Induction of Interpretable Possibilistic Logic Theories from Relational Data

Ondřej Kuželka
Cardiff University, UK
KuzelkaO@cardiff.ac.uk

Jesse Davis
KU Leuven, Belgium
jesse.davis@cs.kuleuven.be

Steven Schockaert
Cardiff University, UK
SchockaertS1@cardiff.ac.uk

Abstract

The field of Statistical Relational Learning (SRL) is concerned with learning probabilistic models from relational data. Learned SRL models are typically represented using some kind of weighted logical formulas, which make them considerably more interpretable than those obtained by e.g. neural networks. In practice, however, these models are often still difficult to interpret correctly, as they can contain many formulas that interact in non-trivial ways and weights do not always have an intuitive meaning. To address this, we propose a new SRL method which uses possibilistic logic to encode relational models. Learned models are then essentially stratified classical theories, which explicitly encode what can be derived with a given level of certainty. Compared to Markov Logic Networks (MLNs), our method is faster and produces considerably more interpretable models.

1 Introduction

The aim of Statistical Relational Learning (SRL) is to learn models that can make predictions from sets of relational facts. Many popular SRL frameworks, such as Markov Logic Networks (MLNs [Richardson and Domingos, 2006]), probabilistic soft logic [Bach *et al.*, 2015], and various forms of probabilistic logic programs [De Raedt and Kimmig, 2015], use weighted logical formulas to encode the statistical regularities that have been observed in training data. Despite the use of logical formulas, learned models are often surprisingly hard to interpret. Consider, for instance, the following fragment of an MLN that was learned from the UWCSE dataset[1]:

$$\begin{aligned} -5.11 &: \textit{student}(X) \\ 5.11 &: \textit{professor}(X) \\ -12.01 &: \neg\textit{student}(X) \vee \textit{faculty-adj}(X) \vee \textit{professor}(X) \\ &\quad \vee \textit{faculty}(X) \vee \textit{faculty-aff}(X) \end{aligned}$$

The first two formulas intuitively mean that, all things being equal, a given individual is unlikely to be a student and likely to be a professor. However, if a given individual is a professor, then the third formula becomes satisfied. Due to the large negative weight of this formula, it turns out that being a student is actually considered to be more likely than being a professor. In practice, there can be many formulas that interact in such a way, making it hard to predict the behavior of the MLN by inspecting the weighted formulas. This limits the usefulness of MLNs for explorative data analysis, and makes it almost impossible for domain experts to manually tweak a learned MLN.

Probabilistic logic programming (PLP) languages attach probabilities to either rules or facts. For programs with neither *negation as failure* nor cyclic dependencies, the individual weights have a clearer meaning than in MLNs, as they are directly related to probabilities. However, using negation as failure and cyclic dependencies often leads to PLPs that can be counter-intuitive (e.g. see Section 8.3 in [Buchman and Poole, 2017]). Yet, even for propositional PLPs, excluding negation as failure limits the expressivity of the language [Buchman and Poole, 2017]. While the interpretability of AI systems is becoming increasingly important [Baehrens *et al.*, 2010; Sanchez *et al.*, 2015; Ribeiro *et al.*, 2016], we are not aware of any existing methods for learning joint relational models that focus on interpretability.

Possibilistic logic [Lang *et al.*, 1991] also uses weighted formulas, usually written as (α, λ) with α a classical formula and $\lambda \in [0, 1]$ a certainty weight. As suggested in [Kuželka *et al.*, 2016], we can use *possibilistic* logic to encode *probability* distributions. The formula (α, λ) then expresses the constraint that the probability of any world violating α can be at most $1 - \lambda$. Because of this constraint based semantics, formulas can safely be interpreted in isolation from the rest of the theory, which we believe is crucial for interpretability. The method proposed in [Kuželka *et al.*, 2016] derives a possibilistic logic theory from a density estimation tree [Ram and Gray, 2011], which is in turn learned from a set of training examples. Compared to Markov Random Fields (MRFs), the possibilistic logic theories resulted in a higher accuracy for Maximum A Posteriori (MAP) queries with small evidence sets, while MRFs were more accurate for larger evidence sets. Essentially, inference from possibilistic logic theories captures the conclusions that we can obtain by applying a form of commonsense reasoning (see [Kuželka *et al.*, 2015] for a theoretical justification of this view). In the presence of

[1] https://alchemy.cs.washington.edu/data/uw-cse/

large amounts of evidence, however, MRFs can make predictions even when there is no obvious "default knowledge" that applies, by aggregating large amounts of individually weak and/or conflicting pieces of evidence.

In this paper we introduce a method for learning possibilistic logic theories from relational data. In principle, such theories could be learned by "lifting" the approach from [Kuželka *et al.*, 2016] to the relational setting. However, this technique relies on identifying a set of formulas $\alpha_1, ..., \alpha_n$ which are mutually exclusive and jointly exhaustive (corresponding to the branches of the density estimation tree). In a relational setting, this essentially requires us to enumerate isomorphism classes, of which there are typically exponentially many. As a result, the possibilistic logic theories we obtain quickly become prohibitively large (even though these theories could subsequently be pruned). Therefore, we follow a different strategy in this paper. To obtain suitable formulas, we first learn a set of hard constraints. These hard constraints allow us to generate non-trivial negative examples, which together with the positive examples obtained from the training data, allow us to learn a set of Horn rules that describe how the different predicates relate to each other. The restriction to Horn rules increases the interpretability of the learned theories, and leads to theories that are optimized for predicting positive literals, which is usually what is needed in applications. Note that the hard constraints are not restricted to Horn rules, which means that our theories can still be used to predict negative literals. In the last step, we use a form of relational model counting to associate a weight with each of the learned Horn rules.

To the best of our knowledge, our approach is the first that represents joint relational models in such a way that each weighted formula can be interpreted in isolation. The most closely related work is [Serrurier and Prade, 2007], where first-order possibilistic logic theories were learned in an Inductive Logic Programming (ILP) setting. However, the learned theories from that work are aimed at predicting a single target predicate. Moreover, their approach was based on a non-standard semantics for possibilistic logic, in which formulas cannot be interpreted in isolation. Finally, their approach is purely qualitative, i.e. formulas are ranked but are not given weights with a probabilistic interpretation.

We also provide an online appendix to this paper[2] with additional illustrating examples and experimental results.

2 Preliminaries

Throughout the paper, we consider a function-free first-order logic language $\mathcal{L}$, which is built from a set of constants *Const*, variables *Var* and predicates $\mathit{Rel} = \bigcup_i \mathit{Rel}_i$, where Rel_i contains the predicates of arity i. For $a_1, ..., a_k \in \mathit{Const} \cup \mathit{Var}$ and $R \in \mathit{Rel}_k$, we call $R(a_1, ..., a_k)$ an atom. If $a_1, .., a_k \in \mathit{Const}$, this atom is called ground. A literal is an atom or the negation of an atom, and a clause is a disjunction of literals. The formula α_0 is called a grounding of α if α_0 can be obtained from α by substituting each variable by a particular constant from *Const*. A formula is called closed if all variables are bound by a quantifier. A possible world ω is defined as a set of ground atoms. The satisfaction relation $\models$ is defined in the usual way.

A Markov logic network (MLN) [Richardson and Domingos, 2006] is a set of weighted formulas $w : F$, with $w \in \mathbb{R}$ and F a function-free and quantifier-free first-order formula. The semantics are defined w.r.t. the groundings of the first-order formulas, relative to some finite set of constants. An MLN is seen as a template that defines an MRF. Specifically, an MLN $\mathcal{M}$ induces the following probability distribution on the set of possible worlds ω:

$$p_{\mathcal{M}}(\omega) = \frac{1}{Z}\exp\left(\sum_{w:F\in\mathcal{M}} w n_F(\omega)\right) \tag{1}$$

where $n_F(x)$ is the number of groundings of F that are satisfied in ω, and Z is a normalization constant to ensure that $p_{\mathcal{M}}$ is a probability distribution. A key inference task for MLNs is computing the Maximum A Posteriori (MAP) consequences, i.e. determining which ground atoms are true in the most probable models of a given set of ground atoms. Formally, $(\mathcal{M}, E) \models \alpha$, for E a set of ground atoms and α a ground atom, iff $\forall\omega\,.\,(p_{\mathcal{M}}(\omega) = \max_{\omega'} p_{\mathcal{M}}(\omega')) \Rightarrow (\omega \models \alpha)$.

A possibilistic logic theory [Lang *et al.*, 1991] is a set of weighted formulas (α, λ) with α a propositional formula and $\lambda \in [0, 1]$. A possibilistic logic theory Θ induces a mapping $\pi : \Omega \to [0, 1]$, with Ω the set of propositional interpretations, which is defined for $\omega \in \Omega$ as:

$$\pi_\Theta(\omega) = \min\{1 - \lambda \,|\, (\alpha, \lambda) \in \Omega, \omega \not\models \alpha\} \tag{2}$$

The distribution π_Θ is called a possibility distribution. The possibilistic logic theories we consider will be constructed such that $\sum_\omega \pi_\Theta(\omega) = 1$, in which case π_Θ can be interpreted as a probability distribution. There is a common inconsistency-tolerant inference relation in possibilistic logic, which is actually the direct counterpart of MAP inference. Specifically, for E a set of propositional formulas and α a propositional formula, we write $(\Theta, E) \models \alpha$ if $\forall\omega\,.\,(\pi_\Theta(\omega) = \max_{\omega'} \pi_\Theta(\omega')) \Rightarrow (\omega \models \alpha)$. Interestingly, the formulas α which are entailed in this sense can easily be determined syntactically. In particular, for $\mu \in [0, 1]$ let $\Theta_\mu = \{\alpha \,|\, (\alpha, \lambda) \in \Theta, \lambda \geq \mu\}$. Let μ_0 be the smallest threshold for which $\Theta_{\mu_0} \cup E$ is consistent. Then $(\Theta, E) \models \alpha$ iff $\Theta_{\mu_0} \cup E \models \alpha$. Hence inference in possibilistic logic can straightforwardly be implemented using a SAT solver.

In this paper we will learn possibilistic logic theories with first-order formulas instead of propositional formulas. Like MLNs, these first-order possibilistic logic theories should simply be seen as templates for normal (propositional) possibilistic logic theories that are obtained by replacing each weighted first-order formula (α, λ) by the formulas $(\alpha_1, \lambda), ..., (\alpha_k, \lambda)$, with $\alpha_1, ..., \alpha_k$ the groundings of α. It is easy to see that when $p_{\mathcal{M}} = \pi_\Theta$ for an MLN $\mathcal{M}$ and first-order possibilistic logic theory Θ, it holds that $(\mathcal{M}, E) \models \alpha$ iff $(\Theta, E) \models \alpha$. [Kuželka *et al.*, 2015] demonstrated how to construct a possibilistic logic theory Θ from a given MLN $\mathcal{M}$, such that $p_{\mathcal{M}} = \pi_\Theta$. However, the resulting possibilistic logic theory is exponential in size. In practice, the possibilistic logic theories we learn from data can thus only approximate what could be encoded in an MLN. This makes

[2] http://arxiv.org/abs/1705.07095

MLNs potentially better equipped to make predictions from large amounts of evidence, while making possibilistic logic less prone to making spurious predictions in situations where the amount of evidence is more limited.

3 Relational Marginals

In the context of SRL, we are typically given a large set of ground atoms $\mathcal{A}$ as training data. This set essentially corresponds to a single example of a relational structure. Intuitively, we want to learn a probability distribution over such relational structures, but we clearly cannot estimate such a distribution from one example. The solution we propose is to construct a large number of training examples by sampling small fragments of this global relational structure, and then estimating a probability distribution over these fragments[3]. We will refer to $\Upsilon = (\mathcal{A}, \mathcal{C})$, with $\mathcal{C}$ the set of constants appearing in $\mathcal{A}$, as an example. We now explain how we can obtain a collection of "local" training examples, which will correspond to (isomorphism classes of) fragments of this "global" example.

Definition 1. *A (global) example is a pair $(\mathcal{A}, \mathcal{C})$, with $\mathcal{C}$ a set of constants and $\mathcal{A}$ a set of ground atoms which only use constants from $\mathcal{C}$. Let $\Upsilon = (\mathcal{A}, \mathcal{C})$ be an example and $\mathcal{S} \subseteq \mathcal{C}$. The fragment $\Upsilon\langle S\rangle = (\mathcal{B}, \mathcal{S})$ is defined as the restriction of Υ to the constants in $\mathcal{S}$, i.e. $\mathcal{B}$ is the set of all atoms from $\mathcal{A}$ which only contain constants from $\mathcal{S}$.*

Intuitively, we can repeatedly sample subsets $\mathcal{S}$ and then consider each $\Upsilon\langle S\rangle$ as a training example. However, the constants appearing in each of these fragments will be different, hence to enable generalization we need to consider their isomorphism classes.

Definition 2 (Isomorphism)**.** *Two examples $\Upsilon_1 = (\mathcal{A}_1, \mathcal{C}_1)$ and $\Upsilon_2 = (\mathcal{A}_2, \mathcal{C}_2)$ are isomorphic, denoted as $\Upsilon_1 \approx \Upsilon_2$, if there exists a bijection $\sigma : \mathcal{C}_1 \to \mathcal{C}_2$ such that $\sigma(\mathcal{A}_1) = \mathcal{A}_2$, where σ is extended to ground atoms in the usual way.*

Definition 3 (Local example)**.** *Let $k \in \mathbb{N}$ and let $\mathcal{L}_k$ be the language which contains the same predicates and variables as $\mathcal{L}$ but only constants from the set $\{1, 2, ..., k\}$. A local example of width k is a pair $\omega = (\mathcal{A}, \{1, ..., k\})$, where $\mathcal{A}$ is a set of ground atoms from the language $\mathcal{L}_k$. For an example $\Upsilon = (\mathcal{A}, \mathcal{C})$ and $\mathcal{S} \subseteq \mathcal{C}$, we write $\Upsilon[\mathcal{S}]$ for the set of all local examples of width $|\mathcal{S}|$ which are isomorphic to $\Upsilon\langle\mathcal{S}\rangle$.*

To distinguish local examples from global examples, we will denote them using lower case Greek letters such as ω instead of upper case letters such as Υ.

Example 1. *For $\Upsilon = (\{fr(alice, bob), fr(bob, alice), fr(bob, eve), fr(eve, bob), sm(alice)\}, \{alice, bob, eve\})$ we have:*

$$\Upsilon\langle\{alice, bob\}\rangle = (\{fr(alice, bob), fr(bob, alice), sm(alice)\}, \{alice, bob\})$$

$$\Upsilon[\{alice, bob\}] = \{(\{fr(1,2), fr(2,1), sm(1)\}, \{1,2\}) \; (\{fr(2,1), fr(1,2), sm(2)\}, \{1,2\})\}$$

We can now naturally define a probability distribution over local examples of width k.

[3] Additional examples illustrating the concepts introduced in this section are described in the online appendix.

Definition 4 (Relational marginal distribution)**.** *Let $\Upsilon = (\mathcal{A}, \mathcal{C})$ be an example and $k \in \mathbb{N}$. The relational marginal distribution of Υ of width k is a distribution $P_{\Upsilon,k}$ over local examples, where $P_{\Upsilon,k}(\omega)$ is defined as the probability that ω is sampled by the following process:*

1. *Uniformly sample a subset $\mathcal{S}$ of k constants from $\mathcal{C}$.*
2. *Uniformly sample a local example ω from the set $\Upsilon[\mathcal{S}]$.*

For a closed formula α, we also define:

$$P_{\Upsilon,k}(\alpha) = \sum_{\omega:\omega\models\alpha} P_{\Upsilon,k}(\omega)$$

In the following, constant-free existentially-quantified conjunctions of atoms will play an important role, as they are the syntactic counterpart of the isomorphism classes $\Upsilon[\mathcal{S}]$. For such a conjunction α, it holds that $P_{\Upsilon,k}(\alpha)$ is equal to the probability that a randomly sampled set $\mathcal{S}$ of k constants satisfies $\Upsilon\langle\mathcal{S}\rangle \models \alpha$. In this sense, relational marginal distributions faithfully model the probabilities of isomorphism classes of local examples. Naturally, other probability distributions on local examples might also faithfully model the probabilities of these isomorphism classes, but it is easy to see that relational marginal distributions have the highest entropy among such models.

The idea of relational marginals is similar to the random selection semantics used in [Schulte *et al.*, 2014], but the difference is that for relational marginals, we restrict the sample sets to have fixed cardinality and then standardize them as local examples. This allows us to construct a standard probability distribution over local examples.

4 Possibilistic Logic Encoding of Relational Marginals

In this section we describe how relational marginals can be encoded in possibilistic logic. As we show first, in principle we can use a direct generalization of the approach from [Kuželka *et al.*, 2016], by taking advantage of the fact that each isomorphism class $\Upsilon[\mathcal{S}]$ of local examples corresponds to a constant-free existentially-quantified conjunction of atoms. For an example Υ, let $g_k(\Upsilon) = \{\alpha_1, ..., \alpha_n\}$ be a set that contains one such formula for each isomorphism class of local examples of width k.

Definition 5 (Possibilistic encoding of relational marginals)**.** *Let Υ be an example and let $k \in \mathbb{N}$. The possibilistic logic theory corresponding to $P_{\Upsilon,k}$ is defined as*

$$\Theta_{\Upsilon,k} = \left\{ \left(\neg\alpha, 1 - \frac{1}{c(\alpha)} P_{\Upsilon,k}(\alpha) \right) \middle| \, \alpha \in g_k(\Upsilon) \right\}$$

where $c(\alpha)$ is the cardinality of the isomorphism class represented by α.

Proposition 1. *Let Υ be an example, $k \in \mathbb{N}$, and ω a local example of width k. It holds that $P_{\Upsilon,k}(\omega) = \pi(\omega)$ where $\pi(.)$ is the possibility distribution associated with $\Theta_{\Upsilon,k}$.*

Proof. Let ω be a local example of width k. By definition, $g_k(\Upsilon)$ contains a unique formula α^* such that $\omega \models \alpha^*$, since the formulas in $g_k(\Upsilon)$ define a partition of local examples

into isomorphism classes. Accordingly, $\neg\alpha^*$ is the unique formula appearing in $\Theta_{\Upsilon,k}$ which is not satisfied by ω. By (2), we therefore have $\pi(\omega) = 1 - (1 - P_{\Upsilon,k}(\alpha^*)/c(\alpha^*) = P_{\Upsilon,k}(\alpha^*)/c(\alpha^*) = P_{\Upsilon,k}(\omega)$, where the last equality hold because all local examples from the same partition class have the same probability in a relational marginal distribution. □

The number of isomorphism classes typically grows very quickly with increasing k, so the exact transformation from Definition 5 can only be used for very simple problem domains. In practice, representing the relational marginal distribution exactly is typically not feasible. An exact representation would moreover not necessarily generalize well to previously unseen data. Therefore, for the remainder of this paper, we will focus on learning approximate possibilistic logic representations of relational marginal distributions.

Specifically, our aim is to construct a possibilistic logic theory $\Theta = \{(\alpha_1, \lambda_1), ..., (\alpha_n, \lambda_n)\}$ such that for the associated possibility distribution π it holds that $\pi(\omega)$ is approximately equal to $P_{\Upsilon,k}(\omega)$. This problem can be decomposed in two steps. The first step is structure learning, i.e. choosing suitable formulas $\alpha_1, ..., \alpha_n$. In this paper, we will only consider constant-free and quantifier-free formulas. However, recall that first-order possibilistic logic theories are seen as templates for propositional theories, which means that all variables in the formulas $\alpha_1, ..., \alpha_n$ are implicitly universally quantified. The second step is weight learning. In this step, we aim to find the weights $\lambda_1, ..., \lambda_n$ for which π, seen as a probability distribution, maximizes the likelihood of a set of training examples. Note that if $\lambda_1 \leq ... \leq \lambda_n$ we can assume w.l.o.g. that $\alpha_1 = \bot$. We need to include such a formula α_1 to encode the probability of the most probable worlds (which is then given by $1 - \lambda_1$).

As the transformation from Definition 5 illustrates, weight learning becomes very simple when using mutually exclusive formulas. However, using mutually exclusive formulas is not desirable, as such formulas quickly become very large[4], which also makes the resulting theories difficult to interpret. Therefore, in practice, we will rely on greedy methods for weight learning. These will be discussed in Section 6.

5 Structure Learning

In this section, we propose a method to learn Horn rules that can be used to predict all predicates from Υ. Using Horn rules makes the resulting possibilistic logic theories more interpretable, and allows us to optimize them for predicting atoms, which is what is usually required. Learning Horn rules using methods based on inductive logic programming [Muggleton and De Raedt, 1994] typically requires both positive and negative training examples. In Subsection 5.1 we explain how to construct examples and then discuss our method for learning Horn rules in Subsection 5.2.

5.1 Constructing Training Examples

Constructing positive examples for a given predicate P is straightforward: we can simply take all, or a subsample, of the true P-atoms from Υ, Typically, there are significantly more negative examples than positive ones; e.g. in a typical social network there are many more examples of non-friends than of friends. Simply subsampling the negative examples is unlikely to be effective, as most of the resulting negative examples might be uninteresting, in the sense that they can be explained by some simple hard rules that hold for the domain. Hence, we first learn a set of such hard rules, and then only consider negative examples that are consistent with them.

We are interested in hard rules that are universally quantified, constant-free clauses with no counterexamples in Υ. We find such clauses by exhaustively constructing all clauses (modulo isomorphism) containing at most t literals and at most k variables, where k is the width of the relational marginal distribution and t is a parameter of the method. For each clause, we check whether $\Upsilon \not\models \neg\alpha$ holds with a CSP solver. We store each such clause in a list if the list does not contain another clause that subsumes it. Because learning hard rules that only contain unary literals is typically easier than learning more general rules, we use a higher size limit $t' > t$ for these rules.

Let Δ be the set of discovered hard rules, and $\Upsilon = (\mathcal{A}, \mathcal{C})$ be the global example. To select negative training examples, we reject all samples a for which $\bigwedge \mathcal{A} \wedge a \wedge \Delta$ does not have a model when grounded[5] over the set of constants $\mathcal{C}$. The result is a subsample of non-trivial negative examples. In addition, this process allows us to estimate the total number of non-trivial negative examples, which we use to compute the weight of the negative examples when estimating the accuracies of the Horn rules.

5.2 Learning Horn Rules

To find Horn rules, we employ a beam search method, which relies on two parameters: the size of the beam b and the maximum number of literals in the body of a rule l. As before, k is the width of the local examples. For a given target predicate P of arity m, we initialize the list of candidate rules with the rule $P(X_1, ..., X_m) \leftarrow \top$. In each iteration of the search, we construct all possible single-literal extensions of each rule in the beam such that the constraints on the number literals and variables are not violated. From these candidate rules, we select a set of non-isomorphic rules and evaluate their accuracy on the (weighted) sets of positive and negative examples. The algorithm then selects the b most accurate rules to serve as the candidate rules for the next iteration. The algorithm terminates when no new candidate rules can be generated without violating the constraints on the number of literals and variables and returns the best found rule. This beam search method is repeated several times for each predicate P. Most rules found during one run of the beam search typically entail similar sets of examples. To promote diversity within each run of beam search, we discard rules that are subsumed by previously found rules.

We employ several well-known techniques to speed up the search. First, instead of checking isomorphism for every pair of candidate rules, we efficiently select non-isomorphic rules by hashing each one using a straightforward generalization

[4] One exception is when $k = 1$, which corresponds to the propositional case, where density estimation trees can be used, as was proposed in [Kuželka *et al.*, 2016].

[5] We use incremental grounding for efficiency.

of the Weisfeiler-Lehman labeling procedure [Weisfeiler and Lehman, 1968]. Then, we only check if two rules are isomorphic if they have the same hash value, and if so one of them is removed. Second, the algorithm maintains a set *Forbidden* of minimal rules which entailed zero positive examples in the previous iterations of the beam search. Before evaluating new candidate rules, the algorithm discards candidate rules which are subsumed by a rule from the set *Forbidden*. Third, to reduce the negative plateau effect, known from relational learning [Alphonse and Osmani, 2008], we add to every constructed rule a literal *AllDiff*$(V_1, \ldots, V_k)$, which is true iff all variables in its argument are mapped to different terms. This also improves the interpretability of the rules.

6 Weight Learning

Let us first assume that an ordering of the formulas $(\alpha_1 = \bot, \alpha_2 \ldots, \alpha_n)$ is given, and we want to learn weights $\lambda_1 \leq ... \leq \lambda_n$ which maximize the likelihood of a set of local examples $\mathcal{E}$ that have been sampled from $P_{\Upsilon,k}$. These weights can be found by solving the following optimization problem:

- Variables: $\lambda'_1, \lambda'_2, \ldots, \lambda'_n$.
- Maximize: $\prod_{\omega \in E} P(\omega) = \prod_{i=1}^{n} (1 - \lambda'_i)^{|\mathcal{E}_{i+1}| - |\mathcal{E}_i|}$ where $\mathcal{E}_i = \{\omega \in \mathcal{E} | \omega \models \alpha_i \wedge \cdots \wedge \alpha_n\}$.
- Subject to:

$$\lambda'_1 \leq \lambda'_2 \leq \cdots \leq \lambda'_n \tag{3}$$

$$\sum_{i=1}^{k} (1 - \lambda'_i) \cdot (|M_{i+1}| - |M_i|) = 1 \tag{4}$$

  where $M_i = \{\omega | \omega \models \alpha_i \wedge \cdots \wedge \alpha_n\}$ and (4) forces probabilities of all possible worlds to sum to 1.

This optimization problem can be converted to a geometric programming problem, similar to the geometric programming encoding proposed in [Kuželka *et al.*, 2016]. Note that geometric programming problems can be converted to convex programming problems by a change of variables, and can thus be solved using standard convex programming methods [Boyd *et al.*, 2007].

We can think of $\mathcal{E}$ as an IID sample from the set of local examples in the multi-set $\{\omega | \omega \in \Upsilon[\mathcal{S}], \mathcal{S} \subseteq \mathcal{C}, |\mathcal{S}| = k\}$, where $\Upsilon = (\mathcal{A}, \mathcal{C})$ is the given global example. However, assuming all α_i are constant-free, it is easy to check that we will get the same values (in expectation) of the parameters $|\mathcal{E}_i|$ if we instead use the set $\{\omega | \omega = \Upsilon\langle\mathcal{S}\rangle, \mathcal{S} \subseteq \mathcal{C}, |\mathcal{S}| = k\}$. A detailed description of how we can efficiently estimate the parameters $|\mathcal{E}_i|$ and $|M_i|$ is provided in the online appendix[2].

Computing the parameters $|\mathcal{E}_i|$ and $|M_i|$ needed for weight learning is difficult (#P-hard), so the algorithm uses a greedy approach to search for the best ordering of the formulas. It starts with a possibilistic logic theory containing only the learned hard rules. It iteratively tries to add, at each possible positions, one rule α from the set of candidate rules found by the structure learning algorithm. If adding the rule α increases the likelihood score, we keep α in the theory, at the position that yielded the best improvement. This approach permits caching and reusing many of the parameter computations (i.e. the computed parameters for many cuts of the stratified theory will be the same for many iterations of the algorithm). During the learning process, the algorithm simplifies the constructed theories (using a relational SAT solver). It removes rules which are implied by other rules in the theory that have higher weights, and it also removes redundant literals from the individual rules.

7 Experiments

We compare our approach's learned models to learned MLNs for various MAP inference tasks. We learned MLNs using the default structure learner in the Alchemy package [Kok and Domingos, 2005].[6] For the MLNs, we used RockIt [Noessner *et al.*, 2013] to perform MAP inference.

7.1 Methodology

Our learning algorithm is implemented in Java and uses the SAT4j library [Berre and Parrain, 2010]. Cryptominisat [Soos, 2010] is used for our implementation of relational version of the model counter [Chakraborty *et al.*, 2016]. It uses the JOptimizer package to solve the geometric programming problems[7] needed for the maximum likelihood estimation.

We use two standard SRL datasets: UWCSE and Yeast-Proteins.The UWCSE dataset described relations among students, professors, papers, subjects, terms and projects in the CS department of the University of Washington. This dataset contains among other the following relations (predicates) *AdvisedBy/2*, *TempAdvisedBy/2*, *Publication/2*, *TaughtBy/3*, *TA/3*, *Student/1*, *Professor/1*, *PostQuals/1*. This dataset is split into five groups: AI, language, theory, graphics, and systems. We use AI, language and theory as a training set and graphics and systems as a test set. The Yeast-Proteins dataset contains proteins and the relations among them. We use a version in which the interaction relation is symmetric. This dataset contains the following relations: *Interacts/2*, *Enzyme/2*, *Complex/2*, *ProteinClass/2*, *Function/2*, *Phenotype* and *Location/2*. We randomly divide the constants (entities) in this dataset into two disjoint sets of equal size. The training set consists of atoms containing only the constants from the first set and the test set contains only the constants from the second set. This ensures that no information leaks from the training set into the test set.

We evaluate the performance of the learned models as follows. For each $k = 1, \ldots, k_{max}$, we sample a set of evidence literals from the test set. We then predict the MAP state by each of the learned models and compute the Hamming error, which measures the size of the symmetric difference of the predicted MAP world and the set of the literals in the test set. We then report the cumulative differences between the errors of the models, as this clearly highlights the overall trends.

7.2 Results

The possibilistic logic theory learned for the Yeast-Proteins dataset is shown in Table 1. The rules seem to encode meaningful relations that hold in the dataset. For instance, if a protein A is contained in a complex C, another protein D is in C

[6] http://alchemy.cs.washington.edu/

[7] http://www.joptimizer.com

Table 1: The possibilistic logic theory learned in the Yeast-Proteins dataset, not showing the hard rules and actual weights (but note that $\lambda_\perp < \lambda_1 < ... < \lambda_5$). All rules are implicitly constrained by *AllDiff* constraints.

... (112 hard constraints not shown here)
$(\mathit{Complex}(A,B) \leftarrow \mathit{ProteinClass}(A,C) \wedge \mathit{Interaction}(D,A) \wedge \mathit{Complex}(D,B) \wedge \mathit{ProteinClass}(D,C), \lambda_5)$
$(\mathit{Phenotype}(A,B) \leftarrow \mathit{Interaction}(C,A) \wedge \mathit{ProteinClass}(A,D) \wedge \mathit{Phenotype}(C,B) \wedge \mathit{ProteinClass}(C,D), \lambda_4)$
$(\mathit{ProteinClass}(A,B) \leftarrow \mathit{ProteinClass}(D,B) \wedge \mathit{Complex}(A,C) \wedge \mathit{Complex}(D,C), \lambda_3)$
$(\mathit{Enzyme}(A,B) \leftarrow \mathit{ProteinClass}(A,C) \wedge \mathit{Interaction}(A,D) \wedge \mathit{Enzyme}(D,B) \wedge \mathit{ProteinClass}(D,C), \lambda_2)$
$(\mathit{Location}(A,B) \leftarrow \mathit{Location}(D,B) \wedge \mathit{Complex}(A,C) \wedge \mathit{Complex}(D,C), \lambda_1)$
$(\perp, \lambda_\perp)$

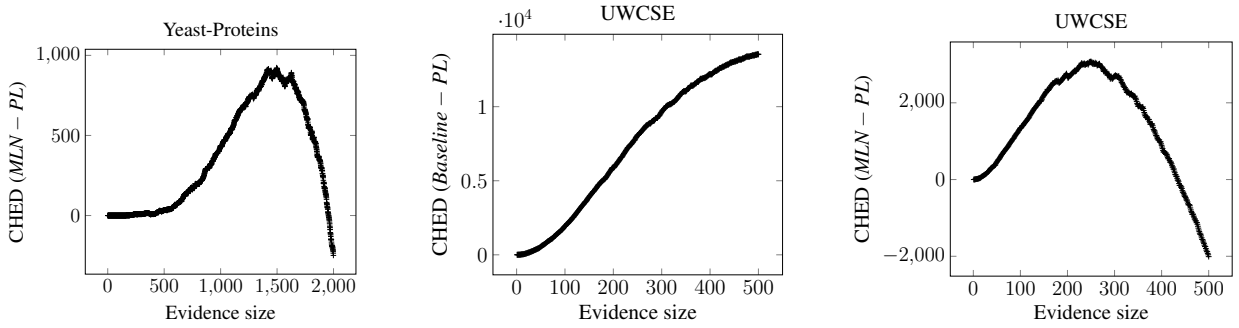

Figure 1: Cumulative Hamming error differencs (CHED) of the "all false" baseline, the learned MLNs, and the learned possibilistic logic theories (PL). Positive numbers indicate that our approach outperforms the indicated reference model.

as well and D is at location B then A is also at that location. The learned MLN, on the other hand, only contained rules that model the prior probabilities of the individual predicates and one additional rule that expresses the symmetry of the *Interaction* relation. Hence, the only type of prediction made by this MLN consists in computing the symmetric closure of the interaction literals, which is why we do not show any separate baseline prediction for this dataset. The possibilistic logic theory has lower Hamming errors for evidence sets up to around 1500 literals (see Figure 1, left panel), which can be seen from the fact that the cumulative difference is increasing over this range. For larger evidence sets, the possibilistic theory intuitively predicts "too much", resulting in a higher Hamming error than the MLN predictions.

The theory which was learned for the UWCSE dataset is larger, and is therefore shown in the appendix, where we also show the corresponding learned MLN. The possibilistic logic theory again contains rules which are intuitive, capturing meaningful relations for this domain. The formulas in the MLN are much harder to interpret. As shown in Figure 1, the possibilistic logic theory again reaches smaller Hamming errors than the learned MLN for small evidence sets, in this case for evidence sets of up to about 250 literals (right panel). It is always better than the baseline which predicts everything not in the evidence as false (middle panel).

Inference in the possibilistic logic theories is, on average, substantially faster than MAP inference in the MLNs (using RockIt). For UWCSE, the speed-up was between one and two orders of magnitude. The possibilistic logic prediction only requires us to solve a logarithmic number of SAT queries, whereas computing MLN MAP predictions requires solving a weighted MAX-SAT problem.

8 Conclusions

We have proposed a method for learning relational possibilistic logic theories. These theories are seen as templates for constructing "ground" (i.e. standard propositional) possibilistic logic theories, similar to how Markov logic networks can be seen as templates for constructing Markov random fields. In particular, as in standard possibilistic logic, each weighted formula has an intuitive interpretation as a constraint on the probability distribution that is being modelled. To formally describe what this probability distribution represents, we have introduced the notion of a relational marginal distribution, which we can intuitively think of as a probability distribution over fixed-sized fragments of a given relational structure. We learn the clauses in the theories using a standard ILP strategy and weights of the clauses using geometric programming.

The main design consideration of our method was to learn interpretable theories. However, as our experimental results have revealed, our method also leads to more accurate MAP predictions than Markov Logic Networks (MLNs) for small to moderately sized evidence sets. For larger evidence sets, MLNs lead to more accurate predictions, which is intuitively due to the fact that they are better equipped to aggregate large amounts of individually weak pieces of evidence. Inference in possibilistic logic is also considerably faster than methods for computing MAP queries from MLNs.

Acknowledgments

This work was supported by a Leverhulme Trust grant (RPG-2014-164) and ERC Starting Grant 637277. JD is partially supported by the KU Leuven Research Fund (C22/15/015), and FWO-Vlaanderen (G.0356.12, SBO-150033).

References

[Alphonse and Osmani, 2008] Erick Alphonse and Aomar Osmani. A model to study phase transition and plateaus in relational learning. In *Proceedings of the 18th International Conference on Inductive Logic Programming*, pages 6–23, 2008.

[Bach *et al.*, 2015] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *Arxiv preprint*, arXiv:1505.04406 [cs.LG], 2015.

[Baehrens *et al.*, 2010] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11:1803–1831, 2010.

[Berre and Parrain, 2010] Daniel Le Berre and Anne Parrain. The SAT4J library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7:50–64, 2010.

[Boyd *et al.*, 2007] Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi. A tutorial on geometric programming. *Optimization and Engineering*, 8(1):67–127, 2007.

[Buchman and Poole, 2017] David Buchman and David Poole. Negative probabilities in probabilistic logic programs. *Int. J. Approx. Reasoning*, 83:43–59, 2017.

[Chakraborty *et al.*, 2016] Supratik Chakraborty, Kuldeep S Meel, and Moshe Y Vardi. Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic sat calls. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2016.

[De Raedt and Kimmig, 2015] Luc De Raedt and Angelika Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, 100:5–47, 2015.

[Kok and Domingos, 2005] Stanley Kok and Pedro M. Domingos. Learning the structure of Markov logic networks. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, pages 441–448, 2005.

[Kuželka *et al.*, 2015] Ondřej Kuželka, Jesse Davis, and Steven Schockaert. Encoding Markov logic networks in possibilistic logic. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 454–463, 2015.

[Kuželka *et al.*, 2016] Ondřej Kuželka, Jesse Davis, and Steven Schockaert. Interpretable encoding of densities using possibilistic logic. In *Proceedings of the 22nd European Conference on Artificial Intelligence*, pages 1239–1247, 2016.

[Lang *et al.*, 1991] Jérôme Lang, D. Dubois, and Henri Prade. A logic of graded possibility and certainty coping with partial inconsistency. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 188–196, 1991.

[Muggleton and De Raedt, 1994] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

[Noessner *et al.*, 2013] Jan Noessner, Mathias Niepert, and Heiner Stuckenschmidt. RockIt: exploiting parallelism and symmetry for MAP inference in statistical relational models. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 739–745, 2013.

[Ram and Gray, 2011] Parikshit Ram and Alexander G Gray. Density estimation trees. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 627–635, 2011.

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[Sanchez *et al.*, 2015] Ivan Sanchez, Tim Rocktaschel, Sebastian Riedel, and Sameer Singh. Towards extracting faithful and descriptive representations of latent variable models. *AAAI Spring Syposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches*, 2015.

[Schulte *et al.*, 2014] Oliver Schulte, Hassan Khosravi, Arthur E. Kirkpatrick, Tianxiang Gao, and Yuke Zhu. Modelling relational statistics with Bayes nets. *Machine Learning*, 94(1):105–125, 2014.

[Serrurier and Prade, 2007] Mathieu Serrurier and Henri Prade. Introducing possibilistic logic in ILP for dealing with exceptions. *Artificial Intelligence*, 171(1617):939 – 950, 2007.

[Soos, 2010] Mate Soos. Cryptominisat 2.5. 0. *SAT Race competitive event booklet*, 2010.

[Weisfeiler and Lehman, 1968] Boris Weisfeiler and AA Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.