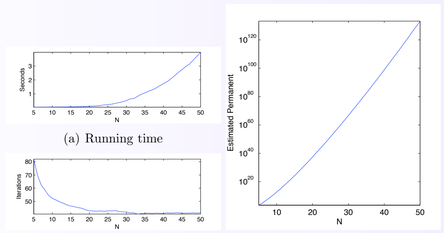
Approximating the Permanent with Belief Propagation

New York Academy of Sciences Machine Learning Symposium 2007
Bert Huang, Tony Jebara
Columbia University



(a) Running time

(b) Iterations

(c) Approximate Permanent

Figure 1: (a) Average running time until convergence of BP for $5 \leq n \leq 50$. (b) Number of iterations. (c) Approximate permanent output by Bethe method (y-axis is log scale).

1 Overview

- This work describes a method of approximating non-negative matrix permanents efficiently using belief propagation (BP).
- While the determinant of an $n \times n$ matrix can be evaluated exactly in sub-cubic time, computing the permanent of an $n \times n$ matrix is #P-complete
- Significant progress has produced an FPRAS that can handle arbitrary $n \times n$ matrices with non-negative entries [5]. The method uses Markov chain Monte Carlo and only requires a large polynomial order of samples.
- We formulate a probability distribution whose partition function is exactly the permanent, then use Bethe free energy to approximate this partition function.

2 The Permanent as a Partition Function

- Given an $n \times n$ non-negative matrix W, the matrix permanent is

$$\sum_{\pi \in S_n} \prod_{i=1}^{n} W_{i\pi(i)}. \tag{1}$$

- S_n is the symmetric group on the set $\{1, \ldots, n\}$: the set of all permutations of the columns of W.
- W defines some function $f(\pi; W)$ over S_n. The permanent can also be written

$$\text{per}(W) = \sum_{\pi \in S_n} f(\pi; W), \qquad \text{where} \qquad f(\pi; W) = \prod_{i=1}^{n} W_{i\pi(i)}.$$

The output of f is non-negative, so we consider f a density function over the space of all permutations.

- We can factor this density function over permutations. Max-product loopy BP on the resulting factor graph has guaranteed convergence [1], but we will use sum-product.
- Consider permutation assignment variables $X = \{x_1, \ldots, x_n\}$, and $Y = \{y_1, \ldots, y_n\}$, such that $x_i, y_j \in \{1, \ldots, n\}, \forall i, j$.

$$\phi(x_i) = \sqrt{W_{ix_i}}, \quad \phi(y_j) = \sqrt{W_{y_j j}}, \quad \psi(x_i, y_j) = I(\neg(j = x_i \oplus i = y_j)).$$

$I()$ is an indicator function such that $I(\text{true}) = 1$ and $I(\text{false}) = 0$. Then the ψ function outputs zero whenever any pair (x_i, y_j) have settings that cannot come from a true permutation.

- Given these definitions, we can define the equivalent density function to $f(\pi)$:

$$\hat{f}(X, Y) = \prod_{i,j} \psi(x_i, y_j) \prod_{k} \phi(x_k)\phi(y_k), \qquad \text{per}(W) = \sum_{X,Y} \hat{f}(X, Y) \tag{2}$$

- Finally, we treat density function $\hat{f}$ as a probability, adding a normalization constant to it:

$$p(X, Y) = \frac{1}{Z(W)} \prod_{i,j} \psi(x_i, y_j) \prod_{k} \phi(x_k)\phi(y_k). \tag{3}$$

- The normalizer or partition function $Z(W)$ is the sum of $f(X, Y)$ for all possible inputs X, Y. Therefore $Z(W) = \text{per}(W)$

3 Bethe Free Energy

- The minimum of **Gibbs Free Energy**, which is a function over proposal distributions b, is the negative log-partition function $-\ln Z$.
- Minimizing the Gibbs free energy is intractable because it requires computations over the entire state space to compute necessary entropy and average energy terms.
- Instead, we approximate by summing the energies and entropies of marginals or pseudo-marginals of subsets of the variables and subtracting out overcounted regions. This is called the Bethe approximation or **Bethe free energy**.
- Given a belief state b:

$$\begin{aligned} F_{Bethe} &= -\sum_{ij}\sum_{x_i,y_j} b(x_i, y_j) \ln \psi(x_i, y_j)\phi(x_i)\phi(y_j) + \sum_{ij}\sum_{x_i,y_j} b(x_i, y_j) \ln b(x_i, y_j) \\ &\quad -(n-1)\sum_{i}\sum_{x_i} b(x_i) \ln b(x_i) - (n-1)\sum_{j}\sum_{y_j} b(y_j) \ln b(y_j) \end{aligned} \tag{4}$$

- Belief state b is a set of pseudo-marginals that are only locally consistent, but do not have to be true marginals of a single global distribution. The marginals obey

$$\sum_{y_j} b(x_i, y_j) = b(x_i), \quad \sum_{x_i} b(x_i, y_j) = b(y_j), \ \forall i, j, \quad \sum_{x_i, y_j} b(x_i, y_j) = 1.$$

- Since belief propagation has been shown to minimize Bethe free energy [3, 7, 8], we will use the approximation

$$\text{per}(W) \approx \exp\left(-\min_{b} F_{\text{Bethe}}(b)\right) \tag{5}$$

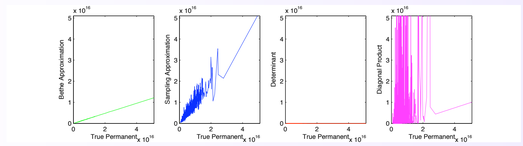


Figure 2: Approximate permanent for $N = 8$ vs. true permanent using four methods. On these small matrices, sampling achieved better absolute accuracy, but Bethe is more monotonic. See Table 1 for Kendall distances.

4 Belief Propagation

- We use the dampened belief propagation described in [3], which the author derives as a local minimization of Bethe free energy.
- Belief Propagation is a message passing algorithm that iteratively updates messages between variables that define the local beliefs.
- Let $m_{x_i}(y_j)$ be the message from x_i to y_j. The beliefs are defined by the messages:

$$b(x_i, y_j) = \frac{1}{Z}\psi(x_i, y_j)\phi(x_i)\phi(y_j)\prod_{k \neq j} m_{y_k}(x_i)\prod_{\ell \neq i} m_{x_\ell}(y_j)$$

$$b(x_i) = \frac{1}{Z}\phi(x_i)\prod_{k} m_{y_k}(x_i), \ \ b(y_j) = \frac{1}{Z}\phi(y_j)\prod_{k} m_{x_k}(y_j) \tag{6}$$

- In each iteration, the messages are updated according to the following formula:

$$m^{\text{new}}_{x_i}(y_j) = \sum_{x_i}\left[\phi(x_i)\psi(x_i, y_j)\prod_{k \neq j} m_{y_k}(x_i)\right] \tag{7}$$

- We dampen the messages to encourage a smoother optimization in log-space.

$$\ln m_{x_i}(y_j) = \ln m_{x_i}(y_j) + \epsilon\left[\ln m^{\text{new}}_{x_i}(y_j) - \ln m_{x_i}(y_j)\right] \tag{8}$$

ϵ is a dampening rate as in [3] and dampen in log space because the messages of BP are exponentiated Lagrange multipliers of Bethe optimization.

4.1 Algorithmic Speedups

- Quickly updating messages is challenging since each variable sends a message vector of length-n to n neighbors, so there are $2n^3$ values to update each iteration. We need to avoid redundant computation.
- In Equation (7), the only factor affected by the value of y_j is the ψ function.
- Therefore, we can explicitly define the update rules based on the ψ function, taking advantage of the fact that the computation for each setting of y_j is similar. When $y_j = i$, we have:

$$m^{\text{new}}_{x_i}(y_j = i) = \frac{1}{Z}\left(\left(\sum_{x_i}\phi(x_i)\prod_{k \neq j} m_{y_k}(x_i)\right) - \sum_{x_i \neq j}\phi(x_i)\prod_{k \neq j} m_{y_k}(x_i)\right).$$

When $y_j \neq i$,

$$m^{\text{new}}_{x_i}(y_j \neq i) = \frac{1}{Z}\left(\left(\sum_{x_i}\phi(x_i)\prod_{k \neq j} m_{y_k}(x_i)\right) - \phi(x_i = j)\prod_{k \neq j} m_{y_k}(x_i = j)\right).$$

- In each of these formulas, the expensive computation is the sum over all settings of x_i of the product of all but one incoming messages. Computing this as it is written takes $\mathcal{O}(n^2)$ operations just to perform one of the $2n^3$ updates. Instead, we compute parts of the formulas in advance:

$$g(x_i) = \phi(x_i)\prod_{k} m_{y_k}(x_i), \qquad h_{y_j \to x_i} = \sum_{x_i}\frac{g(x_i)}{m_{y_j}(x_i)}. \tag{9}$$

- After computing these values and storing them, we can substitute these reusable values into the full message update formula, without having to recompute them for each entry.

$$m^{\text{new}}_{x_i}(y_j = i) = \frac{1}{Z}\left(\left(\sum_{x_i}\frac{g(x_i)}{m_{y_j}(x_i)}\right) - \sum_{x_i \neq j}\frac{g(x_i)}{m_{y_j}(x_i)}\right) = \frac{1}{Z}\left(\frac{g(x_i = j)}{m_{y_j}(x_i = j)}\right) \tag{10}$$

$$m_{x_i}(y_j \neq i) = \frac{1}{Z}\left(\left(\sum_{x_i}\frac{g(x_i)}{m_{y_j}(x_i)}\right) - \frac{g(x_i = j)}{m_{y_j}(x_i = j)}\right) = \frac{1}{Z}\left(h_{y_j \to x_i} - \frac{g(x_i = j)}{m_{y_j}(x_i = j)}\right)$$

- We can now update all message vectors in $\mathcal{O}(n^3)$ time per iteration.
- Finally, we compute the beliefs

$$b(x_i) = \frac{1}{Z}g(x_i), \quad b(y_j) = \frac{1}{Z}g(y_j), \quad b(x_i, y_j) = \frac{1}{Z}\psi(x_i, y_j)\frac{g(x_i)g(y_j)}{m_{y_j}(x_i)m_{x_i}(y_j)}$$

- (As a caveat, Equation (9) requires that $m_{y_j}(x_i)$ is non-zero. We can fix this by adding a tiny constant.)

5 Experiments

5.1 Convergence and Running Time

- We ran belief propagation to approximate the permanents of random matrices of sizes $n = [5, 50]$, recording the total running time and the number of iterations to convergence.
- These experiments were run on a 2.8 Ghz Intel Xeon processor. The code is in C and compiled using `gcc` version 3.4.6.

n	Bethe Approximation	Random Sampling	Determinant	Diagonal
10	**0.00023**	0.0248	0.3340	0.0724
8	**0.0028**	0.1285	0.4995	0.4057
5	**0.0115**	0.0914	0.4941	0.3834

Table 1: Normalized Kendall distances between the rankings of random matrices via their true permanents and the rankings via approximation. See Fig. 2 for visualization.

5.2 Accuracy of Approximation

- We evaluate the accuracy of our algorithm by computing 1000 random matrices of sizes 5, 8 and 200 matrices of size 10.
- We computed the true permanents of these matrices, then computed approximate permanents using
  - our Bethe approximation
  - sampled random permutations (summing their products and scaling)
  - the determinant
  - the scaled product of the diagonal entries.
- To be able to compare to the true permanent, we had to limit this analysis to small matrices. However, since MCMC sampling methods such as in [5] take $\mathcal{O}(n^{10})$ time to reach less than some ϵ error, we expect that as matrix size increases, the achievable precision decreases.
- In our results, determinants and the products of diagonals are neither accurate nor consistent approximations of the permanent.
- Sampling is accurate w. r. t. absolute distance to the permanent.
- Bethe approximation seems the most consistent. While the approximations of the permanent are off by a large amount, they seem to be consistently off by some monotonic function of the true permanent.
- In many cases, this virtue is more important than the absolute accuracy, since most applications requiring a matrix permanent likely compare the permanents of various matrices.
- To measure the monotonicity and consistency of these approximations, we consider the Kendall distance [2] between the ranking of the random matrices according to the true permanent and their rankings according to the approximations.
- The Kendall distance between two permutations π_1 and π_2 is

$$D_{\text{Kendall}}(\pi_1, \pi_2) = \frac{2}{n(n-1)}\sum_{i=1}^{n}\sum_{j=i+1}^{n} I\left((\pi_1(i) < \pi_1(j)) \wedge (\pi_2(i) > \pi_2(j))\right).$$

(the total number of pairs where π_1 and π_2 disagree on the ordering.)

- Table 1 lists the Kendall distances between the true permanent ranking and the four approximations. The Bethe ranking is always closer to the true ranking than sampling.

6 Discussion

- We have described an algorithm based on BP over a specific distribution that allows an efficient approximation of the $\#P$ matrix permanent operation.
- We plan to explore higher order approximations Kikuchi free energies. Unfortunately the structure of our graphical model causes higher order interactions to become expensive quickly, since each variable has exactly N neighbors.
- We attempted convexity analysis of the Bethe free energy of our distribution, but found that our formulation did not meet the sufficient conditions provided in [4, 6].
- However, our empirical evidence implies that BP always converges, we suspect that we have not yet correctly analyzed the true space traversed during optimization.
- In particular, the distribution described by Equation 3 is defined over the set of all n^n possible X, Y states, while it is only nonzero in $n!$ states.
- Any beliefs derived from belief propagation obey similar constraints, so it is reasonable to suspect that careful analysis of the optimization with special attention to the oddities of the distribution could yield more promising theoretical guarantees.

References

[1] M. Bayati, D. Shah, and M. Sharma. Maximum weight matching via max-product belief propagation. In *Proc. of the IEEE International Symposium on Information Theory*, 2005.

[2] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists, 2003.

[3] T. Heskes. Stable fixed points of loopy belief propagation are local minima of the bethe free energy. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 343–350. MIT Press, Cambridge, MA, 2003.

[4] T. Heskes. Convexity arguments for efficient minimization of the bethe and kikuchi free energies. *Journal of Artificial Intelligence Research*, 26, 2006.

[5] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM*, 51(4):671–697, 2004.

[6] S. Tatikonda and M. Jordan. Loopy belief propagation and Gibbs measures. In *Proc. Uncertainty in Artificial Intell., vol. 18*, 2002.

[7] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7), 2005.

[8] A. L. Yuille. Cccp algorithms to minimize the bethe and kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14(7):1691–1722, 2002.