Inference over cycle-free discrete FGs

Introduction to Graphical Models and Inference for Communications

UC3M

March 3, 2018

Today

- We want to perform inference over discrete probability distributions that are represented by cycle-free Factor Graphs (a.k.a. trees).
- One critical result: inference over trees can be done at low cost $\mathcal{O}(n)$, where n is the number of latent variables.

Exact Inference: the discrete case

Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two discrete random vectors with joint pmf $p_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x} \in \mathcal{X}^n$, $\boldsymbol{y} \in \mathcal{Y}^m$. For a given observation $\boldsymbol{Y} = \boldsymbol{y}$, we are interested in drawing conclusions about $\boldsymbol{X}$.

Two basic kind of computations:

Find the most probable realization of the posterior probability distribution:

$$\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x} \in \mathcal{X}^n} p_{\boldsymbol{X}|\boldsymbol{y}}(\boldsymbol{x}) = \arg\max_{\boldsymbol{x} \in \mathcal{X}^n} \frac{p_{\boldsymbol{Y}|\boldsymbol{x}}(\boldsymbol{y}) p_{\boldsymbol{X}}(\boldsymbol{x})}{p_{\boldsymbol{Y}}(\boldsymbol{y})} = \arg\max_{\boldsymbol{x} \in \mathcal{X}^n} p_{\boldsymbol{Y}|\boldsymbol{x}}(\boldsymbol{y}) p_{\boldsymbol{X}}(\boldsymbol{x})$$

Complexity $\rightarrow \mathcal{O}(|\mathcal{X}|^n)$.

Exact Inference: the discrete case (II)

Compute the *marginal* probability distribution for $X_i \subset \{X_1, X_2, \ldots, X_n\}$:

$$\begin{aligned} p_{X_i\,|\,\boldsymbol{y}}(x_i) &= \sum_{\boldsymbol{x}_{\sim i} \in \mathcal{X}^{n-1}} p_{\boldsymbol{X}\,|\,\boldsymbol{y}}(\boldsymbol{x}) = \sum_{x_i \in \mathcal{X}^{n-1}} \frac{p_{\boldsymbol{Y}\,|\,\boldsymbol{x}}(\boldsymbol{y}) p_{\boldsymbol{X}}(\boldsymbol{x})}{p_{\boldsymbol{Y}}(\boldsymbol{y})} \\ &= \frac{1}{p_{\boldsymbol{Y}}(\boldsymbol{y})} \sum_{x_i \in \mathcal{X}^{n-1}} p_{\boldsymbol{Y}\,|\,\boldsymbol{x}}(\boldsymbol{y}) p_{\boldsymbol{X}}(\boldsymbol{x}), \end{aligned}$$

Complexity $\rightarrow \mathcal{O}(|\mathcal{X}|^n)$.

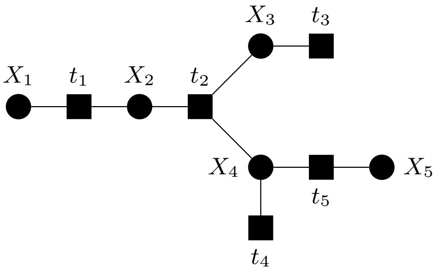
Index

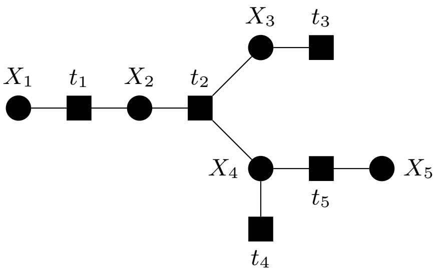
Motivation

Consider the random vector $\boldsymbol{X} = \{X_1, X_2, \ldots, X_5\}$ with p.m.f.

$$p_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{Z} t_1(x_1, x_2) t_2(x_2, x_3, x_4) t_3(x_3) t_4(x_4) t_5(x_4, x_5), \qquad \boldsymbol{x} \in \mathcal{X}^5$$

where $t_j : \mathcal{X}^{d_j} \to \mathbb{R}^+$ for $j = 1, \ldots, 5$.





Assume we want to compute

$$p_{X_1}(x_1) = \sum_{\boldsymbol{x} \sim x_1} p_{\boldsymbol{X}}(\boldsymbol{x})$$

By brute force... $|\mathcal{X}|^5$ sums.

The variable elimination method

$$p_{X_1}(x_1) = \frac{1}{Z} \sum_{x_2,x_3,x_4,x_5 \in \mathcal{X}^4} t_1(x_1,x_2)t_2(x_2,x_3,x_4)t_3(x_3)t_4(x_4)t_5(x_4,x_5)$$

- Distributive law ...

$$p_{X_1}(x_1) = \frac{1}{Z} \sum_{x_2,x_3,x_4 \in \mathcal{X}^3} t_1(x_1,x_2)t_2(x_2,x_3,x_4)t_3(x_3)t_4(x_4) \underbrace{\sum_{x_5 \in \mathcal{X}} t_5(x_4,x_5)}_{t_6(x_4)},$$

- $t_6 : \mathcal{X} \to \mathbb{R}^+$ is only a function of x_4 and its computation for all $x_4 \in \mathcal{X}$ requires $|\mathcal{X}|^2$ additions.

$$p_{X_1}(x_1) = \frac{1}{Z} \sum_{x_2, x_3 \in \mathcal{X}^2} t_1(x_1, x_2) t_3(x_3) \overbrace{\sum_{x_4 \in \mathcal{X}} t_2(x_2, x_3, x_4) t_4(x_4) t_6(x_4)}^{t_7(x_2, x_3)}$$

$$= \frac{1}{Z} \sum_{x_2 \in \mathcal{X}} t_1(x_1, x_2) \underbrace{\sum_{x_3 \in \mathcal{X}} t_3(x_3) t_7(x_2, x_3)}_{t_8(x_2)}$$

$$= \frac{1}{Z} \underbrace{\sum_{x_2 \in \mathcal{X}} t_1(x_1, x_2) t_8(x_2)}_{t_9(x_1)} = \frac{t_9(x_1)}{Z}$$

Overall, the variable elimination method computes the marginal $p_{X_1}(x_1)$ in $|\mathcal{X}|^3 + 3|\mathcal{X}|^2$ additions. (**Brute force $|\mathcal{X}|^5$ sums**).

- Imagine we are also interested in he marginal p.m.f. for the random variable X_2.
- We could follow the same procedure and compute the marginal after $|\mathcal{X}|^3 + 3|\mathcal{X}|^2$ additions.
- **Good news!** Note that most of the computations needed to compute $p_{X_2}(x_2)$ coincide with those performed to compute $p_{X_1}(x_1)$!

$$p_{X_2}(x_2) = \frac{1}{Z} \sum_{x_1,x_3 \in \mathcal{X}^2} t_1(x_1,x_2) t_3(x_3) \overbrace{\sum_{x_4 \in \mathcal{X}} t_2(x_2,x_3,x_4) t_4(x_4) t_6(x_4)}^{t_7(x_2,x_3)}$$

$$= \frac{1}{Z} \sum_{x_2 \in \mathcal{X}} t_1(x_1,x_2) \underbrace{\sum_{x_3 \in \mathcal{X}} t_3(x_3) t_7(x_2,x_3)}_{t_8(x_2)}$$

$$= \frac{1}{Z} \underbrace{\sum_{x_1 \in \mathcal{X}} t_1(x_1,x_2) t_8(x_2)}_{t_{10}(x_2)} = \frac{t_{10}(x_2)}{Z}$$

- If we had store intermediate computations required to compute $p_{X_1}(x_1)$, then computing $p_{X_2}(x_2)$ **only requires $|\mathcal{X}|^2$ additions**.
- What if we need to compute **all marginals** $p_{X_i}(x_i)$, $i = 1, \ldots, n$?
- The Sum-Product algorithm (a.k.a. the BP algorithm)!

Index

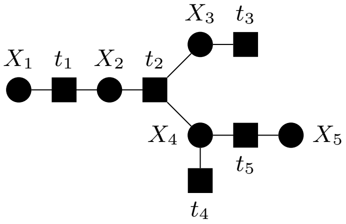
The BP algorithm

- Systematic implementation of the variable elimination procedure.
- Efficient computation of the the marginal probability distribution for any variable node when the joint distribution maps over a cycle-free factor graph (tree factor graph).

Consider a discrete random vector $\boldsymbol{X}$ with probability distribution

$$p_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{Z} \prod_{j \in \mathtt{J}} t_j(\boldsymbol{x}_j) \qquad x_i \in \mathcal{X}, \quad i = 1, \ldots, n$$

and assume that the FG associated is a tree.



$\mathtt{Ne}(X_4) = \{2, 4, 5\}$

$\mathtt{Ne}(t_2) = \{2, 4, 3\}$

- $\mathtt{Ne}(X_i)$ represents the index set of those factor nodes connected to the variable node X_i.
- $\mathtt{Ne}(t_j)$ the index set of those variable nodes connected to the factor node t_j

A message-passing iterative algorithm

Local computations are regarded as *messages* between the nodes in the probabilistic graph.

$m_{x_i \to t_j}(x_i)$ denotes the message sent from node X_i to factor node t_j.

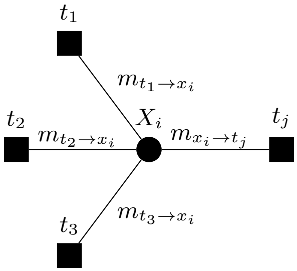
$m_{t_j \to x_i}(x_i)$ denotes the message sent from factor node t_j to variable node X_i

- Both messages are indeed functions of X_i, namely each message is a stored table of $|\mathcal{X}|$ values.
- At each iteration, messages are updated following simple **update rules** according to a valid **schedule**.

Update rules

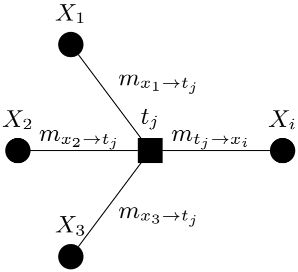
Variable-to-factor message

$$m_{x_i \to t_j}(x_i) = \prod_{\substack{k \in \mathtt{Ne}(X_i) \\ k \neq j}} m_{t_k \to x_i}(x_i)$$



Factor-to-variable message

$$m_{t_j \to x_i}(x_i) = \sum_{\boldsymbol{x}_j \sim x_i} t_j(\boldsymbol{x}_j) \prod_{\substack{m \in \mathtt{Ne}(t_j) \\ m \neq i}} m_{x_m \to t_j}(x_m)$$

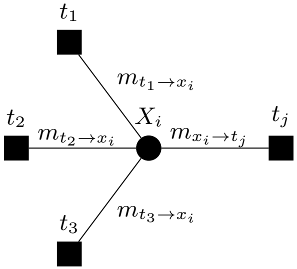


Update rules

Variable-to-factor message

$$m_{x_i \to t_j}(x_i) = \prod_{\substack{k \in \mathtt{Ne}(X_i) \\ k \neq j}} m_{t_k \to x_i}(x_i)$$

$|\mathtt{Ne}(X_i)| \times |\mathcal{X}|$ non-trivial multiplications

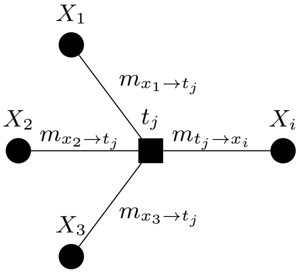


Factor-to-variable message

$$m_{t_j \to x_i}(x_i) = \sum_{\boldsymbol{x}_j \sim x_i} t_j(\boldsymbol{x}_j) \prod_{\substack{m \in \mathtt{Ne}(t_j) \\ m \neq i}} m_{x_m \to t_j}(x_m)$$

$|\mathtt{Ne}(t_j)| \times |\mathcal{X}|$ non-trivial multiplications

$|\mathtt{Ne}(t_j)|^{|\mathcal{X}|}$ additions



Updating message schedule

Messages will be updated according to a valid updating message schedule. Any updated schedule must satisfy the following:

1. After initialization, a message is sent for the first time from a node only when that node has received all requisite messages.
2. A message is updated and resent from a node to all its neighbours only when that node has received an updated incoming message.

Convergence is guaranteed for any tree factor graph. After convergence, the marginal p.m.f. $p_{X_i}(x_i)$ for $i = 1, \ldots, n$ can be computed as

$$p_{X_i}(x_i) = \frac{1}{Z_i} \prod_{k \in \mathtt{Ne}(X_i)} m_{t_k \to x_i}(x_i),$$

where the normalization constant is trivially obtained using the fact that the marginal must sum up to 1.

Initialization

Two common choices

- **Initialization at the leaf nodes.** Leaf nodes broadcast their messages to their respective neighbors from the start. Messages are initialized as follows:

$$\text{For all leaf variable nodes:} \quad m_{x_i \to t_j}(x_i) = 1$$

$$\text{For all leaf factor nodes:} \quad m_{t_j \to x_i}(x_i) = \sum_{\boldsymbol{x}_j \sim x_i} t_j(\boldsymbol{x}_{t_j})$$

- **Global initialization.** We set all the initial messages from variables to 1.

$$\text{For all variable nodes:} \quad m_{x_i \to t_j}(x_i) = 1 \quad x_i \in \mathcal{X}$$

Initialization

- **Initialization at the leaf nodes. Recommended for cycle-free FGs**. Messages will flow through the tree, one in each direction along every edge, and after a number of steps equal to the diameter of the graph, every message will have been created.
- **Global initialization.** Note that the global initialization leads to a load of wasted computations, whose results are gradually flushed out by the correct answer computed by the first initialization method. **This is the standard initialization when the graph is not cycle-free**.

Index

Message normalization

- After several iterations of the SP algorithm, the numerical value of messages can become very small and numerical precision issues can occur.
- **Large graphs**.
- One simple way to fix this issue is to normalize the variable-to-factor messages so they represent a valid p.m.f. (it sums up to one):

$$m_{x_i \to t_j}(x_i) = \frac{\prod\limits_{\substack{k \in \texttt{Ne}(X_i) \\ k \neq j}} m_{t_k \to x_i}(x_i)}{\sum\limits_{x_i \in \mathcal{X}} \prod\limits_{\substack{k \in \texttt{Ne}(X_i) \\ k \neq j}} m_{t_k \to x_i}(x_i)}$$

This method avoids numerical problems but it is only valid when we are interested in computing the marginal probability distribution of the variables in the graph.

The BP messages can be also used to efficiently compute (or approximate) the normalization constant Z. In this case, we cannot normalize the BP messages.

Log-messages

Another useful computational trick involves passing the logarithms of the messages. Define:

$$l_{x_i \to t_j}(x_i) = \log m_{x_i \to t_j}(x_i)$$

then the variable-to-factor update rule becomes simply:

$$l_{x_i \to t_j}(x_i) = \sum_{\substack{k \in \mathtt{Ne}(X_i) \\ k \neq j}} l_{t_k \to x_i}(x_i)$$

More care is required for the update of the factor-to-variable messages:

$$l_{t_j \to x_i}(x_i) = \log \left(\sum_{\boldsymbol{x}_j \sim x_i} t_j(\boldsymbol{x}_j) \exp \left(\sum_{\substack{m \in \mathtt{Ne}(t_j) \\ m \neq i}} l_{x_m \to t_j}(x_m) \right) \right).$$

The exponentiation of the log messages will cause potential numerical precision problems.

Log-messages (II)

By finding the largest value of the incoming log messages, we can express the factor-to-variable update rule as follows:

$$l^*_{\to t_j} = \max_{\substack{m \in \mathtt{Ne}(t_j) \\ m \neq i}} \max_{x_m \in \mathcal{X}} l_{x_m \to t_j}(x_m)$$

and then

$$l_{t_j \to x_i}(x_i) = l^*_{\to t_j} + \log\left(\sum_{\boldsymbol{x}_j \sim x_i} t_j(\boldsymbol{x}_j) \exp\left(\sum_{\substack{m \in \mathtt{Ne}(t_j) \\ m \neq i}} l_{x_m \to t_j}(x_m) - l^*_{\to t_j}\right)\right),$$

where all exponential terms will be ≤ 1 by construction. This ensures that the dominant numerical contributions to the summation are computed accurately.