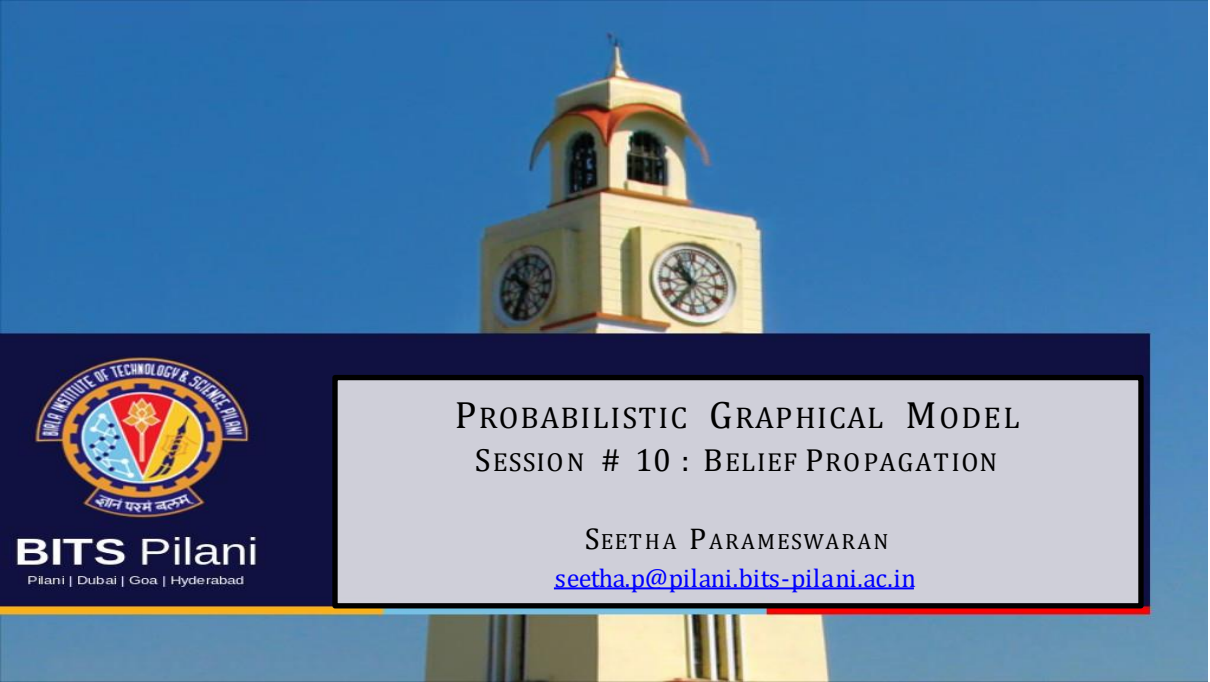


BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Probabilistic Graphical Model

Session # 10 : Belief Propagation

Seetha Parameswaran
seetha.p@pilani.bits-pilani.ac.in

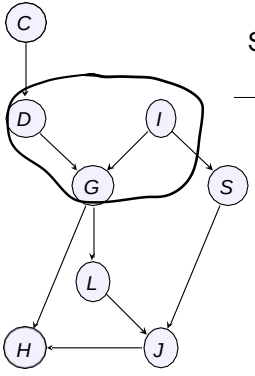
Table of Contents

VARIABLE ELIMINATION ALGORITHM

Variable elimination algorithm is the manipulation of the factors.

1. Create a factor ψ_i by multiplying existing factors.
2. Eliminate a variable in ψ_i to generate a new factor τ_i.
3. τ_i is then used to create another factor.

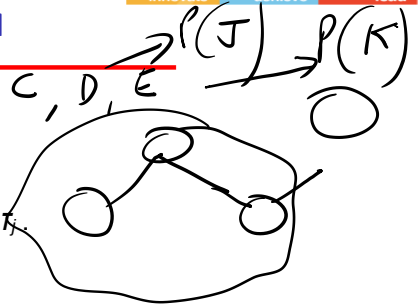
VARIABLE ELIMINATION IN BN



Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\varphi_C(C) \cdot \varphi_D(C,D)$	C, D	$\tau_1(D)$
2	D	$\tau_D(D) \cdot \varphi_G(G,I,D)$	G, I, D	$\tau_2(G,I)$
3	I	$\varphi_I(I) \cdot \varphi_S(S,I) \cdot \tau_2(G,I)$	G ,S, I	$\tau_3(G,S)$
4	H	$\varphi_H(H,G,J)$	H, G, J	$\tau_4(G,J)$
5	G	$\tau_3(G,S) \cdot \tau_4(G,J) \cdot \varphi_L(L,G)$	G, L, S, J	$\tau_5(J,L,S)$
6	S	$\varphi_J(J,L,S) \cdot \tau_5(L,S)$	J, L, S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J, L	$\tau_7(J)$

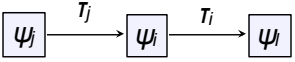
Table of Contents

VARIABLE ELIMINATION ALGORITHM



Different Perspective

1. Factor ψ_i is a data structure, which takes messages τ_j.
2. τ_j is generated by the other factor ψ_j.
3. Factor ψ_i generates message τ_i.
4. Message τ_i is used by another factor τ_l.



CLUSTER GRAPH

DEFINITION

A cluster graph $\mathcal{U}$ for a set of factors over $\mathcal{X}$

- an undirected graph
- nodes i are clusters $C_i \subseteq \mathcal{X}$
- edges between C_i and C_j associated with sepset $S_{i,j} \subseteq C_i \cap C_j$.

Family Preserving Property

- Each factor $\varphi \in \Phi$, the distribution, must be associated with a cluster C_i such that

$$Scope[\varphi] \subseteq C_i$$

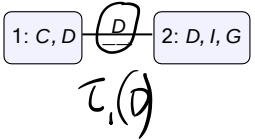
GENERATE CLUSTER GRAPH

- An execution of variable elimination defines a cluster graph.
- Mark a cluster for each factor ψ_i used in the computation, which is associated with the set of variables $C_i = Scope[\psi_i]$.
- Draw an edge between two clusters C_i and C_j if the message τ_i is produced by eliminating a variable in ψ_i is used in the computation of τ_j.

VARIABLE ELIMINATION AND CLUSTER GRAPH

Step	VE	New factor	Message
1	C	$\psi_1(D)$	$\tau_1(D)$
2	D	$\psi_2(G, I)$	$\tau_2(G, I)$
3	I	$\psi_3(G, S)$	$\tau_3(G, S)$
4	H	$\psi_4(G, J)$	$\tau_4(G, J)$
5	G	$\psi_5(J, L, S)$	$\tau_5(J, L, S)$
6	S	$\psi_6(J, L)$	$\tau_6(J, L)$
7	L	$\psi_7(J)$	$\tau_7(J)$

- $C_1 = \psi_1 = \varphi_C(C) \cdot \varphi_D(C, D)$
- $C_2 = \psi_2 = \psi_1 \cdot \varphi_G(G, I, D)$
- C_1 generates τ_1.
- τ_1 is used for computing ψ_2.
- Hence an edge between C_1 and C_2.



Variable Elimination and Cluster Graph

Step	VE	New factor	Message
1	C	$\psi_1(D)$	$\tau_1(D)$
2	D	$\psi_2(G, I)$	$\tau_2(G, I)$
3	I	$\psi_3(G, S)$	$\tau_3(G, S)$
4	H	$\psi_4(G, J)$	$\tau_4(G, J)$
5	G	$\psi_5(J, L, S)$	$\tau_5(J, L, S)$
6	S	$\psi_6(J, L)$	$\tau_6(J, L)$
7	L	$\psi_7(J)$	$\tau_7(J)$

Each of the factors in the initial set of factors Φ is also associated with a cluster C_i.

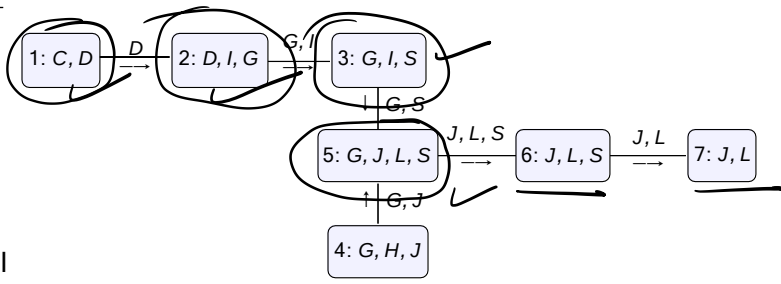


Table of Contents

Clique Tree

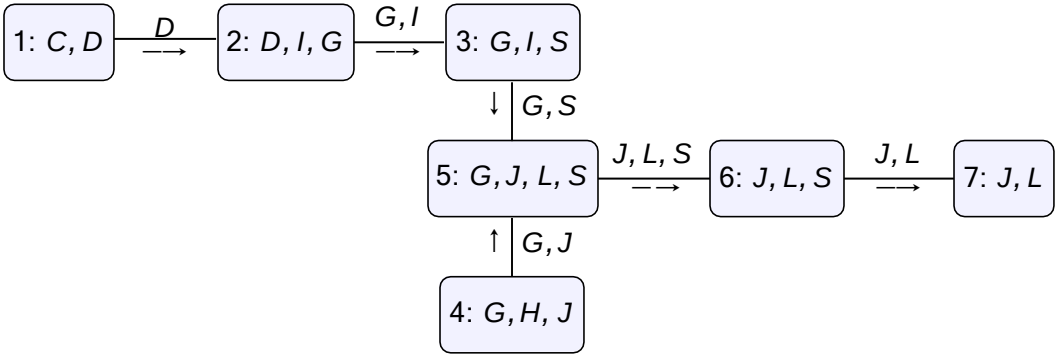
Definition

A clique tree $\mathcal{T}$ for a set of factors over $\mathcal{X}$

- an undirected tree
- nodes i are clusters $C_i \subseteq \mathcal{X}$
- edges between C_i and C_j associated with sepset $S_{i,j} = C_i \cap C_j$.

- also called Junction Tree or Join Tree.
- The clusters are called **cliques**.
- Used to draw inferences.

Clique Tree Example



Cliques C_6 and C_7 are nonmaximal or redundant. Hence can be removed.

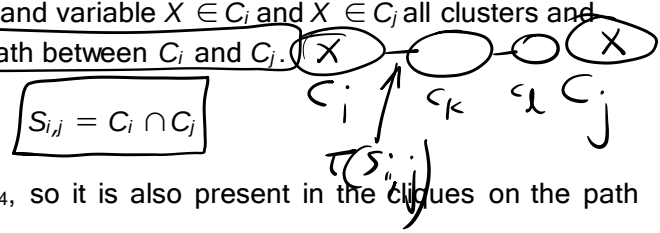
Properties of Clique Tree

1. Family Preserving Property
   - Each factor $\varphi \in \Phi$ must be associated with a cluster C_i such that

$$Scope[\varphi] \subseteq C_i$$

2. Running Intersection Property
   - For each pair of clusters C_i and C_j and variable $X \in C_i$ and $X \in C_j$ all clusters and sepsets contain X in the unique path between C_i and C_j.

$$S_{i,j} = C_i \cap C_j$$



Example: G is present in C_2 and in C_4, so it is also present in the cliques on the path between them: C_3 and C_5.

Clique Tree Theorem 1

Theorem

Let T *be a cluster tree induced by a variable elimination algorithm over a set of factors* Φ*. Then* T *satisfies the running intersection property.*

Proof

Let C and C' be two clusters that contain X

Let C_X be the cluster at which X is finally eliminated

We shall show

(1) all clusters along the path from C to C_X contain X

(2) all clusters along the path from C' to C_X contain X

Proof

Since we are dealing with a cluster tree, the path between C and C_X containing C' or C_X falls on the path from C to C' so we either have

(1) C - C' - C_X or (2) C - C_X - C'

In (1), since we show C - C_X has X in all nodes in between, there must be X in all nodes between C and C'.

Proof

In (2) $C - C_X$ has X in all intermediate nodes and so does $C' - C_X$. Therefore $C - C'$ has X in all intermediate nodes.

∴ We must prove that $C - C_X$ has X in all intermediate nodes

Proof

The computation at C_X happens after C

- When X is eliminated at C_X all factors involving X are multiplied into a big factor and X is summed out
- N - cluster node beyond C_X will contain X
- X is not eliminated in C, so the message computed at C must contain X and goes upstream

Proof

Since X is not eliminated on the path from C to C_X it is present on all nodes between C and C_X

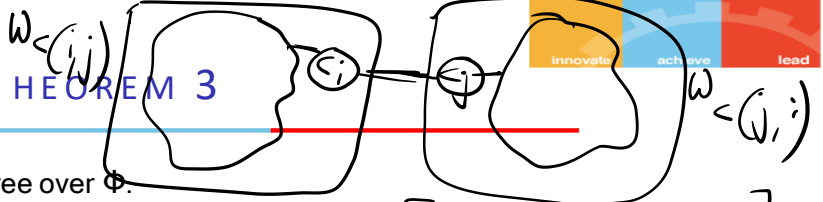
Clique Tree Theorem 2

Theorem

Let T be a cluster tree induced by a variable elimination algorithm over a set of factors Φ. Let C_i and C_j be two neighbouring clusters, such that C_i passes the message τ_i to C_j. Then the scope of the message τ_i is precisely $C_i \cap C_j$.

$$Scope[\tau_i] = S_{i,j} = C_i \cap C_j$$

Clique Tree Theorem 3



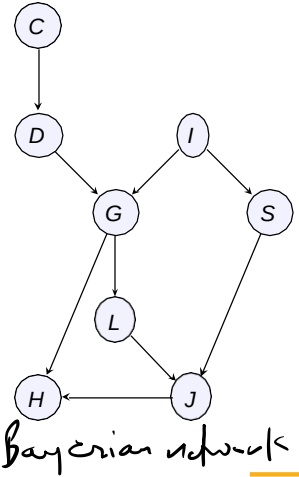
- Let T be a cluster tree over Φ.
- Let H_Φ be the undirected graph associated with Φ.
- $W_{<(i,j)}$ = all variables that appear only on C_i side of T.
- $W_{<(j,i)}$ = all variables that appear only on C_j side of T.
- $S_{(i,j)}$ = all variables that appear on both sides.

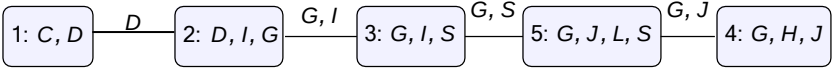
Theorem

T *satisfies the running intersection property, if and only if,*
for every sepset $S_{i,j}$, *the nodes* $W_{<(i,j)}$ *and* $W_{<(j,i)}$ *are seperated in* H_Φ *given* $S_{i,j}$.

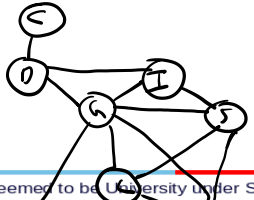
$$P_\Phi \models (W_{<(i,j)} \perp W_{<(j,i)} | S_{(i,j)})$$

Clique Tree Independence





$$P_\Phi \models (\{C, I, D\} \perp \{J, L, H\} | \{G, S\})$$

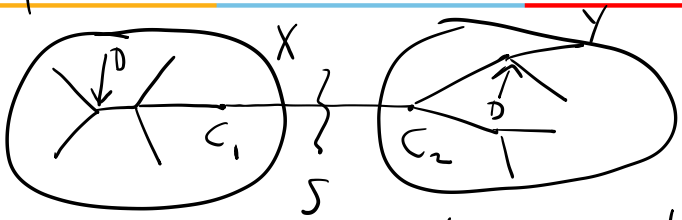


Proof

(1) Running intersection property in clique tree
$\Rightarrow$ original Markov network can be separated into conditionally independent pieces

Let S be a clique separator in clique tree T such that variables X are on one side of the separator and variables Y are on the other side
Show that $Sep_T(X; Y | S)$

Proof



Lemma: All nodes that are both in X and Y are in S

Proof of lemma: In digraph T_1 let S separate digraphs C_1 and C_2. Consider a node D that is in

Proof

both X and Y.

D is in some clique C_α in the left hand side of the separator and in some clique C_β on the right hand side of the separator

Then exists a path from C_α to C_β in T through C_1 and C_2

Proof

Since the running intersection property holds true

D is in both C_1 and C_2 and also in $S = C_1 \cap C_2$

Now, show that $sep_I(X; Y | S)$

We will prove this result by contradiction

Suppose S does not separate X and Y

Proof

There exists a node A in X, a node B in Y such that a path π in H_f between A and B does not pass through S

This implies that each node in π is either in X or in Y but not in both X and Y since by the lemma above such a node would be in S!

Proof

The path Π must pass through some node D in X to some node E in Y which means DE is an edge in H_ϕ. So DE forms a clique in H_ϕ which must be part of some maximal clique in T. But we know that all cliques in T are subsets of X or Y, so there cannot exist a clique containing D, E.

Proof

Reverse Direction

separability in H_ϕ $\Rightarrow$ running intersection property in T

Also by contradiction

assume separability in H_ϕ and that running intersection property in T does not hold and arrive at a contradiction

Proof

Assume S is a separating set in H_ϕ. T is a cluster tree corresponding to H_ϕ when the running intersection property does not hold

Let $A \in C_1$ and $A \in C_k$ but $A \notin C_2$

Proof in H_t

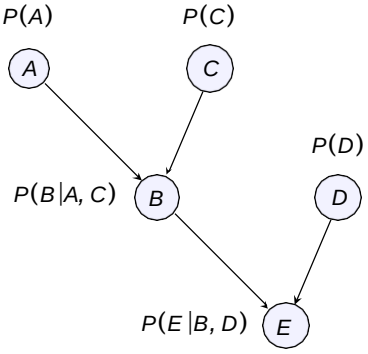
There exists a path α between a vertex in C_i and a vertex in C_k through A where this path does not go through S

$\therefore$ S is not a separator which is a contradiction

Construct a Clique Tree

1. Triangulate the graph G over factor Φ to create a chordal graph H_Φ.
2. Find the maximal cliques in H_Φ and assign them as nodes to an undirected graph.
3. Assign weights to the edges between two nodes of the undirected graph as the numbers of elements in the sepset of the two nodes.
4. Construct the clique tree using the maximum spanning tree algorithm.
5. Compute the cluster potential for each cluster as the product of factors associated with the nodes present in the cluster.

Construct a Clique Tree – Example



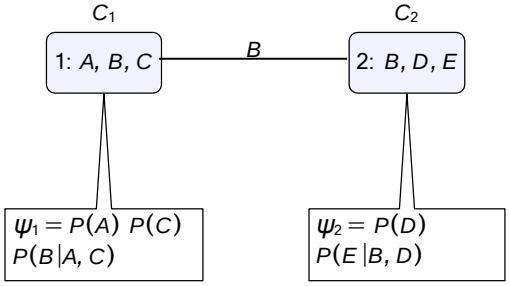
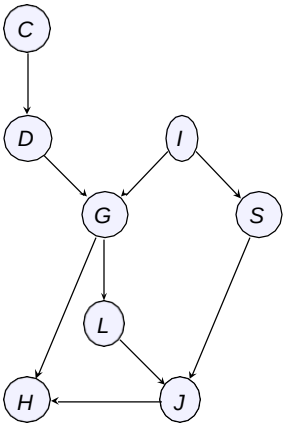


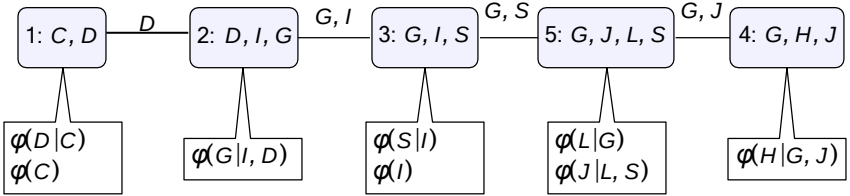
Table of Contents

Clique Tree for a Bayesian Network



$$\Phi = \varphi(C) \cdot \varphi(D|C) \cdot \varphi(I) \cdot \varphi(G|D, I) \cdot \varphi(S|I) \cdot \varphi(L|G) \cdot \varphi(J|L, S) \cdot \varphi(H|G, J)$$

Construct the clique tree T over Φ.



Clique Tree Message Passing

- Compute the **initial potentials** associated with each clique.

$$\psi_1(C,D) = \varphi_C(C) \cdot \varphi_D(C,D)$$
$$\psi_2(G,I,D) = \varphi_G(G,I,D)$$
$$\psi_3(S,I) = \varphi_I(I) \cdot \varphi_S(S,I)$$
$$\psi_4(H,G,J) = \varphi_H(H,G,J)$$
$$\psi_5(J,L,G,S) = \varphi_L(L,G) \cdot \varphi_J(J,L,S)$$

Clique Tree Message Passing

Compute the probability $P(J)$.

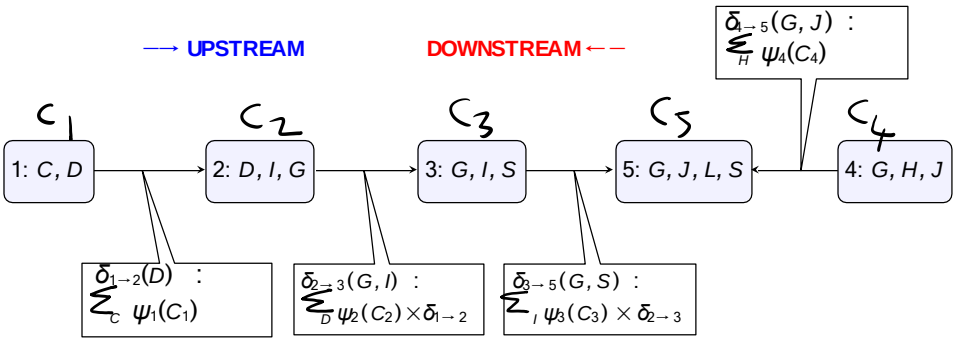
- Select root clique as a clique that contains J. $root = C_5$
- In C_1:
  - Eliminate C by performing $\sum_C \psi_1(C, D)$.
  - The resulting factor $\delta_{1\to2}(D)$ is send as a message to C_2.
- In C_2:
  - Define $\beta_2(G, I, D) = \delta_{1\to2}(D) \cdot \psi_2(G, I, D)$.
  - Eliminate D by performing $\sum_D \beta_2(G, I, D)$.
  - The resulting factor $\delta_{2\to3}(G, I)$ is send as a message to C_3.
- In C_3:
  - Define $\beta_3(G, I, S) = \delta_{2\to3}(G, I) \cdot \psi_3(G, I, S)$.
  - Eliminate I by performing $\sum_I \beta_3(G, I, S)$.
  - The resulting factor $\delta_{3\to5}(G, S)$ is send as a message to C_5.

Clique Tree Message Passing

- In C_4:
  - Eliminate H by performing $\sum_H \psi_4(H, G, J)$.
  - The resulting factor $\delta_{4\to5}(G, J)$ is send as a message to C_5.
- In C_5:
  - Define $\beta_5(G, J, S, L) = \delta_{3\to5}(G, S) \cdot \delta_{4\to5}(G, J) \cdot \psi_5(G, J, S, L)$.
  - Compute $P(J)$ by summing out G, L, S.

Clique Tree Message Passing

Compute $P(J)$.
$root = C_5$

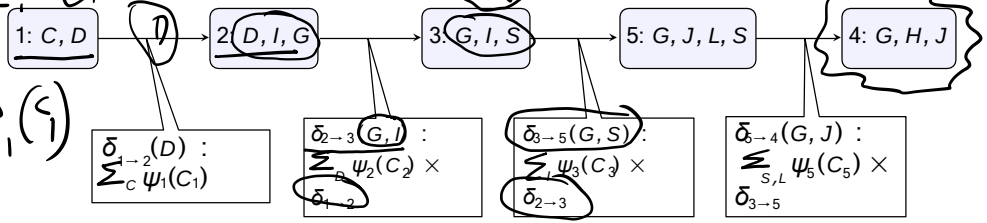


$$\beta_5(G,J,S,L) = \delta_{3\to5}(G,S) \cdot \delta_{4\to5}(G,J) \cdot \psi_5(G,J,S,L)$$

Clique Tree Message Passing

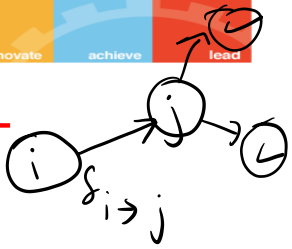
Compute $P(J)$.

$root = C_4$



$$\beta_4(H, G, J) = \delta_{5\to 4}(G, J) \cdot \psi_4(H, G, J)$$

Clique Tree Message Passing

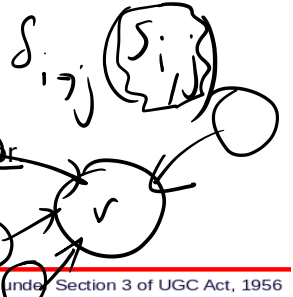


1. Assign each factor φ to some clique $\alpha(\varphi)$.
2. Compute the **initial potentials** associated with each clique.

$$\psi_j(C_j) = \prod_{\varphi:\alpha(\varphi)=j} \varphi$$

3. Except for the root, the message from C_i to another clique C_j is computed using the **sum-product message passing** computation.

$$\delta_{i\to j} = \sum_{C_i - S_{i,j}} \psi_i \cdot \prod_{k\in(Nb_i-\{j\})} \delta_{k\to i}$$



4. At the root, after receiving all messages, compute the **belief** factor.

$$\beta_r(C_r) = \psi_r \cdot \prod_{k\in Nbc_r} \delta_{k\to r}$$

Clique Tree and Variable Elimination

Goal: Compute the marginal probability of any set of query nodes Y which is fully contained in some clique.

1. Select one such clique C_r to be the root.
2. Perform the Clique Tree message passing toward that root.
3. Extract $\tilde{P}_\Phi(Y)$ from the final potential at C_r by summing out the other variables $C_r - Y$.

$$\tilde{P}_\Phi(Y) = \sum_{C_r - Y} \prod_{\phi} \varphi$$

$$\sum_{C_r - Y} \beta_r(C_r)$$

Clique Tree Message Passing Algorithm

Algorithm 10.1 Upward pass of variable elimination in clique tree

Procedure CTree-SP-Upward (
  Φ, // Set of factors
  $\mathcal{T}$, // Clique tree over Φ
  α, // Initial assignment of factors to cliques
  C_r // Some selected root clique
)

1 Initialize-Cliques
2 **while** C_r is not ready
3 Let C_i be a ready clique
4 $\delta_{i \to p_r(i)}(S_{i,p_r(i)}) \leftarrow$ SP-Message$(i, p_r(i))$
5 $\beta_r \leftarrow \psi_r \cdot \prod_{k \in \text{Nb}_{C_r}} \delta_{k \to r}$
6 **return** β_r

Procedure Initialize-Cliques (
)

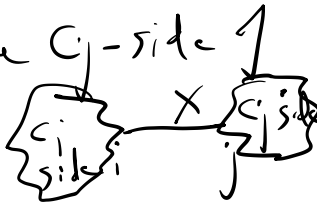
1 **for each** clique C_i
2 $\psi_i(C_i) \leftarrow \prod_{\phi_j : \alpha(\phi_j)=i} \phi_j$
3

Procedure SP-Message (
  i, // sending clique
  j // receiving clique
)

1 $\psi(C_i) \leftarrow \psi_i \cdot \prod_{k \in (\text{Nb}_i - \{j\})} \delta_{k \to i}$
2 $\tau(S_{i,j}) \leftarrow \sum_{C_i - S_{i,j}} \psi(C_i)$
3 **return** $\tau(S_{i,j})$

Correctness of Algorithm 10.1

Proposition: Assume that X is eliminated when a message is passed from C_i to C_j. Then X does not appear anywhere on the C_j-side of the edge $(i-j)$



Proof: Assume the contrary

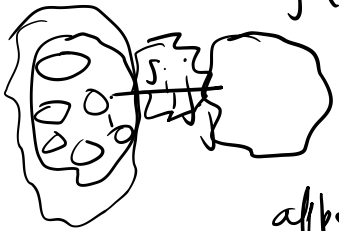
violates running intersection

Correctness of Algorithm 10.1

Theorem: Let $\delta_{i \to j}$ be the message passed from C_i to C_j. Then

$$\delta_{i \to j}(S_{ij}) = \sum_{U_{<(i \to j)}} \prod_{\phi \in F_{<(i \to j)}} \phi$$

set of factors on the inside



set of variables appearing on the inside of $i \to j$ but not in the subset

Correctness of Algorithm 10.1

Proof is by induction. Given formula is trivially true at a leaf

Inductive Step: Let $i_1, i_2, \ldots i_m$ be neighbours of i other than j

$$\sum_{V_{<(i\to j)}} \prod_{\phi \in F_{<(i\to j)}} = \sum_{y_i} \sum_{V_{<(i_1\to i)}} \sum_{V_{<(i_2\to i)}} \cdots \sum_{V_{<(i_m\to i)}} \prod_{\phi \in F_{<(i\to i)}} \phi \prod_{\phi \in F_{<(i_1\to i)}} \phi \prod_{\phi \in F_{<(i_2\to i)}} \phi \cdots \prod_{\phi \in F_{<(i_m\to i)}} \phi$$

Correctness of Algorithm 10.1

This sum-product can be rearranged as

$$\sum_{y_i} \left(\prod_{\phi \in F_i} \phi \right) \sum_{v < (i_1 \to i)} \left(\prod_{\phi \in F_{<(i_1 \to i)}} \phi \right) \cdots \sum_{v < (i_m \to i)} \left(\prod_{\phi \in F_{<(i_m \to i)}} \phi \right)$$

$$= \sum_{y_i} \psi \cdot \delta_{i_1 \to i} \, \delta_{i_2 \to i} \cdots \delta_{i_m \to i}$$

which is what we understand by $\delta_{i \to j}$

Correctness of Algorithm 10.1

Let us use the formula to compute β_r at the root

$$\beta_r \leftarrow \rho_r \cdot \prod_{k \in NbG_r} \delta_{k \to r} \quad \text{[from Algo 10.1]}$$

Let $i_1, i_2 \ldots i_m$ be the neighbours to root r

$$\delta_{i \to r} = \sum_{\nu \langle (i \to r)} \prod_{\phi \in F \langle (i \to r)} \phi$$

Correctness of Algorithm 10.1

$$\beta_v = \psi_v \cdot \delta_{i_1 \to v} \delta_{i_2 \to v} \cdots \delta_{i_M \to v}$$

$$\beta_v = \psi_v \cdot \left(\sum_{V \setminus \{i_1 \to v\}} \prod_{\phi \in F_{\setminus (i_1 \to v)}} \phi \right) \left(\sum_{V \setminus \{i_2 \to v\}} \prod_{\phi \in F_{\setminus (i_2 \to v)}} \phi \right)$$

$$\beta_v = \sum_{X - C_v} \psi_v \prod_{\phi} \phi = P_v(C_v)$$

CLIQUE TREE MESSAGE PASSING ALGORITHM

- A message sent between two cliques in the same direction is necessarily the same.
- For any given clique tree, each edge has two messages associated with it: one for each direction of the edge.
- For any given clique tree with c cliques, there are $c - 1$ edges and $2(c - 1)$ messages are computed.
- The messages are computed by following a simple asynchronous algorithm.
- Uses dynamic programming.

SUM-PRODUCT BELIEF PROPAGATION ALGORITHM

Algorithm 10.2 Calibration using sum-product message passing in a clique tree

Procedure CTree-SP-Calibrate (
  Φ, // Set of factors
  $\mathcal{T}$ // Clique tree over Φ
)

1 Initialize-Cliques
2 **while** exist i, j such that i is ready to transmit to j
3 $\delta_{i \to j}(\boldsymbol{S}_{i,j}) \leftarrow$ SP-Message(i, j)
4 **for** each clique i
5 $\beta_i \leftarrow \psi_i \cdot \prod_{k \in \text{Nb}_i} \delta_{k \to i}$
6 **return** $\{\beta_i\}$

SUM-PRODUCT BELIEF PROPAGATION ALGORITHM

- Uses dynamic programming.
- The algorithm is defined asynchronously, with each clique sending a message as soon as it is ready.
- In the upward pass, first pick a root and send all messages toward the root.
- In the downward pass, the root sends all the downward pass messages to all of its children.
- The algorithm continues until the leaves of the tree are reached.

CLIQUE CALIBRATION

DEFINITION

Two adjacent cliques C_i and C_j are calibrated if every pair of adjacent clusters C_i and C_j agree on their sepset $S_{i,j}$.

$$\sum_{C_i - S_{i,j}} \beta_i(C_i) = \sum_{C_j - S_{i,j}} \beta_j(C_j)$$

$$\text{Cluster Belief } \beta_i(C_i) = \psi_i \times \prod_{k \in Nbi} \delta_{k \to i}$$

Clique Calibration

Definition

A clique tree T *is calibrated if all pairs of adjacent cliques are calibrated. For a calibrated clique tree, clique beliefs are* $\beta_i(C_i)$ *and sepset beliefs are* $\mu_{i,j}$

$$\mu_{i,j}(S_{i,j}) = \sum_{C_i - S_{i,j}} \beta_i(C_i) = \sum_{C_j - S_{i,j}} \beta_j(C_j)$$

TABLE OF CONTENTS

MESSAGE PASSING WITH DIVISION

- For any edge between two clusters C_i and C_j , two messages are computed; $\delta_{j \to i}$ and $\delta_{i \to j}$.
- Let the first message be passed from C_i to C_j.
- Then the return message from C_j to C_i would be passed when C_j has received all messages from its neighbours.
- Once C_j has received all messages from its neighbours, compute its belief β_j

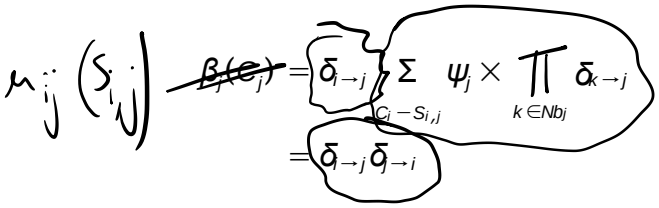
$$\beta_j(C_j) = \psi_j \times \prod_{k \in Nbj} \delta_{k \to j}$$

- Alternatively, message from C_j to C_i can be computed as

$$\delta_{j \to i} = \sum_{C_i - S_{i,j}} \psi \cdot \prod_{k \in (Nbj - i)} {}_j\delta_{k \to j i}$$

MESSAGE PASSING WITH DIVISION

- Rewrite



- Hence,

$$\delta_{j \to i} = \frac{\beta_j(C_j)}{\delta_{i \to j}}$$

LAURITZEN-SPIEGELHALTER ALGORITHM

Message Passing with Division

1. For each cluster C_i, initialize the cluster belief β_i as its cluster potential ψ_i and sepset potential $S_{i,j}$ between adjacent clusters C_i and C_j as 1.
2. In each iteration, the cluster belief β_i is updated by multiplying it with the message from its neighbours and the sepset potential $\{i - j\}$ is used to store the previous message passed along the edge $(i - j)$, irrespective of the direction of the message.
3. Whenever a new message is passed along an edge, it is divided by the old message to ensure that we don't count this message twice.
4. Marginalize the belief to get the message passed.

$$\delta_{i \to j} = \frac{\sum_{C_i - S_{i,j}} \beta_i}{\mu_{i,j}}$$

LAURITZEN-SPIEGELHALTER ALGORITHM

Message Passing with Division

5. Update the belief by multiplying it with the message from its neighbours.

$$\beta_j \leftarrow \beta_j \cdot \delta_{i \to j}$$

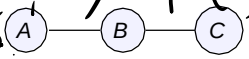
6. Update the sepset belief

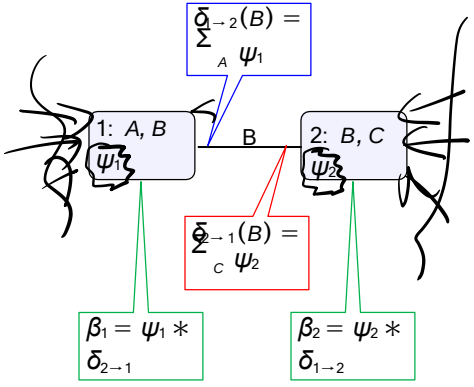
$$\mu_{i,j} = \sum_{C_i - S_{i,j}} \beta_i$$

7. Repeat Steps 2 and 3 until the tree is calibrated for each adjacent edge$(i - j)$

$$\mu_{i,j}(S_{i,j}) = \sum_{C_i - S_{i,j}} \beta_i = \sum_{C_j - S_{i,j}} \beta_j$$

BP – Belief Computation – Example





ψ_1		
	a^1b^1	3
	a^2b^1	-1
	a^1b^2	0
	a^2b^2	1

ψ_2		
	b^1c^1	4
	b^1c^2	-1
	b^2c^1	1
	b^2c^2	2

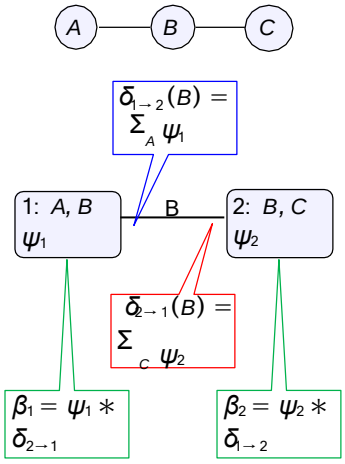
$\delta_{1\to 2}$	
b^1	2
b^2	1

$\delta_{2\to 1}$	
b^1	3
b^2	3

β_1		
	a^1b^1	3*3 = 9
	a^2b^1	-1*3 = -3
	a^1b^2	0*3 = 0
	a^2b^2	1*3 = 3

β_2		
	b^1c^1	4*2 = 8
	b^1c^2	-1*2 = -2
	b^2c^1	1*1 = 1
	b^2c^2	2*1 = 2

BP – CALIBRATION – EXAMPLE



ψ_1		
	a^1b^1	3
	a^2b^1	-1
	a^1b^2	0
	a^2b^2	1

ψ_2		
	b^1c^1	4
	b^1c^2	-1
	b^2c^1	1
	b^2c^2	2

$\delta_{1\to 2}$		
	b^1	2
	b^2	1

$\delta_{2\to 1}$		
	b^1	3
	b^2	3

β_1		
	a^1b^1	3*3 = 9
	a^2b^1	-1*3 = -3
	a^1b^2	0*3 = 0
	a^2b^2	1*3 = 3

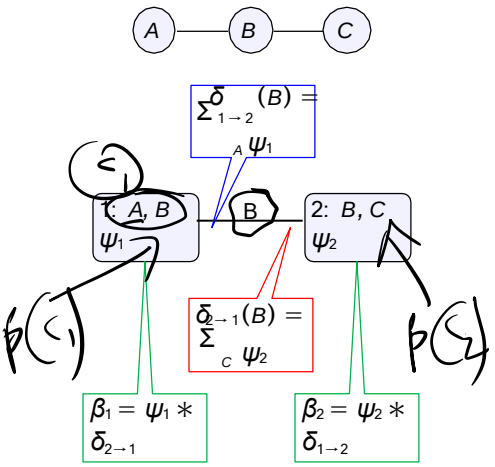
β_2		
	b^1c^1	4*2 = 8
	b^1c^2	-1*2 = -2
	b^2c^1	1*1 = 1
	b^2c^2	2*1 = 2

$\sum_A \beta_1$	
b^1	6
b^2	3

=

$\sum_C \beta_2$	
b^1	6
b^2	3

BP – CALIBRATION – EXAMPLE



$$\delta_{1\to 2} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\delta_{2\to 1} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

$$\sum_{C_1 - S_{12}} \beta(C_1) = \sum_{A} \beta_1 = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \end{bmatrix}$$

$$\sum_{C_2 - S_{12}} \beta(C_2) = \sum_{C} \beta_2 = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \end{bmatrix}$$

$$\mu(B) = \delta_{1\to 2} * \delta_{2\to 1} = \begin{bmatrix} 6 \\ 3 \end{bmatrix}$$

CALIBRATED CLIQUE TREE AS DISTRIBUTION

- At Convergence of clique tree calibration algorithm,

$$\beta_i = \psi_i \cdot \sum_{k \in Nbi} \delta_{k \to i} \quad (1)$$

$$\mu_{i,j}(S_{i,j}) = \delta_{j \to i} \delta_{i \to j} \quad (2)$$

$$\tilde{P}_\Phi(X) = \frac{\prod_{i \in V_T} \beta_i(C_i)}{\prod_{(i-j) \in E_T} \mu_{i,j}(S_{i,j})} \quad (3)$$

- Clique and sepsets beliefs provide a reparameterization of the unnormalized measure. This property is called the **clique tree invariant**.
- The distribution represented by a clique tree

$$\mathcal{P}_T = \frac{\prod_{i \in V_T} \beta_i(C_i)}{\prod_{(i-j) \in E_T} \mu_{i,j}(S_{i,j})} \quad (4)$$

Equivalence between sum-product and belief update

Theorem 10.5: Consider a set of sum-product initial potentials $\{\psi_i, i \in V_T\}$ and messages $\{\delta_{i\to j}, \delta_{j\to i} : i-j \in E_T\}$, and a set of belief-update beliefs $\{\beta_i : i \in V_T\}$ and messages $\{\mu_{i\to j}: i-j \in E_T\}$, for which equation (10.8) and equation (10.9) hold. For any pair of neighboring cliques C_i , C_j , let $\{\delta'_{i\to j}, \delta'_{j\to i} : i-j \in E_T\}$ be the set of sum-product messages following an application of SP-Message(i, j), and $\{\beta'_i : C_i \in T\}$, $\{\mu'_{i\to j} : (i-j) \in E_T\}$, be the set of belief-update beliefs following an application of BU-Message(i, j). Then equation (10.8) and equation (10.9) also hold for the new beliefs $\delta'_{i\to j}$, β'_i, $\mu'_{i\to j}$.

Equivalence between sum-product and belief update

For reference Equation 10.8 and Equation 10.9 are:

$$\beta_i = \psi_i \cdot \prod_{k \in Nb_i} \delta_{k \to i} \qquad (10.8)$$

$$\mu_{i,j}(s_{i,j}) = \delta_{i \to j} \cdot \delta_{j \to i} \qquad (10.9)$$

Equivalence Proof

Consider SPMessage (i, j)

$$\beta_j = \psi_j \prod_{k \in Nb_j} \delta_{k \to j} \quad j\mu_{ij}(s_{ij}) = \delta_{j \to i}\, \delta_{i \to j}$$

$$\beta_j' = \psi_j \prod_{k \in Nb_j - \{i\}} \delta_{k \to j} \quad \delta'_{i \to j}$$

$$\boxed{\beta_j' = \beta_j \frac{\delta'_{i \to j}}{\delta_{i \to j}}}$$

Equivalence Proof

Now consider BOMessage(i,j)

$$\beta_j' = \beta_j \frac{\sigma'_{i\to j}}{\mu_{i,j}}$$

$$\sigma'_{i\to j} = \sum_{c_i - \{s_{i,j}\}} \beta_i \qquad \beta_i = \sum_{c_i - s_{i,j}} \psi_i \prod_{k \in Nb_i} \delta_{k\to i}$$

$$= \sum_{c_i - s_{i,j}} \left(\psi_i \prod_{k \in Nb_i - \{j\}} \delta_{k\to i} \right) \delta_{j\to i} = \delta'_{i\to j} \delta_{j\to i}$$

Equivalence Proof

Thus in BUMessage(i,j) we have

$$\beta_j' = \frac{\beta_j \, \delta_{i\to j} \, \delta_{j\to i}}{\delta_{i\to j} \, \delta_{j\to i}} = \frac{\beta_j \, \delta_{i\to j}'}{\delta_{i\to j}}$$

same as β_j' in SPMessage(i,j)

Equivalence Proof

In BUMessage(i, j)

$$\mu'_{ij}(s_{ij}) = \sigma'_{i \to j}$$

We already saw that $\sigma'_{i \to j} = \delta'_{i \to j}\, \delta_{j \to i}$

$$\therefore \boxed{\mu'_{ij}(s_{ij}) = \delta'_{i \to j}\, \delta_{j \to i}}$$

But this is the same in SPMessage(i, j) too!

QUESTIONS

1. Construct a cluster graph for a BN or MN.
2. Construct a clique tree for a BN or MN and verify the properties.
3. Complete the missing statements in the given code snippet which implements belief propagation algorithm.
4. Complete the missing statements in the given code snippet which implements Lauritzen-Spiegelhalter algorithm.