# Sparse-Matrix Belief Propagation

Reid Bixler and Bert Huang
Dept. of Computer Science
Virginia Tech
http://learning.cs.vt.edu

DISCOVERY ANALYTICS CENTER

VIRGINIA TECH™

## Overview

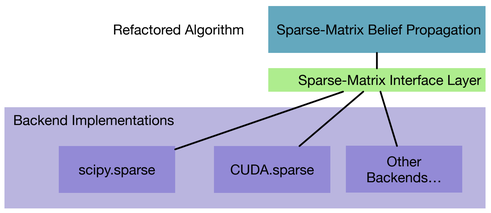
Loopy belief propagation (BP) and its variants are important approximate inference algorithms.

But they are often frustrating to implement, with intricate indexing to manage graph structure.

Software for BP is often difficult to maintain and difficult to integrate with other machine learning tools or computing backends.

We introduce an abstraction that refactors BP and variants to use sparse-matrix operations while preserving **exact algorithm behavior**.

This abstraction enables **seamless portability** to any backend that accelerates sparse and dense matrix operations.




## Belief Propagation for Pairwise Markov Random Fields

Probability: $\Pr(X = \mathbf{x}) = \frac{1}{Z} \exp\left(\sum_{i \in \mathcal{V}} \phi_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \phi(x_i, x_j)\right)$

Message update: $m_{i \to j}[x_j] = \log\left(\sum_{x_i} \exp\left(\phi_{ij}(x_i, x_j) + \phi_i(x_i) + \sum_{k \in N_i \setminus j} m_{k \to i}[x_i] - d_{ij}\right)\right)$

Belief formula: $\Pr(x_j) \approx \exp(b_j[x_j]) = \exp\left(\phi_j(x_j) + \sum_{i \in N_j} m_{i \to j}[x_j] - z_j\right)$

Log beliefs: $b_j[x_j] = \phi_j(x_j) + \sum_{i \in N_j} m_{i \to j}[x_j] - z_j$

## Simplification

Using the equivalence

$$\phi_i(x_i) + \sum_{k \in N_i \setminus j} m_{k \to i}[x_i] - d_{ij} = \phi_i(x_i) + \sum_{k \in N_i} m_{j \to i}[x_i] - m_{j \to i}[x_i] - d_{ij} = b_i[x_i] - m_{j \to i}[x_i]$$

Belief and message updates simplify to

$$b_j[x_j] = \phi_j(x_j) + \sum_{i \in N_j} m_{i \to j}[x_j] - z_j,$$

$$m_{i \to j}[x_j] = \log\left(\sum_{x_i} \exp\left(\phi_{ij}(x_i, x_j) + b_i[x_i] - m_{j \to i}[x_i]\right)\right)$$

## Sparse-Matrix Belief Propagation

While not converged:

$$\tilde{\mathbf{B}} \leftarrow \mathbf{\Phi} + \mathbf{MT}$$
$$\mathbf{B} \leftarrow \tilde{\mathbf{B}} - \mathbf{1}\,\text{logsumexp}\left(\tilde{\mathbf{B}}\right)$$
$$\mathbf{M} \leftarrow \text{logsumexp}(\mathbf{\Gamma} + \mathbf{B}\mathbf{F}^\top - \mathbf{MR})$$

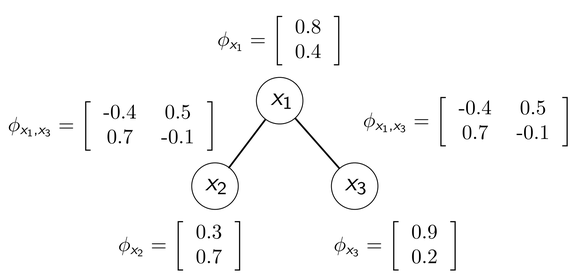
**Legend**

| | |
|---|---|
| $\tilde{\mathbf{B}}$ | c by n matrix of unnormalized unary log beliefs |
| $\mathbf{\Phi}$ | c by n matrix of unary log potentials |
| $\mathbf{M}$ | c by \|E\| matrix of log messages |
| $\mathbf{\Gamma}$ | c by c by \|E\| tensor of log pairwise potentials |
| $\mathbf{T}$ | \|E\| by n **sparse** matrix of binary indicators of which variable each message is sent **to** |
| $\mathbf{F}$ | \|E\| by n **sparse** matrix of binary indicators of which variable each message is sent **from** |
| $\mathbf{R}$ | \|E\| by \|E\| **sparse** permutation matrix indicating the reverse-direction messages |

$\text{logsumexp}(\mathbf{A}) = \log(\exp(\mathbf{1} \cdot \mathbf{A}))$ (with numerical stability trick)

## Example Computation

Simple MRF:




Unnormalized belief update: $\tilde{\mathbf{B}} \leftarrow \mathbf{\Phi} + \mathbf{MT}$

$$\underbrace{\begin{bmatrix} ? & ? & ? \\ ? & ? & ? \end{bmatrix}}_{\tilde{\mathbf{B}}} \leftarrow \underbrace{\begin{matrix} \phi_{x_1} & \phi_{x_2} & \phi_{x_3} \\ \end{matrix} \begin{bmatrix} 0.8 & 0.3 & 0.9 \\ 0.4 & 0.7 & 0.2 \end{bmatrix}}_{\mathbf{\Phi}} + \underbrace{\begin{bmatrix} 0.2 & 0.6 \\ -0.4 & 0.7 \\ 0.9 & -0.3 \\ 0.5 & 0.1 \end{bmatrix}^\top}_{\mathbf{M}^\top} \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{\mathbf{T}}$$

$$\begin{bmatrix} 1.6 & 0.2 & -0.4 \\ -0.2 & 0.6 & 0.7 \end{bmatrix} \leftarrow \underbrace{\begin{matrix} m_{1\to2} & m_{1\to3} & m_{2\to1} & m_{3\to1} \end{matrix} \begin{bmatrix} 0.2 & -0.4 & 0.7 & 0.9 \\ 0.6 & 0.7 & -0.3 & 0.1 \end{bmatrix}}_{\mathbf{M}} \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{\mathbf{T}}$$

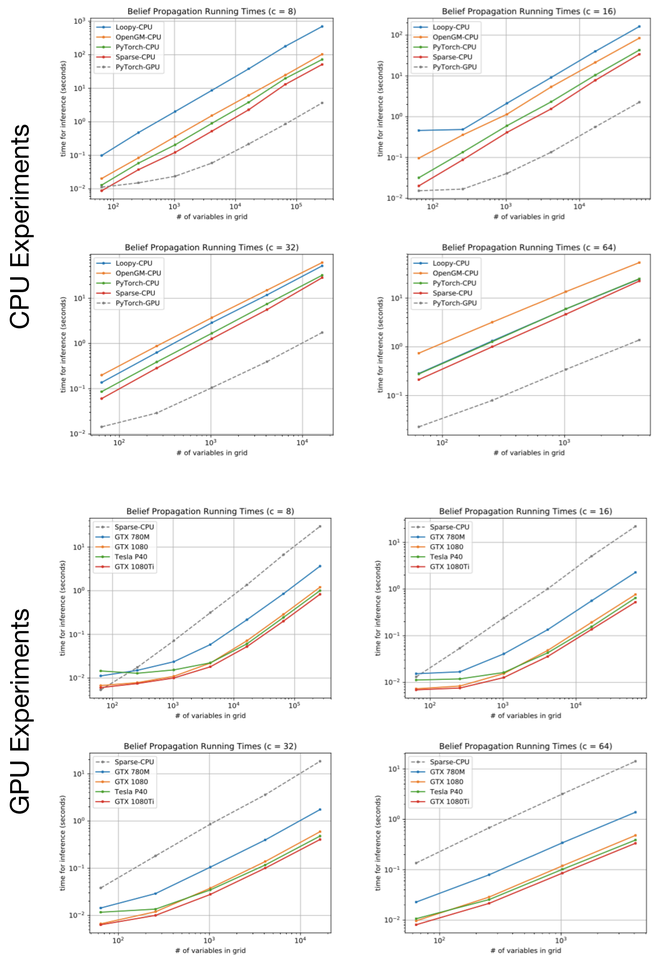
$\tilde{b}(x_1 = 1)$ $0.2 \times 0 + (-0.4) \times 0 + 0.7 \times 1 + 0.9 \times 1 = 1.6$

$\tilde{b}(x_2 = 2)$ $0.6 \times 1 + 0.7 \times 0 + (-0.3) \times 0 + 0.1 \times 0 = 0.6$

$$\underbrace{\begin{bmatrix} 2.4 & 0.5 & 0.5 \\ 0.2 & 1.3 & 0.9 \end{bmatrix}}_{\tilde{\mathbf{B}}} \leftarrow \underbrace{\begin{bmatrix} 0.8 & 0.3 & 0.9 \\ 0.4 & 0.7 & 0.2 \end{bmatrix}}_{\mathbf{\Phi}} + \underbrace{\begin{bmatrix} 1.6 & 0.2 & -0.4 \\ -0.2 & 0.6 & 0.7 \end{bmatrix}}_{\mathbf{MT}}$$

## Experiments

Timing BP inference on grid MRFs of different sizes and variable cardinalities




## Take Away Points

We recasted BP with matrix operations to replace tricky indexing, exactly preserving parallel message-passing.

We demonstrated that our abstraction enables seamless porting to alternate backends.

Code available in Python package. Include sparse-matrix implementations of BP, convexified BP, TRBP, max-product, MPLP. (Only BP is tested on GPU port so far.)

Future work: try on different backends, e.g., networked clusters (via Apache Spark), FPGAs, TPUs, etc.