Closed-Form Learning of Markov Networks from Dependency Networks

Daniel Lowd, University of Oregon
<lowd@cs.uoregon.edu>

Introduction

Markov networks (MNs) have good semantics but are hard to learn and awkward to specify by hand.
Dependency networks (DNs) are easy to learn, but have annoying semantics based on Gibbs sampling.
Best of both worlds: Learn a DN and convert it into an equivalent MN before running inference.

We present the first ever method for converting a DN to an MN.
Our solution is:

- Flexible – Works with any kind of conditional distribution.
- Accurate – Exact for consistent DNs, very effective on general DNs.
- Efficient – Linear time.

MNs and DNs

Dependency network (DN):
Set of conditional probability distributions (CPDs)
e.g., $\{P_1(X_1 \mid X_3), P_2(X_2 \mid X_1, X_3), P_3(X_3 \mid X_1, X_2)\}$

Semantics: Probabilities are given by the stationary distribution of a Gibbs sampler, which may depend on variable order.

Learning: Train a probabilistic classifier (decision tree, logistic regression, etc.) to predict each variable.

A DN is consistent if its CPDs are consistent with some probability distribution. When learned from data, this is rarely the case.

Markov network (MN):
Normalized product of factors
e.g., $P(X_1, X_2, X_3) = \frac{1}{Z}\phi_1(X_1, X_2)\phi_2(X_2, X_3)\phi_3(X_1, X_3)$
Consistent semantics and many inference algorithms, but weight learning requires iterative optimization, even for pseudo-likelihood.

Defining a Distribution

If we can compute the relative probability of any two instances x and x', then we can define a probability distribution as follows:

$$f(x) = \frac{P(x)}{P(x')}$$

(x' is an arbitrarily chosen but fixed state.)

$$P(x) = \frac{1}{\sum_x f(x)} f(x) = \frac{1}{Z} f(x)$$

If we can represent f(x) as a product of factors, then we have a Markov network.

Since we're converting from a DN, we must express these factors in terms of the conditional distributions, $P(X_i \mid X_{-i})$.

Computing Relative Probabilities

Suppose x and x' only differ in the ith variable:

$$\frac{P(\mathbf{x})}{P(\mathbf{x}')} = \frac{P(x_j, \mathbf{x}_{-j})}{P(x'_j, \mathbf{x}_{-j})} = \frac{P(x_j \mid \mathbf{x}_{-j})P(\mathbf{x}_{-j})}{P(x'_j \mid \mathbf{x}_{-j})P(\mathbf{x}_{-j})} = \frac{P(x_j \mid \mathbf{x}_{-j})}{P(x'_j \mid \mathbf{x}_{-j})}$$

If x and x' differ in many variables, construct a sequence of intermediate states, $x^{(0)}$ through $x^{(n)}$, each differing in one variable:

$$\begin{aligned}
\frac{P(\mathbf{x})}{P(\mathbf{x}')} &= \frac{P(\mathbf{x}^{(0)})}{P(\mathbf{x}^{(n)})} \\
&= \frac{P(\mathbf{x}^{(0)})}{P(\mathbf{x}^{(n)})} \times \frac{P(\mathbf{x}^{(1)})}{P(\mathbf{x}^{(1)})} \times \cdots \times \frac{P(\mathbf{x}^{(n-1)})}{P(\mathbf{x}^{(n-1)})} \\
&= \frac{P(\mathbf{x}^{(0)})}{P(\mathbf{x}^{(1)})} \times \frac{P(\mathbf{x}^{(1)})}{P(\mathbf{x}^{(2)})} \times \cdots \times \frac{P(\mathbf{x}^{(n-1)})}{P(\mathbf{x}^{(n)})} \\
&= \prod_{i=1}^{n} \frac{P(\mathbf{x}^{(i-1)})}{P(\mathbf{x}^{(i)})} = \prod_{i=1}^{n} \frac{P(x_{o[i]} \mid \mathbf{x}^{(i)}_{-o[i]})}{P(x'_{o[i]} \mid \mathbf{x}^{(i)}_{-o[i]})}
\end{aligned}$$

Thus, the resulting MN has one factor for each variable X_i, which computes the conditional probability of that variable's value relative to its base value. All variables that come before X_i in the ordering are fixed to their base values in this computation.

Example

Suppose we have the following two CPDs:

$$\begin{aligned}
P_1(X_1 = T \mid X_2 = T) &= 4/5 & P_2(X_2 = T \mid X_1 = T) &= 2/3 \\
P_1(X_1 = F \mid X_2 = T) &= 1/5 & P_2(X_2 = F \mid X_1 = T) &= 1/3 \\
P_1(X_1 = T \mid X_2 = F) &= 2/5 & P_2(X_2 = T \mid X_1 = F) &= 1/4 \\
P_1(X_1 = F \mid X_2 = F) &= 3/5 & P_2(X_2 = F \mid X_1 = F) &= 3/4
\end{aligned}$$

Using base instance $x' = [T,T]$ and order $[1,2]$, we obtain two factors:

$$\phi_1(x_1, x_2) = \frac{P(x_1 \mid \mathbf{x}^{(1)}_{-1})}{P(x'_1 \mid \mathbf{x}^{(1)}_{-1})} = \frac{P(x_1 \mid x_2)}{P(X_1 = T \mid x_2)} =$$

X_1	X_2	$\phi_1(X_1, X_2)$
T	T	1
T	F	1
F	T	1/4
F	F	3/2

$$\phi_2(x_2) = \frac{P(x_2 \mid \mathbf{x}^{(2)}_{-2})}{P(x'_2 \mid \mathbf{x}^{(2)}_{-2})} = \frac{P(x_2 \mid X_1 = T)}{P(X_2 = T \mid X_1 = T)} =$$

X_2	$\phi_2(X_2)$
T	1
F	1/2

Inconsistent DNs

Problem: For inconsistent DNs, the conversion may depend on the variable ordering and the base instance x'.
Solution: Average over multiple orders and base instances.

- We can average over all base instances, weighted by the marginal distribution in the data. No increase in asymptotic running time!
- We can average over all rotations of an ordering. Increases running time by factor of *m*, where *m* is max number of parents.

See the paper for details!

Experiments

Methods

- We learned DNs with decision tree (DT) or logistic regression (LR) CPDs on 12 standard datasets.
- We converted each DN to an MN in two ways:
  - DN2MN with several different averaging strategies
  - Maximum pseudo-likelihood weight learning (Used by [Lowd & Davis, 2010] and [Ravikumar et al., 2009] to learn MNs.)
- We computed pseudo-log-likelihood (PLL) and conditional marginal log-likelihood (CMLL) on held out test data. Marginals for CMLL were computed via Gibbs sampling.

Results

1. **Averaging helps a lot!** Typically reduces the difference between MN and DN PLL by >90% with DT CPDs and >50% with LR CPDs.
2. With DT CPDs, DN2MN has **similar accuracy** to weight learning and is **60 times faster**.
3. With LR CPDs, DN2MN is **more accurate** and **360 times faster**.

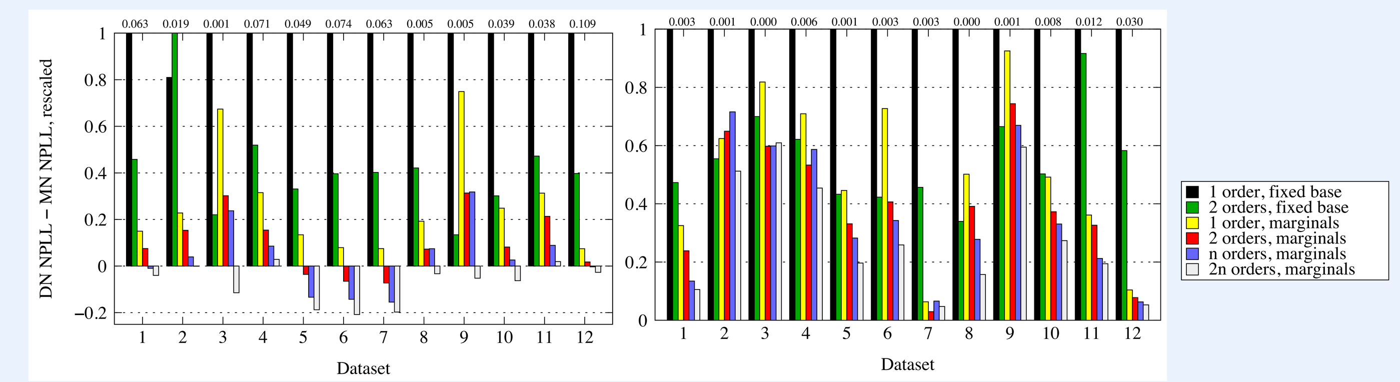

Table 4: Test set PLL and CMLL of converted DNs with tree and logistic regression CPDs.

	Tree CPDs				LR CPDs			
	PLL		CMLL		PLL		CMLL	
Dataset	LW	DN2MN	LW	DN2MN	LW	DN2MN	LW	DN2MN
NLTCS	-5.02	**-4.93**	-5.25	**-5.20**	-4.96	**-4.95**	-5.23	**-5.23**
MSNBC	-4.32	**-4.31**	**-5.75**	-5.80	-6.06	**-6.06**	**-6.28**	-6.28
KDDCup 2000	**-2.05**	-2.05	-2.08	**-2.07**	**-2.06**	-2.07	**-2.11**	-2.12
Plants	**-8.75**	-9.17	**-10.00**	-10.67	**-9.39**	-9.50	**-10.76**	-10.91
Audio	-38.01	**-37.77**	**-38.25**	-38.35	-36.17	**-36.11**	-36.93	**-36.88**
Jester	-51.42	**-50.77**	-51.49	**-51.42**	-49.01	**-48.81**	-49.83	**-49.76**
Netflix	-54.32	**-53.60**	-54.62	**-54.50**	-51.15	**-51.10**	-52.37	**-52.31**
MSWeb	**-8.20**	-8.33	**-8.72**	-8.77	-8.70	**-8.64**	-8.96	**-8.93**
Book	**-34.60**	-35.14	**-34.49**	-35.44	-33.86	**-33.41**	-34.75	**-34.09**
WebKB	-149.37	**-148.42**	**-149.99**	-151.88	-153.13	**-139.39**	-158.51	**-143.05**
Reuters-52	**-82.57**	-82.67	**-82.53**	-85.16	-81.44	**-77.62**	-81.82	**-79.60**
20 Newsgroups	-159.14	**-152.84**	-156.08	**-154.06**	-151.53	**-147.76**	-151.93	**-148.82**

	Vars
1. NLTCS	16
2. MSNBC	17
3. KDD Cup	64
4. Plants	69
5. Audio	100
6. Jester	100
7. Netflix	100
8. MSWeb	294
9. Book	500
10. WebKB	839
11. Reuters-52	889
12. 20 Newsgroups	910

Conclusion

- DN2MN performs exact conversion of consistent DNs, and very accurate conversion of inconsistent DNs learned from data.
- Combines with any DN learner to produce some of the fastest and most accurate methods for learning an MN.
- With decision trees, DN2MN is often more accurate than the DN.
- With logistic regression CPDs, DN2MN is often more accurate than weight learning.

Source code: **http://libra.cs.uoregon.edu**

References:
1. Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., & Kadie, C. (2000). Dependency networks for inference, collaborative filtering, and data visualization. *JMLR, 1,* 49-75.
2. Lowd, D., & Davis, J. (2010). Learning Markov network structure with decision trees. *ICDM 2010*. Sydney, Australia: IEEE Computer Society Press.
3. Ravikumar, P., Wainwright, M. J., & Lafferty, J. (2009). High-dimensional ising model selection using L1-regularized logistic regression. *Annals of Statistics.*