Learning Models of Relational Stochastic Processes

Sumit Sanghai Pedro Domingos Daniel Weld

University of Washington, Seattle WA 98195, USA
sanghai,pedrod,weld@cs.washington.edu

Abstract. Processes involving change over time, uncertainty, and rich relational structure are common in the real world, but no general algorithms exist for learning models of them. In this paper we show how Markov logic networks (MLNs), a recently developed approach to combining logic and probability, can be applied to time-changing domains. We then show how existing algorithms for parameter and structure learning in MLNs can be extended to this setting. We apply this approach in two domains: modeling the spread of research topics in scientific communities, and modeling faults in factory assembly processes. Our experiments show that it greatly outperforms purely logical (ILP) and purely probabilistic (DBN) learners.

1 Introduction

Stochastic processes involving the creation and modification of objects and relations over time are widespread, but relatively poorly studied. Examples of such systems include social networks, manufacturing processes, bioinformatics, natural language, etc. Until recently, graphical models like DBNs and HMMs were the most powerful representations for reasoning about stochastic sequential phenomena. However, modeling relational domains using these graphical models requires exhaustively representing all possible objects and the relations among them. Such a model is both hard to learn and difficult to understand. For example, consider a social network such as an evolving scientific community. One might wish to model the spread of topics across the various groups of researchers. This might mean discovering rules such as "An author's interest in topics in the future is influenced by the interests of his main collaborators and the communities in which he has recently participated." Such rules, being probabilistic, cannot be encoded using pure first-order logic. But a DBN or an HMM would require a model for each individual researcher, and would not generalize from one author to another.

In recent years, researchers have proposed many approaches to combining aspects of first-order logic with probabilistic representations [6]. The most powerful of these is Markov logic networks (MLNs), which combine Markov networks and (for the first time) the full power of first-order logic [14]. However, these models lack the dynamic nature of DBNs and HMMs. Previously, we introduced dynamic probabilistic relational models (DPRMs) [15] and relational dynamic Bayesian networks (RDBNs) [17] for modeling relational stochastic processes, but no learning methods have been proposed for these models, limiting their applicability.

In this paper we extend MLNs to model time-changing relational data. We term this extension DMLNs. Learning DMLNs is relatively easy, requiring only straightforward modifications to an MLN learner. We apply DMLN learning in two domains: the

evolution of research topics in high-energy physics and fault modeling of mechanical assembly plans. Our experiments show that DMLNs greatly outperform a purely probabilistic approach (DBN learning) and a purely logical approach (ILP).

In the next section, we cover MLNs. Then, we introduce DMLNs and describe the learning methods for them (see [16] for more details). Finally, we report our experimental results and conclude with a discussion of related and future work.

2 Markov Logic Networks

A *Markov network* (also known as Markov random field) is a model for the joint distribution of a set of variables $X = (X_1, X_2, \ldots, X_n)$ [5]. It is composed of an undirected graph G on the variables and a set of non-negative potential functions ϕ_k for the state of each clique in the graph. The joint distribution represented by a Markov network is given by $P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}})$ where $x_{\{k\}}$ is the state of the kth clique (i.e., the state of the variables that appear in that clique). Z, known as the *partition function*, is given by $Z = \sum_{x \in X} \prod_k \phi_k(x_{\{k\}})$. Markov networks are often conveniently represented as *log-linear models*, with each clique potential replaced by an exponentiated weighted sum of features of the state: $P(X = x) = \frac{1}{Z} \exp(\sum_j w_j f_j(x))$. This paper will focus on binary features, $f_j(x) \in \{0, 1\}$. In the presence of large cliques, logical functions of the state of the cliques can be used as features leading to a more compact representation than the potential-function form. MLNs take advantage of this.

A first-order knowledge base (KB) can be seen as a set of hard constraints on the set of possible worlds: if a world violates even one formula, it has zero probability. The basic idea in MLNs is to soften these constraints: when a world violates one formula in the KB it is less probable, but not impossible. The fewer formulas a world violates, the more probable it is. Each formula has an associated weight that reflects how strong a constraint it is: the higher the weight, the greater the difference in log probability between a world that satisfies the formula and one that does not, other things being equal.

Definition 1. *[14] A Markov logic network L is a set of pairs (F_i, w_i), where F_i is a formula in first-order logic and w_i is a real number. Together with a finite set of constants $C = \{c_1, c_2, \ldots, c_{|C|}\}$, it defines a Markov network $M_{L,C}$ as follows:*

1. *$M_{L,C}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground predicate is true, and 0 otherwise.*
2. *$M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w_i associated with F_i in L.*

Thus there is an edge between two nodes of $M_{L,C}$ iff the corresponding ground predicates appear together in at least one grounding of one formula in L. An MLN can be viewed as a *template* for constructing Markov networks. The probability distribution over possible worlds x specified by the ground Markov network $M_{L,C}$ is given by

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{F} w_i n_i(x)\right) \quad (1)$$

where F is the number formulas in the MLN and $n_i(x)$ is the number of true groundings of F_i in x. As formula weights increase, an MLN increasingly resembles a purely logical KB, becoming equivalent to one in the limit of all infinite weights. In this paper we focus on MLNs whose formulas are largely function-free clauses and assume domain closure, ensuring that the Markov networks generated are finite [14]. In this case the groundings of a formula are obtained simply by replacing its variables with constants in all possible ways.

3 Modeling Relational Stochastic Processes

Traditionally, graphical models like dynamic Bayesian networks (DBNs), hidden Markov models (HMMs), etc., have been used to model the joint distribution of variables involved in complex stochastic processes. They have been quite successful in practice (e.g., [12]), but they cannot be used to compactly model complex domains with multiple classes, objects and relationships. Modeling such domains requires the representational power of first-order logic. DPRMs [15] and RDBNs [17] provide some of it, and could in principle be learned using ILP and PRM learning techniques, but to date this has not been attempted. In this paper, we instead extend MLNs and their learning algorithms [14, 11] to dynamic domains. This extension turns out to be quite straightforward, and gives us the full power of MLNs in dynamic domains. We experimentally demonstrate the effectiveness of this approach.

3.1 Dynamic Markov Logic Networks

In a relational stochastic process, the *world* is not static. A ground predicate can be $true$ or $false$ depending on the time step t. To model a dynamic relational domain we use the following approach:

1. Instead of standard first-order predicates, we use *fluents*, a special form of predicate having an additional time argument. Time is modeled as a non-negative integer variable. Each predicate in the network is now of the form $R(x_1, \ldots, x_n, t)$, where t denotes time.
2. Our model includes a successor function $succ(t)$, which maps the integer t, representing time, to $t + 1$, i.e., $succ(0) = 1$, $succ(1) = 2$, and so on.
3. We define a *dynamic Markov logic network* (DMLN) to be a set of weighted formulas defined on the fluents.
4. Each formula in the DMLN contains exactly one variable denoting a time slice, and constants may not be used as a fluent's time argument. For example, we disallow formulae such as:
   $\forall \texttt{Auth}, \texttt{Topic}, \texttt{t}, \texttt{t}' : \texttt{Writes}(\texttt{Auth}, \texttt{Topic}, \texttt{t}) => \texttt{Writes}(\texttt{Auth}, \texttt{Topic}, \texttt{t}')$.
   The only exception to this is that we allow formulas where all time arguments are the constant 0; these represent the initial distribution.
5. To enforce the first-order Markov assumption, each term in each formula in the DMLN is restricted to at most one application of the $succ$ function, i.e., a term such as $succ(succ(t))$ is disallowed. This precludes a ground predicate at time t from depending on ground predicates at time $t - 2$ or before.

Given the domain of constants, i.e., the objects at each time slice and the time range of interest, the DMLN will give rise to a ground Markov network whose nodes correspond to the grounding of the predicates (fluents) for each time slice.

3.2 Learning DMLNs

As is common in graphical models, we divide learning DMLNs into two separate problems: parameter learning and structure learning.

The weights of a DMLN can be learned in the same manner as in MLNs, by maximizing the likelihood of a relational database [14]. (A closed-world assumption is made, whereby all ground atoms not in the database are assumed false.) However, as in Markov networks, this requires computing the expected number of true groundings of each formula, which can take exponential time. Although this computation can be done approximately using Markov chain Monte Carlo (MCMC) inference [5], Richardson and Domingos found this to be too slow. Instead, they maximized the pseudo-likelihood of the data, a widely used alternative measure [3]. If x is a possible world (relational database) and x_l is the l^{th} ground atom's truth value, the pseudo-log-likelihood of x given weights w is

$$\log P_w^*(X=x) = \sum_{l=1}^{n} \log P_w(X_l=x_l|MB_x(X_l)) \tag{2}$$

where $MB_x(X_l)$ is the state of X_l's *Markov blanket* in the data (*i.e.*, the truth values of the ground atoms co-occuring with it in some ground formula). Computing the pseudo-likelihood and its gradient does not require inference, and is therefore much faster.

3.3 Structure Learning in DMLNs

In theory one can learn a fully general DMLN for any relational stochastic process. In other words, one could use a single large example to store a history of the entire relational process and learn a DMLN which does not obey the first-order Markovian restrictions. However, this might lead to an unintuitive model and costly inference. In addition, if the number of time slices is large, formulas involving the time variable may become complex and difficult to learn.

However, we can make certain restrictions on the formulas learned and also equip the learner with background knowledge, making the task easier. We divide the structure learning problem into distinct classes depending on the information provided to the learner:

Learning DMLNs with a Markovian assumption: Like learning in a DBN, we can split the domain into multiple examples. Each example, corresponding to a time step t, is a pair of states at time t and $t+1$. When learning, one can avoid formulas which only involve predicates at time t. A model learned in this way is automatically first-order Markovian and stationary.

Learning in presence of background knowledge: The learner is provided with a set of formulas as background knowledge and allowed to modify existing formulas and add a small number of additional formulas so as to maximize the likelihood of the data.

Learning with templates: In ILP systems learning becomes practical only when combined with a *declarative bias*. For example, when learning a relational stochastic process involving actions, one might want to make sure that each formula contains at least one action. This restriction can be specified using templates. Other forms of bias include restricting the number of predicates in a formula, defining an order on the predicates to be considered during search, creating new predicates and formulas for them, etc.

All of these cases can be handled by appropriately extending the MLN learning techniques [11], as we now show.

The structure learned is the one that maximizes the pseudo-likelihood of the database [11]. The algorithm starts with a set of unit clauses and greedily adds or modifies clauses that give the best pseudo-likelihood. At every iteration, weights for all candidate structures are learned. To do this, the weights are initialized to their values from the previous iteration and they quickly converge to the optimum. Each new candidate clause is obtained by adding or removing predicates from already present clauses, or flipping the signs of the predicates. One of two search techniques is used: (i) a beam search where a set of b best clauses is selected and modified until the pseudo-likelihood ceases to improve, and finally the clause which gives the best pseudo-likelihood is added, or (ii) shortest-first search where all good clauses of a smaller length l are added before adding any of a higher length.

The above algorithm may be combined with the first-order Markovian assumption and templates to learn DMLNs. As we will see in the experiments below, no other techniques are needed.

In this paper, we apply our learned model to infer a distribution over the immediately succeeding time step. Additionally, in our examples, all the state variables are observed. So, we use a standard Gibbs sampler for inference.

4 Experiments

In this section we learn DMLNs for two domains in an effort to answer the following questions about methods for learning models of relational stochastic processes.

Q1 Do DMLNs outperform purely logical approaches such as ILP?
Q2 Do DMLNs outperform purely probabilistic methods such as DBN learning?
Q3 Can formulas that model the dynamics of a relational world be learned, and do such formulas outperform pure parameter learning?
Q4 Does enforcing the Markovian assumption improve the accuracy of a learned DMLN?
Q5 Do templates help in learning better DMLNs?

We investigate these questions by applying our algorithms to problems in two domains: (a) modeling the spread of research topics in the theoretical high-energy physics community, and (b) modeling faults in factory assembly processes.

4.1 Evolution of Topics in High-Energy Physics

For our first domain we used the dataset from the KDDCup 2003 [8] which is a collection of papers from the theoretical high energy physics (hep-th) area of `arXiv.org`. This dataset consists of 30,000 papers authored by 9,000 scientists over 10 years. We restricted the author set to scientists who have published at least 10 papers. To identify the topics of the papers we ran Kleinberg's burst-detection algorithm [10] on the words appearing in the titles and abstracts of the papers. We intersected the top "bursty" words with words appearing in highly-cited papers and chose the top fifty. Thus, each paper may be associated with multiple topics. In addition, we clustered both authors and journals using K-means, and added a relation connecting them to their clusters. We organized this dataset into constants of different types, e.g., `Author`, `Paper`, `Journal`, etc., and predicates, e.g., `AuthorOf`, `HasTopic`, `Cites`, etc.

Our task was modeling the evolution of topic popularity over time. Specifically, we wished to predict the distribution of each author's paper topics for one year, given the

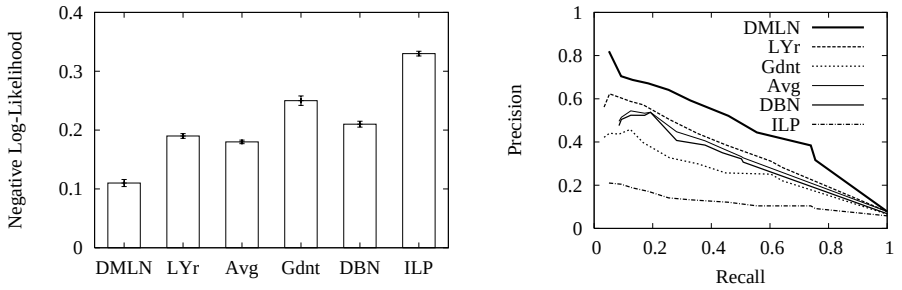

Fig. 1. DMLNs outperform the other methods at predicting author-topic distributions.

distribution over previous years along with citation patterns, and the interests of scientific communities in which she publishes. The test predicate was set to `Authored`$($`A`$,$ `Topic`$,$ `Year`$)$.

We compared DMLNs with five alternative methods: a purely logical approach (the CLAUDIEN ILP engine [4]), a purely probabilistic method (DBN learning), and three approaches that use only statistics concerning the author's topic distribution in previous years. These latter approaches include predicting the most recent topic distribution (*LYr*), predicting the average of distributions across the last three years (*Avg*), and extrapolating the average gradient of the topic distributions over the last three years (*Gdnt*). To compute probabilities using CLAUDIEN, we associate a very high weight (of 20) with each formula. We used three test sets, the topic distributions for years 2000, 2001 and 2002, always using a training set consisting of the data up to the test year. Our results were similar for all three years, so we report results only for 2002.

We compared structure learning with and without background knowledge. The knowledge consisted of formulas such as: authors' future interests are influenced by their past interests, collaborators' interests, and the interests of highly cited authors from the same author cluster; authors are more likely to publish on "bursting" topics than on "dead" ones; authors publish on topics that are "hot" in their favorite journals or other journals in the same journal cluster; etc. We used these formulas first as the complete structure (i.e., we only performed parameter learning) and then as background knowledge (here we learned additional formulas). Learning DMLNs without background knowledge does better than CLAUDIEN, DBN learning and the other methods, but the difference is insignificant. Both knowledge-based approaches did well, and we report on the latter approach, which was marginally better.

We present our results by plotting the negative log-likelihood (Figure 1(a)) and the precision-recall curves (Figure 1(b)). Each measure has its own advantages: the negative log-likelihood directly measures the quality of the probability estimates, while the precision-recall curves are insensitive to the large number of true negatives (i.e., ground predicates that are false and predicted to be false). The figures show clearly that DMLNs learned with background knowledge surpass all other approaches in this domain.

4.2 Faults in Manufacturing Assembly Plans

For our second test, we used a completely-observable version of Sanghai *et al.*'s [15, 17] manufacturing domain. An example in this domain is an execution trace of

an assembly plan comprised of actions such as `Paint`, `Polish`, `Bolt`, etc. There are three classes of objects: `Plate`, `Bracket` and `Bolt`, with propositional attributes such as weight, shape, color, surface type, hole size and hole type, and relations for the parts to which they are attached. Actions are performed at every time step and are fault-prone; for example, a `Weld` action may fail or may weld two incorrect objects based on their similarity to the original objects. This gives rise to uncertainty in the domain and the corresponding dependence model for the various attributes. Given the simulated execution trace of a plan in this domain, we wished to use learning to recover each action's exact fault model.

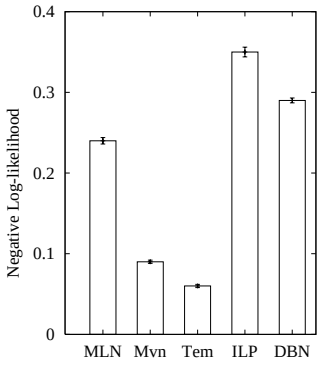

Fig. 2. DMLNs with templates/Markovian assumption outperform other algorithms when learning action models.

Since it is clear that DMLN parameter learning would excel in this domain if given a good set of formulas, we compared DMLN learning (with *no* background knowledge) to CLAUDIEN and DBN learning. Our goal was to answer questions Q3, Q4 and Q5 by learning structure in various settings, i.e., without any help, splitting the examples into time slice pairs, and using templates. We considered a template that made the following restrictions: 1) at most one action predicate per clause, 2) only negated action predicates in clauses, 3) at most five predicates per clause, and 4) biasing to shorter clauses.

Figure 2 shows the negative log-likelihood of unbiased DMLN learning, DMLNs with a Markovian assumption, and DMLNs with templates, compared with ILP and DBN learning, when applied to a 1000-step assembly plan with 100 objects. These results illustrate the ability of DMLNs to learn both the relational structure and the probabilistic parameters of a time-changing process. DBNs have the disadvantage that they separately learn formulas for each ground predicate, while CLAUDIEN has the disadvantage that it gives inaccurate probability predictions. DMLNs combine the capabilities of each. We also note that templates and formulas obeying the first-order Markovian assumption lead to improved learning.

We also tested our algorithms on plans of varying length. As expected, every algorithm improves with an increasing number of time slices.

The DMLN structure learning algorithm took around 7 hours on the hep-th data and 5 hours on the assembly data. CLAUDIEN was allowed to run for a maximum of 15 hours on both the datasets.

5 Discussion

Several researchers have worked on temporal prediction in relational domains like social networks. Successful models include preferential attachment [2] and its extensions. But, many of these models are domain specific and do not explicitly represent uncertainty. DMLNs allow easy specification of complex models using first-order formulae. Learning probabilistic relational planning rules has also received some attention [13]. These application-specific techniques may be viewed in terms of DMLN learning.

In recent years, much research has focused on combining uncertainty with first-order logic (or some subset of it) [6]. Relational Markov models (RMMs) [1] and logical

hidden Markov models (LOHMMs) [9] can be viewed as special cases of DMLNs. Dynamic object-oriented Bayesian networks (DOOBNs) [7] combine DBNs with OOBNs, but no learning algorithms for them have been proposed.

In conclusion, we have shown that MLNs can be successfully extended to learning models of relational stochastic process. Experimental results show that DMLNs are more accurate than previous approaches, such as DBN learning and ILP. Some directions for future work include handling continuous variables, learning in the presence of missing data and hidden state, modeling object creation and deletion, and applying DMLNs to other complex real-world problems.

Acknowledgements

This work was partly funded by NSF grant IIS-0307906, ONR grants N00014-02-1-0408 and N00014-02-1-0932, DARPA project CALO through SRI grant number 03-000225, a Sloan Fellowship to the second author, and an NSF CAREER Award to the second author.

References

1. C. Anderson, P. Domingos, and D. Weld. Relational Markov models and their application to adaptive Web navigation. In *KDD'02*, pages 143–152, 2002.
2. A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286, 1999.
3. J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24:179–195, 1975.
4. L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26(2-3):99–146, 1997.
5. S. Della Pietra, V. Della Pietra, and J.. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4), 1997.
6. T. Dietterich, L. Getoor, and K. Murphy, editors. *Proceedings of the ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields*. ACM press, Banff, Canada, 2004.
7. N. Friedman, D. Koller, and A. Pfeffer. Structured representation of complex stochastic systems. In *Proceedings of Fifteenth National Conference on Artificial Intelligence*, 1998.
8. Johannes Gehrke, Paul Ginsparg, and Jon M. Kleinberg. Overview of the 2003 KDD Cup. *SIGKDD Explorations*, 2003.
9. K. Kersting and T. Raiko. 'Say EM' for selecting probabilistic models of logical sequences. In *UAI-05*, Edinburgh, Scotland, 2005. Morgan Kaufmann.
10. Jon Kleinberg. Bursty and hierarchical structure in streams. In *KDD '02*, 2002.
11. S. Kok and P. Domingos. Learning the structure of Markov logic networks. In *ICML'05*, Bonn, Germany, 2005. ACM Press.
12. K. Murphy. *Dynamic Bayesian networks: representation, inference and learning*. PhD thesis, University of California Berkeley, 2002.
13. Hanna Pasula, Luke S. Zettlemoyer, and Leslie Pack Kaelbling. Learning probabilistic relational planning rules. In *ICAPS'04*, pages 683–691, Whistler, Canada, 2004.
14. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 2005. To appear.
15. S. Sanghai, P. Domingos, and D. Weld. Dynamic probabilistic relational models. In *IJCAI'03*, pages 992–1002, Acapulco, Mexico, 2003. Morgan Kaufmann.
16. S. Sanghai, P. Domingos, and D. Weld. Learning models of relational stochastic processes, 2005. http://www.cs.washington.edu/homes/sanghai/lmrsp.pdf.
17. S. Sanghai, P. Domingos, and D. Weld. Relational dynamic Bayesian networks. *Journal of Artificial Intelligence Research*, 2005. To appear.