MAC6916 PROBABILISTIC GRAPHICAL MODELS
LECTURE 8: MESSAGE PASSING IN CHORDAL GRAPHS

DENIS D. MAUÁ

1. Introduction

Chordal graphs are the intersection of the classes of distributions represented by d-separation and u-separation. They are also essential in understanding the complexity of variable elimination. Trees are one of the simplest family of chordal graphs, and one that originates the sum-product algorithm. As we will see in this lecture, chordal graphs generalize trees in many ways. Importantly, (maximal) chordal graphs can be defined recursively; chordal graphs also *decompose*: there are sets of nodes which if removed turn the graph into two chordal graphs. These properties allow a very peculiar parametrization of chordal graphs, one that can be used to compute exact inference. Since any graph can be completed to become a chordal graph (e.g., by graphical variable elimination), the algorithm for chordal graphs is generic (i.e., it computes inference in any Bayesian or Markov network).

The *clique-tree* algorithm, which operates in chordal graphs, is most useful when solving a generalization of the sum-product problem, which is called the *all-marginals problem*: Given potentials $\phi_1, \dots, \phi_m$ over a set of variables $\mathcal{V}$, compute

$$p(X) = \frac{1}{Z} \sum_{\mathcal{V}-\{X\}} \prod_j \phi_j \, , \qquad \text{[all-marginals problem]}$$

for every $X \in \mathcal{V}$. Clearly, we can solve the all-marginals problem by solving n instances of the sum-product problem. However, as we have seen with the sum-product algorithm in trees, we can save computations by passing messages in parallel. The clique tree performs sequential message passing in a tree-like structure.

Clique-tree algorithms first appeared as methods to compute marginal probabilities in Bayesian networks [1, 6] (although many ideas had been developed in database theory earlier [3]). Motivated by the fact that computations in tree-shaped graphical models are efficient and straightforward, clique-tree algorithms re-cast any graphical model as a tree. The efficiency of the computations in a clique tree is a function of the resemblance of the original graph to a tree as measured by its treewidth.

2. Markov Trees

We begin by showing an interesting property satisfied by Markov trees.

Date: April 12, 2016.

Theorem 1. *Let (G, p) be a Markov tree where p factorizes as*

$$p(\mathcal{V}) = \frac{1}{Z} \prod_{X-Y \in \mathcal{E}} \phi(X, Y) ,$$

and $G = (\mathcal{V}, \mathcal{E})$. Then,

$$p(\mathcal{V}) = \prod_{X \in \mathcal{V}} p(X) \prod_{X-Y \in \mathcal{E}} \frac{p(X,Y)}{p(X)p(Y)} = \frac{\prod_{X-Y} p(X,Y)}{\prod_X [p(X)]^{deg(X)-1}} , \tag{1}$$

where $\mathcal{V}$ is the set of nodes/variables and $\mathcal{E}$ is the set of edges of G.

Proof. Root the tree at an arbitrary node X_1 and let $X_1, \ldots, X_n$ be a topological ordering. In a tree, every node is connected to its non-descedants only through its parent. Thus,

$$p(\mathcal{V}) = \prod_i p(X_i | X_1, \ldots, X_{i-1}) = \prod_i p(X_i | \mathrm{pa}(X_i)) = \prod_i \frac{p(X_i, \mathrm{pa}(X_i))}{p(\mathrm{pa}(X_i))} .$$

Note that in a directed tree we have that $\mathrm{pa}(X_i) = X_j$ for $i = 2, \ldots, n$ and $\mathrm{pa}(X_1) = \emptyset$. Moreover, every relation $(\mathrm{pa}(X_i), X_i)$ is an arc $X_j \to X_i$ in the tree. Therefore,

$$\begin{aligned} p(\mathcal{V}) &= p(X_1) \prod_{X_i \to X_j} \frac{p(X_i, X_j)}{p(X_i)} \\ &= p(X_1) \prod_{X_i \to X_j} \frac{p(X_i, X_j)}{p(X_i)p(X_j)} p(X_j) \\ &= \prod_i p(X_i) \prod_{X_i \to X_j} \frac{p(X_i, X_j)}{p(X_i)p(X_j)} , \end{aligned}$$

where in the last step we used the fact that each internal node has in-degree 1. The result follows by dropping the direction of the arcs. □

According to the previous result, any Markov tree can be characterized in terms of its edge marginals $p(X, Y)$ (since the node marginals $p(X) = \sum_Y p(X, Y)$ are themselves characterized in terms of edge marginals). Thus, if p and q are distributions over $\mathcal{V}$ such that $p(X, Y) = q(X, Y)$ for each edge $X - Y$, then $p(\mathcal{V}) = q(\mathcal{V})$.

Another consequence of Theorem 1 is that any Markov tree can be easily converted into an equivalent tree-shaped Bayesian network. This fact was actually used to prove the correctness of the result. Hence, Markov trees and tree-shaped Bayesian networks are simply different characterizations of otherwise identical models.

We can see the sum-product algorithm in a tree as a way of reparametrizing the model as the product of marginals. We say that the tree is *calibrated* when $p(X_i) \propto \prod_{X_j \in \mathrm{ne}(X_i)} \psi^{(\rho)}_{j \to i}(X_i)$ at every node. Our goal in this lecture is to extend this result and terminology to chordal graphs.

calibrated

3. Chordal Graphs

Recall that an undirected graph is chordal only if every cycle longer than 3 has a chord, that is, a subcycle of size 3. A directed graph is chordal only if its moral graph is chordal. An equivalent notion is that of a decomposable graph.

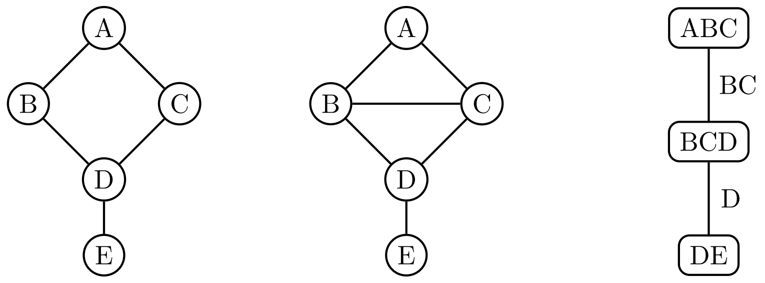

FIGURE 1. Non-chordal undirected graph (left), chordalization (center) and a tree decomposition (right).

Definition 1. A graph G is decomposable if it is either complete or its nodes can be partitioned into sets $\mathcal{V} = \mathcal{A} \cup \mathcal{B} \cup \mathcal{C}$ such that

(i) $\mathcal{A}$ and $\mathcal{C}$ are non-empty and $\mathcal{B}$ is a clique;
(ii) $\mathcal{B}$ separates $\mathcal{A}$ and $\mathcal{C}$;
(iii) $\mathcal{A} \cup \mathcal{B}$ and $\mathcal{B} \cup \mathcal{C}$ induce decomposable graphs.

It is easy to see that trees are decomposable: let $X - Y$ be any edge, and define $\mathcal{A}$ be the set of all nodes that are connected to X by a path that does not contain Y, and similarly, let $\mathcal{C}$ be the set of all nodes that are connected to Y by a path that does not contain X. Then $\mathcal{A} \cup \{X, Y\} \cup \mathcal{B}$ is a decomposition of the graph since $G[\mathcal{A} \cup \{X, Y\}]$ and $G[\mathcal{C} \cup \{X, Y\}]$ are both trees, hence decomposable, and $\{X, Y\}$ separates $\mathcal{A}$ and $\mathcal{C}$).

Definition 2. A *tree decomposition* of an undirected graph $G = (\mathcal{V}, \mathcal{E})$ is a tree $T = (\mathcal{V}_T, \mathcal{E}_T)$ such that tree decomposition

(i) $\mathcal{V}_T$ is a set of subsets of $\mathcal{V}$;
(ii) For every clique $\mathcal{C}$ of G there is a clique node $\mathcal{C}_j \in \mathcal{V}_T$ such that $\mathcal{C} \subseteq \mathcal{C}_j$;
(iii) Every node $\mathcal{C}_k$ in a path between clique nodes $\mathcal{C}_i$ and $\mathcal{C}_j$ in $\mathcal{V}_T$ contains $\mathcal{C}_i \cap \mathcal{C}_j$.

The *width* of a tree decomposition is the size of the largest set $\mathcal{C} \in \mathcal{V}_T$ node minus one. width

Property (ii) ensures that the scopes of each potential appears in a clique node of a tree decomposition of its domain graph; it also ensures that every maximal clique of G is a node in T (but some nodes might be non-maximal cliques). Property (iii) ensures that a node in a clique $\mathcal{C}_j$ in any path of T either appears for the last time (in that path) or appears also in the next clique node. We often label the edges $\mathcal{C}_i - \mathcal{C}_j$ with the respective *separator set* $\mathcal{C}_i \cap \mathcal{C}_j$ (this name will be made clearer later). separator set

We will often be interested in a special case of tree decompositions.

Definition 3. A *clique tree* is a tree-decomposition of G where every node $\mathcal{C}_j$ of $\mathcal{V}_T$ is a clique in G. clique tree

The following result relates chordal graphs, decomposable graphs, elimination sequences, and clique trees.

Theorem 2. *The following statements are equivalent.*

(i) G is chordal.

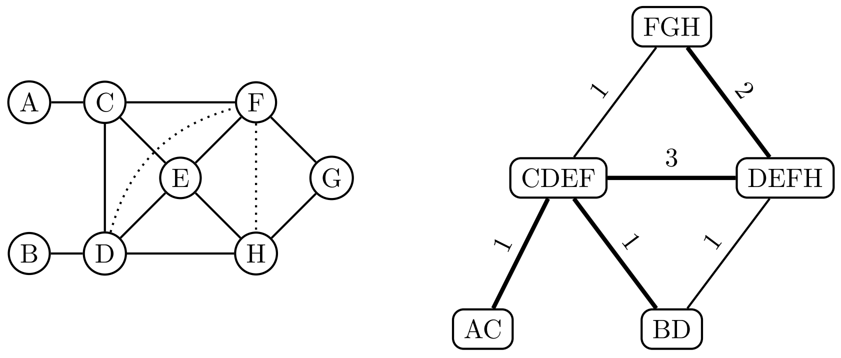

Figure 2. Constructing a tree-decomposition.

(ii) G is decomposable.
(iii) G admits a perfect elimination ordering.
(iv) G has a clique tree.
(v) There is an orientation of the edges of G that induces a directed acyclic graph whose moral graph is G.

As a consequence of the previous result, the treewidth of the a chordal graph is the minimum width of a tree decomposition of it.

Theorem 3. *Let (G, p) be a Markov or Bayesian network whose graph G is chordal, and $T = (\mathcal{V}_T, \mathcal{E}_T)$ be a clique tree for G. Then*

$$p(\mathcal{V}) = \frac{\prod_{\mathcal{C} \in \mathcal{V}_T} p(\mathcal{C})}{\prod_{\mathcal{S} \in \mathcal{E}_T} p(\mathcal{S})} .$$

Proof. Root the clique tree T at a node $\mathcal{R}$ and let $\mathcal{C}_1, \ldots, \mathcal{C}_m$ be a topological ordering of the cliques. Then,

$$p(\mathcal{V}) = \prod_{j=1}^{m} p(\mathcal{C}_j | \mathcal{C}_1, \ldots, \mathcal{C}_{j-1}) = \prod_{j=1}^{m} p(\mathcal{C}_j | \mathcal{C}_j \cap \mathrm{pa}(\mathcal{C}_j)) = \frac{\prod_{\mathcal{C} \in \mathcal{V}_T} p(\mathcal{C})}{\prod_{\mathcal{S} \in \mathcal{E}_T} p(\mathcal{S})} ,$$

where the second equality follows from the decomposability of the graph using $\mathcal{B} = \mathrm{pa}(\mathcal{C}_j) \cap \mathcal{C}_j$, $\mathcal{A}$ the nodes that are connected to $\mathcal{C}_j$ without passing through $\mathrm{pa}(\mathcal{C}_j)$ and $\mathcal{C}$ the nodes that are connected to $\mathcal{C}_j$ without passing through $\mathrm{pa}(\mathcal{C}_j)$. □

4. Clique Tree Propagation

Constructing a clique tree (MST method):

(1) chordalize graph (e.g. by graphical VE)
(2) find (maximal) cliques (e.g. as greater neighbors of a variable according to a perfect elimination ordering)
(3) build clique graph (with edges weighted by the cardinality of their separating sets)
(4) find maximum-weight spanning tree

The width of the clique tree produced is the induced width of the elimination ordering used. Finding a minimal width clique tree equals finding a perfect elimination ordering (and it is thus NP-hard).

minimal clique tree

A clique tree is *minimal* if no clique node is a proper subset of other node (i.e., if a node is not a maximal clique). Any clique tree can be transformed into minimal form by deleting non-maximal cliques. This can also be achieved when building the clique tree using the procedure described (at each step we check whether a clique is contained in some already built clique).

4.1. **Message Passing Algorithm for Clique Trees.** Take a set of potentials $\phi_1, \dots, \phi_m$ and a tree decomposition $T = (\mathcal{V}_T, \mathcal{E}_T)$ such that $\mathrm{sc}(\phi_k) \in \mathcal{C}_j \in \mathcal{V}_T$ for $k = 1, \dots, m$ (we say that such a tree *covers* the potentials).

4.1.1. *Initialization.* Associate every potential ϕ_j with a clique $\mathcal{C}_k$ such that $\mathrm{sc}(\phi_j) \subseteq \mathcal{C}_k$. Note that there might be cliques with multiple potentials assigned and cliques with no potential assigned. Call ψ_j the combination of all potentials associated to clique $\mathcal{C}_j$; if there is no potential associated, set $\psi_j = 1$.

4.1.2. *Message passing.* We distinguish arcs $\mathcal{C}_i \to \mathcal{C}_j$ into two classes:

Empty: when $\mathcal{C}_i$ has *not* yet sent a message $\psi_{i \to j}$ to $\mathcal{C}_j$.
Full: when $\mathcal{C}_i$ has *already* sent a message $\psi_{i \to j}$ to $\mathcal{C}_j$.

The empty arcs $\mathcal{C}_i \to \mathcal{C}_j$ can be further divided into

Ready: When *all* adjacent arcs $\mathcal{C}_k \to \mathcal{C}_i$ other than $\mathcal{C}_j \to \mathcal{C}_i$ are full.
Not ready: When at least two adjacent arcs are empty.

Consider an arc $\mathcal{C}_i \to \mathcal{C}_j$ and let $\mathcal{C}_{j_1}, \dots, \mathcal{C}_{j_r}$ be the neighbors of $\mathcal{C}_i$ other than $\mathcal{C}_j$. Message passing in tree decompositions consists in recursively selecting a ready arc $\mathcal{C}_i \to \mathcal{C}_j$ and computing the message

$$\psi_{i \to j} = (\psi_i \times \psi_{j_1 \to i} \times \cdots \times \psi_{j_r \to i}) - (\mathcal{C}_i \setminus \mathcal{C}_j)\,.$$

Initially, only the leaves have ready arcs. The algorithms terminates when every arc is full, and not more messages can be send. With an appropriate scheduling of the messages, this algorithm can be seen as first sending messages toward a designated root clique node $\mathcal{C}_r$, then sending messages away from that node towards the leaves. These two phases are called the upward and downward passes. In practice, the root node is chosen automatically by any scheduling.

4.1.3. *Termination.* We say that the tree is *calibrated* when every arc is full. The process of sending messages until every arc is full is called *calibration*. The clique and separator marginals can be obtained from the messages once the tree is calibrated. Let $\mathcal{C}_{i_1}, \dots, \mathcal{C}_{i_r}$ be (all) the neighbors of a clique $\mathcal{C}_j$. Then calibration

$$p(\mathcal{C}_j) \propto \psi_j \times \psi_{i_1 \to j} \times \cdots \times \psi_{i_r \to j}$$
$$p(\mathcal{C}_i \cap \mathcal{C}_j) \propto \psi_{i \to j} \times \psi_{i \to i}$$

Note that the marginal of a variable $p(X)$ can be obtained from any clique or separator containing X.

4.1.4. *Soundness.* Let $\mathcal{C}_i$ be any node and $j = \mathrm{pa}(i)$. Define $\mathcal{D}_{i \to j} = \bigcup_{k \in \mathrm{de}(i)} \mathcal{C}_k \setminus \mathcal{C}_j$ to be the variables that appear in the subtree rooted at i but not at subtree rooted at j.

Theorem 4 (Upward Pass Invariant)**.** *The messages sent in the upward pass satisfy:*

$$\psi_{i \to j} = \left(\prod_{k \in de(i) \cup \{i\}} \psi_k \right) - \mathcal{D}_{i \to j}\,.$$

Proof Sketch. Say that a node of the clique tree is inactive if it has not sent message, otherwise it is active. Thus at some iteration, the active nodes are $\mathrm{de}(i)$ and the remaining ones are inactive. Recall that by the running intersection property of tree decompositions (Property (iii)), any variable in $\mathcal{C}_i \setminus \mathcal{C}_j$ is not in any further clique in the path from $\mathcal{C}_j$ to the root. Hence, all potentials assigned to inactive cliques do not have the eliminated variables in their scope. This suffices to show by an inductive argument in the height of the subtree that the result holds. □

Corollary 1. *Let r be the root of the tree, and $i_1, \ldots, i_s$ be its children. Then,*

$$p(\mathcal{C}_r) \propto \psi_r \times \psi_{i_1 \to r} \times \cdots \times \psi_{i_s \to r} \,.$$

Proof. By the previous theorem we have that

$$\psi_{i_k \to r} = \left(\prod_{k \in \mathrm{de}(i_k) \cup \{i_k\}} \psi_k \right) - \mathcal{D}_{i_k \to r} \,.$$

Note that by the running intersection property, the sets $\mathcal{D}_{i_k \to k}$ form a disjoint union of $\mathcal{V} \setminus \mathcal{C}_r$. Hence,

$$\begin{aligned}
&\psi_r \times \psi_{i_1 \to r} \times \cdots \times \psi_{i_s \to r} = \\
&\quad \psi_r \times \prod_k \Big(\Big(\prod_{k \in \mathrm{de}(i_k) \cup \{i_k\}} \psi_k \Big) - \mathcal{D}_{i_k \to r} \Big) \\
&\quad \Big(\psi_r \times \prod_{k \in \mathrm{de}(i_k) \cup \{i_k\}} \psi_k \Big) - \Big(\bigcup_k \mathcal{D}_{i_k \to r} \Big) \\
&\quad \propto p(\mathcal{C}_r) \,,
\end{aligned}$$

where we used the fact that $\mathcal{D}_{i_k \to k}$ are disjoint and do not intersect with $\mathcal{C}_r$ to distribute the combinations. □

Sanity checking:

$$\begin{aligned}
p(\mathcal{V}) &= \frac{\prod_{j \in \mathcal{V}_T} p(\mathcal{C}_j)}{\prod_{i - j \in \mathcal{E}_T} p(\mathcal{C}_i \cap \mathcal{C}_j)} &&\propto \frac{\prod_{\mathcal{C}_j \in \mathcal{V}_T} \psi_j \times \psi_{i_1 \to j} \times \cdots \times \psi_{i_r \to j}}{\prod_{i - j \in \mathcal{E}_T} \psi_{i \to j} \times \psi_{j \to i}} \\
&= \prod_{j \in \mathcal{V}_T} \left(\psi_j \times \frac{\prod_{i \in \mathrm{ne}(j)} \psi_{i \to j}}{\prod_{i \in \mathrm{ne}(j)} \psi_{i \to j}} \right) &&= \prod_{k=1}^m \phi_k \,,
\end{aligned}$$

where we have notated the constraints $\mathcal{C}_j \in \mathcal{V}_j$ as $j \in \mathcal{V}_T$ and $\mathcal{C}_i - \mathcal{C}_j \in \mathcal{E}_T$ as $i - j \in \mathcal{E}_T$.

Note that in a calibrate tree we have that

$$p(\mathcal{C}_i \cap \mathcal{C}_j) = p(\mathcal{C}_i) - (\mathcal{C}_i \setminus \mathcal{C}_j) = p(\mathcal{C}_j) - (\mathcal{C}_j \setminus \mathcal{C}_i) \,.$$

An alternative method of passing messages is by fixed point iteration: start with $p(\mathcal{C}_i) \propto \psi_i$ and $p(\mathcal{C}_i \cap \mathcal{C}_j) = 1$ and pass messages using the equation above until it is satisfied. This equivalent to passing messages with division, and has the same asymptotic properties of the message passing algorithm discussed.

4.1.5. *Complexity.* The cost of passing messages upward and downward (i.e., towards and away from a designated root node) is linear in the clique tree size, which is at least exponential in the treewidth of the graph. Hence, clique tree propagation is asymptotically similar to variable elimination.

5. Clique-Tree and Variable Elimination

Message passing in clique-trees is very similar to VE. The clique tree defines a perfect elimination ordering according to which variables are eliminated. Compared to VE, message passing has the following advantages:

- All marginals are computed with two passes in the tree, saving computations (compared to multiple runs of VE);
- Multiple variables are eliminated from a single combined potential when sending a message (in VE, this would result in more computations);
- Evidence can be inserted and retracted easily.

The downsides of clique-tree are:

- small overhead for single marginal computation (compared to VE), and
- the fact that it does not exploit the additional independences and irrelevances (barren nodes) introduced by inserting evidence.

The latter can be partly fixed by lazy propagation [7].

6. Reading

Read Jordan's tutorial on graphical models [8], and Box 10.A of Koller and Friedman [5, page 358].

7. Exercises

No exercises this week.

8. Assignment

No assignments this week.

References

[1] S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society (Series B), pp. 157–225, 1988.

[2] P.P. Shenoy and G. Shafer. Axioms for probability and belief-function propagation. In *Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 169–198, 1988.

[3] C. Beeri, R. Fagin, D. Maier and M. Yannakakis. On the desirability of acyclic database schemes. Journal of the ACM, volume 30, issue 3, pp. 479–513, 1983.

[4] A. Darwiche. Modeling and reasoning with Bayesian networks. Cambridge University Press, 2009.

[5] D. Koller and N. Friedman. Probabilistic Graphical Models MIT Press, 2009.

[6] P. Shenoy and G. Shafer. Axioms for probability and belief-function propagation. In *Proceedings of the 4th Conference on Uncertainty in Artificial Intelligence*, pp. 169–198, 1988.

[7] A.L. Madsen and F.V. Jensen. Lazy propagation: A junction tree inference algorithm based on lazy evaluation. Artificial Intelligence, volume 113, issues 12, pp. 203–245, 1999.

[8] M.I. Jordan. Graphical Models. Statistical Science, volume 19, issue 1, pp. 140–155, 2004.