CPD tree learning using contexts as background knowledge

Gérard Ramstein, Philippe Leray

CPD tree learning using contexts as background knowledge

Gerard Ramstein and Philippe Leray

LINA, DUKe research group, 44306 Nantes, France
gerard.ramstein@univ-nantes.fr,philippe.leray@univ-nantes.fr

Abstract. Context specific independence (CSI) is an efficient means to capture independencies that hold only in certain contexts. Inference algorithms based on CSI are capable to learn the Conditional Probability Distribution (CPD) tree relative to a target variable. We model motifs as specific contexts that are recurrently observed in data. These motifs can thus constitute a domain knowledge which can be incorporated into a learning procedure. We show that the integration of this prior knowledge provides better learning performances and facilitates the interpretation of local structure.

Keywords: context specific independence, CPD tree, bayesian network

1 Introduction

Our work falls within the framework of context-specific independence (CSI) [1]. It has been shown that the identification of context-specific relationships within probabilistic relational models constitutes a powerful tool to discover local structures, i.e. interactions that hold on the studied domain. In many applications, conditional independence relationships are true only in specific contexts. A context is a partial configuration of variables that alone induces an effect on a target variable.

In diagnosis for instance, in spite of the variety of human symptoms, a small subset of them may suffice to infer a disease. This restrained set of symptoms forms an example of context. Contexts are valuable pieces of information that can be collected as background knowledge. Recurrent contexts observed over distinct datasets form motifs that can be exploited to discover unexpected associations between previous studies and a new experiment. This is specially the case when the same causes induce different effects (i.e. the same motif affects distinct target variables; for example, a symptom set is shared over previously unrelated diseases).

The problem of learning local structure has already been addressed, notably in [2]. This paper outlines the use of prior domain knowledge for inferring local structures. From a general point of view, incorporating prior domain knowledge into learning algorithms can greatly enhance their performances. Another advantage is that this strategy enables the user to identify recurrent motifs in his

own dataset.

We discuss in section 2 some related works for incorporating knowledge into learning procedures. Section 3 introduces some basic concepts associated to domain knowledge and local structure. In section 4, we propose a method for learning local structure from previously acquired motifs. This strategy is evaluated through experimental results that are presented in section 5. Concluding remarks and future work are given in Section 6.

2 Related work

The identification of CSI in Bayesian networks [1] provides compact data structures for representing probabilistic information. In [2], the authors have proposed CPD trees to express context independencies. Other alternative models have been suggested, such as Recursive Probability Trees (RPTs) [3]. RPTs are a generalization of probability trees that can hold potentials in a factorized way. Factorization yields a more compact representation, but the flexibility of RPTs makes the discovery of motifs more complex. CPD trees have then been adopted for this preliminary work.

The incorporation of prior knowledge in BN learning algorithms has already been investigated. In [4], the authors exploit an ontology by translating concepts and relations into a BN structure. Ontology-based construction of BNs requires the existence of a formal representation of a specific domain, which is not guaranteed. Rather than a global formal approach, we suggest to infer local structure from a collection of motifs, from which a small subset is expected to be consistent with the investigated data. Our strategy, which rather consists in assembling fragmented pieces of information, has also been tackled in [5]. Contrary to our work, this study mainly focus on the reuse of BN fragments, using object-oriented formalism and building blocks called idioms. This work is more an attempt to represent general types of reasoning and does not exploit CSIs. Other approaches address the issue of updating a knowledge base (KB) according to new evidences. In [6], a cyclic approach has been proposed, which incorporates causal discoveries and ontology evolution. Some authors suggest to tune a KB when conflicts have been detected [7]. In [8], the model is capable to evolve, depending of its current state. Contrary to these works, our paper assumes the existence of a well-formed KB of motifs and examines the impact of its incorporation into learning algorithms. Motif discovery has been extensively studied in data mining [9]. Some authors [10] have proposed to reveal interesting attribute sets using BN as background knowledge. If one models motifs as itemsets, this approach shares some similarities with our work, since it combines BN and itemsets. However, it differs in the fact that we guide BN construction using motifs rather than the opposite.

3 Concepts related to the notion of local structure

3.1 Context-specific independence

A variable assignment (*VA*) is a couple (X, x), noted $(X = x)$, where X is a random variable and x the value taken by X. A *context* $\mathbf{c}$ generalizes this concept to a set of variables $\mathbf{C} = \{C_1, \ldots, C_n\}$. A context will be represented in extension as follows: $\mathbf{c} = (C_1 = c_1, \ldots, C_n = c_n)$. The notion of context is generally used to define a set of conditions reducing the interaction between a variable and its parents. For instance, a meteorological context including wind, heavy rain and storms will strongly affect the probability that a tennis match will be played.
As pointed out in [2], the notion of context provides an explicit representation of the local structure. Contexts also yield a simpler encoding of the real complexity of the underlying interactions. To capture this local structure, we introduce a formal foundation for the concept of context. Following definition of *context-specific independence* (CSI) due to [1], let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{C}$ be four disjoint sets of variables. $\mathbf{X}$ and $\mathbf{Y}$ are independent given $\mathbf{Z}$ in context $\mathbf{C} = \mathbf{c}$ if

$$P(\mathbf{X}|\mathbf{Z}, \mathbf{c}, \mathbf{Y}) = P(\mathbf{X}|\mathbf{Z}, \mathbf{c}) \text{ whenever } P(\mathbf{Y}, \mathbf{Z}, \boldsymbol{c}) > 0.$$

3.2 Conditional Probability Distribution Tree

The conventional representation of conditional probability distribution (CPD) takes the form of a table indexed by all possible values of the set of parents. Consequently, a CPD table has $2^{|\mathbf{S}|}$ rows, where $|\mathbf{S}|$ is the number of parents. As explained in [2], such tabular representation is locally exponential and largely overestimates the actual complexity of the involved interactions. An alternative representation exploiting the concept of context defined above is the *CPD tree*. This notion designates a tree whose leaves represent the distribution of the target variable and whose internal nodes represent the parents branching over their values. A tree *path* is an ordered list of *VAs* corresponding to the path from the root towards a given node. A path will be denoted as follows: $[X_1 = x_1, \ldots, X_n = x_n]$. Note that we use brackets for ordered lists and parenthesis for unordered lists such as contexts and motifs. Inducing a CPD tree from a dataset can be performed using learning procedures such as greedy hill climbing [2], using an approach that has been designed for learning decision trees.

4 Learning local structures from motifs

Our objective is to build a CPD tree from a list of motifs collected in a KB=$\{m_k\}$ where $m_k = (X_1^k = x_1^k, \ldots, X_{n_k}^k = x_{n_k}^k)$. A *motif* is a context that is considered as relevant. The interestingness of a motif can be explicitly stated by experts or be related to its recurring nature over different datasets. In this latter case, the same motif has been observed in many situations, but *not necessarily over the same target variable*: in the previous example of meteorological context, the

same causes may affect different outdoor games, such as baseball. Due to their similarity, the same notation will be used for contexts and motifs.

In this section, we first propose an extended version of the concept of CPD tree. Then, we present a learning algorithm from data, using an existing KB and based on two phases: the first one constructs a maximally expanded CPD tree and the second one trims this candidate tree. These two steps are described, as well as a Tabu search extracting an optimal subset of motifs from a given KB.

4.1 Extended definitions for CPD tree learning

We propose to extend the concept of CPD tree by introducing a categorization of its leaves. A leaf represents a CPD associated to a particular configuration (assignment of a variable set). Two situations may arise: either this configuration reveals an remarkable context impacting the target variable, or it is only the consequence of the construction of alternative paths. We call *M-leaf* a leaf associated to a specific context. The prefix M indicates that this leaf may be the evidence for a motif. A M-leaf is graphically represented by symbol $\triangle$. The second type is called a *D-leaf* and is represented by symbol $\square$. This type corresponds to a *default* probability distribution shared by all the D-leaves. We introduce this category to express the absence of a particular context. It presents two advantages: simplification of the encoding of the local structure (all default leaves share the same distribution parameters); better identification of specific interactions (paths leading to a M-leaf). To illustrate these concepts, let consider a voluntary simplified example of a network dedicated to medical diagnosis. Our target variable is associated to heart rate measurement (denoted H), in association with a restricted list of symptoms: chest pain (P), cough (C), indigestion(I) and fatigue (F). The parent set of H is $\mathbf{S} = \{P, C, I, F\}$. All the members of $\mathbf{S}$ are random variables that can take two values: 0(false) and 1(true). In our example, heart rate depends only on a reduced number of symptoms related to a specific disease. Figure 1 shows an example of extended CPD tree that could have be learned from an actual dataset. Note that this tree presents five leaves, but only reveals two interesting features: $\triangle_1$ (bradycardia due to hypothyroidism); $\triangle_2$ (tachycardia due to pulmonary embolism). In our oversimplified example, D-leaves ($\square_1, \square_2, \square_3$) reflect the fact that patients which are not suffering from either hypothyroidism or pulmonary embolism tend to have a normal heart rate. While a CPD table would require 16 rows for variable H, a CPD tree can be summarized into two paths and three parameters (distribution related to bradycardia, tachycardia, and normal heart rate).

We state that a motif is retrieved in an inferred CPD tree $\mathcal{T}$ if there exist at most one path π of $\mathcal{T}$ for which the two following conditions are met : (i) π leads to a M-leaf; (ii) π contains p, that is every VA in p exists in π. Finally, one needs to introduce the concept of *consistency*. A motif p is said to be consistent with a path π if the following rule applies: for any $VA = (X = x)$ in π such as X exists in p, then its assignment in p is x. Similarly, two motifs are said to be consistent if all their common variables also share the same assignment.

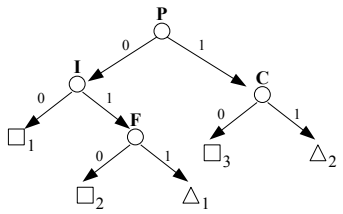

Fig. 1. Example of an extended CPD tree

4.2 Inference of CPD trees

The proposed method for inferring CPD trees follows the standard approach of heuristic search. This procedure considers a dataset D as well as a knowledge base KB. It starts with an initial tree $\mathcal{T}$ consisting in a unique node, labeled as D-leaf. This initialization assumes that D does not contain any motif.

```
      procedure learnCPDtree(T,D,KB)
1.        hasChanged=false
2.        Leaves=getAllLeaves(T)
3.        For each leaf L in Leaves do
4.            UKB=update(KB,L)
5.            if UKB ≠ ∅ then
6.                T=grow(T,L,D,UKB)
7.                hasChanged=true
8.            end
9.        end
10.       if hasChanged then learnCPDtree(T,D,KB)
      end.
```

Procedure *learnCPDtree* first collects all the leaves of $\mathcal{T}$ (*line 2*). These nodes represent potential locations for growing the CPD tree, using the internal method *grow*. The *update* function (*line 4*) returns an updated version of KB that contains the remaining constraints that apply when node $\mathcal{L}$ has been reached. To illustrate this point, let us return to the example of Figure 1 and suppose that our initial KB is $\{(C = 1, P = 1), (I = 1, F = 1)\}$. The update of KB for a current node consists into two actions:

- removal of all the motifs from KB that are not consistent with the actual path. For instance, if this latter is $[P = 0]$ (node marked I in Figure 1), then motif $(C = 1, P = 1)$ has to be removed.
- removal of all VAs that have already met the conditions expressed by a motif. For instance path $[P = 0, I = 1]$ (node marked F) contains a VA $(I = 1)$ that has already been visited. Then, motif $(I = 1, F = 1)$ must be updated into $(F = 1)$.

If UKB is not empty, function *grow* is called (*line 6*). The aim of *grow* is to replace a current leaf $\mathcal{L}$ by a new node. Finally, the procedure *learnCPDtree* is

recursively called as long as the tree can been expanded (*line 10*).

```
      function grow((T,L,D,KB)
1.          pbest=bestPattern(L,D,KB)
2.          vabest=bestVariableAssignment(L,D,pbest)
3.          T'=addNode(T,L,vabest)
4.          return T'
      end.
```

Function *grow* is the core of the tree building. Firstly, it searches for the motif *pbest* that achieves the local maximum score when placed at node $\mathcal{L}$ (*line 1*). We estimated that growing the tree node by node is a better approach than replacing a leaf by a branch (i.e. a whole motif). This strategy yields a more accurate node assignment and provides a more compact tree representation. This is the reason why we select from *pbest* the variable assignment $vabest = (vbest = ibest)$ that achieves the highest score (*line 2*). $\mathcal{L}$ is replaced by a new node $\mathcal{N}$ denoted by *vbest* (*line 3*). By nature, any interior node of a CPD tree possesses a set of outgoing arcs to its children, each one associated with a unique variable assignment. Therefore, a child is added to $\mathcal{N}$ for the arc corresponding to assignment *ibest*. This particular child is a M-leaf; the remaining children are labeled as D-leaves. Since *learnCPDtree* replaces leaves by interior nodes, these labels are updated as long as the tree grows. Note that multiple and possibly interleaved motifs may appear in the same path π, as long as they are consistent.

This algorithm generates a maximally expanded tree, in order to circumvent the problem of local maxima (see [11] for a justification). In a second phase, the tree is trimmed in a bottom-up fashion, using procedure *trimCPDtree*. This method is based on a selection of the node to be pruned (*line 1*). Function *cut* then replaces the node by a leaf $\mathcal{L}$ and creates a new tree $\mathcal{T}'$ (*line 3*). The type of $\mathcal{L}$, either M-leaf or D-leaf, is determined by testing the score of $\mathcal{T}'$ for both options and by selecting the option achieving the highest score. This new tree is then compared to $\mathcal{T}$. If the trimmed tree obtains a better score, it is retained (*line 4*). Finally, this procedure is recursively called as long as a pruning node is available (*line 5*).

```
      procedure trimCPDtree(T,D,KB)
1.          node=selectNode(T)
2.          if node ≠ NIL then
3.                T'=cut(T,node)
4.                if score(T',D)¿score(T,D ) then T=T'
5.                trimCPDtree(T,D,KB)
            end.
      end.
```

Procedure *selectNode* defines the best location for pruning $\mathcal{T}$. This selection can only be performed if a set of candidate nodes has already been determined.

For this purpose, during the previous growing phase, all the nodes have been marked by an additional boolean label *prune* indicating if it can be pruned or not. The following rules were applied:

- a branch can be removed at node $\mathcal{N}$ if it corresponds to the beginning of a new motif. This property can be easily detected by comparing an original motif m to its updated version. If motif m starts at $\mathcal{N}$, then $prune(\mathcal{N}) = true$,
- a node $\mathcal{N}$ that has been created as D-leaf is labeled as $prune(\mathcal{N}) = true$,
- a node $\mathcal{N}$ that does not meet the two previous conditions is labeled as $prune(\mathcal{N}) = false$.

Function *selectNode* returns the node which is the most appropriate cutting point in $\mathcal{T}$. As previously said, the trimming strategy operates in a bottom-up manner. Therefore, our method searches for the nodes $\mathcal{N}$ verifying $prune(\mathcal{N})$ and retains the node having the maximal depth in the tree (in case of ex-aequo, one candidate node is chosen at random). The label *prune* of this node is set to false, in order to reduce the candidate list. When no more candidates remain, *selectNode* returns NIL.

4.3 Motif selection using Tabu search

The inference method presented above is capable to reconstruct a CPD tree from the exact list of motifs that are effectively concealed in a dataset D. From a practical point of view, one can only assume that some motifs in a knowledge base may be effectively retrieved in D. The existence of false positives degrades the performances of our learning procedure. In fact, even if our trimming method reduces the presence of false positives in inferred CPD trees, it cannot eliminate all of them. For instance, the first motif selected in the growing phase cannot be trimmed without the removal of all the motifs that follows it. Therefore, we adopted a Tabu method [12] that models a solution as a boolean vector of size n, where n is the number of motifs in KB. The i^{th} motif is retained if its boolean value is set to true. The optimization algorithm starts with an initial empty solution, assuming that D does no contain any motifs. Neighbors of the current candidate are then generated to find a more adequate solution based on the same list of motifs, except some random mutations (in our implementation, a neighbor contains 1 to 5 boolean changes in relation to the current solution). The fitness of a candidate is defined by the score achieved by the inference algorithm, when applying its motifs. Note that the trimming procedure is still needed, since Tabu search only defines the optimal motif list, but does not prevent a given motif to be present in multiple occurrences in the inferred tree.

5 Experimental results

To evaluate the relevance of using a knowledge base for inferring local structure, we performed multiple experiments with various settings. We simulated

datasets encompassing a certain number of motifs of different sizes. For comparison purpose, we adopted two methods. The first one (further referred as *standard* method) discovered new motifs without any a priori knowledge. This learning procedure was based on the method described in [2]. The second one (further referred as *motif-based* method) was our technique exploiting a knowledge base of motifs.

5.1 Experimental setup

Our experiments were carried out using the following methodology:

- Generation of a golden reference which is a random extended CPD tree containing motifs of different sizes. The total number of variables has been set to 100 for all experiments. We controlled the complexity of the golden reference by defining two random parameters: the number n of motifs; the size s_m of each motif (number of variables composing the motif m). Both n and s_m followed a uniform law on a predefined interval. Three ranges have been fixed for n: $[1, 3]$, $[4, 6]$ and $[7, 10]$. Similarly, we specified three ranges for s_m: $[2, 4]$, $[5, 7]$ and $[8, 10]$. The combination of these intervals defined nine complexity groups. For each of these groups, 20 random extended CPD tree were generated.
- Extraction of a list L_T of true positive motifs from the generated CPD tree.
- Generation of a random dataset using L_T. All the generated datasets contain the same number of instances set to 20,000. One third of instances followed a default normal distribution ($\mu = 0$, $\sigma = 1$). Remaining instances were randomly and uniformly associated to one of the n motifs. The distribution assigned to the i^{th} motif was a normal law ($\mu_i = 3 + i$, $\sigma_i = 0.1$).
- Creation of a KB of motifs belonging to the golden reference as well as false motifs. The proportion of false motifs issued from L_T has been set to 90%. False motifs have been randomly generated from the initial list of variables, so that (i) they observed a comparable complexity (i.e. motif size) in relation to the true motifs; (ii) they were not a subset of any true motif.
- Inference of CPD trees using standard as well as motif-based methods.
- Performance comparison based on precision and recall of the extracted motifs, as well as the compactness of the learned CPD trees.

5.2 Results

Both standard and motif-based methods are capable to retrieve relevant motifs. These method achieve a precision of 1 in respectively 91.5% and 95.1 % of the cases. These good results may be chiefly attributed to the high separability of the original motif distributions. Conversely, the sensitivity was generally more contrasted: a recall of 1 was obtained in only 26.3% of the cases for the standard method, compared to 62.7% for our method. Likewise, recall scores were lower for the first method (mean=0.49), compared to the second one (mean=0.90). Figure 2 details the influence of data complexity on the sensitivity. The number of

motifs impacts on the recall performance for both methods, but more specifically on the standard one. As expected, motif-based method is also more robust with regards to motif size. Another advantage of our method is that it strongly reduces the complexity of the inferred trees. Our procedure induces a tree complexity (number of nodes) that is comparable to that of the golden reference (t-test p-value of 0.32). This is clearly not the case for the standard approach which tends to build large trees (mean relative increase of 34%), making the interpretation of the inferred motifs much more difficult.

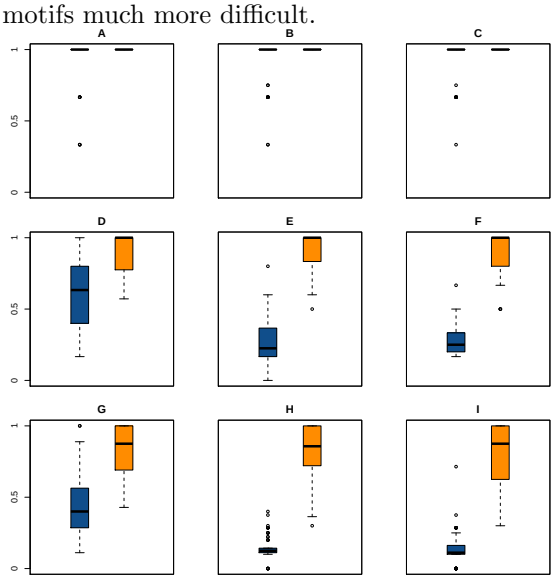

Fig. 2. Recall performances. Rows correspond to the followings ranges of n: $[1, 3]$ (top), $[4, 6]$ (middle) and $[7, 10]$(bottom). Columns correspond to the following ranges of s_p: $[2, 4]$(left), $[5, 7]$(middle) and $[8, 10]$(right). For each figure, left (resp. right) box plot corresponds to the standard (resp. motif-based) method.

6 Conclusion and future work

In this paper,we have proposed a new approach to discover local structure using a priori knowledge defined by a set of "interesting" motifs. We have shown that the incorporation of such motifs greatly improves learning procedures aiming at inferring CPD trees, leading to better performances and tree compactness. The same concept also provides an efficient means to interpret new datasets. Recurrent trends could thus be revealed, allowing experts to investigate new connections and to infer some common causes for previously unassociated phenomenons. There is room for substantial improvement in the current implementation. If the proposed strategy has proven to be efficient for retrieving known motifs, it is not capable in its current form to discover new motifs that are not a mere combination of predefined ones. To address this shortcoming, a hybrid approach could be investigated, that would associate discovery of new motifs

and discrimination between known and new motifs. Another line of future work concerns the definition and the consistency of our knowledge base. We have only considered the information relative to motifs, assuming that the variable domain is stable during the motif acquisition process. In many situations, datasets may contain distinct sets of variables, due to incomplete or incremental experimental designs. Therefore, a context must be associated with its background, that is the variable set in which the context independency has been observed. Finally, we intend to apply our prototype to real-world problems. Functional genomics is a research field that is particularly well adapted for that purpose: public databases gather a vast amount of gene-related data collected from various sources. This information needs to be analyzed in a systematic way and we believe that motif-based approaches would help biologists to make unexpected links between separate studies.

Acknowledgments
This work was supported by the GRIOTE Bioinformatics Research Project of Pays de la Loire Region, France.

References

1. C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in bayesian networks. pages 115–123, 1996.
2. N. Friedman and M. Goldzsmidt. Learning bayesian networks with local structure. In *Learning in Graphical Models*, pages 252–262. MIT Press, 1996.
3. A. Cano, M. Gómez-Olmedo, S. Moral, and C. B. Pérez-Ariza. Recursive probability trees for bayesian networks. In *Proceedings of the Current Topics in Artificial Intelligence, and 13th Conference on Spanish Association for Artificial Intelligence*, CAEPIA'09, pages 242–251, Berlin, Heidelberg, 2010.
4. S. Fenz, A. M. Tjoa, and M. Hudec. Ontology-based generation of bayesian networks. In Leonard Barolli, Fatos Xhafa, and Hui-Huang Hsu, editors, *CISIS*, pages 712–717. IEEE Computer Society, 2009.
5. M. Neil, N. Fenton, and L. Nielson. Building large-scale bayesian networks. *The Knowledge Engineering Review*, 15:257–284, 9 2000.
6. M. B. Messaoud, P. Leray, and N. B. Amor. Semcado: A serendipitous strategy for learning causal bayesian networks using ontologies. In *ECSQARU*, Lecture Notes in Computer Science, pages 182–193, 2011.
7. S. Jr. Eugene, G. Qi, and E. S. Eunice. Bayesian knowledge base tuning. *International Journal of Approximate Reasoning*, 54(8):1000 – 1012, 2013.
8. J. H. Bolt., L.C. van der Gaag, and S. Renooij. Introducing situational signs in qualitative probabilistic networks. *Int. J. Approx. Reasoning*, 38(3):333–354, March 2005.
9. J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: Current status and future directions. *Data Min. Knowl. Discov.*, 15(1):55–86, August 2007.
10. S. Jaroszewicz, T. Scheffer, and D. A. Simovici. Scalable pattern mining with bayesian networks as background knowledge. *Data Min. Knowl. Discov.*, 18(1):56–100, 2009.
11. J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description length principle. *Inf. Comput.*, 80(3):227–248, March 1989.
12. F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.