Approximate Counting for Fast Inference and Learning in Probabilistic Programming

Mayukh Das*, Devendra Singh Dhami*, Gautam Kunapuli*, Kristian Kersting†, Sriraam Natarajan*

*The University of Texas at Dallas, †TU Darmstadt;
*{mayukh.das1, devendra.dhami, gautam.kunapuli, sriraam.natarajan}@utdallas.edu, †kersting@cs.tu-darmstadt.de

Abstract
Inference and Parameter Learning inside Probabilistic Programming use an important operation that can be approximated: counting. We present an efficient approximation scheme that allows for fast counting and consequently faster inference and learning.

Keywords Scaling, Approximate Counting, Hypergraph

1 Introduction
Learning and inference in Probabilistic Programming and related Statistical Relational Learning (SRL) is difficult, particularly due to one major bottleneck – counting over supporting instances of partial programs (partially instantiated clauses/rules in probabilistic logic representation), [4, 7, 8]. Since counting is hard (#P-complete), learning and inference becomes intractable for large data sets. But for most learning/inference tasks, exact counts are unnecessary in presence of relatively high number of instances and hence can be approximated. For example, there will not be any significant change in the belief over the popularity of a Professor, based on whether (s)he has 400 or 427 publications.

Several recent approaches for fast, approximate counting [3, 9], while reasonably successful, have major limitations due to some restrictive assumptions – including restricted arity [3] or MLN-specific algorithms and lack of support for partial groundings [9]. Our approach, Motif-based Approximate Counting via Hypergraphs (MACH), is a generalized count approximation technique enabling scalable counting of partial programs.

2 Approximate Counting via Hypergraphs
Our approach for computation of satisfied instances (true groundings) of rules (in First Order Logic), exploits its equivalence with the problem of counting matching subgraphs. However, subgraph matching is hard problem in itself (#P-complete) and Das et al. [3] shows how pre-computed graph statistics followed by a message passing strategy, results in a reasonable approximation in large (dense) data sets. While effective, their formulation is empirical at best and lacks theoretical basis. MACH provides a fundamentally robust formulation using Hypergraphs.

Compilation: A hypergraph [1] is a generalization of a graph. A hyperedge can connect an arbitrary number of vertices and thus, can faithfully represent relations/predicates of any arity, as opposed to an ordinary edge (limited to binary). The key idea is to transform the data (set of facts/ground literals) into a hypergraph \( G \) and a FOL clause (whose \#instances we are interested in) to a hypergraph motif \( M \), termed herein as a Partially Grounded Structural Motif (PGSM). We then (approximately) count the sub(hyper)graphs in \( G \) that match the pattern \( M \). The transformation creates vertices from the arguments of a predicate and the predicate/relation itself forms a partially-directed labeled hyperedge. ”Partially Grounded” indicates that some of the terms in \( M \), may be instantiated. However, all vertices/nodes in \( G \) are grounded.

Example. A FOL clause \( C_1 = \text{AdvisedBy}(s, p) \land \text{TA}(s, c, t) \land \text{Teaches}(p, c, t) \) transforms into a PGSM \( M_1 \) as shown in Figure 1. Hypergraph \( G_1 \) is created from the facts. Note, there are only 2 instances in \( G_1 \) that match \( M_1 \) (purple shaded regions)

Approximation: The number of subgraphs matching a motif \( M \) can be expressed as the expected vertex product of \( M = (V_M, E_M) \) given \( G \).

\[
P(M[G]) = \prod_{v \in V_M} P(e[G]) \prod_{v \in V_M} n(v)
\]

where, \( n(v) \), the typecount, is the #entities in the domain of the variable \( v \). Ex: In Figure 1, \( n(s) = 5 \) as there are 5 students.

If it is grounded (\( v = C \) ), \( n(v) = 1 \lor 0 \) based on existence of \( C \) in \( G \). \( P(e[G]) \) represents the local model of a hyperedge (regarded as a random variable) in \( M \). Thus, the joint model of the motif \( M \) is factorizable into a product over the local models. The second part \( \prod_{v \in V_M} n(v) \) of equation 1 is straightforward and is the Cartesian product over all the variables in a clause. The first part \( P(M[G]) \), is what we are interested in estimating. If the distribution can be computed exactly, eqn. 1 will return the exact counts. However, exact

Figure 1. (left) Motif \( M_1 \) for \( C_1 \); (middle) Facts in the world; (right) Ground hypergraph, \( G_1 \). Ternary predicates Teaches and TA are represented as hyperedges, via blue dashed lines and red solid lines respectively, in \( M_1 \) and \( G_1 \).
We evaluate MACH based on 2 experimental questions, (Q1) Is MACH effective and efficient in full model learning with n-ary relations? and (Q2) Is faithful modeling of n-ary relations crucial? MACH is implemented as a pluggable module using Java-based HypergraphDB architecture [6]. We integrated it into the state-of-the-art Boosted MLN framework [7] for full model (structure+parameters) learning. MACH is compared against the following baselines, (1) FACT [3] and (2) MLN-Boost (Boosted MLN learner w/o approximate counting).

We evaluate with 2 data sets, UWCSE, standard link-prediction data set over staff, faculty and students and NELL-Sports, a sports domain data set extracted from NELL [2]. Table 1 shows how MACH allows significant reduction in learning time with no deterioration in predictive performance (AUC-ROC & PR) answering (Q1) affirmatively. Note that, UWCSE contains ternary predicates which were not modeled correctly by FACT leading to significantly worse performance, corroborating the need for faithful representation of n-ary relations (Q2).

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>Methods</th>
<th>Performance (AUC)</th>
<th>Efficiency (L-Time [s])</th>
</tr>
</thead>
<tbody>
<tr>
<td>UWCSE**</td>
<td>MACH</td>
<td>0.981</td>
<td>0.337</td>
</tr>
<tr>
<td></td>
<td>FACT</td>
<td>0.500</td>
<td>0.0068</td>
</tr>
<tr>
<td></td>
<td>MLN-Boost</td>
<td>0.736</td>
<td>0.361</td>
</tr>
<tr>
<td>NELL-Sports</td>
<td>MACH</td>
<td>0.78</td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td>FACT</td>
<td>0.76</td>
<td>0.64</td>
</tr>
<tr>
<td></td>
<td>MLN-Boost</td>
<td>0.78</td>
<td>0.66</td>
</tr>
</tbody>
</table>

Table 1: Results: Performance (AUC) vs. Efficiency (Learning time in seconds). ** denotes presence of n-ary predicates.

4 Conclusion
We present a generalized and robust count approximation approach MACH for scalable and efficient probabilistic relational learning. Theoretical guarantees on error as well as a unifying framework integrating domain reduction and approximation are some interesting future research directions.

References