
Noise-Tolerant Rule induction from Multi-Instance data

Yann Chevalere

Jean-Daniel Zucker

LIP6-CNRS University Paris VI, 4, place Jussieu, F-75252 Paris Cedex 05, France

YANN.CHEVALEYRE@LIP6.FR

JEAN-DANIEL.ZUCKER@LIP6.FR

Abstract

This paper addresses the issue of multiple-instance induction of rules in the presence of noise. It first proposes a multiple-instance extensions of rule-based learning algorithms. Then, it shows what kind of noise can appear in multiple-instance data, and how to handle it theoretically. Finally, it describes the implementation of such a noise-tolerant multiple instance learner, and shows its performance on several problems, including the well-known mutagenesis prediction task.

1. Introduction

The multiple instance learning problem has received growing attention among other fields of supervised machine learning, since Dietterich et al. (97) solved it by building axis-parallel rectangles. In their article, they applied it to the task of classifying aromatic molecules according to whether they were "musky" or not. What motivated their choice of using the multiple-instance representation was that to each molecule corresponded several steric configurations. Thus, by coding a configuration with a numerical feature vector, a molecule was represented by a bag of such vectors, called "instances". Chemistry is a domain par excellence where these multiple-instance (abbreviated MI) data are to be found.

Later researches concentrated on showing the PAC learnability of the MI learning problem, under strong assumptions such as statistical independence between instances, and hypothesis restricted to hyper-rectangles. Blum et Kalai (98) proved that the MI problem could be solved by a noise-tolerant propositional learner, without the restrictions of having numerical instance or hyper-rectangle theories.

Recently, De Raedt (98) showed in which way the MI problem could be seen as a bias on inductive logic programming. In addition, he suggested that the MI paradigm could be the key between the propositional and

relational representations, being more expressive than the former, and much more easy to learn than the latter.

Thus, in addition to being a particularly suitable representation for some domains such as chemistry, the MI representation is of a great interest for ILP : propositional learners were often used to solve ILP problems by propositionalizing them in some way, which generally wasn't satisfying for non-determinate domains; much closer to ILP and intrinsically non-determinate, the use of MI representation to solve an ILP problem is a very promising issue. Zucker et Ganascia (96,98) presented REPART, an ILP system based on an ingenious bias which firstly reformulates the relational examples in a MI database, and then induces the final hypothesis with a MI learner.

Another interesting issue, not yet exploited according to the author's knowledge, is the use of MI learners on propositional databases. For this purpose, one must define a criteria which will determine how to group propositional examples together to form bags of instances. This could be viewed as interactive learning: by specifying the grouping criteria, the user directs the knowledge discovery process.

Finally, it seems likely that researches on MI problem will benefit to machine learning in general and ILP in particular, because this new paradigm brings about lots of new ideas, some of which will be transposable to other fields, such as ILP.

To our knowledge, there had not been specific work for handling noise in the Multiple-Instance framework. Both the promising applications of the Multiple-instance problem and the lack of a modelisation of noise motivates the present work. Our goal is to design a general-purpose MI learner, which could be used in ILP systems like REPART as well as for any MI learning task without making any assumption on the data. It ought to be fast, accurate, tolerant to noise, and able to generate interpretable theories. Therefore, we chose to adapt a rule-based propositional learner to handle MI data. In order to improve accuracy, we developed a noise model specific to the MI problem.

Section 1 will briefly present how to extend a rule-based learner in order to handle MI examples, after exposing the basic concepts and notations of the domain. In Section 2, the problem of noise specific to MI data is analyzed and a solution is proposed. Section 3 describes our implementation of those ideas, as well as some results on traditional learning problems.

2. Extending a propositional learner to handle multiple-instance data

In order to extend propositional learners to the multiple-instance case, we will show that multiple-instance data as well as hypothesis can be represented by propositional data and hypothesis. Then, the inevitable modifications of core functions of the learner will be explained.

2.1 Concepts and notations

In usual propositional framework (which we will also call the mono-instance framework), the learner is provided with a set of n examples $\{x_1 \dots x_n\}$, also called instances, which take the form of attribute-value vectors, as well as a label for each of them $f(x_1) \dots f(x_n)$. Let X be the instance space. Given a language bias, the goal of the learner becomes to induce a function h mapping X onto $\{0,1\}$ ¹, which best approximates f .

In the MI framework, instead of seeing individual labeled points from X , the learner is provided with labeled bags of instances. For the sake of simplicity, we will here suppose that each bag contains exactly r instances². Thus, we will call these bags “*r*-examples”, as in (Blum, Kalai 98). Functions mapping the *r*-example space X^r to $\{0,1\}$ will be written upper-case, and *r*-examples will be represented by a vector. Thus, a labeled *r*-example will be noted :

$$(\vec{x}; F(\vec{x})) = (x_1, \dots, x_r, F(\vec{x}))$$

In (Dietterich et al. 1997), a molecule was considered active iff at least one of its configuration had given properties. This led to a widely used MI bias, which we will formulate as follows : Any function F mapping X^r to $\{0,1\}$ can be reformulated with a function f mapping X to $\{0,1\}$.

$$F(\vec{x}) = f(x_1) \wedge f(x_2) \wedge \dots \wedge f(x_r)$$

The goal of the MI learner is to search for a function H approximating F , which will be greatly simplified by the above restriction.

¹ We restrict ourselves to the two-class learning problem.

² Note that for each result presented here, the generalization to a variable number of instances is straightforward; we implemented it.

2.2 Representing MI data and theories within propositional framework

Due to the language bias presented above, multiple-instance functions can be reformulated as mono-instance functions, used in any propositional learner. Therefore, the search space becomes the same for both paradigms. The task of the MI learner will thus be to explore the space of mono-instance functions h , and to evaluate the accuracy of their corresponding multiple-instance function H .

The task of representing *r*-examples within propositional framework is very easy : each instance will become an example labeled with the label of its original *r*-example. In order to allow the learner to identify *r*-examples, each instance will be added with a new numerical attribute, identifying the *r*-example it belonged to.

For example, let an instance x having two attributes $(att_1(x), att_2(x))$ belong to an *r*-example labeled positively. To represent x in the propositional framework, we will label it and add an attribute *r-ex-id*(x) identifying its *r*-example. It will thus become:

$$(att_1(x), att_2(x), r-ex-id(x) ; \text{positive})$$

2.3 Inducing MI hypothesis

Most learning algorithm use a generate-and-test scheme. In the case of rule-based learners, rules are generally refined one step at a time, and refinements are kept if satisfying a criteria, often base on entropy or coverage.

Having shown that the search space was the same, under the traditional MI bias, for both frameworks, we can tell that the “generate” step of the algorithm, which corresponds to the refinement of the rule for rule-based learners, will be kept as-is. On the contrary, the evaluation of the hypothesis will have to be changed, as we want the learner to evaluate the function H , instead of h .

These hypothesis evaluation function, whether they are entropy-based or not, always rely on the counting of examples: a generated hypothesis will be judged from the number of true positives (covered examples which label was correctly predicted by the hypothesis) and false positives. Without any modification, the counting would occur on instances. To count *r*-examples, we will have to implement the following: First, two functions testing whether the k^{th} *r*-example is well (respectively badly) covered by the hypothesis. Let $\{x_i, f(x_i)\}$ be the set of labeled examples, $lab(H)$ be the predicted class of the hypothesis H , and $cov(h, x_i)$ be a function indicating whether h covers instance x_i .

$$Cov_well(H, k) = \sum_i cov(h, x_i) \wedge f(x_i) = lab(H) \quad r-ex-id(x_i) = k$$

$$Cov_bad(H, k) = \sum_i cov(h, x_i) \wedge \neg f(x_i) \quad r-ex-id(x_i) = k$$

Now that we can determine whether H covers a *r*-example, simply by knowing its corresponding mono-instance

function h , the counting of covered r -examples is straightforward

$$\text{Positively_covered} = |\{k, \text{Cov_well}(H,k)\}|$$

In the case we are learning with more than two classes, conflicts may appear in the classification of an r -example: what if two instances of the same r -example are classified differently? Among different possible methods, Ruffo (2000) chose to combine the predicted classes of each instance to predict the class of the r -example. In our MI-learner, such conflicts never arise, as we chose to induce decision-lists; our policy is: the first rule of the list covering one of the instances of the r -example will predict its class.

Note that it is easy to compute the coverage of each r -example by examining each instance only once. Thus, the complexity of the extended learner relatively to the number of instances and attributes isn't changed by this extension.

3. A multiple-instance noise model

In this section, we will present a new noise model specifically designed for multiple-instance induction. First, we will analyze what kind of noise can occur specifically in multiple-instance data, and why. Then, we will focus on the theoretical modeling of that noise. Therefore, Statistical Queries, which is a useful tool for modeling noise, will be briefly introduced, after which our noise model will be derived. Finally, we will discuss about how to implement this model in a learning algorithm.

3.1 Types of noise on multiple-instance data

The two most widely studied noise models in propositional learning are CN – label noise – and AN – attribute noise – (Decatur 1995). The former, class noise, supposes that a given ratio of all labels have been flipped randomly. The latter model, attribute noise, concerns data for which attributes have also been flipped randomly. Both model could apply to the multiple instance framework without much modifications.

In addition to class and attribute, a third entity vulnerable to noise appears in the MI framework: the instance. Many types of noise can thus be imagined: on noise-free MI-data, instances could be randomly removed (resp. added) by the noise process. In the musk problem, one can easily imagine that an important configuration of a molecule could be missing, because occurred to rarely.

A third possible noise process would consist of randomly exchanging an instance from the data with an instance drawn from a noise-distribution. In the musk problem, this could correspond to a failing instrument, randomly replacing a measured configuration with noise. Unfortunately, this model would require the knowledge of the noise distribution. To bypass this problem, we will

suppose that the noisy instances have the same distribution as the noise-free instances. This model, which we will call MIN_η (Multiple Instance Noise model, with noise level η) will be detailed in further sub-sections.

3.2 Statistical Queries

The SQ model – statistical queries – (Kearns,93), will be useful to formalise our new noise model. Instead of apprehending the learning process as inducing hypotheses directly from examples, the statistical queries model can be viewed as learning from global statistical properties on the examples. This learning model has two main advantages: at first, the formalisation of noise is much easier, compared to using the standard PAC framework, and at second, most of the induction tools, like C4.5, incrementally construct their hypotheses using global statistics on the data, which can be viewed as SQ. Thus, any result formalised with SQ will be easily adapted to real induction tools using the same principle. Proving PAC-learnability will not be our purpose here, as this was extensively studied within the multiple-instance context by other authors.

Let \mathcal{D}^r , be the distribution of r -examples. Let $F : X^r \rightarrow \{0,1\}$, be the target concept. In the standard PAC learning model (Valiant,84) applied to our context, the learner is given access to an oracle $EX(F, \mathcal{D}^r)$ which, on each call, will draw an r -example \mathbf{x} randomly and independently according to \mathcal{D}^r , and return the labelled r -example $(\mathbf{x}, F(\mathbf{x}))$

Later, we will need to suppose the independence assumption between instances. With this assumption, it becomes possible to relate the distribution of instances \mathcal{D} , to the distribution of r -examples \mathcal{D}^r , which is thus derived from the former by a cross product.

Let $f : X^r \times \{0,1\} \rightarrow \{0,1\}$ be a predicate on labelled r -examples. A statistical query will be a request for an approximation of the value of $P = \Pr_{\mathbf{x} \sim \mathcal{D}^r} [f(\mathbf{x}, F(\mathbf{x}))] = \Pr_{EX(F, \mathcal{D}^r)} [f = 1]$.

According to our multiple instance noise model (MIN_η), we will define a new oracle, which will return a noisy r -example: $EX^{MIN_\eta}(F, \mathcal{D}^r)$.

Our goal will be to relate the value of P to that of $P^{MIN_\eta} = \Pr_{EX^{MIN_\eta}(F, \mathcal{D}^r)} [f = 1]$. Nevertheless, we will not study the sample complexity, i.e. how many calls to the oracle are needed to estimate these probabilities.

To reflect the hypothesis language bias, and for simplicity, we will restrict the predicates used in SQ to be of the following form:

$$\begin{aligned} f(\mathbf{x}, F(\mathbf{x})) &= H(\mathbf{x}) \quad [F(\mathbf{x}) = l] \\ &= [h(x_1) \dots h(x_r)] \quad [F(\mathbf{x}) = l] \end{aligned}$$

3.3 The noise model

Given the noise-free oracle $EX(F, \mathcal{D}^r)$, we will now define a noisy oracle. On each call, $EX^{MIN\eta}(F, \mathcal{D}^r)$ will first draw a labelled r -example $(\vec{x}, F(\vec{x}))$ from $EX(F, \mathcal{D}^r)$; it will return the r -example $(\vec{x}^*, F(\vec{x}^*))$ such that for each $i \in \{1..r\}$, with a probability $1 - \eta$, $x_i^* = x_i$, and with a probability η , x_i^* will be randomly re-drawn from the distribution \mathcal{D} .

Thus, the label of the noise-free r -example \vec{x} is kept in its noisy counterpart \vec{x}^* , even if $F(\vec{x}) \neq F(\vec{x}^*)$. Also, a proportion η among all instances of noisy r -example may now be considered as non-informative, because, absolutely not related to the label. Note that our model doesn't make any assumption on the function F .

Notation: From now on, we will omit to mention the oracles in the probability calculus. \vec{x} will represent a noise-free r -example drawn by $EX(F, \mathcal{D}^r)$, \vec{x}^* will represent a r -example drawn by $EX^{MIN\eta}(F, \mathcal{D}^r)$, and x_i (resp. x_i^*) will be the i^{th} instance of \vec{x} (resp. \vec{x}^*). Thus, in probability $\Pr[H(\vec{x}) | H(\vec{x}^*)]$, \vec{x} and \vec{x}^* represent a r -example drawn by $EX(F, \mathcal{D}^r)$, and its noisy counterpart

To calculate P , we now define, for each k , the new predicate H_k as : $H_k(\vec{x}) = 1$ iff $|\{i \in \{1..r\} \mid h(x_i) \text{ is true}\}| = k$. Let $\Pr_k(\vec{x})$ be $\Pr[H_k(\vec{x})]$.

Note that $\Pr_k = \Pr_{k-1} + \Pr_k$ and that $\{\Pr_0, \dots, \Pr_r\}$ is a partition on $X^r \times \{l\}$. Thus, we can write the following :

$$\Pr[(\vec{x}, F(\vec{x}))] = \sum_{k=0..r} \Pr_k(\vec{x}) \times \Pr_k(\vec{x}^*, F(\vec{x}^*))$$

which we can re-write as follows :

$$P = \sum_{k=0..r} \Pr_k(\vec{x}, F(\vec{x})) \times \Pr_k(\vec{x}^*, F(\vec{x}^*)) \times P^{MIN\eta}_k \quad (1)$$

Assume that $\Pr_k(\vec{x}^*, F(\vec{x}^*))$ is true. Let $i_1..i_k$ denote the index of the instances of \vec{x}^* for which the predicate h is true, and $i_{k+1}..i_r$ denote other instances. During the generation of the noisy data, the instances can be kept as is or re-drawn. Let \neg_j denote the fact that the j^{th} instance of \vec{x} has not been redrawn during the noise process. Therefor, $\Pr_j(\neg_j) = \eta$. Let $\Pr_{i_1..i_k} = \Pr_{i_1} \dots \Pr_{i_k}$. As $\Pr(\neg_j) = 1 - \eta$, note that $\Pr(\neg_{i_1} \dots \neg_{i_k}) = \eta^k$. We can now calculate the terms of equation (1).

$$\begin{aligned} \Pr_k(\vec{x}, F(\vec{x})) &= \Pr_k(\vec{x}^*, F(\vec{x}^*)) \times \Pr_k(\vec{x}, F(\vec{x})) \\ &= \Pr_k(\vec{x}^*, F(\vec{x}^*)) \times \Pr_k(\vec{x}, F(\vec{x})) \times \Pr_k(\neg_{i_1} \dots \neg_{i_k}) \\ &= \Pr_k(\vec{x}^*, F(\vec{x}^*)) \times \Pr_k(\vec{x}, F(\vec{x})) \times \eta^k \end{aligned} \quad (2)$$

We will suppose that $\Pr_k(\vec{x}^*, F(\vec{x}^*))$ and $\Pr_k(\vec{x}, F(\vec{x}))$ are independent³. Thus $\Pr_k(\vec{x}^*, F(\vec{x}^*)) \times \Pr_k(\vec{x}, F(\vec{x}))$ is considered equal to $\Pr_k(\vec{x}, F(\vec{x}))$. Their remains two unknown conditional probabilities in equation (2). The first one is straightforward :

$$\Pr_k(\vec{x}, F(\vec{x})) = \Pr_k(\vec{x}, F(\vec{x})) \times \Pr_k(\neg_{i_1} \dots \neg_{i_k}) = \Pr_k(\vec{x}, F(\vec{x})) \times \eta^k$$

$$\text{Therefor, } \Pr_k(\vec{x}, F(\vec{x})) = \Pr_k(\vec{x}^*, F(\vec{x}^*)) \times \eta^k \quad (3)$$

Concerning the last unknown probability of equation (2):

$$\Pr_k(\vec{x}, F(\vec{x})) = \Pr_k(\vec{x}^*, F(\vec{x}^*)) \times \Pr_k(\neg_{i_1} \dots \neg_{i_k}) = \Pr_k(\vec{x}^*, F(\vec{x}^*)) \times \eta^k$$

For $j \in \{1..k\}$, x_{ij}^* is redrawn, thus, $h(x_{ij})$ is independent from $\Pr_k(\vec{x}^*, F(\vec{x}^*))$. For $j \in \{k+1..r\}$, $\neg h(x_{ij}^*)$ is true. Therefor, $h(x_{ij})$ is true iff x_{ij}^* has not been redrawn.

$$\begin{aligned} \Pr_{j=1..k}[h(x_{ij}) | \Pr_k(\vec{x}^*, F(\vec{x}^*))] &= p_{\mathcal{D}}(h) \\ \Pr_{j=k+1..r}[h(x_{ij}) | \Pr_k(\vec{x}^*, F(\vec{x}^*))] &= \Pr_{j=k+1..r}[h(x_{ij}) | \neg h(x_{ij}^*)] \\ &= \Pr_{j=k+1..r}[h(x_{ij})] = p_{\mathcal{D}}(h) \end{aligned}$$

$$\Pr_k(\vec{x}, F(\vec{x})) = \Pr_k(\vec{x}^*, F(\vec{x}^*)) \times \eta^k \times p_{\mathcal{D}}(h)^{r-k} \quad (4)$$

Equation (2), using the results (3) and (4), now becomes :

$$\begin{aligned} \Pr_k(\vec{x}, F(\vec{x})) &= \Pr_k(\vec{x}^*, F(\vec{x}^*)) \times \eta^k \times p_{\mathcal{D}}(h)^{r-k} \\ &= (1 - \eta)^k \times [1 - p_{\mathcal{D}}(h)]^k \times [1 - p_{\mathcal{D}}(h)]^{r-k} \\ &= (1 - \eta)^k \times [1 - p_{\mathcal{D}}(h)]^k \times [1 - p_{\mathcal{D}}(h)]^{r-k} \end{aligned} \quad (5)$$

We can now write the final formulae, deriving from equations (1) and (5):

$$P = \sum_{k=0..r} \{ (1 - \eta)^k \times [1 - p_{\mathcal{D}}(h)]^k \times [1 - p_{\mathcal{D}}(h)]^{r-k} \} \times P^{MIN\eta}_k \quad (6)$$

3.4 Using the formulae in our algorithm

Suppose we have generated a propositional predicate h , and we would like to evaluate its corresponding MI-predicate H .

Let m , be the number of r -examples, and p be the total number of instances covered by h . Then, $\Pr_{\mathcal{D}}(h) = p/(m \times r)$

Each time a r -example has k of its instances covered by h , then, the probability of this r -example being covered by H would be :

$$1 - [1 - p/(m \times r)]^k \times [1 - p/(m \times r)]^{r-k}$$

³ This assumption is acceptable most case. Calculating the exact theoretical value, which we did, greatly increase the size of the final formulae without bringing much benefit to accuracy.

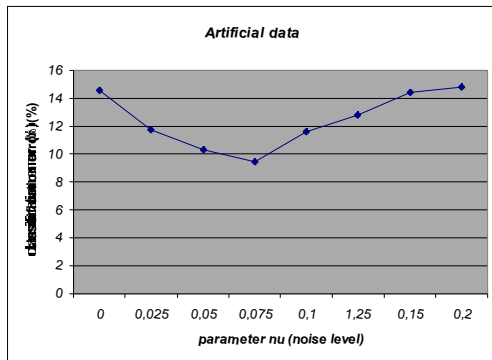
4. Implementation and experiment

We developed an algorithm implementing all these ideas in C++. It can handle mono-instance as well as MI-data. It generates decision lists, which are pruned by an iterative reduced error pruning algorithm.

Experiments have first been tried with ILP data, using REPART (Zucker et Ganascia 96,98) to reformulate the first order examples towards MI-data. On the well-known mutagenesis problem (Muggleton et Srinivasan, 1994), we obtained an accuracy of 88% (with 10-fold validation). Due to a lack of time, we couldn't try experiments on this domain with our noise model.

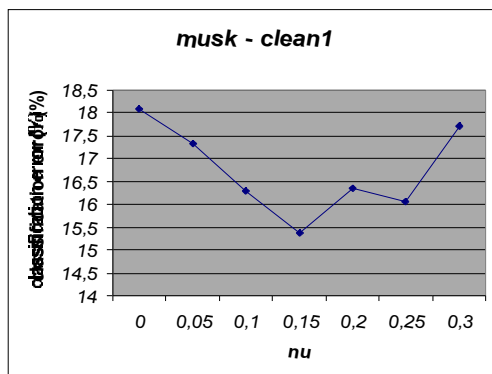
Then, we chose to experimentally validate our noise model in two steps: first, with artificial data, then with a real-world dataset. All measurements have been obtained by 10 times 10-fold validations.

We generated artificially a MI-dataset, on which we added 10% noise, following the specifications of our noise model. Then, for various settings of the noise parameter ν , we launched the learner.



Here, the gain of noise modeling is obvious : classification error decreases from 15% (without error handling) to 9.5% (with error handling, and parameter set to 8%)

For the real-world dataset validation, we chose the "musk" database introduced by (Dietterich et al 97), from which we used the "clean1" dataset.



Here, the benefit of noise handling is less clear, as the classification error is only reduced by 2.8%, for ν set to 15%, with an accuracy of 84.7%. It does not reach the performance of the best MI-algorithms, like diverse-density (Maron et al, 1998) or iterated-discrim (Dietterich et al. 1997), but it generates very concise theories, easy to interpret, consisting of a few rules of two or three literals each. During these experiments on "musk", the induced theories contained an average of 10 literals, which were generated in less than two seconds without noise handling, and less than 25 seconds with noise handling.

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 a propositional learner could easily be extended to induce multiple-instance theories, without losing its speed nor its efficiency. Therefore, the state-of-the art propositional learner could be extended to MI framework.

Our practical experiments on the mutagenesis problem show that our approach performs well, and that multiple-instance algorithms can handle numeric as well as symbolic data.

We also developed a subtle theoretical noise model specific to the multiple-instance problem, from which we derived a formulae for evaluating coverage probabilities of predicates on noise-free data, from the coverage of noisy examples. The empirical evaluation of this model gave encouraging results.

The results of our experiments on the "musk" problem, compared to other MI-algorithms, are good, but on the contrary of other learners performing well on this problem, our algorithm generates small decision lists, which are highly interpretable by humans.

In the inductive logic programming process, reformulation of relational examples to MI representation seems much more promising than propositionalization, as the relation representation is much closer to MI paradigm than to attribute-value representation. In addition, we now see that MI learners can benefit from advanced techniques of propositional learners. Thus, developing other MI-specific issues, like the noise model proposed in this article, will benefit to ILP.

References

Blum, A., & Kalai, A. (1998). A Note on Learning from Multiple-Instance Examples. *Machine Learning*, 30(1) (pp. 23-29)

- Decatur, S.E., & Gennaro, R. (1995). On Learning from Noisy and Incomplete Examples. *Computational Learning Theory*, (pp. 353-360)
- De Raedt, L. (1998). Attribute-Value Learning versus Inductive Logic Programming: The Missing Links. *Proceedings of the Eighth International Conference on Inductive Logic Programming, volume 1446 of Lecture Notes in Artificial Intelligence* (pp.1-8)
- Dietterich, T.G., Lathrop, R. H., & Lozano-Pérez, T. (1997). Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89, (pp.31-71).
- Kearns, M. (1993). Efficient Noise-Tolerant Learning From Statistical Queries. *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, (pp. 392-401)
- Long, P.M., & Tan, L. (1996). PAC-learning axis aligned rectangles with respect to product distributions from multiple-instance examples. *Proceedings of the Ninth Annual Conference on Computational Learning Theory* (pp. 228-234). New York: ACM Press.
- Maron, O., & Lozano-Pérez, T. (1998). A framework for multiple-instance learning. *Advances in Neural Information Processing Systems*, 10. MIT Press.
- Maron, O., & Ratan, A. L. (1998). Multiple-instance learning for natural scene classification. *Proceedings of the Fifteenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann.
- Maron, O. (1998). Learning from ambiguity. *Doctoral dissertation*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.
- Muggleton, S., & Srinivasan, A. (1994). ILP experiments in a non-determinate biological domain. ILP-94
- Ruffo, G. (2000). Learning single and multiple instance decision tree for computer security applications. *Doctoral dissertation*, Department of Computer Science, University of Turin, Torino, Italy.
- Sebag, M., & Rouveirol, C. (1997). Tractable induction and classification in first order logic via stochastic matching. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence* (pp. 888-893). Nagoya, Japan: Morgan Kaufmann.
- Valiant, L. G., (1984). A Theory of the Learnable. *Communications of the ACM*, 27(11), (pp. 1134-1142).
- Zucker, J.-D., & Ganascia, J.-G. (1996). Changes of representation for efficient learning in structural domains. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 543-551). Bary, Italy: Morgan Kaufmann.
- Zucker, J.-D., & Ganascia, J.-G. (1998). Learning structurally indeterminate clauses. *Proceedings of the Eighth International Conference on Inductive Logic Programming, volume 1446 of Lecture Notes in Artificial Intelligence* (pp. 235-244). Springer-Verlag