

Learning Rules from Multisource Data for Cardiac Monitoring

Élisa Fromont, René Quiniou, and Marie-Odile Cordier

IRISA, Campus de Beaulieu, 35000 Rennes, France
{efromont, quiniou, cordier}@irisa.fr

Abstract. This paper aims at formalizing the concept of learning rules from multisource data in a cardiac monitoring context. Our method has been implemented and evaluated on learning from data describing cardiac behaviors from different viewpoints, here electrocardiograms and arterial blood pressure measures. In order to cope with the dimensionality problems of multisource learning, we propose an Inductive Logic Programming method using 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. The results show that the 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.

1 Introduction

Monitoring devices in Cardiac Intensive Care Units (CICU) use only data from electrocardiogram (ECG) channels to diagnose automatically cardiac arrhythmias. However, data from other sources like arterial pressure, phonocardiograms, ventilation, etc. are often available. This additional information could also be used in order to improve the diagnosis and, consequently, to reduce the number of false alarms emitted by monitoring devices. From a practical point of view, only severe arrhythmias (considered as *red alarms*) are diagnosed automatically, and in a conservative manner to avoid missing a problem. The aim of the work that has begun in the Calicot project [1] is to improve the diagnosis of cardiac rhythm disorders in a monitoring context and to extend the set of recognized arrhythmias to non lethal ones if they are detected early enough (considered as *orange alarms*). To achieve this goal, we want to combine information coming from several sources, such as ECG and arterial blood pressure (ABP) channels.

We are particularly interested in learning temporal rules that could enable such a multisource detection scheme. To learn this kind of rules, a relational learning system that uses Inductive Logic Programming (ILP) is well-adapted. ILP not only enables to learn relations between characteristic events occurring on the different channels but also provides rules that are understandable by doctors since the representation method relies on first order logic.

One possible way to combine information coming from difference sources is simply, to aggregate all the learning data and then, to learn as in the monosource

(i.e one data source) case. However, in a multisource learning problem, the amount of data and the expressiveness of the language, can increase dramatically and with them, the computation time of ILP algorithms and the size of the hypothesis search space. Many methods have been proposed in ILP to cope with the search space dimensions, one of them is using a declarative bias [2]. This bias aims either at narrowing the search space or at ranking hypotheses to consider first the better ones for a given problem. Designing an efficient bias for a multisource problem is a difficult task. In [3], we have sketched a divide-and-conquer strategy (called biased multisource learning) where symbolic rules are learned independently from each source and then, the learned rules are used to bias automatically and efficiently a new learning process on the aggregated dataset. This proposal is developed here and applied on cardiac monitoring data.

In the first section we give a brief introduction to inductive logic programming. In the second section we expose the proposed method. In the third section we describe the experiments we have done to compare, on learning from cardiac data, the monosource, naive multisource and biased multisource methods. The last section gives conclusions and perspectives.

2 Multisource Learning with ILP

In this section, we make a brief introduction to ILP (see [4] for more details) and we give a formalization of this paradigm applied to multisource learning. We assume familiarity with first order logic (see [5] for an introduction).

2.1 Introduction to ILP

Inductive Logic Programming (ILP) is a supervised machine learning method. Given a set of examples E and a set of general rules B representing the background knowledge, it builds a set of hypotheses H , in the form of classification rules for a set of classes C . B and H are logic programs i.e. sets of rules (also called definite clauses) having the form $h:- b_1, b_2, \dots, b_n$. When $n=0$ such a rule is called a fact. E is a labeled set of ground facts. In a multi-class problem, each example labeled by c is a positive example for the class c and a negative example for the class $c' \in \{C - c\}$. The following definition for a multi-class ILP problem is inspired by Blockeel et al. [6].

Definition 1. *A multi-class ILP problem is described by a tuple $\langle L, E, B, C \rangle$ such that:*

- $E = \{(e_k, c) | k = 1, m; c \in C\}$ is the set of examples where each e_k is a set of facts expressed in the language L_E .
- B is a set of rules expressed in the language L . $L = L_E \cup L_H$ where L_H is the languages of hypotheses.

The ILP algorithm has to find a set of rules H such that for each $(e, c) \in E$:

$$H \wedge e \wedge B \models c \text{ and } \forall c' \in C - \{c\}, H \wedge e \wedge B \not\models c'$$

The hypotheses in H are searched in a so-called *hypothesis space*. A generalization relation, usually the θ -subsumption [7], can be defined on hypotheses. This relation induces a lattice structure on L_H which enables an efficient exploration of the search space. Different strategies can be used to explore the hypothesis search space. For example, ICL [8], the ILP system we used, explores the search space from the most general clause to more specific clauses. The search stops when a clause that covers no negative example while covering some positive examples is reached. At each step, the best clause is refined by adding new literals to its body, applying variable substitutions, etc. The search space, initially defined by L_H , can be restricted by a so-called *language bias*. ICL uses a declarative bias (DLAB [9]) which allows to define syntactically the subset of clauses from L_H which belong to the search space. A DLAB bias is a grammar which defines exactly which literals are allowed in hypotheses, in which order literals are added to hypotheses and the search depth limit (the clause size). The most specific clauses of the search space that can be generated from a bias specification are called *bottom clauses*. Conversely, a DLAB bias can be constructed from a set of clauses (the method will not be explained in this article).

2.2 Multisource Learning

In a multisource learning problem, examples are bi-dimensional, the first dimension, $i \in [1, s]$, refers to a source, the second one, $k \in [1, m]$, refers to a situation. Examples indexed by the same situation correspond to contemporaneous views of the same phenomenon. Aggregation is the operation consisting in merging examples from different views of the same situations. The aggregation function F_{agg} depends on the learning data type and can be different from one multisource learning problem to another. Here, the aggregation function is simply the set union associated to inconsistency elimination. Inconsistent aggregated examples are eliminated in the multisource learning problem. The aggregation knowledge, such as correspondence between example attributes on different channels and temporal constraints is entirely described in the background knowledge B . Multisource learning for a multi-class ILP problem is then defined as follows:

Definition 2. Let $\langle L_i, E_i, B_i, C \rangle$, $i = 1, s$, be ILP problems such that L_i describes the data from source i . $E_i = \{(e_{i,k}, c) | k = 1, m; c \in C\}$.

A multisource ILP problem is defined by a tuple $\langle L, E, B, C \rangle$ such that:

- $E = F_{agg}(E_1, E_2, \dots, E_m) = \{(e_k, c) | e_k = \bigcup_{i=1}^s e_{i,k}, k = 1, m\}$
- $L = L_E \cup L_H$ is the multisource language where
 $L_E = F_{agg}(L_{E_1}, L_{E_2}, \dots, L_{E_m})$ and $L_H \supseteq \bigcup_{i=1}^s L_{H_i}$,
- B is a set of rules in the language L .

The ILP algorithm has to find a set of rules H such that for each $(e, c) \in E$:

$$H \wedge e \wedge B \models c \text{ and } \forall c' \in C - \{c\}, H \wedge e \wedge B \not\models c'$$

A *naive* multisource approach consists in learning directly from the aggregated examples and with a global bias that covers the whole search space related

to the aggregated language L . The main drawback of this approach is the size of the resulting search space. In many situations the learning algorithm is not able to cope with it or takes too much computation time. The only solution is to specify an efficient language bias, but this is often a difficult task especially when no information describing the relations between sources is provided. In the following section, we propose a new method to create such a bias.

3 Reducing the Multisource Learning Search Space

We propose a multisource learning method that consists in learning rules independently from each source. The resulting clauses, considered as being bottom clauses, are then merged and used to build a bias that will be used for a new learning process on the aggregated data. Algorithm 1 shows the different steps of the method on two source learning. It can be straightforwardly extended to n source learning. We assume that the situations are described using a common reference time. This is seldom the case for raw data, so we assume that the data set have been preprocessed to ensure this property.

A literal that describes an event occurring on some data source, as $grs(R0,normal)$, is called an *event literal*. Literals of predicates common to the two sources and describing relations between two events, as $suc(R0,R1)$ or $rr1(R0,R1,normal)$, are called *relational literals*.

Algorithm 1

1. **Learn** with bias $Bias_1$ on the ILP problem $\langle L_1, E_1, B_1, C \rangle$. Let H_{c_1} be the set of rules learned for a given class $c \in C$ (rules with head c).
2. **Learn** with bias $Bias_2$ on the ILP problem $\langle L_2, E_2, B_2, C \rangle$. Let H_{c_2} be the set of rules learned for the class c (rules with head c).
3. **Aggregate** the sets of examples E_1 and E_2 giving E_3 .
4. **Generate** from all pairs $(h_{1j}, h_{2k}) \in H_{c_1} \times H_{c_2}$ a set of bottom clauses BT such that each $bt_i \in BT$ built from h_{1j} and h_{2k} is more specific than both h_{1j} and h_{2k} . The literals of bt_i are all the literals of h_{1j} and h_{2k} plus new relational literals that synchronize events in h_{1j} and h_{2k} .
5. For each $c \in C$ **Build** bias $Bias_{c_3}$ from BT . Let $Bias_3 = \{Bias_{c_3} | c \in C\}$.
6. **Learn** with $Bias_3$ on the problem $\langle L, E_3, B_3, C \rangle$ where :
 - L is the multisource language as defined in section 2
 - B_3 is a set of rules expressed in the language L

One goal of multisource learning is to make relationships between events occurring on different sources explicit. For each pair (h_{1j}, h_{2k}) , there are as many bottom clauses as ways to intertwine events from the two sources. A new relational predicate, *suci*, is used to specify this temporal information. The number of bottom clauses generated for one pair (h_{1j}, h_{2k}) is C_{n+p}^n where n is the number of event predicates belonging to h_{1j} and p is the number of event predicates belonging to h_{2k} . The number of clauses in BT is the total number of bottom clauses generated for all possible pairs. This number may be very high if H_{c_1}

and H_{c_2} contain more than one rule and if there are several event predicates in each rule. However, in practice, many bottom clauses can be eliminated because the related event sequences do not make sense for the application. The bias can then be generated automatically from this set of bottom clauses. The multisource search space bounded by this bias has the properties 1, 2 and 3.

Property 1 (Correctness). There exist hypotheses with an equal or higher accuracy than the accuracy of H_{c_1} and H_{c_2} in the search space defined by $Bias_3$ of algorithm 1.

Intuitively, property 1 states that, in the worst case, H_{c_1} and H_{c_2} can also be learned by the biased multisource algorithm. The accuracy¹ is defined as the rate of correctly classified examples.

Property 2 (Optimality). There is no guaranty to find the multisource solution with the best accuracy in the search space defined by $Bias_3$ in algorithm 1

Property 3 (Search space reduction). The search space defined by $Bias_3$ in algorithm 1 is smaller than the naive multisource search space.

The size of the search space specified by a DLAB bias can be computed by the method given in [9]. The biased multisource search space is smaller than the naive search space since the language used in the first case is a subset of the language used in the second case.

In the next section, this biased multisource method is compared to monosource learning from cardiac data coming from an electrocardiogram for the first source and from measures of arterial blood pressure for the second source. The method is also compared to a naive multisource learning performed on the data aggregated from the two former sources.

4 Experimental Results

4.1 Data

We use the MIMIC database (Multi-parameter Intelligent Monitoring for Intensive Care [10]) which contains 72 patients files recorded in the CICU of the Beth Israel Hospital Arrhythmia Laboratory. Raw data concerning the channel V1 of an ECG and an ABP signal are extracted from the MIMIC database and transformed into symbolic descriptions by signal processing tools. These descriptions are stored into a logical knowledge database as Prolog facts (cf. Figures 1 and 2). Figure 1 shows 7 facts in a ventricular doublet example : 1 *P wave*, 3 *QRSs*, the first one occurring at time *5026* as well as relations describing the order of

¹ The accuracy is defined by the formula $\frac{TP+TN}{TP+TN+FP+FN}$ where *TP* (true positive) is the number of positive examples classified as true, *TN* (true negative) the number of negative examples classified as true, *FN* (false negative) is the number of positive examples classified as false and *FP* (false positive), the number of negative examples classified as true.

```

begin(model).
doublet_3_I.
.....
p_wave(p7,4905,normal).
qrs(r7,5026,normal).
suc(r7,p7).
qrs(r8,5638,abnormal).
suc(r8,r7).
qrs(r9,6448,abnormal).
suc(r9,r8).
.....
end(model).

```

Fig. 1. Example representation of a ventricular doublet ECG

```

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

```

Fig. 2. Example representation of a normal rhythm pressure channel

these waves in the sequence. Additional information such as the wave shapes (normal/abnormal) is also provided. Figure 2 provides a similar description for the pressure channel.

Seven cardiac rhythms (corresponding to seven classes) are investigated in this work: normal rhythm (*sr*), ventricular extra-systole (*ves*), bigeminy (*bige*), ventricular doublet (*doub*), ventricular tachycardia (*vt*) which is considered as being a red alarm in CICU, supra-ventricular tachycardia (*svt*) and atrial fibrillation (*af*). On average, 7 examples were built for each of the 7 classes.

4.2 Method

To verify empirically that the biased multisource learning method is efficient, we have performed three kinds of learning experiments on the same learning data: monosource learning from each sources, multisource learning from aggregated data using a global bias and multisource learning using a bias constructed from rules discovered by monosource learning (first experiment). In order to assess the impact of the learning hardness, we have performed two series of experiments:

	monosource: ECG		monosource: ABP		naive multisource		biased multisource	
	Nodes	Time *	Nodes	Time *	Nodes	Time	Nodes	Time (▷ *)
sr	2544	176.64	2679	89.49	18789	3851.36	243	438.55
ves	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
doub	2100	52.88	4593	64.11	22281	2417.77	1071	290.17
vt	999	26.40	3747	40.01	8442	724.69	30	70.84
svt	945	29.67	537	17.85	4218	1879.71	20	57.58
af	896	23.78	972	21.47	2319	550.63	19	63.92
TOT	11163	404.51	19018	315.24	108437	15823.59	2138	1377.66

Table 1. Number of nodes visited for learning and computation times.

	monosource ECG			monosource ABP			naive multisource			biased multisource		
	TrAcc	Acc	Comp	TrAcc	Acc	Comp	TrAcc	Acc	Comp	TrAcc	Acc	Comp
sr	0.84	0.84	9	1	0.98	5	0.98	0.98	3	0.98	0.98	6
ves	1	0.96	5/6	0.963	0.64	3/2/3/2	0.976	0.76	4/3	0.96	0.98	5
bige	1	1	5	0.998	0.84	3/2	0.916	0.7	4/2	1	1	5
doub	1	1	4/5	0.995	0.84	4	0.997	0.8	3/4	0.967	0.9	5
vt	1	1	3	0.981	0.78	3/3/5	0.981	0.94	4	1	1	3
svt	1	1	3	1	0.96	2	1	0.98	2	1	1	2
af	1	1	3	0.981	0.98	2/2	1	1	2	1	1	2

Table 2. Results of cross validation for monosource and multisource learnings

in the first series (4.3) the representation language was expressive enough to give good results for the three kinds of experiments; in the second series (4.4) the expressibility of the representation language was reduced drastically. Three criteria are used to compare the learning results: computational load (CPU time), accuracy and complexity (Comp) of the rules (each number in a cell represents the number of cardiac cycles in each rule produced by the ILP system). As the number of examples is rather low, a *leave-one-out* cross validation method is used to assess the different criteria. The average accuracy measures obtained during cross-validation training (*TrAcc*) and test (*Acc*) are provided.

4.3 Learning from the Whole Database

Table 1 gives an idea of the computational complexity of each learning method (monosource on ECG and ABP channels, naive and biased multisource on aggregated data). *Nodes* is the number of nodes explored in the search space and *Time* is the learning computation time in CPU seconds on a Sun Ultra-Sparc 5.

Table 1 shows that, on average, from 5 to 10 times more nodes are explored during naive multisource learning than during monosource learning and that about 500 to 1000 times less nodes are explored during biased multisource learning than during naive multisource learning. However the computation time does not grow linearly with the number of explored nodes because the covering tests (determining whether an hypothesis is consistent with the examples) are more complex for multisource learning. Biased multisource learning computation times take into account monosource learning computation times and are still very much smaller than for naive multisource learning (8 to 35 times less).

Table 2 gives the average accuracy and complexity of rules obtained during cross validation for the monosource and the two multisource learning methods. The accuracy of monosource rules is very good for ECG and a bit less for ABP, particularly the test accuracy. The naive multisource rules have also good results. Furthermore, for the seven arrhythmias, these rules combine events and relations occurring on both sources. Only *sr*, *ves*, *svt* and *af* got combined biased multisource rules. In the three other cases, the learned rules are the same as the ECG rules. The rules learned for *svt* in the four learning settings are given

```

class(svt):-
qrs(R0,normal),
p(P1,normal), qrs(R1,normal),
suc(P1,R0), suc(R1,P1),
rr1(R0,R1,short),
p(P2,normal), qrs(R2,normal),
suc(P2,R1), suc(R2,P2),
rythm(R0,R1,R2,regular).

```

Fig. 3. Example of rule learned for class *svt* from ECG data

```

class(svt):-
cycle_abp(Dias0,_,Sys0,normal),
cycle_abp(Dias1,normal,Sys1,normal),
suc(Dias1,Sys0),
amp_dd(Dias0,Dias1,normal),
ss1(Sys0,Sys1,short),
ds1(Dias1,Sys1,long).

```

Fig. 4. Example of rule learned for class *svt* from ABP data

```

class(svt):-
qrs(R0,normal),
cycle_abp(Dias0,_,Sys0,normal),
p(P1,normal),qrs(R1,normal),
suci(P1,Sys0),suc(R1,P1),
systole(Sys1),
rr1(R0,R1,short).

```

Fig. 5. Example of rule learned for class *svt* by naive multisource learning

```

class(svt):-
qrs(R0,normal),
p(P1,normal),qrs(R1,normal),
suc(P1,R0),suc(R1,P1),
rr1(R0,R1,short),
cycle_abp(Dias0,_,Sys0,normal),
suci(Dias0,R1).

```

Fig. 6. Example of rule learned for class *svt* by biased multisource learning

in Figures 3, 4, 5 and 6. All those rules are perfectly accurate. The predicate *cycle_abp(D, ampsd, S, ampsd)* is a kind of macro predicate that expresses the succession of a diastole named *D*, and a systole named *S*. *ampsd* (resp. *ampds*) expresses the symbolic pressure variation ($\in \{short, normal, long\}$) between a systole and the following diastole *D* (resp. between the diastole *D* and the following systole *S*). The biased multisource rule and the naive multisource rule are very similar but specify different event orders (in the first one the diastole-systole specification occurs before two close-in-time QRS whereas in the second one, the same specification occurs after two close-in-time QRS).

As expected from the theory, the biased multisource rules accuracy is better than or equal to the monosource rules accuracy except for the *doublet*. In this case, the difference between the two accuracy measures comes from a drawback of the cross-validation to evaluate the biased multisource learning rules with respect to the monosource rules. At each cross-validation step, one example is extracted for test from the example database and learning is performed on the remaining database. Sometimes the learned rules differ from one step to another. Since this variation is very small we have chosen to keep the same multisource bias for all the cross-validation steps even if the monosource rules upon which it should be constructed may vary. According to this choice, the small variation between the biased multisource rules accuracy and the monosource rules accuracy is not significant. Table 2 also shows that when the monosource results are good, the biased multisource rules have a better accuracy than the naive multisource rules and rules combining events from different sources can also be learned.

4.4 Learning from a Less Informative Database

The current medical data we are working on are very well known from the cardiologists, so, we have a lot of background information on them. For example, we know which event or which kind of relations between events are interesting for the learning process, which kind of constraints exists between events occurring on the different sources etc. This knowledge is very useful to create the learning bias and can explain partly the very good accuracy results obtained in the learning experiments above. These good results can also be explained by the small number of examples available for each arrhythmia and the fact that our examples are not corrupted. In this context, it is very difficult to evaluate the usefulness of using two data sources to improve the learning performances.

We have thus decided to set ourselves in a more realistic situation where information about the sources is reduced. In this experiment we do not take into account the P waves nor the shape of the QRS on the ECG and the diastole on the ABP channel. This experiment makes sense as far as signal processing is concerned since it is still difficult with current signal processing algorithms to detect a P wave on the ECG. Besides, in our symbolic description of the ABP channel, the diastole is simply the lowest point between two systoles. This specific point is also difficult to detect. Note that cardiologists view the diastole as the time period between two systoles (the moment during which the chambers fill with blood).

The results of this second experiment are given in Table 3. This time again, the biased multisource rules have as good or better results than the monosource rules. For arrhythmias *sr*, *bige* and *af* the biased method acts like a voting method and learns the same rules with the same accuracy as the best monosource rules (small variations in accuracies come from the cross validation drawback). For arrhythmias *ves*, *doublet*, *vt* and *svt* the biased multisource rules are different from both monosource rules corresponding to the same arrhythmia and accuracy and test are better than for the monosource rules.

	monosource ECG			monosource ABP			naive multisource			biased multisource		
	TrAcc	Acc	Comp	TrAcc	Acc	Comp	TrAcc	Acc	Comp	TrAcc	Acc	Comp
sr	0.38	0.36	5	1	0.96	5/4	1	0.92	5	1	0.98	5/4
ves	0.42	0.4	5	0.938	0.76	4/3	0.945	0.64	4/4/6	0.94	0.9	4/3
bige	0.96	0.92	4	1	0.98	4/4	0.98	0.96	4	1	0.98	4/4
doub	0.881	0.78	4/4	0.973	0.86	4/4/5	1	0.92	4/4	0.941	0.9	3/4/5
vt	0.919	0.84	5/5	0.943	0.84	3/4/5	0.977	0.76	6/5	0.96	0.86	3/5/5
svt	0.96	0.94	5	0.962	0.86	4	0.76	0.76	2	0.96	0.94	4
af	0.945	0.86	4/4	1	0.9	3/4/5	0.962	0.82	2/3	1	0.98	3/4/5

Table 3. Results of cross-validation for monosource and multisource learnings without knowledge on P wave nor QRS shape nor diastole

5 Conclusion

We have presented a technique to learn rules from multisource data with an inductive logic programming method in order to improve the detection and the recognition of cardiac arrhythmias in a monitoring context. To reduce the computation time of a straightforward multisource learning from aggregated examples, we propose a method to design an efficient bias for multisource learning. This bias is constructed from the results obtained by learning independently from data associated to the different sources. We have shown that this technique provides rules which have always better or equal accuracy results than monosource rules and that it is much more efficient than a naive multisource learning.

In future work, the method will be tested on corrupted data. Besides, since this article only focuses on accuracy and performance results, the impact of the multisource rules on the recognition and the diagnosis of cardiac arrhythmias in a clinical context will be more deeply evaluated by experts.

Acknowledgments

Elisa Fromont is supported by the French National Net for Health Technologies as a member of the CEPICA project. This project is in collaboration with LTSI-Université de Rennes1, ELA-Medical and Rennes University Hospital.

References

1. Carrault, G., Cordier, M., Quiniou, R., Wang, F.: Temporal abstraction and inductive logic programming for arrhythmia recognition from ECG. *Artificial Intelligence in Medicine* **28** (2003) 231–263
2. Nédellec, C., Rouveirol, C., Adé, H., Bergadano, F., Tausend, B.: Declarative bias in ILP. In De Raedt, L., ed.: *Advances in Inductive Logic Programming*. IOS Press (1996) 82–103
3. Fromont, E., Cordier, M.O., Quiniou, R.: Learning from multi source data. In: *PKDD'04 (Knowledge Discovery in Databases)*, Pisa, Italy (2004)
4. Muggleton, S., De Raedt, L.: *Inductive Logic Programming: Theory and methods*. *The Journal of Logic Programming* **19 & 20** (1994) 629–680
5. Lloyd, J.: *Foundations of Logic Programming*. Springer-Verlag, Heidelberg (1987)
6. Blockeel, H., De Raedt, L., Jacobs, N., Demoen, B.: Scaling up inductive logic programming by learning from interpretations. *Data Mining and Knowledge Discovery* **3** (1999) 59–93
7. Plotkin, G.: A note on inductive generalisation. In Meltzer, B., Michie, D., eds.: *Machine Intelligence 5*. Elsevier North Holland, New York (1970) 153–163
8. De Raedt, L., Van Laer, W.: *Inductive constraint logic*. *Lecture Notes in Computer Science* **997** (1995) 80–94
9. De Raedt, L., Dehaspe, L.: Clausal discovery. *Machine Learning* **26** (1997) 99–146
10. Moody, G.B., Mark, R.G.: A database to support development and evaluation of intelligent intensive care monitoring. *Computers in Cardiology* **23** (1996) 657–660 <http://ecg.mit.edu/mimic/mimic.html>.