

Learning Rules from Multiple Instance Data: Issues and Algorithms

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

Abstract

In a multiple-instance representation, each learning example is represented by a “bag” of fixed-length “feature vectors”. Such a representation, lying somewhere between propositional and first-order representation, offers a tradeoff between the two. This paper proposes a generic extension to propositional rule learners to handle multiple-instance data. It describes NAIVE-RIPPERMI, an implementation of this extension on the rule learning algorithm RIPPER. It then explains several pitfalls encountered by this naive extension during induction. It goes on to describe algorithmic modifications and a new multiple-instance coverage measure which are shown to avoid these pitfalls. Experimental results show the benefits of this approach for solving propositionalized relational problems in terms of speed and accuracy.

keywords: Multiple-instance learning problem, rule learning, propositionalization, relational learning, mutagenesis learning task

1 Introduction

In most ML applications, the choice of knowledge representation for a learning example is between a fixed-length "feature vector" and a first-order representation, be it a logic formula as in structural learners (ML-SMART (Bergadano & Giordana, 1988)) or a closed Horn clause as in inductive logic programming (FOIL(Quinlan, 1990), PROGOL(Srinivasan & Muggleton, 1995)). The motivation for using first-order representation is that it is the natural extension to propositional representation.

However, a known drawback of using first-order logic is that its expressivity is so high that in order to learn efficiently, strong biases such as *determinacy*, are often required on the hypothesis space. Giordana *et al.* have recently shown that there is a phase transition in relational learning (Giordana et al., 2000) linked to the exponential complexity of matching. They argued that relational learners could hardly search in practice for target concepts having more than four non-determinate variables (Giordana et al., 2000).

There have been several attempts to extend Attribute/Value representation rather than directly using first-order logic based representation. In particular, Cohen has proposed the use of *set-valued attributes* that have been implemented in RIPPER (Cohen, 1995). Cohen has shown the usefulness of such representation on text categorization problems. Multiple-instance representation, where each example is represented by a "bag" of fixed-length "feature vectors" (Dietterich et al., 1997), is a more general extension that offers a good tradeoff between the expressivity of relational learning and the low complexity of propositional learning. Data represented as bags of vectors may either be found naturally in chemical domains (Dietterich et al., 1997), in images classification tasks (Maron & Ratan, 1998), or be produced after non-determinate propositionalization of first-order representation (Zucker & Ganascia, 1998; Alphonse & Rouveirol, 1999).

Much work has been done on multiple-instance learning. Unfortunately, available multiple-instance learners are not able to efficiently generate rule sets or decision trees. This paper proposes a generic extension to propositional rule learners to handle multiple-instance data. This extension is implemented in the RIPPER rule learner. Several pitfalls encountered by this naive extension called NAIVE-RIPPERMI are then characterized before showing that a modification of the refinement procedure as well as a new multiple-instance probabilistic coverage measure implemented in RIPPERMI-REFINED-COV avoid these pitfalls. Experiments on artificial datasets are used to validate these improvements. The last section presents experiments on relational data and shows the benefits of a multiple-instance learner for relational learning. As our algorithms generate rule sets, it is possible to use them on relational learning problems reformulated into multiple-instance learning tasks, to generate first-order rules. NAIVE-RIPPERMI and RIPPERMI-REFINED-COV are compared to two popular relational learners on the mutagenesis prediction problem.

2 The Multiple Instance Learning Problem

The multiple instance (MI) learning problem was first introduced by (Dietterich et al., 1997). In this section, it will be formally defined as well a language bias traditionally used in MI learners (Dietterich et al., 1997; Auer, 1997; Blum & Kalai, 1998; Maron & Ratan, 1998). Finally, related work will be briefly reviewed.

2.1 Definition and Notation

In the traditional setting of machine learning, an object is represented by a feature vector x , to which is associated a label $f(x)$. Let \mathcal{X} be a feature vector space, and \mathcal{Y} the finite set of labels or classes. For the sake of simplicity, we will restrict ourselves to the two-class case, i.e. $\mathcal{Y} = \{\oplus, \ominus\}$. The goal then, typically, is to find a classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$ which minimizes the probability that $f(x) \neq h(x)$ on a newly observed example $(x, f(x))$.

Within the multiple instance framework, objects are represented by *bags of vectors* of variable size. Vectors are also called *instances*. As in the traditional setting, they can contain numeric as well as symbolic features. The size of a bag b is noted $\sigma(b)$. Its instances are noted $b_1 \dots b_{\sigma(b)}$. The multiple instance induction task consists of finding a classifier $H : 2^{\mathcal{X}} \rightarrow \mathcal{Y}$, which accurately predicts the label $F(b)$ ¹.

The multiple instance learning problem has been associated to a bias introduced by (Dietterich et al., 1997), which will here be referred to as the *single-tuple bias*, as opposed to the multi-tuple bias proposed by (Raedt, 1998). It can be formally defined as follows:

Definition 1 *The single-tuple bias is a restriction on the set of functions $H : 2^{\mathcal{X}} \rightarrow \{\oplus, \ominus\}$ to those for which there exists a function $h : \mathcal{X} \rightarrow \{\oplus, \ominus\}$ such that $H(b) \equiv \exists i, h(b_i)$.*

The underlying idea is that for certain learning tasks, if a bag is labeled positively, then at least one of its instances must be responsible for this.

This idea can be particularly well illustrated in an example on which this bias has been extensively used: the task of predicting whether molecules smell musky or not (Dietterich et al., 1997). Dietterich chooses to represent molecules as bags of vectors, each vector describing a steric configuration of the molecule. It is well known by chemists that a molecule is musky iff at least one of its configurations has given properties, which make the entire molecule smell musky. Thus, there exists a function h representing these properties, such that the function $H(b)$ - which is derived from h as shown earlier - is an accurate classification function for this learning task.

2.2 Related Work

Previous work on learning from multiple-instance examples has focused on the problem of learning axis-parallel rectangles (APR) under the single-tuple bias. In particular, (Dietterich et al., 1997) has designed APR algorithms to solve the task of predicting whether a molecule is musky or not. Other APR algorithms such MULTIINST (Auer, 1997) have been tested on this learning task, and many interesting learnability results have been obtained (Blum & Kalai, 1998; Auer et al., 1997). More recently, (Maron & Ratan, 1998) proposed a new multiple-instance algorithm called *Diverse Density*, which they applied to image classification. Finally, the lazy learning approach to multiple-instance learning has been investigated in (Wang & Zucker, 2000). Note that the algorithms mentioned here do not generate interpretable hypothesis such as rule sets. The following section presents a method for inducing multiple-instance rules with a modified traditional rule learner.

¹Note that functions on the instance space (resp. bag space) will be noted lower case (resp. upper case).

3 Extending a propositional learner

3.1 Motivation

This section presents a method for the extension of a propositional learner to handle multiple-instance data using the single-tuple bias. Our choice to adapt a propositional learner instead of designing new multiple-instance algorithms is justified by the three following points. First, the two learning problems are very similar. In fact, multiple-instance data can easily be represented by a single set of vectors, and under the single-tuple bias, the multiple-instance and the mono-instance search spaces are identical. Thus, the extension may be simple. Secondly, the existing multiple-instance learners (Dietterich et al., 1997; Auer, 1997; Maron & Ratan, 1998) do not generate interpretable rules or decision trees. Note that an MI learner able to generate rule sets can be used to solve relational learning problem with an appropriate reformulation algorithm such as REPART (Zucker & Ganascia, 1998). This will be detailed in the final section. Finally, propositional learning is a mature field, and many of the available algorithms can efficiently handle large databases, while achieving low error rates. The extension of such a learner to the multiple-instance case will thus benefit from this maturity.

Extending a decision tree induction algorithm to the multiple-instance case raises several algorithmic problems, due to the *separate-and-conquer* aspect of the learner. These issues are beyond the scope of this paper and will be addressed elsewhere. Fortunately, these problems are not encountered in the rule learning coverage algorithms. We have therefore chosen to propose an extension of propositional rule learners using a coverage algorithm.

3.2 A single-tuple naive extension of RIPPER

Let us now study the modifications needed by a mono-instance (i.e. traditional) rule learning algorithm in order to incorporate the single-tuple bias. Let us consider a generic mono-instance rule learner using a coverage algorithm. It can be seen as an algorithm iteratively searching for a function h under a given bias, such that this function will minimize a given criterion related to the prediction error of h on the training dataset D . This criterion varies from one algorithm to another. RIPPER (Cohen, 1995) and C4.5 both use a criterion based on the information gain. To compute the value of this criterion, the learners first evaluate $count(h, D, \oplus)$ and $count(h, D, \ominus)$, which denote the number of positive (resp. negative) examples from D covered by h .

In order to adapt a mono-instance learner to the multiple instance case, we first need to transcribe a multiple instance bag set into a simple set of vectors, without any loss of information. This can be done by adding to each instance two attributes, the first one named *bagid* identifying the bag it belongs to and the second one named *bagclass* encoding the class of its bag.

After having done this, we must now modify the evaluation criterion, such that $H(b) = \exists i, h(b_i)$ is evaluated instead of h . To do so, we will replace the function $count(h, D, c)$ by $count_{single-tuple}(h, D, c)$ evaluating the number of bags of class c encoded in D covered by H . Note that because of the single-tuple bias, if h covers a single vector x , then the bag identified by $bagid(x)$ will be considered as covered by H . Thus, we have:

$$count_{single-tuple}(h, D, c) = |\{bagid(x); x \in D \wedge h(x) \wedge bagclass(x) = c\}|$$

We have chosen to implement these modifications in RIPPER, a fast efficient rule learner designed by (Cohen, 1995) which has been shown to be as accurate as C4.5 on classical

datasets. In addition, the rule sets induced by RIPPER are usually very short, thus being easily interpretable. RIPPER includes several functionalities such as pruning and rule optimization, which also had to be adapted to handle single-tuple hypotheses. The rule refinement strategy of RIPPER consists in greedily adding the best literal without any backtracking. The optimization phases are thus important to improve the accuracy of the rules induced. The resulting algorithm, which we call NAIVE-RIPPERMI, inherits most of RIPPER’s qualities, such as the ability to efficiently handle large datasets in nearly linear time with the number of examples and the number of features.

3.3 Evaluating NAIVE-RIPPERMI

In order to compare NAIVE-RIPPERMI to the other multiple-instance learners, we chose to run experiments on the `musk1` dataset, already presented in section 2.2. For a detailed description of the dataset, see (Dietterich et al., 1997). Table 1 presents the results of NAIVE-RIPPERMI measured using a tenfold cross-validation on the data.

Learner	Accuracy	Theory
ITERATED-DISCRIM	0.92	APR
DIVERSE DENSITY	0.89	Points in \mathcal{X}
NAIVE-RIPPERMI	0.88	rule set
TILDE	0.87	rule set
ALL-POS-APR	0.80	APR
MULTIINST	0.77	APR

Table 1: Compared accuracy of NAIVE-RIPPERMI with other learners on the `musk1` dataset.

The hypotheses generated by NAIVE-RIPPERMI contain an average of seven literals. This is primarily due to efficient pre-pruning and post-pruning techniques implemented in RIPPER. In addition, the average induction time is less than six seconds on a PowerPC G3 300MHz computer. The two algorithms which are here more accurate than NAIVE-RIPPERMI generate models which are not directly interpretable. Both ITERATED-DISCRIM APR and ALL-POS-APR have been specifically designed for this learning task (Dietterich et al., 1997). In contrast, the ILP algorithms such as TILDE which give comprehensible theories are slower than our learner.

Results obtained on the `musk2` dataset are not as good as those presented here on the `musk1` dataset. In the former, the average number of instances per bag is much bigger than in the latter. More precisely, during cross-validations, NAIVE-RIPPERMI generates from the `musk2` dataset concise hypotheses achieving low error rates on the training data, but whose error rates on the test data are significantly higher. This may be due to the large number instances and attributes, which causes some non predictive hypotheses to be consistent with the training data. Attribute selection or instance selection techniques may be useful to overcome this problem.

In the following, the relation between consistency on training data and predictive power will not be addressed. Instead, we will focus on the ability of NAIVE-RIPPERMI to find consistent hypotheses. Considering that NAIVE-RIPPERMI is a simple MI extension of an optimized mono-instance algorithm, it is likely to be sub-optimal. For example the greedy search procedure of RIPPER may not be adapted to finding consistent MI hypotheses on datasets

containing many instances. In the following section, the behavior of our algorithm will be analyzed carefully on artificial datasets, in order to design improvements.

4 Analysis of RIPPERMI algorithms

The purpose of this section is to analyze and to understand the behavior of the algorithm presented earlier as NAIVE-RIPPERMI. This analysis will enable us to discover potential drawbacks, which we will try to solve. The following questions will guide our research. When the number of instances is equal to one, NAIVE-RIPPERMI is equivalent to RIPPER; how, therefore, does the algorithm react when the number of instances increases? Is the search procedure of NAIVE-RIPPERMI adapted to large numbers of instances? When does the algorithm fail to induce a theory? Considering that a multiple-instance learner can be viewed as a biased ILP learner (Raedt, 1998), how well does an ILP algorithm compare to ours?

To answer these question, we need datasets on which all is known, in order to run several experiments. We have therefore decided to design an artificial dataset generator. The following subsection presents the generation of these datasets and their use.

4.1 Validation protocol using artificial datasets

In order to validate and test the multiple instance abilities of NAIVE-RIPPERMI, we constructed an artificial dataset generator which builds MI datasets according to parameters provided by the user. As stated above, we were primarily interested in understanding the behavior of our algorithm as the number of instances per bag increases. For this reason, we measured the accuracy of our algorithms on several randomly generated datasets having a given number of instances per bag.

Each artificial dataset contains 100 bags, a given number of instances, and 12 boolean attributes. The target concept is a boolean function combining 3 attributes out of 12. The distribution of the values of each attribute for each label is chosen randomly by the artificial dataset generator. Thus, some attributes are likely to be correlated with the bag label without being part of the target concept. The decision to use boolean attributes to validate our algorithm was intended to focus only on the multiple instance aspect of NAIVE-RIPPERMI, without taking into account its capability of handling numerical attributes.

The different MI extensions of RIPPER described in this paper were run on these datasets with the default parameters, which consist of two optimization passes each followed by a pruning phase. Hundreds of datasets containing a given number k of instances per bag were generated; then, the accuracy of each algorithm was measured by averaging cross-validations over these datasets. The average classification error is plotted on figure 1, as well as the corresponding learning time using a log-scale. For example, on datasets containing 15 instances per bag, NAIVE-RIPPERMI obtains an average classification error rate of 26.5%, and the induction phase lasts less than a third of a second on a PowerPC G3 300 MHz computer.

The ILP learner FOIL was also run on these datasets in order to evaluate the ability of an ILP tool on multiple-instance data. The top curve in both figures shows its accuracy and induction time with various numbers in instances per bag. On this particular task, it is outperformed by NAIVE-RIPPERMI both in terms of speed and accuracy.

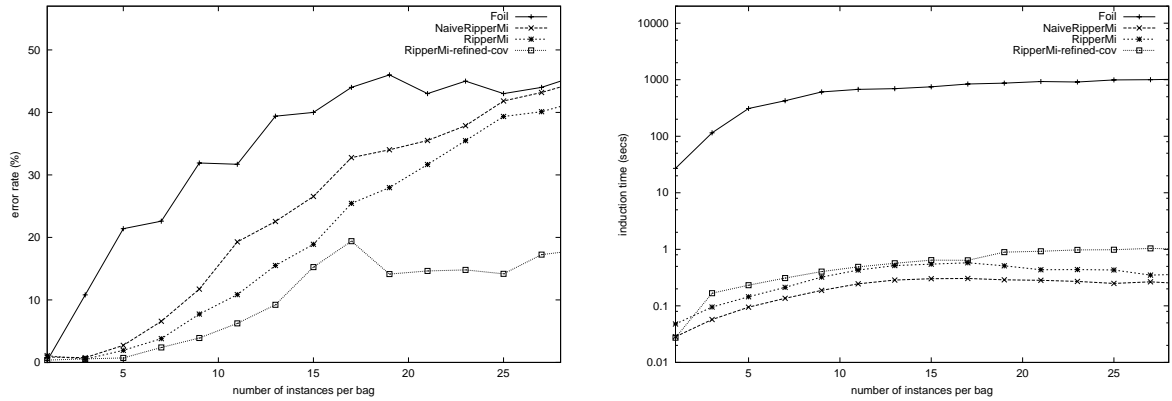


Figure 1: Classification error rate (left figure) and induction time (right figure) of FOIL, NAIVERIPPER-MI, RIPPERMI, and RIPPERMI-REFINED-COVERAGE on artificial datasets with various numbers of instances per bag

4.2 Avoiding pitfalls during induction

A careful examination of the theories induced on the artificial datasets revealed various drawbacks of NAIVERIPPERMI. Let $x_j(b_i)$ denote the j^{th} attribute of the instance b_i . For the sake of simplicity, multiple-instance rules of the form $H(b) \equiv \exists i, x_1(b_i) = 0 \wedge x_2(b_i) = 1 \wedge \dots \wedge x_j(b_i) = 0$ will be noted as $(x_1 = 0) \wedge (x_2 = 1) \wedge \dots \wedge (x_j = 0)$. To illustrate these drawbacks, let us consider the four bags shown in table 2. The target concept is $(x_1 = 1) \wedge (x_2 = 0) \wedge (x_3 = 1)$. NAIVERIPPERMI's strategy to refine a rule will be to examine each possible literal, and to add the one which brings the highest gain. Here, starting with an empty rule, the candidate rules $(x_1 = 1)$, $(x_2 = 0)$, and $(x_3 = 1)$ each cover all four bags, $(x_1 = 0)$ and $(x_3 = 0)$ both cover one positive and two negative bags, and $(x_2 = 1)$ covers two positive and one negative. Thus the best literal to start with, in terms of information gain, is $(x_2 = 1)$. This literal is clearly *incompatible with the target concept*. Thus it is impossible to obtain the correct concept by successive refinements. Note that this is a typical multiple-instance phenomenon. In mono-instance cases, literals which increase the information gain cannot contradict the target concept.

bag	class	x_1	x_2	x_3
bag1	\oplus	1	0	1
		1	1	0
bag2	\oplus	0	1	1
		1	0	1
bag3	\ominus	0	0	0
		1	1	1
bag4	\ominus	0	0	1
		1	0	0

Table 2: Two positive bags and two negative bag, with two instances each. Target: $(x_1 = 1) \wedge (x_2 = 0) \wedge (x_3 = 1)$

To overcome this drawback, a modification of NAIVE-RIPPERMI's search procedure is necessary. Suppose we are refining a rule R which is known not to have fallen in a pitfall

yet. Let ℓ be the best literal to add to R , according to NAIVERIPPERMI’s greedy strategy. Of course, we cannot be sure that $R \cup \ell$ is not in contradiction with the target concept. Instead by considering both $R \cup \ell$ and $R \cup \neg\ell$, at least one of the two rules does not contradict the target concept. The induction process thus undoubtedly avoids the pitfalls. Note that the process of examining two rules at each refinement step can be seen as building a binary decision tree. Hence, the solution proposed here consists in building a decision tree from which a single rule will be extracted. The decision tree may be built using an information gain measure together with the counting function $count_{single-tuple}(h, D, c)$ introduced earlier.

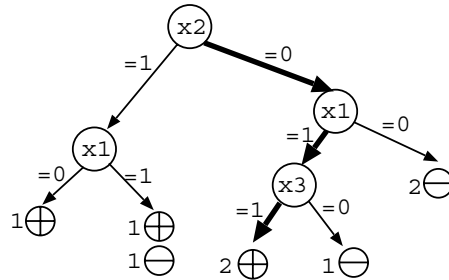


Figure 2: Decision tree induced from bags in table 2

Figure 2 shows the decision tree built from the bags shown in table 2. The leaves of the tree indicate how many positive and negative bags are covered by the corresponding rule. Here, the rule $(x2 = 0) \wedge (x1 = 1) \wedge (x3 = 1)$ covers two positive bags ($2\oplus$) and no negative one. This rule will thus be extracted from the tree.

Much work has been done recently on the use of decision trees as a temporary representation for mono-instance rule induction. Many ideas, such as partial tree induction (Frank & Witten, 1998), could be adapted in our case in order to speed up the result. Nevertheless, as stated by Frank and Witten (Frank & Witten, 1998), in the mono-instance case decision trees are used as a substitute to a global optimization on rule sets. Thus they do not provide a qualitative algorithmic improvement, unlike in the multiple instance case, for which they enable pitfalls to be avoided.

In addition to incompatible literals, attributes correlated with the target concept but not belonging to it are more likely to appear in multiple-instance data than in mono-instance data. Thus, induced theories may contain irrelevant literals anywhere in the rules. In the mono-instance case, irrelevant literals usually appear at the end of the rule, because of overfitting. To avoid this, RIPPER implements a *reduced error pruning technique* which tests and removes literals at the end of rules. We therefore extended the pruning algorithm to a *global pruning*, examining literals in rules in any order.

Using the same validation procedure as earlier, the graphs of figure 1 respectively show the average classification error rate and induction time using this new algorithm, which we will refer to as RIPPERMI. With the dataset containing 15 instances per bag, for example, the classification error rate decreases from 26.5% to 18.9%.

4.3 A more refined coverage measure

Up to now, the coverage measure that has been used in our experiments is the function $cover_{signe-tuple}(h, D, c)$ introduced in section 3. This function returns an integer indicating the number of bags in D of class c covered by H . If a bag b^1 has one of its instances covered

by h , and a bag b^2 has 15 of its instances covered by h , both bags will count exactly the same way. This has severe drawbacks; in our experiment, when the number of instances becomes high, given any rule made of a single literal, the probability that each bag has at least one instance covered by this rule is very high. Thus the learner does not have enough information to choose which literal to add first. Note that even though each bag is covered by the candidate rule, some bags may have more covered instances than others. This section will present a new single-tuple coverage measure exploiting this information.

We will now suppose that there exists function f , such that $F(b) \equiv \exists i, f(b_i)$, with $F(b)$ denoting the label of any bag b . Suppose we succeed in inducing an hypothesis $H(b) \equiv \exists i, h(b_i)$ complete and consistent with the training data. If f and h often disagree on instances of the training data, then it is likely that H often disagrees with F on test data. H will thus have a low predictive power. Thus, inducing complete and consistent hypotheses is not sufficient. The associated functions h must cover at least one instance covered by f per positive bag. Because we do not have access to f , the previous requirement cannot deterministically be guaranteed. Hence, we will evaluate its probability. Let b be a positive bag containing σ instances. Let k be the number its instances covered by h . Let $R = \{b_{i1} \dots b_{ik}\}$ be those instances. The probability that at least one instance of b is covered by h and f is: $P(f(b_{i1}) \vee \dots \vee f(b_{ik}))$.

The new coverage measure on positive bags is thus the sum of the above probability on each positive bag. Of course, the coverage measure on negative bags remains the same as before. From now on, we will need to assume that bags were built by drawing instances independently from a distribution \mathcal{D} , and that F was then used to assign labels to these bags. Let q be the index of an instance of b chosen randomly from all instances of b covered by f . The above probability becomes:

$$\begin{aligned} &= P(b_q \in R) \times P(f(b_{i1}) \vee \dots \vee f(b_{ik}) \mid b_q \in R) \\ &+ P(b_q \notin R) \times P(f(b_{i1}) \vee \dots \vee f(b_{ik}) \mid b_q \notin R) \end{aligned} \quad (1)$$

Clearly, the first term is equal to $\frac{k}{\sigma}$. The first probability of the second term is thus $1 - \frac{k}{\sigma}$. Note that in a positive bag at least one instance fires f . Thus, the distribution of instances in positive bags differs from \mathcal{D} . Suppose that an instance firing f is removed from each positive bag; then the distribution of instances in these bags becomes \mathcal{D} . Hence, when $b_q \notin R$, we have $P(f(b_{ij})) = P_{\mathcal{D}}(f)$. Thus, the equation becomes:

$$= \frac{k}{\sigma} + \left(1 - \frac{k}{\sigma}\right) \times (1 - P_{\mathcal{D}}(\neg f)^k) \quad (2)$$

$$= 1 - \left(1 - \frac{k}{\sigma}\right) \times P_{\mathcal{D}}(\neg f)^k \quad (3)$$

This refined probabilistic coverage function has the following desired properties: if $k = 0$, the probability is null, and if $k = \sigma$, it is equal to one. Note that $P_{\mathcal{D}}(f)$ is still unknown. To evaluate it, our algorithm generates a first hypothesis h , and computes the total number of instances covered by h . This number is divided by the total number of instances and is used as an approximation to $P_{\mathcal{D}}(f)$.

This measure was implemented in RIPPERMI (thereby referred to as RIPPERMI-REFINE-COV). The results can be seen in figure 1. This refined measure prevents the accuracy from decreasing suddenly (when the number of instances increases), as it did with the earlier algorithms. For reasons mentioned earlier results on the musk datasets were not significantly improved by these new algorithms.

5 Experiments on relational data

In order to validate the above algorithm, we have begun investigating a new application of multiple-instance learning on relational data. It has been shown that under various biases, the problem of learning from first-order data can be converted to a lower-order learning problem, in particular to attribute-value learning tasks. To do so, the relational data are *propositionalized*: each first-order example is reformulated into a single feature vector and an attribute-value learner is used to induce hypotheses. Many popular ILP algorithms have been using this approach (Sebag & Rouveirol, 1997; Lavrac et al., 1991). Alternatively, another type of propositionalization (which will be referred to as *MI-propositionalization*) has been investigated in (Zucker & Ganascia, 1998), consisting in reformulating the first-order examples into multiple-instance bags. The main advantage of this technique compared to classical propositionalization is that bags of instances are more adapted to representing non-determinate predicates. In this section NAIVE-RIPPERMI and RIPPERMI-REFINED-COV will be used in association with REPART (Zucker & Ganascia, 1998) to solve a traditional ILP problem: the mutagenesis prediction task (Srinivasan & Muggleton, 1995).

5.1 The Mutagenesis Prediction Problem

The mutagenesis prediction problem (Srinivasan & Muggleton, 1995) consists in inducing a theory which can be used to predict whether a molecule is mutagenic or not. To achieve this, a dataset describing 230 molecules with prolog facts is available. In the following, a subset of 188 molecules known as being *regression-friendly* will be used. This dataset contains highly non-determinate predicates that many early relational algorithms like LINUS (Lavrac et al., 1991) could not handle. Also, it is frequently used as a benchmark problem to compare ILP algorithms. Several relational descriptions of the domain are available. In the simplest description termed \mathcal{B}_0 (Srinivasan & Muggleton, 1995), the background predicates describing molecules are $\text{atm}(m, d, e, t, c)$ and $\text{bond}(m, d1, d2, bt)$, where d identifies an atom of the molecule m , where e, t, c represent the element, type and partial charge of an atom respectively, and $d1, d2$ identify atoms linked by a bt type bond. Another description termed \mathcal{B}_2 will also be considered here. The latter is composed of \mathcal{B}_0 to which four predicates describing global properties of the molecules such as their hydrophobicity have been added.

5.2 Multiple Instance propositionalization

The algorithm REPART (Zucker & Ganascia, 1998) has been used to generate several MI-propositionalizations. After each MI-propositionalization, the MI learner is launched on the reformulated data, and the process stops if the accuracy of the induced theory is sufficient. If

	\mathcal{B}_0	\mathcal{B}_2
RIPPERMI-REFINED-COV	0.82	0.91
NAIVERIPPERMI	0.78	0.91
PROGOL	0.79	0.86
FOIL	0.61	0.83

Table 3: Compared accuracy of RipperMi with other ILP learners on the *regression-friendly* mutagenesis dataset.

not, another more complex reformulation is chosen, and so forth. Using the description \mathcal{B}_0 , REPART first represents molecules as bags of atoms. Thus, each instance contains the three attributes describing a single atom. As expected, this reformulation did not yield good results. During the second step, REPART represented molecules as bags of pairs of bonded atoms. The following subsection describes the results using this reformulation. With the \mathcal{B}_2 description level, the first reformulation chosen by REPART has shown to be sufficient. This reformulation consisted in representing each molecule as a bag of atoms each to which was added global molecular properties.

5.3 Experiments and Results

The results of NAIVE-RIPPERMI and RIPPERMI-REFINED-COV are compared to those of PROGOL (Srinivasan & Muggleton, 1995) and FOIL (Quinlan, 1990). Table 3 shows the accuracy of these learners measured with a tenfold cross-validation. Both NAIVE-RIPPERMI and RIPPERMI-REFINED-COV perform equally well on the \mathcal{B}_2 description, which is not surprising, because most literals added to the induced theories are global literals. Therefore, their multiple-instance ability is not challenged here. On the other hand, the reformulation using the \mathcal{B}_0 description level does not contain any global attributes. This explains the higher accuracy obtained by RIPPERMI-REFINED-COV compared to that of NAIVERIPPERMI. 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.

Both MI learners are faster than ILP algorithms. For example, on the \mathcal{B}_0 description level, NAIVERIPPERMI induces an hypothesis less than 15 seconds on a PowerPC G3 300 MHz. In comparison, PROGOL requires 117039 seconds, and FOIL requires 4950 seconds.

6 Conclusion

The problem of supervised multiple-instance learning is a recent learning problem which has raised interest in the machine learning community. This problem is encountered in contexts where an object may have several alternative vectors to describe its different possible configurations. Solving multiple-instance problems using propositional algorithms raises subtle issues that are related to the notion of bags of instances whose coverage is by essence different from that of mono-instance problems. We have proposed an method to extend a propositional rule learning algorithm to the multiple-instance case. Some drawbacks of this method have been detected and a more refined search procedure has been developed. In addition, a probabilistic coverage measure has been proposed, which purpose is to take into account the *number* of instances per bag covered by the candidate rules, in order to guide induction more precisely. All these refinements have been validate on artificial datasets.

With the help of the REPART algorithm, which reformulates first-order examples into bags of instances, our algorithm has been tested on the well known mutagenesis relational dataset. RIPPERMI-REFINED-COV yielded good results compared to those of FOIL and PROGOL on this problem. In addition, it showed to be significantly faster. We therefore argue that the multiple instance paradigm may be very useful for solving a wide range of relational problems.

Some multiple-instance learning tasks may require more powerful bias than single-tuple. In particular, hypotheses taking into account the number of instances per bag having a given

property may prove useful. Future work includes proposing efficient algorithms for these biases.

References

- Alphonse, E., & Rouveirol, C. (1999). Selective propositionalization for relational learning. *PKDD*.
- Auer, P. (1997). On learning from multi-instance examples: Empirical evaluation of a theoretical approach. *Proc. 14th International Conference on Machine Learning*. Morgan Kaufmann.
- Auer, P., Long, P., & Srinivasan, A. (1997). Approximating hyper-rectangles: Learning and pseudo-random sets. *Proc. of 29th Annual ACM Symposium on Theory of Computing*.
- Bergadano, F., & Giordana, A. (1988). A knowledge intensive approach to concept induction. *Fifth International Conference on Machine Learning*.
- Blum, A., & Kalai, A. (1998). A note on learning from multiple-instance examples. *Machine Learning*, 30.
- Cohen, W. W. (1995). Fast effective rule induction. *Proc. 12th International Conference on Machine Learning*. Morgan Kaufmann.
- Dietterich, T. G., Lathrop, R. H., & Lozano-Pérez, T. (1997). Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89.
- Frank, E., & Witten, I. H. (1998). Generating accurate rule sets without global optimization. *Proc. 15th International Conf. on Machine Learning*.
- Giordana, A., Saitta, L., Sebag, M., & Botta, M. (2000). Analyzing relational learning in the phase transition framework. *Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA.
- Lavrac, N., Dzeroski, S., & Grobelnik, M. (1991). Learning non recursive definitions of relations with linus. *European Working Session on Learning'91*.
- Maron, O., & Ratan, A. L. (1998). Multiple-instance learning for natural scene classification. *Proc. 15th International Conf. on Machine Learning* (pp. 341–349). Morgan Kaufmann, San Francisco, CA.
- Quinlan, J.-R. (1990). Learning logical definitions from relations. *Machine Learning*, 239–266.
- Raedt, L. D. (1998). Attribute-value learning versus inductive logic programming: The missing links. *Proc. 8th International Conference on ILP*. Springer-Verlag.
- Sebag, M., & Rouveirol, C. (1997). Tractable induction and classification in first order logic. *IJCAI*. Nagoya, Japan.
- Srinivasan, A., & Muggleton, S. (1995). Comparing the use of background knowledge by two ilp systems. *ILP-95*. Katholieke Universiteit Leuven.
- Wang, J., & Zucker, J.-D. (2000). Solving multiple-instance problem: a lazy learning approach. *International Conference on Machine Learning*.
- Zucker, J.-D., & Ganascia, J.-G. (1998). Learning structurally indeterminate clauses. *Proc. 8th International Conference on ILP*. Springer-Verlag.