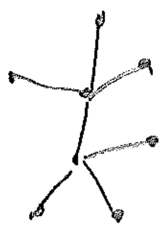
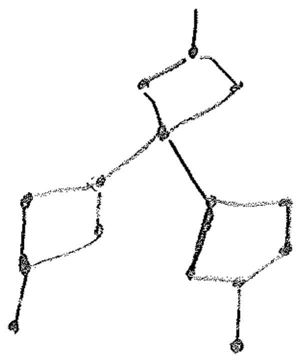
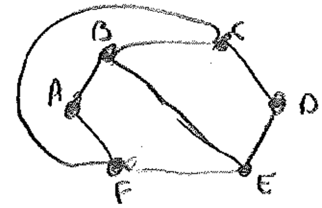
Recall, <u>width</u> of <u>induced</u> graph is # nodes in largest clique − 1

<u>Treewidth</u>, or "minimal induced width," of graph G is $\min_{\prec} \omega_{G,\prec}$, where $\omega_{G,\prec}$ is width of graph induced by applying elimination ordering $\prec$ to G.

What is the treewidth of...







?

(Affects running time of elimination alg – exponential in treewidth)

(Eliminate A + D, gives clique on B, C, E, F. Any other ordering clearly worse.)

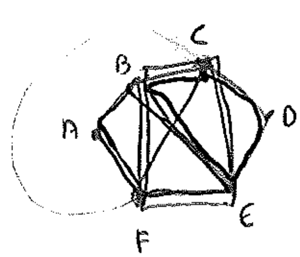
Graph is <u>chordal</u>, or triangulated, if every cycle of length ≥ 3 has a shortcut (called a "chord")

<u>Thm:</u> Every induced graph is chordal

<u>Thm:</u> Every chordal graph has an elimination ordering which does not introduce any fill edges

→ <u>found</u> by Max Cardinality algorithm, (see book), Sec. 9.4

e.g.



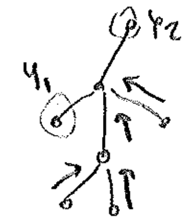
Conclusion
Finding a good elim. ordering is equivalent to making graph chordal with minimal width.

What if you want to compute marginals for many variables?

i.e. $pr(y_i \mid \vec{x}) \quad \forall i \in Y$?

Can re-run variable elimination, once for each $i \in Y$, but this duplicates much of the effort, e.g.

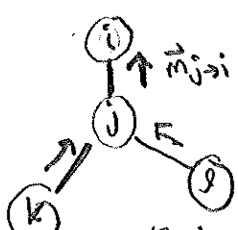
i.e. $p(x_1, \dots, x_n) \propto \prod \varphi_i \prod_{ij \in T} \varphi_{ij}$



Consider the case of trees. The sum-product belief propagation algorithm computes all marginals with just double the computation and using linear space.

Based on message-passing of "messages" (tables of partial summations) between neighboring vertices of the graph.

The message sent from variable j to $i \in N(j)$ is:



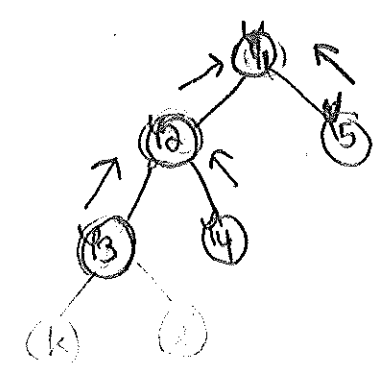
Sum-product messages

$$m_{j\to i}(x_i) = \sum_{x_j} \left(\phi_j(x_j)\, \phi_{ij}(x_i, x_j) \prod_{k \in N(j)\setminus i} m_{k\to j}(x_j) \right)$$

Each message $\vec{m}_{j\to i}$ is a vector with one value for each state of X_i.

→ In order to compute $\vec{m}_{j\to i}$, must already have $\vec{m}_{k\to j}$ for $k \in N(j)\setminus i$. Thus, there must be a specific ordering to the messages.

Suppose we want to compute $pr(y_1) \propto \sum_{y_2 \dots y_5} \prod \phi \dots$. Root tree at Y_1 and send messages from leafs to root:



Root $= Y_1$

$$m_{5\to 1}(y_1) = \sum_{y_5} \phi_5(y_5)\, \phi_{15}(y_1, y_5)$$

$$m'_{3\to 2}(y_2) = \sum_{y_3} \phi_3(y_3)\, \phi_{23}(y_2, y_3)$$

$$m_{4\to 2}(y_2) = \sum_{y_4} \phi_4(y_4)\, \phi_{24}(y_2, y_4)$$

$$m'_{2\to1}(y_1) = C_3 \sum_{y_2} \Phi_2(y_2)\, \Phi_{12}(y_1, y_2)\, m_{3\to2}(y_2)\, m_{4\to2}(y_2)$$

Finally, $p(y_1) \propto C_3 \Phi_1(y_1)\, m_{2\to1}(y_1)\, m_{5\to1}(y_1)$.

Elimination algorithm on trees is <u>equivalent</u> to message passing!

What about computing $pr(Y_5)$? Could re-run algorithm, rooted at Y_5. Or...

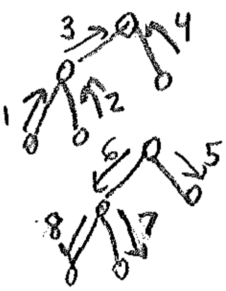
<u>Belief Propagation</u> (BP)

Input: Tree T with potentials $\phi_i(x_i)$, $\phi_{ij}(x_i, x_j)$ $\forall ij \in T$.

Choose root r. (arbitrary)

Pass messages from leafs to r.

Pass messages from r to leafs.



Compute $p(y_i) \propto \phi_i(y_i) \prod_{j \in N(i)} m_{j\to i}(y_i)$. $\forall i$

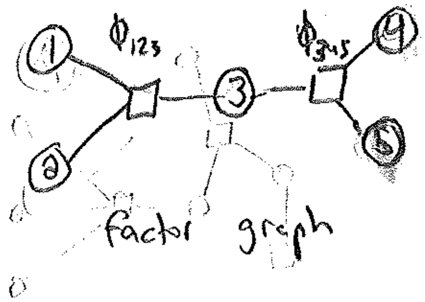
Running time = $2 \cdot$ cost of one var. elimination = $\Theta(nk^2)$, where n = # nodes and k = # states per node.

Possible numerical difficulties if multiplying many small factors. Notice that constants passed on from message to messages then canceled during normalization.

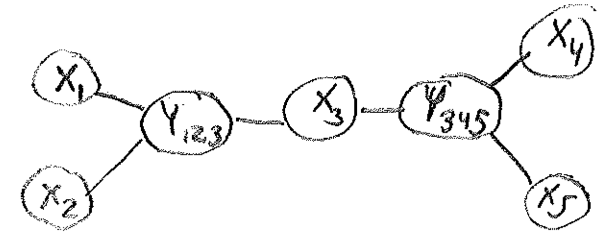
Alternative update:

$$m^{new}_{j\to i}(x_i) = \frac{m^{orig}_{j\to i}(x_i)}{\underbrace{\sum_{\hat{x}_i} m^{orig}_{j\to i}(\hat{x}_i)}_{\text{constant}}}$$

This applies to any tree-structured pairwise MRF. What about a tree-structured <u>factor graph</u>?



Use transf. from homework ⟶



pairwise MRF

$\phi_{123}(y_{123})$, $\phi_{345}(y_{345})$ single node potentials

$\phi(x_1, y_{123}) = \begin{cases} 0 & \text{if } x_1 \text{ inconsistent w/ } y_{123} \\ 1 & \text{o.w.} \end{cases}$ pairwise node potentials

Messages can be computed slightly more efficiently:

$$m_{i\to f}(y_f) = \sum_{x_i} \phi_i(x_i)\phi_{i,f}(x_i, y_f) \prod_{f' \in N(i)} m_{f'\to i}(x_i)$$

$$= \phi_i(x_i(y_f)) \prod_{f' \in N(i)\setminus f} m_{f'\to i}(x_i(y_f)) \Big\}$$ one value for each x_i. Just store $m_{i\to f}(x_i)\ \forall x_i$.

$$m_{f\to i}(x_i) = \sum_{y_f} \phi_f(y_f)\phi_{i,f}(x_i, y_f) \prod_{j \in N(f)\setminus i} m_{j\to f}(y_f)$$

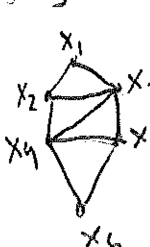
$$= \sum_{\substack{y_f:\\ x_i = \hat{x}_i(y_f)}} \phi_f(y_f) \prod_{j \in N(f)\setminus i} m_{j\to f}(y_f)$$

$$= \sum_{\vec{x}_{f\setminus i}} \phi_f(x_i, \vec{x}_{f\setminus i}) \prod_{j \in f\setminus i} m_{j\to f}(x_j)$$

What about graphs that aren't trees? e.g.,

<u>Junction tree algorithm</u> (also called "clique trees")

Turn graph into <u>tree</u> by <u>triangulating</u> (making chordal) and replacing every maximal clique with a variable



elim. ordering x_1, x_2, x_3 ⟶

$$Y_{123} \overset{x_1}{-} Y_{234} \overset{x_2}{-} Y_{345} \overset{x_3}{-} Y_{456}$$

Elim. ordering found by max-cardinality algorithm.

Again, messages can be computed slightly more efficiently using the structure.