Apprentissage multisource par programmation logique inductive (Relational learning from multisource data)

Élisa Fromont[1] René Quiniou [1] Marie-Odile Cordier [2]

[1] INRIA , [2] Université de Rennes1

Équipe DREAM, IRISA
Campus de Beaulieu, 35042 Rennes Cedex France
{efromont, quiniou, cordier}@irisa.fr

Résumé

Cet article présente l'apprentissage par programmation logique inductive de règles provenant de données multisources. Un apprentissage multisource naïf consisterait à apprendre globalement sur l'ensemble des données et avec un langage d'expression des concepts permettant de couvrir toute la richesse des données représentées. En alternative à un tel type d'apprentissage nous proposons une méthode qui s'appuie sur des apprentissages monosources, ie. effectués sur chacune des sources séparément, pour biaiser l'espace de recherche multisource. Le fait de s'appuyer sur les règles monosources permet, en effet, de restreindre le langage des hypothèses ainsi que le nombre de relations possibles entre les objets représentés sur les différentes sources. Cette méthode est évaluée pour l'apprentissage de règles caractérisant des arythmies cardiaques à partir de plusieurs sources de données telles que les différentes voies d'un électrocardiogramme ou la mesure de la pression artérielle. Les résultats montrent que les règles apprises par apprentissage multisource sont au moins aussi bonnes que les règles monosources dans le cas où les données sont redondantes et meilleures dans les cas où les sources sont complémentaires. La technique d'apprentissage biaisé permet en outre d'apprendre des règles de manière beaucoup plus efficace que dans le cas naïf en bénéficiant d'un biais de langage généré automatiquement.

Mots Clés

Programmation Logique Inductive, Données multisources, Arythmies cardiaques.

Abstract

This paper aims at formalizing the concept of learning rules from multisource using inductive logic programming. In order to cope with the dimensionality problems of multisource learning, we propose a two-step strategy. Firstly, rules are learned independently from each sources. Secondly, the learned rules are used to bias a new learning process from the aggregated data. Our method has been implemented and evaluated on learning from data describing cardiac behaviors from different viewpoints, here electrocardiograms and arterial blood pressure measures. The results show that the proposed method is much more efficient than learning directly from the aggregated data. Furthermore, it yields rules having better or equal accuracy than rules obtained by monosource learning.

Keywords

Inductive Logic Programming, Multisource data, Cardiac arrhythmias.

1 Introduction

Dans de nombreux domaines tels que la météorologie, la bourse ou la médecine, l'utilisation de plusieurs points de vue reflétant un même phénomène est souvent nécessaire pour caractériser le phénomène de façon précise. Lorsque les données proviennent de signaux clairs et lorsque les sources de données sont redondantes, le problème est de sélectionner la source la plus porteuse d'information. Lorsque la redondance entre les données est difficile à établir *a priori* ou lorsque les données provenant des différentes sources sont reconnues comme complémentaires, l'utilisation conjointe de plusieurs sources peut améliorer la robustesse et la précision de cette caractérisation.

Nous nous intéressons au problème de l'apprentissage de règles caractérisant des arythmies cardiaques à partir de données multisources provenant d'un électrocardiogramme (ECG) et de mesures de pression artérielle (ABP). Nous avons choisi de traiter le problème de l'apprentissage par la Programmation Logique Inductive (PLI). En effet cet apprentissage relationnel basé sur la logique du premier ordre est particulièrement adapté à des données faisant intervenir des relations temporelles. En outre, les règles obtenues sont directement interprétables par des cardiologues.

Lors d'un apprentissage multisource, chaque source apporte un langage propre permettant de décrire les données, et le volume de ces données augmente proportionnellement au nombre de sources. La richesse du langage influe sur la

taille de l'espace de recherche des solutions et le volume des données sur les temps de calcul. Le premier problème, en particulier, est bien connu en programmation logique inductive et a engendré de nombreuses solutions permettant de restreindre l'espace de recherche, comme l'écriture de biais déclaratifs [10]. Sans un tel biais, le système de PLI peut produire des solutions aberrantes voire ne produire aucune solution dans le pire des cas, ou des temps de calcul trop élevés dans le meilleur. Cependant, l'écriture d'un tel biais, pour qu'il soit suffisamment restrictif, demande des connaissances très fines sur le domaine d'application et dans le cas multisource, des connaissances liées aux relations entre les éléments des différentes sources. Pour résoudre ce problème d'apprentissage multisource, nous avons proposé une méthode décrite dans [5] qui tire parti d'apprentissages effectués source par source pour biaiser automatiquement l'espace de recherche lors de l'apprentissage sur l'ensemble des données multisources. Cet article approfondit la formalisation de l'apprentissage multisource et donne une caractérisation de l'espace de recherche obtenu avec la méthode présentée. De plus, nous validons cette méthode sur un volume plus important de données et montrons que les apprentissages multisources biaisés ont une correction (ie. nombre d'exemples bien classés sur l'ensemble des exemples) aussi bonne, voire meilleure lorsque les sources sont complémentaires, que les apprentissages monosources tout en étant plus efficaces que les apprentissages multisources naïfs.

Nous rappelons dans la première section de cet article le problème de l'apprentissage multisource en PLI. Dans la deuxième section, nous exposons la méthode que nous avons proposée permettant de réduire l'espace de recherche de l'apprentissage multisource et nous donnons les propriétés de l'espace ainsi réduit. En troisième section, nous présentons les résultats obtenus par cette méthode (appelée apprentissage "biaisé") en comparaison avec des apprentissages monosources et multisources naïfs sur l'apprentissage de sept arythmies cardiaques dans différentes situations.

2 Apprentissage multisource

Dans cette section, nous présentons la définition de l'apprentissage multisource par programmation logique inductive que nous avons proposée pour traiter des données provenant de différentes sources d'observation d'un même phénomène. Nous ne rappelons pas les grands principes de la PLI mais nous invitons le lecteur à consulter [9] pour plus de détails. De même, une introduction à la logique du premier ordre et à la théorie des modèles sur lesquels s'appuie la PLI est donnée dans [6].

Dans la suite, nous appellerons *prédicat événementiel* tout prédicat qui, comme *qrs(R0,normal)*, décrit un événement se produisant sur une des sources (on supposera qu'il y a un et un seul prédicat événementiel par événement) et *prédicat relationnel global* tout prédicat qui, comme *suc(R0,R1)*, dénote une relation particulière (ici l'ordonnancement chronologique) liant des événements se produisant sur plusieurs sources. Par opposition, un *prédicat relationnel local* comme *rr1(R0,R1,normal)* dénote un prédicat relationnel qui ne peut mettre en relation que deux événements provenant de la même source.

Un problème de PLI sur une source i est la donnée d'un langage L_i permettant de décrire le problème sur cette source, d'un ensemble d'exemples E_i, d'une base de connaissances *a priori* sur le domaine B_i décrite sur le langage L_i et d'un ensemble de concepts C que l'on cherche à caractériser. La définition multisource 1 s'inspire de la formulation de Blockeel et al. [1] pour un apprentissage multiclasse par PLI. H_c est un ensemble d'hypothèses décrites sur le langage L_H et représenté sous forme de théorie clausale décrivant une classe $c \in C$ à partir d'un ensemble d'exemples E. Les exemples (e, c) sont des ensembles de faits Prolog clos e étiquetés par la classe c qu'ils représentent et définis sur le langage L_E.

Définition 1 (Apprentissage multisource)
Soient les problèmes de PLI $< L_i, E_i, B_i, C >, i \in [1, s]$, *tels que* L_i *décrit les données de la source* i. *Un problème multisource en PLI est défini par un tuple* $< L, E, B, C >$ *tel que :*

- $E = F_{agg}(E_1, E_2, \ldots, E_s)$ *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 pour toute classe $c \in C$, *un ensemble de règles* $H_c \subset L_H$ *caractérisant la classe* c *et telles que :* $\forall (e, c) \in E, \forall c' \in C - \{c\}$:

- $H_c \wedge e \wedge B \vDash c$ *(*H_c *couvre* (e, c) *noté* $couvre(H_c, (e, c))$)
- $H_c \wedge e \wedge B \nvDash c'$ *(*H_c *est discriminante)*

Chaque source fournit des données permettant de décrire plusieurs situations correspondant à différents exemples. On considère que chaque source décrit exactement les mêmes situations que les autres sources. Un exemple multisource correspond à la vision simultanée d'une situation donnée sur toutes les sources à la fois.

Plus formellement, les exemples $e_{i,k}$ provenant des différentes sources sont bidimensionnels. La première dimension, $i \in [1, s]$, fait référence à une source, la seconde, k, fait référence à une situation. L'agrégation est l'opération consistant à fusionner les vues contemporaines d'une même situation. 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. Les exemples correspondant à une même situation sont supposés globalement consistants sur les différentes sources : si $e_{i,k}$ est étiqueté par la classe c et $e_{j,k}, j \neq i$ est étiqueté par la classe c' alors $c = c'$. Cette

définition s'apparente à celle de Blum et al. [2] pour les fonctions compatibles.

Définition 2 (Exemples agrégés)
Soit s le nombre de sources.
Soit $E_i = \{(e_{i,k}, c) | k \in [1,p]$ et $c \in C\}$. $\forall c \in C, F_{agg}(E_1, E_2, \ldots, E_s) = E = \{(e_k, c) | k \in [1,p], e_k = \bigcup_{i=1}^{s} e_{i,k}$ et e_k est consistant$\}$. E contient 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 $L_E = \bigcup_{i=1}^{s} L_{E_i}$. 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} l'union ensembliste. $L_{i_{ev}}$ dénote la restriction du langage L_i aux prédicats événementiels.*

1. $couvre(H_{i_c}, (e_{i,k}, c)) \Rightarrow couvre(H_{i_c}, (e_k, c))$.
2. *Soit H_i une hypothèse décrivant la classe c. Soit $c' \in C$ telle que $c' \neq c$.*
   $((L_{i_{ev}} \cap \bigcup_{j=1, j\neq i}^{s} L_{j_{ev}} = \emptyset)$
   $\wedge(\neg couvre(H_{i_c}, (e_{i,k}, c'))))$
   $\Rightarrow \neg couvre(H_{i_c}, (e_k, c'))$.

Le premier point de la propriété 1 signifie que si l'hypothèse apprise H_{i_c} couvre l'exemple *monosource* $(e_{i,k}, c)$ alors H_{i_c} couvre l'exemple agrégé (e_k, c). Le second point signifie que si les langages L_i n'ont pas de prédicats événementiels en commun et si l'hypothèse apprise H_{i_c} ne couvre pas l'exemple $(e_{i,k}, c')$ alors H_{i_c} ne couvre pas l'exemple agrégé (e_k, c'). La preuve de cette propriété est la suivante :

Preuve 1

1. *Les exemples provenant des différentes sources sont consistants donc s'il existe $i = 1, s$ tel que $H_{i_c} \wedge e_{i,k} \wedge B \vDash c$ alors $H_{i_c} \wedge [e_{1,k} \wedge e_{2,k}, \wedge \ldots \wedge e_{n,k}] \wedge B \vDash c$ et donc $H_{i_c} \wedge e_k \wedge B \vDash c$.*
2. *On a $(L_{i_{ev}} \cap \bigcup_{j=1, j\neq i}^{s} L_{j_{ev}} = \emptyset) \Rightarrow \forall k, \forall j \neq i, \forall c' \in C - \{c\}, \neg(couvre(H_{i_c}, (e_{j,k}, c')))$.*
   On a $\forall k, \forall c' \in C - \{c\}, \neg(couvre(H_{i_c}, (e_{i,k}, c')))$.
   Donc $\forall k, \forall l, H_{i_c} \wedge e_{l,k} \wedge B \nvDash c'$.
   De plus, les exemples provenant des différentes sources sont consistants donc $H_{i_c} \wedge [e_{1,k} \wedge e_{2,k}, \wedge \ldots \wedge e_{n,k}] \wedge B \nvDash c'$.
   On en conclut donc $H_{i_c} \wedge e_k \wedge B \nvDash c'$.

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 étendu que chacun des espaces de recherche associé aux L_i. L'espace de recherche des hypothèses défini initialement par L_H peut être restreint par un *biais de langage*. ICL [4, 11], le système de PLI que nous utilisons, fournit un langage de biais déclaratif nommé DLAB [3] qui permet de définir, de manière syntaxique, les clauses de L_H qui appartiennent à l'espace de recherche. Un exemple de grammaire DLAB est donné Figure 1.

```
1   1-1:[
2     len-len:[p_wave(P1,1-1:[normal,abnormal]),
3        suc(P1,R0),
4        qrs(R1,1-1:[normal,abnormal]),
5        suc(R1,P1),
6        0-len:[rr1(R0,R1,1-1:[short,normal,long]),
7        pr1(P1,R1,1-1:[short,normal,long])]],
8     len-len:[p_wave(P1,1-1:[normal,abnormal]),
9        suc(P1,R0),
10       pp1(P0,P1,1-1:[short,normal,long])],
11    len-len:[qrs(R1,1-1:[normal,abnormal]),
12        suc(R1,R0),
13        0-1:[rr1(R0,R1,1-1:[short,normal,long])]]
14        ],
```

FIG. 1 – Spécification syntaxique d'un cycle cardiaque en DLAB

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 1 décrivant un cycle cardiaque. Cette expression 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 6-7. 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), qrs(R1, abnormal), pr1(P1, R1, long)` satisfait cette spécification DLAB. L'onde P (`P1`) suit directement un complexe QRS (`R0`),
– une onde P seule associée à un prédicat obligatoire `pp1` (lignes 8-10) qui suit un complexe QRS `R0`,
– un QRS (`R1`) associé à un prédicat optionnel `rr1`(lignes 11-13), qui suit un complexe QRS (`R0`)

Dans la méthode présentée dans la section suivante, nous cherchons à générer automatiquement un biais DLAB à partir d'un ensemble de clauses définissant un espace de recherche et d'un ensemble de contraintes syntaxiques et sémantiques sur les hypothèses recherchées permettant de rendre cet espace fini. Les clauses utilisées étant les plus spécifiques de l'espace de recherche, elles sont appelées *bottom clause* en référence à [8].

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 étendu que chacun des espaces de recherche associés à des apprentissages indépendants à partir de chaque source. L'approche multisource 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. Par exemple, lorsqu'il s'agit de synchroniser les prédicats événementiels apparaissant sur les différentes sources, on cherche à spécifier dans le biais, tous les ordonnancements possibles de ces événements. Ceci se matérialise par des configurations particulières des prédicats événementiels. De telles configurations dans un biais déclaratif augmentent considérablement la combinatoire du problème.

3 Réduction de l'espace de recherche

Nous proposons une méthode d'apprentissage multisource qui consiste à apprendre indépendamment à partir de chaque source, puis à s'appuyer sur les règles apprises pour obtenir des règles multisources en effectuant une nouvelle étape d'apprentissage sur les données agrégées. Dans le cas des arythmies cardiaques, la construction s'apparente à une *synchronisation* des règles monosources. La synchronisation des évènements sur les différentes sources est décrite par le prédicat `suci(X,Y)` qui signifie que l'évènement X est le successeur immédiat de l'évènement Y. 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.

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 classe c donnée.*
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 pour produire P_3, N_3.*
4. ***Construire** à partir de toutes paires $(h_{1j}, h_{2k}) \in H_1 \times H_2$ un ensemble de bottom clauses $BT = \{bt_1, bt_2, \ldots, bt_n\}$ tel que chaque clause bt_i construite à partir de h_{1j} et h_{2k} est plus spécifique que h_{1j} et h_{2k}. En particulier bt_i contient tous les prédicats apparaissant dans h_{1j} et h_{2k} ainsi que de nouveaux prédicats relationnels permettant de synchroniser h_{1j} et h_{2k} tout en respectant l'ordonnancement relatif des événements sur chacune des sources.*
5. ***Construire** $Bias_3$ à partir de BT. Chaque clause de BT correspond à une branche de l'espace de recherche (que nous appellerons "bloc"). La syntaxe des littéraux dans $Bias3$ doit être telle que les hypothèses envisagées dans l'espace de recherche soient identiques ou plus générales que les bottom clauses bt_i et ces hypothèses doivent être ordonnancées dans l'espace de recherche de manière à garantir la séquence des évènements spécifiques à chaque bloc.*
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 1,*
   - *B_3 est un ensemble de règles exprimées dans le langage L.*

Pour chaque paire (h_{1j},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 (cf. l'exemple Figure 2 pour une paire (h_1,h_2)). 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 BT est égal au nombre total de bottom clauses créées pour chaque paire possible (h_{1j},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 2, 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 elles sont physiologiquement impossibles. Cette contrainte élimine, en particulier, les bottom clauses bt_1 et bt_2 dans l'exemple de la Figure 2.

À L'instar des *bottom clauses* utilisées dans des systèmes de PLI tels que PROGOL [8], chacune des *bottom clauses* construites précédemment nous sert à limiter un espace de recherche. L'intersection des espaces de recherche définis par l'ensemble des *bottom clauses* est a priori non vide. Pour chaque espace de recherche défini, on recherche des hypothèses plus générales (ou égales) à la *bottom clause* qui borne cet espace. L'espace des hypothèses est maintenu fini car le nombre de littéraux dans les hypothèses recherchées est limité par le nombre de littéraux présents dans le corps de *la bottom clause*.

Le biais est généré automatiquement à partir de l'ensemble des *bottom clauses* créées et de l'ensemble des contraintes fournies par l'utilisateur (cf. Figure 3 : dans un souci de

Soit h_1 =
```
class(x) :- %séquence P0-R0
        p(P0,normal), qrs(R0,normal),
        pr1(P0,R0,normal), suc(R0,P0).
```
la règle induite pour la classe *x* sur les données de la source 1.
Soit h_2 =
```
class(x) :- %séquence D0-S0
        dias(D0,normal), sys(S0,normal),
        suc(S0,D0).
```
la règle apprise pour la même classe *x* sur les données de la source 2. Les bottom clauses générées pour la paire (h_1,h_2) sont :

bt_1 =
```
class(x) :- %séquence P0-D0-R0-S0
        p(P0,normal), dias(D0,normal),
        suci(D0,P0),
        qrs(R0,normal), pr1(P0,R0,normal),
        suci(R0,D0), suc(R0,P0),
        sys(S0,normal), suci(S0,R0),
        suc(S0,D0).
```
bt_2 =
```
class(x) :- %séquence D0-P0-S0-R0
        dias(D0,normal), p(P0,normal),
        suci(P0,D0),
        sys(S0,normal), suci(S0,P0),
        suc(S0,D0),
        qrs(R0,normal), pr1(P0,R0,normal),
        suci(R0,S0), suc(R0,P0).
```
...,

bt_{n-1} =
```
class(x) :- %séquence D0-S0-P0-R0
        dias(D0,normal), sys(S0,normal),
        suc(S0,D0), p(P0,normal),
        suci(P0,S0), qrs(R0,normal),
        pr1(P0,R0,normal), suc(R0,P0).
```
bt_n =
```
class(x) :- %séquence P0-R0-D0-S0
        p(P0,normal), qrs(R0,normal),
        pr1(P0,R0,normal), suc(R0,P0),
        dias(D0,normal), suci(D0,R0),
        sys(S0,normal), suc(S0,D0).
```

FIG. 2 – Exemple de génération d'un ensemble de bottom clauses à partir d'une paire de clauses (h_1,h_2) décrivant une même classe C.

lisibilité, une seule bottom clause, ici bt_n de la Figure 2, a été explicitée). À chaque bottom clause est associée une partie (ou bloc) du biais correspondant à une partie de l'espace de recherche total considéré. Tous les blocs doivent être évalués mais seul un bloc doit être choisi pour apprendre une hypothèse multisource. Cette contrainte est exprimée par l'expression `1-1 :[...]` qui parenthèse le biais DLAB exposé Figure 3. Chaque nœud de l'arbre ainsi exploré correspond à une clause sémantiquement acceptable : d'une part, les ordonnancements physiquement impossibles ne sont pas générés, et d'autre part, les littéraux utilisés sont sémantiquement acceptables du point de vue de la classe considérée puisqu'ils apparaissent dans une règle apprise lors de l'apprentissage monosource.

Propriété 2 (Correction) *L'espace de recherche engendré par le biais $Bias_3$ de l'algorithme 1 permet*

```
1-1:[
%on choisit un et un seul des bloc suivants :
 ...,
1-1:[len-len:[...]],  %(n-1)ieme bloc :  bt(n-1)

1-1:[len-len:[ % btn
  p(P0,normal),qrs(R0,normal),suc(R0,P0),
  0-1:[pr1(P0,R0,normal)]
   ],
    0-1:[len-len:[
   len-len:[
   dias(D0,normal),suci(D0,R0)
   ],
    0-1:[len-len:[
    len-len:[
    sys(S0,normal),suc(S0,D0)
   ]
]]]]]
]
```

FIG. 3 – Exemple de biais créé à partir d'un ensemble de bottom clauses

d'obtenir des solutions dont la correction est égale ou supérieure à celle obtenue avec H_1 et H_2.

La mesure de correction de l'apprentissage est calculée par la formule $\frac{VP+VN}{VP+FN+FP+VN}$. 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. Intuitivement la propriété 2 exprime le fait que l'espace de recherche défini par le biais $Bias_3$ est construit de telle manière qu'il contient également les ensembles d'hypothèses H_1 et H_2. Grâce à la propriété 1 sur la couverture des exemples multisources, les hypothèses de H_1 et de H_2 peuvent être retenues si aucune hypothèse de l'espace de recherche n'a une meilleure correction.

Propriété 3 (Optimalité) *L'espace de recherche engendré par le biais $Bias_3$ de l'algorithme 1 ne permet pas de garantir l'obtention d'une solution optimale pour l'apprentissage multisource.*

L'espace de recherche limité par le biais construit automatiquement au point 5 de l'algorithme 1 ne contient pas forcément la solution optimale du problème multisource. En effet, considérons la fenêtre temporelle englobant tous les événements participant à une règle monosource, il n'est pas possible d'obtenir une hypothèse multisource avec la méthode proposée s'il n'existe pas de recouvrement entre les fenêtres temporelles concernées par les règles provenant des différentes sources.

Propriété 4 (Réduction de l'espace) *L'espace de recherche engendré par le biais $Bias_3$ de l'algorithme 1 est plus petit que l'espace de recherche multisource naïf.*

La taille de l'espace de recherche de l'apprentissage à partir du biais DLAB peut être quantifiée explicitement à partir de la définition 18 de [3]. Dans notre cas, la grammaire DLAB est réduite à un seul "DLAB template". Intuitivement, le nombre de littéraux autorisés dans un biais déclaratif dépend uniquement du langage utilisé. Ce nombre est donc moins grand pour l'apprentissage biaisé puisque le langage est un sous ensemble du langage utilisé dans le cas de l'apprentissage naïf.

4 Résultats

4.1 Protocole d'expérimentation

Les signaux utilisés proviennent de la base de données MIMIC (Multi-parameter Intelligent Monitoring for Intensive Care [7]). Cette base contient des données multisources 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 pour former cinquante exemples décrivant sept arythmies différentes : le bigéminisme (bige), le doublet ventriculaire (doublet), l'extra-systole ventriculaire (esv), la tachycardie supra-ventriculaire (tsv), des accès de tachycardie ventriculaire (tv), la fibrillation auriculaire (fa) et le rythme sinusal (rs) ou rythme normal qui permet de différencier un trouble de rythme réel d'un rythme non pathologique.

Pour chaque exemple 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 différents types d'apprentissage sont évalués par une technique de validation croisée de type "leave-one-out" à cause du faible nombre d'exemples disponibles (environ 7 pour chacune des 7 classes considérées). Dans la suite, *CorAp* désigne la correction moyenne des règles calculée lors des apprentissages pendant la validation croisée. *CorT* désigne la correction moyenne calculée lors des tests de la validation croisée. Le nombre de cycles cardiaques moyen (*Nbcycles*) donne une indication sur la taille de la fenêtre temporelle concernée par les règles produites. Plus cette fenêtre est petite, plus l'arythmie pourra être détectée tôt et moins le risque de présence de bruit sur les sources est élevé. Ce nombre est calculé sur les règles d'apprentissage apprises sur l'ensemble de la base d'apprentissage. Si plusieurs règles ont été apprises pour une même classe, la colonne *Nbcycles* donne le nombre de cycles correspondant à chacune des règles.

Diverses expériences sont réalisées. Dans une première partie nous utilisons toutes les données dont nous disposons pour comparer les performances de l'apprentissage multisource biaisé avec les apprentissages monosources d'une part et de l'apprentissage multisource biaisé avec l'apprentissage multisource naïf d'autre part. Dans une seconde partie nous réduisons artificiellement les connaissances sur les sources pour évaluer les performances des apprentissages multisources lorsque les apprentissages monosources ont une correction plus faible. Dans une dernière expérience nous mettons en évidence l'intérêt de l'apprentissage multisource lorsque les sources sont réellement complémentaires.

4.2 Comparaison des performances d'apprentissage sur l'ensemble des données

Cette première série de tests a été effectuée en utilisant toutes les connaissances du domaine dont nous disposons.

Le tableau 1 donne une idée de la taille des espaces de recherche explorés lors de chaque apprentissage. Le nombre de nœuds (*Nds*) correspond au nombre de nœuds visités dans l'espace de recherche et les temps de calcul (*Tps*), correspondent à des mesures en secondes de temps CPU effectuées sur un ordinateur SUN Ultra-Sparc 5. Les temps de calcul donnés pour l'apprentissage biaisé prennent en compte les temps des apprentissages monosources (noté $\supset$ *).

On remarque que l'espace de recherche parcouru lors de l'apprentissage multisource biaisé est très inférieur à celui parcouru pour les autres apprentissages ce qui est conforme à la propriété 4 notamment en comparaison avec l'apprentissage naïf.

En revanche, le temps de calcul n'est pas corrélé au nombre de nœuds puisque le test de couverture en chaque nœud de l'espace de recherche est beaucoup plus complexe pour les apprentissages multisources notamment à cause des relations de succession entre les prédicats événementiels provenant de différentes voies (dans le cas d'une seule voie, les relations de succession sont codées directement dans la base d'exemples), qui induisent une combinatoire plus importante. Les temps de calculs de l'apprentissage multisource biaisé sont très inférieurs à ceux de l'apprentissage multisource naïf.

On peut remarquer que les arythmies *doublet, tv, tsv* et *fa* dont les caractéristiques sont très éloignées du rythme normal, sont plus faciles à discriminer que les autres arythmies ce qui se traduit par de meilleures performances lors des apprentissages monosources.

Le tableau 2 donne les résultats de l'évaluation des performances des apprentissages monosources d'une part et des apprentissages multisources effectués avec et sans la méthode de réduction de l'espace de recherche et de génération de biais d'autre part. Les résultats de l'apprentissage monosource sont très bons, notamment sur l'ECG où la mesure de correction sur la base d'apprentissage donne une correction de 1 pour toutes les classes excepté pour le rythme sinusal (*rs*). Cette correction de 1 se retrouve pour l'apprentissage du rythme sinusal sur la source ABP. Ces résultats étant presque parfaits du point de vue de l'apprentissage, il est illusoire d'espérer faire mieux avec l'apprentissage multisource et en particulier avec l'ap-

	monosource : ECG		monosource : ABP		multisource naïf		multisource biaisé	
	Nds	Tps *	Nds	Tps *	Nds	Tps	Nds	Tps ($\supset$ *)
rs	2544	176.64	2679	89.49	18789	3851.36	243	438.55
esv	2616	68.15	5467	68.04	29653	3100.00	657	363.86
bige	1063	26.99	1023	14.27	22735	3299.43	98	92.74
doublet	2100	52.88	4593	64.11	22281	2417.77	1071	290.17
tv	999	26.40	3747	40.01	8442	724.69	30	70.84
tsv	945	29.67	537	17.85	4218	1879.71	20	57.58
fa	896	23.78	972	21.47	2319	550.63	19	63.92

TAB. 1 – Nombre de nœuds visités lors de l'apprentissage et temps de calcul

	monosource ECG			monosource ABP			multisource naïf			multisource biaisé		
	CorAp	CorT	Nbcycl	CorAp	CorT	Nbcycl	CorAp	CorT	Nbcycl	CorAp	CorT	Nbcycl
rs	0.62	0.62	7	1	0.98	5	1	0.96	5	1	0.98	5
esv	0.99	0.96	6	0.98	0.82	3/3/6	0.962	0.88	4	0.99	0.96	6
bige	1	1	4	0.96	0.80	3	0.993	0.82	4/5	1	1	4
doub	1	1	5	0.94	0.80	4/4	1	0.98	3/3	1	1	5
tv	1	1	6	0.98	0.96	4	1	0.98	3	1	1	6
tsv	1	0.98	3	1	0.82	3	1	1	2	1	0.98	3
fa	1	1	2	0.99	0.90	3/4	1	1	2	1	1	2

TAB. 2 – Résultats de la validation croisée pour les apprentissages monosources et multisources

prentissage multisource biaisé. Pour cette expérience, le système de PLI s'est comporté comme un système de vote entre les différentes sources pour retrouver la règle de correction parfaite. Les règles apprises pour l'apprentissage multisource biaisé sont donc égales aux règles apprises sur l'ECG seul pour les cas *esv, doublet, tv, tsv* et *fa* et à celle apprise sur la source ABP pour *rs*.

4.3 Comparaison des performances sur les données réduites

Les sources de données sur lesquelles nous travaillons sont bien connues des cardiologues et nous possédons de nombreuses informations pour, en particulier, constituer les biais d'apprentissage. Ces informations telles que les attributs intéressants pour décrire une source, les prédicats événementiels, les contraintes entre les événements provenant des différentes sources, expliquent en partie les très bons résultats obtenus pour les apprentissages monosources dans l'expérience précédente. Ces résultats sont également expliqués par le nombre réduit d'exemples de la base d'apprentissage et l'absence d'exemples "bruités". L'apport de l'utilisation conjointe des deux sources pour améliorer les performances d'apprentissage peut paraître faible dans ce contexte.

Apprentissage sans onde P, sans forme du QRS et sans diastole. Nous avons décidé de nous placer dans une situation plus réaliste dans laquelle la quantité d'information sur les sources disponible est réduite. En particulier, nous avons décidé de ne pas prendre en compte l'onde P ni la forme du complexe *QRS* pour l'ECG, et la diastole pour la pression. Le fait de ne pas prendre en compte l'onde P a un sens du point de vue du traitement du signal puisque cette onde est encore difficilement détectable automatiquement. La diastole correspond d'un point de vue médical à l'intervalle de temps entre deux systoles (le moment ou les ventricules se vident). Dans la modélisation symbolique que nous avons choisie, la diastole correspond simplement au point de pression le plus bas entre deux systoles. Ce point peut également être difficile à détecter par des algorithmes de traitement du signal.

```
class(tsv):-
qrs(R0), qrs(R1), suc(R1, R0),
qrs(R2), suc(R2, R1),
rr1(R1, R2,short),
rythm(R0, R1, R2,regular),
qrs(R3),suc(R3, R2),
rr1(R2, R3, short), qrs(R4),
suc(R4, R3), rr1(R3, R4, short).

(couvre également 2 fa)
```

FIG. 4 – Exemple de règle apprise pour la classe *tsv* sur l'ECG sans l'onde P ni la forme du QRS

Les résultats de cette expérience sont donnés Table 3. Les règles multisources obtenues avec la méthode naïve sont mixtes pour les 7 arythmies, c'est à dire que les règles possèdent à la fois des éléments de l'ECG et des mesures de pression. Les règles obtenues pour les apprentissages multisources biaisés ne sont mixtes que pour 4 arythmies (*rs, esv,tv et tsv*). Dans les trois autres cas, les règles apprises sont les mêmes que celles apprises sur la source ABP seule. Les mesures de correction pour l'apprentissage

	monosource ECG			monosource ABP			multisource naïf			multisource biaisé		
	CorAp	CorT	Nbcycl	CorAp	CorT	Nbcycl	CorAp	CorT	Nbcycl	CorAp	CorT	Nbcycl
rs	0.4	0.36	6	0.98	0.96	5	1	0.92	5	1	0.92	9
esv	0.44	0.4	4	0.94	0.86	5	0.945	0.64	4/4/6	0.98	0.90	8/5
bige	0.96	0.9	6	0.98	0.98	4	0.98	0.96	4	0.98	0.98	4
doub	0.92	0.92	4	0.96	0.92	4/4	1	0.92	4/4	0.96	0.92	4/4
tv	0.92	0.84	6	0.894	0.66	3	0.977	0.76	6/5	0.94	0.86	5/7
tsv	0.96	0.94	5	0.962	0.84	3	0.76	0.76	2	0.99	0.90	5
fa	0.943	0.88	4/4	0.99	0.96	5/5	0.962	0.82	2/3	0.99	0.96	5/5

TAB. 3 – Résultats de le validation croisée pour les apprentissages monosources et multisources sans information sur l'onde P, la forme du QRS ni sur la diastole

```
class(tsv):-
systole(Sys0), systole(Sys1), suc(Sys1, Sys0),
amp_ss(Sys0, Sys1, normal),
systole(Sys2),suc(Sys2,Sys1),
amp_ss(Sys1, Sys2, normal),
ss1(Sys1, Sys2, short).

(couvre également 1 fa et
ne couvre pas 1 tsv)
```

FIG. 5 – Exemple de règle apprise pour la classe *tsv* sur la pression sans diastole

```
class(tsv):-
qrs(R0,_ ,Sys0),
qrs(R1,R0,Sys1),
rr1(R0,R1,short),
notequal(Sys1,Sys0).

(couvre également
2 esv, 2 bige, 5 tv et 3 fa)
```

FIG. 6 – Exemple de règle apprise pour la *tsv* par apprentissage multisource naïf

sont, comme le montre la théorie, supérieures ou égales aux résultats monosources.

Les règles apprises en multisource biaisé sont, pour cette expérience, meilleures que pour l'apprentissage multisource naïf du point de vue de la correction et du point de vue de l'efficacité, excepté pour la *tv* dont la correction à l'apprentissage est de 0.94 contre 0.977 pour l'apprentissage naïf. Notons que dans plus de la moitié des cas, l'apprentissage multisource biaisé permet également l'acquisition de règles mixtes.

À titre d'exemple, nous donnons Figures 4, 5, 6 et 7 la règle apprise pour la classe *tsv* lors des apprentissages monosources sur l'ECG et la pression d'une part et lors des apprentissages multisources naïfs et biaisés d'autre part. Toutes ces règles, excepté la règle obtenue par apprentissage multisource biaisé, couvrent, en plus des exemples de la classe *tsv*, des exemples d'autres classes (notamment de la classe *fa*). Dans ces règles, `rythm(R0,R1,R2,regular)` signifie que le rythme

```
class(tsv):-
qrs(R0), qrs(R1),suc(R1,R0),
qrs(R2), qual_rr1(R1,R2,short),
rythm(R0,R1,R2,regular),
suc(R2,R1), qrs(R3), qual_rr1(R2,R3,short),
suc(R3,R2), systole(Sys0),
suci(Sys0,R3), qrs(R4),suci(R4,Sys0),
suc(R4,R3), systole(Sys1),
amp_ss(Sys0,Sys1,normal),
suci(Sys1,R4), suc(Sys1,Sys0).
```

FIG. 7 – Exemple de règle apprise pour la *tsv* par apprentissage multisource biaisé

cardiaque est régulier entre les évènements R0, R1 et R2 et `amp_ss(S0,S1,normal)` signifie que la différence d'amplitude entre les deux systoles S0 et S1 est normale.

Apprentissage séparés pour l'onde P et le QRS sans forme. Pour mettre en évidence l'intérêt de l'utilisation de plusieurs sources dans un contexte non bruité, il peut être très intéressant de tester l'apprentissage quand les données provenant des différentes sources sont réellement complémentaires. Nous n'avons pas connaissance de l'existence d'une base de données multisources dont les données sont complémentaires. Nous avons donc choisi d'utiliser les informations de l'ECG sur l'onde P d'une part et sur l'instant d'apparition du complexe QRS d'autre part comme deux sources de données séparées. Une première étude a été conduite et les résultats sont donnés Table 4. Les résultats montrent une réelle amélioration entre les corrections des apprentissages multisources et celles des apprentissages monosources sur des données non bruitées, non seulement au niveau de l'apprentissage mais aussi au niveau du test. On remarque que les apprentissages multisources naïfs couvrant tout l'espace de recherche multisource donnent des résultats comparables à ceux de l'apprentissage multisource biaisé.

4.4 Discussion et perspectives

Pour obtenir des résultats multisources selon la méthode biaisée présentée dans cette article, les règles monosources apprises indépendamment doivent posséder des prédicats relationnels communs décrivant une relation d'ordre (ici la relation temporelle de succession). Dans

	Onde P seule			QRS sans forme			multisource naïf			multisource biaisé		
	CorAp	CorT	Nbcycl	CorAp	CorT	Nbcycl	CorAp	CorT	Nbcycl	CorAp	CorT	Nbcycl
rs	0.62	0.62	7	0.38	0.36	5	0.6	0.6	7	0.7	0.68	11
esv	0.76	0.74	6	0.42	0.4	5	1	0.94	6	1	1	6
bige	1	0.98	4	0.96	0.92	4	1	0.98	4	1	1	4
doub	0.78	0.72	3/7	0.92	0.92	4	1	1	4	1	1	6
tv	0.92	0.9	2	0.901	0.84	5	1	1	4	0.92	0.84	2
tsv	1	1	2	0.96	0.94	5	1	1	2	1	1	2
fa	0	0	0	0.94	0.86	4/4	0.98	0.94	7	0.94	0.92	4

TAB. 4 – Ondes ECG séparées

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 même, si les données provenant des différentes sources sont connues comme étant redondantes *a priori*, la méthode biaisée effectue un apprentissage multisource inutile puisqu'un simple vote sur la source ayant la meilleure mesure de correction lors de l'apprentissage suffirait. En revanche, lorsque les sources sont complémentaires, la méthode biaisée donne de très bons résultats par rapport aux apprentissages monosources et des mesures de correction proches de celles obtenues avec la méthode naïve.

Notons que les règles multisources mixtes sont a priori plus sensibles que les règles monosources à la présence de bruit sur les sources puisqu'elles nécessitent des données claires sur les deux sources à la fois pour bénéficier d'une bonne correction en test. De plus amples expériences devront être réalisées dans un contexte bruité pour évaluer l'intérêt de cette approche.

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 présenté une méthode permettant de mettre en oeuvre l'apprentissage multisource malgré les problèmes de dimensionnalité introduits. L'idée est de s'appuyer sur des apprentissages effectués préalablement sur chacune des sources indépendamment et de générer automatiquement un biais déclaratif réduisant ainsi l'espace de recherche multisource.

Cette méthode diffère des techniques d'apprentissage multisource proposées dans la littérature telles que le *co-training* [2] puisque le but est d'apprendre des règles mixtes permettant de mettre en relation les évènements des différentes sources contrairement au *co-training* qui permet d'apprendre des classifieurs monosources plus performants. Une autre manière d'utiliser plusieurs sources de données consiste à employer des méthodes de votes mais celles ci nécessitent des connaissances sur la fiabilité des sources qu'on ne considère pas disponibles *a priori*.

La méthode "biaisée" d'apprentissage multisource proposée est évaluée dans le cadre de l'apprentissage de règles caractérisant des arythmies cardiaques à partir d'électrocardiogramme et de mesures de pression artérielle. Les résultats obtenus sont comparés à ceux obtenus avec un algorithme naïf d'apprentissage multisource ainsi qu'à ceux obtenus avec un apprentissage monosource sur chacune des deux sources indépendamment.

L'ensemble des expériences montrent que les résultats appris avec la méthode d'apprentissage multisource biaisé sont toujours aussi bons et souvent meilleurs que les apprentissages monosources du point de vue de l'apprentissage et de la reconnaissance. Ces résultats, déjà vérifiés lorsque les règles monosources sont très bonnes, sont encore plus nets lorsque les données monosources et en conséquence les règles monosources apprises sont dégradées ou lorsque les données provenant des différentes sources sont complémentaires.

De plus, les résultats de la méthode biaisée sont comparables à ceux obtenus par apprentissage multisource naïf du point de vue de l'apprentissage et de la reconnaissance. Cependant, la méthode biaisée permet d'obtenir des règles dans des temps de calcul 8 à 35 fois inférieurs aux temps de la méthode naïve. En outre, l'apprentissage multisource biaisé permet également d'apprendre des règles mixtes, tirant parti de la complémentarité des deux sources. Notons que lorsqu'une des sources de données est beaucoup plus performante que la seconde source utilisée, la méthode multisource biaisée se comporte comme un algorithme de vote qui choisit automatiquement et sans connaissance *a priori* la règle la plus intéressante.

Remerciements

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

Références

[1] H. Blockeel, L. De Raedt, N. Jacobs, and B. Demoen. Scaling up inductive logic programming by learning

from interpretations. *Data Mining and Knowledge Discovery*, 3(1) :59–93, 1999.

[2] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT : Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1998.

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

[4] L. De Raedt and W. Van Laer. Inductive constraint logic. In *Proceedings of the 6th Conference on Algorithmic Learning Theory*, LNCS 997, pages 80–94. Springer Verlag, 1995.

[5] E. Fromont, M.-O. Cordier, and R. Quiniou. Extraction de connaissances provenant de données multisources pour la caractérisation d'arythmies cardiaques. *RNTI-E-4, Cepaduès Editions*, (à paraître), 2005.

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

[7] G. B. Moody and Roger G. Mark. A database to support development and evaluation of intelligent intensive care monitoring. *Computers in Cardiology*, 23 :657–660, 1996. http ://ecg.mit.edu/mimic/mimic.html.

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

[9] S. Muggleton and L. De Raedt. Inductive Logic Programming : Theory and methods. *The Journal of Logic Programming*, 19 & 20 :629–680, May 1994.

[10] C. Nédellec, C. Rouveirol, H. Adé, F. Bergadano, and B. Tausend. Declarative bias in ILP. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 82–103. IOS Press, 1996.

[11] W. 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.