Evidence-Based Clustering for Scalable Inference in Markov Logic

Deepak Venugopal
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75080, USA
dxv021000@utdallas.edu

Vibhav Gogate
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75080, USA
vgogate@hlt.utdallas.edu

Abstract

Lifted inference algorithms take advantage of symmetries in first-order probabilistic logic representations such as Markov logic networks (MLNs), and are naturally more scalable than propositional inference algorithms which ground the MLN. However, lifted inference algorithms have an "evidence problem" – evidence breaks symmetries, and the performance of lifted inference algorithms is the same as propositional inference algorithms (or sometimes worse, due to overhead). In this paper, we propose a general method for addressing this problem. The main idea in our method is to approximate the given MLN having, say, n objects by an MLN having k objects such that $k << n$ and the results obtained by running potentially much faster inference on the smaller MLN are as close as possible to the ones obtained by running inference on the larger MLN. We achieve this by finding clusters of "similar" groundings using standard clustering algorithms (e.g., K-means), and replacing all groundings in the cluster by their cluster center. To this end, we develop a novel distance (or similarity) function for measuring the similarity between two groundings, based on the evidence presented to the MLN. We evaluated our approach on different benchmarks utilizing various clustering and inference algorithms. Our experiments clearly show the generality and scalability of our approach.

Introduction

Markov Logic Networks (MLNs) (Richardson and Domingos 2006; Domingos and Lowd 2009) unify first-order logic and probabilistic models and are arguably the most popular representation for statistical relational learning. They have been used in a wide variety of application domains including NLP (Poon and Domingos 2008), computer vision (Tran and Davis 2008) and planning (Singla and Mooney 2011). As with other probabilistic models, the key challenge in MLNs is to develop scalable inference algorithms. The scalability requirement is even more crucial in MLNs because, even a seemingly simple MLN can yield an arbitrary large (propositional) probabilistic model as the number of objects in the real-world domain increases.

Existing MLN inference algorithms can be broadly classified into two categories, propositional algorithms such as Gibbs Sampling (Geman and Geman 1984) and Belief Propagation (Yedidia, Freeman, and Weiss 2001; Wainwright, Jaakkola, and Willsky 2003) which operate on the Markov network obtained by grounding the MLN and lifted algorithms (Poole 2003; de Salvo Braz 2007; Gogate and Domingos 2011; van den Broeck et al. 2011; Singla and Domingos 2008; Gogate, Jha, and Venugopal 2012; Venugopal and Gogate 2012; Niepert 2012; Bui, Huynh, and de Salvo Braz 2012; Bui, Huynh, and Riedel 2013), which operate directly on the first-order representation, grounding only as necessary. Propositional algorithms do not scale well because the Markov network representing the MLN is huge for large domain-sizes. On the other hand, all lifted inference algorithms, in principle, can scale significantly better than propositional inference.

Wins(A,A)	0.56
Wins(A,B)	0.56
Wins(A,C)	0.56
Wins(B,A)	0.56
Wins(B,B)	0.56
Wins(B,C)	0.56
Wins(C,A)	0.56
Wins(C,B)	0.56
Wins(C,C)	0.56

(a)

Strong(C)
Wins(A,C)
Wins(B,B)
Wins(B,C)
Wins(C,A)

(b)

Wins(A,A)	0.6
Wins(A,B)	0.6
Wins(B,A)	0.63
Wins(C,B)	0.85
Wins(C,C)	0.85

(c)

Figure 1: Effect of evidence on an MLN with one formula, 1.75 Strong(x) $\Rightarrow$ Wins(x,y). The marginal probabilities which were equal in (a) become unequal in (c) due to evidence shown in (b).

Lifted inference algorithms typically suffer from two problems. First, they require MLNs to have a specific symmetric structure (de Salvo Braz 2007; Jha et al. 2010; van den Broeck 2011), which is not always the case in real-world applications. For example, to apply certain inference operations, the MLN needs to be composed of purely singleton atoms (Jha et al. 2010). Second, a far more serious problem is that, in the presence of evidence most MLNs are not liftable because evidence breaks symmetries. As a concrete example, the symmetrical marginal probabilities in Fig. 1 (a) are broken as shown in (c) with evidence (b). Therefore, a lifted algorithm that could potentially exploit the symmetry in (a) can no longer do so in (c) and needs to ground the MLN. To solve this evidence problem, recently, (van den Broeck and Darwiche 2013) suggest approximating the evidence to make the MLN liftable. While this work is specific to binary evidence, our main contribution in this paper is a general approach for scalable inference on any MLN given arbitrary evidence.

Our main idea is to reduce the number of objects in the domain of the MLN, thereby approximating it by a much smaller MLN such that the results obtained by performing inference on the smaller MLN are as close as possible to the ones obtained by running an expensive inference algorithm on the original MLN. For this, we utilize standard clustering algorithms such as K-means to merge together several "similar/related" groundings in an MLN. Importantly, this allows us to plug-in existing grounded/lifted inference algorithms where the sampling-space (for sampling-based inference) or search-space (for search-based inference) can be controlled, which makes inference feasible even on very large domains.

In order to obtain a domain-reduced approximation of the original MLN, we specify a novel distance function that measures similarity based on the evidence presented to the MLN. The distance function helps cluster together objects in the real-world domain that have similar evidence-structure. The inherent symmetry in MLN representation makes it more likely that the marginal probabilities corresponding to objects in a common cluster are approximately close to each other. Thus, we compute the marginal probability for a single element of the cluster and project the same results to all elements in the cluster.

We evaluate our approach on benchmark MLNs available on the Alchemy website (Kok et al. 2008). In our experiments, we leverage clustering algorithms from data-mining/machine learning literature implemented in Weka (Hall et al. 2009) to scale-up to large domain-sizes. We experimented with two inference algorithms, a propositional sampling-based algorithm, Gibbs sampling and a lifted message-passing algorithm, Lifted Belief Propagation, to show the generality of our approach. Our results clearly illustrate the scalability and accuracy of our approach.

Preliminaries

First-order logic (FOL) consists of predicates (e.g. `Friends`) that represent relations between objects, logical connectives (e.g. $\vee$, $\neg$ etc.) and quantifiers ($\forall$, $\exists$). Each predicate has a parenthesized list of arguments which can be substituted by a term which can either be a logical variable (x), a constant (X) or a function. A formula in first order logic is a predicate (atom), or any complex sentence that can be constructed from atoms using logical connectives and quantifiers. A *ground* atom corresponding to a predicate is one where each term is substituted by a constant symbol. In contrast to full FOL, we make two relaxations here. First, we make the assumption of a *closed universe*, i.e., there is a one-to-one mapping between the constant symbols and objects. This means that any possible *world* is simply an assignment of `True` or `False` to every distinct *ground* atom. Secondly, we assume a function-free language where each term is universally quantified and the number of constant symbols is finite. Therefore, for any variable x, we can define a finite set Δ_x (domain of x) which consists of all the constants that can be substituted for x. A ground formula is a formula obtained by substituting all of its variables with a constant.

Markov logic (Domingos and Lowd 2009) extends FOL by softening the hard constraints expressed by the formulas. A soft formula or a weighted formula is a pair (f, w) where f is a formula in FOL and w is a real-number. A MLN denoted by $\mathcal{M}$, is a set of weighted formulas (f_i, w_i). Given a set of constants that represent objects in the domain, an MLN defines a Markov network or a log-linear model. The Markov network is obtained by grounding the weighted first-order knowledge base and represents the following probability distribution.

$$P_{\mathcal{M}}(\omega) = \frac{1}{Z(\mathcal{M})} \exp\left(\sum_i w_i N(f_i, \omega)\right) \quad (1)$$

where ω is a world, $N(f_i, \omega)$ is the number of groundings of f_i that evaluate to `True` in the world ω and $Z(\mathcal{M})$ is a normalization constant or the partition function.

The two main inference problems in MLNs are computing the partition function and the marginal probabilities of query atoms given evidence. In this paper, we focus on the latter.

Domain Clustering

Problem Formulation

We first begin with some notation. Let $\mathcal{M}$ represent an MLN with M predicates $\mathrm{R}_1, \mathrm{R}_2, \ldots, \mathrm{R}_M$, and N weighted formulas $f_1, f_2, \ldots, f_N$. Let $G_{\mathcal{M}}$ represent the ground KB of the formulas in $\mathcal{M}$. Let $\mathbf{E} = \{E_k\}_{k=1}^{S}$ be the set of *evidences*. Each $E_k \in \mathbf{E}$ represents a single ground atom that is known to be either `True` or `False`. Let $\mathbf{I}$ be a set of indices of the form (i, j) such that $1 \le i \le M$, $1 \le j \le A_i$, where A_i is the arity of the i^{th} predicate. In other words, (i, j) is an index to the j^{th} argument of the i^{th} predicate in $\mathcal{M}$. We now define a partition of $\mathbf{I}$ and its domain as follows.

Definition 1. Let $\mathcal{I} = \{\mathcal{I}_1\ \mathcal{I}_2 \ldots \mathcal{I}_P\}$ be a partition of $\mathbf{I}$ such that, $\{(i', j'), (i'', j'')\} \subseteq \mathcal{I}_k$ if and only if there exists a formula $f_m \in \mathcal{M}$ where (i', j') and (i'', j'') refer to the same logical variable in f_m. The domain (or groundings) of $\mathcal{I}_k = \{(i_1, j_1), (i_2, j_2), \ldots (i_{k'}, j_{k'})\}$ represented as $\Delta_{\mathcal{I}_k}$ is equal to $\Delta_{(i_1,j_1)} = \Delta_{(i_2,j_2)} = \ldots \Delta_{(i_{k'},j_{k'})}$, where $\Delta_{(i,j)}$ represents the domain of (i, j).

Example 1. Let $\mathcal{M}$ contain exactly one formula $\mathrm{R}_1(x,y) \wedge \mathrm{R}_2(y,z) \Rightarrow \mathrm{R}_3(z,x)$. Let $\Delta_x = \Delta_y = \Delta_z = \{A,B\}$. $\mathcal{I} = \{\{(1,1), (3,2)\}, \{(1,2), (2,1)\}, \{(2,2), (3,1)\}\}$. $\Delta_{\mathcal{I}_1} = \{A,B\}$ and grounding $\mathcal{I}_1$ with A, yields the partially ground formula, $\mathrm{R}_1(A,y) \wedge \mathrm{R}_2(y,z) \Rightarrow \mathrm{R}_3(z,A)$.

To reduce the total number of formulas in $G_{\mathcal{M}}$, we reduce the number of groundings in each $\mathcal{I}_k \in \mathcal{I}$ independently. Specifically, for each $\Delta_{\mathcal{I}_k}$, we learn a new domain, $\hat{\Delta}_{\mathcal{I}_k}$ and a mapping ζ, where $|\hat{\Delta}_{\mathcal{I}_k}| \le |\Delta_{\mathcal{I}_k}|$, for each $\mu \in \hat{\Delta}_{\mathcal{I}_k}$, $\zeta(\mu) = \mathbf{C}' \subseteq \Delta_{\mathcal{I}_k}$ and $\forall\ \mu', \mu'', \mu' \neq \mu'' \Rightarrow \zeta(\mu') \cap \zeta(\mu'') = \emptyset$. We formulate this domain-reduction problem as a standard clustering problem below.

Definition 2. Given a distance measure d between any two groundings of $\mathcal{I}_k \in \mathcal{I}$ and the number of clusters for $\mathcal{I}_k$ equal to r_k, we define the clustering problem as,

$$\min_{\mathbf{C_1}\ldots\mathbf{C_P}} \sum_{k=1}^{P}\sum_{j=1}^{r_k}\sum_{C_{kj}\in\mathbf{C}_{kj}} d(C_{kj},\mu_{kj}) \quad (2)$$

where $\mathbf{C}_{kj}$ corresponds to groundings of $\mathcal{I}_k$ in cluster j, μ_{kj} is the cluster-center of $\mathbf{C}_{kj}$, i.e., it represents the "average grounding" for that cluster and $\zeta(\mu_{kj}) = \mathbf{C}_{kj}$.

The modified (reduced) domain for $\mathcal{I}_k$, $\hat{\Delta}_{\mathcal{I}_k}$ is equal to $\{\mu_{kj}\}_{j=1}^{r_k}$. The cluster-centers in some sense form a compressed representation of the original domain. From this compressed representation, we generate a modified MLN $\hat{\mathcal{M}}$ with a reduced number of predicate groundings. Formally,

Definition 3. For the i^{th} predicate in $\mathcal{M}$, its reduced domain in $\hat{\mathcal{M}}$ is given by,

$$\hat{\Delta}_{\mathrm{R}_i} = \bigcup_{j=1}^{r_{i_1}} \mu_{i_1 j} \times \bigcup_{j=1}^{r_{i_2}} \mu_{i_2 j} \ldots \bigcup_{j=1}^{r_{i_{A_i}}} \mu_{i_{A_i} j} \quad (3)$$

where $(i,1) \in \mathcal{I}_{i_1}, (i,2) \in \mathcal{I}_{i_2} \ldots (i,A_i) \in \mathcal{I}_{i_{A_i}}$

The formulation in Eq. (2) allows us to leverage existing clustering algorithms in machine learning/data mining and makes inference feasible even when $G_{\mathcal{M}}$ is very large. This is important because, for arbitrary MLN structures or for inference with evidence, even state-of-the-art inference techniques end up working on a model whose size is comparable to $G_{\mathcal{M}}$ and in most cases, $G_{\mathcal{M}}$ grows rapidly with domain-size. For example, consider the MLN, $\mathrm{R}(x, y) \wedge \mathrm{S}(y, z) \Rightarrow \mathrm{T}(z, x)$, if $\Delta_x = \Delta_y = \Delta_z = 100$, $G_{\mathcal{M}}$ has a million formulas. Further, the search space (for search-based algorithms) or the sampling space (for sampling-based algorithms) is massive, i.e., exponential in the number of ground predicates. By clustering, we are essentially compressing this large space and now any existing inference algorithm can be used to solve large problems as they implicitly work in this reduced space. The key advantage is that this space complexity can now be controlled based on the number of clusters. Specifically,

Proposition 1. *The number of ground predicates in $\hat{\mathcal{M}}$ is $O(Mr^A)$, where M is the number of predicates in $\mathcal{M}$, $r = \max_k r_k$ and A is the maximum arity of a predicate in $\mathcal{M}$.*

From Eq. (3), it is clear that the domain of a predicate in $\hat{\mathcal{M}}$ is a set of a cluster-centers rather than a set of concrete objects in $\mathcal{M}$. Thus, one grounding of a predicate in $\hat{\mathcal{M}}$ implicitly corresponds to multiple groundings of that predicate in $\mathcal{M}$. This means that in $\hat{\mathcal{M}}$, the original evidence $\mathbf{E}$ is not valid anymore since $\mathbf{E}$ is specified on the original groundings of a predicate. Therefore, we approximate $\mathbf{E}$ with $\hat{\mathbf{E}}$ which specifies the evidence on predicates ground with cluster-centers instead of the original objects in $\mathcal{M}$. To specify this, we define the *expansion* of a predicate grounding in $\hat{\mathcal{M}}$ as the set of groundings that it represents in $\mathcal{M}$.

Definition 4. The *expansion* of the j^{th} grounding of the i^{th} predicate ($\mathrm{R}_i(\mu_{i_1 j_1}, \ldots \mu_{i_{A_i} j_{A_i}})$) in $\hat{\mathcal{M}}$ is given by,

$$\pi_{ij} = \mathrm{R}_i\left(\zeta(\mu_{i_1 j_1}) \times \ldots \zeta(\mu_{i_{A_i} j_{A_i}})\right) \quad (4)$$

Clearly, if we assert in $\hat{\mathbf{E}}$ that a grounding of a predicate in $\hat{\mathcal{M}}$ is `True` (or `False`), this implicitly asserts that every grounding in the expansion of that predicate is `True` (or `False`). Given a clustering of the domains, in order to best approximate $\mathbf{E}$ for this clustering, we minimize the symmetric difference between the new evidence and the original evidence as represented below.

$$\min_{\hat{\mathbf{E}}} |\mathbf{E} \triangle \bar{\pi}(\hat{\mathbf{E}})| \quad (5)$$

where $\hat{\mathbf{E}}$ is a subset of predicate groundings in $\hat{\mathcal{M}}$ and each grounding is assigned a sign (positive/`True` or negative/`False`), $\bar{\pi}(\hat{\mathbf{E}})$ expands every grounding in $\hat{\mathbf{E}}$ and assigns each grounding in the expansion the same sign as its corresponding grounding in $\hat{\mathbf{E}}$. The $\triangle$ operator computes the symmetric difference between $\mathbf{E}$ and $\bar{\pi}(\hat{\mathbf{E}})$ (A grounding with different signs is treated as distinct elements).

We can easily show the following result to construct the optimal $\hat{\mathbf{E}}$ according to Eq. (5).

Proposition 2. *Let π_{ij} be the expansion of one grounding ($\hat{E}$) in $\hat{\mathbf{E}}$. Let n_+ be the count of positive-sign elements and n_-, the count of negative-sign elements in $\pi_{ij} \cap \mathbf{E}$. Eq. (5) is optimized by assigning $\hat{E}$ as positive if $n_+ \geq \frac{|\pi_{ij}|}{2}$ and $\hat{E}$ as negative if $n_- \geq \frac{|\pi_{ij}|}{2}$.*

Alg. 1 computes the marginal probabilities in an MLN given evidence. It needs three other methods as input namely, the distance function, clustering algorithm and the inference algorithm. The amount of reduction applied to each domain is specified as α. The algorithm starts by computing the partition $\mathcal{I}$ from the term dependencies in $\mathcal{M}$. Next, to each $\mathcal{I}_k \in \mathcal{I}$, the clustering algorithm $\mathcal{L}$ is applied which outputs the clustered domain $\Delta_{\mathcal{I}_k}$ as well as the mapping function ζ that maps each cluster-center to its original domain. $\Delta_{\mathcal{I}_k}$ is now replaced by its approximation in the new MLN $\hat{\mathcal{M}}$. Once all the domains are suitably reduced, the next step is to approximate the evidence based on the reduced domains. Using Proposition 2, for every grounding of every predicate in the approximate MLN $\hat{\mathcal{M}}$, we make a decision as to whether it is to be considered positive evidence, negative evidence or unknown. This yields the approximate evidence set $\hat{\mathbf{E}}$. $\hat{\mathcal{M}}$ is then re-weighted where each grounded formula's weight is multiplied by the number of groundings it represents in $\mathcal{M}$. We then invoke the inference algorithm $\mathcal{F}$ to compute the marginals in $\hat{\mathcal{M}}$. Finally, we project the results obtained on $\hat{\mathcal{M}}$ back to the original domains. Specifically, if $\hat{\mathcal{M}}$ has a probability p, then each grounding in its expansion is assigned the same probability.

Distance Function

The distance function d is a key parameter that affects the clustering in Eq. (2) and in turn the inference results in Alg. 1. The advantage of our formulation is that it is easy to plug-in a new distance function and generate inference algorithms targeted towards specific applications or datasets. Here, we develop a distance function using the evidence, such that groundings within a cluster have similar marginals.

```
Algorithm 1: Compute-Marginals
 Input: MLN $\mathcal{M}$, Evidence $\mathbf{E}$, Query $\mathbf{Q}$, Distance
        function $d$, Clustering function $\mathcal{L}$, Inference
        algorithm $\mathcal{F}$, cluster-bound $\alpha$
 Output: Marginal probabilities $\mathcal{P}$ for $\mathbf{Q}$
1  Compute the partition $\mathcal{I}$ from $\mathcal{M}$
2  $\hat{\mathcal{M}} = \mathcal{M}$
3  for $\mathcal{I}_k \in \mathcal{I}$ do
4      $numclusters = \alpha \times \Delta_{\mathcal{I}_k}$
5      $(\hat{\Delta}_{\mathcal{I}_k}, \zeta) = \mathcal{L}(numclusters, d)$
6      Replace $\Delta_{\mathcal{I}_k}$ with $\hat{\Delta}_{\mathcal{I}_k}$ in $\hat{\mathcal{M}}$
7  $\hat{\mathbf{E}} = \emptyset$
8  for $\mathtt{R}_i \in \mathcal{M}$ do
9      Compute $\hat{\Delta}_{\mathtt{R}_i}$ using Eqn. (3)
10     for Each j, where j indexes the groundings of $\hat{\Delta}_{\mathtt{R}_i}$
       do
11         Compute its expansion $\pi_{ij}$ using Eqn. (4)
12         Add the $j^{th}$ grounding of $\mathtt{R}_i$ to $\hat{\mathbf{E}}$ based on
           Proposition 2
13     $\hat{\mathcal{P}} = \mathcal{F}(\hat{\mathcal{M}}, \hat{\mathbf{E}}, \mathbf{Q})$
14     for Each $\mathtt{R}_k \in \mathbf{Q}$ do
15         for Each j, where j indexes the groundings of
           $\mathtt{R}_k$ in $\hat{\mathcal{M}}$ do
16             for Each t, where t indexes the groundings
               of $\mathtt{R}_k$ in the expansion $\pi_{kj}$ do
17                 $\mathcal{P}(\mathtt{R}_k, t) = \hat{\mathcal{P}}(\mathtt{R}_k, j)$
18 return $\mathcal{P}$
```

Example 2. Consider the MLN with one formula $\mathtt{R}(x) \Rightarrow \mathtt{S}(x,y)$ with weight 1.75 and domain $\Delta_x = \{A, B, C\}$. Let the evidence $\mathbf{E} = \{\mathtt{R}(A), \mathtt{R}(B)\}$. The task is to compute the marginal probabilities of all groundings of $\mathtt{S}(x,y)$ which we refer to as the query. The exact marginal probabilities for the query are, $\mathtt{S}(A,y) = \mathtt{S}(B,y) = 0.5$, $\mathtt{S}(C,y) = 0.56$. Thus, an ideal distance function should give us a clustering of Δ_x where B and C are placed in the same cluster as they have the same marginals w.r.t the query variable. To do this, we observe that the evidence on $\mathtt{R}(A) \Rightarrow \mathtt{S}(A,y)$ and $\mathtt{R}(B) \Rightarrow \mathtt{S}(B,y)$ are "symmetrical" i.e. they satisfy the same number of groundings and consequently the number of groundings that are left unsatisfied in both the formulas is the same. In other words, when $x = A$, the relevant evidence yields MLN $\mathcal{M}'$ and when $x = B$, its yields $\mathcal{M}''$ and if $\mathcal{M}'$ is sufficiently close to $\mathcal{M}''$, we would want all the groundings where $x = A$ clustered together with the groundings where $x = B$ because they are likely to have the same marginal probabilities. We formalize this intuitive idea below.

Let $\mathcal{M}_{C_{kj}}$ represent the MLN obtained after grounding $\mathcal{I}_k$ with the j^{th} constant in $\Delta_{\mathcal{I}_k}$. Clearly, in the general case, for any two distinct j_1, j_2, $\mathcal{M}_{C_{kj_1}}$ and $\mathcal{M}_{C_{kj_2}}$ are not necessarily independent MLNs as there may be atoms in $\mathcal{M}_{C_{kj_1}}$ that are also present in $\mathcal{M}_{C_{kj_2}}$. However, in our

```
Algorithm 2: Build-Query
 Input: Clausal formula $f_t$
 Output: Relational-Algebra expression $\mathcal{Q}$
1  $\mathcal{Q} = \emptyset$
2  for $\mathtt{R}_i \in f_t$ do
3      $Rvalue = 1$
4      if $\mathtt{R}_i$ is positive then
5          $Rvalue = 0$
6      if $\mathcal{Q} = \emptyset$ then
7          $\mathcal{Q} = \mathcal{Q} + \sigma_{R_i.val=Rvalue}(R_i)$
8      else
9          $\mathcal{Q} = \mathcal{Q} \bowtie_\theta \sigma_{R_i.val=Rvalue}(R_i)$
```

distance function, we relax the constraints/dependencies between $\mathcal{M}_{C_{kj_1}}$, $\mathcal{M}_{C_{kj_2}}$ and assume these to be independent MLNs and compute the distance between these two MLNs. Specifically, we define a feature vector $\mathbf{U}_{C_{kj}} = c_{f_1}, c_{f_2} \ldots c_{f_N}$, where c_{f_k} is the number of satisfied groundings in formula f_k of MLN $\mathcal{M}_{C_{kj}}$ due to the evidence $\mathbf{E}$. The distance is computed as $d(C_{kj_1}, C_{kj_2}) = ||\mathbf{U}_{C_{kj_1}} - \mathbf{U}_{C_{kj_2}}||$.

Even though the above distance function seems like an intuitive and reasonable heuristic, it turns out that computing the distance function efficiently is infeasible in general because computing the counts in $\mathbf{U}_{C_{kj}}$ is a hard problem. In order to formalize this clearly, we specify the predicates and their groundings using a relational database. Further, we also assume that each formula is reduced to a clausal form. The i^{th} predicate $\mathtt{R}_i$ is stored as a table R_i with $A_i + 1$ columns (A_i is the arity), namely, $id_1 \ldots id_{A_i}$ and val. The first A_i columns store the constants corresponding to a grounding of the predicate and the val column specifies the assignment. Given such a database, borrowing results from (Domingos and Lowd 2009),

Theorem 1. *Computing the number of satisfied groundings of a first-order clause in a database is #P-complete in the length of the clause.*

Thus, computing $\mathbf{U}_{C_{kj}}$ is hard as it requires queries that join an arbitrary number of tables. Therefore, we further relax the constraints/dependencies within the atoms in a formula to guarantee feasibility of computing $\mathbf{U}_{C_{kj}}$ as illustrated below.

Example 3. Let $\mathcal{M}$ contain one formula, $\neg\mathtt{R}(x, y) \vee \neg\mathtt{S}(y,z) \vee \mathtt{T}(z,x)$, where $\Delta_x = \{A, B, C\}$. To compute the count of satisfied groundings for $x = A$, we compute its inverse, i.e., the number of unsatisfied groundings for $x = A$. The satisfied count is simply the difference between the total number of groundings and the number of unsatisfied groundings. As the total number of groundings $\Delta_y \times \Delta_z$ is a constant for all groundings of x, we simply ignore it. The unsatisfied groundings for $x = A$ is given by,

$$\sigma_{R.val=1 \wedge S.val=1 \wedge T.val=0}((\sigma_{R.id_0=A}(R) \bowtie_{R.id_1=S.id_0} S) \bowtie_{S.id_1=T.id_0 \wedge R.id_0=T.id_1} T) \quad (6)$$

where σ is the selection operator and $\bowtie$ is the join operator. Clearly, the above expression has two joins. However, if we

Algorithm 3: Compute-Features

```
Input: $\mathcal{M}$ and its associated relational DB, join-bound $J, \mathcal{I}_k \in \mathcal{I}$
Output: Feature vector set $\{\mathbf{U}_{C_{kj}}\}_{j=1}^{\Delta_{\mathcal{I}_k}}$
1  $\mathbf{U} = \emptyset$
2  for $C_{kj} \in \Delta_{\mathcal{I}_k}$ do
3      $\mathbf{U}_{C_{kj}} = \emptyset$
4      for $f_t \in \mathbf{F}$ do
5          if $f_t$ is not relevant to $\mathcal{I}_k$ then
6              continue
7          $\mathcal{Q}$ = Build-Query($f_t$)
8          while $\mathcal{Q}$ do
9              $\mathcal{Q}'$ = Select a sub-query containing up to the first $J$ joins in $\mathcal{Q}$
10             for $R_i \in \mathcal{Q}'$ do
11                 if $\exists\ p$ such that $(i,p) \in \mathcal{I}_k$ then
12                     Wrap a select ($\sigma_{R_i.id_p=C_{kj}}$) around $R_i$
13             $\mathbf{U}_{C_{kj}}$.append(Count($\mathcal{Q}'$))
14             Let $R_s$ be a table in $\mathcal{Q}'$ whose attribute participates in the $\theta$-join after $\mathcal{Q}'$
15             if $R_s = \emptyset$ then
16                 $\mathcal{Q} = (\mathcal{Q} - \mathcal{Q}')$
17             else
18                 Relax the $\theta$-join and include only those constraints involving $R_s$
19                 $\mathcal{Q} = R_s \bowtie_\theta (\mathcal{Q} - \mathcal{Q}')$
20     $\mathbf{U}$.append($\mathbf{U}_{C_{kj}}$)
21 return $\mathbf{U}$
```

impose a constraint that no joins are allowed during the computation of the feature vector, we approximate Eq. (6) by implicitly assuming that each predicate in the formula is independent i.e. we ignore the joins to obtain a vector of counts from 3 separate queries, $\sigma_{R.val=1 \wedge R.id_0=A}(R)$, $\sigma_{S.val=1}(S)$ and $\sigma_{T.val=0 \wedge T.id_1=A}(T)$. An alternate distance function can be obtained if we only allow exactly one join in a query. In this case, we can get a better approximation of Eq. (6) by considering two pairwise joins in the formula as,

$$\sigma_{R.id_0=A \wedge R.val=1}(R) \bowtie_{R.id_1=S.id_0} \sigma_{S.val=1}(S)$$

$$\sigma_{S.val=1}(S) \bowtie_{S.id_1=T.id_0} \sigma_{T.val=0 \wedge T.id_1=A}(T)$$

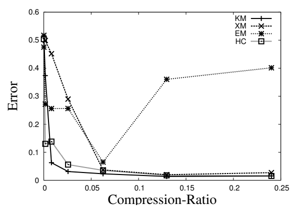
Alg. 3 generalizes the idea in the above example and computes the feature vectors for a specific $\mathcal{I}_k \in \mathcal{I}$. The algorithm generates multiple queries corresponding to each grounding of $\mathcal{I}_k$ such that the number of joins in each query is lesser than J. For this, we go over each formula f_t, and first check if f_t is *relevant* to $\mathcal{I}_k$ i.e. if f_t contains at least one predicate R_i such that for some p, $(i, p) \in \mathcal{I}_k$, then f_t is a relevant formula for clustering $\mathcal{I}_k$, otherwise, we ignore f_t. This is because, if f_t is not relevant to $\mathcal{I}_k$, then f_t yields identical counts for every grounding of x and therefore never affects the clustering. For every relevant f_t, we first build the complete query which is a sequence of θ-joins on the tables corresponding to every predicate in f_t. Each predicate is wrapped with a "Select" to obtain tuples whose value is equal to the inverse of the predicate's sign in f_t. The θ in the join specifies equality conditions on all the variables shared between the predicates in f_t. For e.g. In a formula $\neg\mathtt{R}(x) \vee \mathtt{S}(x)$, the θ-join is specified as $\sigma_{R.val=1}(R) \bowtie_{R.id_0=S.id_0} \sigma_{S.val=0}(S)$. Once we build the full query, we simply walk through the query and execute no more than J joins at a time. For each predicate which has a variable that corresponds to some element of $\mathcal{I}_k$, we ground the variable by enforcing the select condition in line 12 of the algorithm. We execute the partial query $\mathcal{Q}'$ with a maximum of J joins and store the result (count) in the feature vector. Next, we remove $\mathcal{Q}'$ from $\mathcal{Q}$ and relax the next θ- join condition as follows. Among all the tables mentioned in $\mathcal{Q}'$, we select one table R_s, that participates in the next join operation in $\mathcal{Q} - \mathcal{Q}'$. We only retain the join conditions related to R_s in the next join in $\mathcal{Q} - \mathcal{Q}'$ and remove the rest of the conditions. We continue until we empty the original query $\mathcal{Q}$. Finally, we return the accumulated set of feature vectors.

Experiments

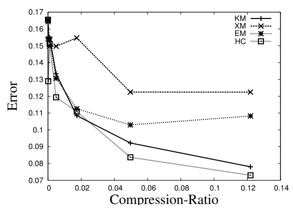
We evaluate our approach on 3 benchmark MLNs available in the Alchemy website (Kok et al. 2008), namely Segmentation (Seg), WebKB, Protein and a new MLN, Student ($\mathtt{Teaches}(i, c) \wedge \mathtt{Prereq}(c, c1) \Rightarrow \mathtt{Takes}(s, c1)$). In the inference subroutine, we used two algorithms from Alchemy, a lifted algorithm based on message passing, Lifted-BP (Singla and Domingos 2008) and a propositional algorithm based on sampling, Gibbs sampling (Geman and Geman 1984). For clustering, we used four algorithms available in Weka (Hall et al. 2009) namely, KMeans++ (KM), Expectation-Maximization (EM), Hierarchical clustering (HC) and XMeans (XM). We set the bound on the number of joins (J) as 1. We ran our experiments on a quad-core Ubuntu machine with 6GB RAM.

Approximation results on benchmarks

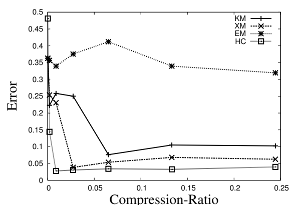
Fig. 2 illustrates the approximation error on each benchmark for all combinations of the inference and clustering algorithms. The *x-axis* plots $CR = \frac{N_c}{N_g}$, where N_c is the total number of ground formulas in $\hat{\mathcal{M}}$ and N_g is the total number of ground formulas in $\mathcal{M}$. The *y-axis* shows the approximation error calculated as follows. $Err = \frac{\sum_{g \in G} D_{KL}(P_g || P'_g)}{|G|}$, where D_{KL} is the standard KL-Divergence distance measure, G refers to all groundings of a query predicate, P_g is the marginal distribution of g computed from $\mathcal{M}$ and P'_g is computed from $\hat{\mathcal{M}}$. For fairness, both marginals are computed using the same inference algorithm. We set 50% of arbitrary groundings as evidence, where 25% are `True` and 25% are `False`. Fig. 2 illustrates the trade-off between accuracy and complexity. On all experiments, as CR increases, the complexity increases, however, the approximation error reduces as we map the original domains to a larger set thereby reducing the difference between the original MLN and the approximate MLN. In almost all cases



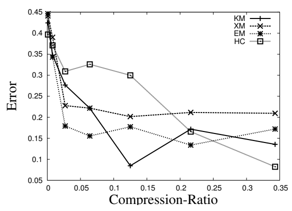
(a) Student



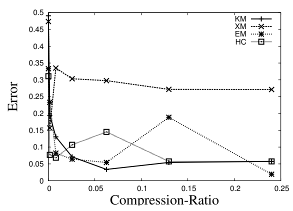
(b) Seg



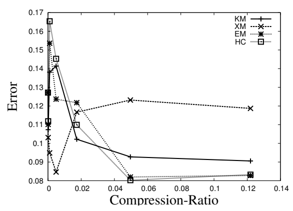
(c) Webkb



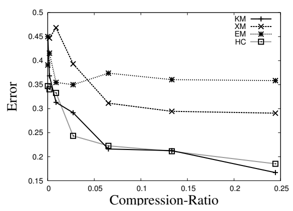
(d) Protein



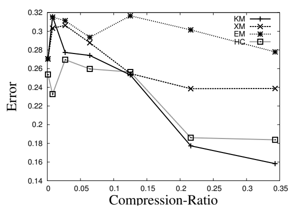
(e) Student



(f) Seg

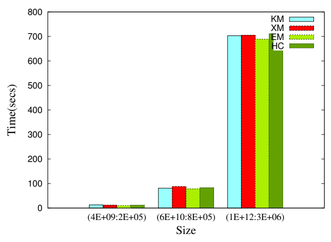


(g) Webkb

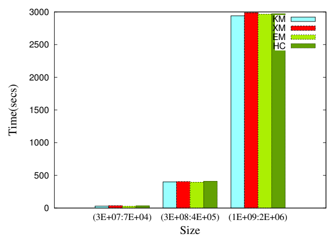


(h) Protein

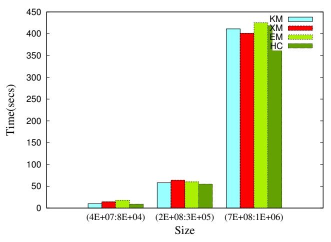
Figure 2: Approximation error for varying $CR = \frac{N_C}{N_G}$), where N_C is the number of ground formulas $\hat{\mathcal{M}}$ and N_G the number of ground formulas in $\mathcal{M}$. The y-axis shows the average KL-Divergence of the marginals computed from $\hat{\mathcal{M}}$ to those computed using $\mathcal{M}$ (smaller is better). (a) - (d) show the results using Lifted-BP, (e) - (h) using Gibbs sampling.



(a) Student



(b) Seg



(c) Webkb

Figure 3: Scalability experiments. The y-axis shows the time in seconds to create the approximate MLN and the x-axis shows (N_f : N_a), where N_f is the number of ground formulas and N_a is the number of ground predicates.

for Lifted-BP, the approximation error was below 0.2 for even small va;ues of CR. For Gibbs sampling though, in some cases (g,h), it took a larger CR value for the approximation error to reduce significantly. One of the reasons for this could be that Gibbs sampling in general performs poorly in MLNs when there are hard constraints (evidence) in the model (Poon and Domingos 2006) and therefore yields worse approximation.

Scalability

Fig. 3 illustrates the scalability of our approach when handling large domain-sizes. For different domain-sizes, we show the time in seconds it takes to compute the approximate MLN after clustering. We used an α value of 0.25 for these experiments and introduced 50% random evidence with half of them `True` and the other half `False`. As expected, the time taken to compute the approximate MLN increases as the domain-size grows. However, we were able to complete processing the MLN in a reasonable amount of time even when the ground MLN had a trillion formulas (a). Also, the number of first-order formulas play a role in determining the complexity due to the distance function computation. Recall that we compute a vector for every formula in the MLN. Therefore, a larger number of formulas mean more computations on the database. For instance, Fig. 3 (a) has just one formula while (b) has 7. Therefore, even though the number of ground formulas in (a) is a trillion while in (b) it is a billion, (b) takes longer to process. Further, for each MLN, the third (largest) instance takes a visibly longer time. This is expected because, the size of the database grows really large and query-processing requires more hard disk accesses. Finally, the type of clustering has minimal impact on the processing time.

Acknowledgements This research was partly funded by ARO MURI grant W911NF-08-1-0242, by the AFRL under contract number FA8750-14-C-0021 and by the DARPA Probabilistic Programming for Advanced Machine Learning Program under AFRL prime contract number FA8750-14-C-0005. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, AFRL, ARO or the US government.

References

Bui, H.; Huynh, T.; and de Salvo Braz, R. 2012. Exact lifted inference with distinct soft evidence on every object. In *AAAI*.

Bui, H.; Huynh, T.; and Riedel, S. 2013. Automorphism groups of graphical models and lifted variational inference. In *UAI*, 132–141. AUAI Press.

de Salvo Braz, R. 2007. *Lifted First-Order Probabilistic Inference*. Ph.D. Dissertation, University of Illinois, Urbana-Champaign, IL.

Domingos, P., and Lowd, D. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. San Rafael, CA: Morgan & Claypool.

Geman, S., and Geman, D. 1984. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6:721–741.

Gogate, V., and Domingos, P. 2011. Probabilistic Theorem Proving. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, 256–265. AUAI Press.

Gogate, V.; Jha, A.; and Venugopal, D. 2012. Advances in Lifted Importance Sampling. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The weka data mining software: An update. *SIGKDD Explor. Newsl.* 11(1):10–18.

Jha, A.; Gogate, V.; Meliou, A.; and Suciu, D. 2010. Lifted Inference from the Other Side: The tractable Features. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems (NIPS)*, 973–981.

Kok, S.; Sumner, M.; Richardson, M.; Singla, P.; Poon, H.; Lowd, D.; Wang, J.; and Domingos, P. 2008. The Alchemy System for Statistical Relational AI. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA. http://alchemy.cs.washington.edu.

Niepert, M. 2012. Markov chains on orbits of permutation groups. In *UAI*, 624–633. AUAI Press.

Poole, D. 2003. First-Order Probabilistic Inference. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 985–991. Acapulco, Mexico: Morgan Kaufmann.

Poon, H., and Domingos, P. 2006. Sound and Efficient Inference with Probabilistic and Deterministic Dependencies. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Boston, MA: AAAI Press. This volume.

Poon, H., and Domingos, P. 2008. Joint Unsupervised Coreference Resolution with Markov Logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 649–658. Honolulu, HI: ACL.

Richardson, M., and Domingos, P. 2006. Markov Logic Networks. *Machine Learning* 62:107–136.

Singla, P., and Domingos, P. 2008. Lifted First-Order Belief Propagation. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 1094–1099. Chicago, IL: AAAI Press.

Singla, P., and Mooney, R. J. 2011. Abductive Markov Logic for Plan Recognition. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 1069–1075.

Tran, S. D., and Davis, L. S. 2008. Event modeling and recognition using markov logic networks. In *ECCV*.

van den Broeck, G., and Darwiche, A. 2013. On the complexity and approximation of binary evidence in lifted inference. In *Advances in Neural Information Processing Systems 26*, 2868–2876.

van den Broeck, G.; Taghipour, N.; Meert, W.; Davis, J.; and De Raedt, L. 2011. Lifted Probabilistic Inference by First-Order Knowledge Compilation. In *Proceedings of the Twenty Second International Joint Conference on Artificial Intelligence*, 2178–2185.

van den Broeck, G. 2011. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *NIPS*, 1386–1394.

Venugopal, D., and Gogate, V. 2012. On lifting the gibbs sampling algorithm. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS)*, 1664–1672.

Wainwright, M. J.; Jaakkola, T. S.; and Willsky, A. S. 2003. Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching. In *In Nineth Workshop on Artificial Intelligence and Statistics*.

Yedidia, J. S.; Freeman, W. T.; and Weiss, Y. 2001. Generalized Belief Propagation. In Leen, T.; Dietterich, T.; and Tresp, V., eds., *Advances in Neural Information Processing Systems 13*. Cambridge, MA: MIT Press. 689–695.