Leveraging Ontologies for Lifted Probabilistic Inference and Learning

Chloé Kiddon and **Pedro Domingos**
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350, U.S.A.
{*chloe, pedrod*}*@cs.washington.edu*

Abstract

Exploiting ontologies for efficient inference is one of the most widely studied topics in knowledge representation and reasoning. The use of ontologies for probabilistic inference, however, is much less developed. A number of algorithms for lifted inference in first-order probabilistic languages have been proposed, but their scalability is limited by the combinatorial explosion in the sets of objects that need to be considered. We propose a coarse-to-fine inference approach that leverages a class hierarchy to combat this problem. Starting at the highest level, our approach performs inference at successively finer grains, pruning low-probability atoms before refining. We provide bounds on the error incurred by this approach relative to full ground inference as a function of the pruning threshold. We also show how to learn parameters in a coarse-to-fine manner to maximize the opportunities for pruning during inference. Experiments on link prediction and biomolecular event prediction tasks show our method can greatly improve the scalability of lifted probabilistic inference.

Introduction

Inference in AI problems is generally intractable. One way to make it (more) tractable is to exploit ontological information (Staab and Studer 2004). This has been studied extensively, but almost entirely in the context of purely logical inference. However, the need for it is arguably even greater in probabilistic inference. Most widely used probabilistic representations are propositional, but in the last decade many first-order probabilistic languages have been proposed (Getoor and Taskar 2007). Initially, inference in these languages was carried out by first converting to propositional form, but more recently algorithms for lifted inference have been developed (Poole 2003; de Salvo Braz, Amir, and Roth 2007; Singla and Domingos 2008; Kersting, Ahmadi, and Natarajan 2009; Kisynski and Poole 2009). While lifting can yield very large speedups over propositionalized inference, the blowup in the combinations of objects and relations that have to be considered still greatly limits its applicability. One solution is to perform approximate lifting, by grouping objects that behave similarly, even if they are not exactly alike (Singla 2009; Sen, Deshpande, and Getoor 2009; de Salvo Braz et al. 2009).

Ontologies provide a natural hierarchical grouping of similar objects. We propose a lifted inference method that exploits them by first performing inference at the coarsest level, pruning atoms with probabilities close to 0 or 1 by treating them as certain, performing inference at the next finer level, and repeating until the finest level is reached or all atoms are pruned. This will give good results as long as the dependences in the domain correlate well with the ontology structure, which proper design of the latter should ensure. We provide bounds on the approximation error as a function of the pruning threshold and model parameters. We also propose a parameter learning algorithm that uses the lower levels of an ontology to refine the parameters at the higher levels, maximizing the gains from hierarchical lifting.

Our algorithms are formulated in terms of Markov logic (Domingos and Lowd 2009), but should be applicable to essentially any first-order probabilistic language. They can be viewed as a generalization of the ideas in coarse-to-fine parsing (Petrov and Klein 2007). Hierarchical inference has also been used in vision (e.g., Felzenszwalb and Huttenlocher 2006). Hierarchical models are widespread in machine learning and statistics, but almost entirely in the context of propositional representations (Gelman and Hill 2006). They have also been used in first-order domains (e.g., Pfeffer et al. (1999)). Interest in probabilistic ontologies for the Semantic Web is growing, but most work to date focuses on representation issues or specific applications (Costa et al. 2008). Our approach incorporates many of the advantages of lazy inference (Poon, Domingos, and Sumner 2008). It is also related to non-monotonic reasoning (Pearl (1988), Ch. 10).

We begin with some necessary background and then present the algorithm and theoretical results. We then report our experiments on two real-world domains (a social network one and a molecular biology one). These show that our approach can indeed be highly effective compared to previous methods like propositional and lifted belief propagation.

Background

Belief Propagation

Graphical models compactly represent the joint distribution of a set of variables $\mathbf{X} = (X_1, X_2, \ldots, X_n) \in \mathcal{X}$ as a product of factors (Pearl 1988): $P(\mathbf{X}=\mathbf{x}) = \frac{1}{Z} \prod_k f_k(\mathbf{x}_k)$,

where each factor f_k is a non-negative function of a subset of the variables $\mathbf{x}_k$, and Z is a normalization constant. Under appropriate restrictions, the model is a *Bayesian network* and $Z = 1$. A *Markov network* or *Markov random field* can have arbitrary factors. If $P(\mathbf{X}=\mathbf{x}) > 0$ for all x, the distribution can be equivalently represented as a *log-linear model*: $P(\mathbf{X}=\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_i w_i g_i(\mathbf{x})\right)$, where the *features* $g_i(\mathbf{x})$ are arbitrary functions of (a subset of) the state. The *factor graph* representation of a graphical model is a bipartite graph with a node for each variable and factor in the model (Kschischang, Frey, and Loeliger 2001). (For convenience, we consider one factor $f_i(\mathbf{x}) = \exp(w_i g_i(\mathbf{x}))$ per feature $g_i(\mathbf{x})$, i.e., we do not aggregate features over the same variables into a single factor.) Undirected edges connect variables with the appropriate factors.

The main inference task in graphical models is to compute the conditional probability of some variables (the query) given the values of some others (the evidence), by summing out the remaining variables. If the graph is a tree, the marginal probabilities of the query variables can be computed in polynomial time by *belief propagation* (BP), which consists of passing messages from variable nodes to the corresponding factor nodes and vice-versa. The message from a variable x to a factor f is

$$\mu_{x \to f}(x) = \prod_{h \in nb(x) \setminus \{f\}} \mu_{h \to x}(x)$$

where $nb(x)$ is the set of factors x appears in. The message from a factor to a variable is

$$\mu_{f \to x}(x) = \sum_{\sim\{x\}} \left(f(\mathbf{x}) \prod_{y \in nb(f) \setminus \{x\}} \mu_{y \to f}(y) \right)$$

where $nb(f)$ are the arguments of f, and the sum is over all of these except x. The messages from leaf variables are initialized to 1, and a pass from the leaves to the root and back to the leaves suffices. The (unnormalized) marginal of each variable x is then given by $\prod_{h \in nb(x)} \mu_{h \to x}(x)$. Evidence is incorporated by setting $f(\mathbf{x}) = 0$ for incompatible states $\mathbf{x}$. This algorithm can still be applied when the graph has loops by repeating the message-passing until convergence. Although this *loopy* belief propagation has no guarantees of convergence or correctness, in practice it often does and can be much more efficient than other methods.

Markov Logic

First-order probabilistic languages combine graphical models with elements of first-order logic, by defining template features that apply to whole classes of objects at once. A simple and powerful such language is *Markov logic* (Richardson and Domingos 2006). A *Markov logic network (MLN)* is a set of weighted first-order clauses.[1] Together with a set of constants representing objects in the domain of interest, it defines a Markov network with one node per ground atom and one feature per ground clause. The weight of a feature is the weight of the first-order clause that originated it. The probability of a state $\mathbf{x}$ in such a network is given by $P(\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_i w_i g_i(\mathbf{x})\right) = \frac{1}{Z} \prod_i f_i(\mathbf{x})$, where w_i is the weight of the ith clause, $g_i = 1$ if the ith clause is true, and $g_i = 0$ otherwise.

Inference in Markov logic can be carried out by creating the ground network and applying belief propagation to it, but this can be extremely inefficient because the size of the ground network is $O(d^c)$, where d is the number of objects in the domain and c is the highest clause arity.

Lifted Belief Propagation

Lifted inference establishes a more compact version of the ground network in order to make inference more efficient. In *lifted belief propagation* (LBP), subsets of components in the ground network are identified that will send and receive identical messages during belief propagation (Singla and Domingos 2008). The subsets of indistinguishable ground atoms become the *supernodes* of the *lifted network* and the subsets of indistinguishable ground clause become the *superfeatures*. The size of the lifted network is $O(mn)$ where n is the number of supernodes and m is the number of superfeatures. In the best case, when no evidence is provided, all ground atoms will behave in the same way and the lifted network will have the same size as the MLN. In the worst case, the lifted network will be identical to the ground network. Standard belief propagation can be applied to the lifted network with minor changes to correct for duplicate messages lost in the lifting process. LBP is guaranteed to give the same results as BP. The process that constructs the lifted network simulates BP to determine equivalent component sets and is guaranteed to converge to a minimally lifted network. If the process is stopped before convergence, the result is an *approximate lifted network* (Singla 2009).

Coarse-to-Fine Inference and Learning

Representation

The standard definition of an MLN assumes an undifferentiated set of constants. We begin by extending it to allow for a hierarchy of constant types.

Definition 1 A *type* is a set of constants $t = \{k_1, \ldots, k_n\}$. A type t is a *subtype* of another type t' iff $t \subset t'$. A type t is a *supertype* of another type t' iff $t' \subset t$. A *refinement* of a type t is a set of types $\{t_1, \ldots, t_m\}$ such that $\forall_{i,j}\ t_i \cap t_j = \emptyset$ and $t = t_1 \cup t_2 \cup \cdots \cup t_m$.

Definition 2 A *typed predicate* is a tuple $a = (a_0, t_1, \ldots, t_n)$, where a_0 is a predicate, n is a_0's arity, and t_i is the type of a_0's ith argument. A *typed clause* is a tuple $c = (c_0, t_1, \ldots, t_n)$, where c_0 is a first-order clause, n is the number of unique variables in c_0, and t_i is the type of the ith variable in c_0. The set of types in a typed atom or clause is referred to as the atom's or clause's *type signature*.

Definition 3 A *typed MLN* $\mathbf{M}$ is a set of weighted typed clauses, $\{(c_i, w_i)\}$. It defines a ground Markov network with one atom for each possible grounding of each typed predicate in $\mathbf{M}$, and one feature for each possible grounding

[1] In this paper we assume function-free clauses and Herbrand interpretations.

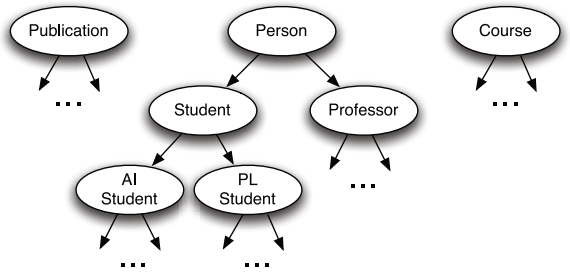

Figure 1: Example type hierarchy for an academia domain.

of each typed clause in $\mathbf{M}$ with constants from the corresponding types. The weight of a feature is the weight of the typed clause that originated it.

Definition 4 Given a set of types $\mathbf{T}$, $t_i \in \mathbf{T}$ is a *direct subtype* of $t_j \in \mathbf{T}$ iff $t_i \subset t_j$ and $\nexists_{t \in \mathbf{T}}\ t_i \subset t \subset t_j$. $\{t_1, t_2, \ldots, t_m\} \subset \mathbf{T}$ is a *direct refinement* of $t \in \mathbf{T}$ iff it is a refinement of t and $t_1, \ldots, t_m$ are direct subtypes of t. A set of types $\mathbf{T}$ is a *type hierarchy* iff within $\mathbf{T}$, each type has no subtypes or exactly one direct refinement, and $\forall_{i,j}\ (t_i \cap t_j = \emptyset) \vee (t_i \subset t_j) \vee (t_j \subset t_i)$. A *root type* has no supertypes; a *leaf type* has no subtypes.

A type hierarchy is a forest of types. It may be a tree, but an all-encompassing root type will usually be too general to be useful for inference. An example type hierarchy is given in Figure 1 for an academia domain.

Inference

We assume that full type information for all objects is known. That is, given a type hierarchy $\mathbf{T}$, each object is assigned a set of types $\{t_1, \ldots, t_n\} \subset \mathbf{T}$ where t_1 is a root type, t_i is a direct subtype of t_{i-1} for all $i > 1$, and t_n is a leaf type.

Algorithm 1 shows pseudocode for the OLPI algorithm (Ontological Lifted Probabilistic Inference). It takes as input a type hierarchy $\mathbf{T}$, a typed MLN $\mathbf{M_T}$ over types in $\mathbf{T}$, a database of evidence $\mathbf{E}$, and a pruning threshold γ. OLPI begins by choosing an MLN $\mathbf{M}$ containing the weighted clauses in $\mathbf{M_T}$ whose type signatures are composed exclusively of the highest level types chosen for consideration. These could be root types or a set of types from any cut of the type hierarchy. For example, in an academia domain, it may make more sense to consider students and professors separately from the start. OLPI then calls a pre-specified lifted probabilistic inference algorithm to compute the marginals of all the non-evidence atoms based on $\mathbf{M}$, the constants in $\mathbf{T}$ and the evidence $\mathbf{E}$. Atoms whose marginal is at most γ are added to the evidence as false, and atoms whose marginal is at least $1-\gamma$ are added as true. The marginal probabilities of the pruned nodes are stored and returned in the output of OLPI. Any clauses now valid or unsatisfiable given the expanded evidence will not affect the results of subsequent inferences, and are removed from $\mathbf{M}$.

OLPI then refines $\mathbf{M}$, replacing every clause c in $\mathbf{M}$ with the set of clauses obtained by direct refinement of the types

Algorithm 1 Ontological Lifted Probabilistic Inference

inputs: $\mathbf{M_T}$, a typed Markov logic network
$\mathbf{T}$, a type hierarchy
$\mathbf{E}$, a set of ground literals
γ, pruning threshold
calls: *Infer()*, a probabilistic inference algorithm
Refine(), a type refinement algorithm
$\mathbf{M} \leftarrow$ *Coarsest*$(\mathbf{M_T})$
repeat
  $P(\mathbf{x}|\mathbf{E}) \leftarrow$ *Infer*$(\mathbf{M}, \mathbf{T}, \mathbf{E})$
  for each atom x_i
    if $P(x_i|\mathbf{E}) \leq \gamma$ **then** $\mathbf{E} \leftarrow \mathbf{E} \cup \{\neg x_i\}$
    else if $P(x_i|\mathbf{E}) \geq 1-\gamma$ **then** $\mathbf{E} \leftarrow \mathbf{E} \cup \{x_i\}$
  $\mathbf{M} \leftarrow \mathbf{M} \backslash \{$valid and unsatisfiable clauses under $\mathbf{E}\}$
  $\mathbf{M} \leftarrow$ *Refine*$(\mathbf{M}, \mathbf{M_T})$
until *Refine*$(\mathbf{M}, \mathbf{M_T}) = \mathbf{M}$
$P(\mathbf{x}|\mathbf{E}) \leftarrow$ *Infer*$(\mathbf{M}, \mathbf{T}, \mathbf{E})$

in c's type signature. If v is a variable in c, v's type in a refined clause is a direct subtype of its type in c, and there is a refined clause for each possible combination of direct subtypes for the variables in c. Any leaf types are left unrefined. In general, it might be useful to refine some types and leave others unrefined, but this substantially increases the complexity of the algorithm and is left for future work. The clauses returned are the *direct clause refinements* of the clause c. The process ends when no more direct clause refinements are possible on the clauses in $\mathbf{M}$ or all atoms have been pruned; in either case, *Refine*$(\mathbf{M}, \mathbf{M_T})$ returns $\mathbf{M}$.

At every step, the MLN grows by refining clauses, but also shrinks by pruning. The goal is to contain the complexity of inference, while keeping it focused on where it is most needed: the atoms we are most uncertain about. The following theorem gives bounds on the approximation error incurred by this process, relative to using the full typed MLN $\mathbf{M_T}$ for inference. (Proofs are included in the full version of this paper (Kiddon and Domingos 2010).)

Theorem 1 *Let γ be the OLPI pruning threshold, ϵ the maximum error in probabilities computed by Infer(), n the total number of ground atoms, n_i the number of ground atoms pruned at level i, w_i the maximum error in weights at level i, l the level at which OLPI stops, and δ the average error in the marginals returned by OLPI. Then*

$$\delta \leq \frac{(\gamma+\epsilon)(2n_l+1)}{n} \sum_{i=1}^{l-1} \frac{n_i(l-i)}{\gamma+\epsilon+(1-\gamma-\epsilon)e^{-2w_i}}.$$

Infer() can be any lifted probabilistic inference algorithm (or even propositionalization followed by ground inference, although this is unlikely to scale even in the context of OLPI). However, realistic domains generally require approximate inference. In this paper we use lifted belief propagation (Singla and Domingos 2008). We call OLPI with lifted BP as the inference algorithm OBP (Ontological Belief Propagation). We now provide an error bound for OBP. Since lifted BP computes the same marginals as ground BP, for proof purposes it can be treated as the latter. We can view the errors in the messages passed during

BP in level k of OBP as multiplicative errors on the messages from factors to nodes at each step of BP, due to weight approximations at that level and the loss of pruned nodes.

Theorem 2 *For a binary node x, the probability estimated by BP at convergence over the network at level k (p_x^k) can be bounded as follows in terms of the probability estimated by OBP ($\hat{p}_x^k$) after n iterations of BP where σ^- and σ^+ are the sets of low- and high-probability nodes pruned in OBP's previous $k-1$ runs of BP and γ is the pruning threshold:*
For $x \in \sigma^-$: $0 \leq p_x^k \leq \gamma$
For $x \in \sigma^+$: $1-\gamma \leq p_x^k \leq 1$
And for $x \notin \sigma^- \cup \sigma^+$:

$$p_x^k \geq \frac{1}{(\xi_x^{k,n})^2[(1/\hat{p}_x^k)-1]+1} = lb(p_x^k)$$

$$p_x^k \leq \frac{1}{(1/\xi_x^{k,n})^2[(1/\hat{p}_x^k)-1]+1} = ub(p_x^k)$$

where $\log \xi_x^{k,n} = \sum_{f \in nb(x)} \log \nu_{f,x}^{k,n}$, $\nu_{f,x}^{k,1} = d(f)^2$, $\nu_{f,x}^{k,n}$ is defined in the recursive fashion of Theorem 15 of Ihler et al. (2005), the dynamic range of the outgoing error from a factor to a node is:

$$d(\varepsilon_{f,x}^k) = \gamma^{-\frac{1}{2}|\sigma_f^-|}(1-\gamma)^{-\frac{1}{2}|\sigma_f^+|}e^{\frac{1}{2}\alpha_f^k},$$

and nodes are only pruned at level k' when $ub(p_x^{k'}) \leq \gamma$ or $lb(p_x^{k'}) \geq 1-\gamma$.

If no nodes have been pruned at previous levels, the fixed point beliefs returned from OBP on its kth level of BP after n iterations will be equivalent to those returned by BP after n iterations on the network at that level.

OBP can be made more efficient in a number of ways: the factor graph can be refined directly from the previous run instead of being recreated from scratch at each level; approximate lifted network construction through early stopping can be used to defer exact lifting until more pruning occurs.

Learning

Stronger weights on clauses used in earlier levels of OLPI allows for earlier pruning decisions which speeds up later iterations of inference. To achieve models of this type, we learn weights in a coarse-to-fine manner through a series of successive refinements of clauses guided by a given type hierarchy. At each iteration of learning, we fix all weights learned in preceding iterations, add in all possible direct clause refinements, and then learn the weights for these new clauses. The effect is that the weight learned for a typed clause $c_{\mathbf{t}}$ at iteration i that was created by a refinement of clause $c_{\mathbf{t}'}$ in iteration $i-1$ is the additional weight given to a ground clause based on having that extra type information. The learned weights should become successively smaller as the more refined type information becomes less important. Once this occurs, we do not refine the clause any further. The result of this is a sparser model which will correspond to fewer possible refinements during the inference process and therefore more efficient inference.

Proposition 1 *For a typed MLN $\mathbf{M_T}$ learned in the coarse-to-fine framework, there is an equivalent typed MLN $\mathbf{M'_T}$ where no clause $c \in \mathbf{M'_T}$ can be obtained through a series of direct clause refinements of any other clause $c' \in \mathbf{M'_T}$.*

The weight of the feature for a ground clause in $\mathbf{M'_T}$ will be the sum of the weights for all features in $\mathbf{M_T}$ defined for the same ground clause. Therefore, the weights learned for a coarse-to-fine typed MLN $\mathbf{M_T}$ are equivalent to a *type-flattened* MLN $\mathbf{M'_T}$. When *Refine*$(\mathbf{M}, \mathbf{M_T})$ replaces a clause c in $\mathbf{M}$ by the set of clauses from $\mathbf{M_T}$, $c'_1, c'_2, \ldots, c'_n$, obtained by direct refinement of the types in c's type signature, the weight of each new typed clause c'_i added to $\mathbf{M}$ is actually $w + w'_i$, where w is the weight of c in $\mathbf{M}$ and w'_i is the weight of c'_i in $\mathbf{M_T}$. When no more refinements are possible, the resulting typed MLN will be a subset of $\mathbf{M'_T}$, accounting for pruned clauses.

Experiments

We experimented on a link prediction task in a social networking domain and an event prediction task in a molecular biology domain to compare the running time and accuracy of OBP and LBP. We implemented OBP as an extension of the open-source Alchemy system (Kok et al. 2007). Currently Alchemy does not allow for duplicate clauses with different type signatures. Instead we added type predicates to the formulas in our model to denote the correct type signatures. We compared running OBP over a typed MLN to running LBP over the equivalent type-flattened MLN. We ran each algorithm until either it converged or the number of iterations exceeded 100. We did not require each algorithm to run for the full 100 iterations since the network shrinkage that occurs with OBP may allow it to converge faster and is an integral part of its efficiency that should not be penalized.

Link Prediction

The ability to predict connections between objects is very important in a variety of domains such as social network analysis, bibliometrics, and micro-biology protein interactions. We experimented on the link prediction task of Richardson & Domingos (2006), using the UW-CSE database that is publicly available on the Alchemy project website.[2] The task is to predict the *AdvisedBy* relation given evidence on teaching assignments, publication records, etc. We created a type hierarchy that corresponded well to the objects in the domain. The *Person* type is split into a *Professor* and a *Student* type, both of which are split further by area (e.g., AI, Graphics); the *Student* type is split further by point in the graduate program (e.g., Pre-Quals, Post-Generals). The *Class* type is split by area followed by level. We tested on 43 of the 94 formulas in the UW-CSE MLN; formulas with existential quantifiers were removed and after the removal of "type" predicates such as *Student(x)* and *Professor(x)*, all duplicate formulas were discarded. The type-flattened MLN had 10,150 typed clauses from matching the 43 formulas with varying type signatures. The full database contains ∼4,000 predicate groundings including type predicates. To evaluate inference over different numbers of objects in the domain, we randomly selected graph cuts of various sizes from the domain. Figure 2 shows a comparison of

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

Algorithm	Pruning Threshold	Init (sec)	Infer (sec)	Prune (sec)	Avg. CLL	# Superfeatures
LBP	N/A	3441.59	2457.34	N/A	-0.00433	8.2 million
OBP	0.001	2172.45	208.59	0.99	-0.00433	485,507
OBP	0.01	537.93	2.08	1.15	-0.00431	10,328

Table 1: Results for the full UW-CSE data set. For OBP, number of superfeatures counts the most used during any level.

Algorithm	Pruning Threshold	Init (sec)	Infer (sec)	Prune (sec)	Avg. CLL	# Superfeatures
LBP	N/A	1305.08	846.21	N/A	-0.01062	8.5 million
OBP	0.01	415.31	0.40	0.36	-0.01102	3,478

Table 2: Results for GENIA over 150 abstracts. For OBP, number of superfeatures counts the most used during any level.

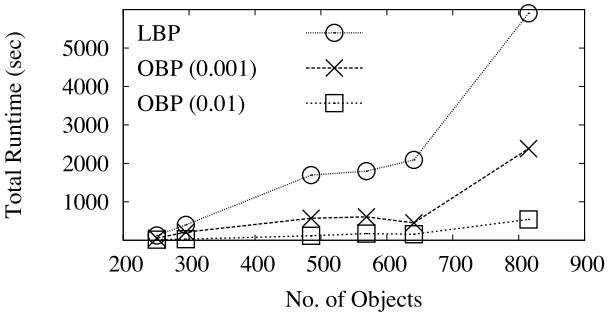

Figure 2: Total runtime of algorithms over UW CSE.

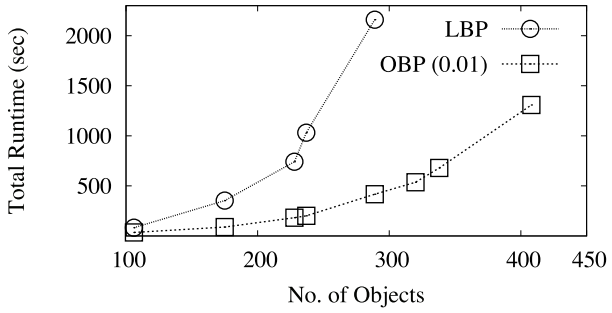

Figure 3: Total runtime of algorithms over GENIA.

the runtimes of OBP and LBP for different sized cuts of the UW-CSE data set. We ran OBP with pruning thresholds of $\gamma = 0.01$ and $\gamma = 0.001$. The time represents the sum of both initialization of the network and the inference itself; the times for OBP also include the refinement times after each level. For each cut of the UW-CSE data set, the average conditional log likelihood (CLL) of the results returned by OBP with either pruning threshold were virtually the same as the average conditional log likelihood returned by LBP. Table summarizes the results of the UW-CSE link prediction experiment over the full UW CSE data set. The full data set contained 815 objects, including 265 people, and 3833 evidence predicates. With $\gamma = 0.01$, we achieve an order of magnitude speedup.

Biomolecular Event Prediction

As new biomedical literature continues to accumulate at a rapid pace, text mining systems tailored to the domain of molecular biology are becoming increasingly important. One important task is the identification and extraction of biomolecular events from text. Event prediction is a very challenging task (Kim et al. 2003), and is not the focus of this paper. Our simplified task is to predict which entities are the causes and themes of identified events contained in the text, represented by two predicates: *Cause(event,entity)* and *Theme(event,entity)*. We used the GENIA event corpus which marks linguistic expressions that identify biomedical events in scientific literature spanning 1,000 Medline abstracts; there are 36,114 events labeled, and the corpus contains a full type hierarchy of 32 entity types and 28 event types (Kim, Ohta, and Tsujii 2008). Our features included semantic co-occurrence and direct semantic dependencies with a set of key stems (e.g., *Subj(entity, stem, event)*). Global features learned the roles that certain entities tend to fill. We used the Stanford Parser,[3] for dependency parsing and a Porter stemmer to identify key stems.[4] We restricted our focus to events with one cause and one theme or no cause and two themes where we could extract interesting semantic information at our simple level. The model was learned over half the GENIA event corpus and tested on the other half; abstract samples of varying sizes were randomly generated. From 13 untyped clauses, the type-flattened MLN had 38,020 clauses.

Figure 3 shows a comparison of the runtimes of OBP with $\gamma = 0.01$ and LBP. For each test set where both OBP and LBP finished, the average conditional log likelihoods were almost identical. The largest difference in average conditional log likelihood was 0.019 with a dataset of 175 objects; in all other tests, the difference between the averages was never more than 0.001. Table summarizes the results of the the largest GENIA event prediction experiment where both LBP and OBP finished without running out of memory. This test set included 125 events and 164 entities.

[3] http://nlp.stanford.edu/software/lex-parser.shtml

[4] http://tartarus.org/∼martin/PorterStemmer

Conclusion and Future Work

We presented a method for scaling up lifted probabilistic inference, by performing it at successively finer levels of an ontology while pruning nodes with probabilities close to 0 or 1 at each level. We provided bounds on the approximation error incurred in this way. We also proposed a simple weight learning method that maximizes the gains obtainable by this type of inference. Experiments on two domains show the benefits of our approach. Directions for future work include: learning ontology refinements from data for use in OLPI; broadening the types of ontological structure allowed by OLPI (e.g., multiple inheritance, partly unknown type information); applying OLPI to other lifted probabilistic inference algorithms besides LBP; further scaling up OLPI (e.g., using more compact data structures), etc.

Acknowledgements

This research was partly funded by ARO grant W911NF-08-1-0242, AFRL contract FA8750-09-C-0181, DARPA contracts FA8750-05-2-0283, FA8750-07-D-0185, HR0011-06-C-0025, HR0011-07-C-0060 and NBCH-D030010, NSF grants IIS-0534881 and IIS-0803481, and ONR grant N00014-08-1-0670. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, DARPA, NSF, ONR, or the United States Government.

References

Costa, P. C. G.; Ladeira, M.; Carvalho, R. N.; Laskey, K. B.; Santos, L. L.; and Matsumoto, S. 2008. A first-order bayesian tool for probabilistic ontologies. In *Proc. FLAIRS–08*, 631–636.

de Salvo Braz, R.; Amir, E.; and Roth, D. 2007. In L. Getoor and B. Taskar, ed., Lifted first-order probabilistic inference. In Getoor, L., and Taskar, B., eds., *Introduction to Statistical Relational Learning*, 433–450. MIT Press.

de Salvo Braz, R.; Natarajan, S.; Bui, H.; Shavlik, J.; and Russell, S. 2009. Anytime lifted belief propagation. In *Proc. SRL–09*.

Domingos, P., and Lowd, D. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan Kaufmann.

Felzenszwalb, P. F., and Huttenlocher, D. P. 2006. Efficient belief propagation for early vision. *International Journal of Computer Vision* 70(1): 41–54.

Gelman, A., and Hill, J. 2006. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press.

Getoor, L., and Taskar, B., eds. 2007. *Introduction to Statistical Relational Learning*. MIT Press.

Ihler, A. T.; III, J. W. F.; and Willsky, A. S. 2005. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research* 6: 905–936.

Kersting, K.; Ahmadi, B.; and Natarajan, S. 2009. Counting belief propagation. In *Proc. UAI–09*, 277–284.

Kiddon, C., and Domingos, P. 2010. Lifted probabilistic inference and learning over a hierarchy of classes. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA.

Kim, J.-D.; Ohta, T.; Tateisi, Y.; and Tsujii, J. 2003. GENIA corpus–semantically annotated corpus for bio-textmining. *BMC Bioinformatics* 19(1): 180–182.

Kim, J.-D.; Ohta, T.; and Tsujii, J. 2008. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics* 9(1): 10.

Kisynski, J., and Poole, D. 2009. Lifted aggregation in directed first-order probabilistic models. In *Proc. IJCAI–09*, 1922–1929.

Kok, S.; Sumner, M.; Richardson, M.; Singla, P.; Lowd, H. P. D.; and Domingos, P. 2007. 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.

Kschischang, F. R.; Frey, B. J.; and Loeliger, H.-A. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* 47: 498–519.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Petrov, S., and Klein, D. 2007. Learning and inference for hierarchically split PCFGs. In *Proc. AAAI–07*, 1663-1666.

Pfeffer, A.; Koller, D.; Milch, B.; and Takusagawa, K. T. 1999. SPOOK: A system for probabilistic object-oriented knowledge representation. In *Proc. UAI–99*, 541–550.

Poole, D. 2003. First-order probabilistic inference. In *Proc. IJCAI–03*, 985–991.

Poon, H.; Domingos, P.; and Sumner, M. 2008. A general method for reducing the complexity of relational inference and its application to MCMC. In *Proc. AAAI–08*, 1075–1080.

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

Sen, P.; Deshpande, A.; and Getoor, L. 2009. Bisimulation-based approximate lifted inference. In *Proc. UAI–09*, 496–505.

Singla, P., and Domingos, P. 2008. Lifted first-order belief propagation. In *Proc. AAAI-08*, 1094–1099.

Singla, P. 2009. *Markov Logic: Theory, Algorithms and Applications*. PhD in Computer Science & Engineering, University of Washington, Seattle, WA.

Staab, S., and Studer, R. 2004. *Handbook on Ontologies (International Handbooks on Information Systems)*. SpringerVerlag.