CSC412 Tutorial 4

Image Denoising with Belief Propagation

Chen-Hao (Lance) Chao

Probabilistic Model

- An undirected tree $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- The set of nodes $\mathcal{V}$
- The set of edges $\mathcal{E}$
- The set of neighbors of a node $i \in \mathcal{V}$: $N(i) = \{j : (i, j) \in \mathcal{E}\}$

$$p(x_1, \ldots, x_n) \propto \prod_{i \in \mathcal{V}} \psi_i(x_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j)$$

Message Passing

- When x_i is discrete with K possible values, $m_{j \to i}$ is a vector with K values

- If x_j is unobserved

$$m_{j \to i}(x_i) = \sum_{x_j} \psi_j(x_j)\, \psi_{ij}(x_i, x_j) \prod_{k \in N(j) \setminus \{i\}} m_{k \to j}(x_j)$$

- If x_j is observed with value $\bar{x}_j$

$$m_{j \to i}(x_i) = \psi_j(\bar{x}_j)\, \psi_{ij}(x_i, \bar{x}_j) \prod_{k \in N(j) \setminus \{i\}} m_{k \to j}(\bar{x}_j)$$

BP on Trees

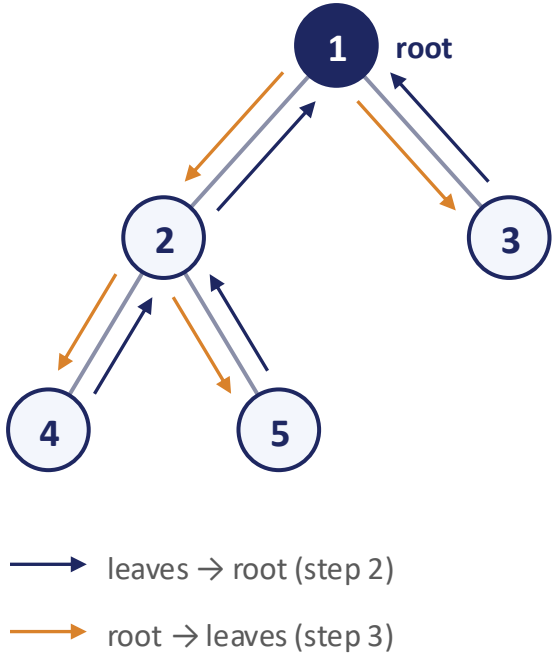
1. Choose an arbitrary root
2. Pass messages from leaves to root
3. Pass messages from root to leaves
4. For every node x_i, we have

$$b(x_i) \propto \psi_i(x_i) \prod_{j \in N(i)} m_{j \to i}(x_i)$$

BP on Trees

1. Choose an arbitrary root
2. Pass messages from leaves to root
3. Pass messages from root to leaves
4. For every node x_i, we have

$$b(x_i) \propto \psi_i(x_i) \prod_{j \in N(i)} m_{j \to i}(x_i)$$



Loopy BP

1. Initialize all messages with $m_{j\to i}(x_i) = 1/K$
2. For some number of iterations, keep going through each edge and do BP updates

$$m_{j\to i}(x_i) = \sum_{x_j} \psi_j(x_j)\, \psi_{ij}(x_i, x_j) \prod_{k\in N(j)\setminus\{i\}} m_{k\to j}(x_j)$$

3. It will generally not converge, but that's ok.
4. Compute beliefs

$$b(x_i) \propto \psi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i)$$

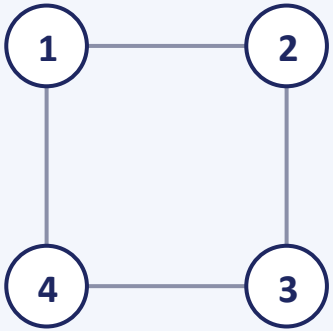
Loopy BP

1. Initialize all messages with $m_{j\to i}(x_i) = 1/K$
2. For some number of iterations, keep going through each edge and do BP updates

$$m_{j\to i}(x_i) = \sum_{x_j} \psi_j(x_j)\,\psi_{ij}(x_i, x_j) \prod_{k\in N(j)\setminus\{i\}} m_{k\to j}(x_j)$$

3. It will generally not converge, but that's ok.
4. Compute beliefs

$$b(x_i) \propto \psi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i)$$



Example: a graph with a loop

Image Denoising

- A binary image is a $\sqrt{n} \times \sqrt{n}$ matrix where each entry is $+1$ or -1.
- We vectorize this matrix and denote the image as $x \in \mathbb{R}^n$.
- For example, the Mona Lisa on the right is a 128×128 image, vectorized to be $x \in \mathbb{R}^{16384}$.

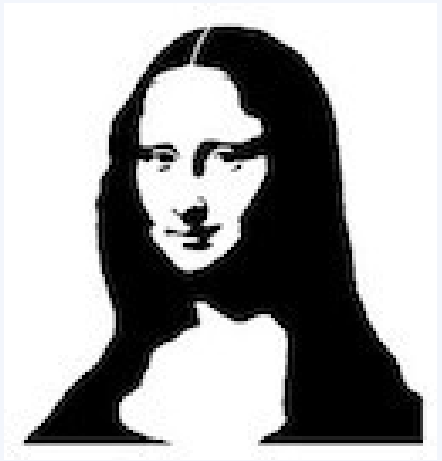


Image Denoising

- Assume that the image has been sent through a noisy channel, where each pixel is flipped with a small probability ε.
- The true pixels x are unobserved, and we observe the noisy pixels y.
- We have the following Ising model (MRF).

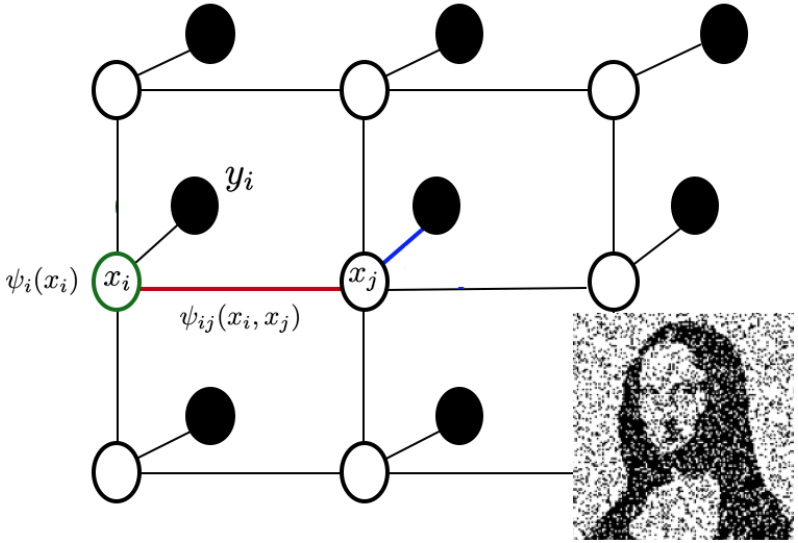


Image Denoising

- The goal of image denoising is to estimate x that maximizes $p(x \mid y) \propto p(x, y)$.

Each pixel is flipped independently with probability ε, so the observation model is

$$\begin{aligned}
\mathbb{P}(y_s \mid x_s) &= (1-\epsilon)^{\frac{1+y_s x_s}{2}}\, \epsilon^{\frac{1-y_s x_s}{2}} \qquad \text{for all } s \\
&= \exp\left\{ \tfrac{1+y_s x_s}{2} \log(1-\epsilon) + \tfrac{1-y_s x_s}{2} \log \epsilon \right\} \\
&\propto \exp\left\{ y_s x_s \, \tfrac{1}{2} \log\left(\tfrac{1-\epsilon}{\epsilon}\right) \right\} \\
&= \exp\{\beta \, y_s x_s\}, \qquad \text{where } \beta = \tfrac{1}{2} \log\left(\tfrac{1-\epsilon}{\epsilon}\right)
\end{aligned}$$

Ising Model

- We have the Ising model for the prior probability of x, where $s \sim t$ means $(s, t) \in \mathcal{E}$:

$$p(x) \propto \prod_{s \sim t} \psi_{s,t}(x_s, x_t)$$

$$\psi_{s,t}(x_s, x_t) = \exp(J\, x_s x_t)$$

Ising Model

- Therefore, we have

$$
\begin{aligned}
p(x \mid y) \propto p(x, y) &= p(x)\, p(y_1, \ldots, y_n \mid x_1, \ldots, x_n) \\
&= p(x) \prod_s p(y_s \mid x_s) \\
&\propto \exp\left\{ J \sum_{s \sim t} x_s x_t + \beta \sum_s y_s x_s \right\} \\
&= \prod_{s \sim t} \psi_{st}(x_s, x_t) \prod_s \psi_s(x_s)
\end{aligned}
$$

Loopy BP for Image Denoising

Messages

Each message $m_{j\to i}(x_i)$ is stored as **a 2-dimensional vector**, where its first and second coordinates are $m_{j\to i}(+1)$ and $m_{j\to i}(-1)$ respectively.

$$m_{j\to i} = \begin{pmatrix} m_{j\to i}(+1) \\ m_{j\to i}(-1) \end{pmatrix}$$

Beliefs

Similarly, beliefs $b(x_i)$ are **2-dimensional** vectors, with $b(+1)$ and $b(-1)$ being the first and second coordinates.

$$b_i = \begin{pmatrix} b(+1) \\ b(-1) \end{pmatrix}$$

Loopy BP for Image Denoising

- Initialize all messages uniformly: $m_{j\to i}(x_i) = 1/2$
- Keep doing BP updates until it (nearly) converges:

$$m_{j\to i}(x_i) = \sum_{x_j\in\{-1,+1\}} \psi_j(x_j)\,\psi_{ij}(x_i,x_j) \prod_{k\in N(j)\setminus\{i\}} m_{k\to j}(x_j)$$

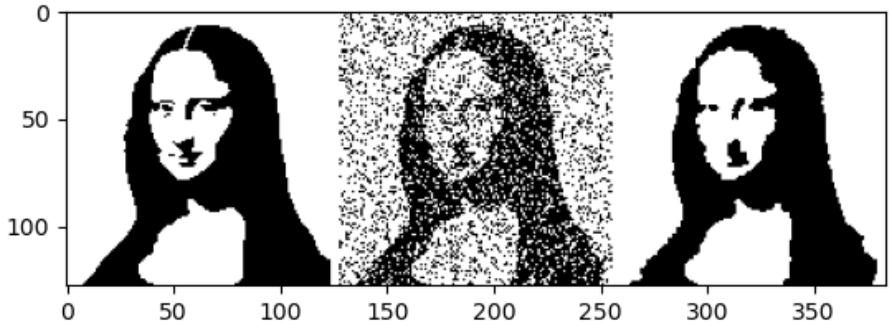
  and normalize for stability $\quad m_{j\to i}(x_i) \leftarrow \dfrac{m_{j\to i}(x_i)}{\sum_{x_i'} m_{j\to i}(x_i')}$

- Compute beliefs after message passing is done:

$$b(x_i) \propto \psi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i)$$

Loopy BP for Image Denoising

- While loopy BP may not converge, 10–20 iterations suffice to perform approximate inference on the posterior.
- The computed beliefs correspond to $p(x_i \mid y)$.
- One decision rule to estimate x_i is $\hat{x}_i = \underset{x_i}{\operatorname{argmax}}\ b(x_i)$



Original · Noisy observation · Loopy BP estimate