Max-Product algorithm for cycle-free discrete FGs.

Introduction to Graphical Models and Inference for Communications

UC3M

March 3, 2018

uc3m

Today

- We present an algorithm to efficiently compute the mode of a probability distribution that factorizes in a cycle-free Factor Graph.
- The max-product algorithm.
- Similar to BP.

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

Index

We might in many cases be interested in determining which valid configuration $\boldsymbol{x}^* \in \mathcal{X}^n$ maximizes a given p.m.f. $p_{\boldsymbol{X}}(\boldsymbol{x})$:

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} p_{\boldsymbol{X}}(\boldsymbol{x}).$$

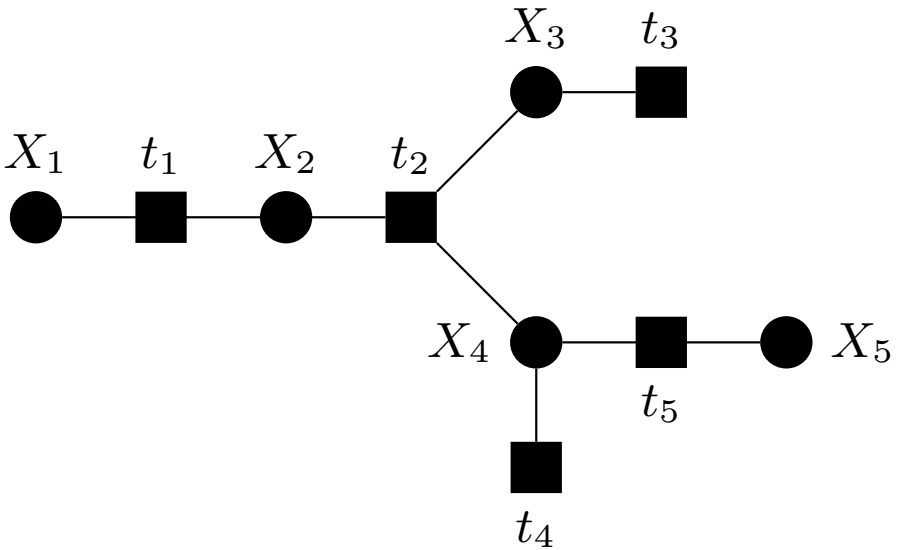
Analogous to the BP algorithm for marginalization, this problem can be efficiently solved if $p_{\boldsymbol{X}}(\boldsymbol{x})$ factorizes over a cycle-free FG.

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 $j = 1, \ldots, 5$.



Since factors are non-negative, we may write:

$$\begin{aligned}
\max_{\boldsymbol{x}\in\mathcal{X}^5} p_{\boldsymbol{X}}(\boldsymbol{x}) &= \max_{\boldsymbol{x}\in\mathcal{X}^5} t_1(x_1,x_2)t_2(x_2,x_3,x_4)t_3(x_3)t_4(x_4)t_5(x_4,x_5) \\
&= \max_{(x_1,x_2,x_3,x_4)\in\mathcal{X}^4} t_1(x_1,x_2)t_2(x_2,x_3,x_4)t_3(x_3)t_4(x_4) \underbrace{\max_{x_5\in\mathcal{X}} t_5(x_4,x_5)}_{t_6(x_4)} \\
&= \max_{(x_1,x_2)\in\mathcal{X}^2} t_1(x_1,x_2) \underbrace{\max_{(x_3,x_4)\in\mathcal{X}^2} t_2(x_2,x_3,x_4)t_3(x_3)t_4(x_4)t_6(x_4)}_{t_7(x_2)} \\
&= \max_{x_1\in\mathcal{X}} \underbrace{\max_{x_2\in\mathcal{X}} t_1(x_1,x_2)t_7(x_2)}_{t_9(x_1)} = \max_{x_1\in\mathcal{X}} t_9(x_1),
\end{aligned}$$

Computing the potentials $t_6(x_7), t_7(x_2)$ and $t_9(x_1)$ requires $\mathcal{X}^2, \mathcal{X}^3$ and $\mathcal{X}^2$ operations.

Backtracking

We obtain $\boldsymbol{x}^*$ recursively

- $x_1^* = \arg\max_{x_1 \in \mathcal{X}} t_9(x_1)$.
- $x_2^* = \arg\max_{x_2 \in \mathcal{X}} t_1(x_1^*, x_2) t_7(x_2)$.
- $(x_3^*, x_4^*) = \arg\max_{x_3, x_4 \in \mathcal{X}^2} t_2(x_2^*, x_3, x_4) t_3(x_3) t_4(x_4) t_6(x_4)$.
- $x_5^* = \arg\max_{x_5 \in \mathcal{X}} t_5(x_4^*, x_5)$.

The cycle-free structure of the graph ensures that the maximal value (and its state) can be computed in time which scales linearly with the number of factors in the function.

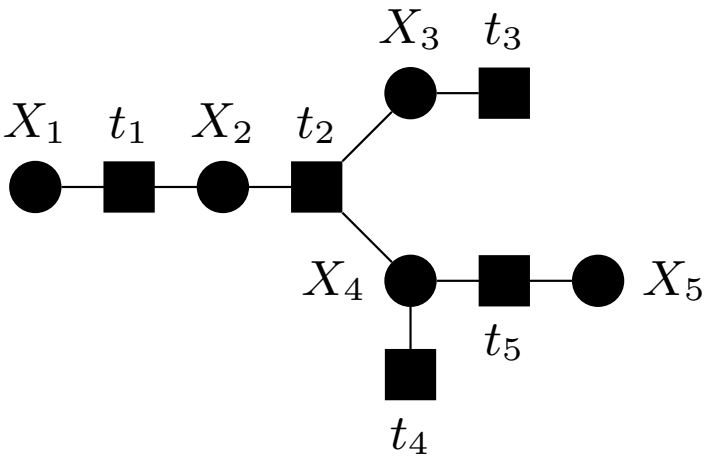
Index

The Max-Product (MP) message passing algorithm

- Message passing algorithm that defines a parallel procedure to avoid the sequential backtracking.
- Consider

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

and assume that the FG associated is a tree.



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

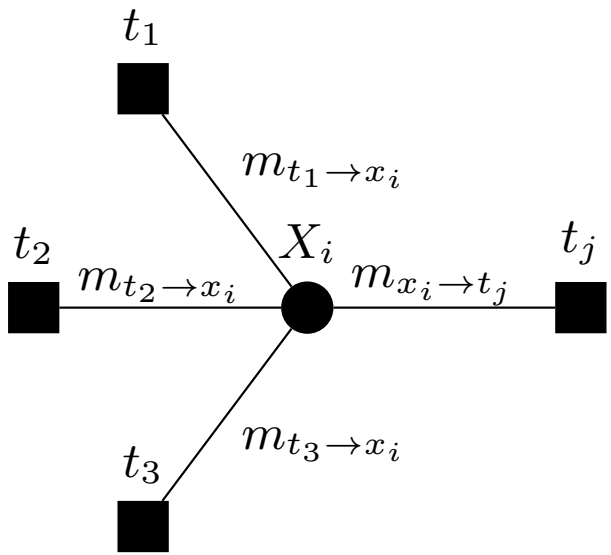
$\mathtt{Ne}(t_2) = \{2, 4, 3\}$

- $\mathtt{Ne}(X_i)$ 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

Update rules

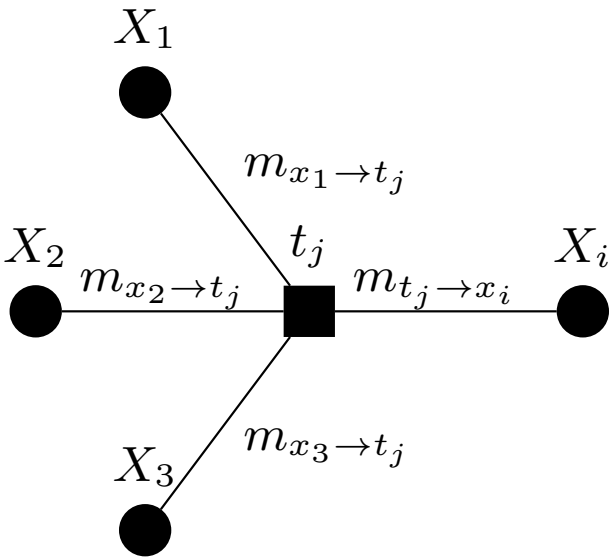
Variable-to-factor message

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



Factor-to-variable message

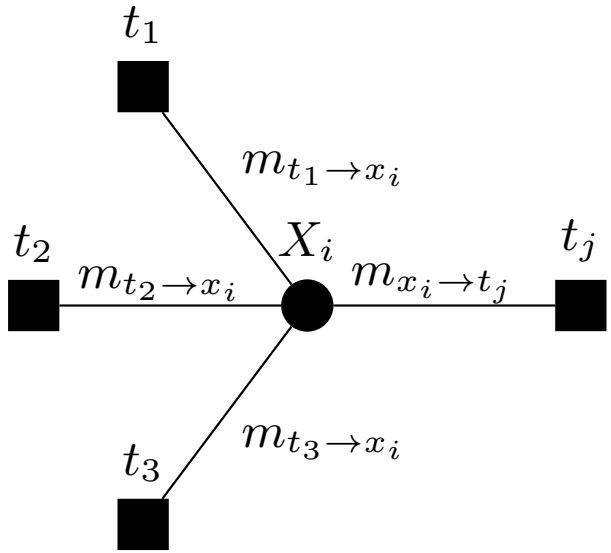
$$m_{t_j \to x_i}(x_i) = \max_{\boldsymbol{x}_j \sim x_i} \; t_j(\boldsymbol{x}_j) \prod_{\substack{m \in \mathtt{Ne}(t_j) \\ m \neq i}} m_{x_m \to t_j}(x_m)$$



Update rules

Variable-to-factor message

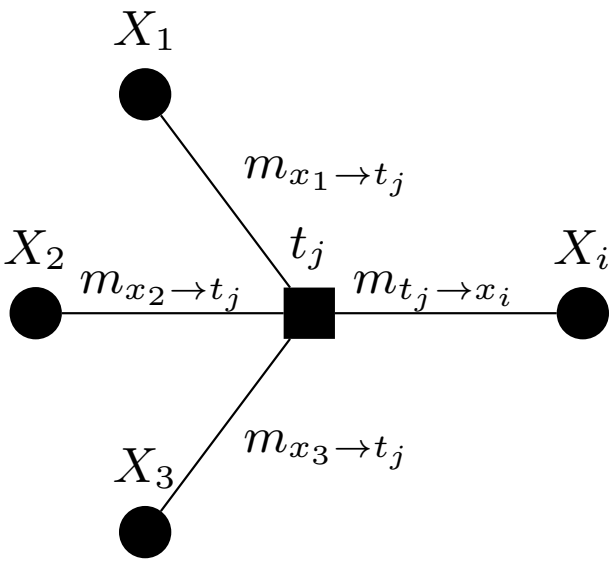
$$m_{x_i \to t_j}(x_i) = \prod_{\substack{k \in \mathtt{Ne}(X_i) \\ k \neq j}} m_{t_k \to x_i}(x_i)$$



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

Factor-to-variable message

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



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

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

Messages from leaf node factors are initialized to the factor and messages from the leaf variable nodes are set to all-one messages.

As in the case of the SP update rules, any valid updating schedule has to verify 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 maximal state for each variable node is computed as follows:

$$x_i^* = \arg\max_{x \in \mathcal{X}} \prod_{k \in \mathtt{Ne}(X_i)} m_{t_k \to x_i}(x_i)$$

Index

The Viterbi Algorithm

Note that the MP algorithm applied to a *state-space model* is equivalent to the Viterbi algorithm.

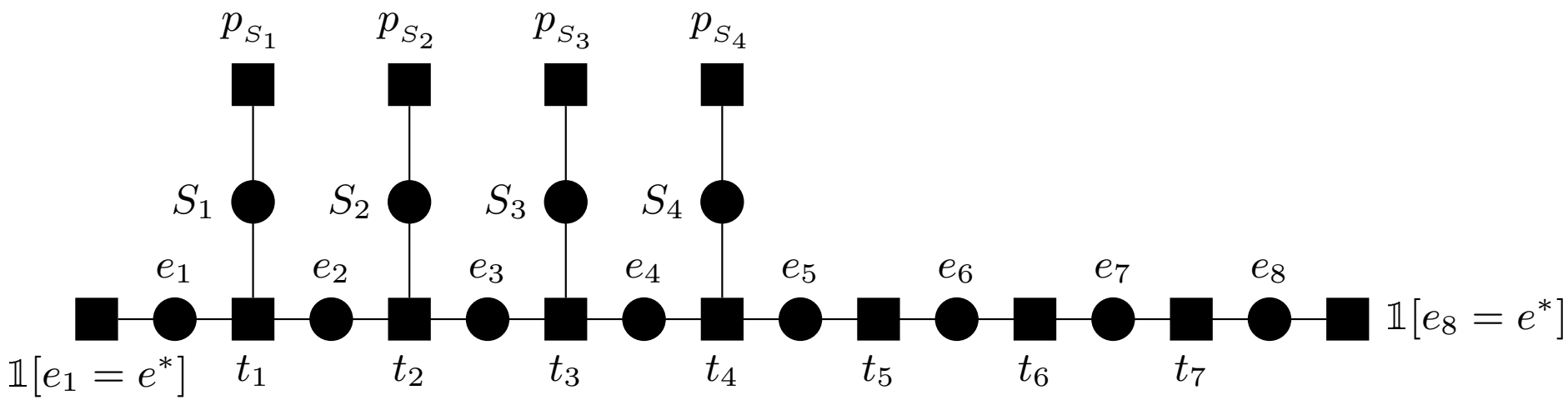


Figure: Factor graph associated tothe joint distribution of symbols and states in the scenario of tramission of QAM symbols over an ISI channel. $\mathtt{h} = 3$ and $\mathtt{s} = 4$.