

Solving Multiple-Instance and Multiple-Part Learning Problems with Decision Trees and Rule Sets. Application to the Mutagenesis Problem

Yann Chevaleyre and Jean-Daniel Zucker

LIP6-CNRS, University Paris VI,
4, place Jussieu,
F-75252 Paris Cedex 05, France
{Yann.Chevaleyre, Jean-Daniel.Zucker}@lip6.fr

Abstract. In recent work, Dietterich et al. (1997) have presented the problem of supervised multiple-instance learning and how to solve it by building axis-parallel rectangles. This problem is encountered in contexts where an object may have different possible alternative configurations, each of which is described by a vector. This paper introduces the multiple-part problem, which is related to the multiple-instance problem, and shows how it can be solved using the multiple-instance algorithms. These two so-called “multiple” problems could play a key role both in the development of efficient algorithms for learning the relations between the activity of a structured object and its structural properties and in relational learning. This paper analyzes and tries to clarify multiple-problem solving. It goes on to propose multiple-instance extensions of classical learning algorithms to solve multiple-problems by learning multiple-decision trees (ID3-MI) and multiple-decision rules (RIPPER-MI). In particular, it suggests a new multiple-instance entropy function and a multiple-instance coverage function. Finally, it successfully applies the multiple-part framework on the well-known mutagenesis prediction problem.

1 Introduction

Supervised learning can be seen as the search for a function h , a set of objects O towards a set of results R that will be a good approximation of a function f for which the result is only known for a certain number of objects of O , the examples of f (Dietterich [5]). This problem consists in inducing the description of h from a set of pairs ($description(object_i), result_i = f(object_i)$) - the learning examples - and criteria - learning bias - that enable a space of functions of O towards R to be chosen and one function to be preferred to another. The description of $object_i$ is often referred to as an instance of $object_i$. Recent research has shown that this traditional framework could be too limited for complex learning problems [6, 11, 2, 1]. This is particularly the case when several descriptions of the same object are associated with the same result, baptized a multiple-instance problem (MIP) by Dietterich et al. [6]. Thus the term multiple-instance characterizes the case

where the result $f(object_i)$ is associated not with one instance but with a set of instances $\{instance_{i,1}, instance_{i,2}, \dots, instance_{i,\sigma_i}\}$.

Chemistry is a domain *par excellence* where these multiple-instance problems are to be found. Dietterich et al. present the task of classifying aromatic molecules according to whether or not they are "musky" [6]. Several steric configurations of the *same* molecule can be found in nature, each with very different energy properties. In this way it is possible to produce several descriptions of the different configurations - instances - of this molecule. These descriptions correspond to measurements obtained in each of the different configurations. To simplify, let us say that a molecule is said to be musky if, in one of its configurations, it binds itself to a particular receptor. The problem of learning the concept "musky molecule" is one of multiple-instance learning. Maron and Lozano-Perez [9] consider other possible applications, such as learning a simple description of a person from a series of images.

Dietterich et al. have proposed different variations of a learning algorithm where the concepts are represented by axis-parallel rectangles (APR). They observed that "*a particularly interesting issue is how to design multiple-instance modifications for decision trees, neural networks and other popular machine learning algorithms*" [6]. This paper will analyze the difficulties raised by multiple-instance problems in general. It will show the link between this problem and the multiple-part problem (MPP), in which instances are not necessarily alternative descriptions of the object but may be descriptions of different parts of the object. "Multiple-extensions" will be proposed for classical algorithms in order to handle MIP and MPP problems by learning decision trees and rule-based systems. The main reasons that motivate us for finding such algorithms are that MPPs play a central role in learning structure-activity relations. This is the problem that was solved in the REMO learning system (Zucker and Ganascia [11]), and in the REPART (Zucker, Ganascia et al. [12]) Inductive Logic Programming system. Section 2, which is a more formal presentation of the MIP problem, shows how it is linked to the MPP problem and explains how in the two cases problem solving comes down to learning special concepts called multiple ones. Section 3 proposes extensions to classical algorithms in order to solve the multiple-problems and in particular suggests an entropy function and a multiple-instance coverage function. Section 4 presents the results of predicting mutagenicity with the multiple-part framework.

2 Multiple-instance and multiple-part problems

2.1 Definition of multiple-instance problems

For the sake of clarity, let us consider the case where f is a function with boolean values - a concept - the value of which is known for a subset $\{object_i\}$ of O . Thus $f(object_i) = TRUE$ (positive example) or $FALSE$ (negative example) - depending on whether or not $object_i$ belongs to the concept. We shall note $instance_{i,j}$ the j^{th} description of object $object_i$. We shall call X the representation space for instances and co-instances of $instance_{i,k}$, the other instances

of the example $object_i$, i.e. the set $\{instance_{i,j \neq k}\}$. Function h , which we are trying to learn and must be a good approximation of f , is a function which associates a boolean value with a subset of the parts of X , which can be noted by $h: 2^X \rightarrow \{TRUE, FALSE\}$. A learning example in the multiple-instance framework is represented in the following form: $(\{instance_{i,1}, \dots, instance_{i,j}, \dots, instance_{i,\sigma i}\}, f(object_i))$. It should be added that the number σi can vary depending on $object_i$ and that the suffix j of 1 to σi given to instances $instance_{i,j}$ is purely arbitrary. Note that in the limited theoretical research that has been done on the PAC-learnability of this problem, the number σi is equal to a constant r [2,3]. In the multiple-instance framework, Dietterich et al. [6] suggest that if the result of f is positive for an $object_i$ it is because *at least one of its instances has produced this result*. If the result is negative it means that none of its instances can produce a positive result. The researchers support this hypothesis by the fact that in the domain of molecular chemistry they are studying this is precisely the case. Here, this hypothesis will be called *the linearity hypothesis*. If we use the vocabulary introduced above, the multiple-instance problem presented by Dietterich et al. [6] in their seminal paper can be defined as follows:

Definition 1 (MIP). *The multiple-instance learning problem consists in learning a concept from examples that are represented by bags of instances that describe them, on the linearity hypothesis.*

2.2 Representation shifts for MIPs

The function h to be learned is more complex to learn than a traditional concept since it takes its values from the set 2^X of the parts of X which has a cardinal that increases exponentially with that of X . Today, no algorithm exists that is capable of solving this problem directly. A possible approach to too complex a problem would be to try to change the representation in order to find a representation where learning would be less complex (Giordana and Saitta [7]). Using the linearity hypothesis, it is possible to introduce a boolean concept rv_f which no longer applies to sets of instances but instead to one single instance of these sets. An instance belongs to this boolean concept if "the instance has produced the result". This representation shift of a concept defined on 2^X by a concept defined on X can be said to be isomorphic (Korf [8]) in that it changes the structure of the problem but not the amount of information. The concept thus defined will be called a "multiple-concept". Following on from the linearity hypothesis, h is therefore defined as a disjunction of the multiple-concept applied to the different instances of an object:

$$f(object_i) = rv_f(instance_{i,1}) \vee \dots \vee rv_f(instance_{i,\sigma i})$$

Property 1. The problem of multiple-instance learning of a concept f comes down to the mono-instance learning of a concept rv_f . The description of f is given as the logical OR of the values of rv_f on the different instances of an object.

Concept rv_f can be read as "responsible for the value of f ". The multiple-instance problem can be reformulated with respect to this new function. Figure 1 gives Property 1 in graphic form. If defining MIP is relatively easy, understanding and solving it are far less simple. To illustrate the problem intuitively, let us consider the problem we have decided to call the simple jailer problem. Let there be a locked door and a set of N bunches of keys containing a variable number of keys such that N_+ of the bunches of keys are labeled "useful" and N_- are labeled "useless" (not useful). A bunch of keys is said to be useful if at least one of its keys opens the door, otherwise it is considered useless. The concept of usefulness could be represented by two classes: that of useful bunches of keys and that of useless bunches of keys. Learning the concept "useful bunch of keys" is a MIP problem. Starting from a set of positive and negative examples of f (here, useful and useless bunches of keys), the concept rv_f must be learned, which characterizes the keys which open the door. This problem is said to be "simple" as it presumes the linearity hypothesis to hold, i.e. at least one key per useful bunch of keys is sufficient to open the door.

2.3 The multiple-part problem and how it is linked to the multiple-instance problem

In work done before the development of MIP problems, researchers have introduced a problem that was apparently similar to the MIP and that was baptized a reformulated problem (Zucker and Ganascia [11]) but which, for reasons of clarity, will henceforth be called the *multiple-part problem* (MPP). Informally, the MPP characterizes concept learning from the description of parts of examples.

In MPP as in MIP, each example is represented by a bag of instances. In MIP, an instance is a snapshot of the entire object, whereas in MPP, an instance is a small part of the object. Lets consider, for example, the application of MIP and MPP to chemistry. Has shown before, in MIP, the bag of instances related to a molecule would be measurements on various configurations of this molecule. In MPP, we would have to cut a molecule in small parts, each of which would become an instance. Of course, these parts will have to be homogenous. Putting the description of a single atom, or even of a pair of bonded atoms in each instance would both be valid MPP representations. In the first case, the example would be represented by a bag of attribute-value descriptions of each atom. In the second case, each possible pair of bonded atoms of a molecule will

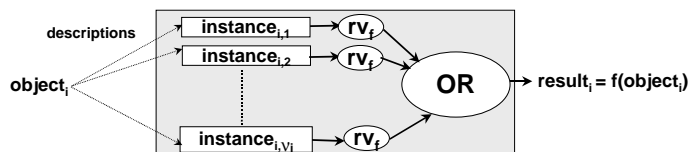


Fig. 1. Multiple instance learning of f and mono instance learning of rv_f

become an instance of that molecule. We can see now that the jailer problem mentioned above is more a MPP problem than a MIP problem. In fact, the keys are seen as parts of the same bunch and each of the instances describes one of the keys.

As seen above, there can be many valid MPP representation of the same data, depending on the size of the chosen parts. The linearity hypothesis, stating that a single instance can be used to identify the belonging of an example to the studied concept, depends here on the representation. For example, if we know that the presence of a carbon linked to a nitrogen atom makes a molecule highly active, it will then be impossible to predict such an activity by examining atoms individually. Hence, the MPP representation for which an instance corresponds to a single atom wont respect linearity hypothesis, whereas the one for which an instance corresponds to a pair of bonded atoms will.

Choosing the appropriate representation in MPP, and moreover shifting between representations is a crucial problem which is deeply studied in (Zucker and Ganascia [11]). In their article, they propose an algorithm called REMO which chooses such an appropriate representation dynamically. Starting with a very simple representation, it generates increasingly complex representations on which a MIP learner is launched, until an accurate enough hypothesis is obtained. In fact, if the linearity hypothesis does not hold on a simple representation, the MIP learner will not perform well and REMO will shift towards more complex representations, until the linearity hypothesis is recovered.

The following sections will focus on designing multiple-instance learners, which as shown here are needed to solve both MIP and MPP.

2.4 Multiple-concept learning

As presented in sections 2.1 and 2.3, MIP problems and MPP problems can be reduced to multiple-concept mono-instance learning (rv_f). Once learned, such concepts can be used to characterize the initial concept that is looked for. One of the difficulties of multiple-concept learning comes from the fact that we don't know any examples of these multiple-concepts in the traditional meaning of the term. All we know is whether the disjunction of the rv_f applied to co-instances is positive or negative. Intuitively, we can say that ignoring the specificities of the problem will not help to learn multiple-concepts satisfactorily. Ignoring a multiple-problem means that we consider that all the instances of the same example are from the same class as the example. In the jailer problem, this comes down to considering that all the keys on a bunch of keys open the door if the bunch is useful. A classical learning algorithm that is applied without any modifications to multiple-problems would thus fail to learn an accurate description of the target concept.

3 Adapting concept learning algorithms to learn multiple-concepts

As demonstrated informally in section 2, the learning of multiple-concepts comes down to the mono-instance learning of multiple-concepts. Moreover, it has been shown that the difficulty of learning such multiple-concepts is that there are no learning examples as classically used in the mono-instance framework. There exists a large number of algorithms that solve the mono-instance concept learning problem. The most popular ones are top-down inductive systems. They may be separated into two categories: divide-and-conquer approaches and cover-and-differentiate ones: The divide-and-conquer algorithms generally represent hypotheses as decision trees (ID3, C4.5, etc.) and many use a heuristics based on a variant of the entropy function to build the tree iteratively. The cover-and-differentiate algorithms generally represent hypotheses as sets of if-then rules (AQ, RIPPER [4], etc.). Many use a heuristics based on the number of examples covered by the rules. To date, the main approach for solving the multiple-instance problem is to learn APR. This section proposes extensions to classical concept learning algorithms in order to solve the multiple-problems, in particular through a multiple entropy function and a multiple coverage function.

3.1 Representing multiple-concepts and classifying multiple-instances

It is assumed that the basic notions of decision trees as defined in ID3 or C4.5 (Quinlan [10]) and those of IF-THEN rules as defined in AQ or RIPPER [4] are familiar to the reader. A decision tree used to represent a multiple-concept rv_f will be called multiple-decision tree for the sake of clarity. Similarly, a rule set used to represent multiple-concepts will be called a multiple-ruleset. Multiple-decision trees and multiple-rulesets have the same structure as a classical decision trees and rule sets. In fact, multiple-classifiers differ from traditional classifiers in the way they are used to classify a new bag and in the way they are learned. To classify a new object in the MIP where the concept rv_f is represented as a multiple-tree, the entire bag of instances is passed through the tree. If (or as soon as) one positive leaf is reached by one of the instances, the object is classified positive, negative otherwise. Similarly, to classify an object in the MIP with a multiple-ruleset, each instance is passed through each rule. If at least one instance fires a rule, then the entire bag is classified positive, negative otherwise. Figure 2 uses the jailer problem to illustrate these notions.

3.2 Learning multiple decision-trees and rules

Both MIP and mono-instance framework use attribute-value representation. In addition, classical learning tools use generate-and-test algorithms, exploring a search space which is as suitable for mono-instance as for MIP. Hence, only the test part of the algorithm will have to be modified. We will therefore describe the MIP adaptation of heuristics used in mono-instance learners.

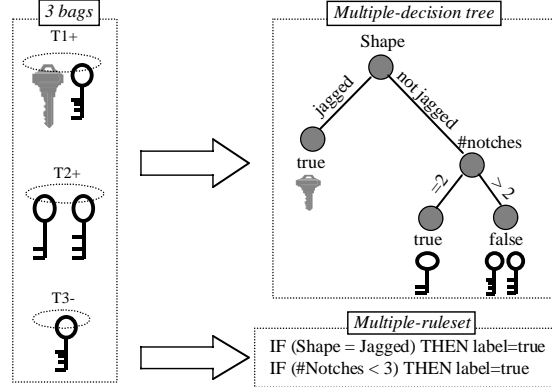


Fig. 2. Two useful bunches of keys T1+ and T2+ and a useless bunch T3- are used to induce a multiple-decision tree and a multiple-ruleset. All three bags are correctly classified by both classifiers.

3.3 Multiple-instance entropy and coverage for multiple- concept learning

Classically, the growing of a decision tree is guided by a heuristics based on entropy or a related criterion. Given a collection S containing p positive instances and n negative instances of some target concept, the entropy of S relative to this boolean classification is : $Info_{mono}(S(p, n)) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$ The information gain $Gain(S, A)$ of an attribute A relative to a collection of instances S is defined as: $Gain(S(p, n), A) = Info(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times Info(S_v)$. Let us define an extension to both the entropy of S and the gain of an attribute w.r.t. S in the multiple-instance framework. In this context, let us consider a set S containing p positive instances of the concept rv_f and n negative instances of the concept. Let us introduce two functions π and ν that, given a set of instances S , return the number of different positive examples and negative examples that the elements of S are instances of respectively. The entropy that characterizes the (im)purity of an arbitrary collection of examples ought to be redefined here so as to take into account the fact that one example is represented by several instances.

In multiple-problems, the goal is to learn a concept for discriminating examples and not instances. Therefore, the (im)purity ought not to be measured by p and n , the number of positive or negative instances of the concepts rv_f but using $\pi(S)$ and $\nu(S)$, which represent the number of examples that have representatives in S . The multiple- instance entropy and gain may therefore be defined as:

$$Info_{multi}(S(p, n)) = -\frac{\pi(S)}{\pi(S)+\nu(S)}\log_2\left(\frac{\pi(S)}{\pi(S)+\nu(S)}\right) - \frac{\nu(S)}{\pi(S)+\nu(S)}\log_2\left(\frac{\nu(S)}{\pi(S)+\nu(S)}\right)$$

$$Gain_{multi}(S(p, n), A) = Info_{multi}(S) - \sum_{v \in Values(A)} \frac{\pi(S_v)+\nu(S_v)}{\pi(S)+\nu(S)} \times Info_{multi}(S_v)$$

This multiple-instance entropy directly implemented in a decision tree learner will result in overly complex trees. Let us illustrate this drawback with an example. Suppose that only the root node of the tree has been induced. Suppose that one of the positive bags from the training set contains two instances, the first one following the left branch of the root node, and other one following the right branch. If the left subtree being induced succeeds in classifying positively the first instance of this positive bag, then trying to correctly classify the other instance during the induction of the right subtree is useless. Nevertheless, a learner such as ID3 extended with the multiple-instance entropy will try to classify both instances correctly, which will lead to an overly complex tree.

To avoid this drawback, the induction process has to be modified as follows: when an instance from a positive bag is correctly classified by the tree being induced, remove *all* its co-instances from the training set. In our example, as soon as a subtree attached to the root node correctly classifies an instance of the positive bag, the other instance is immediately removed. Note that this drawback is specific to separate-and-conquer algorithms. Based on the multiple entropy measure and this algorithmic modification, ID3-MI has been built as a multiple version of the well known ID3 decision tree learner.

3.4 Learning multi-rules

This section focuses on ruleset learners that are based on a coverage measurement. The growing procedure of the set of rules used in such kinds of algorithms relies on the notion of coverage. To learn multiple-rules, it is necessary to redefine this very notion of coverage. In a classical framework, an instance x is covered by a generalization G (noted $COVER(G, x)$) if G is more general than x . To measure the degree of generality of a generalization w.r.t. a set of examples, this notion should be refined. In the multiple instance framework, a generalization G "multi-covers" an $object_i$ if it covers at least one of its instances: $COVER_{multi}(G, Object_i) \leftarrow \exists j / COVER(G, instance_{i,j})$. The number of covered bags is thus: $COVER_{multi}(G) = |\{object_i \mid COVER_{multi}(G, object_i)\}|$. Based on this measure, RIPPER-MI has been built as a multiple-instance extension of Cohen's efficient rule learner RIPPER [4]. In the next section, various experiments will be done using this algorithm.

4 Predicting mutagenicity using MPP framework

The prediction of mutagenicity problem is considered as a typical benchmark for first-order induction tools. In fact, The highly structured nature of molecules prohibits the straightforward use of propositional representation. The learn goal is here to generate a theory which, provided a given molecule, will best predict its mutagenic activity. The available database consist of 230 chemical compounds. In the following, a subset of 188 molecules known as being *regression-friendly* will be used. For each molecule, the available background knowledge provides several description levels. At the atomic level, two predicates describe the atoms

Table 1. Accuracy measured with a tenfold validation of various learners on the 188 compounds of the *regression-friendly* set

Learner	Accuracy with \mathcal{B}_0	Accuracy with \mathcal{B}_2
RIPPER-MI on individual atoms	0.75 (0.04)	0.90 (0.02)
RIPPER-MI on pairs of bonded atoms	0.78 (0.02)	0.90 (0.02)
PROGOL	0.79	0.86
FOIL	0.61	0.83

and the bonds. At the molecular level, global informations concerning the entire molecule are given, such as the hydrophobicity and the lowest molecular orbital of a molecule. The dataset restricted to the atomic description level is often referred to as \mathcal{B}_0 , whereas the \mathcal{B}_2 refers to the dataset including both atomic and molecular informations.

Together with RIPPER-MI, the REMO [11] algorithm was used to generate various multiple-instance representations from \mathcal{B}_0 and \mathcal{B}_2 . Using \mathcal{B}_0 , REMO began by generating a multiple-instance dataset in which each bag contains attribute-value descriptions of individual atoms. During the next run, REMO generated a dataset in which each instance of each bag represents a pair of bonded atoms. Using \mathcal{B}_2 , the two first datasets generated were the same as for \mathcal{B}_0 , except that molecular attributes were added to each instance. Table 1 displays the accuracy of RIPPER-MI measured with a tenfold cross validation, on these four MPP-datasets, as well as the accuracy of two popular relational learners. PROGOL and FOIL.

The best accuracy was obtained using the simple representation \mathcal{B}_2 with individual atoms. Using pairs of bonded atoms in this case does not lead to an increase of the accuracy. In fact, the global molecular features available in \mathcal{B}_2 are highly correlated with the activity of the molecule. Hence, taking into account pairs of atoms does not bring much more information than using only individual atoms. On the contrary, using \mathcal{B}_0 , the accuracy of RIPPER-MI significantly increases when using pairs of bonded atoms instead of just individual atoms.

Despite the fact that RIPPER-MI uses a greedy-search algorithm, it is competitive in terms of predictive accuracy, with respect to other learners. In addition, it is much faster: using the dataset which represents individual atoms and global molecular properties, rules sets are generated in an average times of 2.6 seconds. In addition, the induced hypotheses are concise, as they contain an average of six rules. The following is an example of rule generated by our learner: **active** \leftarrow (**type1 = 1**) \wedge (**ch1 < 0.288**) \wedge (**ch2 < -0.404**). It indicates that if a molecule has a pair of bonded atoms such that the first one is of type 1 and has a partial charge lower than 0.288 and that the second one has a partial charge lower than -0.404, then the molecule is mutagenic.

The results obtained on this real-world problem show that the multi-instance paradigm, sort of missing link between propositional and first order representation, is very promising for a wide range of future learning problems.

5 Conclusion

The problem of supervised multiple-instance learning is a recent learning problem which has excited interest in the learning community. The problem is encountered in contexts where an object may have several alternative vectors to describe its different possible configurations. This paper has shown that the problem is subsumed by the multiple-part problem, which can play a key role in relation-learning algorithms and in inductive logic programming (Zucker et al. [12]). Multiple-instance learning were first applied to the prediction of drug activity. Very recently, Maron et Lozano-Perez [9] have proposed a framework called *Diverse Density* for solving multiple-instance problems. Solving multiple-problems using classical algorithms raises important subtle issues that have been analyzed here. The paper has shown how these problems can be solved using multiple-concept mono-instance learning algorithms. Extensions to classical algorithms have been proposed to solve these problems by learning decision trees and decision rules. They are based on two notions: multiple- instance entropy and multiple-instance coverage. Thanks to these modifications it was possible to implement the learners ID3-MI, RIPPER-MI. Our experiments on the mutagenesis problem show that our approach performs well, and that MIP algorithms can handle numeric as well as symbolic data. It also suggests efficient multiple-instance algorithms could be of primary interest for relational learning tasks.

References

1. Alphonse, E., Rouveirol, C. 1999. Selective Propositionalization for Relational Learning. Principles and Practice of Knowledge Discovery in Databases.
2. Auer, P. 1997. On learning from multi-instances examples. Empirical evaluation of a theoretical approach. 14th International Conference on Machine Learning.
3. Blum, A. and A. Kalai 1997. A note on learning from multiple-instances examples. Machine Learning .
4. Cohen, W. 1995. Fast Effective Rule Induction. International Conference on Machine Learning.
5. Dietterich, T. 1990. Inductive Learning from Preclassified Training Examples. Readings in Machine Learning: 45-56.
6. Dietterich, T., R. Lathrop, et al. 1997. Solving the Multiple-Instance Problem with Axis-Parallel Rectangles. Artificial Intelligence 89(1-2): 31-71.
7. Giordana, A. and L. Saitta 1990. Abstraction: a general framework for learning. AAAI Workshop on Automated Generation of Approximations and Abstraction.
8. Korf, R. E. 1980. Towards a Model for Representation Change. Artificial Intelligence 14: 41-78.
9. Maron, O. and T. Lozano-Prez. 1998. A Framework for Multiple-Instance Learning. Neural Information Processing Systems
10. Quinlan, J.-R. 1986. Induction of Decision Trees. Machine Learning(1): 81-106.
11. Zucker, J.-D. and J.-G. Ganascia 1996. Changes of Representation for Efficient Learning in Structural Domains. International Conf. in Machine Learning.
12. Zucker, J.-D., J.-G. Ganascia, Bournaud. I. 1998. Relational Knowledge Discovery in a Chinese Characters Database. Applied Artificial Intelligence Journal, Special Issue on KDD in Structural Domains.