Recap.

Inference Algorithms

	Marginalization	Maximization			
√	Elimination Algo	Elimination Algo	Exact,	any G,	$O(\lvert\mathcal{X}\rvert^{\text{Treewidth}})$ at best but NP-hard to find ordering
√	sum-product = Belief Propagation	Max-product	Approx.,	Pairwise MRF,	$O(\lvert\mathcal{X}\rvert^{2})$
	Sum-product algorithm on factor graphs	Max-product algo on factor graphs	Approx,	any G,	$O(\lvert\mathcal{X}\rvert^{\text{max-degree}})$
	Elimination algorithm on Junction trees	Elimination Algorithm on Junction trees	Exact,	Junction Trees,	$O(\lvert\mathcal{X}\rvert^{\text{Treewidth}})$ NP-hard to find the best ordering.

---

* Belief Propagation for factor graphs.

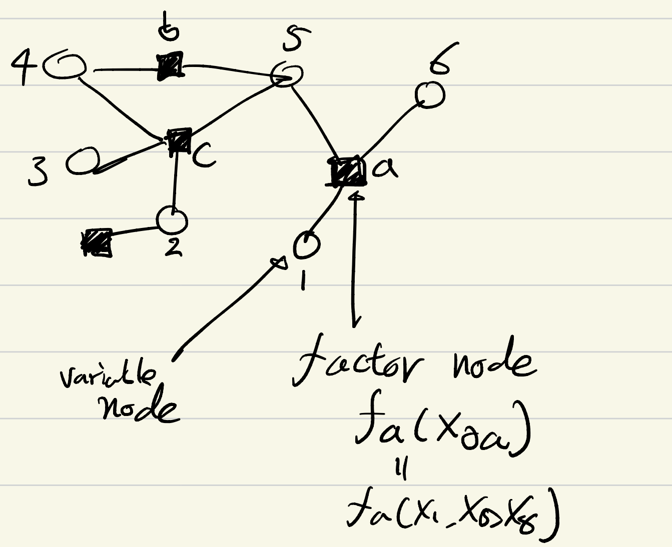
Def. factor graph $G(V \cup F, E)$

V: variable nodes

W: factor nodes

a probability distribution factorizes according to G if

$$P(x) = \frac{1}{Z} \prod_{a \in F} f_a(x_{\partial a})$$



- factor graphs are strict generalizations of MRFs & can encode more detailed factorizations. fine grained

- FGs have the same Markov property as MRFs.

  A-B-C separated in $G = (V \cup F, E)$, then $X_A \perp\!\!\!\perp X_C \mid X_B$.

Belief Propagation

- Sum-Product Algorithm on factor graphs

recall: $E = \{ (\underset{\text{variable}}{i}, \underset{\text{factor}}{a}) \}$

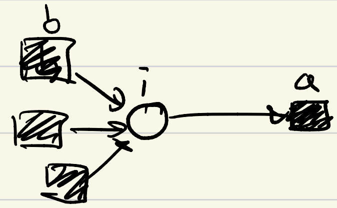
Input: $G = (V \cup F, E)$, T,

output: $\{ \hat{P}(X_i) \}_{i \in V}$

Initialize: $\{ (\underbrace{m_{i \to a}}_{[\ ]^{|\mathcal{X}|}}, m_{a \to i}) \}_{(i,a) \in E} = \vec{\mathbb{1}} \frac{1}{|\mathcal{X}|}$ or RANDOM

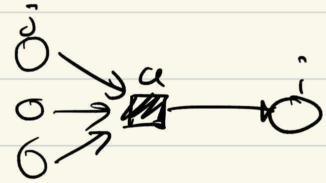
Repeat $t = 1 \ldots T$

Update $m_{i \to a}$'s:



$$m_{i \to a}(X_i) = \prod_{b \in \partial i \setminus \{a\}} \tilde{m}_{b \to i}(X_i)$$

Update $\tilde{m}_{a \to i}$'s:



$$\underbrace{\tilde{m}_{a \to i}(X_i)}_{P(X_i)} = \sum_{\underbrace{\partial a \setminus \{i\}}_{\text{Cond dist}}} f_a(X_{\partial a}) \prod_{\underbrace{j \in \partial a \setminus \{i\}}_{\text{Prior on } X_j\text{'s}}} m_{j \to a}(X_j)$$

has the interpretation of

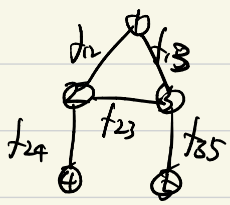
Cond dist $P(X_i | X_{j_1}, X_{j_2}, X_{j_3})$ Prior on X_j's $P(X_{j_1}) P(X_{j_2}) P(X_{j_3})$

Compute marginal:

$$m_i(X_i) = \prod_{a \in \partial i} \tilde{m}_{a \to i}(X_i)$$

Claim: this marginal is exact if FG is a tree

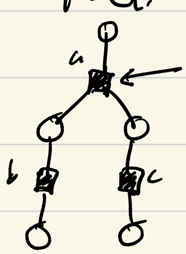
Pairwise MRF



BP is approximate as there is a loop.

Complexity $O(|\mathcal{X}|^2)$

F.G.



$f_a = f_{12} \cdot f_{13} \cdot f_{23}$

BP is exact as it is a tree

Complexity $O(|\mathcal{X}|^{3}_{p})$

max degree of graph

Example > Decoding LDPC codes. – one of most ~~successful~~ impactful application of B.P.

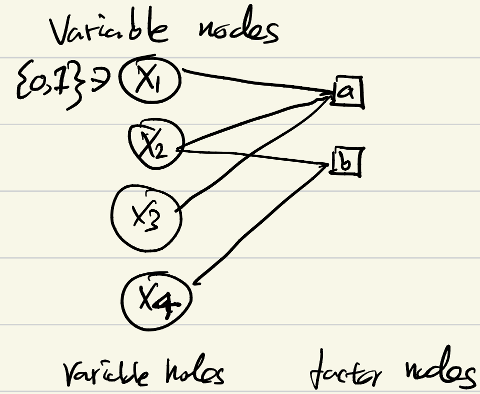
Def. LDPC (Low Density Parity Check) codes are a family of codes defined as follows

$$G = (V \cup F, E)$$

with factors for parity checking:

$$f_a(X_{\partial a}) = \mathbb{I}(X_1 \underset{\text{XOR}}{\oplus} X_2 \oplus X_3 = 0)$$

$$X_{\partial a} = X_a$$



Def. $\text{Codebook} = \{X \in \mathbb{Z}_2^n \mid \text{satisfy all parities}\} = \left\{\begin{matrix} 0000 \\ 0111 \\ 1010 \\ 1101 \end{matrix}\right\}$

$$|\text{Codebook}| = 2^{\overbrace{|V| - |F|}^{}_{K}} = 2^K$$

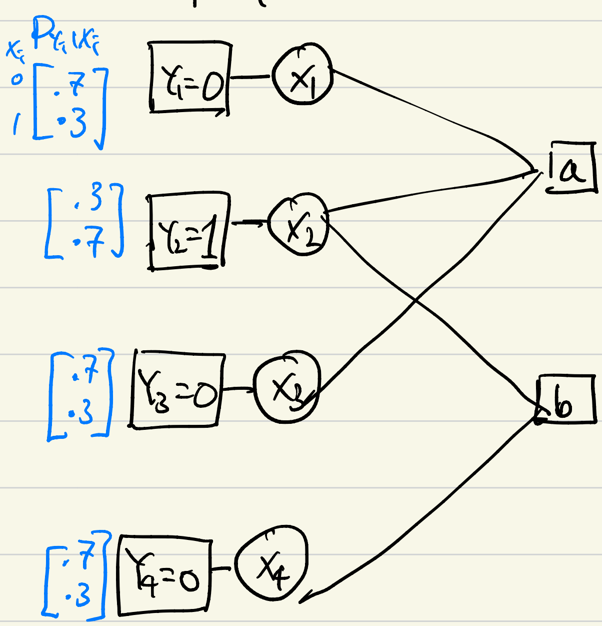
At transmission, one of the codeword from codebook is sent over a noisy channel, defined by $P(Y_i \mid X_i)$ and we need to recover X from observed Y's.

$$\text{e.g. } P(Y_i \mid X_i) = \begin{array}{c|cc|c} & 0 & 1 & \\ & X_i & & \\ \hline & .7 & .3 & 0 \\ & .3 & .7 & 1 \\ & & & Y_i \end{array}$$

Strategy: Use Belief Propagation to estimate $P(X_i \mid Y_1 \cdots Y_n)$, $\forall i \in [n]$ and output $\left(\text{sign}\left(\log \dfrac{P(X_i = 1 \mid Y)}{P(X_i = 0 \mid Y)}\right) + 1\right)\dfrac{1}{2} \in \{0, 1\}$

* How to start with a BN formulation and get Factor Graph seamlessly.
* How to include observed variables in inference/Modeling.

F. G



factorization:

$$P(X|Y) \propto P(X,Y) = \frac{1}{Z} \underbrace{\prod_{a\in F} \mathbb{I}(\oplus x_a = 0)}_{\text{only code word is sent}} \times \underbrace{\prod_{i\in V} P(Y_i|X_i)}_{\text{this is a singleton factor}}$$

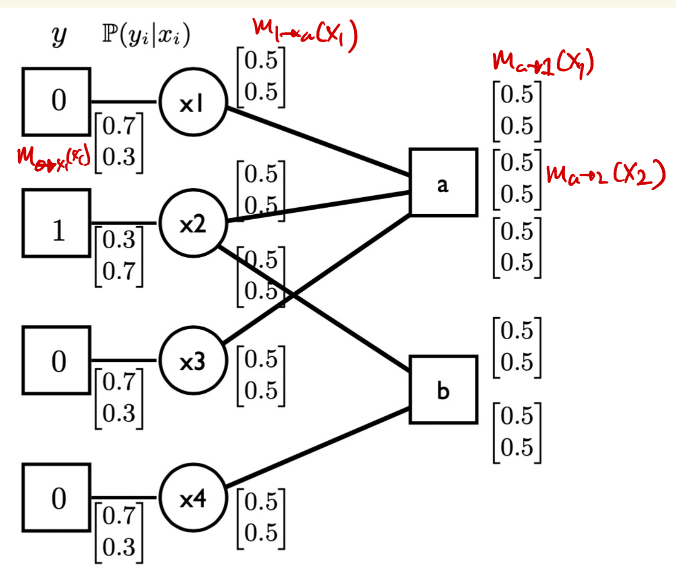
$$P(Y_i|X_i) = \begin{bmatrix} .7 & .3 \\ .3 & .7 \end{bmatrix}$$

$$f_a = \begin{array}{cc} & (x_2, x_3) \\ & \begin{array}{cccc} 00 & 01 & 10 & 11 \end{array} \\ x_1 \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

$$f_b = \begin{array}{cc} & x_2 \\ & \begin{array}{cc} 0 & 1 \end{array} \\ x_4 \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array}$$

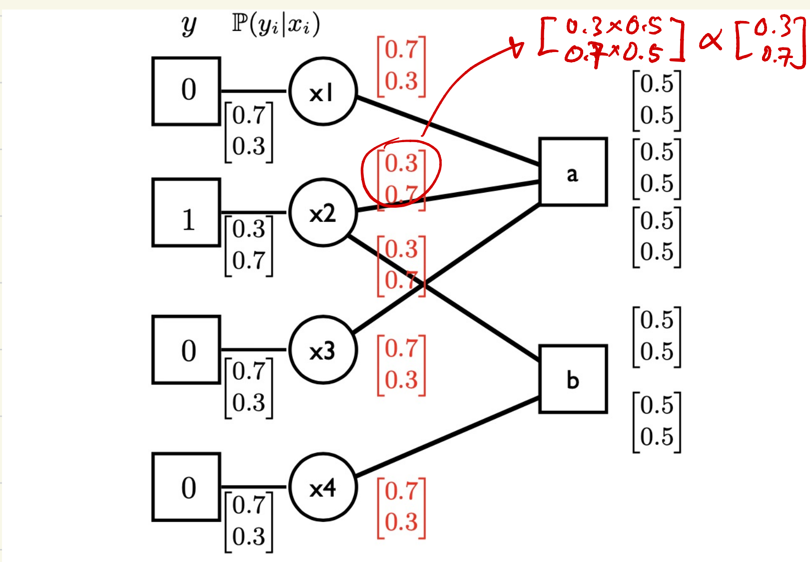
$$m_{i\to a}(x_i) = P(Y_i|X_i) \cdot \prod_{b\in\partial i\setminus\{a\}} \tilde{m}_{b\to i}(x_i)$$

$$\tilde{m}_{a\to i}(x_i) = \sum_{x_{\partial a}\setminus\{i\}} \prod_{j\in\partial a\setminus\{i\}} m_{j\to a}(x_j) \cdot \mathbb{I}(\oplus x_{\partial a} = 0)$$



$\nu_{i\to a}^{(t+1)}(x_i) = \mathbb{P}(y_i|x_i) \prod_{b\in\partial i\setminus\{a\}} \tilde{\nu}_{b\to i}^{(t)}(x_i)$

$\tilde{\nu}_{a\to i}^{(t+1)}(x_i) = \sum_{x_{\partial a\setminus\{i\}}} \prod_{j\in\partial a\setminus\{i\}} \nu_{j\to a}(x_j)\mathbb{I}(\oplus x_{\partial a} = 0)$



$\nu_{i\to a}^{(t+1)}(x_i) = \mathbb{P}(y_i|x_i) \prod_{b\in\partial i\setminus\{a\}} \tilde{\nu}_{b\to i}^{(t)}(x_i)$

$\tilde{\nu}_{a\to i}^{(t+1)}(x_i) = \sum_{x_{\partial a\setminus\{i\}}} \prod_{j\in\partial a\setminus\{i\}} \nu_{j\to a}(x_j)\mathbb{I}(\oplus x_{\partial a} = 0)$

Sum-product algorithm

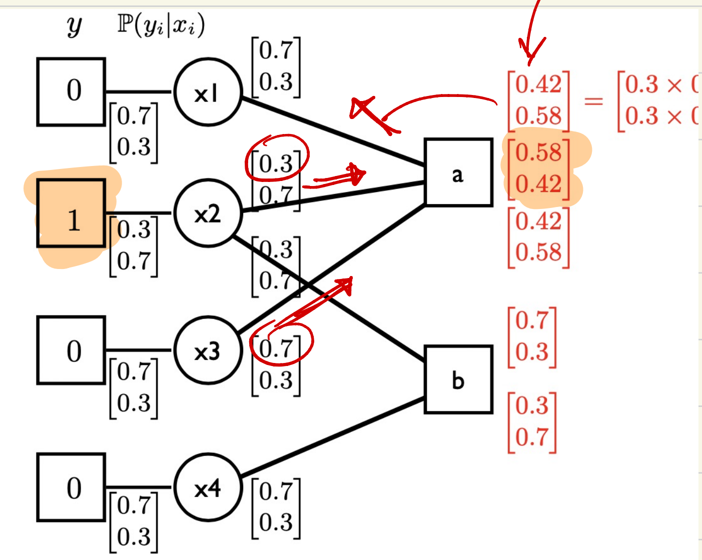
$m_{i\to a}$ Update

* normalization within each message does not matter

* often normalize after each update for numerical stability.

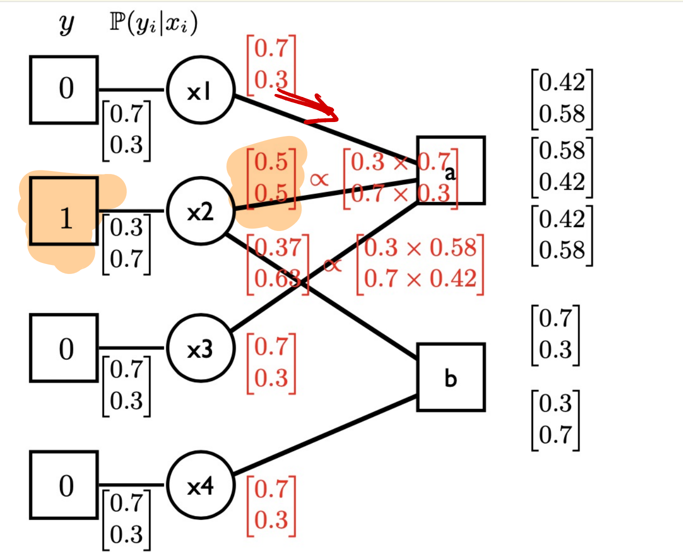
$m_{a\to 1}(X_1=0) = m_{2\to a}(0)\, m_{3\to a}(0) + m_{2\to a}(1)\, m_{3\to a}(1)$

$0.42 = 0.3\times 0.7 + 0.7\times 0.3$



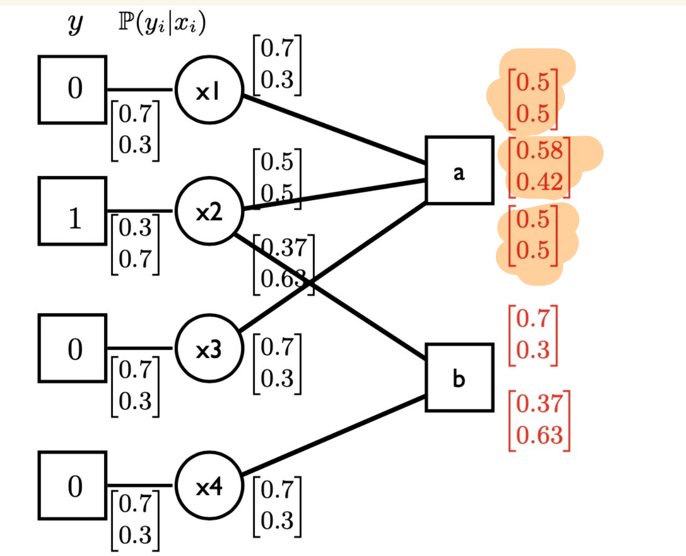
$\nu_{i\to a}^{(t+1)}(x_i) = \mathbb{P}(y_i|x_i)\prod_{b\in\partial i\setminus\{a\}} \tilde{\nu}_{b\to i}^{(t)}(x_i)$

$\tilde{\nu}_{a\to i}^{(t+1)}(x_i) = \sum_{x_{\partial a\setminus\{i\}}}\prod_{j\in\partial a\setminus\{i\}} \nu_{j\to a}(x_j)\mathbb{I}(\oplus x_{\partial a}=0)$



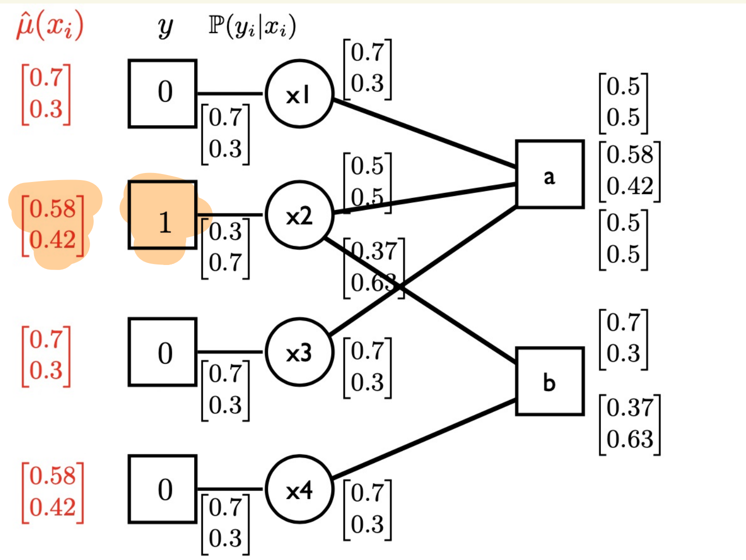
$\nu_{i\to a}^{(t+1)}(x_i) = \mathbb{P}(y_i|x_i)\prod_{b\in\partial i\setminus\{a\}} \tilde{\nu}_{b\to i}^{(t)}(x_i)$

$\tilde{\nu}_{a\to i}^{(t+1)}(x_i) = \sum_{x_{\partial a\setminus\{i\}}}\prod_{j\in\partial a\setminus\{i\}} \nu_{j\to a}(x_j)\mathbb{I}(\oplus x_{\partial a}=0)$



$\nu_{i\to a}^{(t+1)}(x_i) = \mathbb{P}(y_i|x_i)\prod_{b\in\partial i\setminus\{a\}} \tilde{\nu}_{b\to i}^{(t)}(x_i)$

$\tilde{\nu}_{a\to i}^{(t+1)}(x_i) = \sum_{x_{\partial a\setminus\{i\}}}\prod_{j\in\partial a\setminus\{i\}} \nu_{j\to a}(x_j)\mathbb{I}(\oplus x_{\partial a}=0)$



$\nu_{i\to a}^{(t+1)}(x_i) = \mathbb{P}(y_i|x_i)\prod_{b\in\partial i\setminus\{a\}} \tilde{\nu}_{b\to i}^{(t)}(x_i)$

$\tilde{\nu}_{a\to i}^{(t+1)}(x_i) = \sum_{x_{\partial a\setminus\{i\}}}\prod_{j\in\partial a\setminus\{i\}} \nu_{j\to a}(x_j)\mathbb{I}(\oplus x_{\partial a}=0)$

Codebook $\{0000,\ 0111,\ 1010,\ 1101\}$ $\xleftrightarrow{\text{minimum}}$ Hamming Distance

Output $(0,0,0,0)$

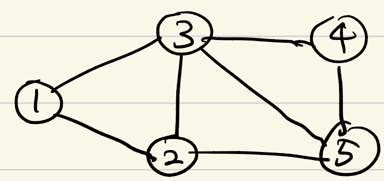
Received $(0,1,0,0)$

* Junction Tree Algorithm : Elimination on Junction $\boxed{\text{Tree}}$ ↳ exact.

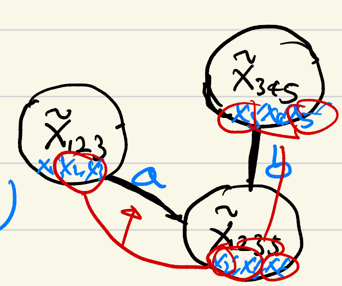
- Once the tree is constructed, inference is easy using any method.
- Conceptually the approach is closer to Elimination algo [exact ordering.
- finding the right data structure for elimination to be <u>efficient</u>

*Given a MRF $G=(V,E)$ we can construct clique Tree (none unique).

MRF: $P(x) = \frac{1}{Z} f_{123}(x_1,x_2,x_3)\, f_{235}(x_2,x_3,x_5)\, f_{345}(x_3,x_4,x_5)$



one example of a clique Tree for the MRF :
- Create a (giant) node for each clique
  Variable $\tilde{x}_C \in \mathcal{X}^{|C|}$
  each node has a local copy of a variable
  $\tilde{x}_{123} = (x_1, x_2, x_3)$
  $\tilde{x}_{235} = (x_2', x_3', x_5'),\ \tilde{x}_{345} = (x_3'', x_4'', x_5'')$
- assign edges so that it forms a tree



edges in C.T. ensure consistency of the local copies of variables.

Clique Tree CT_1

$$\tilde{P}(\tilde{x}_{123}, \tilde{x}_{235}, \tilde{x}_{345}) = \frac{1}{Z} f_{123}(\tilde{x}_{123}) \cdot f_{235}(\tilde{x}_{235}) \cdot f_{345}(\tilde{x}_{345}) \cdot \mathbb{I}([\tilde{x}_{123}]_2 = [\tilde{x}_{235}]_1)$$

$$\mathbb{I}([\tilde{x}_{235}]_3 = [\tilde{x}_{345}]_3) \cdot \mathbb{I}([\tilde{x}_{235}]_2 = [\tilde{x}_{345}]_1) \qquad \mathbb{I}([\tilde{x}_{123}]_3 = [\tilde{x}_{235}]_2)$$

$[\tilde{x}_{235}]_2 = x_3'$, $[\tilde{x}_{345}]_1 = x_3''$

$f_b(\tilde{x}_{345}, \tilde{x}_{235})$ $\qquad$ $f_a(\tilde{x}_{123}, \tilde{x}_{235})$

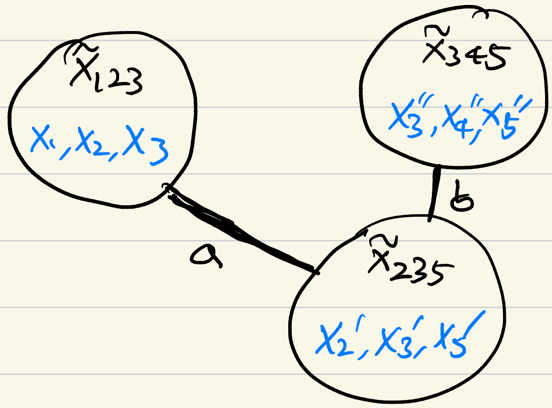
* add consistency for all shared variables

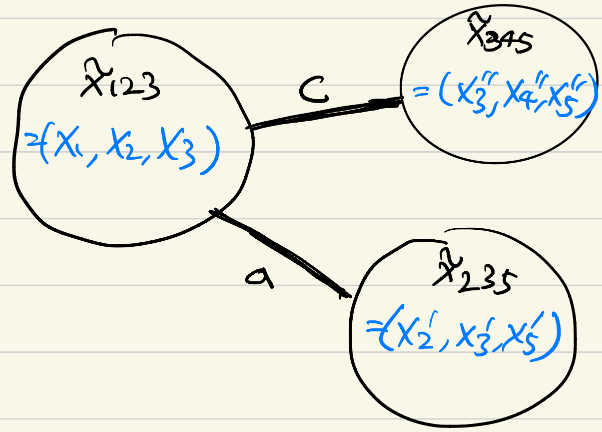
$$= \frac{1}{Z} f_a(\tilde{x}_{123}, \tilde{x}_{235}) \cdot f_b(\tilde{x}_{345}, \tilde{x}_{235}).$$

*If all local copies can be made consistent, then we have <u>global consistency</u>

$$P(x_1) = \sum_{x_2, x_3} \tilde{P}(\tilde{x}_{123})$$

example when Global consistency is not achievable. (depending on the tree)





we cannot ensure $x_5'' = x_5'$

with factors $f_c(\tilde{x}_{345}, \tilde{x}_{123})$

$f_a(\tilde{x}_{123}, \tilde{x}_{235})$

because this nodes are separated by a node that does not contain a local copy of x_5

Q. When is a clique tree globally consistent?

Def. Junction Tree Property.

Given a MRF $G=(V,E)$ with $\mathcal{C}$ set of maximal cliques,

for a tree T over $\mathcal{C}$, a node $i \in V$ satisfy J.T.P if

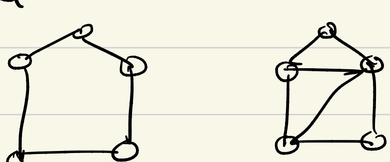
all cliques containing i form a connected subtree in T.

Def. T is a Junction Tree for G if all $i \in V$ has J.T.P.

[existence]

Q. - When does G have a Junction Tree?

If the graph G is chordal ($\triangleq$ any loop ≥ 4 has a chord).



4 loop without chord.

chordal

[Construction]

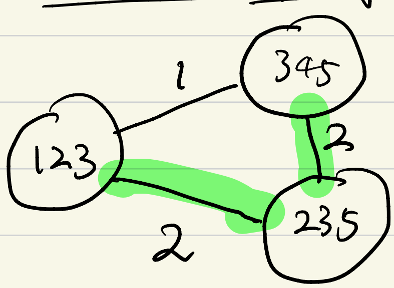
Q. How do we find a Junction tree? ①

Q. If the graph is not chordal, what can we do? ②

① Algorithm Sketch to find J.T. given $G=(V,E)$ with $\mathcal{C}$: set of max cliques.

- consider a complete graph on $\mathcal{C}$
- assign weight on edges as # of shared variables between the two cliques
- find the max weight spanning tree. ($\triangleq$ a tree whose sum of edge weight is maximized).

can be easily found by Kruskal's algorithm.



Claim. [Correctness] A clique tree T is a Junction tree if and only if it is a maximal spanning tree.

$$w(T) = \sum_{(C_\ell, C_r)\in T} |C_\ell \cap C_r|$$

for any tree

$$= \sum_{(C_\ell, C_r)\in T} \sum_{i\in V} \mathbb{I}(i\in C_\ell \,\&\, i\in C_r)$$

$$= \sum_{i\in V} \boxed{\sum_{(C_\ell, C_r)\in T} \mathbb{I}(i\in C_\ell \,\&\, i\in C_r)} \le m_i - 1$$

because cliques containing node i is a sub-tree of size m_i

cliques containing node i

$$\le \sum_{i\in V} (m_i - 1)$$

equal if and only if all subgraph ~~tree~~ containing a node i ~~form~~ is connected.

∴ Junction Tree T $\rightarrow$ achieve max-weight

② If you are given a graph $G=(V,E)$ that is not chordal.

sketch of algorithm

- choose an elimination ordering
- find the reconstituted graph (which is chordal)
- construct a complete clique graph
- Find max-weight spanning tree

$\Rightarrow$ Junction Tree

we can run any algorithm we learned on this Junction tree for exact inference

* Different Elimination ordering gives different complexity / max-clique size

at best we get $O(|\mathcal{X}|^{\text{treewidth}})$.