Mean Field Inference in Dependency Networks: An Empirical Study

Daniel Lowd and **Arash Shamaei**
Department of Computer and Information Science
University of Oregon
Eugene, OR 97403
{lowd,arash}@cs.uoregon.edu

Abstract

Dependency networks are a compelling alternative to Bayesian networks for learning joint probability distributions from data and using them to compute probabilities. A dependency network consists of a set of conditional probability distributions, each representing the probability of a single variable given its Markov blanket. Running Gibbs sampling with these conditional distributions produces a joint distribution that can be used to answer queries, but suffers from the traditional slowness of sampling-based inference. In this paper, we observe that the mean field update equation can be applied to dependency networks, even though the conditional probability distributions may be inconsistent with each other. In experiments with learning and inference on 12 datasets, we demonstrate that mean field inference in dependency networks offers similar accuracy to Gibbs sampling but with orders of magnitude improvements in speed. Compared to Bayesian networks learned on the same data, dependency networks offer higher accuracy at greater amounts of evidence. Furthermore, mean field inference is consistently more accurate in dependency networks than in Bayesian networks learned on the same data.

Introduction

In many domains, including recommender systems, fault diagnosis, and medicine, we wish to learn a joint probability distribution from data and use it to compute the probabilities of various events. The learning and inference tasks are often treated separately, but it is their combination that determines the overall accuracy of a system. A standard approach is to learn a probabilistic graphical model, such as a Bayesian or Markov network, and answer queries using an approximate inference algorithm, such as mean field inference or Gibbs sampling (Gilks, Richardson, and Spiegelhalter 1996). These methods have been largely successful, but have several limitations. For Bayesian networks, the need to avoid cycles significantly restricts the set of probability distributions that can be compactly represented. Undirected models such as Markov networks are somewhat more flexible, but suffer from computationally expensive weight learning and structure learning.

Dependency networks are an often overlooked alternative that combine some of the best qualities of Bayesian and Markov networks. Like Bayesian networks, dependency networks can be learned efficiently from data. Like Markov networks, they can represent many symmetric and cyclical dependencies more compactly than a Bayesian network. For these reasons, dependency networks have been successfully applied to tasks such as collaborative filtering (Heckerman et al. 2000) and part-of-speech tagging (Toutanova et al. 2003). A relational extension of dependency networks has also been successfully applied to domains such as fraud detection (Neville and Jensen 2007) and entity resolution (Natarajan et al. 2010).

However, dependency networks are traditionally limited to a single inference algorithm – Gibbs sampling – which may be slow to run and may produce different results depending on the order in which variables are sampled. This arises from the fact that the conditional probability distributions which comprise a dependency network are often inconsistent with each other, leading to a poorly-specified joint probability distribution. This disadvantage has led dependency networks to be largely ignored in favor of Bayesian and Markov networks, which have clearer semantics.

In this paper, we show that an approximate version of the mean field inference algorithm can be applied to dependency networks, in spite of the fact that the conditional distributions in a dependency network may be inconsistent. We find that mean field inference typically converges much faster than Gibbs sampling and offers similar accuracy.

We then compare the speed and accuracy of learning and inference in dependency networks to Bayesian networks, by learning and evaluating both models on 12 publicly available datasets. We find that learning times are similar, but dependency networks have consistently better test set pseudo-likelihood. In our queries, mean field runs faster in dependency networks than in Bayesian networks while producing consistently more accurate results. When we run Gibbs sampling in both models, their relative accuracy depends on the amount of evidence. On queries with more evidence, dependency networks are more accurate; with less evidence, Bayesian networks are more accurate. However, the differences in probabilities are typically very small. This means that dependency networks are a good choice for many applications, especially when queries will involve a large amount of evidence.

Background

Bayesian Networks

Let $\mathcal{X}$ be a set of random variables $\{X_1, X_2, \ldots, X_n\}$. A *Bayesian network (BN)* (Pearl 1988) represents a probability distribution over $\mathcal{X}$, $P(\mathcal{X})$, using a directed, acyclic graph and a set of conditional probability distributions (CPDs). The graph contains one node for each variable X_i. Each CPD $P(X_i|\Pi_i)$ specifies the probability of a single variable (X_i) given its parents in the graph (Π_i). The product of these CPDs gives the full joint probability distribution:

$$P(\mathcal{X} = \mathbf{x}) = \prod_i P(X_i = x_i|\Pi_i = \pi_i) \qquad (1)$$

Bayesian networks and other graphical models can be defined over both discrete and continuous variables. For this paper, we restrict our attention to discrete variables, although many of the methods can be naturally applied to the continuous and hybrid cases as well.

The *Markov blanket* of a variable X_i, denoted $\text{MB}(X_i)$, is the set of variables that render X_i independent from all other variables in the domain. In a Bayesian network, this set consists of X_i's parents in the graph, X_i's children, and all variables that have a child in common with X_i. These independencies, and others, are entailed by the CPD factorization from Equation 1.

The simplest form for a CPD is a full table that specifies the distribution of the child variable, X_i, given each configuration of its parent variables, Π_i. However, this representation is exponential in the number of parent variables, effectively limiting the maximum number of parents to a small number.

A more efficient and flexible alternative is to represent conditional probabilities with probabilistic decision trees (Chickering, Heckerman, and Meek 1997). A decision tree is a directed, rooted tree, in which each leaf contains a distribution over the target variable X_i. Each interior node is labeled with one of X_i's parents, $X_j \in \Pi_i$, and each outgoing arc from this node is labeled with one or more values of the parent variable, $x_j \in \text{Val}(X_j)$. The distribution at a leaf gives the conditional probability of X_i given the configurations of parent variables specified along the path from the leaf to the root of the tree.

BN Inference

To answer queries, we use the BN to compute marginal and conditional probabilities. Since exact inference in a BN is #P-complete (Roth 1996), approximation algorithms are typically necessary. Popular approximate inference algorithms include Markov chain Monte Carlo methods, such as Gibbs sampling; variational approximations such as mean field (MF); and loopy belief propagation (BP).

Gibbs sampling proceeds by initializing the query variables randomly and then resampling each in turn, according to its conditional probability given its Markov blanket. Evidence variables remain fixed to their set values. The probability of a particular query is computed as the fraction of the samples that match the query. For positive distributions (i.e., all configurations have non-zero probability), Gibbs sampling is guaranteed to eventually converge to the correct distribution. However, this can take a very long time, and convergence can be difficult to detect.

Sampling methods can often be improved by using Rao-Blackwellization, in which some of the variables are marginalized analytically. One way to apply Rao-Blackwellization to Gibbs sampling is to add *fractional* counts to the different states that could result from resampling the current variable, based on their relative probabilities. Then the variable is randomly assigned a single value, as in regular Gibbs sampling. These fractional counts can lead to a significant reduction in variance.

Belief propagation (BP) is an exact inference algorithm for tree-structured Bayesian and Markov networks. Loopy belief propagation is the application of the belief propagation algorithm to graphs with cycles (Murphy, Weiss, and Jordan 1999). We use the flooding message propagation scheme in our experiments, which sends messages from variables to CPDs and then CPDs to variables in each iteration, as described by Kschischang et al. (2001).

We defer a full description of mean field inference to the next section.

BN Learning

The objective of parameter learning is to select a set of model parameters (CPD probabilities) that maximize a particular objective function on the training data. For Bayesian networks, the most common objective function is log-likelihood, penalized by a prior distribution on each conditional distribution. We used the Beta(1,1) distribution as the prior in all of our experiments, which is equivalent to Laplace smoothing. In this case, parameter estimation can be done in closed form from the sufficient statistics of the training data.

In structure learning, the objective is to find the structure that maximize an objective function such as the Bayesian score (Heckerman, Geiger, and Chickering 1995). Finding the optimal structure is NP-hard (Chickering, Heckerman, and Meek 2004), but greedy search typically works well in practice. A common approach is to do hill climbing with random restarts, where the search operators include adding, deleting, and reversing arcs. For decision tree CPDs, the search operators consist of adding a split to a tree CPD, which replaces a decision tree leaf node with a new interior node that has leaves as its children (Chickering, Heckerman, and Meek 1997). Operations that would lead to a cycle in the BN are excluded from the search.

Dependency Networks

A dependency network (DN) (Heckerman et al. 2000) consists of a set of conditional probability distributions $P_i(X_i|\text{MB}(X_i))$, each defining the probability of a single variable given its Markov blanket. These conditional distributions may or may not be consistent with each other. The graph of a DN is directed, with a node for each variable in the domain. For each variable X_i, there are edges from the nodes representing variables in X_i's Markov blanket to the node representing X_i.

Unlike a BN, the product of all CPDs gives the pseudo-likelihood of an instance, P^*, not its probability:

$$P^*(\mathcal{X}) = \prod_i P_i(X_i|\text{MB}(X_i)) \tag{2}$$

Pseudo-likelihood (Besag 1975) is a consistent estimator commonly used for learning the parameters of Markov networks, since its local approach to normalization avoids the intractability of partition function. However, pseudo-likelihood learning tends to produce models that handle long-range dependencies poorly.

The joint distribution of a DN is defined as the stationary distribution of a Gibbs sampler run by using its CPDs for the required Markov blanket distributions, $P_i(X_i|\text{MB}(X_i))$. Heckerman et al. (2000) also describe a second method for computing probabilities in which the probability of each variable is computed in turn using the Gibbs sampler, conditioned on the states of previous variables. This reduces the variance when computing rare joint probabilities with sampling, and is theoretically equivalent. Toutanova et al. (2003) adapt the Viterbi algorithm to perform MAP inference in a chain-structured DN. To our knowledge, no other inference algorithms have ever been applied to DNs.

A dependency network can be constructed from any Bayesian or Markov network by constructing a conditional probability distribution for each variable given its Markov blanket. A DN can also be learned from data by the same methods used to learn a BN with tree-structured CPDs, except without enforcing acyclicity. This makes DN learning trivial to parallelize with a separate process for each variable.

However, when learned from data, the resulting CPDs may be inconsistent with each other. Heckerman et al. refer to this as a "general dependency network." Given a particular sampling order, a Gibbs sampler will still have a unique stationary distribution when the distribution is positive (all probabilities are non-zero). However, the stationary distribution may depend on the ordering chosen. Furthermore, the joint distribution determined by the Gibbs sampler may be inconsistent with the conditional probabilities asserted by the CPDs. When learned from data, DNs tend to roughly approximate consistency, since each CPD is approximating the same empirical distribution of the training data.

Mean Field Inference in Dependency Networks

Mean field inference (MF) approximates an intractable distribution P with a fully factorized distribution Q:

$$Q(\mathcal{X}) = \prod_i Q(X_i) \tag{3}$$

Q is selected to be as close as possible to the desired distribution P, where distance[1] is measured as the Kullback-Leibler (KL) divergence between Q and P:

$$\text{KL}(Q \parallel P) = \sum_X Q(X) \log \frac{Q(X)}{P(X)} \tag{4}$$

$$= E_{X\sim Q}\left[\log \frac{Q(X)}{P(X)}\right] \tag{5}$$

By deriving the fixed point equations for MF, it can be shown that every local optimum of the KL divergence satisfies the following equation for every marginal in the Q distribution:

$$Q(X_i) = \frac{1}{Z_i} \exp\left(E_{\mathcal{X}\setminus\{X_i\}\sim Q}[\log P(X_i|\mathcal{X}\setminus\{X_i\})]\right) \tag{6}$$

where Z_i is a normalizing constant. This can be interpreted as setting $Q(X_i)$ to match the expectation of X_i in log space, where the expectation is computed according to the distribution of the remaining variables in Q. By using this equation to iteratively update each $Q(X_i)$ marginal in turn, the KL divergence can be shown to decrease in each iteration until a local minimum is reached. A common ordering for the Q updates is to use a queue, so that after $Q(X_i)$ is updated, the neighbors of X_i are added to the end of the queue if not already in it. This continues until convergence.

These updates were derived by Haft et al. (1999). They additionally observed that, since these equations only depend on P through the conditional distributions $P(X_i|\mathcal{X}\setminus\{X_i\})$ (or more specifically, $P(X_i|\text{MB}(X_i))$, since the Markov blanket renders all other variables independent), they could be used to define MF update equations for any representation. However, Haft et al. did not apply these equations to dependency networks where the conditional distributions could be inconsistent.

Algorithm 1 contains pseudo-code for mean field inference in dependency networks. It is similar to Algorithm 11.7 from Koller and Friedman (2009), but with three modifications. The main change is line 8, which uses the conditional distributions $P_i(X_i|\text{MB}(X_i))$ to perform the update from Equation 6. This makes the algorithm applicable to dependency networks, whether or not their conditional distributions are consistent. The algorithm also features a maximum number of iterations, *MaxIters*, so that the algorithm will halt early if convergence is slow. In our experiments, we set this to 50 times the number of non-evidence variables. Mean field almost always converged in fewer iterations. When we increased the maximum number of iterations, it always converged. The second modification is to set a convergence threshold, ϵ, on the distance between the old and new marginals. In our experiments, we used a Euclidean distance of 0.0001 as the threshold. Algorithm 1 can be made equivalent to the standard mean field inference algorithm on a Bayesian or Markov network by setting ϵ to zero, *NumIters* to infinity, and each P_i to the Markov blanket distribution of X_i given its neighbors in the Bayesian or Markov network.

In a consistent DN, MF will always converge to a local minimum of the KL divergence, as in any other graphical model. With inconsistent CPDs and determinism, it is easy to produce oscillations. Given two binary variables A and

[1] We use the term "distance" informally. KL divergence is not symmetric, and therefore not a metric.

Algorithm 1 Mean field inference for dependency networks

$Q \leftarrow Q_0$
Iters $\leftarrow 0$
Unprocessed $\leftarrow \mathcal{X}$
while *Unprocessed* $\neq \emptyset$ and *Iters* $<$ *MaxIters* **do**
  Choose X_i from *Unprocessed*
  $Q_{old}(X_i) \leftarrow Q(X_i)$
  for $x_i \in Val(X_i)$ **do**
    $Q(x_i) \leftarrow \exp\{E_{\text{MB}(X_i)\sim Q}[\log P_i(X_i|\text{MB}(X_i)]\}$
  end for
  Normalize $Q(X_i)$ to sum to one
  if $\|Q_{old}(X_i) - Q(X_i)\| > \epsilon$ **then**
    Unprocessed $\leftarrow$ *Unprocessed* $\cup$ $\text{MB}(X_i)$
  end if
  Unprocessed $\leftarrow$ *Unprocessed* - X_i
  Iters $\leftarrow$ *Iters* $+1$
end while

B, let $P(A|B) = 1$ when $A = B$ and $P(B|A) = 1$ when $B = A$. MF run in this DN will oscillate forever unless initialized to the fixed point $Q(A) = Q(B) = 0.5$. With positive CPDs, where no probabilities are 1 or 0, it remains unknown if convergence is always guaranteed. MF converged in our experiments, but we do not know if this is always the case.

Methods

In our experiments, we sought to evaluate the relative effectiveness of different learning and inference schemes on real-world data. We were particularly interested in how MF compares to the standard approach of Gibbs sampling in DNs, both in speed and accuracy. Furthermore, since BNs are an alternative to DNs, we felt it was important to benchmark our results against standard BN learning and inference methods. If BNs can produce more accurate answers than DNs in less time, then the case for using DNs is relatively weak.

We ran our experiments on 12 publicly available datasets collected and prepared by Davis and Domingos (2010) and also used by Lowd and Davis (2010) for evaluating Markov network structure learning algorithms. We excluded the EachMovie dataset, since it is no longer publicly available. All variables are binary-valued. The datasets vary widely in size and number of variables. See Table 1 for a summary, and Davis and Domingos (2010) for more details on their origin. Datasets are listed in increasing order by number of variables.

We learned DNs and BNs with tree CPDs using the WinMine toolkit (Chickering 2002), which is based on the algorithms of Chickering et al. (1997) and Heckerman et al. (2000).[2] WinMine uses a structure prior that penalizes models based on the number of parameters:

$$P(S) \propto \kappa^s \tag{7}$$

[2]We also experimented with BNs with table CPDs, but found their performance to be similar to BNs with tree CPDs. For brevity, we only present the tree CPD results in this paper.

Table 1: Data Set Characteristics

Data Set	Train Set Size	Tune Set Size	Test Set Size	Num. Vars.
NLTCS	16,181	2,157	3,236	16
MSNBC	291,326	38,843	58,265	17
KDDCup 2000	180,092	19,907	34,955	64
Plants	17,412	2,321	3,482	69
Audio	15,000	2,000	3,000	100
Jester	9,000	1,000	4,116	100
Netflix	15,000	2,000	3,000	100
MSWeb	29,441	3,270	5,000	294
Book	8,700	1,159	1,739	500
WebKB	2,803	558	838	839
Reuters-52	6,532	1,028	1,540	889
20 Newsgroups	11,293	3,764	3,764	910

where s is the number of parameters in the model. To avoid overfitting, we tuned the κ parameter on the tune set. We evaluated on test data.

We ran inference using a Rao-Blackwellized Gibbs sampler, with 100 burn-in iterations and 1000 sampling iterations. We experimented with running the Gibbs sampler for ten times as long, but found that additional iterations produced little benefit on most of our datasets. We also ran mean field and belief propagation. All inference algorithms were implemented in the open-source Libra Toolkit.[3] We use DN.Gibbs and DN.MF to refer to Gibbs sampling and mean field in a learned dependency network, respectively; and BN.Gibbs, BN.MF, and BN.BP to refer to Gibbs sampling, mean field, and belief propagation in a learned Bayesian network, respectively.

All inference algorithms are evaluated using test set conditional marginal log-likelihood (CMLL), a common evaluation metric (Lee, Ganapathi, and Koller 2007; Davis and Domingos 2010). The CMLL represents the expected log loss of the query variable marginals, given the evidence variables:

$$CMLL(X = x) = \frac{1}{|\mathbf{X}_Q|} \sum_{X_i \in \mathbf{X}_Q} \log P(X_i = x_i | \mathbf{X}_E = \mathbf{x}_e) \tag{8}$$

where $\mathbf{X}_Q$ is the set of query variables, $\mathbf{X}_E$ is the set of evidence variables, and $\mathbf{x}_e$ denotes the configuration of the evidence variables in the example. Unlike Davis and Domingos (2010), we randomly select a separate set of query and evidence variables for each instance in the test set. The fraction of evidence variables ranges from 10% to 90% of the total variables in the domain, by 10% increments. All other variables are used as query variables. In order to reduce the variance among results with different amounts of evidence, the evidence variables in each query at 10% evidence are a subset of those at 20% evidence, which are a subset of those at 30% evidence, and so on. This makes our queries at different levels of evidence more similar, while remaining unbiased.

[3]The Libra Toolkit is available from http://libra.cs.uoregon.edu

Table 2: Learning time (in seconds), complexity (number of parameters), test set pseudo-log-likelihood (PLL), and test set log-likelihood (LL) of learned models. Reported error range is one standard deviation of the mean. Standard errors of less than 0.0005 are reported as 0.000. When differences in PLL or LL are statistically significant, the better result is in bold.

	Dependency Networks				Bayesian Networks			
Data set	Time	Cmplx.	PLL	LL	Time	Cmplx.	PLL	LL
NLTCS	1.3	875	**-0.311** $\pm$ 0.004	**-0.376** $\pm$ 0.005	**1.2**	130	-0.314 $\pm$ 0.004	-0.378 $\pm$ 0.005
MSNBC	**38.8**	6790	-0.254 $\pm$ 0.001	-0.355 $\pm$ 0.001	155.9	2016	-0.254 $\pm$ 0.001	**-0.354** $\pm$ 0.001
KDDCup 2000	27.1	4018	**-0.032** $\pm$ 0.000	**-0.034** $\pm$ 0.001	**12.3**	577	-0.033 $\pm$ 0.000	-0.034 $\pm$ 0.001
Plants	**8.3**	3482	**-0.131** $\pm$ 0.002	-0.192 $\pm$ 0.003	21.8	3167	-0.136 $\pm$ 0.002	**-0.185** $\pm$ 0.003
Audio	**14.7**	3641	**-0.387** $\pm$ 0.003	-0.405 $\pm$ 0.005	26.4	1802	-0.398 $\pm$ 0.003	-0.404 $\pm$ 0.005
Netflix	**19.0**	4003	**-0.549** $\pm$ 0.002	-0.573 $\pm$ 0.002	28.9	1433	-0.557 $\pm$ 0.002	**-0.572** $\pm$ 0.002
Jester	**12.7**	3033	**-0.523** $\pm$ 0.002	-0.539 $\pm$ 0.003	17.0	1081	-0.540 $\pm$ 0.003	-0.538 $\pm$ 0.003
MSWeb	31.5	6313	-0.029 $\pm$ 0.000	-0.033 $\pm$ 0.001	**24.3**	1608	**-0.029** $\pm$ 0.000	-0.033 $\pm$ 0.001
Book	20.6	4675	**-0.071** $\pm$ 0.002	**-0.072** $\pm$ 0.003	**11.8**	1877	-0.075 $\pm$ 0.003	-0.074 $\pm$ 0.004
WebKB	49.8	5193	**-0.180** $\pm$ 0.004	-0.191 $\pm$ 0.006	**41.1**	2774	-0.186 $\pm$ 0.004	-0.190 $\pm$ 0.006
Reuters-52	**122.0**	8247	**-0.092** $\pm$ 0.002	-0.106 $\pm$ 0.003	126.3	5156	-0.097 $\pm$ 0.002	**-0.102** $\pm$ 0.003
20 Newsgroups	**470.6**	14526	**-0.171** $\pm$ 0.002	-0.172 $\pm$ 0.003	495.2	6350	-0.177 $\pm$ 0.002	-0.172 $\pm$ 0.003

Results

Table 2 summarizes the BN and DN models learned on each dataset.

Learning DNs is faster than learning BNs in 7 out of 12 datasets. Therefore, DN learning time is at least comparable to BNs.

For pseudo-log-likelihood, DNs are significantly better on 10 out of 12 datasets according to a paired t-test ($p < 0.05$). MSWeb is the only dataset where BNs perform better. The better pseudo-likelihood of DNs in this experiment is the consequence of the fact that DNs effectively optimize pseudo-likelihood while learning.

We used Algorithm 1 from Heckerman et. al (Heckerman et al. 2000) to approximate the log-likelihood in dependency networks. Since this is a very slow sampling-based algorithm, we used only 40% of the test data. Log-likelihood for the BNs was computed exactly on the same subset. DNs are significantly better on 3 datasets and BNs are significantly better on 4, according to a paired t-test ($p < 0.05$). We expected BNs to do better, since they are directly optimizing log-likelihood, but DNs were surprisingly competitive.

We then ran multiple inference algorithms on all data sets to evaluate the accuracy of the different models and algorithms in answering conditional queries. Figure 1 shows the per-variable CMLL for each data set, averaged over all amounts of evidence (10% to 90%). With the exception of BN.MF, all algorithms have very similar accuracies. BN.BP, BN.Gibbs, and DN.Gibbs tend to be slightly more accurate than DN.MF. On every dataset, DN.MF is more accurate than BN.MF. Whether DN.Gibbs is more accurate than BN.Gibbs depends on the dataset. Overall, the accuracy of DN.MF is superior to BN.MF and comparable to other inference algorithms.

Figure 2 depicts the average inference time for each data set, averaged over all amounts of evidence. For inference time, the MF methods are the fastest, often 1-2 orders of magnitude faster than the other methods. In addition, DN.MF is faster than BN.MF in 10 data sets out of 12.

To tease apart the differences between DNs and BNs, we looked at wins and losses by amount of evidence. Table 3 compares DN.Gibbs to BN.Gibbs and DN.MF to BN.BP, respectively. We chose to compare DN.MF to BN.BP here because BN.MF is consistently less accurate, and BN.BP is also an iterative message-passing algorithm. Many of these wins and losses are by very small amounts, but the trends are still informative. At greater amounts of evidence, DNs are increasingly accurate relative to BNs. This can be explained by the fact that DNs are optimizing a conditional measure when learned, while BNs are optimizing the overall likelihood. This is also consistent with DNs having better PLL values than BNs (see Table 2), since PLL conditions on the most evidence possible: all other variables.

Table 3: Number and fraction of datasets on which DN.Gibbs and DN.MF are more accurate than BN.Gibbs and BN.BP, respectively.

Evidence	DN.G vs. BN.G	Percent	DN.MF vs. BN.BP	Percent
10%	3	25%	1	8%
20%	1	8%	1	8%
30%	1	8%	1	8%
40%	3	25%	3	25%
50%	5	42%	5	42%
60%	7	58%	8	67%
70%	10	83%	9	75%
80%	10	83%	10	83%
90%	11	92%	11	92%
Total	51	47%	39	36%

Since Gibbs sampling is an anytime algorithm, we can force it to match the speed of mean field. Table 4 shows the results of running Gibbs sampling in either DNs or BNs with the same amount of time required by DN.MF to converge. In most cases Gibbs sampling is less accurate, since DN.MF converges much faster than Gibbs. In total, DN.MF wins 85% of the time compared to DN.Gibbs and 79% of the time compared to BN.Gibbs.

Finally, we measured the difference between the marginal

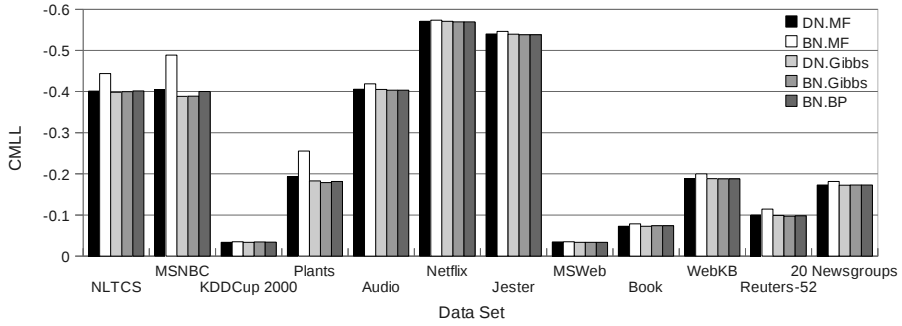

Figure 1: Per-variable CMLL for each dataset, averaged over all amounts of evidence. Shorter bars are better.

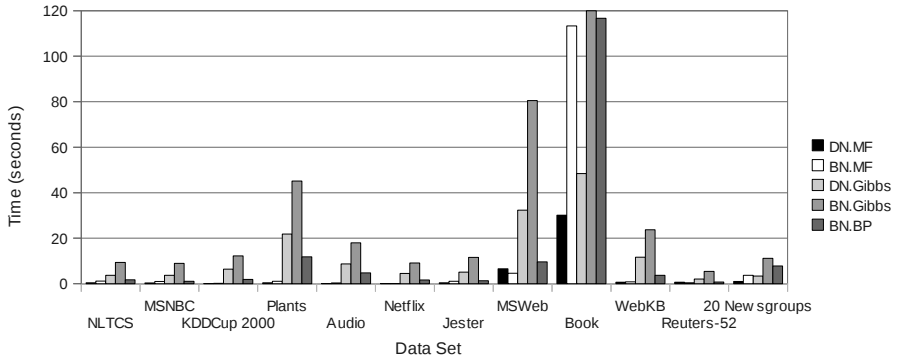

Figure 2: Average inference time for each dataset, averaged over all amounts of evidence.

Table 4: Number and fraction of datasets in which DN.MF is more accurate than DN.Gibbs and BN.Gibbs when run for the same amount of time.

	DN.Gibbs		BN.Gibbs	
Evidence	MF Wins	Percent	MF Wins	Percent
10%	9	75%	7	58%
20%	10	83%	6	50%
30%	10	83%	6	50%
40%	9	75%	10	83%
50%	10	83%	10	83%
60%	10	83%	11	92%
70%	11	92%	11	92%
80%	11	92%	12	100%
90%	12	100%	12	100%
Total	92	85%	85	79%

probabilities inferred by DN.MF and all other methods. We chose root mean squared difference as our metric, since it penalizes large differences more than small differences. For 10 out of 12 datasets, the root mean squared difference between DN.MF and the other methods was less than 0.005. Between DN.MF and BN.Gibbs, the root mean squared difference was less than 0.001 on half of the datasets. Between DN.MF and DN.Gibbs, the root mean squared difference was less than 0.0005 on 9 datasets. This supports the hypothesis that all methods produce similar results. In particular, MF and Gibbs sampling give very similar probabilities in dependency networks.

Source code and additional results are available in the online appendix at `http://ix.cs.uoregon.edu/~lowd/dnmf`.

Conclusion

Mean field inference in DNs offers similar accuracy to Gibbs sampling but in significantly less time. It is also consistently more accurate than running mean field inference in a BN learned on the same data. When we compare to other BN inference algorithms as well, we find that DNs are more accurate when more evidence is available. Therefore, although BNs remain a good choice for many applications of statistical learning and inference, DNs are a compelling alternative.

In future work, we intend to apply mean field to relational dependency networks (Neville and Jensen 2007). In relational domains such as social network analysis, networks are typically cyclical and large-scale, making dependency networks with mean field inference a natural choice over directed models or Gibbs sampling.

Acknowledgments

We thank Jesse Davis, Chloé Kiddon, and Aniruddh Nath for feedback on earlier drafts of this paper, and Christopher Meek for helpful discussions.

References

Besag, J. 1975. Statistical analysis of non-lattice data. *The Statistician* 24:179–195.

Chickering, D.; Heckerman, D.; and Meek, C. 1997. A Bayesian approach to learning Bayesian networks with local structure. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, 80–89. Providence, RI: Morgan Kaufmann.

Chickering, D.; Heckerman, D.; and Meek, C. 2004. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research* 5:1287–1330.

Chickering, D. M. 2002. The WinMine toolkit. Technical Report MSR-TR-2002-103, Microsoft, Redmond, WA.

Davis, J., and Domingos, P. 2010. Bottom-up learning of Markov network structure. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*. Haifa, Israel: ACM Press.

Gilks, W. R.; Richardson, S.; and Spiegelhalter, D. J., eds. 1996. *Markov Chain Monte Carlo in Practice*. London, UK: Chapman and Hall.

Haft, M.; Hofmann, R.; and Tresp, V. 1999. Model-independent mean field theory as a local method for approximate propagation of information. *Computation in Neural Systems* 10(1):93–105.

Heckerman, D.; Chickering, D. M.; Meek, C.; Rounthwaite, R.; and Kadie, C. 2000. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research* 1:49–75.

Heckerman, D.; Geiger, D.; and Chickering, D. M. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning* 20:197–243.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA: MIT Press.

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.

Lee, S.-I.; Ganapathi, V.; and Koller, D. 2007. Efficient structure learning of Markov networks using L1-regularization. In *Advances in Neural Information Processing Systems 19*, 817–824. MIT Press.

Lowd, D., and Davis, J. 2010. Learning Markov network structure with decision trees. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*. Sydney, Australia: IEEE Computer Society Press.

Murphy, K.; Weiss, Y.; and Jordan, M. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. Stockholm, Sweden: Morgan Kaufmann.

Natarajan, S.; Khot, T.; Kersting, K.; Gutmann, B.; and Shavlik, J. 2010. Boosting relational dependency networks. In *Proceedings of the Twentieth International Conference on Inductive Logic Programming*. Firenze, Italy: Springer.

Neville, J., and Jensen, D. 2007. Relational dependency networks. *Journal of Machine Learning Research* 8.

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

Roth, D. 1996. On the hardness of approximate reasoning. *Artificial Intelligence* 82:273–302.

Toutanova, K.; Klein, D.; Manning, C.; and Singer, Y. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics*, 173–180. Association for Computational Linguistics.