

Probabilistic Logical Learning for Biclustering: A Case Study with Surprising Results

Nima Taghipour

Daan Fierens

Hendrik Blockeel

Report CW 597, October 2010



Katholieke Universiteit Leuven
Department of Computer Science

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Probabilistic Logical Learning for Biclustering: A Case Study with Surprising Results

Nima Taghipour

Daan Fierens

Hendrik Blockeel

Report CW 597, October 2010

Department of Computer Science, K.U.Leuven

Abstract

Many approaches to probabilistic logical learning have been proposed by now, and several of these have been implemented into powerful learning and inference systems. Given this state of the art, it appears natural to start using these systems for solving concrete problems. This paper presents some results of a case study where several probabilistic logical learning systems have been applied to a seemingly simple problem that exhibits both probabilistic and relational aspects. The results are surprisingly negative: none of the systems we have tried could adequately handle the problem at hand. We discuss the reasons for this. This leads to several conclusions. First, still more effort must be invested in developing full-fledged implementations that can handle a wide range of realistic problems. Second, the intrinsic limitations of certain approaches may not yet be fully understood. Third, the problem we discuss here may be an interesting application for probabilistic logical learning systems, and we invite other researchers to use it as a benchmark for evaluating the applicability of their favorite systems.

Keywords : probabilistic logical learning, biological applications, biclustering
CR Subject Classification : G.3, I.2.6, I.5.3

1 Introduction

There has been a surge of interest in probabilistic logical formalisms, which combine the expressiveness of (first-order) logic with the ability to handle uncertainty using probabilistic reasoning [1,2]. Many approaches have been proposed for performing the tasks of inference and learning in these formalisms, and several of these have been implemented into powerful systems. Given this state of the art, it seems natural to start employing these systems for solving concrete problems.

In this paper we present the results of a case study where we applied several probabilistic logical learning (PLL) systems to a relatively simple problem: biclustering gene expression data. The reason for using PLL for this problem is threefold. First, while special-purpose solutions for the problem already exist [4,5], it could be the case that by translating the problem into a PLL problem it can be solved more accurately or more efficiently, given the amount of effort that has already been invested in making PLL systems efficient. Second, while solutions exist for the basic biclustering problem, certain natural extensions of the biclustering problem (where background knowledge needs to be taken into account) cannot be handled by current standard biclustering methods, while a PLL approach would naturally take such background knowledge into account (see Section 2). Third, this exercise might teach us something about the practical applicability of particular PLL systems. The first two points are more relevant for bioinformatics, while the third is more relevant for the machine learning community.

The particular problem we study was handed to us by bio-engineers [5], who noted that this problem exhibits both relational and probabilistic structure. As such, it is a good benchmark for comparing PLL systems: the problem is of practical importance, it clearly lies in the application domain of PLL systems, and it was not chosen with a particular system in mind (and hence does not favor any particular PLL approaches).

Although the problem seems relatively simple, our results are surprisingly negative. None of the systems that we tried were able to handle the given problem in a satisfactory manner. In some cases this is due to pure implementation issues (implemented solutions do not work in practice), but there are also some more inherent limitations (it is simply not known how to handle a particular problem). In this paper we discuss in some detail the practical and inherent limitations of the systems that we have tried, and present some conclusions.

This paper is structured as follows. In Section 2 we introduce the problem of biclustering gene expression data. In Section 3 we present a generic PLL solution and discuss the suitability of existing PLL formalisms and systems for implementing this solution. In Sections 4 and 5 we present our experiences with two of these formalisms and systems, namely Markov Logic Networks and BUGS. In Section 6 we conclude.

2 The Problem: Biclustering Gene Expression Data

The problem is that of biclustering gene expression data. The *input* is a real valued $M \times N$ matrix containing measured expression levels of M genes under N conditions. The desired *output* is a set of biclusters. A bicluster consists of a set of genes and a set of conditions, such that the genes are co-expressed (i.e., have similar expression levels) under the conditions. In terms of the matrix, a bicluster is a sub-matrix. A sample input matrix and the desired biclusters in this data are depicted in Figure 1 (top). Visually, the goal of biclustering is to identify in the matrix the rectangles that contain the vertical bars. These vertical bars indicate that under a condition (column) in the bicluster, all the genes (rows) in that bicluster are co-expressed. Note that the biclusters can overlap.

There are several solutions for the biclustering problem [4]. The bio-engineers with whom we collaborate (Van den Bulcke et al. [5]) previously proposed a solution based on their own special-purpose implementation of Probabilistic Relational Models (PRMs). However, these solutions are specialized for this problem and it is not obvious how they could be extended. One extension in which Van den Bulcke et al. are interested is the inclusion of background knowledge about relationships between genes and/or conditions (for instance, which genes are in the same pathway). Taking into account such knowledge has the potential of improving the quality of the biclustering. Interest in this extension was the main motivation of Van den Bulcke et al. for contacting us about the biclustering problem. Having a solution for the basic biclustering problem in a general-purpose PLL system (rather than having a special-purpose solution) would make it easier to implement this extension (and future extensions). While achieving this would be a contribution to bioinformatics, analyzing how well existing PLL systems can handle the problem can provide valuable information to the PLL community.

3 Biclustering Using Probabilistic Logical Learning

We first describe a generic PLL solution to the biclustering problem. This solution is essentially a reformulation in PLL terms of the solution of Van den Bulcke et al. [5]. Next we discuss the suitability of various PLL formalisms and systems for implementing this solution.

3.1 A Generic PLL Solution for Biclustering

The biclustering problem can be mapped to a PLL problem, see Figure 1 (bottom). To represent the expression level matrix (the input), we make use of a predicate *Expr*/3. Concretely, the matrix is represented as a set of ground facts of the form *Expr*(g, c, e), giving the expression level e of gene g under condition c . To represent the biclusters (the output), we make use of the fact that each bicluster is completely determined by the set of genes and the set of conditions that it involves. We use a predicate *Gcl*/2 to represent to which biclusters the genes belong: a fact *Gcl*(g, k) indicates that gene g is in bicluster k . Similarly, we

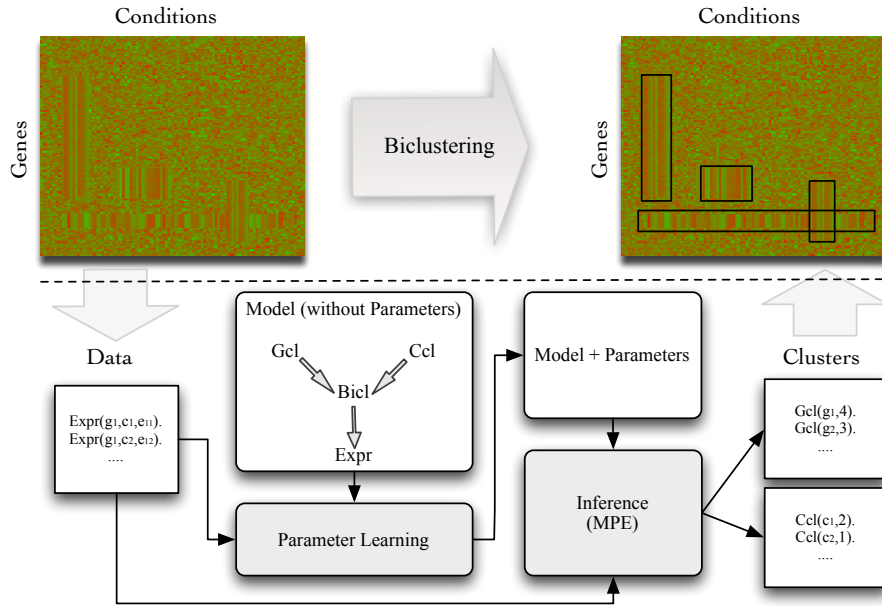


Fig. 1. Top: an example of a gene expression matrix (left) and the biclusters (right). Bottom: a generic PLL solution to biclustering.

use a predicate $Ccl/2$ for conditions. An interpretation of the predicates $Gcl/2$ and $Ccl/2$ fully determines the biclusters. To summarize, the input to the PLL system is an interpretation of the predicate $Expr/3$, and the desired output is an interpretation of the predicates $Gcl/2$ and $Ccl/2$.

The *first step* in solving the problem is to define a probabilistic logical model (see Figure 1). The model should represent the definition of the predicate $Bicl/3$ in terms of the $Gcl/2$ and $Ccl/2$ predicates: $Bicl(g, c, k)$ is true if and only if $Gcl(g, k)$ and $Ccl(c, k)$ are true (this enforces the rectangular shape of biclusters). The model should also represent the dependency of the expression level of each gene-condition pair (g, c) on the bicluster(s) to which it belongs. Each pair (g, c) can be in no, one, or more than one bicluster. In the first case, the expression level is assumed to follow a certain ‘background’ probability distribution. In the second case, the expression level is distributed according to the specific parameters of the bicluster to which (g, c) belongs. In the third case (multiple overlapping biclusters), the expression level is assumed to be distributed as the mean of the distributions of each of the biclusters to which (g, c) belongs.

The *second step* is to learn the parameters of the model given the data (the interpretation of $Expr/3$). These parameters determine the probability of each gene and condition’s cluster assignment, as well as the expression level of gene-condition pairs in each bicluster. Note that this is a learning task with latent variables: the predicates $Gcl/2$, $Ccl/2$, and $Bicl/3$ are unobserved.

The *third step* is to perform inference with the learned model in order to find the biclusters. In PLL terms, we want to find the most likely interpretation of the predicates $Gcl/2$ and $Ccl/2$ given the interpretation of $Expr/3$. This is the so-called *most probable explanation (MPE)* inference problem.

3.2 Requirements for PLL Formalisms and Systems

The above generic PLL solution determines the features that are required from PLL formalisms and systems for solving the biclustering problem.

1. The system should be able to learn from incomplete data, in particular data with *latent variables*.
2. The system should allow us to use *combining rules* or *aggregates*. This is needed to model the effect of overlapping biclusters (we need to be able to compute the mean of several probability distributions).
3. In the absence of combining rules or aggregates, we could still try to discover non-overlapping biclusters (we are then solving a different problem, but this could be seen as a first step towards solving the original problem). In this case, the system should allow us to represent *hard constraints*, namely that clusters should not overlap. The system should then of course also handle these hard constraints correctly during inference and learning.
4. The system should be able to handle *continuous (numerical) data*. This is needed because the expression levels are numerical (real valued).
5. The system should be able to perform *MPE inference*. This is needed for determining the most probable arrangement of biclusters.

The above requirements narrow down our choices among the existing PLL systems, see Table 1 for an overview. Although BUGS is strictly speaking not a logic-based system, we group it together with PLL systems since it offers the required functionalities of PLL systems for our biclustering problem (see Section 5). As Table 1 indicates, BUGS is in fact the only system that meets all the requirements for our problem. The second most suitable system is Alchemy, the available implementation of Markov Logic Networks. Based on this analysis, we selected Alchemy and BUGS for tackling the biclustering problem. We discuss our experiences with these two systems in the following sections.

4 Modeling the Problem with Markov Logic Networks

When modeling the biclustering problem using Markov Logic Networks (MLNs), we were faced with several limitations. The first one is the absence of support for continuous numerical variables (requirement 4 of the previous section) in the MLN system Alchemy.¹ Although so-called hybrid MLNs [8] theoretically handle continuous variables, no support for continuous variables was included in

¹ We use Alchemy as the MLN system in our work because it is the most advanced of all available MLN systems.

Table 1. An overview of various PLL systems with respect to the requirements of the biclustering problem. An entry “Yes/No” stands for a feature that is supported in theory by a PLL system, but is not fully supported in the implementation.

Requirement	BLP			MLNs		
	CP-logic [6]	(Balios) [1, Ch.10]	ProbLog [7]	PRISM [2, Ch.5]	(Alchemy) [1, Ch.12]	BUGS [3]
Learning w. latent variables	Yes	Yes	Yes	Yes	Yes	Yes
Combining rules/aggregates	No	Yes	No	No	No	Yes
Hard constraints	No	No	No	No	Yes/No	Yes
Continuous numerical data	No	No	Yes/No	No	Yes/No	Yes
MPE inference	No	No	No	Yes	Yes	Yes

the available version of Alchemy at the time of our experiments. A second issue is that of representing overlapping biclusters (requirement 2). As described in Section 2, it is possible for a gene-condition pair to belong to multiple overlapping biclusters. In that case, the probability distribution on the expression level should be the mean of the distributions determined by each of the covering biclusters. This is a typical example of a dependency that could be modeled using combining rules [1, Ch.10] or aggregates, but the formalism of MLNs does not include these concepts. An alternative could be to define the formulas in the MLN such that the same effect is achieved. Unfortunately, we do not see how this can generally be done due to the unclear link between the weights of formulas (what we need to specify) and actual probabilities (what we are actually interested in).

As a result of the above limitations, we were forced to move to a restricted version of the problem in which expression levels are discretized and biclusters are not allowed to overlap. In order to impose non-overlapping biclusters, we need to include some *hard* formulas in the MLN (requirement 3). Hard formulas are associated with an infinite weight and should be true in all the possible worlds of the MLN. The hard formulas in the MLN indicate that each gene-condition pair is either in exactly one of the biclusters or in the ‘background’ (indicated by the *Backgr/2* predicate). The final MLN looks as follows.

- (1) $w_1 : \text{Gcl}(g, +n)$
- (2) $w_2 : \text{Ccl}(c, +n)$
- (3) $\infty : \text{Bicl}(g, c, n) \Leftrightarrow \text{Gcl}(g, n) \wedge \text{Ccl}(c, n)$
- (4) $w_3 : \text{Bicl}(g, +c, +n) \Rightarrow \text{Expr}(g, +c, +e)$
- (5) $\infty : \text{Backgr}(g, c) \Leftrightarrow \neg(\text{Bicl}(g, c, 1) \vee \text{Bicl}(g, c, 2) \vee \dots \text{Bicl}(g, c, k))$
- (6) $\infty : \text{Bicl}(g, c, 1) \Rightarrow \neg(\text{Bicl}(g, c, 2) \vee \text{Bicl}(g, c, 3) \vee \dots \text{Bicl}(g, c, k))$
- ...
- (k+5) $\infty : \text{Bicl}(g, c, k) \Rightarrow \neg(\text{Bicl}(g, c, 1) \vee \text{Bicl}(g, c, 2) \vee \dots \text{Bicl}(g, c, k-1))$

The first four formulas in this MLN define the biclusters. The other formulas are used to enforce the non-overlapping property of the biclusters. The *+v* shorthand is used in the same manner as in Alchemy: to represent multiple copies of the

same formula, one for each ground instantiation of variable v . It is also assumed that the number of biclusters k is given [5, Ch.5].

Using the given MLN and the data, we performed parameter learning with Alchemy. While one would expect the infinite weights of hard formulas to remain unchanged (thereby enforcing the desired constraints), we observed that the learning process softens these hard formulas: after the learning phase all these formulas had small weights. This implied, for example, that it was not necessary for gene-condition pairs to be in any of the biclusters, not even the background. As expected, doing MPE inference with such an MLN results in an empty bicluster assignment: no genes were assigned to any bicluster. Note that there is theoretical support for learning MLNs with hard constraints [1, Ch.12] so the above is an implementation problem. Unfortunately, even a patch provided by the Alchemy developers did not solve this problem since the patch is only applicable to some specific cases.

To summarize, despite MLNs/Alchemy being arguably the most popular PLL system of the moment, it was in the end not possible to solve even the restricted version of our problem using this system.

5 Modeling the Problem with BUGS

BUGS (Bayesian inference Using Gibbs Sampling) [3] is a general-purpose statistical package. BUGS supports the plates notation, which provides us with the minimum requirements of relational modeling: it is known that plate models have almost the same expressivity as Probabilistic Relational Models (PRMs) [1, Ch.5+7]. Note that we cannot use PRMs since there is no publicly available implementation of PRMs. Hence, we chose to use BUGS and its plate models instead. As was mentioned in Section 2, BUGS supports all the features needed for modeling the biclustering problem. Hence there was no need to consider a restricted version of the problem, a compromise that we had to make with MLNs. Concretely, the BUGS model deals with continuous expression levels and overlapping biclusters.

Although we were able to *represent* our model using the BUGS formalism, *inference* with the system was not satisfactory. BUGS uses MCMC methods, in particular Gibbs sampling, to perform inference (and Bayesian learning). In our experiments, the sampling process failed to converge in all but the most simple biclustering tasks (simple in terms of for instance the number of biclusters in synthetic data). This failure is due to the known difficulties of sampling methods in the presence of deterministic dependencies between model variables (Gibbs sampling is not guaranteed to converge in such cases). There are indeed a large number of deterministically dependent variables in our model. For example, to model the overlapping biclusters, the model states that an expression value is distributed according to a mixture of Gaussians, all parameters of which are unobserved variables. This definition introduces many deterministic dependencies between the variables, and the same dependencies are repeated through the model, with a number proportional to the size of the problem. It seems that the

available sampling methods in BUGS are not able to manage these dependencies and fail to converge to the solution of the inference task.

To summarize, while it is possible to fully represent the model using BUGS, there are problems when performing inference with this model. These problems have so far prevented us from solving the application in BUGS.

6 Conclusions

There are many successful applications of PLL (see e.g. [1,2]), but often a PLL approach is tried on a case where one knows in advance that it should work, possibly with some small extensions that are implemented on the fly (which is only doable for the original developers of the system). The problem that we investigated is different: we were handed the problem by bio-engineers, and given that the problem has a clear probabilistic relational structure we investigated which PLL systems can be used to solve it. Surprisingly, we have not found any.

The shortcomings of existing systems are often due to practical limitations (a feature is just not implemented), but sometimes also due to deeper limitations that require more research to be solved. We identify the following shortcomings as the most important ones in the existing systems. (1) A common issue in most of the available implementations of PLL systems is the absence of support for numerical data. Support for continuous variables is a desired feature for many real-world applications, including ours. Inclusion of this feature would make PLL systems applicable to a broader range of problems. (2) Difficulty in modeling and handling hard constraints is another limitation of existing PLL systems. As mentioned before (Table 1), many of the existing PLL systems cannot represent hard constraints in the first place. Moreover, the systems that can represent hard constraints in their models have difficulties in dealing with them in practice during inference and learning. In the case of MLNs, using the available implementation, the hard constraints specified in the model become *soft* formulas during the learning process. In the case of BUGS, hard constraints in the form of deterministic dependencies between variables cause the sampling-based inference procedure to fail. In both cases, the problem can be traced back to the difficulty of performing probabilistic inference with a model containing hard constraints. (3) Support for combining rules or aggregates is another important feature that is missing in several of the existing PLL systems (for instance MLNs). Without such support, it is not clear how to represent mixtures of distributions, for instance for dealing with overlapping biclusters.

The problem discussed in this paper may be an interesting benchmark for PLL systems. The problem poses several challenges that the existing PLL systems fail to meet. As discussed, this is due to more fundamental limitations as well as practical limitations. We invite other researchers to lift these limitations and show how our problem can be solved using a general-purpose PLL system. A generator for synthetic datasets [5, Ch.3] for our problem is publicly available at <http://www.biomedcentral.com/content/supplementary/1471-2105-7-43-S3.zip>.

Acknowledgements. We thank Kathleen Marchal, Tim Van den Bulcke and Kristof Engelen (Centre of Microbial and Plant Genetics, Katholieke Universiteit Leuven) for introducing us to the problem of biclustering gene expression data. This research is supported by GOA/08/008 ‘Probabilistic Logic Learning’ and the Research Fund K.U.Leuven.

References

1. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning). The MIT Press (2007)
2. De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S., eds.: Probabilistic Inductive Logic Programming - Theory and Applications. Volume 4911 of Lecture Notes in Computer Science. Springer (2008)
3. Lunn, D.J., Thomas, A., Best, N., Spiegelhalter, D.: Winbugs – a bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing* **10** (2000) 325–337
4. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Trans. Comput. Biology Bioinform.* **1**(1) (2004) 24–45
5. Van den Bulcke, T.: Robust algorithms for inferring regulatory networks based on gene expression measurements and biological prior information. PhD thesis, Faculty of Engineering, Katholieke Universiteit Leuven (2009)
6. Vennekens, J., Denecker, M., Bruynooghe, M.: Representing causal information about a probabilistic process. *Lecture Notes in Computer Science* **4160** (2006) 452–464
7. De Raedt, L., Kimmig, A., Toivonen, H.: Problog: A probabilistic prolog and its application in link discovery. In Veloso, M.M., ed.: *IJCAI*. (2007) 2462–2467
8. Wang, J., Domingos, P.: Hybrid markov logic networks. In Fox, D., Gomes, C.P., eds.: *AAAI*, AAAI Press (2008) 1106–1111