# GloBo : un algorithme stochastique pour l'apprentissage supervisé et non-supervisé

Fabien Torre

Équipe Inférence et Apprentissage, Laboratoire de Recherche en Informatique,
Bâtiment 490, Université Paris-Sud, 91405 - Orsay Cedex
fabien@lri.fr et http://www.lri.fr/~fabien

**Résumé**

Il est maintenant acquis que de très fortes relations existent entre l'apprentissage supervisé et non-supervisé [4, 14].

L'approche décrite dans ce papier est basée sur l'idée que le supervisé et le non-supervisé peuvent être vus comme les instances particulières d'un problème plus général de couverture minimale. Le système GloBo a pour objectif de caractériser les sous-concepts vérifiant une propriété particulière : la correction par rapport aux exemples négatifs dans le cas du supervisé, et une spécificité suffisante en non-supervisé.

Ce système est stochastique et travaille en temps polynômial. Les expérimentations menées sur des problèmes bien connus, en supervisé comme en non-supervisé, valident notre approche.

## 1 Introduction

Il est maintenant acquis que de très fortes relations existent entre l'apprentissage supervisé et non-supervisé [4, 14] : il a été souligné que les méthodes d'apprentissage supervisé pouvaient réaliser des tâches non-supervisées, que les êtres humains apprenaient sans étiquetages explicites des exemples, ou encore que l'apprentissage non-supervisé pouvait être utilisé sur des données non-étiquetées en vue d'enrichir un corpus qui lui, est étiqueté [4].

En apprentissage supervisé, un problème est défini comme suit : étant donnés les ensembles $E^+$ et $E^-$ d'exemples positifs et négatifs pour le concept à apprendre, $S$ est solution ssi

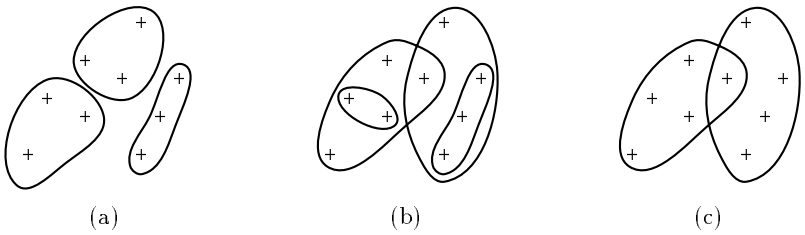
$$\forall e^+ \in E^+, S \geq e^+$$
$$\forall e^- \in E^-, S \not\geq e^-$$

où $\geq$ est un test de subsomption. Il existe beaucoup de méthodes supervisées mais très peu s'intéressent à la partition (la couverture plus exactement) des exemples positifs induite par une solution. La tendance générale est de voir la généralisation comme un problème de recherche [17], et les efforts ont été principalement dévoués à l'optimisation de la recherche et à l'élagage de l'espace de recherche. Tom Mitchell déjà proposait un élagage par rapport à la correction et à la complétude [17]. Plus récemment, des pré-ordres baptisés *relations naturelles* [26] ont été définis pour permettre un élagage dynamique de l'espace de recherche par rapport à toute propriété connue de la solution. Enfin, la recherche elle-même peut être optimisée, par exemple en imposant que chaque hypothèse ne doit être générée qu'une seule fois durant la recherche : ce sont les *opérateurs optimaux* [6].

Malheureusement, l'extension de ces heuristiques du conjonctif au disjonctif est difficile. Quand le langage des hypothèses n'autorise pas l'existence d'une solution conjonctive, il est nécessaire de rechercher une disjonction d'hypothèses conjonctives. Cependant, les optimisations évoquées plus haut deviennent trop faibles face à la taille du nouvel espace de recherche. De plus, si en apprentissage conjonctif toutes les solutions sont équivalentes par rapport à $E^+$ et $E^-$ (elles sont toutes correctes et complètes), il en va différemment lorsque le problème est disjonctif : les solutions sont toujours toutes correctes et complètes mais chacune d'elles induit une couverture différente de $E^+$ comme cela est illustré en Figure 1.

À ce propos, ni les méthodes que nous venons d'évoquer, ni les approches *divide-and-conquer* ou *covering* [3], ne garantissent de trouver une solution de taille minimale (en terme de nombre de clusters dans la couverture), ni même d'obtenir une solution de taille raisonnable. Plus particulièrement, ces méthodes peuvent fournir une solution disjonctive (plusieurs clusters) alors qu'une solution conjonctive (un seul cluster reprenant tous les exemples positifs) existe. Finalement, ces méthodes sont sujettes à l'éclatement injustifié de $E^+$ et à l'explosion de la taille de la disjonction découverte.

Considérons maintenant l'apprentissage non-supervisé. Son but est, étant donné un ensemble d'ins-



(a) (b) (c)

FIG. 1 – *Différentes couvertures.*

tances, de construire un ensemble de concepts qui classifient ces instances [13].

D'une part, nombre de méthodes sont fournies par l'Analyse de Données. Par exemple, la *classification ascendante hiérarchique* [7] réalise un apprentissage non-supervisé de la manière suivante : on démarre avec une instance dans une classe, puis deux classes (choisies selon un indice de similarité) sont fusionnées, et l'on recommence jusqu'à ce que toutes les instances soient dans la même classe. Ainsi, à chaque étape, une partition différente des instances est découverte ; ces partitions permettent finalement de construire une hiérarchie de concepts. Parmi les algorithmes de classification par partitions, évoquons rapidement les *nuées dynamiques* de [7]. $k$ noyaux sont choisis aléatoirement, puis chaque point (instance) est associé au noyau le plus proche, obtenant ainsi $k$ clusters ; ensuite, chaque noyau est remplacé par le centre de gravité de son cluster, et l'on recommence à partir de ces nouveaux noyaux. Les nuées dynamiques sont capables d'optimiser un critère donné sur les clusters à former. Par contre, cette méthode ne permet de trouver un nombre minimal de clusters, puisque la valeur de $k$ doit être fixée à l'avance.

D'autre part, selon [12] (COBWEB) et [13] (CLASSIT), on peut voir l'apprentissage non-supervisé comme une recherche à travers l'espace des hiérarchies. À nouveau, nous trouvons les problèmes soulevés par l'apprentissage supervisé : efficacité de la recherche, optimalité du parcours de l'espace de recherche, définition d'opérateurs, absence de contrôle sur la taille des solutions, etc.

L'approche décrite dans ce papier est basée sur l'idée que le supervisé et le non-supervisé peuvent être vus comme les instances particulières d'un problème plus général de couverture minimale.

Le papier est organisé comme suit. La Section 2 présente quelques définitions et notations générales. La Section 3 décrit l'algorithme générique GloBo et comment il peut être instancié pour réaliser de l'apprentissage soit supervisé, soit non-supervisé. Ensuite, la Section 4 détaille nos expérimentations illustrant les avantages de notre approche sur des problèmes bien connus dans les domaines du supervisé et du non-supervisé. Enfin, dans la Section 5, nous discutons de cette approche vis-à-vis d'autres travaux et décrivons les perspectives ouvertes par ce travail.

# 2 Définitions

**Définition 1 (cluster)**
*Étant donné un ensemble $E$, un* cluster *est un sous-ensemble de $E$.*

**Définition 2 (admissibilité)**
*Un cluster $C$ est* admissible *par rapport à une propriété $P$ ssi $C$ satisfait $P$, autrement dit si $P(C)$ est vrai.*

**Définition 3 (admissibilité maximale)**
*Un cluster $C$ est* maximalement admissible *par rapport à une propriété $P$ ssi $C$ est admissible par rapport à $P$ et si pour tout $e$ n'appartenant pas à $C$, $C \cup \{e\}$ n'est pas admissible par rapport à $P$.*

**Définition 4 (couverture minimale)**
*Étant donné un ensemble $E$ et une propriété $P$ sur les clusters de $E$, le problème de* couverture minimale par des clusters maximalement admissibles *consiste à trouver une couverture minimale de $E$, en utilisant des clusters de $E$ qui soient maximalement admissibles par rapport à $P$. Autrement dit, il s'agit de trouver un nombre minimal de clusters maximalement admissibles $C_1, \ldots, C_n$, tels que leur union couvre $E$.*

Notez que cette définition n'exige pas que les clusters soient disjoints : le recouvrement des clusters nous importe peu, seule la minimalité de la couverture nous intéresse.

Dans la suite, nous nous restreignons aux propriétés $P$ telle que si un ensemble $E$ vérifie $P$ alors chaque sous-ensemble de $E$ vérifie aussi $P$, c'est-à-dire :

$$P(A \cup B) \Rightarrow P(A) \wedge P(B) \qquad (1)$$

# 3 L'algorithme GloBo

## 3.1 Cas général

L'algorithme principal de GloBo est basé sur la construction d'un cluster autour d'un élément particulier appelé *graine*. Le cluster construit doit être maximalement admissible : ce cluster doit satisfaire $P$ et l'ajout

dans ce cluster d'un élément non déjà présent rend la propriété $P$ fausse. Pour cela, nous allons simplement parcourir la liste des éléments autres que la graine, et les ajouter au cluster si l'admissibilité par rapport à $P$ est préservée.

**Algorithme 1 (cluster maximalement admissible)**
*Étant donnés une propriété $P$, une graine $s$ et $E' = \{e_1, e_2 \ldots, e_{|E|-1}\}$ les autres éléments.*

1. *Soit $C = \{s\}$;*
2. *Pour $i$ dans $1 \ldots |E| \Leftrightarrow 1$ faire : si $C \cup \{e_i\}$ satisfait $P$ alors $C = C \cup \{e_i\}$;*
3. *Retourner $C$.*

Par construction, un cluster fourni par cet algorithme est maximalement admissible par rapport à $P$. Dans une approche AQ [16], on dirait que cet algorithme construit un élément de l'étoile de la graine.

($|E| \Leftrightarrow 1$) agrégations d'un élément et ($|E| \Leftrightarrow 1$) tests de $P$ sont nécessaires pour obtenir un tel cluster. Évidemment, le résultat $C$ dépend de l'ordre des exemples dans $E'$ : plusieurs clusters maximalement admissibles peuvent être obtenus à partir d'une même graine $s$, mais en utilisant des ordres différents de $E'$. Cependant, grâce à l'Équation 1, nous avons la garantie que chaque cluster maximalement admissible est constructible par ajout d'un unique élément à chaque étape, et en conservant à chaque fois l'admissibilité par rapport à $P$.

Pour annuler la dépendance vis-à-vis de l'ordre des exemples, GloBo construit un cluster maximalement admissible pour chaque élément de $E$ : chacun de ces éléments étant tour à tour utilisé comme graine et les autres éléments étant présentés dans des ordres différents à chaque fois (ces listes sont mélangées aléatoirement).

Il s'agit cette fois d'une différence nette par rapport à AQ [16] ; en effet, celui-ci calcule uniquement les étoiles des exemples qui ne sont pas couverts par des étoiles déjà calculées.

Par conséquent, un sous-concept peut apparaître à chaque fois que l'on construit un cluster en utilisant un élément de ce sous-concept comme graine. Le clustering échoue si un sous-concept n'apparaît pas parmi les clusters construits (nous verrons plus loin la probabilité d'un tel événement).

GloBo choisit ensuite les clusters les plus intéressants en construisant une couverture de $E$ par un nombre minimal de clusters. Ce problème est NP-difficile ; pour notre part, nous utilisons l'heuristique qui consiste à privilégier les clusters les plus importants [18].

**Algorithme 2 (couverture minimale)**
*Étant donnés $E$ l'ensemble à couvrir et $\{C_i\}$ l'ensemble des clusters candidats.*

1. *Soient noncouverts $= E$ et solution $= \emptyset$.*
2. *Tant que noncouverts $\neq \emptyset$ Faire*
   (a) *Soit $C$ le cluster qui a la plus grande intersection avec noncouverts.*
   (b) *Les éléments de $C$ sont effacés de noncouverts et $C$ est ajouté à la solution.*
3. *Retourner solution.*

Comme tous les éléments ont successivement été utilisés comme graine, il s'en suit que, tous ensemble, les clusters $C_i$ couvrent $E$. Par suite, il existe une couverture de $E$ par les $C_i$ et cet algorithme va terminer. Une couverture minimale approchée d'un ensemble de taille $m$ par $n$ ensembles, en utilisant cette méthode, a une complexité en $mn$ [18]. Ainsi, GloBo trouvera une couverture de $E$ avec une complexité au pire en $|E|^2$.

**Algorithme 3 (GloBo)**
*Étant donnés une propriété $P$ et un algorithme pour calculer la caractéristation d'un cluster.*

1. *Pour chaque $e_i$ de $E$ : calculer $C_i$ un cluster maximalement admissible avec $e_i$ comme graine et $E \Leftrightarrow \{e_i\}$ mélangé aléatoirement comme candidats (Algorithme 1).*
2. *Calculer une couverture minimale de $E$ par des $C_i$ (Algorithme 2).*
3. *Calculer $g_i$ la caractérisation de chaque $C_i$ présent dans la couverture de $E$.*
4. *Retourner la disjonction des $g_i$.*

L'étape de construction d'un cluster est itérée $|E|$ fois. Par conséquent, la complexité de notre algorithme est en $\Theta(|E|^2)$, en termes d'agrégations d'élément à un cluster et de vérifications de $P$. À cette complexité, il faudrait ajouter le coût lié à la caractérisation d'un cluster ; celle-ci dépend du domaine considéré mais la plupart du temps cette tâche est réalisée lors de l'agrégation d'un élément (en effet, cette information est souvent nécessaire à la vérification de $P$).

Discutons maintenant les conditions sous lesquelles GloBo va découvrir la couverture attendue (il restera ensuite à trouver la bonne caractérisation pour chacun des clusters mais ça n'est pas notre propos ici).

On notera tout d'abord que lorsque le concept à découvrir est conjonctif, GloBo fournira bien une solution conjonctive : quelle que soit la graine, l'algorithme 1 construit un cluster regroupant tous les exemples ; ainsi, ce cluster est le seul candidat fourni à l'algorithme 2.

Intuitivement, pour que les sous-concepts soient découverts (naturellement, cette analyse ne peut-être faite

qu'a posteriori), les conditions suivantes doivent être vérifiées.

- Chacun des sous-concepts à découvrir doit être représenté par au moins (mais de préférence plus) un élément de $E$.
- Dans le but de construire le cluster approprié à partir d'une graine, les premiers candidats à l'agrégation doivent *préserver* le cluster : il faudrait que chacun de ces candidats appartiennent au même sous-concept que la graine, ou bien qu'il soit incompatible avec la graine (ce qui signifie, qu'ensemble, ils ne sont pas admissibles par rapport à $P$). L'idée est que tous les clusters de petite taille satisfont $P$, mais au fur et à mesure qu'un sous-concept apparaît, il devient de plus en plus difficile pour les autres éléments d'entrer dans ce cluster (lorsque l'entrée de ces éléments devient strictement impossible, on dit que le cluster est *verrouillé*).

Ces intuitions sont formalisées par la probabilité de succès du clustering, dont la formule suivante permet de calculer une valeur approchée :

$$\left[1 \Leftrightarrow \left(1 \Leftrightarrow \alpha^{b}\right)^{s}\right]^{n} \qquad (2)$$

où $n$ est le nombre de sous-concepts à apprendre, $s$ est le nombre moyen d'éléments de $E$ dans un sous-concept, $\alpha$ la probabilité pour qu'un élément appartienne au sous-concept de la graine, et $b$ la taille minimale d'un cluster interdisant l'arrivée d'éléments inappropriés (quand $b$ éléments du sous-concept sont dans un cluster, celui-ci est considéré comme verrouillé et aucun élément étranger au sous-concept ne peut plus entrer).

Cette formule indique quand GloBo se comportera bien : quand le nombre de sous-concepts à découvrir sera petit devant le nombre d'éléments disponibles pour représenter chaque sous-concept.

Dans les autres cas, c'est-à-dire lorsque ces conditions ne sont pas satisfaites, on dira que $E$ n'est que *faiblement représentatif* des sous-concepts attendus. Il est alors probable que GloBo ne parvienne pas à les découvrir.

## 3.2 Apprentissage supervisé

Dans le cas de l'apprentissage supervisé, notre algorithme générique est instancié de la manière suivante : $E$ est l'ensemble des exemples positifs et la propriété $P$ correspond à la correction par rapport aux négatifs. Le résultat sera donc une couverture minimale des exemples positifs par des clusters maximalement corrects d'exemples positifs.

$$P(S) \Leftrightarrow \forall e^{-} \in E^{-}, \mathrm{lgg}(S) \not\geq e^{-}$$

où lgg est le moindre généralisé d'un ensemble d'hypothèses [19].

Lors de l'agrégation d'un nouvel exemple, GloBo calcule simplement le moindre généralisé de la caractérisation du cluster courant et du nouvel exemple. Le cluster ainsi construit est admissible par rapport à $P$ si cette généralisation ne couvre aucun exemple négatif (dans le pire des cas, $|E^{-}|$ tests de subsomption sont nécessaires).

La complexité de l'ensemble du processus de clustering est $\Theta((|E^{+}| + |E^{-}|)^2)$ en calculs de lgg, et $\Theta((|E^{+}| + |E^{-}|)^3)$ en tests de subsomption. Dans le cadre attribut-valeur, ces opérations sont linéaires dans la taille des exemples.

Pour bénéficier d'une bonne probabilité de succès, GloBo a en particulier besoin de suffisamment d'exemples négatifs pour séparer et rapidement verrouiller les sous-concepts positifs.

L'étape suivante est de caractériser chaque cluster par une formule conjonctive. Cependant, cette phase d'apprentissage n'est pas le but de cet article : pour les expérimentations, nous avons simplement utilisé le moindre généralisé d'un cluster ! Comme nous le verrons, cette généralisation n'est pas systématiquement trop spécifique et offre souvent de bonnes prédictions.

L'algorithme GloBo peut facilement être adapté pour traiter des données bruitées. GloBo peut appréhender les exemples bruités parmi les négatifs en relâchant la propriété de correction : *la généralisation du cluster ne doit couvrir aucun exemple négatif* est affaibli en *la généralisation du cluster couvre au plus $\eta$ exemples négatifs* où $\eta$ doit être fixé. Le cas des exemples positifs bruités est plus intéressant puisqu'il ne nécessite pas de paramètre comme $\eta$. Pendant la phase de clustering, ces exemples bruités forment de petits clusters (ils ne peuvent pas beaucoup monter en généralité sans couvrir de négatifs), si bien que l'on peut les supprimer au moment de construire la couverture minimale : on arrête la couverture lorsque le meilleur cluster n'a plus qu'un seul élément en commun avec les exemples restant à couvrir (Algorithme 2). Les premières expérimentations sur des données bruitées montrent que GloBo se comporte bien.

## 3.3 Apprentissage non-supervisé

Nous allons voir ici que GloBo peut accomplir un apprentissage non-supervisé en définissant la propriété $P$ en termes de spécificité des clusters. Étant donnés une distance $d$ et $E$ l'ensemble des instances disponibles, $P$ est vérifiée pour un cluster si pour chaque paire d'instances de ce cluster, la distance les séparant ne dépasse pas une valeur limite :

$$P(S) \Leftrightarrow \forall (a,b) \in S, d(a,b) \leq d_{max}$$

Par souci de simplicité, nous ne décrivons ici que l'instanciation de GloBo par cette propriété mais toute

fonction d'évaluation des classes vérifiant la relation de l'Équation 1 peut être utilisée ici.

Pour ajouter un élément à un cluster, nous avons à calculer les distances entre le nouvel élément et toutes les instances présentes dans le cluster courant (dans le pire des cas, $|E| \Leftrightarrow 1$ distances à calculer). Le cluster ainsi formé est admissible par rapport à $P$ si aucune de ces distances ne dépasse la valeur limite. Finalement, la complexité de GloBo pour un apprentissage non-supervisé est $\Theta(|E|^2)$ en termes de calculs de distance.

Pour faciliter la découverte des sous-concepts attendus, GloBo a besoin d'instances en plus grand nombre possible pour chaque sous-concept, et que la distance intra-cluster soit limitée à la valeur minimale autorisant l'apparition des clusters attendus (cela pour verrouiller au plus tôt ces clusters).

On peut alors penser que la difficulté majeure ici est de correctement ajuster cette limite sur la distance intra-cluster, mais nous verrons dans la suite (Section 4) que ce réglage peut être facilement réalisé.

Comme pour l'apprentissage supervisé, nous sommes seulement intéressés dans ce papier dans la construction des clusters, et non pas par leurs caractérisations : on pourra par exemple utiliser la lgg des clusters, ou se servir de GloBo dans sa version supervisée sur chacun des clusters (les autres clusters servant alors d'exemples négatifs).

# 4 Expérimentations

## 4.1 Apprentissage supervisé

Les expérimentations ont été réalisées sur plusieurs problèmes connus, provenant de la base UCI [15]. GloBo a été comparé à `C4.5`, `C4.5rules` [20], et `CN2` [5].

Le domaine principalement décrit ici est la fin de jeu du morpion (*Tic-Tac-Toe Endgame*), qui est la référence en apprentissage disjonctif car les sous-concepts du morpion semblent très difficiles à isoler (aucun système n'est parvenu à 100% de bonnes prédictions en utilisant 75% du dataset pour apprendre).

Nous avons tout d'abord utilisé le test `5x2cv` [8] pour déterminer la confiance avec laquelle on peut affirmer que GloBo est meilleur que les autres systèmes sur cette tâche. Les résultats obtenus sont donnés à la Table 1.

TAB. 1 – *Test* `5x2cv`.

| | statistique $\tilde{t}$ | Confiance |
|---|---|---|
| CN2 | $\Leftrightarrow$2.215 | 93.00% |
| C4.5rules | $\Leftrightarrow$4.420 | 99.40% |
| C4.5 | $\Leftrightarrow$7.836 | $>$ 99.90% |

Ensuite, nous avons voulu tester chaque système en fonction du nombre d'exemples disponibles. En appliquant l'Équation 2 au morpion, la probabilité de succès de GloBo en fonction de la taille du dataset est donné à la Figure 2 (nous avons utilisé pour le morpion les valeurs suivantes : $n = 8$, $\alpha = \frac{1}{2}$, $b = 2$ et $s$ vaut approximativement 8% de la taille du dataset). Sur cette même figure, on donne les prédictions moyennes obtenues sur 10 tirages pour des datasets dont la taille varie de 5 à 95% de la taille du dataset original.

Comme prévu, GloBo parvient à de très bonnes prédictions en n'utilisant que 20% du dataset initial, Dès que les exemples sont en nombre suffisant, GloBo atteint rapidement 100% de bonnes prédictions et est ensuite parfaitement stable. Précisons enfin que la solution systématiquement découverte par GloBo, correspond bien à la disjonction des huit manières de faire une ligne avec des croix au morpion.

Nous avons testé GloBo sur bien d'autres problèmes, en particulier sur des problèmes avec des attributs continus. À chaque fois, GloBo obtient de bons résultats même si, selon le test `5x2cv`, GloBo n'est pas toujours significativement meilleur que les autres systèmes. Cependant, on observe systématiquement que GloBo trouve des solutions plus courtes (en nombre de clusters) que les solutions déjà connues (pour un langage identique bien sûr).

TAB. 2 – *Autres résultats.*

| Problème | Taille | Prédiction |
|---|---|---|
| Mushroom (avec négation) | 3 | 100.0 % |
| Breast Cancer | 4 | 93.0 % |
| Pima Indians Diabetes | 24 | 68.8 % |
| Echocardiogram | 3 | 88.2 % |

La Table 2 donne, pour des domaines connus, les prédictions moyennes et les tailles minimales des solutions découvertes par GloBo. *Pima Indians Diabetes* apparaît comme une tâche terriblement difficile, puisque GloBo ne parvient pas à diviser les 200 exemples positifs disponibles en moins de 24 clusters ! Ce problème peut donc être considéré comme *hautement disjonctif* ; cela explique sans doute pourquoi aucun système ne parvient à de bons résultats sur ce problème. Par contre, GloBo obtient une excellente prédiction connue sur le *Echocardiogram* (précisons que, dans ce cas, tous les attributs sont continus).

Dans un problème *multi-instances* [9], chaque exemple est représenté par plusieurs instances : une solution doit couvrir au moins une instance de chaque exemple positif, et aucune de chaque exemple négatif. GloBo peut traiter le multi-instances sans modification importante : on fait des clusters d'*instances* d'exemples positifs, puis on cherche la couverture des *exemples* positifs. Ainsi, la solution découverte couvrira au moins une instance de chaque exemple positif. GloBo parvient à 80% de bonnes prédictions sur le data set *musk*, ce qui n'est pas le meilleur résultat connu puisque Dietterich annonce 89%. Cela dit, nous estimons qu'il s'agit

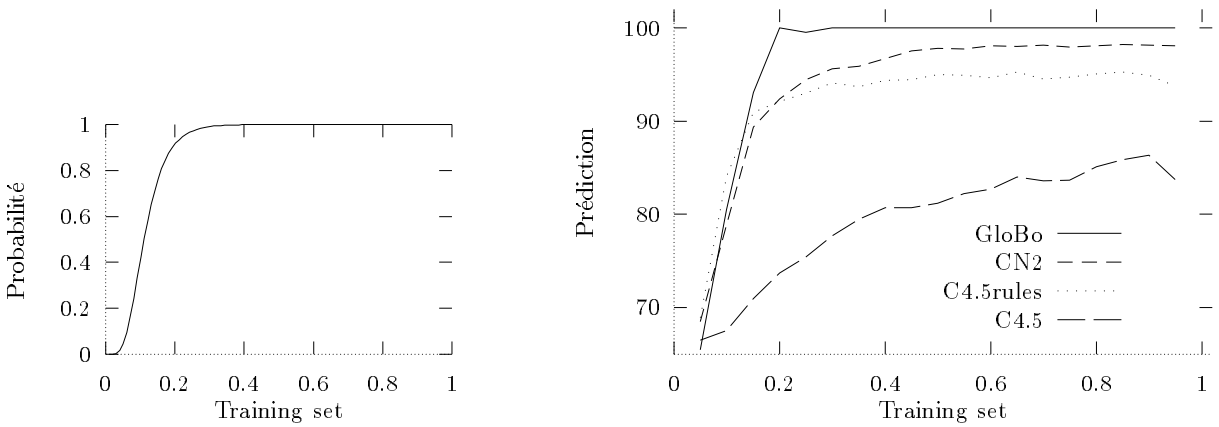



FIG. 2 – *Probabilité de succès et prédictions pour le problème du morpion.*

d'un très bon résultat pour le coup d'essai d'un système qui n'a pas été spécifiquement conçu pour le multi-instances.

Enfin, GloBo a participé au *PTE challenge* [25]. Les résultats [23, 24] indiquent que GloBo arrive en tête des systèmes produisant une théorie compréhensible, et qu'il est deuxième tout type de système confondu. D'autre part, il apparaît que GloBo est le seul des compétiteurs à être FAPP-optimal (*optimal for all practical purposes*).

### 4.2 Apprentissage non-supervisé

Nous avons utilisé ici le problème *Quadruped Animals* décrit dans [13] concernant les mammifères à quatre pattes. Chaque instance appartient à l'une des quatre classes suivantes : chiens, chats, chevaux ou girafes. Chaque animal est décrit par 8 composantes : le cou, quatre jambes, le tronc, la tête, et la queue. Chacun de ces éléments est représenté par un cylindre, lui-même défini par 9 attributs. Pour nos expérimentations, nous avons généré 100 instances en utilisant le générateur mis à disposition sur le site UCI [15]. Enfin, précisons que nous avons simplement utilisé la distance euclidienne.

Le but est de minimiser à la fois le nombre de clusters et la distance intra-cluster. Ces deux optimisations sont incompatibles : pour une distance intra-cluster faible, chaque élément est dans un cluster différent et le nombre de clusters est alors élevé. Inversement, si l'on utilise un seul cluster pour rassembler tous les éléments, la distance intra-cluster est maximale.

Par suite, notre compromis a été d'observer la quantité suivante : le produit du nombre de clusters par la distance intra-cluster *observée*. Le tracé de cette valeur en fonction de distance intra-cluster *autorisée* fait apparaître des plateaux (Figure 3).

Durant ces plateaux, la distance intra-cluster autorisée n'est pas pleinement utilisée : son augmentation ne modifie ni le nombre de clusters ni la distance intra-cluster observée. Intuitivement, on peut penser que pendant ces plateaux, GloBo traverse les espaces inter-clusters. Dans le but de valider cette idée, observons les sous-concepts formés pour des distances limites choisies dans ces plateaux.

Si l'on choisit la distance intra-cluster dans le premier plateau de la courbe (qui correspond au point le plus bas de la courbe), c'est-à-dire entre 23 et 29, quatre clusters sont formés qui correspondent parfaitement aux classes originales. Sur le deuxième plateau (39-85), GloBo trouve trois clusters : les chiens et les chats se sont rassemblés, les chevaux et les girafes forment toujours des clusters disjoints. Sur le troisième plateau de la courbe (126-174), il ne reste que deux clusters : les girafes ont rejoint les chevaux. Enfin, le dernier plateau (à partir de 207) correspond au rassemblement de toutes les instances dans un unique cluster.

Ainsi, en combinant les sous-concepts appris sur les différents plateaux de la courbe, on peut construire une hiérarchie qui utilise les sous-concepts appris sur le plateau le plus bas comme concepts de base.

## 5 Conclusion

Notre approche peut être comparée aux techniques de clustering qui groupent les exemples en fonction de leurs similarités. Notre notion de similarité est cependant plus générale puisqu'elle est encapsulée dans la définition de la propriété $P$. Dans le cas de l'apprentissage supervisé, la *$P$-similarité* entre exemples dépend de l'ensemble des exemples négatifs : selon $E^-$, deux exemples positifs peuvent être très proches et dans le même cluster, ou bien associés à des clusters différents. Il est bien clair qu'une méthode qui, dans le cadre de l'apprentissage supervisé, n'utiliserait qu'une distance purement syntaxique (deux exemples restant à un même

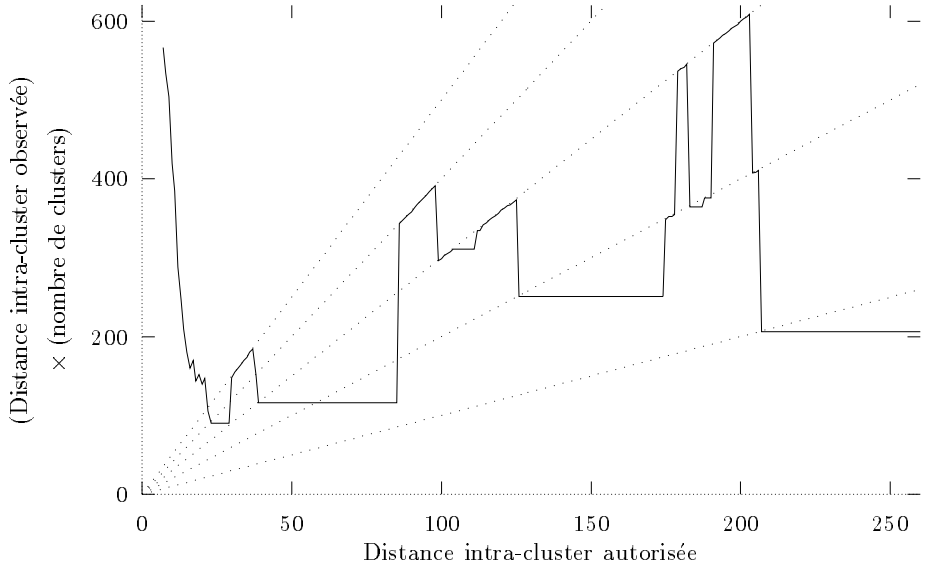



FIG. 3 – *Réglage empirique de la distance limite. Les droites en pointillés correspondent aux droites d'équation* $y = kx$ *pour* $k = 1, 2, 3, 4, 5$.

distance indépendamment des exemples négatifs), ne pourrait pas construire les clusters corrects que nous recherchons.

RISE [10] utilise une telle distance syntaxique mais, à chaque fois qu'un exemple est ajouté au cluster le plus proche, le nouveau cluster doit être validé contre les exemples positifs et négatifs. Nous pensons que cette recherche gloutonne est avantageusement remplacée par le processus de sélection stochastique de GloBo.

Dans le même ordre d'idée, INDIE [1] utilise une méthode heuristique, basée sur une distance entre partitions, pour progressivement s'approcher de la partition correcte idéale des exemples. Ces deux systèmes souffrent de la même limitation : les clusters sont construits par une recherche gloutonne.

[11] est sans le doute le travail le plus proche du nôtre, même s'il se situe dans le domaine de la synthèse de programme. Un graphe est utilisé pour représenter le problème d'apprentissage : les nœuds de ce graphe sont les exemples positifs et une solution disjonctive au problème d'apprentissage est caractérisée par un ensemble de cliques maximales. Néanmoins, la construction du graphe ne prend en compte que des contraintes syntaxiques et pas du tout la couverture des exemples négatifs ; de plus, la recherche de cliques maximales est un problème NP-difficile en soi.

Les travaux sur le *Selective Superiority Problem*, en particulier [21], montrent que pour chaque gain obtenu dans une situation donnée, il existe une autre situation où un gain inverse sera obtenu. Naturellement, ce résultat s'applique également à notre méthode. Cependant, nous avons identifié les situations dans lesquelles les mauvais résultats vont être obtenus : il s'agit simplement, et naturellement, des cas où les données disponibles ne sont pas représentatives du concept à apprendre. Cette notion de représentativité est donnée par l'équation 2 : un concept est représenté par les données disponibles s'il y a des exemples positifs pour chacun des sous-concepts et des exemples négatifs pour délimiter ces sous-concepts.

Cet article propose une vue unifiée de l'apprentissage supervisé et non-supervisé : tous deux peuvent être vus comme une recherche de couverture par des clusters maximalement admissibles par rapport à une propriété donnée (trouver une solution exacte étant connu comme un problème NP-difficile). Nous avons décrit GloBo, un système simple qui construit une solution approchée en temps polynômial. GloBo est basé sur la recherche d'une couverture minimale des exemples mais il est important d'observer que cette recherche est précédée par une construction stochastique des candidats.

Enfin, nous avons testé deux instanciations possibles de GloBo, pour l'apprentissage supervisé et non-supervisé. GloBo a alors obtenu de très bons résultats soit en termes de prédictions sur de nouvelles données, soit sur la formation de concepts usuels. En particulier, GloBo réussit à atteindre 100% de bonnes prédictions sur le morpion en n'utilisant que très peu d'exemples.

Par rapport à l'apprentissage supervisé, notre premier objectif sera de traiter les langages où deux exemples possèdent plusieurs généralisations incomparables. Ce point est crucial pour travailler en ordre un. La $\theta$-subsomption [19] induit un unique moindre généralisé mais pose des problèmes d'efficacité. Cependant, il

existe des relations plus faibles que la $\theta$-subsomption qui autorisent des traitements plus efficaces mais induisent plusieurs généralisations pour deux exemples [22].

En ce qui concerne l'apprentissage non-supervisé, notre but sera aussi de travailler en logique du premier ordre, en utilisant des distances entre clauses [2].

Enfin, nous considérerons toutes les interactions possibles entre nos deux instanciations de GloBo : de l'apprentissage supervisé sur des clusters construits par GloBo non-supervisé, et la reformulation des solutions fournies par GloBo supervisé en utilisant GloBo non-supervisé.

# Références


[1] E. Armengol et E. Plaza. Induction of feature terms with indie. In M. van Someren et G. Widmer, éditeurs, *Proceedings of the ninth European Conference on Machine Learning*, volume 1224 of *Lecture Notes in Artificial Intelligence*, pages 33–48. Springer-Verlag, Avril 1997.

[2] G. Bisson. Learning in FOL with a similarity measure. In W. Swartout, éditeur, *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 82–87. MIT Press, Juillet 1992.

[3] H. Boström. Covering vs. divide-and-conquer for top-down induction of logic programs. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1194–1200, 1995.

[4] R. Caruana, V. de Sa, M. Kearns et A. McCallum, éditeurs. *NIPS*98 Workshop: Integrating Supervised and Unsupervised Learning*, Décembre 1998.

[5] P. Clark et T. Niblett. The cn2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[6] L. De Raedt et M. Bruynooghe. A theory of clausal discovery. In R. Bajcsy, éditeur, *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1058–1063. Morgan Kaufmann, 1993.

[7] E. Diday, J. Lemaire, J. Pouget et F. Testu. *Éléments d'analyse de données*. Dunod, 1982.

[8] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

[9] T. G. Dietterich, R. H. Lathrop et T. Lozano-Perez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[10] P. Domingos. Rule induction and instance-based learning: a unified approach. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1226–1232, 1995.

[11] E. Erdem et P. Flener. A redefinition of least generalizations and its application to inductive logic program synthesis. Technical report, 1997.

[12] D. H. Fisher. Knowledge Acquisition Via Incremental Conceptual Clustering. *Machine Learning*, 2(2):139,172, September 1987.

[13] J. H. Gennari, P. Langley et D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40:11–61, Septembre 1989.

[14] P. Langley, H. A. Simon, G. L. Bradshaw et J. M. Zytkow. *Scientific Discovery: Computational Explorations of the Creative Process*. Machine Intelligence, eds: Meltzer, and Michie, vars. PublishersT Press. ACM CR 8807-0489, 1987.

[15] C.J. Merz et P.M. Murphy. UCI repository of machine learning databases, 1996.

[16] R. S. Michalski. A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell et T. M. Mitchell, éditeurs, *Machine Learning: An Artificial Intelligence Approach*, volume I, pages 83–134, Palo Alto, CA, 1983. Tioga.

[17] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[18] V. T. Paschos. A survey of approximately optimal solutions to some covering and packing problems. *ACM Computing Surveys*, 29(2):171–209, Juin 1997.

[19] G. Plotkin. A note on inductive generalization. In B. Meltzer et D. Mitchie, éditeurs, *Machine Intelligence*, volume 5, pages 153–165. Edinburgh University Press, 1970.

[20] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[21] C. Schaffer. A conservation law for generalization performance. In W. W. Cohen et H. Hirsh, éditeurs, *Proceedings 11th International Conference on Machine Learning*, pages 259–265. Morgan Kaufmann, 1994.

[22] M. Sebag et C. Rouveirol. Tractable induction and classification in FOL via stochastic matching. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pages 888–892, 1997.

[23] A. Srinivasan, R.D. King et D.W. Bristol. An assessment of ILP-assisted models for toxicology and the PTE-3 experiment. In S. Džeroski et P. Flach, éditeurs, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Juin 1999. À paraître.

[24] A. Srinivasan, R.D. King et D.W. Bristol. An assessment of submissions made to the predictive toxicology evaluation challenge. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1999. À paraître.

[25] A. Srinivasan, R.D. King, S.H. Muggleton et M.J.E. Sternberg. The predictive toxicology evaluation challenge. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pages 4–9. Morgan Kaufmann, 1997.

[26] F. Torre et C. Rouveirol. Natural ideal operators in inductive logic programming. In M. van Someren et G. Widmer, éditeurs, *Proceedings of the ninth European Conference on Machine Learning*, volume 1224 of *Lecture Notes in Artificial Intelligence*, pages 274–289. Springer-Verlag, Avril 1997.