Extending Markov Logic to Model Probability Distributions in Relational Domains

Dominik Jain, Bernhard Kirchlechner, and Michael Beetz

Intelligent Autonomous Systems Group
Department of Informatics
Technische Universität München

Abstract. Markov logic, as a highly expressive representation formalism that essentially combines the semantics of probabilistic graphical models with the full power of first-order logic, is one of the most intriguing representations in the field of probabilistic logical modelling. However, as we will show, models in Markov logic often fail to generalize because the parameters they contain are highly domain-specific. We take the perspective of generative stochastic processes in order to describe probability distributions in relational domains and illustrate the problem in this context by means of simple examples.
We propose an extension of the language that involves the specification of a priori independent attributes and that furthermore introduces a dynamic parameter adjustment whenever a model in Markov logic is instantiated for a certain domain (set of objects). Our extension removes the corresponding restrictions on processes for which models can be learned using standard methods and thus enables Markov logic networks to be practically applied to a far greater class of generative stochastic processes.

1 Introduction

In artificial intelligence (AI), a variety of applications can greatly benefit from a unification of logical and probabilistic knowledge representations. The former enable us to deal with complex, relational domains by offering an expressive language, and the latter allow us to soundly handle uncertainty. In any sufficiently complex AI application, both are of utmost importance and need to be fully integrated. Yet for a long time, the development of both strands has been largely separate. Especially in recent years, however, a number of alternative approaches towards a unification have been proposed. Theoretical contributions to the field that has now emerged as statistical relational learning date back to at least Nilsson [1], Halpern [2] and Bacchus [3], while the more practically oriented research has only recently gained momentum [4,5,6,7]. One of the most promising approaches currently available is Markov logic [8,9], as it essentially combines, unlike most other approaches, the full power of first-order logic with probability. It is thus one of the most general, yet it is still supported by a suite of tools that are geared towards practical applicability (the Alchemy system [10]). Because of their exceptional expressivity and simplicity, Markov logic networks have gained a lot of attention lately, including an invited talk by Pedro Domingos at AAAI-06.

In principle, any language that unifies statistical and relational representations and that is furthermore supported by a number of sufficiently efficient learning and inference algorithms, so that it can truly be applied in practice, is not only inherently appealing but can also serve as an interface layer between learning and inference on the one side and higher-level AI applications on the other [11], increasing the interoperability between implementations on both sides of the layer if it is established as a standard. Markov logic is a prime candidate for a representation language on which such an interface layer could be based.

1.1 Application Scenarios

There are countless applications for probabilistic relational representations and the corresponding interface layer. Consider, for example, a system such as ASpoGAMo [12], which observes football games, recognizes and classifies the ball actions that occur within them, determines the parameters of the actions, characterizes the situations in which the actions are performed and analyzes the effects of the actions. In the context of such an application, we could use the data that is collected to build comprehensive models of the game process, which could then be used to answer a wide range of queries automatically. For instance, we might ask for the probability that a pass played by a certain type of player succeeds in a given situation — or the probability that a ball passed by a side midfielder is generally received by a striker if it is played with specific parameters (speed, direction) in a given situation.

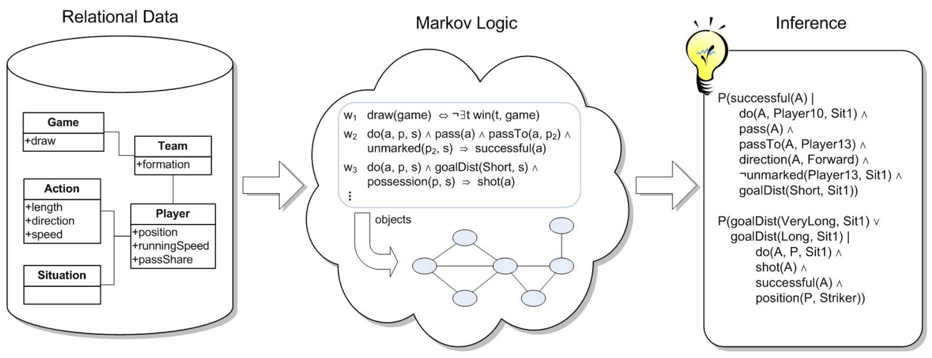

Relational Data
Game
+draw
Team
+formation
Action
+length
+direction
+speed
Player
+position
+runningSpeed
+passShare
Situation
Markov Logic
w1 draw(game) ⇔ ¬∃t win(t, game)
w2 do(a, p, s) ∧ pass(a) ∧ passTo(a, p2) ∧ unmarked(p2, s) ⇒ successful(a)
w3 do(a, p, s) ∧ goalDist(Short, s) ∧ possession(p, s) ⇒ shot(a)
⋮
objects
Inference
P(successful(A) | do(A, Player10, Sit1) ∧ pass(A) ∧ passTo(A, Player13) ∧ direction(A, Forward) ∧ ¬unmarked(Player13, Sit1) ∧ goalDist(Short, Sit1))
P(goalDist(VeryLong, Sit1) ∨ goalDist(Long, Sit1) | do(A, P, Sit1) ∧ shot(A) ∧ successful(A) ∧ position(P, Striker))

Fig. 1. Applying Markov logic

Figure 1 depicts a schematic application of Markov logic that could be capable of answering such queries. It shows the typical process of learning a model from relational data and reasoning about it. The source data could, for example, be given in a relational database. Ideally, we would then only need to provide a set of formulas that hold in the domain and learn the model's parameters in order to be able to query the model by instantiating it for a concrete set of objects and calculating the conditional probabilities we are interested in.

There are several obvious characteristics of such applications and the representations they need in order to perform their tasks. First, the models created and used must represent uncertainty: football games are notoriously nondeterministic, for results of actions are caused by the interplay of situations and actions that are only incompletely characterized, and important information for analysis, such as the intentions of players, cannot be inferred. Second, the questions we ask essentially make use of the power of natural language: They ask about classes of players, infer information about the relation of actions' effects and the situations they are performed in, and the domains of events we query are dynamically determined. Whatever the concrete application, a probabilistic relational representation that is able to represent full-joint distributions over any given domain of a certain type is desirable, for complex queries cannot be answered otherwise.

1.2 Contributions

In this paper, we lay down a number of properties of probabilistic logical representation languages that are essential for the modelling of some aspects of probabilistic relational domains. In particular, the language must be capable of describing general principles about multiple objects having similar properties, which should then be applicable in several contexts, i.e. in several different domains. Markov logic networks that are learned using standard methods in many cases do not possess some of these properties and consequently fail to represent the intended probability distribution when we move from one domain to another, i.e. when we perform a *domain shift*.[1] Because Markov logic seems to have been used mainly in rather specialized applications, the problem we describe may have been hitherto unnoticed. We illustrate the problem using particularly simple examples and subsequently propose an extension of the language that solves it. In essence, our solution involves the formulation of hard constraints on probabilities that are used to dynamically calculate some of the probabilistic parameters of models in Markov logic that would otherwise remain fixed.

In the following section, we thus begin by stating the properties of representation languages we deem desirable. Next, in Sect. 3, we briefly introduce Markov logic networks and define the problems that limit their applicability to relational domains. Subsequently, in Sect. 4, we show why Markov logic networks are, under certain conditions, unable to handle domain shifts and, in Sect. 5, we suggest a corresponding extension of the language, which we explain in detail. Finally, we summarize our results and provide an outlook on future research.

[1] Note that the term *domain* is used at two different levels. At the higher level, we mean the general scenario that a model deals with, i.e. a specification of the types of objects, their attributes and the relations that are considered. At the lower level, which is relevant in this particular case, we mean a concrete set of constants referring to objects in the world (i.e. instances of the types declared at the higher level).

2 Demands on the Representation Language

We believe that, beyond the ability of handling uncertainty as well as a high degree of complexity, a probabilistic logical language should fulfill a number of additional requirements. In this paper, we take the perspective of probabilistic knowledge representation in relational domains, i.e. we seek to model probability distributions over relational data. A common way to view a probability distribution is to see it as the result of a generative stochastic process. Correspondingly, the goal of probabilistic logical modelling is simply to obtain an accurate model of the underlying process. In a relational setting, the process typically generates objects with certain attributes as well as relations between objects according to a set of rules. The representation language should allow us to concisely describe these rules. In this context, the aforementioned criteria that the representation language and its associated learning and inference mechanisms should ideally fulfill are:

1. It should be possible for a domain expert to specify his/her knowledge in a straightforward, declarative way. The addition of new, relevant knowledge should lead to an improvement of the corresponding model (or at least leave the model unchanged); and the failure to represent specific aspects of a domain should not render the model useless with respect to aspects untouched by the omission.
2. It should be possible to unambiguously define a probability distribution over arbitrary domains (of a certain type) by characterizing the corresponding stochastic process. In particular, a model should be universally valid, specifying arbitrarily general rules that may be independent of concrete objects. The probability model should generalize to arbitrary domains and arbitrary objects within them — much in the same way as universally quantified formulas in first-order logic that deal with objects of a certain type are applicable to arbitrary objects of these particular types.

When working with Markov logic, issues related to the first point can be observed — in the sense that additional, relevant formulas may negatively affect certain queries in an unforeseeable fashion — but we found this to be a result of a problem that is more closely related to the second point, for which we provide a solution further on. In Chap. 3.2, we point out the cause of the problem, and in Sect. 4, we analyze it in more detail, providing some examples. But first, we lay the necessary groundwork by introducing Markov logic networks.

3 Markov Logic Networks

Markov logic networks (MLNs) are probabilistic logical models that combine the semantics of probabilistic graphical models (namely Markov networks) with the full power of first-order logic. An MLN can be seen as a set of constraints on the set of possible worlds that is implicitly defined by a set of logical predicates and a set of constants, as each logical atom that can be constructed using these domain elements is viewed as a boolean variable. Specifically, the constraints are formulas in first-order logic with attached numeric weights that quantify their hardness.

Formally, 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, the weight of formula F_i. Together with a finite set of constants C, an MLN defines a Markov network $M_{L,C}$, the *ground Markov network*, as follows:

1. $M_{L,C}$ contains one binary node for each possible grounding of each predicate appearing in the formulas of the Markov logic network L.
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 w_i.

The ground Markov network's set of variables X is the set of ground atoms that is implicitly defined by the predicates in the MLN and the set of constants C. The Markov logic network specifies a probability distribution over the set of possible worlds $\mathcal{X}$, i.e. the set of possible assignments of truth values to each of the ground atoms in X, as follows,

$$\begin{aligned} P(X = x) &= \frac{1}{Z} \cdot \exp\left(\sum_i w_i \cdot n_i(x)\right) \\ &= \frac{\exp\left(\sum_i w_i \cdot n_i(x)\right)}{\sum_{x' \in \mathcal{X}} \exp\left(\sum_i w_i \cdot n_i(x')\right)} \end{aligned} \tag{1}$$

where the inner sums are over indices of MLN formulas and $n_i(x)$ is the number of true groundings of the i-th formula in possible world x.

3.1 Inference

Since (1) provides the full-joint distribution over the variables in X, it can be used to compute arbitrary conditional probabilities: The probability that a formula F_1 holds given that formula F_2 does can be computed as

$$\begin{aligned} P(F_1 \mid F_2, L, C) = P(F_1 \mid F_2, M_{L,C}) &= \frac{P(F_1 \wedge F_2 \mid M_{L,C})}{P(F_2 \mid M_{L,C})} \\ &= \frac{\sum_{x \in \mathcal{X}_{F_1} \cap \mathcal{X}_{F_2}} P(X = x)}{\sum_{x \in \mathcal{X}_{F_2}} P(X = x)} \\ &= \frac{\sum_{x \in \mathcal{X}_{F_1} \cap \mathcal{X}_{F_2}} \exp\left(\sum_i w_i \cdot n_i(x)\right)}{\sum_{x \in \mathcal{X}_{F_2}} \exp\left(\sum_i w_i \cdot n_i(x)\right)} =: \frac{W_{F_1 \wedge F_2}}{W_{F_2}} \end{aligned} \tag{2}$$

where $\mathcal{X}_{F_i}$ is the set of possible worlds in which F_i holds, and $W_{F_1 \wedge F_2}$ and W_{F_2} are the sums of exponentiated sums of weights for possible worlds where $F_1 \wedge F_2$ holds and where F_2 holds respectively (this is a notation that we continue to use further on).

3.2 Problems

Since a Markov logic network is not (necessarily) specific to concrete domain elements but is instead designed to be applicable to arbitrary domains over the classes of objects that it models, MLNs should satisfy the generality requirement made above. However, with the current set of concepts in place, it is in many cases not possible to learn the characteristics of the respective generating processes from data, because parameter learning in MLNs is an ill-posed problem, and the solution that is obtained is usually specific to the concrete set of objects that were used for learning. An MLN obtained via parameter learning cannot, without restrictions, be applied to a domain with a different number of objects than the one it was learned with.

The weights in an MLN are usually learned using MAP estimation or, in the absence of a prior distribution over parameter settings, maximum likelihood — i.e. they are chosen in such a way that the probability of a training database, which specifies the truth values of ground atoms for one particular domain, is maximized. Yet clearly, there can be more than one generative stochastic process that could have produced any given training database, and obviously, a process cannot be uniquely identified through a single sample taken from it (such as a training database). Unfortunately, the MLN language itself is not sufficient in order to make the necessary distinctions prior to parameter learning, for it lacks the ability to specify unconditional independencies in terms of structure; in many cases, the MLN language therefore does not allow us to adequately characterize the concrete process we are dealing with when the weights are yet unknown.

When applying an MLN whose weights were learned using a particular training database is applied to a different domain, it is assumed that the parameters that adequately describe the concrete distribution observed in the training data also fully characterize the process that generated it. This assumption, however, is rarely justified. And whenever the assumption is indeed not justified, the MLN models the desired probability distribution only for a single instantiation, namely the training database. It is thus no more useful than the corresponding ground Markov network, and the general, relational character of the model is essentially lost.

4 Analyzing the Problem

Let us consider a simple example to explain why this is the case. We first introduce the domain we will use for our experiments before turning to the problem of domain shifts in detail.

4.1 The Example Domain

In our example domain, we consider the simplest of scenarios where there are two types of objects and a relation connecting them. Suppose the concrete types of objects are people and drinks, and that there exists a relation that captures the consumption of drinks. Both people and drinks are characterized by a single attribute that classifies them: Each person has a rank (either *Student* or *Professor*), and each drink has a type (either *Tea* or *Coffee*). In an MLN, both attributes can be represented using appropriate predicates, e.g. $\mathit{rank}(\mathit{person}, \mathit{rankvalue})$ and $\mathit{drinkType}(\mathit{drink}, \mathit{typevalue})$.[2] Now suppose there is a difference in the drinking habits of students and professors; professors might, for example, consume more drinks than students do and students might not drink any coffee at all. In general, we can describe arbitrary drinking habits in a Markov logic network simply by including the following conjunctions[3]

$$\begin{array}{lr@{\;}l}
w_1 & \mathit{consumed}(p,d) \wedge \mathit{rank}(p,\mathit{Student}) & \wedge\, \mathit{drinkType}(d,\mathit{Tea}) \\
w_2 & \mathit{consumed}(p,d) \wedge \mathit{rank}(p,\mathit{Student}) & \wedge\, \mathit{drinkType}(d,\mathit{Coffee}) \\
w_3 & \neg\mathit{consumed}(p,d) \wedge \mathit{rank}(p,\mathit{Student}) & \wedge\, \mathit{drinkType}(d,\mathit{Tea}) \\
w_4 & \neg\mathit{consumed}(p,d) \wedge \mathit{rank}(p,\mathit{Student}) & \wedge\, \mathit{drinkType}(d,\mathit{Coffee}) \\
w_5 & \mathit{consumed}(p,d) \wedge \mathit{rank}(p,\mathit{Professor}) & \wedge\, \mathit{drinkType}(d,\mathit{Tea}) \\
w_6 & \mathit{consumed}(p,d) \wedge \mathit{rank}(p,\mathit{Professor}) & \wedge\, \mathit{drinkType}(d,\mathit{Coffee}) \\
w_7 & \neg\mathit{consumed}(p,d) \wedge \mathit{rank}(p,\mathit{Professor}) & \wedge\, \mathit{drinkType}(d,\mathit{Tea}) \\
w_8 & \neg\mathit{consumed}(p,d) \wedge \mathit{rank}(p,\mathit{Professor}) & \wedge\, \mathit{drinkType}(d,\mathit{Coffee})
\end{array}$$

which exhaustively define the various cases for each possible consumption of a drink by a person. The weight w_i of each conjunction describes, when viewed relative to the weights of the other conjunctions, how likely the respective case really is.

4.2 Domain Shifts

The most important observation in this context is that if the only formulas the MLN includes are the above conjunctions, then the marginal distributions of people's ranks and drink types are fully determined by the drinking habits. People are less likely to have a certain rank if people of that rank are less likely to (not) consume drinks. Especially in the case of ranks, a perhaps more intuitive model would state that the marginal distribution of ranks (when we know nothing about consumptions) is independent of *potential* consumptions and that it is in fact rather the ranks that lead to a specific drinking behaviour. During parameter learning, this is, however, a distinction that is not being made, since we obtain any one of infinitely many weight vectors that accurately represent the distribution present in the training data but which generalize to variable-size domains in different ways.

For example, let us look at a single person, say P, and the drinks that P can potentially have consumed. Let the set $\mathcal{X}_m$ contain the set of possible worlds for

[2] Note that when modelling the attributes in this way, it is necessary to include constraints that define the attribute values as mutually exclusive and exhaustive. Henceforth, we silently assume that, for each attribute, the corresponding constraints are included in each model.

[3] We adopted the convention of using lower-case letters as variables and words beginning with upper-case letters as constants. Any free variables in the formulas are (implicitly) universally quantified.

a domain with only one person P and m drinks. The probability of P being a student is, following (2), given by

$$\begin{aligned} p_m &= P_{\mathcal{X}_m}(\mathit{rank}(P, \mathit{Student})) \\ &= \frac{f(m, w_1, w_2, w_3, w_4)}{f(m, w_1, w_2, w_3, w_4) + f(m, w_5, w_6, w_7, w_8)} \end{aligned} \tag{3}$$

with

$$\begin{aligned} & f(m, x_1, x_2, x_3, x_4) \\ &= \sum_{t=0}^{m} \sum_{c_t=0}^{t} \sum_{c_c=0}^{m-t} \binom{m}{t} \binom{t}{c_t} \binom{m-t}{c_c} \exp(x_1 c_t + x_2 c_c + x_3(t - c_t) + x_4(m - t - c_c)) \end{aligned} \tag{4}$$

where t is the number of teas, c_t is the number of teas that are consumed and c_c is the number of coffees that are consumed. If there is an asymmetry in the impact of the weights for students and professors, then p_m is not independent of m. For example, if $w_2 = -100$ and all other weights are 0 (i.e. students hardly ever drink coffee but all other consumption events are equally likely), then $p_1 \approx \frac{4-1}{8-1} \approx 0.4286$, $p_2 \approx \frac{16-7}{32-7} \approx 0.3600$ and $p_3 \approx \frac{64-37}{128-37} \approx 0.2967$. The probability that any given person is a student thus decreases as the number of drinks in the domain increases, which was to be expected, because the worlds in which P is a student and consumes even one cup of coffee are (virtually) impossible, and the fraction of such worlds increases with the number of drinks. So depending on the number of drinks, the weight vector apparently determines a different marginal for $\mathit{rank}(p, \mathit{Student})$. However, the generating process might dictate a more intuitive view, where the marginal probability of a certain rank is independent of the number of drinks; it might, for example, state that the marginal probability of a person being a student is $\frac{1}{3}$.

Let us consider ways in which we might address the problem. A perhaps straightforward formula to add to the MLN would be a unit clause such as $\mathit{rank}(p, \mathit{Student})$, which can obviously be used to influence the probabilities of worlds in which there are students and hence the marginal probability of the rank *Student*. In fact, we could argue that a unit clause is the only candidate formula for such a correction, because unit clauses are the only formulas that affect *only* the marginal distribution we are interested in and nothing else. Unfortunately, the impact of the unit clause's weight does not depend on the number of drinks, so a correction for arbitrary domains is not possible. This is evident from the fact that the sum of exponentiated sums of weights for worlds where P is a student would only be scaled by a factor of $\exp(w)$ if w was the weight of the newly introduced formula $\mathit{rank}(p, \mathit{Student})$. Therefore, we cannot obtain a size-invariant marginal distribution for P's rank when using an asymmetric configuration of weights of the eight conjunctions, because the unit clause can only have the desired effect for a fixed number of objects; and unfortunately, the weight configurations obtained via standard parameter learning are not ensured to be symmetric.[4] The inclusion of the unit clause can, however, result in pre-

[4] By symmetric, we here mean a symmetric impact of the weights, such that the resulting distribution is a uniform distribution.

cisely the correction that we want for a *fixed* number of drinks. Yet obviously, our generality requirement is not met.

Whenever we learn the parameters of an MLN that contains at least one proper relation connecting objects of different types, we will usually obtain very accurate weights for the concrete set of objects found in the training database (provided that our formulas are sufficiently expressive), yet the weights will not generalize to other domains if we impose the requirement that certain marginals remain invariant. In particular, if a relation exists, then the unit clause's weight on its own gives no indication of the probability of the corresponding attribute, for its purpose is primarily to counterbalance the effects of the relation — and these effects depend on the number of tuples that can be constructed.

To further illustrate this problem, let us compare a Markov logic network, which we can instantiate on demand, to several Bayesian networks that we can specifically create for each concrete domain (set of objects) according to a well-defined schema (as we might describe it in, for example, a Bayesian logic program [6]). As MLN formulas, we simply use the eight conjunctions describing consumption behaviour listed above as well as unit clauses for *rank* and *drinkType*, plus a rule that states that a drink cannot be consumed by more than one person. To learn the MLN's parameters, i.e. the formulas' weights, we use a training database containing one student (who drank one cup of tea) and two professors (where the first drank one cup of tea and one cup of coffee and the other drank just one cup of coffee). Let L be the resulting Markov logic network — obtained via maximum likelihood parameter learning.

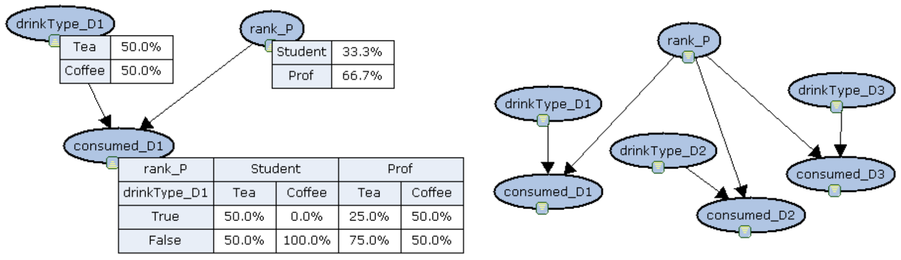
Let us now perform inferences in several ground Markov networks based on L. We compare results obtained for several domains: By $D_{n,m}$ we denote a domain where the set of people contains n elements, $\{P_1 = P, \ldots, P_n\}$, and the set of drinks contains m elements, $\{D_1, \ldots, D_m\}$.[5] In particular, we will look at $D_{1,1}$ and $D_{1,3}$; Figure 2 shows the corresponding Bayesian networks, in which we assume that the marginal distributions of both ranks and drink types are to remain fixed.

Comparing the results in Table 1, we gather that the probabilities computed using the Markov logic network are clearly dependent on the number of objects (as expected). Only for domain $D_{3,4}$, which contains precisely the number of objects found in the training database, does the MLN compute the desired probabilities. There is, unfortunately, no formula that we could have added to the MLN in order to encode the invariants of the generative stochastic process that we imagine to have created the training database.

4.3 Domain-specific Modifications

In fact, the set of formulas included above would in theory already be sufficient to represent precisely the size-invariant probability distribution that we want,

[5] Note that we here assume a typed predicate logic, where the set of constants is not a mere set but rather an ordered set of sets containing one set of objects for each class.



drinkType_D1	
Tea	50.0%
Coffee	50.0%

rank_P	
Student	33.3%
Prof	66.7%

rank_P	Student		Prof	
drinkType_D1	Tea	Coffee	Tea	Coffee
True	50.0%	0.0%	25.0%	50.0%
False	50.0%	100.0%	75.0%	50.0%

Fig. 2. Bayesian Networks $B_{D_{1,1}}$ (left) and $B_{D_{1,3}}$ (right). The conditional probability tables in $B_{D_{1,3}}$ are the same as in $B_{D_{1,1}}$.

	$M_{L,D_{1,1}}$	$B_{D_{1,1}}$	$M_{L,D_{1,3}}$	$B_{D_{1,3}}$	$M_{L,D_{3,4}}$
$P(drinkT(D_1, Tea))$	68.3%	50.0%	68.3%	50.0%	50.0%
$P(rank(P, Stud))$	29.3%	33.3%	32.0%	33.3%	33.3%
$P(rank(P, Stud) \mid cons(P, D_1))$	30.1%	25.0%	32.7%	25.0%	25.0%
$P(rank(P, Stud) \mid \neg cons(P, D_1))$	29.0%	37.5%	31.6%	37.5%	37.5%
$P(rank(P, Stud) \mid cons(P, D_1) \wedge drinkT(D_1, Tea))$	45.4%	50.0%	48.5%	50.0%	50.0%
$P(drinkT(D_1, Tea) \mid cons(P, D_1) \wedge rank(P, Prof))$	51.8%	33.3%	51.8%	33.3%	33.3%
$P(cons(P, D_1) \mid drinkT(D_1, Tea) \wedge rank(P, Prof))$	25.0%	25.0%	25.0%	25.0%	25.0%
$P(rank(P, Stud) \mid cons(P, D_1) \wedge cons(P, D_2))$			33.5%	18.2%	18.2%

Table 1. Inference results: ground Markov networks and Bayesian networks compared.

since all the formulas are conjunctions as they would appear in an MLN translated from a model obtained via knowledge-based model construction (KBMC) [8]. (Such an MLN contains a conjunction of literals for each entry of each conditional probability table in the KBMC model — the conjunction capturing the parent-child configuration that corresponds to the entry and the weight being the logarithm of the probability value.) We still fail to find the set of weights that will generalize in the intended way, because, as mentioned previously, parameter learning is an ill-posed problem: There are infinitely many weight vectors that represent the same probability distribution given a specific domain/set of objects, yet only a subset of these weight vectors generalizes in the intended way to domains of variable size.

Notice that in an MLN derived from a KBMC model, the distributions over attribute values for which unit clauses exist would be uniform distributions if the unit clauses themselves were removed from the MLN; therefore, there is no dependence on domain size. Starting with a weight vector obtained via conversion from a KBMC model, we can, however, construct infinitely many weight vectors such that the probability distribution remains unchanged for a concrete number of objects but changes to varying degrees as we change the number of objects with which the model is instantiated. The idea behind the construction is that if the weight of any unit clause was to be modified, the weights of other formulas within which the unit clause appears could be adjusted to compensate for the modification of the unit clause in such a way that the probability distribution remains unchanged for a specific number of objects. Yet clearly, any modification of a unit clause's weight prevents the remaining formulas from generating a uniform distribution for an arbitrary number of objects. Hence size-dependence results.

In an MLN L derived from a KBMC model, the set of formulas can be partitioned into classes of mutually exclusive and exhaustive formulas, as for each set of conjunctions representing a certain conditional distribution, there is exactly one true grounding among all the groundings of the conjunctions with the same variable bindings. Let C be the set of classes minus the classes that contain only unit clauses. Now if the weight of a unit clause F_j (an atom that makes a statement about an attribute of an object of some type T) was to be changed by adding an arbitrary Δw in a modified MLN L', we could choose an arbitrary non-empty subset $S \subseteq \{C_i \in C \mid \mathit{contains}(C_i, F_j)\}$ of the classes that contain formulas that contain F_j and adjust certain formula weights to cancel out the previous change. In particular, for a concrete domain D, we apply to each $(F_i, w_i) \in C_k$ where F_i contains F_j and $C_k \in S$ the following modification,

$$w_i' = w_i - \Delta w \cdot \frac{1}{n_{C_i,D} \cdot \frac{1}{|D_T|}} \cdot \frac{1}{|S|} \tag{5}$$

where $n_{C_i,D}$ is the number of groundings of each of the conjunctions in class C_i for domain D and D_T is the set of domain elements of type T in D.

We can easily show that the probability of any possible world x defined over the domain D remains unchanged by our modification,

$$\begin{aligned} P_{M_{L',D}}(x) &= \frac{1}{Z'} \cdot \exp\left(\sum_i w_i' \cdot n_i(x)\right) \\ &= \frac{1}{Z'} \cdot \exp\Bigg(\sum_i w_i \cdot n_i(x) + \Delta w \cdot n_j(x) - \\ &\qquad \sum_{C_i \in S} \Delta w \cdot \frac{1}{n_{C_i,D} \cdot \frac{1}{|D_T|}} \cdot \frac{1}{|S|} \cdot \frac{n_{C_i,D}}{|D_T|} \cdot n_j(x)\Bigg) \\ &= \frac{1}{Z} \cdot \exp\left(\sum_i w_i \cdot n_i(x)\right) = P_{M_{L,D}}(x) \end{aligned} \tag{6}$$

as for each object O of the $n_j(x)$ objects for which the unit clause F_j is true, there are $\frac{n_{C_i,D}}{|D_T|}$ combinations for grounding all the variables appearing in each of the formulas in C_i except the one variable we assume to be bound to O, and for each combination, there is exactly one true grounding of a formula with a modified weight.

Therefore, provided that we instantiate ground Markov networks only for domain D, L' thus yields exactly the same probability distribution as L. So when learning parameters using a training database over domain D, L and L' are both optimal solutions, since the way in which the model should generalize to other domains is not considered.

5 Extending Markov Logic

The problem described above essentially arises only because the learning process has no knowledge of fixed marginal distributions or unconditionally independent attributes. While we can modify the learning process to learn weights that are equivalent to the weights we would obtain via a translation from a KBMC model (simply by precomputing the probability distributions of attributes that are subject to a fixed marginal distribution from the training database, using the logarithms of the probabilities as the initial weights of the corresponding unit clauses during learning and ensuring that the weights remain constant throughout the optimization process), this approach requires the MLN to contain only conjunctions (or equivalent formulas) that are fully capable of describing an appropriate factorization of the full-joint, which is a severe limitation.

We now propose an extension of the MLN language that allows us to ensure fixed marginal distributions regardless of the set of formulas. It involves the specification of hard constraints on individual formula probabilities. In our example on the consumption of drinks, we might, for example, specify a constraint such as $P(\mathit{rank}(p, \mathit{Student})) = 0.\overline{3}$. When instantiating a ground Markov network, the probability information can be used to dynamically modify the weight of the corresponding unit clause $F := \mathit{rank}(p, \mathit{Student})$. If the probability that is indicated by the model is originally q and the constraint requires it to be q', then with W_F (see (2)) as the sum of exponentiated sums of weights for possible worlds in which the formula holds (for an arbitrary binding of the variable p), $W_{\neg F}$ as the sum for the remaining worlds and $q = \frac{W_F}{W_F + W_{\neg F}}$ and $q' = \frac{W_F \cdot \lambda}{W_F \cdot \lambda + W_{\neg F}}$, we obtain

$$\lambda = \frac{q'}{q} \cdot \frac{1-q}{1-q'} \tag{7}$$

i.e. we need to add $\log(\lambda)$ to the formula's weight in order to obtain the desired marginal probability.

If the underlying process dictates more than one such constraint on the marginal distributions of certain attributes, we can proceed in a similar fashion. However, specifying just a probability constraint for each corresponding unit clause may not suffice, because the a priori independence of the attributes may need to be modelled explicitly. For example, if the marginal distribution over drink types, too, was to be fixed in all domains, adding a constraint such as $P(\mathit{drinkType}(d, \mathit{Tea})) = 0.5$ would not render ranks and drink types independent. Therefore, if independence is to be modelled, we instead propose to add to the MLN all conjunctions of value statements for all independent attributes[6] in order to indirectly represent the independence by including the corresponding part of the full-joint. In our example, we would add the constraints

$$\begin{aligned}
&P(\mathit{rank}(p, \mathit{Student}) \wedge \mathit{drinkType}(d, \mathit{Tea})) &&= q_1 = q_S \cdot q_T \\
&P(\mathit{rank}(p, \mathit{Student}) \wedge \mathit{drinkType}(d, \mathit{Coffee})) &&= q_2 = q_S \cdot q_C \\
&P(\mathit{rank}(p, \mathit{Professor}) \wedge \mathit{drinkType}(d, \mathit{Tea})) &&= q_3 = q_P \cdot q_T \\
&P(\mathit{rank}(p, \mathit{Professor}) \wedge \mathit{drinkType}(d, \mathit{Coffee})) &&= q_4 = q_P \cdot q_C
\end{aligned}$$

where $\sum_i q_i = 1.0$ (in our above example, $q_S = \frac{1}{3}$, $q_P = \frac{2}{3}$, $q_T = q_C = 0.5$).[7] Naturally, the actual inclusion of these constraints could be handled by the

[6] For conjunctions not already present, we initially assume a zero weight.

[7] One of the four constraints is redundant and may be left out.

learner, and the user would need to specify only which marginal distributions are to remain fixed. All the necessary probabilities can then automatically be computed from the training database.

In general, if m attributes are to have a fixed marginal distribution and the domain of the j-th attribute contains d_j elements, then there are $n := \prod_{j=1}^{m} d_j$ conjunctions for which probability constraints must be specified (we denote these conjunctions by C_i with $i \in \{1, \dots, n\}$). Because these conjunctions are atomic (sub-)events (for a particular binding of the variables), they partition the set of possible worlds $\mathcal{X}_D$ for a concrete domain D, for which the MLN is instantiated, into n corresponding parts, i.e. $\mathcal{X}_D = \biguplus_{i=1}^{n} (\mathcal{X}_D)_{C_i}$. Let the sum of exponentiated sums of weights of possible worlds in the i-th partition be $W_i := W_{C_i}$; the normalizing constant in (1) is thus $Z_D = \sum_{i=1}^{n} W_i$. If $\boldsymbol{q}$ is the vector of conjunction probabilities, then we obtain the scaling factors λ_i with which the weights of the n conjunctions need to be corrected by solving the following linear equation system:

$$\begin{aligned} \frac{W_k \lambda_k}{\sum_{i=1}^{n} W_i \lambda_i} &= q_k \quad \text{for } k \in \{1, \dots, n\} \\ \Rightarrow \quad (W_k - q_k W_k)\lambda_k - \sum_{i=1, i \neq k}^{n} q_k W_i \lambda_i &= 0 \quad \text{for } k \in \{1, \dots, n\} \end{aligned} \tag{8}$$

If the probabilities that are specified in vector $\boldsymbol{q}$ are not contradictory (which the learning process can guarantee), then the solution to the above equation system is unique and well-defined, and we obtain the desired marginals by adding $\log(\lambda_i)$ to the weight of the i-th conjunction C_i.

Since the part of the joint probability distribution that consists exclusively of marginals is fully determined by the above calculations, any existing MLN formulas that contain only atoms that are subject to constraints on marginals cannot provide additional information as far as the probability distribution is concerned. Therefore, they can be removed prior to the above calculations, and the formulas we explicitly require (i.e. the conjunctions C_i), along with their weights ($\log(\lambda_i)$), can be added automatically when instantiating a ground Markov network.

All in all, we can apparently model arbitrary marginal distributions and independencies. Together with a set of conditional distributions, we can thus describe a wide range of probability distributions in relational domains.

In general, we can show that probabilities conditioned on all marginals are unaffected by the dynamic weight modifications described above. Let R be the set of ground atoms for which the marginal distribution is explicitly modelled in an extension to a Markov logic network L, and let $\bar{L}$ be the Markov logic network we obtain from L by applying the constraints for a concrete domain/set of constants C, i.e. by adding the formulas $F_{R=r}$ with weights w_r for $r \in \mathbb{B}^{|R|}$. We thus need to show that $P(F_1 \mid F_2, R = r, M_{\bar{L},C}) = P(F_1 \mid F_2, R = r, M_{L,C})$:

$$P(F_1 \mid F_2, R = r, M_{\bar{L},C}) = \frac{P(F_1 \wedge F_2 \wedge R = r \mid M_{\bar{L},C})}{P(F_2 \wedge R = r \mid M_{\bar{L},C})}$$

$$= \frac{\sum_{x \in \mathcal{X}_{F_1} \cap \mathcal{X}_{F_2} \cap \mathcal{X}_{R=r}} \exp\left(\sum_i w_i \cdot n_i(x) + w_r\right)}{\sum_{x \in \mathcal{X}_{F_2} \cap \mathcal{X}_{R=r}} \exp\left(\sum_i w_i \cdot n_i(x) + w_r\right)}$$

$$= \frac{W_{F_1 \wedge F_2 \wedge R=r} \cdot \exp w_r}{W_{F_2 \wedge R=r} \cdot \exp w_r} = P(F_1 \mid F_2, R=r, M_{L,C}) \qquad (9)$$

Returning to our example from Chap. 4.2, we now apply dynamic modifications (8) to the MLN that was learned, enforcing a fixed marginal on *rank* and *drinkType* by introducing the corresponding constraints but leaving the MLN unchanged otherwise. Looking at Table 2, we observe that the generative stochastic process which we assumed can now clearly be described in the intended way.

	$M_{\bar{L},D_{1,1}}$	$B_{D_{1,1}}$	$M_{\bar{L},D_{1,3}}$	$B_{D_{1,3}}$
$P(\mathit{drinkT}(D_1, \mathit{Tea}))$	50.0%	50.0%	50.0%	50.0%
$P(\mathit{rank}(P, \mathit{Stud}))$	33.3%	33.3%	33.3%	33.3%
$P(\mathit{rank}(P, \mathit{Stud}) \mid \mathit{cons}(P, D_1))$	25.0%	25.0%	25.0%	25.0%
$P(\mathit{rank}(P, \mathit{Stud}) \mid \neg \mathit{cons}(P, D_1))$	37.5%	37.5%	37.5%	37.5%
$P(\mathit{rank}(P, \mathit{Stud}) \mid \mathit{cons}(P, D_1) \wedge \mathit{drinkT}(D_1, \mathit{Tea}))$	50.0%	50.0%	50.0%	50.0%
$P(\mathit{drinkT}(D_1, \mathit{Tea}) \mid \mathit{cons}(P, D_1) \wedge \mathit{rank}(P, \mathit{Prof}))$	33.3%	33.3%	33.3%	33.3%
$P(\mathit{cons}(P, D_1) \mid \mathit{drinkT}(D_1, \mathit{Tea}) \wedge \mathit{rank}(P, \mathit{Prof}))$	25.0%	25.0%	25.0%	25.0%
$P(\mathit{rank}(P, \mathit{Stud}) \mid \mathit{cons}(P, D_1) \wedge \mathit{cons}(P, D_2))$			18.2%	18.2%

Table 2. Inference results: ground Markov networks (with dynamic modifications applied) and Bayesian networks compared.

6 Conclusion

We have shown that Markov logic networks that are learned using standard parameter learning may require additional language constructs if they are to be applicable to arbitrary domains — and not just the single domain that was used for learning — whenever the attributes of related objects are a priori not to be affected by the probability with which they are potentially related to other objects, i.e. whenever there are constraints on marginal probabilities of attribute values. This is in fact a quite common case, for the processes that we intuitively imagine generate worlds sequentially, i.e. there is a certain order in which objects and relations are created (which could be akin to a causal structure). If objects and the attributes that belong to them are created simultaneously, then there is usually some attribute whose values are subject to a fixed marginal. A typical example of such a case is an attribute such as a person's *gender* or *rank* in our above example. Our extension to the MLN language solves this problem by introducing hard constraints on formula probabilities that are used to dynamically modify the weights of the respective formulas whenever a model is instantiated for a concrete domain.

The concrete constraints that are necessary to handle domain shifts need not be specified by a user but can instead be learned from a training database, given that information on unconditional independencies is provided. If we learn with more than one training database, then we can even attempt to deduce these independencies and alleviate the user from specifying any additional information at all.

Nevertheless, there are further problems that may still prevent MLNs from being practically applicable if the goal is to model probability distributions in relational domains. The use of exact parameter learning algorithms based on log-likelihood is intractable, and we found the approximate learning methods involving pseudo-likelihood to be too inexact to be acceptable — at least for some domains. Solving the problems that remain in this regard will be the subject of our future investigations.

Acknowledgements

We would like to thank Pedro Domingos for fruitful email discussions.

References

1. Nilsson, N.J.: Probabilistic Logic. Artif. Intell. **28**(1) (1986) 71–87
2. Halpern, J.Y.: An analysis of first-order logics of probability. In: Proceedings of IJCAI-89, 11th International Joint Conference on Artificial Intelligence, Detroit, US (1989) 1375–1381
3. Bacchus, F.: Representing and Reasoning with Probabilistic Knowledge. MIT Press, Cambridge, Massachusetts (1990)
4. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: IJCAI. (1999) 1300–1309
5. Milch, B., Marthi, B., Russell, S.J., Sontag, D., Ong, D.L., Kolobov, A.: BLOG: Probabilistic Models with Unknown Objects. In: IJCAI. (2005) 1352–1359
6. Kersting, K., Raedt, L.D.: Bayesian Logic Programming: Theory and Tool. In Getoor, L., Taskar, B., eds.: An Introduction to Statistical Relational Learning. MIT Press (2005)
7. Neville, J., Jensen, D.: Dependency networks for relational data. In: ICDM '04: Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04), Washington, DC, USA, IEEE Computer Society (2004) 170–177
8. Richardson, M., Domingos, P.: Markov Logic Networks. Mach. Learn. **62**(1-2) (2006) 107–136
9. Domingos, P., Richardson, M.: Markov Logic: A Unifying Framework for Statistical Relational Learning. In: Proceedings of the ICML 2004 Workshop on Statistical Relational Learning and its Connections to Other Fields. (2004) 49–54
10. Kok, S., Singla, P., Richardson, M., Domingos, P.: The Alchemy system for statistical relational AI. `http://alchemy.cs.washington.edu/` (2004)
11. Domingos, P.: What's Missing in AI: The Interface Layer. In Cohen, P., ed.: Artificial Intelligence: The First Hundred Years. AAAI Press (2006)
12. Beetz, M., Gedikli, S., Bandouch, J., von Hoyningen-Huene, N., Kirchlechner, B., Perzylo, A.: Visually Tracking Football Games Based on TV Broadcasts. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI). (2007)