Extraction de connaissances provenant de données multisources pour la caractérisation d'arythmies cardiaques

Elisa Fromont, René Quiniou, Marie-Odile Cordier

IRISA, Campus de Beaulieu F-35042 RENNES Cedex France
{efromont, quiniou, cordier}@irisa.fr,
http ://www.irisa.fr/dream

Résumé. Nous nous intéressons au problème de l'apprentissage par programmation logique inductive de règles symboliques caractérisant des arythmies cardiaques à partir de données multisources hétérogènes telles que les différentes voies d'un électrocardiogramme ou la mesure de la pression artérielle. Une première approche consiste à agréger les données dans la base d'apprentissage puis à effectuer un apprentissage directement à partir de ces données transformées. Cette méthode d'apprentissage global est peu performante et difficile à mettre en œuvre quand le volume des données est important. Nous proposons une alternative plus efficace permettant de tirer profit d'apprentissages effectués préalablement sur chacune des sources indépendamment, pour construire automatiquement un biais permettant de restreindre l'espace de recherche lors d'un apprentissage multisource et ainsi traiter ces données complexes. Les résultats obtenus par les deux méthodes sont analysés et comparés. Ils montrent que la méthode proposée permet de gagner un ordre de grandeur sur les temps d'apprentissage.

1 Introduction

Nous nous intéressons à l'extraction de connaissances, par exemple sous forme de règles de classification, à partir de données provenant de plusieurs sources reflétant un même phénomène. Ces données peuvent être soit totalement indépendantes, soit corrélées. Elles peuvent aussi avoir une pertinence ou une fiabilité différente selon la source dont elles proviennent.

On peut alors se demander si une seule source contient suffisamment d'information pour traiter un problème donné ou s'il est préférable de profiter de la complémentarité des différentes sources. Dans le cas monosource, on acquiert de la connaissance à partir de chacune des sources indépendamment, puis on utilise une procédure de décision évaluant les résultats obtenus. Cette procédure de décision basée sur la fiabilité et la pertinence des sources [Cholvy, July 2003], est difficile à mettre en œuvre sans une connaissance précise de chacune des sources. Des méthodes telles que le vote [Dubois *et al.*, 2001], la combinaison de classifieurs [Wemmert et Gancarski, 2002], des règles de "bonne fusion" issues des connaissances sur le domaine [Bloch *et al.*, 2001], etc. peuvent alors être utilisées. Cependant, ces méthodes, même si elles sont effi-

caces en pratique, ne permettent pas d'obtenir de nouvelles connaissances synthétisant véritablement les connaissances acquises indépendamment pour chacune des sources.

Nous nous intéressons donc au cas multisource, permettant de tirer parti conjointement de chacune des sources et nécessitant d'agréger les informations. Les données constituant la base de connaissances sont alors exprimées dans un nouveau langage permettant de décrire tous les éléments des différentes sources. L'extraction de connaissance à partir des données agrégées permet d'utiliser les connaissances sur les corrélations entre les différentes sources et l'utilisation conjointe des données peut augmenter la robustesse des résultats.

Nous nous proposons de résoudre le problème d'extraction de connaissances par programmation logique inductive (PLI). Cette méthode d'apprentissage supervisé a prouvé son efficacité sur des données monosources étudiées dans le cadre du projet Calicot [Carrault *et al.*, 2003]. L'utilisation d'une technique basée sur la logique du premier ordre permet, en outre, d'obtenir des règles facilement interprétables.

L'agrégation des données destinée à constituer la base d'apprentissage nécessaire à la PLI pour un apprentissage multisource augmente considérablement la combinatoire du problème et la complexité des calculs effectués. Il faut donc mettre en œuvre une stratégie permettant de restreindre l'espace de recherche des solutions parcourues par l'algorithme de PLI. Pour ce faire, nous utilisons un biais de langage déclaratif [Nedellec *et al.*, 1996]. Sans connaissance particulière pour constituer ce biais, les résultats obtenus peuvent cependant rester peu fiables compte tenu de la complexité du calcul. Pour résoudre ce problème, nous proposons une méthode d'apprentissage en deux étapes. Nous effectuons dans un premier temps un apprentissage monosource sur chacune des sources puis nous tirons parti de l'association des règles obtenues pour écrire un biais permettant de limiter efficacement l'espace de recherche lors d'un apprentissage sur les données agrégées.

L'article débute par une présentation du contexte applicatif et de la nature des données utilisées, puis donne une analyse des sources de complexité. La section suivante présente le cadre formel de l'apprentissage à partir de données multisources et décrit la méthode en deux étapes que nous proposons. Cette méthode est évaluée, dans la section suivante, en comparant l'apprentissage multisource utilisant un biais classique avec un apprentissage basé sur un biais issu d'apprentissages monosources.

2 Contexte applicatif : la rythmologie

Les données dont nous disposons sont enregistrées par des appareils destinés à surveiller l'état d'un patient : *moniteurs* utilisés en Unité de Soins Intensifs pour Coronariens (USIC) ou *holters* destinés à observer un patient pendant 24 h ou 48 h pour déceler ses éventuels problèmes. Nous présentons dans les paragraphes suivants quelques éléments de cardiologie. Nous terminons par une analyse rapide de la complexité des données retenues.

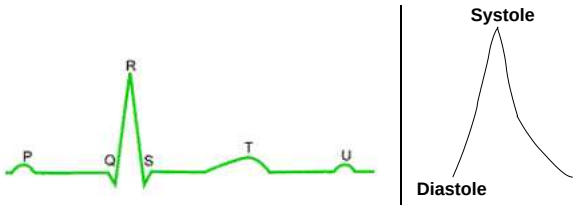

FIG. 1 – Exemple d'un battement cardiaque et d'un cycle de pression normal

FIG. 2 – Exemple de signaux étudiés : voies I et V d'un ECG, voie de pression (ABP)

2.1 Brève introduction à la cardiologie

La contraction du muscle cardiaque a pour origine la propagation d'une onde électrique qui excite les cellules musculaires dans un ordre bien établi afin que la contraction soit la plus efficace possible. L'électrocardiogramme (ECG) reflète l'activité électrique cardiaque. Il permet ainsi de déceler des problèmes de conduction électrique au sein du muscle cardiaque. Ces problèmes sont appelés troubles du rythme ou *arythmies cardiaques*. L'ECG est mesuré en plaçant des électrodes sur le thorax près du cœur (6 voies de V1 à V6) ou sur les bras et les jambes (dérivations bipolaires D1 à D3 et uni-polaires AV). Un ECG *normal* est une succession de battements normaux comme celui de la Figure 1. Deux ondes particulières caractérisent cet ECG : l'onde P et le complexe QRS (toutes deux conséquences d'une propagation électrique dans des parties spécifiques du cœur). Les ondes T et U, beaucoup moins utilisées pour la caractérisation des arythmies, ne sont pas prises en compte dans cette étude. Lorsque l'ECG est anormal (délai entre les ondes ou forme des ondes anormale, absence d'onde, etc.) on peut diagnostiquer une arythmie. L'activité cardiaque se reflète également sur la courbe de pression artérielle. En effet, lorsque les ventricules se contractent pour expulser le sang, la pression augmente fortement : c'est la *systole ventriculaire* (elle se produit peu après l'observation d'un complexe QRS sur l'ECG). La phase de repos (avant l'onde P) est appelée *diastole*.

2.2 Le diagnostic en rythmologie

Les médecins ont pour habitude de diagnostiquer des troubles du rythme cardiaque à partir des seules voies de l'ECG. Pourtant, ils ont souvent à leur disposition des données complémentaires telles que la mesure de la pression artérielle, les phono-cardiogrammes, des mesures de ventilation, d'accélération, etc. Ces informations supplémentaires pourraient être utilisées en USIC pour réduire le nombre de fausses alarmes provenant des appareils programmés pour diagnostiquer des troubles du rythme. D'un point de vue pratique, seules certaines arythmies très graves (considérées comme des *alarmes rouges*) sont diagnostiquées automatiquement, et ce de manière trop sensible dans un souci de sécurité. Le but du travail amorcé dans le projet Calicot [Carrault *et al.*, 2003] est d'améliorer le diagnostic des troubles du rythme cardiaque. En particulier, nous voudrions étendre le diagnostic à d'autres arythmies, non létales si elles sont détectées suffisamment précocement (les alarmes *oranges*), en utilisant les données provenant de plusieurs sources homogènes (plusieurs voies d'un même ECG, par exemple) ou hétérogènes (pression artérielle et voies d'ECG, par exemple).

2.3 Nature des données et sources de complexité

La complexité des données provient essentiellement de deux sources : le temps et le nombre de capteurs utilisés. Les enregistrements sont, en général, longs (de quelques heures à plusieurs jours). Ils génèrent une quantité d'information importante car la fréquence d'échantillonnage doit rester relativement élevée afin que les séries temporelles associées représentent de manière suffisamment détaillée les phénomènes intéressants (le QRS dure de 0,12 s à 0,2 s, par exemple). Les données nécessitent donc une préparation importante avant traitement par fouille de données ou apprentissage : abstraction et focalisation sur les parties où se produisent des phénomènes intéressants.

L'association de plusieurs sources permet d'avoir une vue plus précise et complète de la condition générale d'un patient et de corréler les informations pour améliorer le diagnostic. Pour nos expérimentations, nous utilisons des données provenant des voies 1 et 5 de l'électrocardiogramme ainsi que les données de pression artérielle systolique fournies par la voie hémodynamique (ABP : *Arterial Blood Pressure*). Certaines de ces données utilisées sont très corrélées voire redondantes, puisqu'elles concernent différentes voies d'un même électrocardiogramme (cf. Figure 2). Cependant, les manifestations d'un même phénomène peuvent différer complètement d'une source à l'autre. Ainsi, l'onde du QRS est inversée sur la voie 5 par rapport à la voie 1 ou l'onde P est pratiquement invisible sur certaines voies alors qu'on la distingue aisément sur d'autres. Les corrélations existant entre les informations sur la voie de pression artérielle systolique et les informations sur les voies ECG sont moins évidentes. Elles ont néanmoins des liens forts puisque l'activité électrique enregistrée par l'électrocardiogramme est à l'origine de la contraction et, par voie de conséquence, de la variation de la pression artérielle.

3 Apprentissage multisource : aspects formels

Dans cette section, nous proposons une méthode d'apprentissage permettant de traiter des données provenant de différentes sources d'observation d'un même phénomène. L'idée est d'utiliser les résultats de l'apprentissage partiel à partir des données d'une source pour biaiser un apprentissage global. Nous rappelons tout d'abord les principes de la Programmation Logique Inductive (PLI), puis nous présentons une formalisation de l'apprentissage à partir de données multisources. Enfin, nous présentons l'utilisation de résultats d'apprentissages partiels pour améliorer l'efficacité de l'apprentissage global. Nous supposons une certaine familiarité avec la logique du premier ordre (voir [Lloyd, 1987] pour une introduction).

3.1 Introduction à la PLI

La PLI, encore appelée apprentissage relationnel, est une méthode d'apprentissage supervisé qui construit une hypothèse H expliquant un concept C à partir de la donnée d'un ensemble P d'instances positives et d'un ensemble N d'instances négatives de ce concept. L'apprentissage peut utiliser un ensemble B de règles générales appelées connaissances a priori. Comme dans la formulation de Blockeel et al. [Blockeel *et al.*, 1999], c^+ (resp. c^-) est un prédicat 0-aire (sans argument) satisfait si et seulement si une description exprime le concept (resp. la négation du concept) C. B, P, N et H sont tous des programmes logiques constitués de règles, encore appelées *clauses définies*, de la forme h `:-` b_1`,` b_2`, ...,` b_n. Lorsque $n = 0$ une telle règle est appelée *fait*. Les exemples constituant les ensembles P et N sont des ensembles de faits clos (ils ne contiennent pas de variables).

Définition 1 *Un problème de PLI est un tuple* $< L, P, N, B >$ *tel que :*

- $L = L_H \cup L_E$ *est un langage du premier ordre.* L_H *est appelé* langage des hypothèses *et* L_E langage des exemples.
- $P = \{(e_1, c^+), \ldots, (e_p, c^+)\}$ *est l'ensemble des exemples positifs. Chaque exemple* e_i *est un ensemble de faits exprimés dans le langage* L_E *et* c^+ *est l'étiquette de la classe positive.*
- $N = \{(e_1, c^-), \ldots, (e_n, c^-)\}$ *est l'ensemble des exemples négatifs. Chaque exemple* e_i *est un ensemble de faits exprimés dans le langage* L_E *et* c^- *est l'étiquette de la classe négative.*
- B *est un ensemble de règles exprimées dans le langage* L.

Le but de l'algorithme de PLI est de découvrir $H \subset L_H$ *tel que :*

- H *couvre tout exemple positif :* $\forall (e^+, c^+) \in P, H \wedge e^+ \wedge B \vDash c^+$
- H *ne couvre aucun exemple négatif :* $\forall (e^-, c^-) \in N, H \wedge e^- \wedge B \nvDash c^+$

L'algorithme de PLI recherche l'hypothèse H dans un espace de recherche appelé *espace des hypothèses*. Une relation de généralisation, généralement la θ-subsomption [Plotkin, 1970], est définie sur les hypothèses. Cette relation induit une structure de treillis sur L_H ce qui permet son exploration efficace.

Différentes stratégies peuvent être utilisées pour explorer l'espace des hypothèses. Par exemple, ICL [De Raedt et Van Laer, 1995, Van Laer, 2002], le système de PLI

```
1    1-1:[
2      len-len:[p_wave(P1, 1-1:[normal, abnormal], R0),
3          qrs(R1, 1-1:[normal, abnormal], P1),
4          0-len:[rr1(R0, R1, 1-1:[short, normal, long]),
5            pr1(P1, R1, 1-1:[short, normal, long])]],
6      len-len:[p_wave(P1, 1-1:[normal, abnormal], R0),
7          pp1(P0, P1, 1-1:[short, normal, long])],
8      len-len:[qrs(R1, 1-1:[normal, abnormal], R0),
9          0-1:[rr1(R0, R1, 1-1:[short, normal, long])]]
10         ],
```

FIG. 3 – Spécification syntaxique d'un cycle cardiaque en DLAB

que nous utilisons, et Aleph [Srinivasan, 2003] explorent l'espace de recherche de la clause la plus générale vers les clauses plus spécifiques. Une étape de recherche s'arrête lorsqu'une clause ne couvrant aucun exemple négatif, mais couvrant des exemples positifs, est atteinte. À chaque pas, la meilleure clause (par exemple, celle offrant le meilleur taux de couverture *nb exemples positifs couverts/nb total d'exemples*) est raffinée en ajoutant des littéraux dans son corps, en substituant certaines de ses variables par des constantes, etc. L'espace de recherche défini initialement par L_H peut être restreint par un *biais de langage*. ICL fournit un langage de biais déclaratif nommé DLAB [De Raedt et Dehaspe, 1997] qui permet de définir, de manière syntaxique, les clauses de L_H qui appartiennent à l'espace de recherche. Aleph propose une autre manière de définir le biais au moyen de *bottom clauses* [Muggleton, 1995]. À chaque exemple positif peut être associée une bottom clause qui est plus spécifique que toute hypothèse susceptible de couvrir cet exemple. Par extension, nous appellerons bottom clause toute clause d'un espace de recherche telle qu'il n'existe pas de clause plus spécifique dans l'espace considéré. Pour plus de détails sur la PLI, nous invitons le lecteur à lire l'article [Muggleton et De Raedt, 1994].

Dans la suite, nous appellerons *prédicat événementiel* tout prédicat comme *qrs(R0,normal)* qui décrit un événement se produisant sur une des sources et *prédicat relationnel* tout prédicat comme *suc(R0,R1)* qui dénote une relation particulière (ici l'ordonnancement chronologique) entre deux événements.

3.2 Relations entre DLAB et un ensemble de clauses

Une grammaire DLAB contient des expressions `l-h :[el1,el2,...,eln]` qui signifient : choisir entre `l` et `h` éléments de la liste `[el1,el2,...,eln]`. Le symbole `len` est utilisé pour spécifier la longueur totale de la liste. Par exemple, le terme DLAB `p(2-len :[el1,el2,el3])` génère les expressions `p(el1,el2)`, `p(el1,el3)`, `p(el2,el3)`, `p(el1,el2,el3)`. Les expressions DLAB peuvent être enchâssées comme dans l'expression de la Figure 3 décrivant un cycle cardiaque. La Figure 3 exprime le fait que les hypothèses doivent avoir exactement l'une (signifiée par la contrainte `1-1` ligne 1) des configurations suivantes :

– une onde P nommée P1 suivie par un complexe QRS nommé R1 suivi par des prédicats optionnels (contrainte `0-len`) `pr1` et `rr1` lignes 2-5. Aux ondes P et QRS sont associés des attributs pouvant prendre les valeurs *normal* ou *abnormal* ; le

délai entre deux QRS indiqué par le prédicat `rr1` peut être qualifié par *short*, *normal* ou *long*. Par exemple, l'expression `p_wave(P1, normal, R0), qrs(R1, long, P1), pr1(P1, R1, long)` satisfait cette spécification DLAB,
- une onde P seule associée à un prédicat obligatoire `pp1` (lignes 6 et 7),
- un QRS seul associé à un prédicat optionnel `rr1`(lignes 8 et 9).

Dans la suite nous utiliserons une clause ou un ensemble de clauses pour construire un biais déclaratif DLAB.

Proposition 1 *À un ensemble de (bottom) clauses correspond un biais DLAB qui induit le même espace de recherche que ces (bottom) clauses.*

Clairement, la réciproque n'est pas vraie. Un biais DLAB peut restreindre un espace de recherche de manière beaucoup plus précise qu'une bottom clause. Il contient, en particulier, des contraintes syntaxiques additionnelles spécifiant, par exemple, que certains littéraux sont optionnels, ou donnant des séquences de littéraux particulières. L'espace de recherche correspondant à un biais DLAB est donc inclus dans l'espace de recherche associé à sa clause la plus spécifique (la clause qui contient tous les littéraux qu'il est possible de générer à partir de la spécification DLAB).

3.3 Apprentissage multisource en PLI

L'apprentissage multisource d'un concept C se définit de la manière suivante :

Définition 2 *Soient les problèmes de PLI* $< L_i, P_i, N_i, B_i >$, $i = 1, s$, *tels que* L_i *décrit les données de la source* i. *Un problème multisource en PLI est défini par un tuple* $< L, P, N, B >$ *tel que :*
- $P = F_{agg}(P_1, P_2, \ldots, P_s)$,
- $N = F_{agg}(N_1, N_2, \ldots, N_s)$ *où* F_{agg} *est une fonction d'agrégation dépendant du problème,*
- L *est le langage multisource tel que :*
  $L = L_E \cup L_H$ *avec* $L_E = F_{agg}(L_{E_1}, L_{E_2}, \ldots, L_{E_s})$ *et* $L_H \supseteq \bigcup_{i=1}^{s} L_{H_i}$,
- B *est un ensemble de règles exprimées dans le langage* L.

Le but est de trouver un ensemble de règles $H \subset L_H$ *tel que :*
- $\forall (e^+, c^+) \in P, H \wedge e^+ \wedge B \vDash c^+$
- $\forall (e^-, c^-) \in N, H \wedge e^- \wedge B \nvDash c^+$

3.3.1 Agrégation des exemples

Les exemples sont bidimensionnels. La première dimension, $i \in [1, s]$, fait référence à une source, la seconde, k fait référence à une situation. Les exemples indexés par la même situation correspondent à des vues contemporaines d'un phénomène unique. Nous supposerons dans la suite que l'agrégation d'un ensemble d'exemples est consistante (il est possible de se ramener à cette situation en utilisant des langages L_{E_i} disjoints). L'agrégation est l'opération consistant à fusionner les vues contemporaines d'un phénomène. La fonction d'agrégation F_{agg} dépend du type des données d'apprentissage et peut différer d'un problème d'apprentissage à l'autre. Dans notre cas la fonction d'agrégation est simplement l'union ensembliste :

Définition 3 *Soit* $P_i = \{(e^+_{i,k}, c^+)|k = 1, p\}$, $N_i = \{(e^-_{i,k}, c^-)|k = 1, n\}$, $i = 1, s$. $F_{agg}(P_1, P_2, \ldots, P_s) = P = \{(e^+_k, c^+)|e^+_k = \bigcup_{i=1}^{s} e^+_{i,k}, k = 1, p$ *et* e^+_k *est consistant*$\}$ *et* $F_{agg}(N_1, N_2, \ldots, N_s) = N = \{(e^-_k, c^-)|e^-_k = \bigcup_{i=1}^{s} e^-_{i,k}, k = 1, n$ *et* e^-_k *est consistant*$\}$. *P et N contiennent des faits exprimés dans le langage L_E.*

Il serait possible d'enrichir le langage L_E en y incorporant des relations nouvelles entre événements provenant de sources différentes. Dans ce cas $L_E \supseteq \bigcup_{i=1}^{s} L_{E_i}$. Nous avons décidé de ne pas changer la définition des e_k donc $L_E = \bigcup_{i=1}^{s} L_{E_i}$ et $F_{agg} = \bigcup_{i=1}^{s}$. Toute la connaissance d'agrégation, telle que la spécification de la redondance entre sources, la correspondance entre attributs et les contraintes temporelles sont décrites dans la connaissance a priori B.

Propriété 1 *Soit s le nombre de sources et $F_{agg} = \bigcup_{i=1}^{s}$.*
$L_{i_{ev}}$ dénote la restriction du langage L_i aux prédicats événementiels.

1. *Si H_i couvre l'exemple positif $(e^+_{i,k}, c^+)$ alors H_i couvre aussi un exemple positif agrégé (e^+_k, c^+).*
2. *Si H_i ne couvre pas l'exemple négatif $(e^-_{i,k}, c^-)$ et si $L_{i_{ev}} \cap \bigcup_{j=1, j\neq i}^{s} L_{j_{ev}} = \emptyset$ (les langages L_i n'ont pas de prédicats événementiels en commun) alors, H_i ne couvre pas l'exemple négatif agrégé (e^-_k, c^-).*

Le nouveau langage L est plus riche que chacun des L_i, $i = 1, s$, par conséquent l'espace de recherche associé à L est plus vaste que chacun des espaces de recherche associé aux L_i. Généralement, un e_k agrégé est beaucoup plus complexe que tout exemple $e_{i,k}$, $i = 1, s$, qu'il agrège. Même restreint par un biais de langage, l'espace de recherche d'un processus d'apprentissage à partir d'exemples agrégés a toutes les chances d'être beaucoup plus vaste que chacun des espaces de recherche associés à des apprentissages indépendants à partir de chaque source.

L'approche naïve consiste à apprendre directement à partir des exemples agrégés avec un biais couvrant tout l'espace associé à L. Le principal inconvénient de cette approche est la taille de l'espace de recherche résultant. Dans beaucoup de situations le système d'apprentissage ne pourra le gérer ou nécessitera un temps de calcul trop important. La seule solution consiste donc à spécifier un biais de langage efficace mais cela s'avère être une tâche difficile quand les relations entre les sources ne sont pas explicitées. Dans la suite nous proposons une méthode en deux étapes qui facilite la création d'un tel biais.

3.4 Une méthode en deux étapes pour l'apprentissage multi-source

Nous proposons une méthode d'apprentissage multisource qui consiste à apprendre indépendamment à partir de chaque source, puis d'utiliser les règles apprises pour biaiser une nouvelle étape d'apprentissage sur les données agrégées. L'algorithme 1 décrit la méthode sur l'apprentissage à partir de deux sources, il peut être facilement étendu à l'apprentissage à partir de n sources.

Nous nous intéressons à l'induction de connaissances à partir de l'observation du comportement de systèmes dynamiques. Par conséquent, un exemple, i.e. une situation

sur une source, est une collection d'événements datés. Nous supposons que les situations sont décrites au moyen d'une horloge commune. Ce n'est pas souvent le cas pour des données brutes, nous supposons donc que les données ont été pré-traitées pour assurer cette propriété. De plus, les exemples correspondant à une même situation sont supposés globalement consistants sur les différentes sources. Dans la suite, si les différentes sources possèdent des prédicats relationnels communs, nous faisons l'hypothèse qu'il s'agit au moins d'une relation d'ordre (par exemple, la relation de succession).

Algorithme 1

1. **Apprendre** *avec le biais* $Bias_1$ *sur le problème de PLI* $< L_1, P_1, N_1, B_1 >$. *Soit* H_1 *l'ensemble des règles apprises pour une certaine classe* C.
2. **Apprendre** *avec le biais* $Bias_2$ *sur le problème de PLI* $< L_2, P_2, N_2, B_2 >$. *Soit* H_2 *l'ensemble des règles apprises pour la classe* C.
3. **Agréger** *les ensembles d'exemples* P_1, N_1 *et* P_2, N_2 *ce qui produit* P_3, N_3.
4. *Soit* L_{H_3} *le langage du premier ordre restreint au vocabulaire de* $H_1 = \{h_{11}, h_{12}, \ldots, h_{1l}\}$ *et* $H_2 = \{h_{21}, h_{22}, \ldots, h_{2p}\}$ *et toutes les relations pouvant exister entre les éléments de* H_1 *et les éléments de* H_2 *dans* L.

   Construire *à partir de toute paire* $(h_{1j}, h_{2k}) \in H_1 \times H_2$ *un ensemble de bottom clauses* $BT = \{bt_1, bt_2, \ldots, bt_n\}$ *définies sur* L_{H_3} *tel que* bt_i *contient tous les prédicats événementiels apparaissant dans* h_{1j} *et* h_{2k} *et une combinaison des prédicats relationnels communs à* h_{1j} *et* h_{2k} *qui respecte l'ordonnancement des relations déjà présentes dans* h_{1j} *et* h_{2k}.
5. ***Construire*** $Bias_3$ *à partir de* BT.
6. ***Apprendre*** *avec le biais* $Bias_3$ *sur le problème de PLI* $< L, P_3, N_3, B_3 >$ *où*
   - L *est le langage multisource décrit dans la définition 2,*
   - B_3 *est un ensemble de règles exprimées dans le langage* L.

Dans le cas très simple où il n'existe pas de prédicats relationnels communs entre les événements des deux sources, l'ensemble des règles BT contient une seule bottom clause par paire provenant de H_1 et H_2. L'espace de recherche induit par L_{H_3} est un treillis ordonné par la relation de généralisation basée sur la θ-subsomption. Pour chaque paire considérée, la bottom clause est $bt_i = GSS(h_{1j}, h_{2k})$, le plus grand spécialisé de h_{1j} et h_{2k} sous θ-subsomption (abrégée en GSS) [Nienhuys-Cheng et de Wolf, 1996]. Le GSS est égal à la disjonction des clauses de l'ensemble après standardisation (renommage des variables communes à plusieurs règles). Dans notre cas, le GSS de deux clauses h_{1j} et h_{2k} est une clause ayant pour tête celle commune aux hypothèses de h_{1j} et de h_{2k} et comme corps la conjonction de tous les littéraux apparaissant dans les clauses h_{1j} et h_{2k} après standardisation.

Dans le cas où les règles apprises indépendamment sur les différentes sources comportent des prédicats relationnels communs, pour chaque paire h_{1j} et h_{2k} l'algorithme crée autant de bottom clauses qu'il y a d'ordonnancements maintenant l'ordre relatif des événements sur chacune des sources. Le nombre de bottom clauses ainsi créées est C^n_{n+p} où n est le nombre de prédicats événementiels apparaissant dans la règle h_{1j} et p le nombre de prédicats événementiels apparaissant dans la règle h_{2k}. Le cardinal de

Soit H_1 =`class(normal) :-`
`p(P0,normal), qrs(R0,normal), suc(R0,P0).`
la règle induite pour la classe *normal* sur les données de la source 1.
Soit H_2 = `class(normal) :-`
`dias(D0,normal), sys(S0,normal), suc(S0,D0).`
la règle apprise pour la même classe *normal* sur les données de la source 2. Les bottom clauses générées pour H_1 et H_2 sont :

bt_1 = `class(normal) :-`
`p(P0,normal), dias(D0,normal), suc(D0,P0),`
`qrs(R0,normal), suc(R0,D0),`
`sys(S0,normal), suc(S0,R0).`

bt_2 = `class(normal) :-`
`dias(D0,normal), p(P0,normal), suc(P0,D0),`
`sys(S0,normal), suc(S0,P0),`
`qrs(R0,normal), suc(R0,S0).`

...,

bt_{n-1} = `class(normal) :-`
`p(P0,normal), qrs(R0,normal), suc(R0,P0),`
`dias(D0,normal), suc(D0,R0),`
`sys(S0,normal), suc(S0,D0).`

bt_n = `class(normal) :-`
`dias(D0,normal), sys(S0,normal), suc(S0,D0),`
`p(P0,normal), suc(P0,S0),`
`qrs(R0,normal), suc(R0,P0).`

FIG. 4 – Exemple de génération d'un ensemble de bottom clauses à partir de deux ensembles d'hypothèses réduits à un singleton.

BT est égal au nombre total de bottom clauses créées pour chaque paire h_{1j} et h_{2k}. Ce nombre peut paraître très élevé dans le cas général où les ensembles de règles H_1 et H_2 contiennent plus d'une clause et où le nombre d'évènements caractéristiques d'une classe donnée pour chaque source est important. En pratique, un nombre significatif de bottom clauses ainsi créées peuvent être éliminées car certaines séquences n'ont aucun sens du point de vue de l'application considérée. Sur l'exemple de la Figure 4, un expert du domaine interdirait toutes les séquences comprenant un événement de la source 1 situé entre les événements *dias(A,B)* et *sys(C,D)* de la source 2 car physiologiquement impossibles. Cette contrainte élimine, en particulier, les bottom clauses bt_1 et bt_2 dans l'exemple de la Figure 4.

Le biais est généré automatiquement à partir de l'ensemble des bottom clauses créées (cf. Figure 5 : dans un souci de lisibilité, une seule bottom clause, ici bt_2, a été explicitée). À chaque bottom clause est associée une partie (ou bloc) du biais. Chaque bloc permet d'induire une hypothèse identique ou plus générale que la bottom clause considérée. Un bloc définit le chemin que doit emprunter le système de PLI pour explorer l'espace de recherche, de la racine du treillis de recherche vers la bot-

tom clause considérée. Chaque nœud de l'arbre ainsi exploré correspond à une clause sémantiquement acceptable : les ordonnancements physiquement impossibles n'ont pas été générés et les littéraux utilisés ont un sens du point de vue de la classe considérée puisqu'ils apparaissent dans une règle apprise lors de l'apprentissage monosource. Pour chaque bloc, une toute petite partie de l'espace de recherche induit par LH_3 est couvert, l'exploration est donc très rapide. Notons que deux des blocs générés à partir des clauses bt_{n-1} et bt_n de l'exemple Figure 4 permettent de réapprendre les hypothèses de H_1 et H_2 (cf les chemins en blanc Figure 6). Si aucune autre solution n'est plus satisfaisante que H_1 ou H_2 dans l'espace de recherche, ICL réapprend les règles induites lors des apprentissages monosources indépendants qui couvrent les mêmes exemples que ceux couverts par H_1 et H_2 (cf. propriété 1).

```
1-1:[  %on choisit un et un seul des bloc suivants :
len-len:[...],  %premier bloc : bt1
len-len:[ % bt2
dias(D0,normal),p(P0,normal),suc(P0,D0),
 0-1:[
 len-len:[sys(S0,normal),suc(S0,P0),
   0-1:[ len-len:[qrs(R0,normal),suc(R0,S0)]]
       ]
    ]
],
 ...,
len-len:[...],  %(n-1)ieme bloc : bt(n-1)
len-len:[...]   %nième bloc : btn
]
```

FIG. 5 – Exemple de biais créé à partir d'un ensemble de bottom clauses

La méthode utilisée permet de réduire l'espace de recherche global par rapport à l'approche naïve. De plus, la construction du biais multisource est automatique et repose maintenant sur la construction des biais monosources ce qui s'avère être beaucoup plus aisé que de construire d'emblée un biais global multisource. La construction de ce biais n'assure pas que l'espace de recherche ainsi réduit contiendra la solution optimale du problème multisource mais dans le pire des cas, les règles apprises seront celles des apprentissages monosources.

Dans la suite, nous illustrons l'utilisation de la méthode sur des données cardiaques multisources provenant d'électrocardiogrammes et de mesures de pression artérielle.

4 Résultats

La PLI permet d'obtenir des règles discriminantes caractérisant un ensemble fini d'arythmies cardiaques sous forme d'un programme logique. Le choix de la logique du premier ordre a pour principal avantage de rendre les règles facilement interprétables ce qui s'avère indispensable pour être crédible dans le domaine médical.

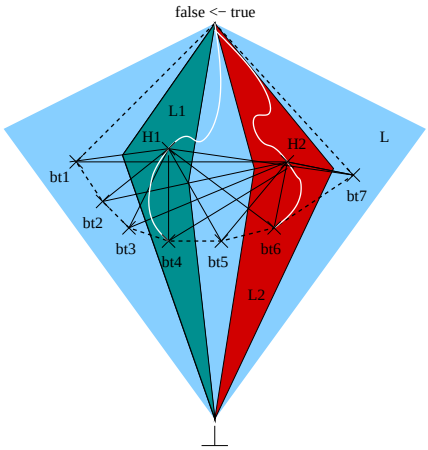

FIG. 6 – Construction de l'espace de recherche pour l'apprentissage biaisé

L'apprentissage monosource produit des règles exploitables indépendamment ou conjointement suivant les contextes : en cas de signaux clairs, une seule voie peut s'avérer suffisante ; en cas de signaux bruités, plusieurs règles peuvent être nécessaires à la reconnaissance, certaines peuvent même s'avérer inexploitables si le bruit est trop important ou si les électrodes sont défaillantes. Par contre, si l'on veut exploiter toutes les informations simultanément, il faut exhiber des relations entre des événements se produisant sur l'électrocardiogramme et des événements se déroulant sur la voie hémodynamique. Pour ce faire, nous effectuons des apprentissages multisources. À des fins d'illustrations, nous nous focalisons sur deux arythmies : la tachycardie supra-ventriculaire (*tsv*) et l'extra-systole ventriculaire (*esv*).

La correction des règles obtenues est estimée par une technique de validation croisée de type "leave-one-out" à cause du faible nombre d'exemples disponibles (environ 6 pour chaque classe). Les résultats sont donnés dans le tableau 1 et sont commentés à la fin de cette section. Les performances indiquées correspondent à des mesures en secondes de temps CPU d'un ordinateur Sun Ultra-Sparc 5.

4.1 Données

Les signaux utilisés proviennent de la base de données MIMIC (Multi-parameter Intelligent Monitoring for Intensive Care [Moody et Mark, 1997]). Cette base contient des données enregistrées sur 72 patients en USIC au Beth Israël Hospital Arrhythmia Laboratory. Les séries temporelles brutes extraites de la base MIMIC sont transformées en descriptions symboliques de signaux par des outils de traitement de signal. Ces descriptions symboliques sont stockées dans des bases de connaissances logiques sous forme de faits (cf. Figures 7 et 8). La Figure 7 donne un exemple de 6 faits apparaissant dans un exemple de doublet ventriculaire. Il s'agit de trois *QRS* intervenant aux temps *5026*, *5638* et *6448*, ainsi que les relations de succession permettant de lier une onde

```
begin(model).
doublet_3_I.
.....
qrs(r7, 5026,normal).
suc(p7,r7).
qrs(r8, 5638,abnormal).
suc(r8,r7).
qrs(r9,6448,abnormal).
suc(r9,r8).
.....
end(model).
```

FIG. 7 – Exemple de données ECG pour un doublet ventriculaire

```
 begin(model).
 rs_3_ABP.
.......
diastole(pd4,3406,-882).
suc(pd4,ps3).
systole(ps4, 3558,-279).
suc(ps4,pd4).
  ......
end(model).
```

FIG. 8 – Exemple de données pression pour un rythme sinusal

avec l'onde qui la précède. Le troisième argument des prédicats décrit la forme de l'onde. La Figure 8 fournit des données similaires pour la pression. Nous disposons pour chaque exemple de l'information sur l'amplitude des ondes étiquetées (ondes P et QRS pour l'ECG, la diastole et la systole pour la voie hémodynamique), sur le délai entre deux ondes, ainsi que sur la forme des ondes. Les données concernent 6 arythmies cardiaques : la tachycardie ventriculaire (considérée comme une alarme rouge en USIC), le doublet ventriculaire, l'extra-systole ventriculaire, la tachycardie supra-ventriculaire, la fibrillation auriculaire (considérées comme des alarmes orange en USIC) et le rythme sinusal ou rythme normal.

4.2 Apprentissage global

4.2.1 Protocole d'expérimentation

Pour chaque enregistrement disponible, les données de pression et les données concernant l'ECG sont agrégées. Les langages restreints aux prédicats événementiels utilisés pour chaque source sont disjoints. Le résultat de l'apprentissage est un ensemble de règles comportant des événements provenant des différentes voies.

Les prédicats choisis pour former les hypothèses doivent contenir, en plus des attributs décrivant chacune des deux sources, des attributs permettant de lier temporellement les éléments des différentes sources. Ainsi une contrainte temporelle indiquant un délai compris entre 0,2 et 0,3 s lie l'instant d'apparition du complexe QRS sur la voie I de l'électrocardiogramme et celui de la *systole* sur la voie hémodynamique. Nous savons également que l'onde P précède le complexe QRS et que la *diastole* commence au début de l'onde P. La pression atteindra donc son point le plus bas (celui utilisé dans les données symboliques) après la fin de l'onde P, mais aussi après l'apparition du complexe QRS. Un cycle cardiaque normal est donc constitué d'une succession d'événements *P-QRS-diastole-systole*. Cet enchaînement sera appelé `cycle_complet_I` puisqu'il est relatif à la voie I. De même, certains prédicats ne sont relatifs qu'à une seule voie comme le `cycle_simple_I` correspondant à une succession P-QRS sur la voie I ou le `cycle_simple_ABP` correspondant à un cycle `diastole-systole` sur la voie hémodynamique. Les prédicats utilisés dans le biais doivent contenir un attribut permettant de maintenir la séquence entre chaque événement. Cet attribut est une variable

```
dlab_template('
false       <--
1-1:[len-len:[
  complete_cycle_I(P0,c1_feat,R0,c1_feat,Dias0,_,Sys0,abp_feat,_,_),
  0-len:[pr1(P0,R0, r_feat), ds1(Dias0,Sys0, r_feat)],
  0-1:[len-len:[
    complete_cycle_I(P1,c1_feat,R1,c1_feat,Dias1,abp_feat,Sys1,abp_feat,R0,Sys0),
    0-len:[pr1(P1,R1,r_feat), rr1(R0,R1,r_feat)],
    0-1:[amp_ss(Sys0, Sys1,1-1:[neg,pos],r_feat)],
    0-1:[amp_dd(Dias0,Dias1, 1-1:[neg,pos],r_feat)],
    0-len:[ds1(Dias1,Sys1,r_feat),ss1(Sys0,Sys1,r_feat),
           dd1(Dias0,Dias1,r_feat),sd1(Sys0,Dias1,r_feat)]
       ],     ... ]').

dlab_variable(r_feat, 1-1, [short, normal, long]).
dlab_variable(c1_feat, 1-1, [normal, abnormal]).
dlab_variable(abp_feat, 1-1, [short, normal, high]).
```

FIG. 9 – Une partie du biais de langage permettant d'induire des règles globales

```
cycle_complet(P,FormP,R,FormQRS,Dias,Symbd,Sys,Symbs,PrecP):-
  p_wave_I(P,TpsP,FormP), suc(P,PrecP),
  qrs_I(R,TpsR,FormQRS), suc(R,P),
  diastole(Dias,TpsDias,Vald),
  qual_amp(Vald,Symbd),
//transforme la valeur numérique en symbolique
  systole(Sys,TpsSys,Val), suc(Sys,Dias),
  qual_amp(Val,Symbs),
//transforme la valeur numérique en symbolique
  const_inter(TpsR,TpsSys,200,300).
//borne l'intervalle entre le QRS et la Systole
...
```

FIG. 10 – Une partie de la connaissance a priori permettant d'induire des règles globales

Prolog. La séquence est maintenue par unification des variables entre elles. Par exemple, dans la première règle caractérisant la classe *esv* dans l'exemple de la Figure 11, le troisième argument du prédicat `qrs/3` est le nom du `QRS` précédent. La variable `R0` s'unifiera avec la variable de même nom en premier argument du `cycle_simple_ABP` pour maintenir le séquencement entre les événements.

Un exemple de biais DLAB est donné Figure 9 (voir le paragraphe 3.2 pour une explication de la syntaxe DLAB). Il indique que le premier prédicat de l'hypothèse générée est soit un `cycle_complet_I`, soit un `cycle_complet_V`, soit un `cycle_simple_ABP`, etc. Un seul choix est possible parmi ces propositions (`1-1`) et l'imbrication des blocs qui suivent, obligent ICL à choisir chacun des prédicats dans l'ordre sans pouvoir "sauter un cycle". Une telle absence de rupture dans la séquence permet d'insérer les attributs temporels liant les données (cf. Figure 10 où le complexe *QRS* est systématiquement relié à la *systole* qui le suit dans un cycle complet).

```
class(esv):-
  cycle_simple_ABP(R0,Dias0,normal,Sys0,normal,_),
  qrs(R1,abnormal,R0),
  cycle_simple_I(P2,normal,R2,normal,R1,Sys2).

class(svt):-
  cycle_simple_I( P0 ,normal, R0 ,normal, _ , Sys0 ),
  cycle_complet_I( P1,normal,R1,normal,Dias1,normal,Sys,normal,R0,Sys0),
  rr1( R0 , R1 ,short), dd1( Dias0 , Dias1 ,short).
```

FIG. 11 – Exemple de règles apprises par apprentissage global sur la voie I de l'ECG et la voie ABP

4.2.2 Résultats

Un exemple de résultat pour les arythmies *Extra systole ventriculaire* (*esv*) et *Tachycardie Supra Ventriculaire* (*tsv*) est donné Figure 11. Ces deux règles sont correctes et complètes, elles couvrent tous les exemples positifs et aucun exemple négatif. La première règle signifie qu'un *QRS* anormal intervenant après un cycle hémodynamique normal et suivi d'un cycle normal sur la voie I discrimine parfaitement une extrasystole ventriculaire. La seconde signifie qu'une *tsv* peut être caractérisée par un cycle complet dont les éléments sont normaux sur la voie I mais dont les différences d'amplitude entre *diastole* et *systole* sont grandes et l'intervalle entre les deux *QRS* est court (le rythme cardiaque est donc rapide).

L'ensemble des résultats est encourageant puisque de telles règles ont pu être apprises pour toutes les arythmies. Elles sont compactes et facilement interprétables. Les temps d'apprentissage sont cependant très longs compte tenu du nombre réduit d'exemples pour chaque classe (7 exemples au maximum).

4.3 Apprentissage biaisé par fusion de règles

La méthode d'apprentissage biaisé par fusion de règles, présentée en section 3, nécessite au préalable un apprentissage indépendant sur chacune des sources.

4.3.1 Apprentissages monosources

Les apprentissages monosources sur la base de données MIMIC (et particulièrement la création des biais) proviennent principalement d'expériences préliminaires effectuées dans le cadre du projet Calicot [Carrault *et al.*, 2003]. La Figure 12 présente un exemple de règles apprises pour les arythmies *esv* et *tsv* sur la voie I de l'électrocardiogramme. La première des règles exprime le fait qu'une *esv* peut être diagnostiquée sur la voie I si deux cycles normaux encadrent un *QRS* anormal. Cette règle a été apprise en 5,6 s de temps CPU. La deuxième règle exprime le fait qu'une *tsv* est caractérisée par deux intervalles *RR* très courts (*rr1 : short*) séparant deux *QRS* normaux, chacun étant précédés d'ondes *P* normales.

Les apprentissages concernant la voie hémodynamique seule ont produit des règles complexes et non totalement discriminantes sur la majorité des apprentissages. Un

```
class(esv):-
 p_wav(P0,normal,_),
 qrs(R0,normal,P0),
 qrs(R1,abnormal,R0),
 p_wav(P2,normal,R1),
 qrs(R2,normal,P2).

class(tsv):-
 qrs(R0,normal,_),
 p_wav(P1,normal,R0),
 qrs(R1,normal,P1),
 rr1(R0,R1,short),
 p_wav(P2,normal,R1),
 qrs(R2,normal,P2),
 rr1(R1,R2,short).
```

FIG. 12 – Exemple de règles apprises sur une voie I de l'ECG

```
class(esv):-
 abp_cycle(Dias0,_,Sys0,high,_),
 abp_cycle(Dias1,normal,Sys1,normal,Sys0),
 amp_ss(Sys0,Sys1,neg,normal),
 ss1(Sys0,Sys1,normal),
 abp_cycle(Dias2,normal,Sys2,normal,Sys1),
 ss1(Sys1,Sys2,normal).

class(esv):-
 abp_cycle(Dias0,_,Sys0,normal,_),
 abp_cycle(Dias1,high,Sys1,high,Sys0),
 amp_ss(Sys0,Sys1,pos,normal),
 ss1(Sys0,Sys1,long).

class(tsv):-
 abp_cycle(Dias0,_,Sys0,normal,_),
 abp_cycle(Dias1,normal,Sys1,normal,Sys0),
 amp_ss(Sys0,Sys1,pos,normal),
 ss1(Sys0,Sys1,short), ds1(Dias1,Sys1,long).
```

FIG. 13 – Exemple de règles apprises sur la voie hémodynamique

exemple d'apprentissage pour les arythmies *esv* et *tsv* est donné Figure 13. Il existe au moins deux façons (deux règles) pour caractériser une *extra systole ventriculaire* sur la voie de pression. La première règle décrit l'*esv* comme une succession de trois cycles de pression avec une différence d'amplitude grande entre la diastole et la systole dans le premier cycle. Cette règle couvre également un exemple de *doublet ventriculaire*. Dans la deuxième règle, l'*esv* est caractérisée par deux cycles de pression avec une différence d'amplitude significative entre la première systole (`Sys0`) et la deuxième diastole (`Dias1`) ainsi qu'entre la deuxième diastole et la deuxième systole (`Sys1`) et un intervalle de temps entre les deux systoles `long` (`ss1(Sys0,Sys1,long)`). Cette règle couvre également un exemple de *tachycardie ventriculaire* ce qui explique la mauvaise précision attribuée à cette règle dans le tableau 1. La *tsv* est caractérisée par 2 cycles de pression normaux avec un intervalle court entre les deux systoles (`ss1(Sys0,Sys1,short)`) et un intervalle de temps court entre la deuxième diastole et la deuxième systole `ds1(Dias1,Sys1,long)`.

4.3.2 Construction du biais par fusion de règles

Nous utilisons les règles obtenues lors de l'apprentissage monosource pour construire le biais permettant de limiter l'espace de recherche lors de l'apprentissage global (cf. section 3.4). Notons que les prédicats apparaissant dans les règles sont liés temporellement. Le nouveau biais constitué doit donc contenir la combinaison de tous les prédicats événementiels présent dans le corps des règles présentées Figures 12 et 13. Un exemple d'un tel biais pour la classe *tsv* est donné Figure 14.

```
dlab_template('
 false       <--
 1-len:[
  1-1:[len-len:[
    qrs(R0, normal, _, Sys0),
    simple_abp_cycle(R0,Dias0, _, Sys0, normal, _)],
  1-1:[len-len:[
    complete_cycle_I(P1, normal,R1,normal,Dias1,normal,Sys1,normal,R0,Sys0),
    1-1:[rr1(R0, R1, short)],
    0-1:[amp_ss(Dias0, Dias1, neg,  normal)],
    0-1:[ss1(Sys0, Sys1, short)],
    0-1:[ds1(Dias0,Sys1,  long)]],
  0-1:[len-len:[
    simple_cycle_I(P2, normal, R2, normal, R1, Sys2),
    0-1:[rr1(R1, R2, short)] ] ] ] ,........]).
```

FIG. 14 – Partie du biais obtenu par fusion de règles

```
class(tsv):-
  qrs( R0, normal, _, Sys0 ),
  cycle_simple_ABP( R0, Dias0, _, Sys0, normal, _ ),
  cycle_complet_I( P1, normal, R1, normal,
  Dias1, normal, Sys1, normal, R0, Sys0 ),
  rr1( R0, R1, short), ss1( Sys0, Sys1, short).

class(esv):-
  cycle_simple_I( P0, normal, R0, normal, _, Sys0 ),
  qrs( R1, abnormal, R0 ),
  cycle_complet_I( P2, normal, R2, normal,
  Dias2, _, Sys2, hight, R1, Sys1 ).
```

FIG. 15 – Exemple de règles induites à partir du nouveau biais obtenu par fusion

4.3.3 Résultats

Les résultats pour la *tsv* et l'*esv* obtenus grâce au biais décrit précédemment sont donnés Figure 15. Les règles produites sont correctes et complètes et calculées dans des temps tout à fait satisfaisants puisque la première règle est induite en 2,88 s CPU et la seconde sur l'*esv* en 18,78 s CPU. Dans notre application les combinaisons entre les prédicats utilisées pour former le biais de langage ont été largement limitées par les connaissances sur l'ordre relatif d'apparition des événements sur les différentes voies.

4.4 Comparaison apprentissage global/apprentissage biaisé par fusion de règles

Le tableau 1 donne les résultats de l'évaluation des performances des deux méthodes. PreApp désigne la précision de l'apprentissage et Prec la précision des tests effectués lors de la validation croisée. VP (vrai positif) dénote le nombre d'exemples positifs correctement classés, FN (faux négatif), le nombre d'exemples négatifs mal classés, FP (faux positif), le nombre d'exemples positifs mal classés et VN (vrai négatif), le nombre

d'exemples négatifs correctement classés. La précision reflète le degré de correction de la règle et est calculée comme suit : $\frac{VP+VN}{VP+FN+FP+VN}$.

	Statistiques		Complexité (moyenne)		Temps de calcul (moyen)
	Prec	PreApp	Règles	Littéraux	CPU (s)
App biaisé (esv)	1	1	1	3	2.88
App global (esv)	1	1	1	3	227.43
App biaisé (tsv)	0.973	1	1	5	18.78
App global (tsv)	0.973	1	1	4	824.28
monosource(ECG I esv)	1	1	1	5	5.60
monosource(ABP esv)	0.676	0.953	2.1	9.8	13.82
monosource(ECG I tsv)	0.973	1	1	6.9	12.00
monosource(ABP tsv)	0.946	1	1	4.9	19.11

TAB. 1 – Résultats de la validation croisée et temps de calcul

Les règles apprises pour la classe *esv* pour les deux stratégies sont complètes et correctes (la précision lors du test et de l'apprentissage est égale à 1). La précision 0.973 pour la classe *tsv* peut être expliquée par un exemple de *doublet* exceptionnel compté comme FP lors des deux tests. Ceci peut s'expliquer car le rythme cardiaque est particulièrement élevé pour cette exemple, il a donc été mal classé. Les règles sont moins complexes pour l'apprentissage global "classique" que pour l'apprentissage biaisé. Le biais utilisé dans le premier cas étant moins précis, la clause la moins complexe a pu être exclue par le biais plus restrictif utilisé pour la seconde méthode. On peut alors s'attendre à ce que les règles apprises lors d'un apprentissage global "classique" soient moins complexes mais calculées en un temps nettement supérieur puisque le nombre d'hypothèses candidates dans l'espace de recherche est supérieur. Les temps de calcul associés à l'apprentissage biaisé sont en effet beaucoup moins importants que ceux de l'apprentissage global "classique". Le temps de calcul cumulé (apprentissage monosource puis apprentissage global biaisé) pour l'apprentissage biaisé est de 22.3 s CPU pour la classe *esv* et de 49.89 s CPU pour la classe *tsv*. Ces temps sont multipliés respectivement par 10 et 16 lors de l'apprentissage global en une seule étape.

On peut noter que l'apprentissage tirant parti des règles apprises indépendamment sur chacune des sources améliore parfois la qualité des règles : par exemple, les règles concernant la voie de pression pour l'*esv* n'étaient pas correctes contrairement à la nouvelle règle apprise. La complexité des nouvelles règles reste raisonnable et la technique permet de ne pas se soucier de la création d'un biais global. En effet, la création d'un biais efficace tout en étant suffisamment large pour apprendre des règles correctes et complètes peut s'avérer difficile lorsque plusieurs sources sont impliquées et qu'il faut prendre en compte des relations entre ces sources. Ici, le biais global est construit à partir d'apprentissages "partiels" pour lesquels le biais est plus facile à concevoir.

4.5 Discussion

Pour obtenir des résultats multisources selon la méthode biaisée présentée dans cette article, les règles monosources apprises indépendamment doivent tout d'abord posséder des prédicats relationnels communs décrivant une relation d'ordre (ici la relation temporelle de succession). Dans le cas contraire, il n'y a pas d'ordonnancement possible entre les événements se produisant sur les différentes sources, et les résultats multisources seront identiques aux résultats monosources. De plus, pour être efficace, la méthode doit s'appuyer sur des règles monosources suffisamment discriminantes pour que l'information qu'elles contiennent soit "centrée" sur les caractéristiques de la classe considérée. En effet, si les règles apprises indépendamment sur chaque source décrivent des événements qui ne se produisent pas dans une même fenêtre temporelle (d'un point de vue de l'horloge commune), il n'y aura pas non plus d'ordonnancements intéressants, ce qui revient au cas précédemment cité.

En outre, les résultats présentés dans la section précédente, ne montrent pas clairement l'avantage de l'utilisation de l'apprentissage multisource par rapport à l'apprentissage monosource (notamment au niveau de la précision des règles) puisque la précision des apprentissages monosources est déjà très élevée (100% ou 93% pour l'ECG). Pour mesurer l'apport du multisource sur les données dont nous disposons, nous envisageons de "bruiter" artificiellement les exemples de chaque classe en conservant l'exemple non bruité. Ceci permettrait d'une part, d'augmenter le nombre d'exemples de la base d'apprentissage, mais aussi, de réduire la précision des apprentissages monosources notamment au niveau de l'électrocardiogramme. Il serait alors possible de mesurer l'apport d'un apprentissage multisource pour réduire l'impact du bruit sur les données (par exemple en apprenant des règles plus compactes, moins sujettes au dysfonctionnement potentiel d'une des source de données, etc).

Enfin, l'algorithme actuel semble très efficace sur un petit volume de données mais des expériences complémentaires doivent être réalisées sur des bases d'apprentissage multisource plus volumineuses pour valider totalement la méthode multisource biaisée.

5 Conclusion

Nous avons formalisé le problème d'apprentissage multisource par programmation logique inductive. La quantité importante de données introduites pour ce type d'apprentissage et la complexité des langages choisis pour décrire les relations pouvant exister entre les différentes sources entraînent des problèmes bien connus de dimensionalité dans le cadre de la PLI. Nous proposons donc une alternative à la méthode "classique" d'apprentissage multisource et nous comparons ces deux méthodes.

La méthode "classique" consiste à agréger toutes les données des différentes sources et à écrire un biais contenant des prédicats permettant de mettre en évidence des relations entre les éléments des différentes sources. Les résultats obtenus sont satisfaisants mais l'écriture du biais est difficile et le temps de calcul de ces règles est très élevé compte tenu du volume des données lorsque l'on fusionne plusieurs sources.

La méthode proposée ici consiste à effectuer un apprentissage indépendant sur chacune des sources puis à tirer parti des règles produites pour construire un biais qui

bornera l'espace de recherche d'un nouvel apprentissage sur les données globales. Les règles apprises avec cette technique sont correctes et complètes et les temps de calcul sont divisés par 10 par rapport à ceux de la méthode globale, ce qui prouve que l'espace de recherche à été efficacement réduit. Cette technique requiert également moins de connaissances préalables sur les données puisqu'elle n'utilise que le langage déjà présent dans les règles induites lors des apprentissages sur les sources indépendantes. Elle permet en outre, de simplifier la conception du biais global puisque celui-ci est créé à partir d'apprentissages "partiels" dont les biais, monosources, sont *a priori* plus simples à concevoir. Cette méthode sera plus amplement validée en testant l'impact des nouveaux résultats pour la reconnaissance et le diagnostic des arythmies dans un contexte hospitalier.

Remerciements

Ce travail est effectué dans le cadre du projet CEPICA avec le soutien du RNTS (Réseau National de Technologies pour la Santé) en collaboration avec le LTSI-Université de Rennes1, ELA-Medical et le département de cardiologie du CHU de Rennes.

Références

[Bloch *et al.*, 2001] I. Bloch, A. Hunter, Alain Appriou, A. Ayoun, Salem Benferhat, P. Besnard, L. Cholvy, R. Cooke, F. Cuppens, D. Dubois, H. Fargier, M. Grabisch, R. Kruse, J. Lang, S. Moral, H. Prade, A. Saffiotti, P. Smets, et C. Sossai. Fusion : General concepts and characteristics. *International Journal of Intelligent Systems*, 16 :1107–1134, 2001.

[Blockeel *et al.*, 1999] H. Blockeel, L. De Raedt, N. Jacobs, et B. Demoen. Scaling up inductive logic programming by learning from interpretations. *Data Mining and Knowledge Discovery*, 3(1) :59–93, 1999.

[Carrault *et al.*, 2003] G. Carrault, M-O. Cordier, R. Quiniou, et F. Wang. Temporal abstraction and inductive logic programming for arrhythmia recognition from ECG. *Artificial Intelligence in Medicine*, 28(231-263), 2003.

[Cholvy, July 2003] L. Cholvy. Information evaluation in fusion : a case study. In *ECSQARU-03 Workshop "Uncertainty, Incompleteness, Imprecision and Conflict in Multiple Data Sources*, Aalborg, Denmark, July 2003.

[De Raedt et Dehaspe, 1997] L. De Raedt et L. Dehaspe. Clausal discovery. *Machine Learning*, 26 :99–146, 1997.

[De Raedt et Van Laer, 1995] L. De Raedt et W. Van Laer. Inductive constraint logic. *Lecture Notes in Computer Science*, 997 :80–94, 1995.

[Dubois *et al.*, 2001] D. Dubois, M. Grabisch, H. Prade, et P. Smets. Using the transferable belief model and a qualitative possibility theory approach on an illustrative example : the assessment of the value of a candidate. *International Journal of Intelligent Systems*, 16 :1183–1192, 2001.

[Lloyd, 1987] J.W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Heidelberg, 1987. Second edition.

[Moody et Mark, 1997] G. B. Moody et Roger G. Mark. A database to support development and evaluation of intelligent intensive care monitoring. Harvard-MIT Division of Health Sciences and Technology, Cambridge, MA, USA, Cardiology Division, Beth Israel Hospital, Boston, MA, USA, 1997. http ://ecg.mit.edu/mimic/mimic.html.

[Muggleton et De Raedt, 1994] Stephen Muggleton et Luc De Raedt. Inductive logic programming : Theory and methods. *The Journal of Logic Programming*, 19 & 20 :629–680, May 1994.

[Muggleton, 1995] S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4) :245–286, 1995.

[Nedellec *et al.*, 1996] C. Nedellec, C. Rouveirol, H. Ade, F. Bergadano, et B. Tausend. Declarative bias in ILP. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 82–103. IOS, 1996.

[Nienhuys-Cheng et de Wolf, 1996] S-H. Nienhuys-Cheng et R. de Wolf. Least generalisations and greatest specializations of sets of clauses. *Journal of Artificial Intelligence Research*, 4 :341–363, May 1996.

[Plotkin, 1970] G.D. Plotkin. A note on inductive generalisation. In B. Meltzer et D. Michie, editors, *Machine Intelligence 5*, pages 153–163. Elsevier North Holland, New York, 1970.

[Srinivasan, 2003] A. Srinivasan. *Aleph Manual V4 and above*, 2003.

[Van Laer, 2002] Wim Van Laer. *From Propositional to First Order Logic in Machine Learning and Data Mining - Induction of first order rules with ICL*. Phd, Department of Computer Science, K.U.Leuven, Leuven, Belgium, June 2002.

[Wemmert et Gancarski, 2002] C. Wemmert et P. Gancarski. A multi-view voting method to combine unsupervised classifications. In *IASTED Artificial Intelligence and Applications*, Malaga, Spain, 2002.

Summary

In many applications dealing with industrial or medical supervision, data are temporal time series related to numerous sensors that provide information which is complementary but also often redundant. We investigate the problem of learning, by inductive logic programming, symbolic rules that characterize cardiac arrhythmias from multisource data such as electrocardiograms or arterial blood pressure measures. A first strategy consists in aggregating the data and then in learning directly from these transformed data. This method is not very efficient and difficult to implement, especially designing the learning bias, when the amount of data is big. We propose an efficient two-step strategy that uses monosource learning to automatically bias and reduce the search space for multisource learning. The results obtained with this method are analysed and compared to those obtained with the naive learning method. We show that an order of magnitude is gained on learning times with the new method.

RNTI - E -