0.7800	0.8000	1.0000	0.9000
MIBoost	Acc	0.8316	0.7737	0.8474	0.8778	0.8700	0.6950	0.8200	0.8150	0.6000	0.5000
	Se	0.7333	0.7333	0.7333	0.8000	0.9000	0.6300	0.8000	0.7400	0.5000	0.6000
	Sp	0.8769	0.7923	0.9000	0.9400	0.8250	0.7600	0.8400	0.8900	0.7000	0.4000
MIEMDD	Acc	0.7316	0.7211	0.7368	0.8889	0.9000	0.5900	0.7450	0.7300	0.4500	0.3000
	Se	0.5000	0.4333	0.5000	0.8750	0.8833	0.3900	0.7100	0.7100	0.7000	0.3000
	Sp	0.8385	0.8538	0.8462	0.9000	0.9250	0.7900	0.7800	0.7500	0.2000	0.3000
MIDD	Acc	0.7263	0.7526	0.7684	0.9222	0.7300	0.6550	0.7400	0.8300	0.5000	0.2500
	Se	0.4167	0.5593	0.5500	0.9500	0.9500	0.6200	0.7200	0.8300	0.7000	0.3000
	Sp	0.8692	0.8615	0.8692	0.9000	0.4000	0.6900	0.7600	0.8300	0.3000	0.2000
MDD	Acc	0.7211	0.7316	0.7684	0.7889	0.7400	0.7000	0.7550	0.8000	0.5500	0.5500
	Se	0.1667	0.4000	0.5500	0.7750	0.8500	0.6300	0.7700	0.8200	0.8000	0.7000
	Sp	0.9769	0.8846	0.8692	0.8000	0.5750	0.7700	0.7400	0.7800	0.3000	0.4000
CitationKNN	Acc	0.7526	0.7316	0.7474	0.9444	0.8500	0.5000	0.5000	0.5000	0.5500	0.5500
	Se	0.5833	0.4333	0.5167	0.9750	0.8667	1.0000	1.0000	1.0000	0.5000	0.7000
	Sp	0.8308	0.8692	0.8538	0.9200	0.8250	0.0000	0.0000	0.0000	0.6000	0.4000
G3P-MI	Acc	0.8526	0.8210	0.8105	0.9445	0.8800	0.7050	0.8700	0.8800	1.0000	0.8500
	Se	0.8462	0.8462	0.9231	1.0000	0.7500	0.7900	0.9400	0.9300	1.0000	0.7000
	Sp	0.8167	0.7833	0.7333	0.9000	0.9180	0.6200	0.8000	0.8300	1.0000	1.0000

IF $[(F11 \leqslant -17.233) \wedge (F148 \leqslant -17.665) \wedge (F107 \leqslant -67.946) \wedge (F26 > -21.367) \wedge (F97 \leqslant 3.100) \vee$
$[(-17.665 < F148 \leqslant 5.884) \wedge (F91 \leqslant 43.487) \wedge (F29 > -120.881) \wedge (F82 > -56.242) \wedge (F97 \leqslant -7.919)]$
THEN *The molecule is musky.*
ELSE *The molecule is not musky.*

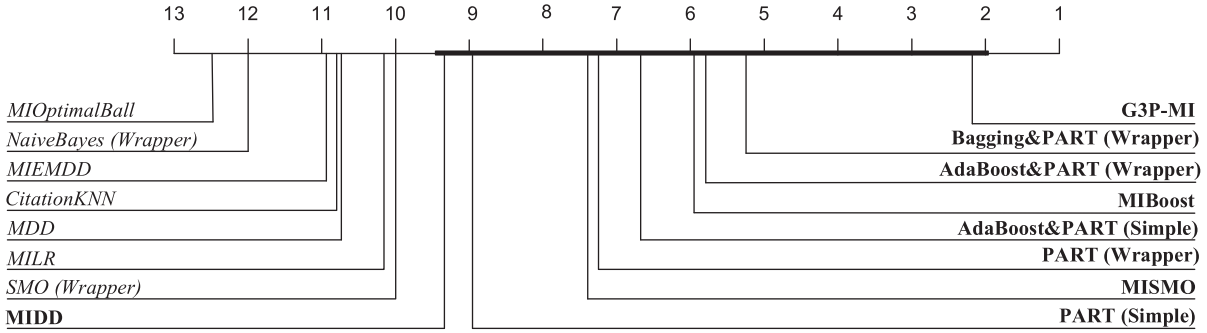
The previous rule is obtained for the Musk data set. This rule shows representative information about the problem of predicting if a molecule has the musky property. The attributes considered in the rule maintain information about the shape of the moleculewhich is why the distance of 162 rays emanating from the origin of the molecule to the molecule surface is considered. In addition to these 162 shape features, the four domain-specific features also considered represent the position of a designated atom (an oxygen atom) on the molecular surface. According to this initial information, the rule provides

Table 6
Average rankings of the algorithms.

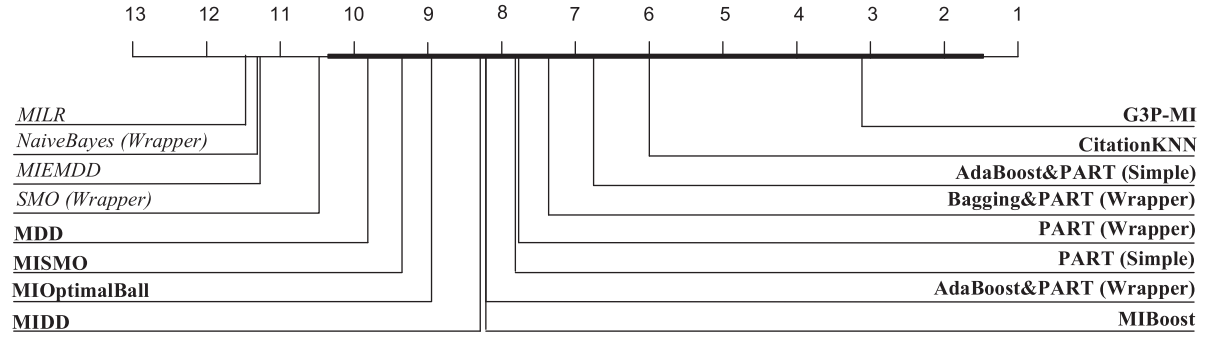
Algorithm	Ranking		
	Acc	Se	Sp
MISMO	7.40	9.35	7.35
MIOptimalBall I	12.50	8.95	13.20
MILR	10.15	11.45	6.70
MIEMDD	10.95	11.25	9.35
MIDD	9.35	8.30	10.35
MDD	10.75	9.80	9.75
CitationKNN	10.80	6.00	10.45
G3PMI	2.20	3.10	8.45
PART (Wrapper)	7.25	7.75	8.45
Bagging & PART (Wrapper)	5.25	7.35	5.85
AdaBoost & PART (Wrapper)	5.80	8.20	6.20
SMO (Wrapper)	10.00	10.45	5.45
NaiveBayes (Wrapper)	12.00	11.30	8.70
PART (Simple)	8.95	7.80	10.40
AdaBoost & PART (Simple)	6.70	6.75	8.85
MIBoost	5.95	8.20	6.50

Table 7
Global results of Friedman test ($p = 0.01$).

	Friedman value	χ_2 Value	Conclusion
Acc	53.1882	30.5780	Reject null hypothesis
Se	31.5838	30.5780	Reject null hypothesis
Sp	28.9213	30.5780	Accept null hypothesis



(a) Bonferroni-Dunn for Accuracy



(b) Bonferroni-Dunn for Sensitivity

Fig. 5. Bonferroni–Dunn test ($p < 0.01$).

information about the geometric structure of the molecule to determine if it satisfies the property to be evaluated or not. Thus, experts in the area can increase their knowledge about identifying or developing molecules with this property.

The next rule represents the most relevant attributes for identifying a tiger inside an image. In this problem, the information about the image includes features of color, texture and the position of pixels in the image. Concretely, with respect to color features, the information maintained is about Lab space color. With respect to texture features, contrast, anisotropy and polarity are taken into consideration; with respect to position features are considered the coordinates (x,y) of the position of the pixel. According to this initial information, the rule provides information about which descriptors of color, texture and position should have a region of an image to contain a tiger. Experts in the area can obtain the main characteristics that an image has to have in order to be able to recognize a tiger.

IF [*atr-14* > 3.142] ∨ [(*atr-1* > 1.048) ∧ (*atr-9* > 1.431)] ∨ [(*atr-96* > 0.496) ∧ ((*atr-1* > 0.909) ∨ (*atr-32* > 0.263)))] ∨
[[(*atr-63* > 0.425) ∨ ((*atr-1* > 0.909) ∧ (*atr-14* > 3.229))] ∨ (*atr-32* > 0.263) ∨ (*atr-147* > 2.952)]
THEN *The image contains a tiger.*
ELSE *The image does not contain a tiger.*

Finally, the last application domain is the East–West problem. In this case the information available concerns the physical description of trains. Thus, the rule provides information about the size, charge and type of charge that will identify if a train is eastbound or westbound.

IF [*charge* > − 0.056 ∧ *typeh* ⩽ 0.594 ∧ *typeo* > 0.719] ∨ [(*typen* ⩽ 0.0713 ∧ *charge* > 0.606] ∨
[(0.827 ⩾ *charge* > 0.807)] ∨ [*quantatype* > 134.146)]
THEN *It is east west.*
ELSE *It is not east west.*

6. Conclusion

This paper presents G3P-MI, a grammar guided genetic programming algorithm for multiple-instance learning. Our proposal is compared to the most important techniques previously designed for solving these problems, considering both the classical algorithms designed exclusively for this learning (DD algorithm and its variants) as well as adaptations of machine learning algorithms (among others, we consider algorithms based on distance, support vector machines, ensembles, logistic regression). All the proposals are tested in ten data sets which contemplate the diversity of domains where MIL has been applied to date. Empirical studies on these characteristics are the greatest weak point of MIL, as most studies to date have empirically compared only a few approaches and used very few data sets. Moreover, as all the information about data sets and partitions is available, the results can be reproduced and any new study can demonstrate their efficacy and efficiency under the same conditions in the resolution of these problems. Experimental results show that our algorithm obtains better results than other algorithms in several applications with respect to accuracy, sensitivity and specificity. The Friedman test confirms these results determining that there are significant differences between the methods with respect to sensitivity and accuracy. In the case of specificity, this test cannot determine real differences between results. The post-test carried out to measure the differences presented, the Bonferroni–Dunn test, proves our proposal to be the best for both accuracy and sensitivity. The other techniques are found to be worse in either one measurement or the other, and so they do not classify one of the classes correctly. As well as the outstanding results obtained by our proposal, another advantage is that it generates a rule-based system. This system provides a natural extension for knowledge representation and is simple, intuitive, modular, and easy to generate from data.

Although the results obtained are of great interest, we feel that the yield of the G3P-MI algorithm in solving multi-instance problems could be improved in some ways. On the one hand it would be interesting to reduce the space dedicated to features, thus facilitating the search for optimal solutions. For this reason it would be of special interest to study the application of feature selection techniques in this learning. On the other hand, classification tasks deal with contrasted measurements, which means that as one is optimized, the other decreases its values; therefore it would be of interest to approach the solution from a multiobjective perspective to see if the results could be improved even further. Finally, it would be of interest to also include some factor in the evaluation to measure the simplicity of the rules in order to optimize this factor while maintaining the results.

Acknowledgment

The authors gratefully acknowledge the financial subsidy provided by the Spanish Department of Research under TIN2005-08386-C05-02 Project and TIN2008–06681-C06-03 Project.

References

[1] E. Alphonse, S. Matwin, Filtering multi-instance problems to reduce dimensionality in relational learning, Journal Intelligence Information System 22 (1) (2004) 23–40.
[2] R.A. Amar, D.R. Dooly, S.A. Goldman, Q. Zhang, Multiple-instance learning of real-valued data, in: ICML'01: Proceedings of 18th International Conference on Machine Learning, Williams College, Williamstown, MA, USA, 2001.

[3] S. Andrews, I. Tsochantaridis, T. Hofmann, Support vector machines for multiple-instance learning, in: NIPS'02: Proceedings of Neural Information Processing System, Vancouver, Canada, 2002.
[4] P. Auer, On learning from multi-instance examples: empirical evaluation of a theoretical approach, in: ICML'97: Proceedings of the 14th International Conference on Machine Learning, Nashville, Tennessee, USA, 1997.
[5] P. Auer, P.M. Long, A. Srinivasan, Approximating hyper-rectangles: learning and pseudo-random sets, in: STOC'97: Proceedings of the 29th Annual ACM Symposium on Theory of Computing, ACM, New York, NY, USA, 1997.
[6] P. Auer, R. Ortner, A boosting approach to multiple instance learning, in: ECML'04: Proceedings of the 5th European Conference on Machine Learning, Lecture Notes in Computer Science, vol. 3201, Pisa, Italy, 2004.
[7] T. Back, D. Fogel, Z. Michalewicz, Handbook of Evolutionary Computation, Oxford University Press, Oxford, UK, 1997.
[8] A. Blum, A. Kalai, A note on learning from multiple-instance examples, Machine Learning 30 (1) (1998) 23–30.
[9] W. Böhm, A. Geyer-Schulz, Exact uniform initialization for genetic programming, in: FOGA'96: Proceedings of the 4th Workshop on Foundations of Genetic Algorithms, Morgan Kaufmann, San Diego, CA, USA, 1996.
[10] C.C. Bojarczuk, H.S. Lopes, A.A. Freitas, Genetic programming for knowledge discovery in chest-pain diagnosis, IEEE Engineering in Medicine and Biology Magazine 19 (4) (2000) 38–44.
[11] Y.-M. Chai, Z.-W. Yang, A multi-instance learning algorithm based on normalized radial basis function network, in: ISSN'07: Proceedings of the 4th International Symposium on Neural Networks, Lecture Notes in Computer Science, vol. 4491, Nanjing, China, 2007.
[12] K. Chellapilla, Evolving computer programs without subtree crossover, IEEE Transactions on Evolutionary Computation 1 (3) (1997) 209–216.
[13] X. Chen, C. Zhang, S. Chen, S. Rubin, A human-centered multiple instance learning framework for semantic video retrieval, IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews 39 (2) (2009) 228–233.
[14] Y. Chen, J. Bi, J. Wang, Miles: multiple-instance learning via embedded instance selection, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (12) (2006) 1931–1947.
[15] Y. Chen, J.Z. Wang, Image categorization by learning and reasoning with regions, Journal of Machine Learning Research 5 (2004) 913–939.
[16] Y. Chevaleyre, N. Bredeche, J. Zucker, Learning rules from multiple instance data: issues and algorithms, in: IPMU'02: Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Annecy, France, 2002.
[17] Y.-Z. Chevaleyre, J.-D. Zucker, Solving multiple-instance and multiple-part learning problems with decision trees and decision rules. Application to the mutagenesis problem, in: AI'01: Proceedings of the 14th of the Canadian Society for Computational Studies of Intelligence, Lecture Notes in Computer Science, vol. 2056, Ottawa, Canada, 2001.
[18] B.-C. Chien, J.Y. Lin, T.-P. Hong, Learning discriminant functions with fuzzy attributes for classification using genetic programming, Expert Systems with Applications 23 (1) (2002) 31–37.
[19] R.A. Davis, A.J. Charlton, S. Oehlschlager, J.C. Wilson, Novel feature selection method for genetic programming using metabolomic 1H NMR data, Chemometrics and Intelligent Laboratory Systems 81 (1) (2006) 50–59.
[20] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[21] T.G. Dietterich, R.H. Lathrop, T. Lozano-Perez, Solving the multiple instance problem with axis-parallel rectangles, Artifical Intelligence 89 (1–2) (1997) 31–71.
[22] D.R. Dooly, Q. Zhang, S.A. Goldman, R.A. Amar, Multiple instance learning of real valued data, Journal Machine Learning Research 3 (2003) 651–678.
[23] O.J. Dunn, Multiple comparisons among means, Journal of the American Statistical Association 56 (293) (1961) 52–64.
[24] S. Feng, D. Xu, Transductive multi-instance multi-label learning algorithm with application to automatic image annotation, Expert Systems with Applications 37 (1) (2010) 661–670.
[25] E. Frank, X. Xu, Applying propositional learning algorithms to multi-instance data, Tech. rep., Department of Computer Science, University of Waikato, 2003.
[26] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: ICML'96: Proceedings of the 13th International Conference on Machine Learning, Garda, Italy, 1996.
[27] T. Gärtner, P.A. Flach, A. Kowalczyk, A.J. Smola, Multi-instance kernels, in: ICML'02: Proceedings of the 19th International Conference on Machine Learning, Morgan Kaufmann, Sydney, Australia, 2002.
[28] T. Gärtner, J.W. Lloyd, P.A. Flach, Kernels and distances for structured data, Machine Learning 57 (3) (2004) 205–232.
[29] A. Geyer-Schulz, Fuzzy Rule-Based Expert Systems and Genetic Machine Learning, first ed., Physica Verlag, Heidelberg, Germany, 1995.
[30] Z. Gu, T. Mei, J. Tang, X. Wu, X. Hua, MILC2: A multi-layer multi-instance learning approach to video concept detection, in: MMM'08: Proceedings of the 14th International Conference of Multimedia Modeling, Kyoto, Japan, 2008.
[31] J.K. Kishore, L.M. Patnaik, V. Mani, V.K. Agrawal, Application of genetic programming for multicategory pattern classification, IEEE Transactions on Evolutionary Computation 4 (3) (2000) 242–258.
[32] P. Kouchakpour, A. Zaknich, T. Brn++unl, Dynamic population variation in genetic programming, Information Sciences 179 (8) (2009) 1078–1091.
[33] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.
[34] J. Larson, R.S. Michalski, Inductive inference of vl decision rules, SIGART Newsletter (63) (1977) 38–44.
[35] N. Lavrač, P.A. Flach, An extended transformation approach to inductive logic programming, ACM Transactions on Computation Logic 2 (4) (2001) 458–494.
[36] T. Lensberg, A. Eilifsen, T.E. McKee, Bankruptcy theory development and classification via genetic programming, European Journal of Operational Research 169 (2) (2006) 677–697.
[37] P.M. Long, L. Tan, PAC learning axis-aligned rectangles with respect to product distributions from multiple-instance examples, Machine Learning 30 (1) (1998) 7–21.
[38] S. Luke, L. Panait, A survey and comparison of tree generation algorithms, in: GECCO'01: Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, California, 2001.
[39] O.L. Mangasarian, E.W. Wild, Multiple instance classification via successive linear programming, Journal of Optimization Theory and Applications 137 (3) (2008) 555–568.
[40] O. Maron, T. Lozano-Pérez, A framework for multiple-instance learning, in: NIPS'97: Proceedings of Neural Information Processing System 10, Denver, Colorado, USA, 1997.
[41] M. Muharram, Evolutionary constructive induction, IEEE Transactions on Knowledge and Data Engineering 17 (11) (2005) 1518–1528.
[42] D.P. Muni, N.R. Pal, J. Das, A novel approach to design classifiers using genetic programming, IEEE Transactions on Evolutionary Computation 8 (2) (2004) 183–196.
[43] H.T. Pao, S.C. Chuang, Y.Y. Xu, H. Fu, An EM based multiple instance learning method for image classification, Expert Systems with Applications 35 (3) (2008) 1468–1472.
[44] J.C. Platt, Fast training of support vector machines using sequential minimal optimization, Advances in Kernel Methods: Support Vector Learning (1999) 185–208.
[45] L.D. Raedt, Attribute-value learning versus inductive logic programming: The missing links (extended abstract), in: ILP '98: Proceedings of the 8th International Workshop on Inductive Logic Programming, Springer-Verlag, London, UK, 1998.
[46] J. Ramon, L. De Raedt, Multi-instance neural networks, in: ICML'00: A Workshop on Attribute-Value and Relational Learning at the 17th Conference on Machine Learning, 2000.
[47] S. Ray, M. Craven, Supervised versus multiple instance learning: an empirical comparison, in: ICML'05: Proceedings of the 22nd International Conference on Machine Learning, ACM Press, Bonn, Germany, 2005.

[48] S. Ray, D. Page, Multiple instance regression, in: ICML'01: Proceedings of the 18th International Conference on Machine Learning, Williams College, Williamstown, MA, USA, 2001.
[49] K. Ron, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: IJCAI'95: International Joint Conference on Artificial Intelligence, Montreal, Canada, 1995.
[50] G. Ruffo, Learning Single and Multiple Instance Decision Tree for Computer Security Applications, Ph.D. Thesis, Department of Computer Science, University of Turin, Torino, Italy, 2000.
[51] S. Scott, J. Zhang, J. Brown, On generalized multiple-instance learning, International Journal of Computational Intelligence and Applications 5 (2005) 21–35.
[52] A. Song, V. Ciesielski, H. Williams, Texture classifiers generated by genetic programming, in: CEC'02: Proceedings of Congress on Evolutionary Computation, Honolulu, HI, USA, 2002.
[53] A. Srinivasan, S.H. Muggleton, M.J.E. Sternberg, R.D. King, Theories for mutagenicity: a study in first-order and feature-based induction, Artificial Intelligence 85 (1-2) (1996) 277–299.
[54] K. Tan, A. Tay, T. Lee, C. Heng, Mining multiple comprehensible classification rules using genetic programming, in: CEC'02: Proceedings of the Congress on Evolutionary Computation, vol. 2, Honolulu, HI, USA, 2002.
[55] Q. Tao, S. Scott, N.V. Vinodchandran, T.T. Osugi, SVM-based generalized multiple-instance learning via approximate box counting, in: ICML'04: Proceedings of the 21st International Conference on Machine Learning, ACM Press, Banff, Alberta, Canada, 2004.
[56] A. Tsakonas, A comparison of classification accuracy of four genetic programming-evolved intelligent structures, Information Sciences 176 (6) (2006) 691–724.
[57] S. Ventura, C. Romero, A. Zafra, J.A. Delgado, C. Hervás, JCLEC: a java framework for evolutionary computation soft computing, Soft Computing 12 (4) (2007) 381–392.
[58] J. Wang, J.-D. Zucker, Solving the multiple-instance problem: a lazy learning approach, in: ICML'00: Proceedings of the 17th International Conference on Machine Learning, Standord, CA, USA, 2000.
[59] N. Weidmann, E. Frank, B. Pfahringer, A two-level learning method for generalized multi-instance problems, in: ECML'03: Proceedings of the 14th European Conference on Machine Learning, Cavtat-Dubrovnik, Croatia, 2003.
[60] P.A. Whigham, Grammatically-based genetic programming, in: Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications, Tahoe City, California, USA.
[61] P.A. Whigham, Grammatical Bias for Evolutionary Learning, Ph.D. Thesis, School of Computer Science, University College, University of New South Wales, Australian Defence Force Academy, Canberra, Australia (14 Oct. 1996).
[62] T.S. Wiens, B.C. Dale, M.S. Boyce, P.G. Kershaw, Three way *k*-fold cross-validation of resource selection functions, Ecological Modelling 212 (3–4) (2008) 244–255.
[63] I. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, second ed., Morgan Kaufman, San Francisco, 2005.
[64] W. Wongseree, N. Chaiyaratana, K. Vichittumaros, P. Winichagoon, S. Fucharoen, Thalassaemia classification by neural networks and genetic programming, Information Sciences 177 (3) (2007) 771–786.
[65] X. Xu, Statistical Learning in Multiple Instance Problems, Ph.D. Thesis, Department of Computer Science. University of Waikato, Tauranga, New Zealand, 2003.
[66] X. Xu, E. Frank, Logistic regression and boosting for labeled bags of instances, in: PAKDD'04: Proceedings of the 8th Conference of Pacific-Asia, Lecture Notes in Computer Science, vol. 3056, Sydney, Australia, 2004.
[67] C. Yang, M. Dong, F. Fotouhi, Region based image annotation through multiple-instance learning, in: Multimedia'05: Proceedings of the 13th Annual ACM International Conference on Multimedia, New York, USA, 2005.
[68] C. Yang, T. Lozano-Perez, Image database retrieval with multiple-instance learning techniques, in: ICDE '00: Proceedings of the 16th International Conference on Data Engineering, IEEE Computer Society, Washington, DC, USA, 2000.
[69] A. Zafra, S. Ventura, C. Romero, E. Herrera-Viedma, Multi-instance genetic programming for web index recommendation, Expert Systems with Applications 36 (9) (2009) 11470–11479.
[70] D. Zhang, F. Wanga, Z. Shib, C. Zhanga, Interactive localized content based image retrieval with multiple-instance active learning, Pattern Recognition 43 (2) (2010) 478–484.
[71] M. Zhang, W. Smart, Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification, Pattern Recognition Letters 27 (11) (2006) 1266–1274.
[72] M.-L. Zhang, Z.-H. Zhou, Improve multi-instance neural networks through feature selection, Neural Processing Letters 19 (1) (2004) 1–10.
[73] M.-L. Zhang, Z.-H. Zhou, Ensembles of multi-instance neural networks, in: IIP'04: International Conference on Intelligent Information Processing II, IFIP International Federation for Information Processing, vol. 163, Beijing, China, 2005.
[74] M.-L. Zhang, Z.-H. Zhou, Adapting RBF neural networks to multi-instance learning, Neural Processing Letters 23 (1) (2006) 1–26.
[75] Q. Zhang, S. Goldman, EM-DD: an improved multiple-instance learning technique, NIPS'01: Proceedings of Neural Information Processing System, vol. 14, Vancouver, Canada, 2001.
[76] Z.-H. Zhou, Multi-instance learning from supervised view, Journal Computer Science and Technology 21 (5) (2006) 800–809.
[77] Z.-H. Zhou, K. Jiang, M. Li, Multi-instance learning based web mining, Applied Intelligence 22 (2) (2005) 135–147.
[78] Z.-H. Zhou, J.-M. Xu, On the relation between multi-instance learning and semi-supervised learning, in: ICML'07: Proceedings of the 24th International Conference on Machine Learning, Corvalis, Oregon, 2007.
[79] Z.-H. Zhou, M.-L. Zhang, Neural networks for multi-instance learning, Technical report, AI Lab, Computer Science and Technology Department, Nanjing University, Nanjing, China, August 2002.
[80] Z.-H. Zhou, M.-L. Zhang, Solving multi-instance problems with classifier ensemble based on constructive clustering, Knowledge and Information Systems 11 (2) (2007) 155–170.
[81] J.-D. Zucker, J.-G. Ganascia, Representation changes for efficient learning in structural domains, in: ICML'96: Proceedings of the 13th International Conference on Machine Learning, Garda, Italy, 1996.