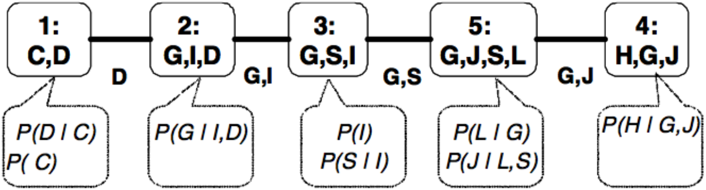
Probabilistic Graphical Models

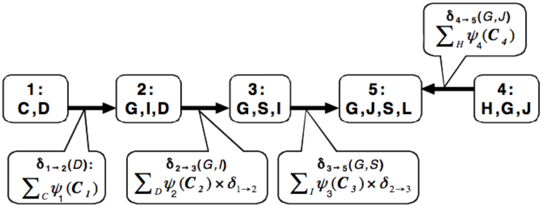
Raquel Urtasun and Tamir Hazan

TTI Chicago

April 27, 2011

Monday in class...





($\mathbf{C}_5$ is the root)

Clique Tree Message Passing I

- A general VE algorithm that can be implemented via message passing in a clique tree.
- Let $\mathcal{T}$ be a clique tree with cliques $\mathbf{C}_1, \cdots, \mathbf{C}_k$.
- Compute the initial potentials by multiplying the factors associated with each clique.
- Use the clique tree data structure to pass messages between neighboring clique.
- The messages are send towards the root.
- For each factor ϕ let's call $\alpha(\phi)$ the assigned clique.
- We define the **initial clique potential** of $\mathbf{C}_j$ as

$$\psi_j(\mathbf{C}_j) = \prod_{\phi:\alpha(\phi)=j} \phi$$

- This definition of ψ is different from the VE ψ. Why?

Clique Tree Message Passing II

- As each factor is assigned to exactly one clique

$$\prod_{\phi \in \Phi} \phi = \prod_j \psi_j$$

- Let $\mathbf{C}_r$ be the root.
- Perform sum-product VE over the cliques, starting from the leaves of the tree.
- For each clique $\mathbf{C}_i$, let Nb_i be the set of indices of cliques that are neighbors of $\mathbf{C}_i$.
- Let $p_r(i)$ be the upstream neighbor, i.e., the next one in the path to the root.
- For each clique $\mathbf{C}_i$, the message is computed by multiplying incoming messages from its downstream neighbors with its initial clique potential, resulting in a factor which scope is the clique.
- We sum out all the variables except those in the sepset between $\mathbf{C}_i$ and $\mathbf{C}_{pr(i)}$, and sends message to its upstream neighbor $\mathbf{C}_{pr(i)}$.

Clique Tree Message Passing III

- When the root clique has received all messages, it multiplies them with its own initial potential.
- The final clique potential is

$$\beta_r(\mathbf{C}_r) = \sum_{\mathcal{X}-\mathbf{C}_r} \prod_{\phi \in \mathbf{\Phi}} \phi$$

- As we will prove later

$$\hat{P}_{\mathbf{\Phi}}(\mathbf{C}_r) = \beta_r(\mathbf{C}_r)$$

Algorithm

Algorithm 10.1 Upward pass of variable elimination in clique tree

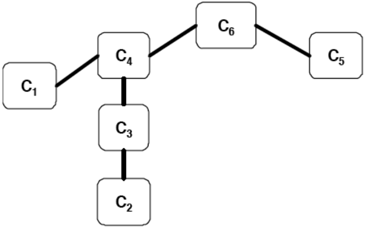
Procedure CTree-Sum-Product-Up (
  Φ, // Set of factors
  $\mathcal{T}$, // Clique tree over Φ
  α, // Initial assignment of factors to cliques
  $\boldsymbol{C}_r$ // Some selected root clique
)
  Initialize-Cliques
  while $\boldsymbol{C}_r$ is not ready
    Let $\boldsymbol{C}_i$ be a ready clique
    $\delta_{i \to p_r(i)}(\boldsymbol{S}_{i,p_r(i)}) \leftarrow$ SP-Message$(i, p_r(i))$
  $\beta_r \leftarrow \psi_r \cdot \prod_{k \in \mathrm{Nb}_{\boldsymbol{C}_r}} \delta_{k \to r}$
  return β_r

Procedure Initialize-Cliques (

)
  for each clique $\boldsymbol{C}_i$
    $\psi_i[\boldsymbol{C}_i] \leftarrow \prod_{\phi_j \,:\, \alpha(\phi_j)=i} \phi$

Procedure SP-Message (
  i, // sending clique
  j // receiving clique
)
  $\psi(\boldsymbol{C}_i) \leftarrow \psi_i \cdot \prod_{k \in (\mathrm{Nb}_i - \{j\})} \delta_{k \to i}$
  $\tau(\boldsymbol{S}_{i,j}) \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi(\boldsymbol{C}_i)$
  return $\tau(\boldsymbol{S}_{i,j})$

Example



- Assume $\mathbf{C}_6$ is the root.
- Which orderings are possible?
- And if $\mathbf{C}_1$ is the root?

Computing Marginals

- We can use the algorithm to compute the marginal probability of any set of query nodes $\mathbf{Y}$.
- We select the clique that contain them as the root $\mathbf{C}_r$, and perform the clique tree message passing towards that root.
- We then extract $\hat{P}_\phi(\mathbf{Y})$ from the final potential by summing out the other variables $\mathbf{C}_r - \mathbf{Y}$.
- We can also compute the partition function. How?

Correctness

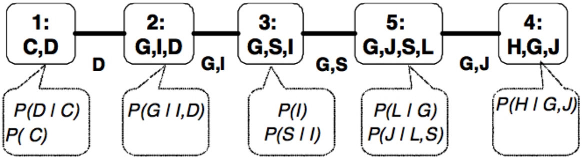
- We need to show that this algorithm when applied to a clique tree that satisfies the family preservation and the running intersection properties, computes the desire expressions to get the marginal probabilities.
- A variable X is eliminated only when a message is sent from $\mathbf{C}_i$ to $\mathbf{C}_j$ and $X \in \mathbf{C}_i$ and $X \notin \mathbf{C}_j$.

Prop: Let X be a variable that it's eliminated when a message is pass from $\mathbf{C}_i$ to $\mathbf{C}_j$. Then X does not appear anywhere in the tree on the $\mathbf{C}_j$ side of the edge $(i - j)$.

- Proof by contradiction, assume X appears in some other clique $\mathbf{C}_k$ which is on the other side of $\mathbf{C}_j$. Then $\mathbf{C}_j$ is on the path from $\mathbf{C}_i$ to $\mathbf{C}_k$.
- However we assume that X appears in $\mathbf{C}_i$ and $\mathbf{C}_k$ but not $\mathbf{C}_j$.
- This is a violation of the running intersection property.

Semantic interpretation of the messages

- For an edge $(i - j)$ let $\mathcal{F}_{\prec(i \to j)}$ be the set of factors in the cliques on the $\mathbf{C}_i$ side of the edge.
- Let $\mathcal{V}_{\prec(i \to j)}$ be the set of variables that appear on the $\mathbf{C}_i$ side but are not on the sepset.



- What's $\mathcal{F}_{\prec(3 \to 5)}$? And $\mathcal{V}_{\prec(3 \to 5)}$?
- The message passed between $\mathbf{C}_i$ and $\mathbf{C}_j$ is the product of all factors in $\mathcal{F}_{\prec(i \to j)}$, sum out all the variables not in the sepset.

Messages

Theorem: Let $\delta_{i \to j}$ be a message from $\mathbf{C}_i$ to $\mathbf{C}_j$, then

$$\delta_{i \to j}(\mathbf{S}_{i,j}) = \sum_{\mathcal{V}_{\prec(i \to j)}} \prod_{\phi \in \mathcal{F}_{\prec(i \to j)}} \phi$$

- Proof by induction. For the leaves $\mathbf{C}_i$ it is true by examining the operations in the clique.
- If $\mathbf{C}_i$ is not a leaf node, let's consider the expression on the right.
- Let $i_1, \cdots, i_m$ be the neighboring cliques of $\mathbf{C}_i$ other than $\mathbf{C}_j$.
- $\mathcal{V}_{\prec(i \to j)}$ is the disjoint union of $\mathcal{V}_{\prec(i_k \to i)}$ for $k = 1, \cdots, m$ and the variables eliminated at $\mathbf{C}_i$ itself.
- $\mathcal{F}_{\prec(i \to j)}$ is the disjoint union of the $\mathcal{F}_{\prec(i_k \to i)}$ and the factors $\mathcal{F}_i$ from which ψ_i was computed. Thus the right hand side is

$$\sum_{\mathbf{Y}_i} \sum_{\mathcal{V}_{\prec(i_1 \to i)}} \cdots \sum_{\mathcal{V}_{\prec(i_m \to i)}} \left(\prod_{\phi \in \mathcal{F}_{\prec(i_1 \to i)}} \phi \right) \cdot \cdots \cdot \left(\prod_{\phi \in \mathcal{F}_{\prec(i_m \to i)}} \phi \right) \cdot \left(\prod_{\phi \in \mathcal{F}_i} \phi \right)$$

Continuation of proof

$$\sum_{\mathbf{Y}_i} \sum_{\mathcal{V}_{\prec(i_1 \to i)}} \cdots \sum_{\mathcal{V}_{\prec(i_m \to i)}} \left(\prod_{\phi \in \mathcal{F}_{\prec(i_1 \to i)}} \phi \right) \cdot \cdots \cdot \left(\prod_{\phi \in \mathcal{F}_{\prec(i_m \to i)}} \phi \right) \cdot \left(\prod_{\phi \in \mathcal{F}_i} \phi \right)$$

- None of the variables in $\mathcal{V}_{\prec(i_k \to i)}$ appears in any other factor, thus

$$\sum_{\mathbf{Y}_i} \left(\prod_{\phi \in \mathcal{F}_i} \phi \right) \cdot \sum_{\mathcal{V}_{\prec(i_1 \to i)}} \left(\prod_{\phi \in \mathcal{F}_{\prec(i_1 \to i)}} \phi \right) \cdot \cdots \cdot \sum_{\mathcal{V}_{\prec(i_m \to i)}} \left(\prod_{\phi \in \mathcal{F}_{\prec(i_m \to i)}} \phi \right)$$

- Using the inductive hypothesis and the definition of ψ_i we have

$$\sum_{\mathbf{Y}_i} \psi_i \cdot \delta_{i_1 \to i} \cdot \cdots \cdot \delta_{i_m \to i}$$

- This is the operation to compute $\delta_{i \to j}$

Independences

- Cond. independence allows the message over the sepset to completely summarize the information on one side of the clique tree that is necessary for the other side.
- Let $\mathbf{C}_r$ be the root clique in a clique tree, and let $\beta_r(\mathbf{C}_r)$ be computed as in Algorithm 10.1 then

$$\beta_r(\mathbf{C}_r) = \sum_{\mathcal{X}-\mathbf{C}_r} \hat{P}_\phi(\mathcal{X})$$

Algorithm

Algorithm 10.1 Upward pass of variable elimination in clique tree

Procedure CTree-Sum-Product-Up (
  Φ, // Set of factors
  $\mathcal{T}$, // Clique tree over Φ
  α, // Initial assignment of factors to cliques
  $\boldsymbol{C}_r$ // Some selected root clique
)
  Initialize-Cliques
  while $\boldsymbol{C}_r$ is not ready
    Let $\boldsymbol{C}_i$ be a ready clique
    $\delta_{i \to p_r(i)}(\boldsymbol{S}_{i,p_r(i)}) \leftarrow$ SP-Message$(i, p_r(i))$
  $\beta_r \leftarrow \psi_r \cdot \prod_{k \in \text{Nb}_{\boldsymbol{C}_r}} \delta_{k \to r}$
  return β_r

Procedure Initialize-Cliques (
)
  for each clique $\boldsymbol{C}_i$
    $\psi_i[\boldsymbol{C}_i] \leftarrow \prod_{\phi_j \,:\, \alpha(\phi_j)=i} \phi$

Procedure SP-Message (
  i, // sending clique
  j // receiving clique
)
  $\psi(\boldsymbol{C}_i) \leftarrow \psi_i \cdot \prod_{k \in (\text{Nb}_i - \{j\})} \delta_{k \to i}$
  $\tau(\boldsymbol{S}_{i,j}) \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi(\boldsymbol{C}_i)$
  return $\tau(\boldsymbol{S}_{i,j})$

Independences

- Cond. independence allows the message over the sepset to completely summarize the information on one side of the clique tree that is necessary for the other side.
- Let $\mathbf{C}_r$ be the root clique in a clique tree, and let $\beta_r(\mathbf{C}_r)$ be computed as in Algorithm 10.1 then

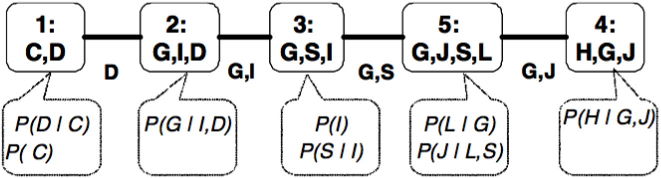
$$\beta_r(\mathbf{C}_r) = \sum_{\mathcal{X}-\mathbf{C}_r} \hat{P}_\phi(\mathcal{X})$$

- This algorithm applies to BN and Markov networks, with and without evidence.
- In Markov networks obtain the partition function by

$$Z_r = \sum_{\mathbf{C}_r} \beta_r(\mathbf{C}_r)$$

Complexity of VE in Clique Tree

- In some applications we are interested in computing the marginal probability for a large set of variables.
- Let's consider the task of computing the posterior distribution over every random variable in the network.
- If we do inference separately for each variable, the number of messages is $\mathcal{O}(nc)$, with c the cost of a single execution of clique tree inference.
- Less naive is to run the algorithm once for every clique, the number of messages is $\mathcal{O}(Kc)$, with K the number of cliques.
- We can do better.



Clique Calibration

- The computation between two neighboring cliques $\mathbf{C}_i$, $\mathbf{C}_j$ does not depend on the choice of root, only on the side on which the root is.
- This follows from the theorem that we just proved where

$$\delta_{i \to j}(\mathbf{S}_{i,j}) = \sum_{\mathcal{V}_{\prec(i \to j)}} \prod_{\phi \in \mathcal{F}_{\prec(i \to j)}} \phi$$

- Therefore, each clique tree has two messages associated with it: one for each direction.
- The complexity is then $\mathcal{O}(2(c-1))$, with c the number of cliques.
- **Def:** Let $\mathcal{T}$ be a clique tree. We say that $\mathbf{C}_i$ is ready to transmit to a neighbor $\mathbf{C}_j$, when $\mathbf{C}_i$ has messages from all of its neighbors except for $\mathbf{C}_j$.
- We can have an asynchronous algorithm, where when $\mathbf{C}_i$ is ready to transmit, it computes $\delta_{i \to j}(\mathbf{S}_{i,j})$.
- This is computed by multiplying all the incoming messages with the initial potentials and marginalizing out the variables $\mathbf{C}_i - \mathbf{S}_{i,j}$.

Calibration Algorithm or Sum-product BP

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

Procedure CTree-Sum-Product-Calibrate (
  Φ, // Set of factors
  $\mathcal{T}$ // Clique tree over Φ
)
  Initialize-Cliques
  while exist i, j such that i is ready to transmit to j
    $\delta_{i \to j}(\boldsymbol{S}_{i,j}) \leftarrow$ SP-Message(i, j)
  for each clique i
    $\beta_i \leftarrow \psi_i \cdot \prod_{k \in \mathrm{Nb}_i} \delta_{k \to i}$
  return $\{\beta_i\}$

(calibration algorithm)

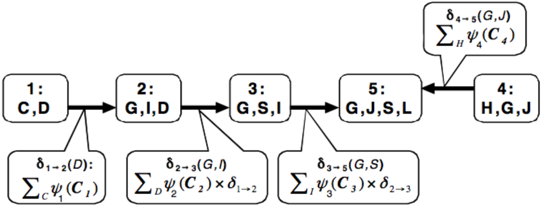
Procedure SP-Message (
  i, // sending clique
  j // receiving clique
)
  $\psi(\boldsymbol{C}_i) \leftarrow \psi_i \cdot \prod_{k \in (\mathrm{Nb}_i - \{j\})} \delta_{k \to i}$
  $\tau(\boldsymbol{S}_{i,j}) \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi(\boldsymbol{C}_i)$
  return $\tau(\boldsymbol{S}_{i,j})$

(message computation)

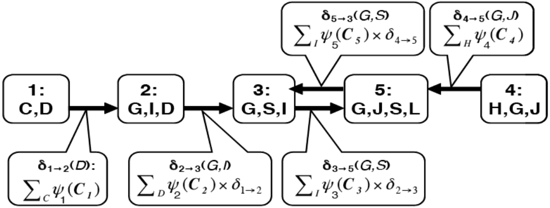
Sum-Product BP

- The calibration algorithm is also called **Sum product Belief Propagation**.
- The algorithm is defined asynchronously, with each clique sending a message when ready.
- Is this process guaranteed to terminate?
- This algorithm is equivalent to another algorithm where there is an upward pass and a downward pass.
- In the upward pass we pick a root and send messages towards it.
- When the process is complete, the root has all messages and sends them downward, until the leaves.
- The asynchronous algorithm is equivalent to this one, where the root is simply the first clique that happens to obtain messages form all of its neighbors.

Example



(Upward pass: $\mathbf{C}_5$ is the root, upward pass)



(First step downward pass: $\mathbf{C}_5$ sends message to $\mathbf{C}_3$)

Marginals and Sum Product BP I

- Assume that for each clique $\mathbf{C}_i$, $\beta_i(\mathbf{C}_i)$ is computed as in Algorithm 10.2 (i.e., Sum product Belief Propagation) then

$$\beta_i(\mathbf{C}_i) = \sum_{\mathcal{X} - \mathbf{C}_i} \hat{P}_\phi(\mathcal{X})$$

- For this to be true, the message $\delta_{i \to j}$ has to be computed based on the initial potential ψ_i, and not the modified potential β_i.
- Otherwise we do double-counting.
- At the end of the algorithm, each clique has the marginal probability over the variables of the scope.
- We compute the marginal over a single variable by selecting a clique that contains this variable in the scope, and marginalizing the other variables.

Marginals and Sum Product BP II

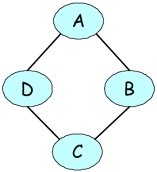
- If X appears in two cliques, they should agree in their marginals.
- **Def:** A clique tree $\mathcal{T}$ with potentials $\beta_i(\mathbf{C}_i)$ for each clique $\mathbf{C}_i$ is said to be **calibrated** if for all pairs of neighboring cliques we have that

$$\sum_{\mathbf{C}_i - \mathbf{S}_{i,j}} \beta_i(\mathbf{C}_i) = \sum_{\mathbf{C}_j - \mathbf{S}_{i,j}} \beta_j(\mathbf{C}_j)$$

- Sum product BP computes the posterior probability of all variables using only twice the computation of the upwards pass!
- Thus the cost is $\mathcal{O}(2c)$.
- Remember that the cost was $\mathcal{O}(nc)$ for independent computations and $\mathcal{O}(Kc)$ when it's done for each clique.

Calibrated Trees and Distribution

- A calibrated tree can be viewed as an alternative representation for $\hat{P}_\Phi$.



Assignment			$\max_C$
a^0	b^0	d^0	600000
a^0	b^0	d^1	300030
a^0	b^1	d^0	5000500
a^0	b^1	d^1	1000
a^1	b^0	d^0	200
a^1	b^0	d^1	1000100
a^1	b^1	d^0	100010
a^1	b^1	d^1	200000

$\beta_1[A, B, D]$

Assignment		$\max_{A,C}$
b^0	d^0	600200
b^0	d^1	1300130
b^1	d^0	5100510
b^1	d^1	201000

$\mu_{1,2}(B, D)$

Assignment			$\max_A$
b^0	c^0	d^0	300100
b^0	c^0	d^1	1300000
b^0	c^1	d^0	300100
b^0	c^1	d^1	130
b^1	c^0	d^0	510
b^1	c^0	d^1	100500
b^1	c^1	d^0	5100000
b^1	c^1	d^1	100500

$\beta_2[B, C, D]$

Assignment				Unnormalized	Normalized
a^0	b^0	c^0	d^0	300000	0.04
a^0	b^0	c^0	d^1	300000	0.04
a^0	b^0	c^1	d^0	300000	0.04
a^0	b^0	c^1	d^1	30	$4.1 \cdot 10^{-6}$
a^0	b^1	c^0	d^0	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^0	d^1	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^1	d^0	5000000	0.69
a^0	b^1	c^1	d^1	500	$6.9 \cdot 10^{-5}$
a^1	b^0	c^0	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^0	d^1	1000000	0.14
a^1	b^0	c^1	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^1	d^1	100	$1.4 \cdot 10^{-5}$
a^1	b^1	c^0	d^0	10	$1.4 \cdot 10^{-6}$
a^1	b^1	c^0	d^1	100000	0.014
a^1	b^1	c^1	d^0	100000	0.014
a^1	b^1	c^1	d^1	100000	0.014

Another example

- Consider the case of a chain $A - B - C - D$.
- What are the cliques?
- What's $\beta_i(\mathbf{C}_i)$?
- What's $\hat{P}_\Phi(C|B)$?
- And $\hat{P}_\Phi(A, B, C)$?
- Can I compute the joint $\hat{P}_\Phi(A, B, C)$ in multiple ways?