Large-Scale Learning and Inference: What We Have Learned with Markov Logic Networks

Pedro Domingos
Dept. of Computer Science & Eng.
University of Washington

Joint work with Jesse Davis, Stanley Kok, Daniel Lowd, Hoifung Poon, Matt Richardson, Parag Singla, Marc Sumner
Markov Logic Networks

- Basic idea: Use first-order logic to compactly specify large non-i.i.d. models
- **MLN** = Set of formulas with weights
- Formula = Feature template (Vars→Objects)
- E.g., social network:

\[
\text{Smokes}(x) \land \text{Friends}(x, y) \Rightarrow \text{Smokes}(y)
\]

\[
P(x) = \frac{1}{Z} \exp \left( \sum_i w_i n_i(x) \right)
\]

- Weight of formula \(i\)
- No. of true instances of formula \(i\) in \(x\)
## MLN Algorithms: The First Three Generations

<table>
<thead>
<tr>
<th>Problem</th>
<th>First generation</th>
<th>Second generation</th>
<th>Third generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP inference</td>
<td>Weighted satisfiability</td>
<td>Lazy inference</td>
<td>Cutting planes</td>
</tr>
<tr>
<td>Marginal inference</td>
<td>Gibbs sampling</td>
<td>MC-SAT</td>
<td>Lifted inference</td>
</tr>
<tr>
<td>Weight learning</td>
<td>Pseudo-likelihood</td>
<td>Voted perceptron</td>
<td>Scaled conj. gradient</td>
</tr>
<tr>
<td>Structure learning</td>
<td>Inductive logic progr.</td>
<td>ILP + PL (etc.)</td>
<td>Clustering + pathfinding</td>
</tr>
</tbody>
</table>
Weighted Satisfiability

- **SAT**: Find truth assignment that makes all formulas (clauses) true
  - Huge amount of research on this problem
  - State of the art: Millions of vars/clauses in minutes
- **MaxSAT**: Make as many clauses true as possible
- **Weighted MaxSAT**: Clauses have weights; maximize satisfied weight
- MAP inference in MLNs is just weighted MaxSAT
- Best current solver: MaxWalkSAT
Lazy Inference

- Exploits sparseness
- In most domains, most atoms (random variables) are false
  E.g.: \texttt{Friends(x,y)}
- As a result, most clauses are trivially true
  E.g.: \texttt{Smokes(x) \land Friends(x,y) \Rightarrow Smokes(y)}
- Materialize only atoms and clauses with non-default values
- Start with evidence and extend lazily
- Vastly reduces time and memory
Cutting Plane Inference

- Basic idea:
  - Solve problem with small subset of constraints
  - Add violated constraints and repeat
  - Redundant constraints $\rightarrow$ Small fraction suffices

- In Markov logic, violated constraints are false clauses ( = true conjunctions)

- Use database queries to efficiently find them

- Applicable with any base solver, not just LP

- Much more scalable than Sontag & Jaakkola (2008)
MC-SAT

- Deterministic dependences break MCMC
- In practice, even strong probabilistic ones do
- **Swendsen-Wang:**
  - Introduce aux. vars. $u$ to represent constraints among $x$
  - Alternately sample $u \mid x$ and $x \mid u$.
- But Swendsen-Wang only works for Ising models
- **MC-SAT**: Generalize S-W to arbitrary clauses
- Uses SAT solver to sample $x \mid u$.
- Orders of magnitude faster than Gibbs sampling, etc.
Lifted Inference

- Consider belief propagation (BP)
- Often in large problems, many nodes are *interchangeable*: They send and receive the same messages throughout BP
- Basic idea: Group them into *supernodes*, forming *lifted* network
- Smaller network $\rightarrow$ Faster inference
Belief Propagation

\[
\mu_{x \rightarrow f}(x) = \prod_{h \in n(x) \setminus \{f\}} \mu_{h \rightarrow x}(x)
\]

\[
\mu_{f \rightarrow x}(x) = \sum_{\sim \{x\}} \left( e^{wf(x)} \prod_{y \in n(f) \setminus \{x\}} \mu_{y \rightarrow f}(y) \right)
\]
Lifted Belief Propagation

\[
\mu_{x \rightarrow f}(x) = \prod_{h \in n(x) \setminus \{f\}} \mu_{h \rightarrow x}(x)
\]

\[
\mu_{f \rightarrow x}(x) = \sum_{\sim\{x\}} \left( e^{w_f(x)} \prod_{y \in n(f) \setminus \{x\}} \mu_{y \rightarrow f}(y) \right)
\]
Lifted Belief Propagation

\( \alpha,\beta : \) Functions of edge counts

\[
\mu_{x \rightarrow f}(x) = \beta \prod_{h \in n(x) \setminus \{f\}} \mu_{h \rightarrow x}(x)
\]

\[
\mu_{f \rightarrow x}(x) = \sum_{\sim \{x\}} \left( e^{w_f(x)} \prod_{y \in n(f) \setminus \{x\}} \mu_{y \rightarrow f}(y) \right)
\]
Forming the Lifted Network

1. Form initial supernodes
   One per predicate and truth value
   (true, false, unknown)

2. Form superfeatures by doing joins of their supernodes

3. Form supernodes by projecting superfeatures down to their predicates
   Supernode = Groundings of a predicate with same number of projections from each superfeature

4. Repeat until convergence
Weight Learning

- Pseudo-likelihood + L-BFGS is fast and robust but can give poor inference results
- Voted perceptron:
  Gradient descent + MAP inference
- Problem: Multiple modes
  - Not alleviated by contrastive divergence
  - Alleviated by MC-SAT
  - Start each MC-SAT run at previous end state
Weight Learning (contd.)

- Problem: Extreme ill-conditioning
- Solvable by quasi-Newton, conjugate gradient, etc.
- But line searches require exact inference
- Stochastic gradient not applicable because data not i.i.d.
- Solution: Scaled conjugate gradient
- Use Hessian to choose step size
- Compute quadratic form inside MC-SAT
- Use inverse diagonal Hessian as preconditioner
Structure Learning

- Standard inductive logic programming optimizes the wrong thing
- But can be used to overgenerate for L1 pruning
- Our approach: ILP + Pseudo-likelihood + Structure priors
- For each candidate structure change: Start from current weights & relax convergence
- Use subsampling to compute sufficient statistics
- Search methods: Beam, shortest-first, etc.
Standard search methods generate a huge number of useless candidates

**Relational pathfinding:** Find paths of true atoms between objects & generalize them

**Relational clustering:** Cluster objects with similar relations to similar objects

**Clustering + Pathfinding:** Cluster objects, find paths between clusters & extract rules
MLNs: The Next Generation

- Weighted model counting
- Compilation to arithmetic circuits/BDDs/etc.
- Approximate compilation
- Generalized, approximate, incremental lifting
- Coarse-to-fine inference and learning
- Bottom-up learning of MLN structure
- Learning tractable high-treewidth MLNs
- Learning from data streams
- Etc.
Resources

- Open-source software/Web site: Alchemy
  - Learning and inference algorithms
  - Tutorials, manuals, etc.
  - MLNs, datasets, etc.
  - Publications

alchemy.cs.washington.edu