

---

# Méthodes de Sélection d'Attributs en Programmation Logique Inductive

---

Nicolas Stroppa

STROPPIA@LRI.FR

LRI – Bâtiment 490

Université Paris-Sud,

91405 ORSAY CEDEX, France

## Résumé

Un paradigme récent a montré comment filtrer un problème de Programmation Logique Inductive (PLI) de manière analogue à la sélection d'attributs existant en apprentissage attribut-valeur. Dans ce paradigme, un problème de PLI est tout d'abord converti en plusieurs problèmes multi-instances, problèmes particuliers d'apprentissage attribut-valeur. Cette conversion introduit de façon inhérente un bruit d'attribut et de classe dans les données. Nous présentons ici une méthode de sélection d'attributs spécialement conçue pour traiter ces problèmes multi-instances bruités particuliers. Nous utilisons cette méthode pour réduire les problèmes multi-instances correspondant au problème de PLI d'origine, le résultat obtenu étant ensuite reconverti en un problème de PLI de façon à pouvoir être traité par un système d'apprentissage approprié. Nous présentons ensuite quelques expérimentations, dans lesquelles les performances sur la tâche d'apprentissage avec ou sans filtrage sont comparées. Les résultats obtenus sont finalement discutés et commentés, et des perspectives d'évolution proposées.

**Mots clés:** Sélection d'Attributs, Programmation Logique Inductive, Problèmes Multi-Instances.

## 1 Introduction

Dans le domaine de l'apprentissage, des méthodes connues sous le nom de sélection d'attributs sont utilisées pour réduire la dimensionnalité d'un problème, en lui supprimant un certain nombre de ses attributs. Ces méthodes sont largement employées dans les domaines de l'apprentissage symbolique (Liu & Motoda, 1998) et des réseaux de neurones (workshop NIPS, 2001). Cependant, la Programmation Logique Inductive (PLI) (Lavrač & Džeroski, 1994), autre branche active du domaine de l'apprentissage, n'a pas encore bénéficié des apports de ces techniques. La sélection d'attributs en PLI est un domaine nouveau, qui n'a pas encore été étudié; ce papier propose un moyen de modifier cet état de fait.

Pour commencer, nous rappelons les principes de la sélection d'attributs, présentons par un exemple les bases de la PLI, et formulons notre problème à l'aide du paradigme permettant de transformer la tâche de filtrage en PLI en un problème de sélection d'attributs en apprentissage attribut-valeur.

## 1.1 La sélection d'attributs

La *sélection d'attributs* (souvent qualifiée de filtrage) a pour but de réduire la dimensionnalité d'un problème d'apprentissage en ne conservant qu'un sous-ensemble de ses attributs initiaux. En apprentissage attribut-valeur, la notion d'attribut est évidente puisque les exemples présentés sont des conjonctions de paires attribut-valeur.<sup>1</sup>

Intuitivement, cette tâche consiste à supprimer d'un problème ses attributs non-pertinents ou redondants, les notions de pertinence et de redondance étant bien entendu à définir. Par exemple, dans le cas d'une tâche de classification, on peut considérer qu'un attribut n'est pas pertinent s'il n'est pas prédictif de la classe, et redondant si son pouvoir prédictif peut être obtenu à partir d'autres attributs. Plus formellement, la tâche de sélection d'attributs consiste à trouver un sous-ensemble d'attributs optimal par rapport à une certaine mesure.

Pour illustrer cette notion, prenons l'exemple d'un problème d'apprentissage dans lequel les exemples sont des données concernant les patients d'un hôpital, dont on sait qu'ils sont atteints ou non par une certaine maladie. La tâche d'apprentissage de ce problème est de construire un classifieur permettant de déterminer si un nouveau patient est malade ou non, à partir de ses seules données. Les données peuvent contenir des mesures de température, de poids ou de présence de boutons sur la peau. Si ces données contiennent également le numéro de sécurité sociale et le numéro de chambre des patients, il est raisonnable de penser que ces derniers attributs ne sont pas discriminants quant à la présence de la maladie, et par conséquent non-pertinents pour la tâche d'apprentissage.<sup>2</sup> En général, on attend de la tâche de filtrage qu'elle supprime ce genre d'attributs.

En filtrant les données, on espère :

- simplifier la tâche d'apprentissage en lui présentant des données réduites,
- améliorer ses résultats en éliminant les attributs non-pertinents, source de bruit.

---

<sup>1</sup>Par exemple,  $J : (Ciel = Ensoleillé) \wedge (Humidité = Normale) \wedge (Vent = Oui)$ , ou encore,  $P : (Température = 38) \wedge (Poids = 64) \wedge (Boutons = Oui) \wedge (n^{\circ} Insee = 34B2)$ .

<sup>2</sup>Soulignons toutefois que, même si cet exemple apparaît trivial, il soulève un certain nombre de problèmes. En effet, dans cet exemple, le numéro de sécurité sociale d'un patient étant unique, son pouvoir prédictif concernant la classe est maximal sur la base d'apprentissage. On comprend ici la difficulté de définir une notion telle que la pertinence d'un attribut.

De façon à évaluer un sous-ensemble d'attributs, différentes mesures ont été proposées dans le cadre de l'apprentissage attribut-valeur. Elles se fondent sur les notions de cohérence, comme dans FOCUS (Almuallim & Dietterich, 1994), de corrélation (Hall, 1998), de distance, comme dans RELIEF (Kira & Rendell, 1992), ou encore sur la capacité à obtenir de bons résultats sur la tâche d'apprentissage, comme dans l'approche "wrapper" (Kohavi & John, 1997).

Soulignons, en outre, que la sélection d'attributs est un problème important et difficile (Liu & Motoda, 1998, workshop NIPS, 2001), agitant plusieurs communautés depuis plus de 50 ans. A cet égard, il peut être utile de rappeler que de manière générale, trouver le meilleur sous-ensemble d'attributs par rapport à une certaine mesure est un problème  $NP$ -complet, l'espace dans lequel la recherche s'effectue étant  $2^E$ , où  $E$  est l'espace des attributs.<sup>3</sup> Notons également que la sélection d'attributs fait désormais partie intégrante de l'étape de prétraitement de nombreux systèmes d'apprentissage, notamment dans les domaines de l'apprentissage attribut-valeur et des réseaux de neurones.

## 1.2 La Programmation Logique Inductive

L'objectif de la *Programmation Logique Inductive* est de savoir apprendre dans un contexte dans lequel le langage de représentation est une restriction de la logique du premier ordre. En PLI, il s'agit de produire un résultat relationnel à partir d'exemples et d'une certaine connaissance de base. Dans ce contexte, les exemples, la connaissance de base et le résultat, s'expriment non pas dans la logique des propositions (logique d'ordre zéro), classiquement utilisée en apprentissage attribut-valeur, mais dans un sous-ensemble de la logique des prédicats (logique du premier ordre). Au niveau propositionnel, tous les exemples décrivent nécessairement le même objet. Pour traiter des exemples structurés, composés de plusieurs parties liées entre elles par certaines relations, la représentation doit contenir des variables, ce qui explique le besoin de se positionner au niveau de la logique du premier ordre.

Les "trains de Michalski" sont un bon exemple de problème nécessitant une description en logique du premier ordre. Dans ce problème, il existe deux sortes de trains, des trains allant vers l'Est et d'autres vers l'Ouest. Un exemple de train Est composé de trois wagons différents, un wagon long contenant une charge rectangulaire, un wagon long, et un wagon court contenant une charge hexagonale, peut être représenté en logique du premier ordre par la formule suivante :

$$\text{Est}(t) \quad :- \quad \text{wagon}(t,c1), \text{long}(c1), \text{charge}(c1,l1), \text{rect}(l1), \\ \text{wagon}(t,c2), \text{long}(c2), \\ \text{wagon}(t,c3), \text{court}(c3), \text{charge}(c3,l3), \text{hex}(l3).$$

---

<sup>3</sup>Dans les approches existantes, certaines sont complètes (FOCUS en est l'exemple type), et d'autres sont heuristiques (id. pour RELIEF).

Remarquons qu'un train pouvant être composé d'un nombre variable de wagons, contenant eux aussi un nombre variable de charges, il est très difficile de le représenter en terme de paires attribut-valeur, alors que sa représentation apparaît naturelle en logique du premier ordre.

La tâche d'apprentissage de ce problème est, étant donnés un certain nombre d'exemples de trains Est et Ouest et une certaine connaissance de base (telle que : *"les rectangles et les hexagones sont des polygones"*), de trouver une description caractérisant chacun des types de train. Dans notre cas, voici une description possible pour les trains Est :

*"un train Est a un wagon long avec un charge polygonale"*,

soit en logique du premier ordre :

$\text{Est}(T) \quad :- \quad \text{wagon}(T, C1), \text{long}(C1), \text{charge}(C1, L1), \text{polygone}(L1).$

La PLI a pour but de pouvoir classifier des exemples dans une telle représentation. Dans l'exemple, cela consiste à pouvoir discriminer entre les différents types de trains. Soulignons, en outre, que la PLI est d'ores et déjà appliquée à de nombreux problèmes réels, dans les domaines de la biochimie, de la toxicologie, de la recherche environnementale, de la prise de décision financière, etc.

Avant de poursuivre, il convient d'expliquer ce que l'on entend par attribut dans un contexte de PLI. En effet, alors que la notion d'attribut est évidente dans le cadre de l'apprentissage attribut-valeur, elle a besoin d'être explicitée en PLI. En apprentissage attribut-valeur, les exemples sont souvent représentés comme la conjonction d'un nombre fixe de paires attribut-valeur :

$$E : (A1 = V1) \wedge (A2 = V2) \wedge (A3 = V3)$$

En PLI, un exemple est constitué de la conjonction d'un certain nombre de littéraux (nombre non défini a priori) :

$$E(A) \quad :- \quad P1(A, B), P2(B, C), P3(D).$$

Il apparaît légitime de penser le littéral comme la "brique" de base de la représentation en PLI. Dans ce papier, nous avons étendu la notion d'attribut dans un contexte de PLI, en considérant, par analogie, les littéraux d'une formule comme des attributs. En PLI, filtrer un exemple signifie supprimer ses littéraux considérés comme non-pertinents ou redondants.

Notons également qu'étant donnée la nature de son expressivité, la réduction de dimensionnalité en PLI est un besoin au moins aussi important qu'en apprentissage attribut-valeur, et pourtant, c'est un fait, les techniques de sélection d'attributs ne sont pas encore appliquées à la PLI. Pour comprendre cela, il faut se rappeler qu'en apprentissage attribut-valeur, les exemples sont présentés sous forme tabulaire, avec un nombre fixé d'attributs, alors qu'en PLI, les exemples sont constitués d'un nombre non-déterminé de littéraux, interdisant l'application directe des techniques usuelles de sélection d'attributs à la PLI.

### 1.3 Formulation du problème

Compte tenu de ce qui a été précédemment dit, notre tâche se définit comme étant *la suppression des littéraux non-pertinents des exemples relationnels présentés à un système d'apprentissage de PLI*. Dans la suite, nous désignerons cette tâche par “filtrage en PLI”.

Alphonse & Matwin (2002) ont proposé un paradigme établissant un parallèle entre la sélection d'attributs en apprentissage attribut-valeur et le filtrage en PLI. Ce paradigme montre que ce dernier problème est équivalent à effectuer une sélection d'attributs sur des problèmes particuliers d'apprentissage attribut-valeur, les problèmes multi-instances (PMI) (Dietterich *et al.*, 1997).

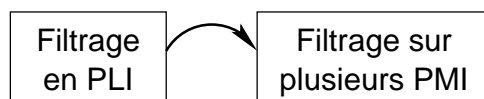


FIG. 1 – CHANGEMENT DE REPRÉSENTATION

Pour les lecteurs intéressés, une description complète du paradigme est disponible en annexe. Pour les autres, il leur suffit de savoir que filtrer en PLI revient à effectuer une sélection d'attributs sur des PMI.

Dans un PMI, un exemple n'est plus représenté comme un unique vecteur d'attributs, mais comme un ensemble de vecteurs. On désigne souvent cet ensemble par le terme de “sac”. Un exemple est donc un “sac” de vecteurs, contenant un nombre indéterminé d'instances. Un sac est étiqueté positif s'il contient au moins une instance positive, et négatif dans la cas contraire. La difficulté de ces problèmes réside dans le fait qu'on ne peut connaître, a priori, la (ou les) instance(s) positive(s) contenue(s) dans un sac positif. La seule information dont on dispose est qu'il en existe au moins une.

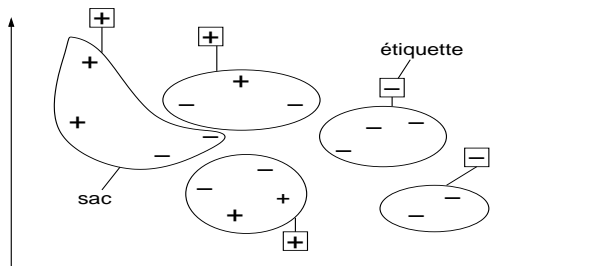


FIG. 2 – PROBLÈME MULTI-INSTANCE

Compte tenu de ces nouvelles informations, notre tâche consiste maintenant à *effectuer une sélection d'attributs sur des problèmes multi-instances*. Les PMI provenant d'une tâche de PLI présentent en outre un certain nombre de particularités, qui seront développées dans la section suivante.

## 1.4 Travaux antérieurs

La sélection d'attributs en PLI est une idée assez récente qui n'a pas encore donné lieu à de nombreux travaux. A notre connaissance, Lavrač *et al.* (1999) sont les seuls à avoir proposé un modèle relatif à ce problème. Toutefois, le langage de représentation qu'ils utilisent est DHDB, langage fortement contraint. D'une part, puisque DHDB n'autorise pas l'utilisation de variables existentielles, le test de couverture est quadratique et tous les problèmes exprimés en DHDB peuvent l'être en logique propositionnelle en un temps quadratique; les techniques de sélection d'attributs classiquement utilisées peuvent alors être appliquées. D'autre part, DHDB est un langage trop contraint pour pouvoir exprimer les données usuellement traitées en PLI, telles que mutagénèse ou les bases de données relationnelles. L'approche présentée ici se place à un niveau d'expressivité plus élevé permettant de traiter les problèmes sus-cités. A ce niveau d'expressivité, aucun modèle n'a encore été proposé pour traiter le problème du filtrage.

## 2 Particularités des PMI issus de la PLI

Les PMI issus de la PLI présentent un certain nombre de particularités. Tout d'abord, les sacs d'un tel PMI sont constitués de vecteurs booléens. Nous les représenterons donc comme des instances dont les attributs prennent les valeurs dans l'ensemble  $\{0, 1\}$ .

Ensuite, dans de tels PMI, il est nécessaire de se placer dans le cadre de *l'apprentissage par implication* et non en apprentissage par interprétation. En effet, dans une tâche de PLI, le but recherché est de trouver une hypothèse qui implique tous les exemples positifs et aucun négatif, ce qui est caractéristique de l'apprentissage par implication :

Etant donné un ensemble  $E^+$  d'exemples positifs et un ensemble  $E^-$  d'exemples négatifs, trouver une hypothèse  $h$ , telle que :

- $\forall e^+ \in E^+, h \geq e^+$
- $\forall e^- \in E^-, h \not\geq e^-$

Il existe une différence fondamentale entre les deux types d'apprentissage mentionnés. En effet, en apprentissage par interprétation, les instances sont des interprétations, incomparables par nature, alors qu'en apprentissage par implication, il existe un ordre partiel sur les instances. Cet ordre correspond à la notion de subsomption, i.e. de généralité.<sup>4</sup> En apprentissage par implication, on recherche une hypothèse au moins aussi générale que tous les positifs, et moins générale que tous les négatifs. De plus, ce type d'apprentissage présente la propriété suivante : dans ce cadre, la tâche de filtrage est identique à celle de l'apprentissage.

---

<sup>4</sup>Dans ce contexte, un zéro dénote l'absence d'attribut, et non sa négation. Un vecteur ne contenant que des zéros représente le concept le plus général qui soit.

Une autre particularité des PMI provenant de la PLI concerne le bruit. Le nombre de vecteurs constituant les sacs d'un tel PMI est très "grand". En fait, Sebag & Rouveirol (1997) ont montré que ce nombre était exponentiel en  $|e|$  (nombre d'exemples) et en  $|P|$  (nombre de littéraux de l'exemple en cours de filtrage). Toutefois, cet ensemble est fortement redondant, et en pratique, il suffit d'une approximation de cet ensemble, obtenue en effectuant sur lui un certain échantillonnage. On obtient alors ce qu'on appelle un PMI "borné". Dans ce papier, le nombre d'instances par sac est fixé à la valeur  $k$ .<sup>5</sup>

Toutefois, cette approximation conduit inévitablement à une introduction de bruit dans les données, bruit qu'il faudra prendre en compte lors de la tâche de filtrage.<sup>6</sup> Ce bruit est un bruit d'attribut unilatéral pouvant être à l'origine d'un changement de bit de 1 vers 0 dans les vecteurs booléens du problème multi-instances. Des précisions concernant l'origine de ce bruit sont disponibles en annexe.

De plus, dans le cas de l'apprentissage d'un concept disjonctif, un bruit de classe vient s'ajouter au bruit d'attribut. Ce bruit de classe est caractérisé par la possibilité pour un sac étiqueté positif d'être en réalité négatif. Soulignons, en outre, que ce bruit de classe particulier peut prendre des valeurs atteignant 80%<sup>7</sup>. Là encore, le lecteur curieux pourra se référer à l'annexe s'il veut comprendre les origines du bruit.

Nous arrivons finalement au résultat suivant : *filtrer en PLI revient à effectuer un apprentissage sur des problèmes multi-instances booléens, soumis à la fois à un bruit d'attribut et de classe, dans un cadre d'apprentissage par implication.*

### 3 Apprendre sur des problèmes multi-instances

Pour résoudre le problème du filtrage en PLI, il nous faut maintenant trouver un moyen d'apprendre dans un contexte multi-instances. Malheureusement, la plupart des travaux effectués en MI concernent l'apprentissage par interprétation et ne sont pas directement adaptables à notre problème.

Dans ce papier, nous nous plaçons dans une *approche Bayésienne*, dans laquelle les instances positives et négatives proviennent de distributions de probabilités. Nous supposons que ces distributions dépendent du concept cible  $h$ , et nous recherchons la valeur la plus vraisemblable pour  $h$ . Le résultat finalement trouvé est considéré comme étant le concept recherché par la tâche d'apprentissage.

---

<sup>5</sup>Même en effectuant une approximation, la valeur de  $k$  peut dépasser la centaine (c'est le cas dans le problème de mutagenèse), ce qui sensiblement plus élevé que la taille des sacs usuellement traités en MI, de l'ordre de la dizaine ou de la vingtaine d'instances.

<sup>6</sup>Seul le bruit provenant du changement de représentation est considéré, les données d'origine étant considérées non-bruitées.

<sup>7</sup>En terme de pourcentage d'exemples bruités.

Les données  $D$  sont composées d'un ensemble  $\{S_1^+, \dots, S_{np}^+\}$  de  $np$  sacs positifs et d'un ensemble  $\{S_1^-, \dots, S_{nn}^-\}$  de  $nn$  sacs négatifs. Chaque sac est lui-même composé de  $k$  instances  $\{S_{i1}^s, \dots, S_{ik}^s\}$ , où le  $s$  représente la nature du sac. Rappelons qu'une instance  $S_{ij}^+$  n'est pas nécessairement positive ; elle appartient seulement à un sac positif. Chaque instance est représentée par un vecteur booléen, ainsi que le concept cible  $h$ , qui appartient au même espace que celui des instances.

### 3.1 La fonction de vraisemblance

En supposant l'indépendance des sacs sachant  $h$ , la fonction de vraisemblance s'écrit  $P(D|h) = \prod_i P(S_i|h)$ . Dans un contexte MI, il est possible d'obtenir les  $P(S_i^-|h)$ , étant donné un modèle génératif pour les instances négatives, puisque, dans un sac négatif, toutes les instances sont négatives. Toutefois, il est très difficile d'accéder aux distributions sous-jacentes  $P(S_i^+|h)$ , puisque la seule information dont on dispose est qu'il existe au moins une instance positive à l'intérieur du sac positif présenté. Ceci est une autre formulation possible - plus pratique dans le cas multi-instances - de la fonction de vraisemblance :

$$P(D|h) = \prod_i P(S_i|h) = \prod_i \frac{P(h|S_i) \times P(S_i)}{P(h)} = \prod_i P(h|S_i) \times C \quad (1)$$

où  $C$  est constant par rapport à  $h$  si l'on considère que les  $h$  sont répartis uniformément dans l'espace des hypothèses.

La quantité à maximiser est maintenant  $\prod_i P(h|S_i)$ . Dans ce contexte, la fonction de vraisemblance est appelée *diverse densité* (Maron & Lozano-Pérez, 1998, Maron, 1998) et plusieurs modèles ont été proposés par Maron (1998) pour exprimer les  $P(h|S_i^+)$ . Un de ces modèles nommé le modèle de la *cause la plus vraisemblable* est défini par :

$$P(h|S_i^+) = \frac{\max_j P(h|S_{ij}^+)}{Z} \quad (2)$$

Cela se justifie par le fait que dans un sac positif, une instance au moins est positive. De façon similaire, le fait que toutes les instances sont positives dans un sac négatif peut être exprimé par :

$$P(h|S_i^-) = \frac{\min_j P(h|S_{ij}^-)}{Z}$$

où  $Z$  est un facteur de normalisation.

Ces modèles conduisent à une approximation des distributions sous-jacentes, mais sont beaucoup plus faciles à exprimer et à calculer que les distributions d'origine  $P(S_i^+|h)$ .

Il reste maintenant à définir  $P(h|S_{ij})$ , la probabilité que l'hypothèse soit  $h$  étant donnée l'instance  $S_{ij}$ . Dans le contexte de l'apprentissage par implication, une instance est positive ssi elle est plus spécifique que le concept cible, et négative sinon, soit :

$$P(h|S_{ij}^+) = \begin{cases} 1/Z & \text{si } h \geq S_{ij}^+ \\ 0 & \text{sinon} \end{cases} \quad P(h|S_{ij}^-) = \begin{cases} 0 & \text{si } h \geq S_{ij}^- \\ 1/Z & \text{sinon} \end{cases} \quad (3)$$

En raison de la nature du changement de représentation utilisé dans le paradigme, nous sommes confrontés à un bruit d'attribut, un bit d'une instance pouvant passer de la valeur 1 à la valeur 0. Dans ce cas, un modèle plus précis peut être :

$$P(h|S_{ij}^+) = \frac{pp^{nz}}{Z} \quad P(h|S_{ij}^-) = \frac{1 - pn^{nz}}{Z} \quad (4)$$

où  $pp$  et  $pn$  sont des paramètres de bruit, et  $nz$  le nombre de zéros dans la projection de  $S_{ij}$  sur  $h$ . Remarquons que lorsque  $pp$  et  $pn$  sont nuls, on retrouve exactement le modèle de l'équation (3).

## 4 Maximisation de la fonction de vraisemblance

Tous les éléments sont maintenant en notre possession pour calculer la fonction de vraisemblance ; nous cherchons à la maximiser. Notons, en outre, que si l'on considère la fonction de vraisemblance comme étant la mesure d'un sous-ensemble d'attributs (représenté par une hypothèse  $h$ ), notre tâche d'apprentissage se confond bien avec celle du filtrage, dans laquelle on cherche un sous-ensemble optimal par rapport à une certaine mesure.

Notre fonction étant discrète, il est impossible de la maximiser directement, par exemple en calculant son gradient. Par conséquent, nous sommes confrontés à un problème d'optimisation combinatoire. Puisqu'il est impensable de calculer LF pour tous les  $h$  possibles, une idée raisonnable est d'utiliser une méthode heuristique pour trouver son maximum. Pour utiliser une telle méthode, nous avons uniquement besoin de savoir calculer  $P(D|h)$  pour les instances de  $h$  requises par la recherche. On a vu dans la Sect. 3.1 comment effectuer un tel calcul.

Dans la suite, deux méthodes sont présentées : un algorithme glouton, et un algorithme fondé sur la méthode d'Espérance-Maximisation. Dans les deux cas l'algorithme est dit "*ascendant*" : le point de départ est l'hypothèse contenant tous les littéraux, et on procède par généralisations successives. Les deux méthodes adoptées sont fondées sur une notion de voisinage : deux hypothèses sont considérées voisines si leurs représentations sous forme de vecteurs booléens sont à une distance de Hamming de 1.

## 4.1 Méthode gloutonne

Dans cette méthode, l'idée est de choisir à chaque étape un voisin qui améliore la situation courante. C'est l'algorithme classique de descente, encore appelé amélioration itérative. Dans notre méthode, nous cherchons le meilleur voisin, i.e. la meilleure amélioration. En outre, on ne remet jamais en cause un choix de spécialisation, et la méthode est dite "gloutonne". Cela nous permet d'accepter un voisin de même valeur que la situation courante, et donc de dépasser des "plateaux", le caractère glouton de la méthode évitant le bouclage. Le résultat final trouvé est, de façon évidente, un maximum local de la fonction de vraisemblance.

L'algorithme est décrit dans la Table 1, dans laquelle  $vois(i, h)$  représente le voisin de  $h$  différant de lui par son  $i^{\text{ème}}$  bit.

TAB. 1 – ALGORITHME GLOUTON

---

### ALGORITHME GLOUTON

1. Mettre  $h$  à  $(1, 1, 1, \dots, 1, 1)$
  2. Répéter :
    - Faire  $h_{max} \leftarrow$  meilleur voisin (non-bloqué) de  $h$ , avec  $h_{max} = vois(i_{max}, h)$
    - si  $P(D|h_{max}) \geq P(D|h)$ , bloquer le changement du  $i_{max}^{\text{ème}}$  bit
    - sinon aller en 3
  3. Fin
- 

Le  $h$  final procure le concept cible et détermine de ce fait la sélection d'attributs. Cette méthode a été proposée d'une part pour sa simplicité et d'autre part parce que l'on espère, dans les cas favorables, être confronté à une fonction de vraisemblance présentant des propriétés de quasi-monotonie. Cela signifie, grosso modo, que l'on considère que le fait qu'un attribut soit pertinent ou pas ne dépend pas des autres attributs.

## 4.2 Méthode EM

L'idée d'utiliser l'algorithme EM dans notre problème a pour origine la constatation suivante :  $k$  peut prendre des valeurs arbitrairement grandes (classiquement, on aimerait pouvoir traiter des problèmes dans lesquels  $k = 1000$ ), et calculer  $P(D|h)$  (obtenu à partir des  $P(h|S_{i,j})$  de toutes les instances de tous les sacs) à chaque pas de l'algorithme peut être très coûteux.

Une idée pour réduire le nombre de calcul est de considérer que certaines instances dans les sacs ne sont pas "intéressantes". En effet, dans un sac positif, seule une instance positive est intéressante, et dans un sac négatif, une instance est d'autant plus informative qu'elle est spécifique. Toutefois, il

n'est pas permis de rejeter définitivement des instances, le caractère "intéressant" d'une instance dépendant du  $h$  courant. Nous sommes, par conséquent, en présence de variables inobservées (les variables "intéressantes" des sacs). L'algorithme EM (Espérance-Maximisation) (Dempster *et al.*, 1977, Zhang & Goldman, 2002) est une technique largement utilisée pour apprendre en présence de variables cachées. Notre approche est fondée sur l'algorithme EM, dans lequel les instances intéressantes des sacs sont considérées comme étant les variables cachées.

#### 4.2.1 Description de l'algorithme EM

Soit  $D$  les données observées,  $Z$  les variables cachées, et  $Y = D \cup Z$  l'intégralité des données. Exprimons  $P(Y|h)$  de la façon suivante :

$$P(Y|h) = \prod_i P(S_i, z_i|h) = \prod_i \frac{P(h, z_i|S_i) \times P(S_i)}{P(h)} = \prod_i P(h, z_i|S_i) \times C \quad (5)$$

où  $C$  est constant par rapport à  $h$  et  $z_i$  la variable aléatoire représentant l'indice de l'instance la plus intéressante dans le sac  $S_i$ .

L'algorithme EM est composé de deux étapes : la phase d'*espérance*, dans laquelle on calcule  $Q(h'|h) = E[\ln P(Y|h')|h, X]$ , et la phase de *maximisation*, où l'hypothèse courante  $h$  est remplacée par l'hypothèse  $h'$  qui maximise la fonction  $Q(h'|h)$ .

La quantité à maximiser ici est :

$$E \left[ \sum_i \ln P(h', z_i|S_i)|h, X \right] = \sum_i \sum_j P(z_i = j|h, X) \times \ln P(h', z_i = j|S_i)$$

où

$$P(z_i = j|h, X) = \begin{cases} pp^{nz}/Z & \text{si } S_i \text{ est un sac positif} \\ pn^{nz}/Z & \text{si } S_i \text{ est un sac négatif} \end{cases} \quad (6)$$

Ces distributions sont choisies de façon à donner le plus de poids aux instances intéressantes. Nous avons aussi :

$$P(h', z_i = j|S_i) = P(z_i = j|h', S_i) \times P(h'|S_i) \quad (7)$$

où les deux termes de l'équation ont précédemment été défini, respectivement dans les équations (6) et dans les équations (2) et (4).

Une itération de l'algorithme EM :

**Etape E :** Calculer  $P(z_i = j|h, X)$  pour tous les  $i$  et tous les  $j$

**Etape M :** Remplacer  $h$  par le  $h'$  qui maximise  $Q(h'|h)$

Ce procédé est itéré jusqu'à vérification d'une certaine condition d'arrêt.

## 4.2.2 Implantation de l'algorithme EM

Le schéma général de l'algorithme a été donné. Cependant, on remarque que l'on a toujours à calculer  $P(h', z_i = j | S_i)$  à chaque étape de notre algorithme, et le fait d'utiliser EM ne nous épargne aucun calcul. Quel intérêt alors a-t-on à utiliser EM ? Pour répondre à cette question, poussons plus loin notre raisonnement. En poursuivant l'idée que toutes les instances dans les sacs ne sont pas intéressantes, on peut s'attendre à ce que de nombreuses valeurs  $P(z_i = j | h, X)$  soient proches de zéro. Dans ce cas, une version d'EM appelée "*sparse algorithm*" (Neal & Hinton, 1998) stipule qu'il n'est pas nécessaire de recalculer ces valeurs à chaque étape d'espérance. Les petites valeurs sont ainsi gelées durant quelques itérations, et à l'occasion, on effectue une véritable étape d'espérance. Dans ce papier, nous utilisons une approximation de ce "*sparse algorithm*", en gelant les petites valeurs à zéro ; de cette manière, nous n'avons plus à calculer  $P(h', z_i = j | S_i)$  pour chaque  $j$ . Notons que si nous gelons toutes les valeurs à zéro sauf une, nous retrouvons l'algorithme EM-DD proposé par Zhang & Goldman (2002). Cependant, nous travaillons dans un espace discret, et l'algorithme proposé est trop instable dans notre cas ; nous devons considérer plus qu'une instance.

Pour définir notre algorithme, trois étapes sont nécessaires :

- Etape E :** Calculer  $P(z_i = j | h, X)$  pour tous les  $i$  et tous les  $j$   
Pour chaque sac, choisir les  $M$  meilleurs instances
- Etape E (rapide) :** Calculer  $P(z_i = j | h, X)$  pour tous les  $i$  et pour les  $M$  instances choisies  
Poser  $P(z_i = j | h, X) = 0$  pour les autres instances
- Etape M :** Faire  $h \leftarrow h'$ , avec  $h'$  qui maximise  $Q(h' | h)$

L'algorithme est finalement décrit dans la Table 2.

TAB. 2 – ALGORITHME EM

---

### ALGORITHME EM

1. Commencer avec une hypothèse  $h$  dont les attributs sont tous égaux à 1  
Effectuer une étape E et faire  $h\_max \leftarrow h$
  2. Alternier une étape E rapide avec une étape M jusqu'à atteinte d'un point stationnaire
  3. Effectuer une étape E et une étape M
  4. Si  $h \neq h\_max$ , faire  $h\_max \leftarrow h$  et aller en 2
  5. Sinon Fin
- 

Ici encore, le  $h$  final procure le concept cible et détermine de ce fait la sélection d'attributs.

## 5 Expérimentations

Pour valider notre approche, nous avons effectué un certain nombre de tests, pour l'instant sur des problèmes artificiels. Nous avons principalement travaillé sur deux formulations du problème des trains de Michalski : une dans laquelle le concept relationnel cible est conjonctif, et une dans laquelle il est disjonctif (3 sous-concepts).

Voici quelques caractéristiques de ces problèmes :

- **Nombre d'exemples** : 100
- **Nombre d'attributs** : environ 30
- **Nombre d'instances par sac** : environ 100
- **Bruit d'attribut** : environ 0.6% par bit.
- **Bruit de classe** : 0% des exemples pour les trains conjonctifs, jusqu'à 80% des exemples pour les trains disjonctifs.

Ainsi, même si ces problèmes sont artificiels, ce ne sont pas des problèmes jouets. Pour preuve, les données usuellement traitées par la communauté de la sélection d'attributs sont caractérisées par un nombre d'attributs de l'ordre de la vingtaine, et le nombre d'instances par sac dans les problèmes étudiés par la communauté du multi-instances est de l'ordre de 20 également.

Voici les différentes étapes de validation de l'approche :

- On génère deux bases d'exemples indépendantes : une base d'apprentissage, et une base de test,<sup>8</sup>
- On filtre la base d'apprentissage, tout en conservant une copie de la base d'origine (non-filtrée),
- On effectue l'apprentissage sur les bases filtrée et non-filtrée,
- On mesure la "qualité" des concept appris, dans les deux cas, sur la base de test.

L'algorithme utilisé lors de l'apprentissage relationnel est PROPAL, un algorithme développé par Alphonse & Rouveirol (2000). Lors de cette phase, nous avons accès au nombre d'hypothèses testées. La "qualité" d'un concept est défini comme étant le pourcentage d'exemples de la base de test n'invalidant pas le concept appris.

Lors de chaque test, nous avons mesuré :

- Le résultat sur la tâche d'apprentissage sans filtrage,
- Le résultat sur la tâche d'apprentissage avec filtrage,
- Le nombre d'hypothèses testées lors de la tâche d'apprentissage sans filtrage,
- Le nombre d'hypothèses testées lors de la tâche d'apprentissage avec filtrage,
- Le ratio de filtrage (rapport moyen de la taille des exemples filtrés sur la taille des exemples non-filtrés).

---

<sup>8</sup>Dans les deux formulations, les bases de tests sont constituées de 100 exemples, i.e. autant que pour les bases d'apprentissage.

Dans la suite, les mesures présentées correspondent à un moyennage<sup>9</sup> sur une dizaine de tests, pour chaque problème et chaque algorithme.

### 5.1 Algorithme Glouton sur les trains conjonctifs

	Résultats	Ratio de filtrage	#hypothèses testées
Sans filtrage	99%	100%	112
Avec filtrage	100%	12,9%	2

Comme attendu, l'algorithme effectue un filtrage efficace. En effet, pour chaque exemple, il conserve les littéraux pertinents, et ne conserve qu'eux. On a donc à la fois une efficacité et une précision maximale. Notons également que suite au filtrage, la tâche d'apprentissage est simplifiée : le nombre d'hypothèses à tester devient dérisoire, et cela se concrétise par une forte diminution du temps d'exécution.

### 5.2 Algorithme EM sur les trains conjonctifs

	Résultats	Ratio de filtrage	#hypothèses testées
Sans filtrage	99%	100%	112
Avec filtrage	100%	26,8%	2

Là encore, le filtrage est efficace et répond à nos attentes. Il ne fait aucune erreur sur les exemples : aucun attribut pertinent n'est supprimé. Notons toutefois que le ratio de filtrage est plus élevé que dans l'algorithme glouton, ce qui signifie que certains attributs non-pertinents sont conservés. Cela se justifie simplement par le fait que, dans l'algorithme EM, pour plusieurs hypothèses prenant la même valeur sur la fonction de vraisemblance, il n'y a pas de biais favorisant l'hypothèse contenant le moins de littéraux, alors que ce biais est présent dans l'algorithme glouton.

### 5.3 Algorithme Glouton sur les trains disjonctifs

	Résultats	Ratio de filtrage	#hypothèses testées
Sans filtrage	91%	100%	1230
Avec filtrage	86%	28,2%	401

Alors que le bruit d'attribut est inclus à l'intérieur même du modèle (cf. choix des distributions), le bruit de classe n'est pas traité explicitement. Dans un premier temps, une idée a été de ne pas prendre en compte spécifiquement

<sup>9</sup>Le besoin de moyennage s'explique par la nature stochastique de l'échantillonnage effectué sur les instances (cf. Sect. 2).

ce bruit de classe, et de le considérer comme un bruit d'attribut supplémentaire. Toutefois, dans ce cas, la mesure de vraisemblance perd un peu de son sens, et l'algorithme glouton est trop "violent" (fort biais, non remise en cause des choix, etc.). Cela explique les résultats trouvés : l'algorithme filtre toujours (il y a une réduction de la taille des exemples), mais trop ; il arrive de se tromper, i.e. de supprimer des littéraux pertinents.

#### 5.4 Algorithme EM sur les trains disjonctifs

	Résultats	Ratio de filtrage	#hypotheses testées
Sans filtrage	91%	100%	1230
Avec filtrage	94%	61,7%	245

Dans l'exemple des trains disjonctifs, l'algorithme EM résiste bien au bruit de classe. En effet, il filtre convenablement, et arrive même à augmenter les résultats sur la tâche d'apprentissage. Contrairement à l'algorithme glouton, EM ne prend pas trop de risque (pas de fort biais), et par conséquent fait moins de choix inconsidérés. Il est prudent ("seulement" 61,7% en ratio de filtrage), et ne commet, de ce fait, que très peu d'erreur.

#### 5.5 Commentaires et remarques

A cet égard, rappelons qu'obtenir un très faible ratio de filtrage n'est pas la priorité première de notre tâche. En effet, un filtre qui ne fait rien est tout de même un filtre, qui aura par ailleurs les mêmes résultats sur la tâche d'apprentissage que la version non-filtrée. En outre, la complexité de la tâche d'apprentissage diminue de façon exponentielle par rapport au nombre de littéraux filtrés ; en pratique, il suffit de ne filtrer que très peu de littéraux pour apprécier l'apport du filtrage. Notre objectif est donc de filtrer, même peu, mais sans faire d'erreur.

Selon les critères que l'on vient de définir, les résultats obtenus sur les deux problèmes sont compréhensibles, cohérents, et surtout très encourageants. L'algorithme glouton réagit très bien face au cas conjonctif, mais résiste très mal au bruit de classe, résultat auquel on pouvait s'attendre. L'algorithme EM, quant à lui, filtre moins sévèrement, mais ne dégrade jamais le résultat sur la tâche d'apprentissage, même dans le cas disjonctif. Par rapport aux critères précédemment évoqués, c'est donc l'algorithme EM qui répond le mieux à nos attentes.

Quelques essais ont été entrepris sur des problèmes réels, tels que mutagénèse. Pour l'instant, les résultats obtenus sont insuffisants pour pouvoir conclure. Cependant, nous ne croyons pas pouvoir résoudre de tels problèmes, hautement disjonctifs, en continuant à traiter le bruit de classe comme un bruit d'attribut, i.e. en ne le prenant pas en compte explicitement.

## 6 Conclusion et perspectives

Nous avons présenté, dans ce papier, des méthodes de sélection d'attributs s'appliquant à la Programmation Logique Inductive. Dans cette optique, nous avons utilisé un paradigme existant, qui réduit le problème du filtrage en PLI à un problème de sélection d'attributs sur des problèmes multi-instances présentant certaines particularités. Après avoir identifié ces particularités, et montré que notre problème consistait, en réalité, à effectuer un apprentissage sur ces problèmes multi-instances particuliers, nous avons pu proposer un modèle adapté, fondé sur une approche probabiliste Bayésienne.

Cette approche s'est concrétisée par le développement de deux algorithmes différents. Elle a ensuite été validée sur des problèmes artificiels, mais non-triviaux, et les résultats obtenus sont, de ce fait, encourageants. Ce faisant, de nombreuses perspectives liées à l'application de la PLI sont désormais envisageables. Rappelons, par ailleurs, que le filtrage en PLI est un domaine nouveau, qui n'a encore jamais été exploré.

Pour la suite de nos travaux, nous comptons étudier et développer des méthodes permettant de traiter explicitement le bruit de classe, de façon à pouvoir nous confronter à plusieurs problèmes réels, tels que DU ("Document Understanding"), et mutagénèse, problèmes hautement disjonctifs.

## Remerciements

Je tiens à remercier tous les membres de l'équipe Inférence et Apprentissage du LRI, et plus particulièrement Erick Alphonse et Stan Matwin, qui ont su, tout au long de ce stage, m'aider, me guider, me conseiller et donner de leur temps précieux pour discuter longuement avec moi.

## Références

- ALMUALLIM H. & DIETTERICH T. G. (1994). Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, **69**(1-2), 279–305.
- ALPHONSE E. & MATWIN S. (2002). Feature subset selection and inductive logic programming. In *Proc. of the 19<sup>th</sup> International Conference on Machine Learning*. (A paraître).
- ALPHONSE E. & ROUVEIROL C. (2000). Lazy propositionalization for relational learning. In W. HORN, Ed., *Proc. of the 14<sup>th</sup> European Conference on Artificial Intelligence*, p. 256–260 : IOS Press.
- DEMPSTER A., LAIRD N. & RUBIN D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, **B**, 39(1) : 1–38.
- T. DIETTERICH, S. BECKER & Z. GHAHRAMANI, Eds. (2002). *Proc. Advances in Neural Information Processing Systems 14*. The MIT Press.
- DIETTERICH T. G., LATHROP R. H. & LOZANO-PÉREZ T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, **89**(1–2), 31–71.
- HALL M. (1998). *Correlation-based Feature Selection for Machine Learning*. PhD thesis, Waikato University.
- KIRA K. & RENDELL L. A. (1992). A practical approach to feature selection. In *Proc. of the 9<sup>th</sup> International Conference on Machine Learning*, p. 249–256 : Morgan Kaufmann.
- KOHAVI R. & JOHN G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, **97**(1-2), 273–324.
- LAVRAČ N. & DŽEROSKI S. (1994). *Inductive Logic Programming : Techniques and Applications*. Ellis Horwood.
- LAVRAČ N., GAMBERGER D. & JOVANOSKI V. (1999). A study of relevance for learning in deductive databases. *Journal of Logic Programming*, **40**(2-3), 215–249.
- LIU H. & MOTODA H. (1998). *Feature Extraction, Construction and Selection : A Data Mining Perspective*. Kluwer Academic Publishers.
- MARON O. (1998). *Learning from ambiguity*. PhD thesis, MIT.
- MARON O. & LOZANO-PÉREZ T. (1998). A framework for multiple-instance learning. In M. I. JORDAN, M. J. KEARNS & S. A. SOLLA, Eds., *Advances in Neural Information Processing Systems 10* : The MIT Press.
- NEAL R. & HINTON G. (1998). A view of the em algorithm that justifies incremental, sparse, and other variants. In M. I. JORDAN, Ed., *Learning in Graphical Models* : Kluwer Academic Publishers.

- SEBAG M. & ROUVEIROL C. (1997). Tractable induction and classification in first order logic via stochastic matching. In *15<sup>th</sup> International Joint Conferences on Artificial Intelligence*, p. 888–893 : Morgan Kaufmann.
- ZHANG Q. & GOLDMAN S. A. (2002). Em-dd : An improved multiple-instance learning technique. In T. G. DIETTERICH, S. BECKER & Z. GHAHRAMANI, Eds., *Advances in Neural Information Processing Systems 14* : The MIT Press.
- ZUCKER J.-D. & GANASCIA J.-G. (1996). Changes of representation for efficient learning in structural domains. In *Proc. of 13<sup>th</sup> International Conference on Machine Learning* : Morgan Kaufmann.

# Annexes

## Contexte du paradigme

Il a été précédemment mentionné que dans un contexte de PLI, les exemples, la connaissance de base et le résultat, s'expriment dans un sous-ensemble de la logique du premier ordre. Dans le paradigme présenté, le sous-ensemble choisi est Datalog, i.e. l'ensemble des clauses de Horn non-récurrentes qui ne contiennent pas des symboles de fonction autres que des constantes. Dans cet ensemble, l'implication logique est équivalente à la  $\theta$ -subsumption, qui est décidable.

## Le paradigme

L'essence du paradigme permettant de passer de la représentation flexible des exemples relationnels à une représentation fixe nécessaire à la sélection d'attributs en attribut-valeur, réside dans l'idée de filtrer les exemples un par un. L'ensemble des littéraux de l'exemple en cours de filtrage est alors utilisé comme un ensemble fixe d'attributs, et l'ensemble du problème relationnel est redécrit en fonction de ces attributs (phase dite de propositionnalisation). Le vecteur  $P$  représentant ces attributs est appelé motif de propositionnalisation. Pour redécrire un exemple en fonction de ces attributs, on extrait toutes les substitutions (éventuellement partielles) entre lui et  $P$ , chaque substitution étant représentée par un vecteur booléen.

Ce changement de représentation convertit donc chaque autre exemple relationnel en un sac de vecteurs d'attributs booléens (chaque vecteur correspondant à une substitution entre cet exemple et l'exemple en cours de filtrage), sac étiqueté de la même façon que l'exemple, et nous fait voir la représentation propositionnelle comme un problème multi-instances. Cette propositionnalisation multi-instances (PMI<sup>10</sup>) a par ailleurs fait l'objet de plusieurs travaux (Zucker & Ganascia, 1996, Alphonse & Rouveirol, 2000).

Tous les vecteurs résultants de la reformulation du problème relationnel d'origine sont ensuite traités par un système de sélection d'attributs approprié. Ayant déterminé quels étaient les attributs non-pertinents dans ce problème, nous pouvons également juger de la pertinence des littéraux de l'exemple relationnel d'origine, et par conséquent le filtrer. Ensuite, ce procédé sera réitéré en considérant chaque exemple positif<sup>11</sup> comme exemple courant.

Le procédé est récapitulé dans la Table 3.

---

<sup>10</sup>Dans la suite, nous ne distinguerons pas le changement de représentation (la propositionnalisation multi-instances) et son résultat, un problème multi-instances, tous deux représentés par le sigle PMI.

<sup>11</sup>Il n'est pas nécessaire de filtrer les exemples négatifs, qui sont d'autant plus informatifs qu'ils sont spécifiques.



TAB. 4 – LA REPRÉSENTATION TABULAIRE D’UN PROBLÈME RELATIONNEL

P	wag(V,W)	court(W)	char(W,X)	rect(X)	wag(V,Y)	court(Y)	char(Y,Z)
$\sigma_P$	1	1	1	1	1	1	1
$\sigma_{E_2,1}$	1	0	1	1	1	0	1
$\sigma_{E_2,2}$	1	0	1	1	1	0	0
...							
$\sigma_{E_2,i}$	1	0	0	0	1	1	1
...							
$\sigma_{NE,1}$	1	0	1	0	1	0	1
$\sigma_{NE,2}$	1	0	1	0	1	1	1
...							
$\sigma_{NE,j}$	1	0	1	1	1	1	1
...							

3. La méthode de sélection d’attributs est ensuite appliquée sur le problème multi-instances. Dans l’exemple ci-dessus, les attributs représentant les littéraux  $wagon(V, W)$ ,  $court(W)$  et  $wagon(V, Y)$  sont considérés comme étant non-pertinents.

4. L’exemple filtré est maintenant :

$$E_1^f = charge(W, X), rect(X), court(Y), charge(Y, Z).$$

Détaillons les étapes 2 et 3 de façon à mieux les comprendre. Dans l’étape 2, les vecteurs résultants d’une unification entre  $P$  et  $E_2$  sont étiquetés  $\sigma_{E_2,i}$  ; ceux provenant de l’exemple négatif  $NE$ ,  $\sigma_{NE,j}$ . Pour des raisons pratiques, nous ne distinguerons pas par la suite une substitution et son vecteur booléen associé. Tous ces vecteurs forment un sac positif (provenant de  $E_2$ ), et un sac négatif (provenant de  $NE$ ). De cette façon, nous avons bien converti le problème relationnel en un problème multi-instances.

Dans l’étape 3, un examen rapide des données montre que les colonnes correspondant à  $wagon(V, W)$ ,  $court(W)$  et  $wagon(V, Y)$  ne sont pas pertinentes (elles ont les mêmes valeurs pour les exemples positifs et négatifs). Cela se comprend aisément puisque le fait d’avoir un wagon court ou d’avoir au moins deux wagons ne permet pas de discriminer les trains Est des trains Ouest dans notre exemple. Cela illustre, en outre, un problème spécifique au filtrage en PLI, qui n’était pas présent en sélection d’attributs en apprentissage attribut-valeur. Dans l’exemple précédent, si nous supprimons les deux premiers littéraux, les littéraux restants ne sont plus liés à la tête de la clause, et l’exemple perd son sens. Ce problème spécifique à la PLI peut néanmoins être traité en aval du paradigme, et concerne davantage la PLI que la sélection d’attributs. Il est donc possible de travailler de façon indépendante, et de ne pas en tenir compte à notre niveau. Des solutions possibles de ce problèmes sont par ailleurs proposées dans (Alphonse & Matwin, 2002).

## Le bruit d'attribut

Le bruit provenant du changement de représentation diffère selon la nature conjonctive ou disjonctive du concept cible.

Tout d'abord, concentrons nous sur le cas d'un concept conjonctif. Supposons que lors de la phase de propositionnalisation, tous les vecteurs correspondant à une substitution soient retenus. Dans ce cas, il n'y a pas de bruit, et trouver une filtre parfait revient à résoudre le problème d'apprentissage. Toutefois, comme cela a déjà été mentionné, l'espace de toutes les substitutions possibles (de taille exponentielle en  $|P|$ , la taille de l'exemple filtré et en  $|e|$ , le nombre d'exemples), est beaucoup trop grand pour être utilisable sur des problèmes réels. Pour surmonter cette difficulté, la notion de propositionnalisation multi-instances bornée a été proposée. Dans cette propositionnalisation, on effectue un échantillonnage sur l'ensemble de toutes les substitutions possibles. Si l'échantillonnage permettait de sélectionner toutes les instances incomparables les plus spécifiques, il n'y aurait pas de perte d'information. Malheureusement, là encore, l'espace en question est de taille exponentielle en  $|P|$  et  $|e|$ . Par conséquent, s'il on effectue un échantillonnage raisonnable en terme calculatoire (polynômial en  $|P|$  et  $|e|$ ), on prend le risque de "manquer" une instance incomparable aux instances échantillonnées. Ce risque peut alors être modélisé par une introduction de bruit dans les données. En effet, les instances obtenues par échantillonnage apparaissent comme des généralisations des instances incomparables les plus spécifiques, et l'approximation liée à l'échantillonnage peut être vue comme un bruit d'attribut, bruit unilatéral pouvant être à l'origine d'un changement de bit de 1 vers 0 dans les vecteurs booléens du problème multi-instances.

## Le bruit de classe

En deuxième lieu, considérons le cas disjonctif, dans lequel un concept est la disjonction de plusieurs sous-concepts, les exemples étant présentés sous forme conjonctive, et appartenant à l'un (ou plus) des sous-concepts. C'est le cas, par exemple du concept *grand\_pere*( $X, Z$ ), disjonction des sous-concepts conjonctifs *pere*( $X, Y$ ), *pere*( $Y, Z$ ) et *pere*( $X, Y$ ), *mere*( $Y, Z$ ) Ici, le concept cible est le sous-concept auquel le motif appartient, puisque la tâche de filtrage concerne uniquement l'exemple relatif au motif. Dans notre exemple, si l'exemple relatif au motif est *pere*( $X, Y$ ), *pere*( $Y, Z$ ), *frere*( $Z, W$ ), le concept à apprendre est *pere*( $X, Y$ ), *pere*( $Y, Z$ ). Cependant, dans le cas disjonctif, le motif et l'exemple reformulé peuvent ne pas appartenir au même sous-concept, et par conséquent l'exemple reformulé peut ne pas contenir les littéraux caractéristiques du concept cible. Dans ce cas, il n'y a pas d'unification (sous aucune substitution) entre le motif et l'exemple, et toutes les instances dans le sac sont négatives. Par conséquent, le sac est lui-même négatif, par définition du problème multi-instances. De cette manière, un bruit

de classe a été introduit dans les données : des exemples sont considérés comme étant positifs (dans l'ensemble d'apprentissage, ils sont représentatifs du concept) alors que ce sont des exemples négatifs relativement au motif (ils appartiennent à un sous-concept différent de celui-ci) dans notre problème multi-instances. Ainsi,  $pere(X, Y)$ ,  $mere(Y, Z)$ ,  $soeur(Z, W)$  est bien représentatif du concept  $grand\_pere(X, Z)$ , mais c'est un exemple négatif relativement au sous-concept  $pere(X, Y)$ ,  $pere(Y, Z)$ . Signalons, par ailleurs, que ce bruit de classe peut atteindre des valeurs avoisinant les 80%, un sous-concept pouvant être fortement sous-représenté.