# Comprendre et résoudre les problèmes d'apprentissage multi-instances et multi-parties

**Jean-Daniel Zucker**

*LIP6-CNRS, Université Paris VI*

*4, place Jussieu*

*75252 Paris Cedex 05 - France*

*Tel. (33-1) 44-27-37-27 ou 44-27-72-07*

*e-mail : Jean-Daniel.Zucker@lip6.fr*

**Yann Chevaleyre**

*LIP6-CNRS, Université Paris VI*

*4, place Jussieu*

*75252 Paris Cedex 05 - France*

*Tel. (33-1) 44-27-37-27 ou 44-27-71-19*

*e-mail : Yann.Chevaleyre@lip6.fr*

RESUME : *Dans des travaux récents, Dietterich et al. (1997) ont présenté le problème d'apprentissage supervisé multi-instances et sa résolution par la construction d'hyper-rectangles parallèles aux axes. Ce problème se rencontre dans des contextes où un objet peut avoir différentes configurations alternatives possibles décrites chacune par un vecteur. Dans cet article nous montrons que ce problème est un cas particulier d'un problème plus général : le problème d'apprentissage multi-parties. Ces deux problèmes que nous avons baptisés «multi», peuvent jouer un rôle clef dans l'élaboration d'algorithmes efficaces pour l'apprentissage de relations entre l'activité d'un objet structuré et ses propriétés structurelles, ainsi qu'en programmation logique inductive. Nous analysons et tentons de clarifier la résolution des problèmes multi. Nous proposons ensuite des extensions multi-instances d'algorithmes d'apprentissage classiques pour résoudre les problèmes multi en apprenant des arbres de décisions multi (ID3-M, C4.5-M) et des règles de décisions multi (AQ-M, CHARADE-M).*

MOTS-CLES : *Problème multi-instances, Discrimination, Classification, Apprentissage de concepts, Arbre de décisions.*

## 1. Introduction

L'apprentissage supervisé peut se voir comme la recherche d'une fonction h, d'un ensemble d'objets O vers un ensemble de résultats R, qui soit une bonne approximation d'une fonction f dont on ne connaît le résultat que pour un certain nombre d'objets de O, des exemples de f (Dietterich, 1990). Ce problème consiste à induire la description de h à partir d'un ensemble de couples (description($objet_i$), $résultat_i$=f($objet_i$)) —les exemples d'apprentissage— et de critères —des biais d'apprentissage— permettant de choisir un espace de fonctions de O vers R et de préférer une fonction plutôt qu'une autre (Ganascia, 1991; Rouveirol, 1994). On appelle souvent la *description* de l'$objet_i$ une *instance* de l'$objet_i$.

Récemment, plusieurs travaux ont montré que ce cadre habituel pouvait s'avérer trop restreint pour des problèmes d'apprentissage complexes (Dietterich et al., 1996; Long and Tan, 1996; Sebag and Rouveirol, 1997; Zucker and Ganascia, 1994). C'est en particulier le cas lorsque plusieurs descriptions d'un même objet sont associées à un même résultat, ce que Dietterich et al. ont appelé le « problème multi-instances » (PMI) (Dietterich et al., 1996). Le terme *multi-instances* caractérise ainsi le cas où ce n'est plus à une instance mais à un ensemble d'instances {$\text{instance}_{i,1}$, $\text{instance}_{i,2}$, ..., $\text{instance}_{i,j}$} de l'$\text{objet}_i$ qu'est associé le résultat f($\text{objet}_i$) (cf. Fig.1).

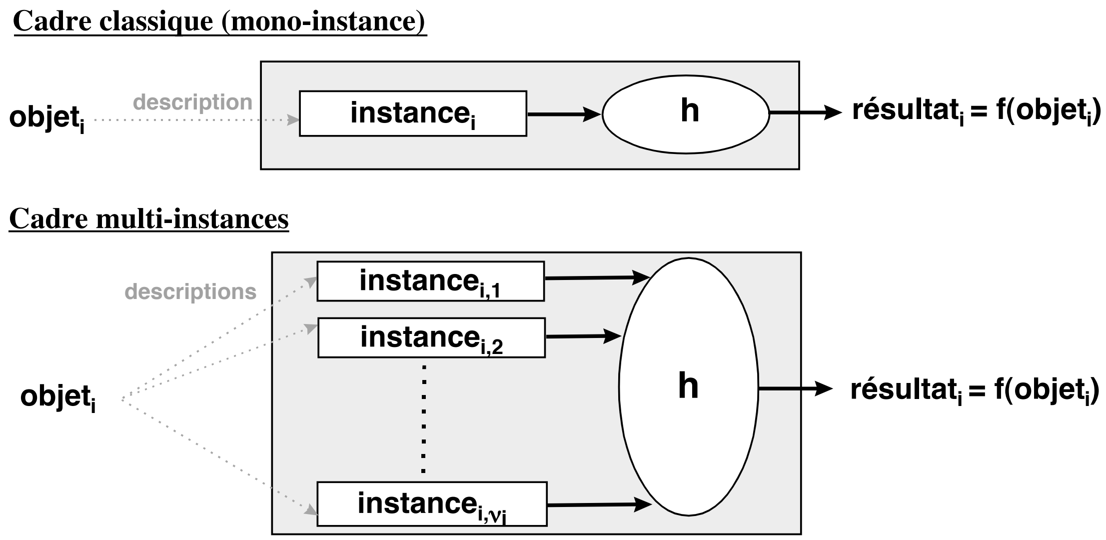



**Figure 1**— Cadres classique et multi-instances

La chimie est l'un des domaines privilégié où l'on rencontre ce problème multi-instances. Dietterich et al. présentent la tâche de classification d'un type de molécules aromatiques selon qu'elles sont ou non "musquées" (Dietterich et al., 1996). On trouve à l'état naturel, plusieurs configurations stériques d'une *même* molécule aux propriétés énergétiques très différentes. On peut ainsi produire plusieurs descriptions des différentes configurations —instances— de cette molécule. Ces descriptions correspondent à des mesures obtenues dans chacune des différentes configurations (instances $m17_1$ et $m17_2$ de la molécule m17, cf. Figure 2). Pour simplifier, on considérera qu'une molécule est dite musquée si dans l'une de ses configurations elle se lie à un récepteur particulier. Le problème qui consiste à apprendre le concept de « molécule musquée» est un problème d'apprentissage multi-instances.

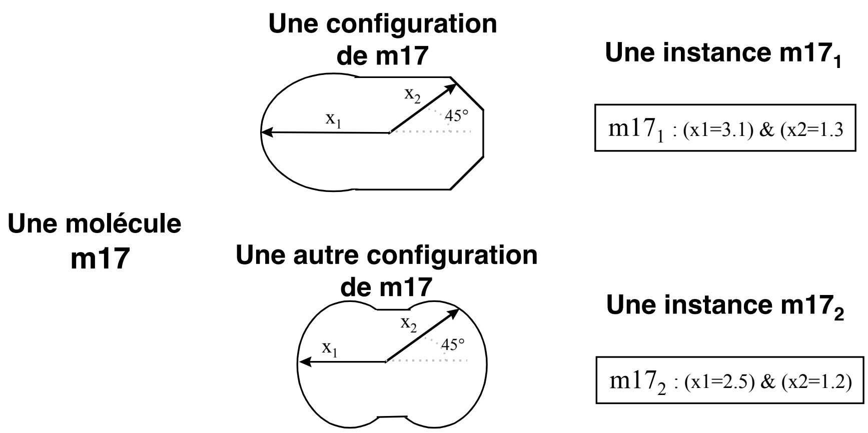



**Figure 2** — Deux instances d'une même molécule

Dietterich et al. ont proposé diverses variantes d'un algorithme d'apprentissage où les concepts sont représentés par des hyper-rectangles parallèles aux axes (APR). Ils notent "*A particularly interesting issue is how to design multiple-instances modifications for decision trees, neural networks, and other popular machine learning algorithms*" (Dietterich et al., 1996).

Dans cet article, nous analysons les difficultés que soulève le problème multi-instances en général. Nous montrons que ce problème correspond au cas particulier d'un problème plus général : le problème multi-parties (PMP) dans lequel les instances ne sont pas forcément des descriptions alternatives de l'objet mais peuvent être des descriptions de différentes *parties* de l'objet. Nous proposons des extensions « multi » aux algorithmes classiques pour traiter les problèmes PMI et PMP en apprenant des arbres de décisions et des règles de décision. Les motivations principales pour trouver de tel algorithmes sont que le PMP joue un rôle central pour l'apprentissage de relations entre la structure et l'activité des objets. C'est ce problème qui est résolu dans le système d'apprentissage REMO (Zucker and Ganascia, 1994; Zucker and Ganascia, 1996) et de Programmation Logique Inductive REPART (Zucker et al., 1998) et STILL (Sebag and Rouveirol, 1997).

Dans la partie 2 nous présentons le problème PMI de manière plus formelle, et montrons son lien avec le problème PMP et comment leurs résolutions respectives se ramènent à l'apprentissage de concepts particuliers dits multi. Dans la partie 3 nous proposons des extensions aux algorithmes classiques pour résoudre ces problèmes multi : ID3-M, AQ-M, C45-M, CHARADE-M. Nous proposons notamment une fonction d'entropie et une fonction de couverture multi-instances.

## 2. Les problèmes multi-instances et multi-parties

### 2.1. Définition du problème multi-instances (PMI)

Pour simplifier le cadre général présenté dans l'introduction, plaçons nous dans le cas où f est une fonction à valeurs booléennes —un concept— dont on connaît la valeur pour un sous ensemble de O $f(objet_i)$=VRAI (exemple positif) ou FAUX (exemple négatif ou contre-exemple) —selon que l'$objet_i$ appartient ou non au concept. Nous notons $instance_{i,j}$ la $j^{\underline{ème}}$ description de l'objet $objet_i$. Nous appelons X l'espace de représentation des instances et *co-instances* de $instance_{i,j}$, les autres instances de l'exemple $objet_i$— $\{instance_{i,j\neq k}\}$—. La fonction h que l'on cherche à apprendre, qui doit être une bonne approximation de f, est une fonction qui associe une valeur booléenne à un *sous-ensemble* des parties de X; ce que l'on peut noter par h : $2^X \longrightarrow \{\text{VRAI}, \text{FAUX}\}$. Un exemple d'apprentissage dans le cadre multi-instances est représenté sous la forme suivante :

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

Précisons d'une part que le nombre $\nu_i$ peut varier suivant $objet_i$ et d'autre part que l'indice j de 1 à $\nu_i$ donné aux instances $instance_{i,j}$ est arbitraire. Soulignons, que dans les quelques travaux théoriques sur l'apprenabilité PAC de ce problème, le nombre $\nu_i$ est égal à une constante r (Auer, 1997; Auer et al., 1997; Blum and Kalai, 1997; Long and Tan, 1996).

Dans le cadre multi-instances, Dietterich et al. (1997) font l'hypothèse que si le résultat de f est positif pour un $objet_i$ c'est parce **qu'au moins une de ses $instance_{i,j}$ a produit ce résultat.** Si le résultat est négatif, c'est qu'aucune de ses instances ne peut produire un résultat positif. Les auteurs justifient cette hypothèse par le fait que dans le domaine de la chimie moléculaire qu'ils étudient c'est précisément le cas. Nous appelons cette hypothèse, *l'hypothèse de linéarité*. En utilisant le vocabulaire introduit ci-dessus, le problème multi-instances présenté par Dietterich et al. (1997) dans leur article séminal se définit comme suit :

**Définition 1 (PMI) :** *Le problème de l'apprentissage multi-instances consiste à apprendre un concept à partir d'exemples qui sont représentés par des* ***ensembles*** *d'instances les décrivant, sous l'hypothèse de linéarité.*

### 2.2. Changement de représentation pour le PMI

La fonction h à apprendre est plus complexe à apprendre qu'un concept traditionnel car elle prend ses valeurs dans l'ensemble $2^X$ des parties de X qui a un cardinal qui croît de manière exponentielle avec celui de X. Il n'existe pas aujourd'hui d'algorithme qui résolve ce problème directement. Une approche possible face à un problème trop complexe consiste à tenter de changer de représentation pour trouver une représentation où l'apprentissage sera moins complexe (Cohen, 1990; Giordana and Saitta, 1990; Subramanian, 1990). Or, grâce à l'hypothèse de linéarité, on peut considérer un concept booléen $rv_f$ qui s'applique non plus à des ensembles d'instances mais à *une* unique instance de ces ensembles. Une instance appartient à ce concept booléen si «l'instance a produit le résultat ». Ce changement d'une représentation d'un concept définit sur $2^X$ par un concept définit sur X peut être qualifié *d'isomorphique en ce qu'il change la structure et non la quantité d'information du problème* (Korf, 1980). Le concept ainsi définit sera dit concept "multi". D'après la définition de $rv_f$, f peut se définir comme la disjonction du concept multi $rv_f$ appliqué aux différentes instances d'un objet :

$$f(objet_i) = rv_f(instance_{i,1}) \vee rv_f(instance_{i,2}) \vee .... \vee rv_f(instance_{i,\nu i})$$

La fonction $rv_f$ peut se lire comme « **r**esponsable de la **v**aleur de **f** ». Nous pouvons reformuler le problème multi-instances par rapport à cette nouvelle fonction booléenne.

**Propriété 1 (PMI) :** *Le problème de l'apprentissage multi-instances d'un concept f se ramène à un apprentissage mono-instance d'un concept $rv_f$. La description de f est donnée comme le OU logique des valeur de $rv_f$ sur les différentes instances d'un objet.*

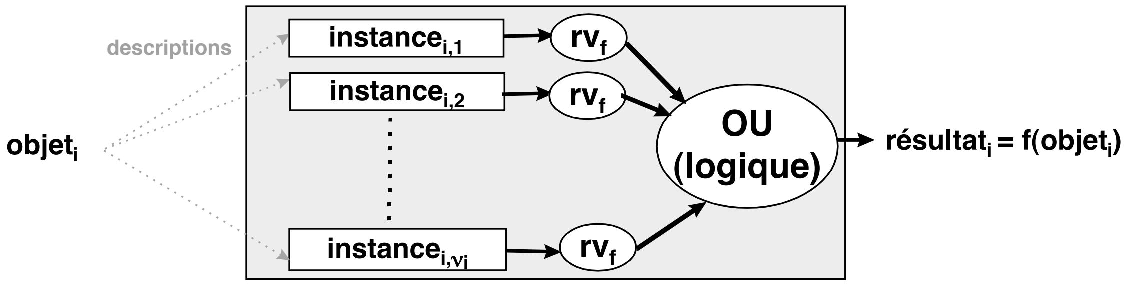



**Figure 3** — Apprentissage multi-instances de f et mono-instance de $rv_f$

La figure 3 donne une traduction graphique de la propriété 1. Si la définition du PMI est relativement simple, sa compréhension et sa résolution le sont beaucoup moins. Pour illustrer ce problème de manière intuitive, considérons le problème que nous avons baptisé le *problème simple du geôlier*. Soit une porte fermée à clef et un ensemble de N trousseaux contenants un nombre variable de clefs telles que N+ des trousseaux sont étiquetés « utiles » et $N^-$ sont étiquetés inutiles. *Un trousseau est dit utile si une de ses clefs au moins permet d'ouvrir la porte, inutile sinon*. Le concept d'utilité pourra être représenté par deux classes : celle des trousseaux utiles, et celle des trousseaux inutiles. Apprendre le concept de trousseau utile est un PMI. A partir d'un ensemble d'exemples positifs et négatifs de f (ici de trousseaux utiles et inutiles) il faut apprendre le concept $rv_f$ qui caractérise les clefs qui ouvrent la porte. Ce problème est dit "simple" car on suppose l'hypothèse de linéarité vérifié, c'est à dire qu'une clef au moins par trousseau utile permet d'ouvrir *seule* la porte.

Pour être tout à fait rigoureux, le problème du geôlier ne correspond pas tout à fait à la définition du problème PMI original. En effet, dans sa définition originale, les instances sont toutes représentatives d'un même objet, d'une même réalité. Ainsi, dans le cas des molécules chimiques, chaque configuration est un état possible de la molécule. Ce qu'il est donc important de noter c'est que dans le PMI les configurations ne peuvent apparaître *simultanément* car chacune d'entre elle caractérise l'objet considéré dans son intégralité. Pour rendre compte dans le problème du geôlier de cet aspect exclusif entre différentes co-instances , il faudrait dire qu'un trousseau est en fait une clef « magique » (ou quantique) $c_i$ qui prend une forme parmi $\nu_i$ formes lorsqu'on la présente devant la serrure et qu'elle est utile si l'une des formes qu'elle peut prendre permet d'ouvrir la porte.

Ce caractère variable des mesures d'un même objet dans le PMI permet d'interpréter le PMI comme le problème classique mono-instance rendu *ambigu* (Blum and Kalai, 1997). En effet, l'étiquette de l'objet n'est pas associée à une unique description de l'objet (une clef magique, une molécule) mais à plusieurs descriptions, qui sont toutes celles du même objet mais représentent des états différents de l'objet (une clef, une configuration); les différentes états étant autant d'explications potentielles du résultat. C'est cette forme d'ambiguïté que doivent prendre en compte les algorithme d'apprentissage MI. C'est aussi sur cette propriété que se basent des travaux sur l'apprenabilité PAC du PMI pour le réduire à des problèmes connus.

### 2.3. Le problème multi-parties et ses liens avec le problèmes multi-instances

Dans des travaux antérieurs à l'apparition du PMI, l'un des auteurs a introduit un problème en apparence similaire au PMI baptisé problème reformulé (Zucker, 1996) mais que pour plus de clarté nous appelons désormais le *problème multi-parties* (PMP). De manière informelle, le PMP caractérise l'apprentissage de concepts à partir de la description de parties d'exemples. Sa résolution est au coeur du système REMO qui permet d'apprendre efficacement des relations à partir de plusieurs milliers d'exemples structurés (Zucker and Ganascia, 1996). Le système STILL, pour construire un espace des versions disjonctif, résout itérativement un problème PMP dans lequel on considère un seul des exemples positifs à la fois (Sebag and Rouveirol, 1997). Ce système a obtenu les meilleurs résultats sur le problème ILP de la mutagenèse (Srinivasan et al., 1997).

Dans le PMP, comme dans le PMI, on cherche à apprendre un concept à partir d'exemples décrits par plusieurs instances. Cependant, dans le PMP deux modifications sont apportés au PMI. La première modification tient à ce que les différentes instances représentant un exemple peuvent ne pas décrire l'objet *tout entier* mais différentes parties de l'objet ayant la même structure[1]. C'est précisément le cas du trousseau dans le problème précédent du geôlier. Les clefs sont vues comme des parties d'un même trousseau et chacune des instances décrit une des clefs. Naturellement, si la structure considérée est celle de l'objet tout entier à un instant donné, les instances peuvent être toutes naturellement des multi-instances de l'objet. Il est en fait toujours possible de considérer que l'ensemble des instances d'un objet, au sens du PMI, sont des instances au sens PMP en prenant pour structure l'objet tout entier. En ce sens, sans en détailler ici la preuve de manière formelle, le problème multi-parties est plus général que le problème multi-instances.

La seconde modification tient à ce que, dans le PMP, l'hypothèse de linéarité selon laquelle pour chaque $objet_i$ une de ses $instance_{i,j}$ *doit avoir produit le résultat du concept* recherché n'est plus considérée. Pour illustrer la signification de l'abandon de cette hypothèse, considérons le *problème complexe du geôlier*. Soit une porte fermée à clef et un ensemble de N trousseaux contenants un nombre variable de clefs tel que N+ des trousseaux sont étiquetés « utiles » et $N^-$ sont étiquetés inutiles. *Un trousseau est dit utile si une utilisation particulière des clefs du trousseau permet l'ouverture de la porte, inutile sinon*. On voit ici qu'on ne suppose plus qu'une clef unique peut ouvrir la porte. Cela peut être le cas mais il est aussi possible qu'il faille faire une *combinaison complexe des clefs (comme avec un coffre-fort qui nécessite plusieurs clefs et plusieurs manipulations)*. Le problème d'apprentissage du concept de « trousseau utile » est un véritable problème multi-parties. En effet, à partir d'un ensemble d'exemples positifs et négatifs de f (ici de trousseaux utiles et inutiles) il faut apprendre ce qui caractérise les clefs qui sont *dans des trousseaux utiles.* Le problème multi-parties peut se définir plus formellement comme suit :

**Définition 2 (PMP) :** *Le problème de l'apprentissage multi-parties consiste à apprendre un concept à partir d'exemples qui sont représentés par des* ***ensembles*** *d'instances décrivant leurs parties, sans l'hypothèse de linéarité.*

### 2.4. Changement de représentation pour le PMP

Du fait de la levée de l'hypothèse de linéarité, peut-on encore introduire une fonction comme nous l'avons fait dans le PMI qui s'applique non plus sur un ensemble d'instances mais sur une unique instance de cet ensemble ? La réponse est affirmative, il suffit de considérer la fonction $in_f$ « appartient à un objet ayant le résultat f ». Cependant, dans le PMP, il devient tout à fait possible, même en l'absence de bruit, que deux instances (par exemple deux instances de clefs) soient rigoureusement identiques mais proviennent d'objets de classes différentes[2] (par exemple l'une d'un trousseau utile et l'autre d'un trousseau inutile). La fonction $in_f$ doit donc a priori prendre la valeur "INDISCERNABLE" sur ce type d'instances, ce qui a pour conséquence qu'$in_f$ n'est pas une fonction booléenne. On peut aussi ramener l'apprentissage multi-parties d'un concept f à l'apprentissage mono-instance de la fonction $in_f$. On obtient f comme un OU modal[3] (pour lequel la valeur INDISCERNABLE est neutre) des valeur de $in_f$ sur les différentes instances d'un objet.

Pour se ramener comme dans le PMI à un problème d'apprentissage de concepts, et donc à une fonction booléenne, il faut soit regrouper les valeurs "FAUX" et "INDISCERNABLE" en une seule valeur "PASVRAI" soit regrouper les valeurs "VRAI" et "INDISCERNABLE" en une valeur "PASFAUX". Ainsi on peut remplacer l'apprentissage de la fonction $in_f$ par l'apprentissage de deux concepts : $oin_{f+}$ ("only in f+") et $oin_{f-}$ ("only in f-"). Dans le cas du geôlier, ils correspondent aux concepts de *clefs se trouvant dans des trousseaux qui permettent d'ouvrir (resp. ne permettent pas) la porte et dans aucun trousseau qui ne le*

[1] Cette même structure est appelée un morion (une partie en grec). L'importance du choix d'une *même* structure est justifié par la nécessité d'apparier les instances, et ce pour un traitement efficace des problèmes d'appariement. Une discussion détaillée de cette question est donnée dans (Zucker, 1996).

[2] Notons que ce cas peut aussi survenir dans le cas du PMI en présence de bruit. L'analyse du bruit dans le PMI et le PMP seront abordés dans un article qui est en préparation. Il apparaît qu'en résolvant le PMP, on résout le problème PMI avec bruit.

[3] Dans le cas où l'on teste un objet dont toutes les instances on été classées après apprentissage comme indiscernables, on prendra la classe majoritaire comme valeur du "*ou" modal* car f est un concept et doit prendre pour valeur VRAI ou FAUX.

*permet pas (resp. le permet)*. Ils peuvent encore se lire « ne fait partie *que* d'objets qui satisfont (resp. ne satisfont pas) le concept f».

**Propriété 2 (PMP):** *Le problème de l'apprentissage multi-parties d'un concept f se ramène à un apprentissage mono-instance de deux concepts $oin_{f-}$ et $oin_{f+}$. Une description de* f s'obtient comme: $f(objet_i) = [oin_{f+}(instance_{i,1}) \vee .... \vee oin_{f+}(instance_{i,vi})]$ OP $\neg[oin_{f-}(instance_{i,1}) \vee .... \vee oin_{f-}(instance_{i,vi})]$ *où OP est un « ou » logique si la classe majoritaire est + et un « et » logique sinon.*

Si le problème PMP est plus général que le problème PMI c'est surtout à un niveau *sémantique*, car d'un point de vue computationnel, il s'agit dans les deux cas d'apprendre des concepts multi puis de faire une opération logique sur les valeurs de ces concepts multi pour les différentes instances d'un objet.

## 2.5. Apprentissage des concepts multi

Les problèmes PMI et PMP peuvent se ramener, on l'a vu dans les parties 2.2 et 2.4, à l'apprentissage mono-instance de concepts multi ($rv_f$ ou $oin_{f-}$ et $oin_{f+}$). Une fois appris, ces concepts permettent de caractériser le concept initial recherché. Une des difficultés de l'apprentissage de concepts multi tient à ce que l'on ne connaît pas d'exemples de ces concepts multi au sens classique du terme. On sait simplement si la disjonction des $in_f$ appliquée à des co-instances est positive ou négative. Il est assez intuitif qu'ignorer les spécificité du problème ne permettra pas d'apprendre de manière satisfaisante les concepts multi. Ignorer un problème multi c'est considérer que toutes les instances d'un même exemple sont de la classe de l'exemple. Dans le problème du geôlier cela revient à considérer que toutes les clefs d'un trousseau ouvrent la porte si le trousseau est utile.

Un algorithme classique d'apprentissage appliqué sans modification aux problèmes multi apprendrait une description qui couvre toutes les instances de tous les exemples positifs du concept. Il est clair qu'une telle description existe rarement dans le problème PMI et qu'elle est encore plus improbable dans le cadre PMP. En effet, il suffit que deux exemples aient une de leurs instances identiques mais soient classées différemment pour qu'une telle description n'existe pas. Dans le problème illustratif du geôlier, si deux trousseaux, l'un utile et l'autre inutile, possèdent chacun une clef dont les descriptions sont très similaires voire identiques, on ne pourra pas trouver de telle description. Supposons que les trousseaux soient représentés par des instances des clefs représentées par des couples (x,y) où x est la taille de la clef, y le nombre de dents. On supposera que l'on dispose de trois trousseaux utiles (dont les instances de clefs sont représentées respectivement par des carrés, des triangles et des rectangles) et deux trousseaux inutiles (dont les instances sont représentées par respectivement des cercles et des ellipses). Considérons l'espace à deux dimensions où l'on a représenté chaque instance d'un trousseau par la forme du trousseau (cf. figure 4).

Les seuls algorithmes existants de résolution du PMI apprennent des hyper-rectangles. La méthode proposée par Dietterich s'applique à des multi-instances représentées par des vecteurs dans $X=R^d$. Le concept multi recherché est représenté par un hyper-rectangle dans $R^d$. L'heuristique utilisée consiste à trouver un petit hyper-rectangle qui soit consistant avec les données dans le sens où tous les exemples initiaux positifs y ont au moins une de leur instances et les exemples initiaux négatifs aucune (l'$APR_1$ sur la figure 4). Cette contrainte peut être relâchée si aucune boîte n'est trouvée pour que seul (100-ε)% des exemples soient couverts (c'est à dire qu'une au moins de leurs instances soit dans l'hyper-rectangle). Notons que ce problème de recherche est déjà en lui même NP-complet (Blum & Kalai). Ensuite, l'APR trouvée est « dilatée » (l'$APR_2$ sur la figure 4) pour en augmenter son pouvoir généralisant en se basant sur des estimations sur la densité de distribution des valeurs des attributs des instances.

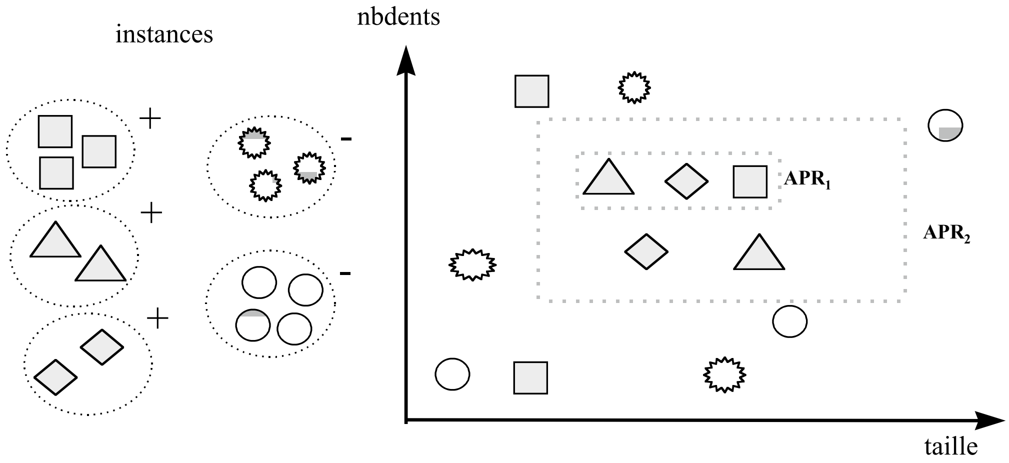



**Figure 4 —** (Hyper)-rectangle solution du PMI du geôlier

### 2.6. Apprenabilité du PMI

Le problème multi-instances suscite un intérêt croissant de la communauté COLT. Long et Tan (Long and Tan, 1996) furent les premiers à montrer l’apprenabilité PAC du problème PMI dans des conditions restrictives peu réalistes. Ils ont montré que dans ce cadre, le problème est apprenabble au sens PAC avec une borne qui a depuis été largement dépassée. Auer et al. (Auer et al., 1997) considèrent l’apprentissage d’APR sous la condition que D (la distribution sur les instances d’un exemple initial) soit un produit de distributions et que le nombre d’instances par exemple soit une constante r. Ils ont montré que les hyper-rectangles sont apprenables au sens PAC avec $m=O(d^2r^2/\varepsilon^2)$ exemples où d est la dimension des vecteurs d'instances, le temps nécessaire à l'apprentissage étant alors en O(dmlog(m)). Auer (Auer, 1997) a fait lui une analyse théorique du problème et en a dérivé un algorithme MULTINST qui lui permet d’avoir des performances comparables à ceux de Dietterich . Il apprend aussi des APR et se libère de la limite d’avoir autant d’instances pour chaque exemple. Ce qui est particulièrement intéressant dans son article c’est le fait qu’il soit parti de considérations théoriques pour arriver à un algorithme pratique compétitif. Quant à Blum et Kalai (Blum and Kalai, 1997), ils ont réduit de manière élégante le PMI au problème de l’apprentissage PAC avec un « one-sided random classification noise ».

## 3. Résoudre les problèmes multi

Pour la résolution du problème multi-instances en tant que tel, il n’existe que des variantes d’algorithmes d’apprentissage d’APR (Dietterich et al., 1996). En ce qui concerne l’apprentissage du PMP, l’algorithme enigme+ (une extension au PMP de CHARADE (Ganascia, 1993)) est le seul existant à apprendre des règles multi (Zucker and Ganascia, 1996). Dans cette section, nous proposons d’analyser les modifications générales nécessaires à la définition d’algorithmes prenant en compte les problèmes multi. Ces algorithmes prendront pour exemples d’apprentissage les instances des différents exemples ainsi qu'une étiquette indiquant l'objet dont ils proviennent.

### 3.1. Des algorithmes propositionnels pour les problèmes multi

D’importantes similitudes peuvent être constatées entre le problème multi-instances et le problème mono-instance ; ceci nous a incité à étudier l’adaptation d’outils d’apprentissage propositionnel aux problèmes multi. Nous considérons ici les plus "souples" de ces outils, à savoir les algorithmes dits « descendants », qui génèrent des hypothèses de plus en plus spécifiques (Ganascia, 1993). Il existe de nombreux algorithmes propositionnels d'apprentissage dont les plus classiques utilisent deux principales approches : celles qui *divisent pour régner (divide-and-conquer)* et celles dites par couverture *(cover and differentiate)* .

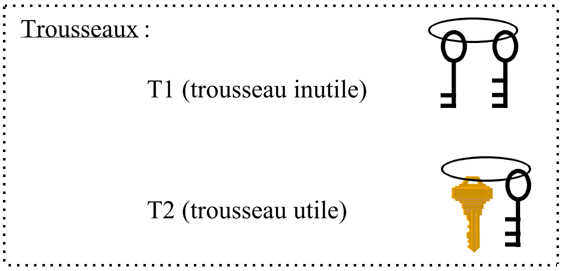
- Les algorithmes qui « divisent pour régner » (ID3, C4.5) utilisent généralement des arbres de décision pour représenter les concepts appris. Pour guider le développement récursif de l’arbre ils reposent sur des heuristiques dans lesquelles l’entropie d’information joue un rôle crucial.

- Les algorithmes qui utilisent une fonction de « couverture » représentent souvent les hypothèses par des disjonctions de monômes de longueur inférieure à k (k-DNF). Les approches par *couverture* recherchent une des expressions conjonctives qui soient « couvrent » *certains* des exemples positifs et aucun exemple négatif. Elles réitèrent cette recherche tant qu'il existe des exemples positifs non encore couverts par les expressions conjonctives déjà trouvées (AQ, CHARADE). D'autres représentations sont aussi utilisées dans ces approches comme les listes de décisions (CN2 ).

Une adaptation au PMI de ces algorithmes est proposée dans les sections suivantes.

### 3.2. Représentation de concepts PMI : les arbres de décision multi et les règles multi

On supposera ici connu le fonctionnement des algorithmes comme ID3 ou C4.5 qui utilisent des arbres de décision pour représenter les concepts appris. Chaque noeud de l'arbre correspond à un attribut, et chaque branche partant de ce noeud sélectionne un sous-ensemble de valeurs pour cet attribut. Dans ces derniers, lors de la construction de l'arbre on peut d'une part étiqueter une feuille si elle est pure (il n'y a plus que des instances d'une classe) ou presque pure (pre-prunning), et d'autre part choisir un nouvel attribut quand l'étiquetage de la feuille n'est pas possible. Dans le cas PMP, l'étiquetage d'un arbre multi pourra être tri-valuée. Les arbres de décisions multi devront donc acheminer les vecteurs attributs/valeurs de chaque instance vers les feuilles étiquetées VRAI, FAUX, ou INDISCERNABLE. Les feuilles étiquetées INDISCERNABLE trahissent la présence d'instances n'apportant pas d'information sur l'exemple.




Instances :
(clef1 (inT = T1) ($in_{utile}$=faux) (nbdent=3) (forme=«n»))
(clef2 (inT = T1) ($in_{utile}$=faux) (nbdent=2) (forme=«n»))
(clef3 (inT = T2) ($in_{utile}$=vrai) (nbdent=3) (forme=«n»))
(clef4 (inT = T2) ($in_{utile}$=vrai) (nbdent=5) (forme=«w»))

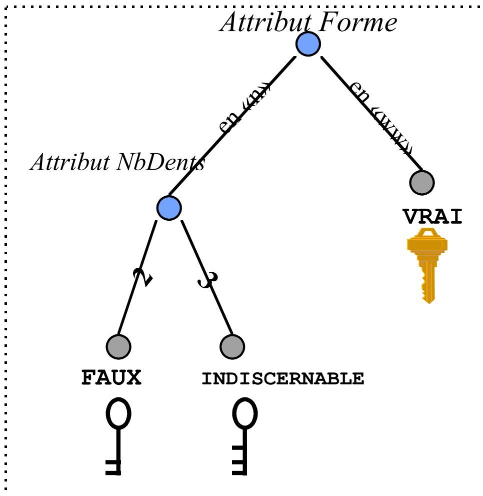



**Figure 5 —** Un exemple positif et négatif ayant chacun une instance identique, et un arbre de décision multi correspondant

En ce qui concerne les règles multi, il y a peut à en dire car elles se présentent sous la forme classique d'un ensemble de règles conjonctives classant les instances positivement ou négativement. De part la nature des algorithmes utilisés, aucune règle concernant des instances indiscernables n'auront à être générées. La principale différence avec le cadre classique, aussi bien pour les arbres de décision multi que pour les règles multi est dans leur utilisation. En effet, ce n'est *qu'après le traitement de l'ensemble des instances d'un objet* que le classement est donné.

### 3.3. Apprentissage de concepts multi

En simplifiant, l'apprentissage de concepts multi, que ces concepts soient représentés par des arbres de décision ou des règles, peut se réduire au pseudo-algorithme décrit ci-après .Nous nous intéressons, selon les deux représentations de S (arbre de décision ou règle multi), aux fonctions suivantes :

- le *développement* de S (pour un arbre de décision ajouter un noeud et pour un système à base de règles, des règles),
- le *critère* d'arrêt, et le calcul du *taux* d'erreur.

```
FAIRE TANT QUE
   (NON CRITERE D'ARRET(S) ET DEVELOPPEMENT POSSIBLE)
   DEVELOPPER(S)
   S' <- PRE-ELAGUAGE(S)
```

```
    FAIRE TANT QUE TAUX D'ERREUR(S') < TAUX D'ERREUR(S)
            S <- S'
            S' <- POST-ELAGUAGE(S)
    FIN TANT QUE
FIN TANT QUE
```

Pour simplifier, nous choisissons, comme critère d'arrêt, un seuillage sur le taux d'erreur. L'élaguage, qu'il s'agisse de supprimer des branches d'un arbre de décision ou des règles, est lui aussi conduit par ce taux d'erreur.

### *3.3.1. L'entropie multi-instances pour l'apprentissage d'arbres de décision multi*

La construction d'un arbre de décision est classiquement guidée heuristiquement par un gain d'entropie d'information, ou par l'une de ses variantes. Dans le cadre mono-instance l'entropie d'information biclasse d'un ensemble de p instances positives et n instances négatives vaut :

$$Info(P) = \frac{nbits(P)}{p+n} = -\frac{p}{p+n} \times \log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n} \times \log_2\left(\frac{n}{p+n}\right)$$

Le gain, indiquant la quantité d'information apportée par l'ajout du noeud, est fréquemment utilisé conjointement à un critère limitant la taille de l'arbre comme le principe du MDL (Quinlan and Rivest, 1989). C'est l'attribut A qui a le meilleur *gain* d'information qui est choisi dans l'algorithme de construction de l'arbre de décision, le gain étant calculé de la manière suivante :

$$Gain(T, A) = Info(P) - \sum_i \frac{p_i + n_i}{p+n} \times Info(Pi)$$

où $p_i$ représente le nombre d'instances positives et $n_i$ le nombre d'instances négatives obtenues pour la ième valeur de l'attribut A.

Nous proposons de définir une nouvelle entropie dite multi-instances qui permette de prendre en compte les spécificités des problèmes multi. Nous supposons que toutes les instances d'un exemple (resp. contre-exemple) doivent être classées positivement ou négativement. Nous ignorons donc délibérément le cas des instances indiscernables, qui peuvent être traitées à part, lors d'un post-prunning de l'arbre de décision. Dans le problème multi-instances reformulé où l'on apprend un concept multi ($rv_f$ ou $oin_{f+}$) ce qui importe c'est de discriminer des instances **provenant** d'exemples différents (cf. fig.3). En conséquence, l'entropie multi-instances d'un ensemble de p instances positives et n instances négatives se calculera à partir des valeur $\pi(p)$ et $\nu(n)$ qui représentent respectivement le nombre d'exemples positifs **différents** dont proviennent les p instances positives et le nombre d'exemples négatifs **différents** dont proviennent les n instances négatives.

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

C'est sur la base de cette mesure d'entropie multi-instances que reposent les algorithmes d'apprentissage d'arbre de décisions multi ID3-M et C4.5-M que nous avons mis en œuvre.

### *3.3.2. L'apprentissage de règle multi*

Nous nous intéressons ici uniquement à l'apprentissage de règles guidé par une mesure de leur couverture[4]. Pour un tel apprentissage de concept multi, la mesure de couverture doit être redéfinie. En effet, cette notion est étroitement liée à la notion d'exemple d'apprentissage qui est elle-même modifiée dans les problèmes PMI et PMP.

Dans le cadre classique, un exemple est dit « couvert » par une généralisation G s'il est plus spécifique que G. On utilise souvent le prédicat *cover* pour exprimer le fait que l'$objet_i$ est couvert par la généralisation *G* : *Cover(G,*$objet_i$*)*. Pour mesurer la portée d'une généralisation G on fait souvent appel à la notion de *taux de couverture* qui dénombre les exemples pour lesquels le prédicat s'évalue à vrai. Dans

[4] Il existe des algorithmes d'apprentissage de règles, comme par exemple C4.5rules, qui extraient des règles à partir d'arbres de décision en se basant sur une mesure d'entropie.

ce cas la couverture de G qui généralise des exemples positifs est le cardinal de l'ensemble des exemples positifs couvert par G.

Dans le cadre multi-instances, on peut dire qu'un objet est couvert par une généralisation G si au moins une de ses $\text{instance}_{i,j}$ est couverte par *G*. Ce qui s'exprime par :

$$Cover_{multi}(G, \text{objet}_i) \leftarrow \exists j \mid Cover(G, \text{instance}_{i,j}) \qquad \text{et} \quad Cover(G, \text{instance}_{i,j}) \leftarrow instance_i \geq G$$

Dans le cadre multi, le taux de couverture doit être donc redéfini comme suit :

$$Couverture_{multi}(G) = card\left\{objet_i ; Cover_{multi}(G, objet_i)\right\}$$

C'est sur la base de cette couverture multi-instances que reposent les algorithmes d'apprentissage de règles multi AQ-M et CHARADE-M que nous avons mis en œuvre.

## 4. Conclusion

Le problème d'apprentissage supervisé multi-instances est un problème d'apprentissage récent qui suscite un intérêt dans la communauté d'apprentissage. Ce problème se rencontre dans des contextes où un objet peut avoir *plusieurs vecteurs alternatifs* pour décrire ses différentes configurations possibles. Dans cet article nous avons montré que ce problème est subsummé par le problème multi-parties. Ce dernier peut jouer un rôle clef dans les algorithmes en apprentissage de relations, et en programmation logique inductive. Zucker, Ganascia, et Bournaud (98) montrent en détail son intérêt en programmation logique inductive dans le système REPART.

La résolution des problèmes multi par des algorithmes classiques soulève des questions importantes et subtiles que nous avons analysé. Nous avons montré comment ces problèmes peuvent être résolus par des algorithmes d'apprentissage mono-instance de *concepts multi*. Nous avons proposé des extensions aux algorithmes classiques pour les résoudre en apprenant des arbres de décisions et des systèmes à base de règles. Ces algorithmes étendus sont basés sur deux notions : *l'entropie multi-instances* et la *couverture multi-instances*. Ces modifications nous ont permis de mettre en oeuvre des programmes ID3-M, AQ-M, C4.5-M et CHARADE-M. Ils donnent lieu à des expérimentations qui font l'objet d'évaluation empiriques en cours. Nous travaillons aussi sur une version multi de l'algorithme RIPPER (Cohen, 1995) pour des applications en fouille de données.

Il reste encore de nombreuses questions en suspens quand à la résolution des problèmes multi qui sont autant de direction de recherches. Au niveau théorique, il n'existe pas de travaux à notre connaissance sur l'apprenabilité du problème PMP proprement dit. Il est cependant vraisemblable que l'on puisse le réduire à un problème multi-instances dans lequel on injecte du bruit. Les travaux d'Auer ont ouvert la voix dans cette direction Au niveau pratique, il reste de nombreuses approches à expérimenter et des algorithmes neuronaux sont encore à trouver pour les problèmes multi-instances et multi-parties.

## Références


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

Auer, P., Long, P., and Srinivasan, A. 1997. Approximating hyper-rectangles: Learning and pseudo-random sets. Proc. *Proceeding of the 29th Annual ACM Symposium on Theory of Computation*, pp. 228-234.

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

Cohen, W. 1990. An analysis of Representation Shift In Concept Learning. Proc. *Seventh International Conference on Machine Learning*, pp. 104-112, Austin, Texas.

Cohen, W. 1995. Fast Effective Rule Induction. Proc. *International Conference on Machine Learning*, pp. 115-123, Nashville, USA.

Dietterich, T. 1990. Readings in Machine Learning, Morgan Kaufmann.

Dietterich, T., Lathrop, R., and Lozano-Perez, T. 1996. Solving the Multiple-Instance Problem with Axis-Parallel Rectangles. *Artificial Intelligence*, 89(1-2): 31-71.

Ganascia, J.-G. 1991. Deriving the Learning Bias from Rule Properties. *Machine Intelligence*(12).

Ganascia, J.-G. 1993. TDIS: an Algebraic Formalization. Proc. *13th Unternational Joint Conference on Artificial Intelligence*, pp. 1008-1013, Chambéry, France.

Giordana, A., and Saitta, L. 1990. Abstraction: a general framework for learning. Proc. *Working notes of the AAAI Workshop on Automated Generation of Approximations and Abstraction*, pp. 245-256, Boston, MA.

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

Long, P., and Tan, L. 1996. PAC Learning Axis-aligned Rectangles with respect to Product Distributions from Multiple-instance Examples. Proc. *Proceedings of the 9th Annual Conference on Computational Learning Theory, COLT' 96*, pp. 228-234, Desenzano del Garda, Italy.

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

Rouveirol, C. 1994. Declarative Bias Workshop. Proc. *MLnet Familiarization Workshops*, Catania, Italy.

Sebag, M., and Rouveirol, C. 1997. Tractable Induction and Classification in First Order Logic. Proc. *Fifteenth International Joint Conference on Artificial Intelligence, IJCAI'97*, pp. 888-893, August 23-29, Nagoya, Japan.

Srinivasan, A., Muggleton, S., King, R. D., and Sternberg, M. E. 1997. The Predictive Toxicology Evaluation Challenge. Proc. *Fifteenth International Joint Conference on Artificial Intelligence, IJCAI'97*, pp. 4-9, August 23-29, Nagoya, Japan.

Subramanian, D. 1990. A Theory of Justified Reformulations. *Change of Representation and Inductive Bias*, ed., Benjamin, D. P., pp. 147-167, Kluwer Academic Publishers, Boston.

Zucker, J.-D. 1996. Appariements et Changements de Représentation pour l'Apprentissage Symbolique, Thèse de Docteur en Sciences (PhD. Thesis), Spécialité Informatique, Université Paris VI.

Zucker, J.-D., and Ganascia, J.-G. 1994. Selective Reformulation of Examples in Concept Learning. Proc. *International Conference on Machine Learning*, pp. 352-360, New-Brunswick.

Zucker, J.-D., and Ganascia, J.-G. 1996. Changes of Representation for Efficient Learning in Structural Domains. Proc. *International Conference in Machine Learning*, pp. 543-551, July 3-6, Bari, Italy.

Zucker, J.-D., Ganascia, J.-G., and Bournaud, I. 1998. Relational Knowledge Discovery in a Chinese Characters Database. *Applied Artificial Intelligence*, Special Issue on KDD in Structural Domains (to appear).