Solving multiple-instance and multiple-part learning problems with decision trees and decision rules.

Application to the mutagenesis problem

Jean-Daniel Zucker, Yann Chevaleyre

LIP6-CNRS, University Paris VI
4, place Jussieu
F-75252 Paris Cedex 05, France
{Jean-Daniel.Zucker,Yann.Chevaleyre}@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 more general than 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 inductive logic programming. 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-M, C4.5-M) and multiple-decision rules (AQ-M, CN2-M,Ripper-M). 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.

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 1990). This problem consists in inducing the description of h from a set of pairs (description(object_i), $\text{result}_i=f(\text{object}_i)$) - 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 (Zucker and Ganascia 1994; Dietterich, Lathrop et al. 1996; Long and Tan 1996; Zucker and Ganascia 1996; Auer 1997). 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. (Dietterich, Lathrop et al. 1996). Thus the term *multiple-instance* characterizes the case where the result $f(\text{object}_i)$ is associated not with one instance but with a set of instances $\{\text{instance}_{i,1}, \text{instance}_{i,2}, \ldots \text{instance}_{i,v_i}\}$, (cf. Fig. 1).

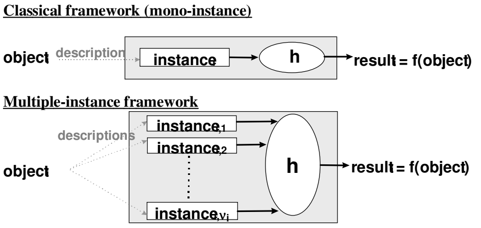


Figure 1 - Classical and multiple-instance frameworks

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" (Dietterich, Lathrop et al. 1996). 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 (instances m17,1 and m17,2 of molecule m17, cf. Fig. 2). 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-Pérez consider other possible applications: one is to learn a simple description of a person from a series of images that are labeled positive if the person is somewhere in the image and negative otherwise. The other is to deal with a high amount of noise in a stock selection problem.

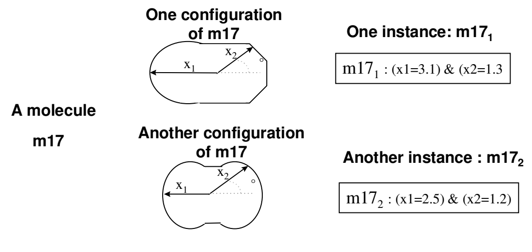


Figure 2 -Two instances of the same molecule

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*" (Dietterich, Lathrop et al. 1996).

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 MMPs play a central role in learning structure-activity relations. This is the problem that was solved in the REMO learning system (Zucker and Ganascia 1994; Zucker and Ganascia 1996), REPART (Zucker, Ganascia et al. 1998) and STILL (Sebag and Rouveirol 1997) Inductive Logic Programming systems.

Section 2 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 mutagenecity with the multiple-part framework.

Multiple-instance and multiple-part problems

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 of O $f(object_i)$=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 $instance_{ik}$, 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 part of X, which can be noted by h: 2^X {TRUE, FALSE}. A learning example in the multiple-instance framework of is represented in the following form:

$(\{instance_{i,1}, ..., instance_{i,j}, ..., instance_{i,\nu i}\}, f(objet_i))$

It should be added that the number ν_i can vary depending on $object_i$ and that the suffix j of 1 to ν_i 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 vi is equal to a constant r (Long and Tan 1996; Auer 1997; Auer, Long et al. 1997; Blum and Kalai 1997).

In the multiple-instance framework, Dietterich et al. (1997) suggest that if the result of f is positive for an $object_i$ it is because **at least one of its $instances_{ij}$ 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, let us call this hypothesis the *linearity hypothesis*. If we use the vocabulary introduced above, the multiple-instance problem presented by Dietterich et al. (1997) 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* ***sets*** *of instances that describe them, on the linearity hypothesis.*

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 (Subramanian 1989; Cohen 1990; Giordana and Saitta 1990). Using the linearity hypothesis, it is possible to introduce a boolean concept in_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 1980) 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:

$$f(objet_i) = rv_f(instance_{i,1}) \vee .... \vee rv_f(instance_{i,\nu i})$$

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.

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.*

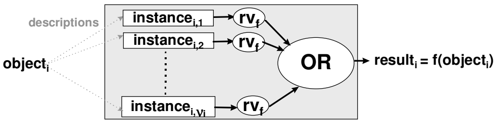


Figure 3 - Multiple-instance learning of f and mono-instance learning of rv_f.

Figure 3 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 an MIP. 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.

Strictly speaking, the jailer problem doesn't correspond exactly to the original MIP problem. In the original definition, the instances are all representative of the same object, of the same reality. Thus, in the case of chemical molecules, each configuration is a possible state of the molecule. What is important therefore is that in the MIP problem, the configurations cannot appear *simultaneously* since each of them characterizes the object taken as a whole. In order to allow for this exclusive aspect between the different co-instances in the jailer problem, we need to say that a bunch of keys is in fact a "magic" (or quantum) key c_i which takes a shape from among ν_i shapes when introduced into the keyhole and that it is useful if one of the shapes it can take opens the door.

This variable character of the measurements of the same object in the MIP problem means we can interpret MIP as a classical mono-instance problem that has been made *ambiguous* (Blum and Kalai 1997). The label of the object is not associated with one single description of the object (a magic key, a molecule) but with several descriptions which are all of the same object but which represent different states of the object (a key, a configuration), these different states all being potential explanations of the result. It is this type of ambiguity that must be allowed for in MIP learning algorithms. It is also on this property that the work on the PAC-learnability of MIP is based in order to reduce it to known problems.

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 1994) but which, for reasons of clarity, we will henceforth be called the *multiple-part problem* (MPP). Informally, the MPP characterizes concept learning from the description of parts of examples. MPP-solving lies at the heart of the REMO system which enables the efficient learning of relations from several thousand structured examples (Zucker and Ganascia 1996). In order to build a disjunctive version space, the STILL system solves an MPP problem iteratively, in which it takes one positive example at a time (Sebag and Rouveirol 1997). This system has obtained the best results for the ILP problem of mutagenesis (Srinivasan, Muggleton et al. 1997).

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. Let's 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 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 now 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 an single atom won't 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 1996). We will ourselves here focus on the induction aspects of the learning process in MIP and MPP, without assuming the linearity hypothesis for MPP, because the validity of the representation won't be assured. The multiple-part problem can be defined more formally as follows:

Definition 2 (MPP)***:*** *The multiple-part learning problem consists in learning a concept from examples that are represented by sets of instances that describe their parts, without the linearity hypothesis.*

Representation shift for MPP problems

Given the fact that the linearity hypothesis is not assumed, one way wonder if it is still possible to introduce a function as in the MIP which no longer applies to sets of instances but instead to *one* single instance of these sets? The answer is yes, all we need to do is to consider a new function in_f with the particular semantics "belongs to an object with the result f". However, in the MPP it becomes possible, even in the absence of noise, for two instances (two instances of keys, for example) to be absolutely identical but to come from objects from different classes (one from a useful bunch of keys, the other from a useless one, for example). Therefore the function in_f must by definition take the value "IMPERCEPTIBLE" on this type of instance, the consequence being that in_f is not a boolean function.

If we want to reduce the problem to one of concept learning, and therefore to a boolean function, as in the case of MIP, it is necessary either to group the values "FALSE" and "IMPERCEPTIBLE" in one value "NOTTRUE", or to group the values "TRUE" and "IMPERCEPTIBLE" in one value "NOTFALSE". In this way it is possible to replace learning the function in_f by learning the two concepts oin_{f+} (only in f+) and oin_{f-} (only in f-). In the jailer problem, they correspond to the concepts of *keys found in bunches of keys which open (resp. do not) open the door and in no bunch of keys which do not (resp. do).* Alternatively, one could also say that they "are *only* to be found among the objects that satisfy (resp. do not satisfy) the concept f". If the MPP problem is more general than the MIP one, it is above all on the *semantic* level that this is true since, from a computational point of view, the two cases require learning multiple-concepts and then performing a logical operation on the values of these multiple-concepts for the different instances of an object.

Property 2: The problem of multiple-part learning of a concept f comes down to the mono-instance learning of two concepts oin_{f-} and oin_{f+}, where f is obtained as follows: $f(\mathrm{objet}_i) = [\mathrm{oin}_{f+}(\mathrm{instance}_{i,1}) \vee .... \vee \mathrm{oin}_{f+}(\mathrm{instance}_{i,vi})\ ]$ OP $\neg[\ \mathrm{oin}_{f-}(\mathrm{instance}_{i,1}) \vee .... \vee \mathrm{oin}_{f-}(\mathrm{instance}_{i,vi})\ ]$ where OP is a logical OR if positive is the most common class and a logical AND otherwise.

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 or oin_{f-} and oin_{f+}). Once learned, these 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 in_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 learn a description which covers all the instances of all the positive examples of a multiple concept. It is obvious that such a description rarely exists in MIP problems and that it is even more highly unlikely in the framework of MPP. After all, all we need is for two examples to have identical instances but to be classified differently for such a description not to exist.

In the jailer problem, if two bunches of keys one of which is useful, the other useless each contains a key the descriptions of which are very similar or even identical, it will be impossible to find this description. Let us suppose that the bunches of keys are represented by instances of keys represented by pairs (x,y), where x is the size of the key and y the number of notches. Let us also suppose that there are three useful bunches of keys (the instances of keys being represented by squares, triangles and rectangles, respectively) and two useless bunches (the instances being represented by circles and ellipses, respectively). Let us consider two-dimensional space

where each instance of a bunch of keys has been represented by the shape of the bunch of keys (cf. Fig. 4).
The only MPP-solving algorithms are rectangle learning algorithms. The method proposed by Dietterich applies to multiple-instances represented by vectors in $X=R^d$. The multiple-concept that is being looked for is represented by a rectangle in Rd. The heuristics used consists in finding a small rectangle that is consistent with the data in that at least one of the instances of each of the positive initial examples is contained in it and none of the instances of the negative initial examples is (APR_1 in figure 4). This constraint can be relaxed if no box has been found so that only (100-ε)% of examples are covered (i.e. that at least one of their instances is in the rectangle). Note that the search problem is already NP-complete in itself (Blum and Kalai 1997). Then the APR that has been found is dilated (APR_2 in figure 4) in order to increase its power to generalize, based on a kernel density estimate.

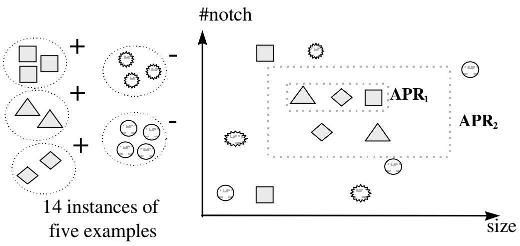


Figure 4 - The rectangle solution to the jailer MIP

MIP learnability

Members of the COLT community are showing increasing interest in multiple-instance problems. Long and Tan (Long and Tan 1996) were the first to show the PAC-learnability of the MIP problem under restrictive conditions. They showed that in this framework, the problem is PAC-learnable with a bound which has since been pushed out considerably. Auer et al. (Auer, Long et al. 1997) consider APR-learning under the condition that D (distribution on instances of an initial example) be a product of distributions and that the number of instances per example be a constant r. They showed that the rectangles are PAC-learnable with $m=O(d^2r^2/\varepsilon^2)$ examples where d is the dimension of the instance vectors, the time necessary for learning therefore being in O(dxmxlog(m)). Auer (Auer 1997) did a theoretical analysis of the problem and derived an algorithm, MULTINST, which gives performance levels comparable to those of Dietterich et al. (97). It also learns APR and is no longer bound by the limitation of having as many instances for each example. What is particularly interesting in his paper is the fact that he started from theoretical considerations and ended up developing a competitive practical algorithm. As for Blum and Kalai (Blum and Kalai 1997), they elegantly reduced the MIP to a problem of PAC-learning with a one-sided random classification noise.

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, section 2.5 has 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 (TDIS) (Ganascia 1993). 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 either as sets of if-then decision rules (AQ, CHARADE, etc.) or as decision lists (CN2). Many use a heuristics based on the number of examples covered by the rules.

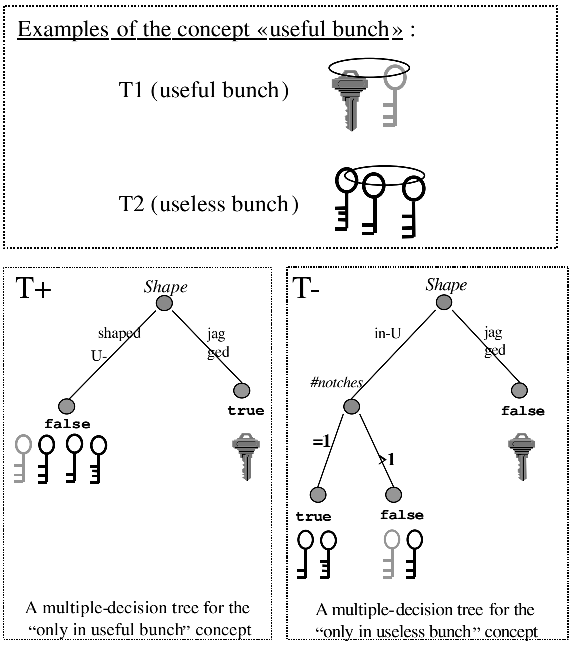
To date, the main approach for solving the multiple-instance problem is to learn APR. This section proposes extensions to classical concept learning algorithms in order to solve the multiple-problems, in particular through a **multiple entropy** function and a **multiple coverage** function.

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 1986) and those of if-then rules as defined in AQ or CHARADE (Ganascia, 1993) are familiar to the reader . A decision tree used to represent multiple-concepts will be called multiple-decision tree for the sake of clarity. Similarly, a classification rule used to represent multiple-concepts will be called multiple-rule.
In a decision tree, a leaf may be labeled positive (resp. negative) if it is pure - contains only positive (resp. negative) instances - or if it contains (100-ε)% positive (resp. negative) instances in the case of pre-pruning. Here the same labeling of tree leaves will be kept but both the way the multiple-tree is used to classify a new object and the procedure used to choose the attribute to grow the tree will be modified. In this way, a multiple-decision tree has the same structure as a classical decision tree. In the MIP, a single multiple-decision tree is enough. To classify a new object in the MIP where the concept in_f is represented as a multiple-tree, the set of instances of a new object 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.

In contrast, in the MPP two multiple-trees have to be learned, a T+ for the concept oin_{f+} and a T- for the concept oin_{f-}. To classify a new object the set of instances of a new object is passed through the two trees T+ and T-. If one positive (resp. negative) leaf is reached by one of the instances in T+ (resp. T+) and none in T- (resp. T+), the object is classified positive (resp. negative). It might happen that neither of these two conditions is reached. In this case it means that the new object only contains IMPERCEPTIBLE instances, i.e. instances that belong to both positive and negative examples, in which case the object may be classified in the most common class. Figure 5 uses the jailer problem to illustrate the notions presented.



A multiple-decision tree for the “only in useful bunch” concept

A multiple-decision tree for the “only in useless bunch” concept

Figure 5 - Two multiple-decision trees used on a jailer problem

Multiple-rules will be defined using the same philosophy as for decision trees. Such rules will differ from classical classification rules both in the way they are used to classify a new object and in the way they are learned.

To classify a new object in multiple problems, the set of instances of a new object is tested on the rules. If at least one positive (resp. negative) conclusion and no negative (resp. positive) conclusions are triggered the object is classified positive (resp. positive). If no rules are triggered, the object only contains IMPERCEPTIBLE instances and the most common class may be chosen.

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 therefor describe the MIP adaptation of heuristics used in mono-instance learners.

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 :

$$Entropy_{mono}(S(n,p)) = -\frac{p}{p+n}\times\log_2(\frac{p}{p+n}) - \frac{n}{p+n}\times\log_2(\frac{n}{p+n})$$

The information gain Gain(S,A) of an attribute A relative to a collection of instances S is defined as:

$$Gain(S(n,p),A) = Entropy(S) - \sum_{v\in Values(A)}\frac{|S_v|}{|S|}\times Entropy(S_v)$$

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 in_f (or oin_{f+} or oin_{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 in_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:

$$Entropy_{multi}(S(n,p)) = -\frac{\pi(S)}{\pi(S)+\nu(S)}\times\log_2(\frac{\pi(S)}{\pi(S)+\nu(S)}) - \frac{\nu(S)}{\pi(S)+\nu(S)}\times\log_2(\frac{\nu(S)}{\pi(S)+\nu(S)})$$

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

The MDL principle used to define an explicit measure of the complexity for encoding the training instances and the multiple-decision tree (Quinlan and Rivest 1989) can be extended similarly. Based on the multiple entropy measure, ID3-M, C4.5-M have been built as multiple versions of the corresponding algorithms.

Learning multi-rules

This section focuses on set of rules 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-cover" an object$_i$ if it covers at least one of its instances:

$$\mathrm{COVER}_{\mathrm{MULTI}}(G,\mathrm{objet}_i) \leftarrow \exists j \mid \mathrm{COVER}(G,\mathrm{instance}_{i,j})$$

$$\mathrm{COVERAGE}_{\mathrm{multi}}(G) = \left| \{\mathrm{objet}_i ; \mathrm{COVER}_{\mathrm{MULTI}}(G,\mathrm{objet}_i)\} \right|$$

Based on this measure, AQ-M, CN2-M and CHARADE-M have been built that are multiple versions of the corresponding algorithms.

Predicting mutagenecity using MPP framework

The prediction of mutagenecity 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 nitro-aromatic compounds, often split in two sets, of 188 and 42 molecules each. For each molecule, the available data consist of :
An atomic representation : The predicate *atom/5* describes each atom, including its chemical element, its type (e.g. aromatic atom), and its partial charge, which depends on the neighboring atoms. The predicate *bond/3* describes the bonded atoms.
A "group level" representation: benzene rings, nitro groups, and other high level structures present in the molecules are described.
A molecular-property level: two global attributes concerning the entire molecule are given, e.g. hydrophobicity (logp/2), lowest molecular orbital (lumo/2).

Results and discussion

We used here a multiple version of Cohen's RIPPER propositional learner. This tool, entitled 'RippMi' induces rules given bags of instances. The REMO algorithm was also involved to generate bags of instances given a first-order description of molecules. To achieve this representation shift, the user was asked by REMO to provided as additional input data a description of the intrinsic structure of the examples. In our case, we specified that each example has a main object - a molecule – associated with two attributes, and that this object contains smaller ones - atoms - associated with their own set of attributes.
Given that information, REMO will generate a fist MPP representation, launch the learner, and iteratively shift towards more complex representation, until the set of rules becomes acceptable.
Such representation will be denoted by the different parts contained in the instances. For example, {molecule+2atom} indicates that for a given molecule, each instance contains the two attributes relative to that molecule, as well as the attributes of 2 atoms. {atom} denotes a representation wherein one instance corresponds to one atom. Table 1 displays the results obtained by a 10-fold crossvalidation, using various MPP-representations.

Representation	Accuracy
{atom}	0.75 (0.04)
{2 atoms}	0.78 (0.02)
{molecule+atom}	0.88 (0.02)
{molecule+2atoms}	0.88 (0.02)

Table1: RippMi's accuracy using various representation of dataset containing 188 compounds.

Surprisingly, the best accuracy was obtained using the near-simplest representation : {molecule+atom}. An example of rules generated by RippMi with this representation is:

(covers 18 examples)

```
active(M):- logp(M,L), L >= 3.37,
       lumo(M,Lu), Lu <= -1.361,
       atm(M,A,Elt,Typ,PCharge),
       PCharge <= -0.376.
```

(covers 46 examples)

```
active(M) :-
       atm(M,A,Elt,Typ,PCharge),
       member(Typ,[27,51,31]),
       PCharge <= -0.085.
```

With such a representation, given some molecular properties, it becomes possible to determin whether a molecule is or not mutagenic only by looking at its atoms one by one. Nevertheless, we cannot state from these results that a single atom is responsible for the mutagenecity of a molecule. In fact, the atomic attributes most often used by RippMi during induction do not depend only on the atom they belong too : the partial charge, for example, can only be computed with the knowledge of the neighboring atoms; the type of the atom also depends on its environment. It is, on the other hand, highly probable that one of the atoms responsible for mutagenecity of a molecule will be covered by a rule. It is therefore conceivable to discover an entire group of atom responsible for mutagenecity, starting from a single atom of this group. One of our research goals would be, assuming the previous hypothesis, to investigate more complex representation (i.e. involving several atoms) leading to simpler theories.

Comparing results with other learning systems

We have performed experiments to compare our system with Progol and a few others, using the same dataset of 188 compounds and the same background knowledge. The results are summarized in the table below.

Progol	Linear regression	Fors	RippMi
0.88	0.89	0.89	0.88(0.02)

Table 2: accuracy using k-fold validation

Despite the fact that RippMi uses a greedy-search algorithm, its shows quite competitive in terms of predictive accuracy, with respect to other learners. In addition, it is much faster – only 2.6 seconds to generate rules with {molecule+atom} representation - probably because of its simplicity, compared to first-order learners.
The results obtained on this real-world problem shows that the multi-instance paradigm, sort of “missing link” between propositional and first order representation, is very promising for a wide range of future biochemical learning problems.

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., 1998). Multiple-instance learning were first applied to the prediction of drug activity. Very recently, Maron et Lozano-Pérez have proposed a framework called *Diverse Density* for solving multiple-instance problems (Maron et Lozano-Pérez 1998). 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 has been possible to implement the programs ID3-M, AQ-M, C4.5-M, CN2-M, CHARADE-M, and RippMi (the Java implementations of these programs will be available by request to the authors). Our practical experiments on the mutagenesis problem show that our approach performs well, and that multi-instance algorithms can handle numeric as well as symbolic data.
Many questions remain unanswered concerning multiple-problem solving and these questions can all be followed up. As far as can be ascertained, no theoretical research has yet been done on MPP problem learnability but it is highly likely that it can be reduced to a multiple-instance problem in which noise is injected. As for practical research, many approaches need to be tried and neural algorithms still have to be found for multiple-instance and multiple-part problems.

References

Auer, P. 1997. On learning from multi-instances examples. Empirical evaluation of a theoretical approach. Fourteenth International Conference on Machine Learning, ICML'97, pp. 21-29.

Auer, P., Long, P. 1997. Approximating hyper-rectangles: Learning and pseudo-random sets. Proceeding of the 29th Annual ACM Symposium on Theory of Computation.

Blum, A. and A. Kalai 1997. “A note on learning from multiple-instances examples” Machine Learning .

Cohen, W. 1990. An analysis of Representation Shift In Concept Learning. Seventh International Conference on Machine Learning, Austin, Texas.

Cohen, W. 1995. Fast Effective Rule Induction. International Conference on Machine Learning.

Dietterich, T. 1990. Inductive Learning from Preclassified Training Examples. Readings in Machine Learning: 45-56.

Dietterich, T., R. Lathrop, et al. 1996. “Solving the Multiple-Instance Problem with Axis-Parallel Rectangles.” Artificial Intelligence 89(1-2): 31-71.

Ganascia, J.-G. 1993. TDIS: an Algebraic Formalization. 13th International Joint Conference on Artificial Intelligence, Chambéry, France, Morgan Kaufmann.

Giordana, A. and L. Saitta 1990. Abstraction: a general framework for learning. Working notes of the AAAI Workshop on Automated Generation of Approximations and Abstraction, Boston, MA.

Korf, R. E. 1980. "Towards a Model for Representation Change." Artificial Intelligence 14: 41-78.

Long, P. and L. Tan. 1996. PAC Learning Axis-aligned Rectangles with respect to Product Distributions from Multiple-instance Examples. Proceedings of the 9th Annual Conference on Computational Learning Theory, COLT' 96, Desenzano del Garda, Italy, ACM, Inc.

Maron, O. and T. Lozano-Pérez. 1998. "A Framework for Multiple-Instance Learning." Neural Information Processing Systems (to appear).

Murphy, P. and D. Aha 1994. UCI repository of machine learning databases, Dept. of Information
And Computer Science, Irvine: CA. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Quinlan, J.-R. 1986. "Induction of Decision Trees." Machine Learning(1): 81-106.

Quinlan, J. R. and R. Rivest 1989. "Inferring Decision Trees Using the Minimum Description Length Principle." Information and Computation 80: 227-248.

Sebag, M. and C. Rouveirol 1997. Tractable Induction and Classification in First Order Logic. Fifteenth International Joint Conference on Artificial Intelligence, IJCAI'97, Nagoya, Japan, Morgan Kaufmann.

Srinivasan, A., S. Muggleton, et al. 1997. The Predictive Toxicology Evaluation Challenge. Fifteenth International Joint Conference on Artificial Intelligence, IJCAI'97, Nagoya, Japan, Morgan Kaufmann.

Subramanian, D. 1989. Representational Issues in Machine Learning. The sixth international Workshop on Machine Learning, Ithaca, NY, Morgan Kaufmann Publishers.

Zucker, J.-D. and J.-G. Ganascia 1994. Selective Reformulation of Examples in Concept Learning. International Conference on Machine Learning, New-Brunswick, Morgan Kaufmann Publishers.

Zucker, J.-D. and J.-G. Ganascia 1996. Changes of Representation for Efficient Learning in Structural Domains. International Conference in Machine Learning, Bari, Italy, Morgan Kaufmann.

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 (to appear).