Constraint Propagation for Efficient Inference in Markov Logic

Tivadar Papai[1] Parag Singla[2] Henry Kautz[1]

[1]University of Rochester, Rochester NY 14627, USA

[2]University of Texas, Austin TX 78701, USA

September 13, 2011

Markov Logic [Richardson & Domingos (2006)]

- A probabilistic first-order logic (FOL)
- Knowledge Base (KB) is a set of weighted FOL formulas $W = \{(w_1, F_1) \ldots, (w_i, F_i), \ldots, (w_N, F_N)\}$
- The probability of a truth assignment x to the ground atoms:

$$\Pr(X = x) = (1/Z) \exp(\sum_{i=1}^{N} w_i n_i(x))$$

where w_i is the weight of F_i (the ith formula in the KB) and $n_i(x)$ is the number of true groundings of F_i under truth assignment x

Markov Logic

- *Evidence* corresponds to a truth assignment to a subset of ground atoms
- The goal typically is to find the marginal probabilities of the truth assignments to each individual ground atoms or find the most likely truth assignment given the evidence

Example

- $KB = \{(w, P(x) \vee Q(x))\}$
- Let x come from the domain: $\{a, b\}$
- There are 2 ground formulas: $P(a) \vee Q(a), P(b) \vee Q(b)$
- Evidence: $\neg P(a)$

Example

- $KB = \{(w, P(x) \vee Q(x))\}$, evidence: $\neg P(a)$
- The distribution defined by the KB:

P(a)	F	F	F	F	F	F	F	F
Q(a)	F	T	F	T	F	T	F	T
P(b)	F	F	T	T	F	F	T	T
Q(b)	F	F	F	F	T	T	T	T
Pr	$\frac{1}{Z}$	$\frac{e^w}{Z}$	$\frac{e^w}{Z}$	$\frac{e^{2w}}{Z}$	$\frac{e^w}{Z}$	$\frac{e^{2w}}{Z}$	$\frac{e^w}{Z}$	$\frac{e^{2w}}{Z}$

- where $Z = 3e^{2w} + 4e^{w} + 1$

Soft and Hard Constraints

- Markov logic theory consists of hard and soft constraints
- Soft constraints (F_S) are the F_i formulas with finite weights
- Hard constraints (F_H) have infinite weights
- A truth assignment to the atoms can only have non-zero probability if it does not violate any of the hard constraints
- F_H can be equivalently represented using a set of clauses without changing the probability distribution

Use Hard Constraints to Create New Evidence

- The more evidence we have the easier it is to perform marginal inference, or to find the truth assignment with the highest probability
- We can remove all the weighted formulas the truth value of which are already determined
- We only have to consider one truth value for every evidence atom

Example

- Assume we have $F_H = \{H(x) \vee O(x)\}$ and $F_S = \{H(x) \vee H_1(x, y_1) \vee H_2(x, y_2)\}$
- H, H_1, H_2 are hidden and O is observed
- If $O(c)$ is false then $H(c)$ has to be true
- When $H(c)$ is true we do not have to create a grounding for the soft clause

Notations

- Let L denote a predicate or its negation ($\neg\neg L = L$)
- Let $D(L)$ denote the set of tuples from which the argument of L can take its values
- Let $N(L) \subseteq D(L)$ be the set of those tuples s.t. $\forall t \in N(L) : L(t)$ has to be true in all models

Example

- $F_H = \{H(x) \vee O(x)\}$
- E.g. let $D(H) = D(O) = \{a, b, c\}$
- $\neg O(a), O(b)$ and $O(c)$ are given as evidence
- $N(O) = \{b, c\}$, $N(\neg O) = \{a\}$
- $N(H) = \{a\}$, $N(\neg H) = \{\}$

Goal

- For every L literal find the maximal $N(L)$ set
- For every L and $t \in N(L)$, add $L(t)$ to the evidence atoms
- Naïve approach would ground all the formulas in the KB - *expensive*
- Do the computations in a lifted way

Generalized Arc Consistency Algorithm

- When the domain is large solving the constraints globally is expensive
- We chose a hyper-arc / generalized arc consistency algorithm, where the hyper-arcs are based on the clauses, and the nodes are the L literals to which we want to determine $N(L)$
- Consider, e.g.:

$$C = L_1(x) \vee \ldots \vee L_k(x)$$

-

$$N(L_i) \leftarrow N(L_i) \bigcup \left[\bigcap_{i \neq j, 1 \leq j \leq k} N(\neg L_j) \right]$$

Join/Project

- S_i - set of tuples ($i = 1, 2$), X_i - corresponding variables

$$Join\{\langle X_i, S_i\rangle\} =$$
$$\langle X, \{c | c \in D(X) \wedge \forall i \exists s \in S_i \; \forall x \in X_i : s[x] = c[x]\}\rangle$$

corresponds to taking a cross product of the S_i tuples and selecting the terms with same values for the shared arguments

$$Project(Y, \langle S, X\rangle) =$$
$$\{c | c \in D(Y) \wedge \exists s \in S \; \forall y \in (Y \cap X) : s[y] = c[y]\}$$

corresponds to projecting a tuple onto a subset of its arguments

General Update Rule

- X_j denotes the variables in the arguments of L_j

$$N(L_i) \leftarrow$$

$$N(L_i) \bigcup [Project(X_i, Join_{j \neq i}\{\langle X_j, N(\neg L_j)\rangle\}]$$

- Example:

$$P(x, y) \vee Q(x, z) \vee R(y, z)$$

- $N(\neg P) = \{(a, b)\},\ N(\neg R) = \{(b, c), (c, d)\}$
- $Join\{\langle (x, y), \{(a, b)\}\rangle, \langle (y, z), \{(b, c), (c, d)\}\rangle\} = \langle (x, y, z), \{(a, b, c)\}\rangle$
- $N(Q) = Project((x, z), \langle (x, y, z), \{(a, b, c)\}\rangle) = \{(a, c)\}$

Lifted Unit Propagation

- Unit propagation: $O(a,b) \vee H(b,c), \neg O(a,b) \models H(b,c)$
- Lifted unit propagation:

$$
\begin{aligned}
&O(x,y) \vee H(y,z),\\
&\forall (x,y) \in N(\neg O) : \neg O(x,y) \models\\
&\forall y \in Project(y, \langle (x,y), N(\neg O) \rangle) : H(y,z)
\end{aligned}
$$

More on Savings

- We can project after some joins if a variable is guaranteed not to occur later in any join
- E.g.,

$$H(x) \vee O_1(x, y_1) \vee \ldots \vee O_n(x, y_n)$$

  We can project the tuples in $N(\neg O_i)$ to x before performing any joins, reducing the space complexity of the algorithm to $O(M^2)$ from $O(M^{n+1})$ if $|D(O_i)| = M^2$ and $|D(H)| = M$
- The update rule with this modification is sensitive to the order we perform the joins

Generalization

- Existential quantifier

$$P(x, z) \vee \exists y \, [Q(x, y) \wedge R(z, y)]$$

- Handling constants and equality can be done by adding auxiliary predicates

Algorithm

- Iterate through all the hard constraints
- In each hard constraint update $N(L_i)$ for every L_i literal
- Repeat until at least one of the $N(L_i)$ sets were updated
- Add the final $N(L_i)$ sets to the evidence
- The algorithm is sound and always terminates

Datasets

- CTF - capture the flag [Sadilek & Kautz (2010)] 17 formulas (15 hard and 2 soft)
- Cora - a standard benchmark problem for MLNs 52 formulas (6 hard and 46 soft)
- Library - synthetic dataset 4 formulas (3 hard and 1 soft)

Experiments

- Measured the running time of taking 1000 samples using MC-SAT (time includes creation of the ground network)
- As a space cost, we measured the maximum number of ground tuples needed at any point by the generalized arc consistency algorithm
- We do not report accuracy because the results are guaranteed to be the same at convergence

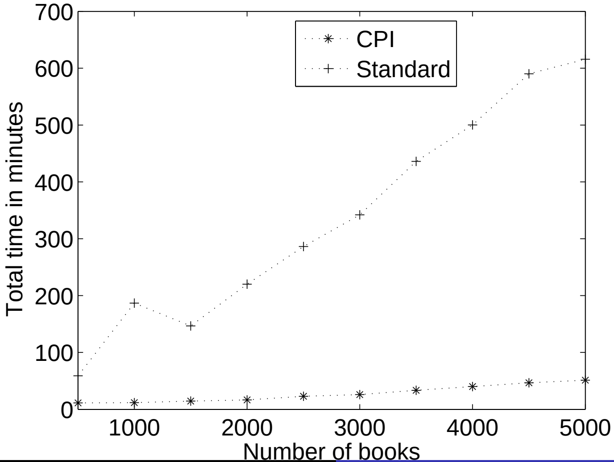
Time costs comparing the two inference approaches

Dataset	Time (in minutes)				
	Constraint Propagation		Probabilistic Inference		Net Reduction
	Standard	CPI	Standard	CPI	
CTF	0	0.37	1536.6	528.0	66%
Cora	0	0.07	181.1	26.2	85%
Library	0	0.20	286.4	23.0	92%

Memory costs comparing the two inference approaches

Dataset	No. of Ground Tuples (in 1000's)				
	Constraint Propagation		Probabilistic Inference		Net Reduction
	Standard	CPI	Standard	CPI	
CTF	0	585.5	2107.8	1308.7	41%
Cora	0	153.6	488.2	81.4	78%
Library	0	318.5	366.2	45.9	13%

Library


CPI
Standard
Total time in minutes
700
600
500
400
300
200
100
0
1000
2000
3000
4000
5000
Number of books

Next Steps

- Experimenting with other domains
- Trying out other forms of consistency requirements
- Symbolic manipulation of the theory
- Combining with lifted inference

Thank you for your attention!

Existential Quantifier

- $$P(x, z) \vee \exists y \, [Q(x, y) \wedge R(z, y)] \tag{1}$$
- For all the instantiations of x and y when $\exists y \, [Q(x, y) \wedge R(z, y)]$ is necessarily false $P(x, z)$ must be true
- All we need to do is to extend the definition of $N(L_i)$ to allow L_i to be the negation of an existentially quantified conjunction

Existential Quantifier

- Let $F = \neg\exists Y\,[L_1(X_1) \wedge \ldots \wedge L_k(X_k)]$ where $Y \subseteq \bigcup_i X_i$.
- Let $X = \bigcup_i X_i \setminus Y$ and let $D(X)$ be the full domain formed by the Cartesian product of the individual domains of the non-quantified variables in X in some ordering of the variables
- $$N(F) \leftarrow D(X) \setminus Project(X, Join_{1 \leq i \leq k}\{\langle X_i, D(L_i) \setminus N(\neg L_i)\rangle\}) \quad (2)$$