Lecture 14: March 9, 2015
cs 573: Probabilistic Reasoning
Professor Nevatia
Spring 2015

Review

- Assignment # 4 due today
- Assignment #5 to be posted Mar 11, due Mar 25
- Exam 1 graded by March 11
  - How to provide access?
- Exam 2, April 29, class period; NOT cumulative
- Previous lecture:
  - Gaussian networks
    - Representation of Gaussian distributions
    - Gaussian BNs and MRFs
    - Inference algorithms
- Today's objective
  - MAP Inference

Canonical Form

- Co-variance Form

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]$$

- Canonical form:

$$\begin{aligned} K &= \Sigma^{-1} \\ \boldsymbol{h} &= \Sigma^{-1}\boldsymbol{\mu} \\ g &= -\frac{1}{2}\boldsymbol{\mu}^T\Sigma^{-1}\boldsymbol{\mu} - \log\left((2\pi)^{n/2}|\Sigma|^{1/2}\right) \end{aligned}$$

$$\mathcal{C}(\boldsymbol{X}; K, \boldsymbol{h}, g) = \exp\left(-\frac{1}{2}\boldsymbol{X}^T K \boldsymbol{X} + \boldsymbol{h}^T\boldsymbol{X} + g\right) \qquad \text{Eq 14.1}$$

Marginalize/Conditionalize a Gaussian Distribution

$$p(\boldsymbol{X}, \boldsymbol{Y}) = \mathcal{N}\left(\left(\begin{array}{c} \boldsymbol{\mu}_{\boldsymbol{X}} \\ \boldsymbol{\mu}_{\boldsymbol{Y}} \end{array}\right); \left[\begin{array}{cc} \Sigma_{\boldsymbol{X}\boldsymbol{X}} & \Sigma_{\boldsymbol{X}\boldsymbol{Y}} \\ \Sigma_{\boldsymbol{Y}\boldsymbol{X}} & \Sigma_{\boldsymbol{Y}\boldsymbol{Y}} \end{array}\right]\right)$$

- Marginal over **Y** (sum out **X**) is given by N($\boldsymbol{\mu}_{\mathbf{Y}}$, $\boldsymbol{\Sigma}_{\mathbf{YY}}$)
- Conditioning is easier in the canonical form:

$$\begin{aligned} K' &= K_{\boldsymbol{X}\boldsymbol{X}} \\ h' &= \boldsymbol{h}_{\boldsymbol{X}} - K_{\boldsymbol{X}\boldsymbol{Y}}\boldsymbol{y} \\ g' &= g + \boldsymbol{h}_{\boldsymbol{Y}}^T\boldsymbol{y} - \frac{1}{2}\boldsymbol{y}^T K_{\boldsymbol{Y}\boldsymbol{Y}}\boldsymbol{y} \end{aligned}$$

Gaussian Bayesian Networks (GBN)

- In a GBN, all variables are continuous; all CPDs are linear Gaussian

Let Y be a continuous variable with continuous parents $X_1, \ldots, X_k$. We say that Y has a linear Gaussian *model if there are parameters $\beta_0, \ldots, \beta_k$ and σ^2 such that*

$$p(Y \mid x_1, \ldots, x_k) = \mathcal{N}\left(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k; \sigma^2\right).$$

In vector notation,

$$p(Y \mid \boldsymbol{x}) = \mathcal{N}\left(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}; \sigma^2\right).$$

– Thm 7.3

Given $p(Y \mid \boldsymbol{x}) = \mathcal{N}\left(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}; \sigma^2\right)$; $X_1, \ldots, X_k$ distributed as $N(\boldsymbol{\mu}, \Sigma)$

We can show that

$$\begin{aligned} \mu_Y &= \beta_0 + \boldsymbol{\beta}^T \boldsymbol{\mu} \\ \sigma_Y^2 &= \sigma^2 + \boldsymbol{\beta}^T \Sigma \boldsymbol{\beta}. \end{aligned}$$

$$Cov[X_i; Y] = \sum_{j=1}^{k} \beta_j \Sigma_{i,j}.$$

Follows that in a GBN, joint distribution is Gaussian

Distribution to GBN

- Previous slide shows how given a GBN (network parameters), we can get joint distribution parameters.
- Reverse: given the joint distribution, recover the linear model (thm 7.4)

Given:

$$p(\boldsymbol{X}, \boldsymbol{Y}) = \mathcal{N}\left(\left(\begin{array}{c} \boldsymbol{\mu}_{\boldsymbol{X}} \\ \boldsymbol{\mu}_{\boldsymbol{Y}} \end{array}\right); \left[\begin{array}{cc} \Sigma_{\boldsymbol{X}\boldsymbol{X}} & \Sigma_{\boldsymbol{X}\boldsymbol{Y}} \\ \Sigma_{\boldsymbol{Y}\boldsymbol{X}} & \Sigma_{\boldsymbol{Y}\boldsymbol{Y}} \end{array}\right]\right)$$

Derive:

$$p(Y \mid \boldsymbol{X}) = \mathcal{N}\left(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{X}; \sigma^2\right)$$

$$\begin{aligned} \beta_0 &= \mu_Y - \Sigma_{Y\boldsymbol{X}} \Sigma_{\boldsymbol{X}\boldsymbol{X}}^{-1} \boldsymbol{\mu}_{\boldsymbol{X}} \\ \boldsymbol{\beta} &= \Sigma_{\boldsymbol{X}\boldsymbol{X}}^{-1} \Sigma_{Y\boldsymbol{X}} \\ \sigma^2 &= \Sigma_{YY} - \Sigma_{Y\boldsymbol{X}} \Sigma_{\boldsymbol{X}\boldsymbol{X}}^{-1} \Sigma_{\boldsymbol{X}Y} \end{aligned}$$

Distribution to GBN

- Given a distribution over *n* variables, we can construct a GBN (BN with linear Gaussian Models) that is an I-map of the distribution
- A key difference with discrete case: number of parameters in GBN is not necessarily smaller than in the joint distribution itself as the joint distribution is compact by itself.
- We can also go from distributions to Markov networks
  - J*ij* help define pairwise (log) potentials
  - However, additional complexity in case of MN; not every set of potentials induces a valid Gaussian distributions
    - Sufficient but not necessary condition is that each edge potential be normalizable (corresponding information matrix is positive definite)
  - We skip other details of Gaussian Markov Random Fields (sec 7.3)

Inference in Networks with Continuous Variables

- Essentially, all the algorithms for discrete case apply
- Sum-Product algorithm
  - Steps consist of multiplying factors (product) and marginalizing over some variables (sum) and passing messages to other nodes
  - Iterate until convergence (two passes suffice for trees)
  - We already know how to marginalize Gaussians
  - Now consider product and division
- Product of two canonical forms over the same set of variables:

  $C(\mathbf{X}, K_1, \mathbf{h}_1, g_1) . C(\mathbf{X}, K_2, \mathbf{h}_2, g_2) = C(\mathbf{X}, K_1 + K_2, \mathbf{h}_1 + \mathbf{h}_2, g_1 + g_2)$
- Division:

$$\frac{\mathcal{C}(K_1, \boldsymbol{h}_1, g_1)}{\mathcal{C}(K_2, \boldsymbol{h}_2, g_2)} = \mathcal{C}(K_1 - K_2, \boldsymbol{h}_1 - \boldsymbol{h}_2, g_1 - g_2)$$

- If scope is different, expand by adding variables and zero entries

SP and BP Algorithms

- Sum-Product algorithm
  - As before; need to show that resulting factors maintain K matrices to be positive definite so integration is not infinite
  - Shown in Proposition 14.1
- Gaussian Belief Propagation
  - Similar steps as in discrete case but the observation that the potentials are quadratic simplifies the equations (eqs 14.7 to 14.9)
  - **Interesting property**: If BP converges, resulting beliefs encode the correct means but estimated variances are underestimates (overconfident estimates)
  - Also convergence guaranteed if pairwise normalizability condition holds.
- We skip other details

Hybrid Networks

- Mix of discrete and continuous variables
- In general, even representation is difficult
  - Restrict to continuous linear Gaussian (CLG) networks
    - No discrete variables with continuous parents
- Even for CLG networks, exact inference is difficult
  - Even though the initial factors are all Gaussians, intermediate factors are not necessarily Gaussians (but mixtures of Gaussians)
  - Number of Gaussian to mix grows exponentially in the size of the network
- Approximation techniques need to be used; we omit details

MAP Queries

- Often, we are interested in joint assignment of variables that have the highest probability, rather than the individual distributions
  - Decoding a signal
  - Word sequence from a speech signal…
  - Classification/diagnosis tasks: debugging a piece of equipment, object recognition.
- MAP (maximum *a posteriori*) Query
  - Maximum a posteriori probability assignment or most probable explanation (MPE)
  - MAP $(\mathbf{W} | \boldsymbol{e}) = \arg\max_{w} P(\mathbf{w},\mathbf{e})$, $\mathbf{W} = \boldsymbol{X} - \mathbf{E}$
  - (Notation: $\arg\max_x f(x)$ = value of x for which f(x) is maximal)
  - Note that maximal joint assignment is not same as maxima of individual assignments, example on next slide.

MAP Example

Consider a two node chain $A \to B$ where A and B are both binary-valued. Assume that:

a^0	a^1
0.4	0.6

A	b^0	b^1
a^0	0.1	0.9
a^1	0.5	0.5

CPD tables (2.13)

$\mathrm{MAP}(A) = a^1$

However, $\mathrm{MAP}(A, B) = (a^0, b^1)$:

$\arg\max_{a,b} P(a, b) \neq (\arg\max_a P(a), \arg\max_b P(b))$

Overview

- Task is to compute:

$$\xi^{map} = \arg\max_{\xi} P_{\Phi}(\xi) = \arg\max_{\xi} \frac{1}{Z}\tilde{P}_{\Phi}(\xi) = \arg\max_{\xi} \tilde{P}_{\Phi}(\xi).$$

- Note that we don't need to compute partition function if we just want variable assignments but do need it for computing the actual max probability
- Can adapt the *sum-product* algorithm to compute MAP by changing the sum operation to a max operation- *max product* algorithm
- If we work with log of probabilities, we get a *max sum* algorithm
- If we work with energy factors, we get a *min sum* algorithm

Marginal MAP Query

- We may only care about the assignments of a subset of the variables
  - Disease diagnosis: full MAP query would compute joint distribution of diseases, symptoms, and test outcomes; we may only be interested in disease probabilities
  - MAP ($\mathbf{Y}$| $\boldsymbol{e}$) = arg $\max_{\mathbf{y}}$ P($\mathbf{y}$|$\mathbf{e}$) = arg $\max_{\mathbf{y}}$ P($\mathbf{y}$, $\mathbf{e}$)
  - Let $\mathbf{Z} = \boldsymbol{X} - \mathbf{Y} - \mathbf{E}$
  - MAP ($\mathbf{Y}$| $\boldsymbol{e}$) = arg $\max_{\mathbf{y}}$ $\{\sum_{\boldsymbol{z}} P\,(\mathbf{Y}, \mathbf{Z} \mid \boldsymbol{e})\}$
  - Note: marginal MAP query can not be computed directly from a MAP query (can not reverse summation and maximization operations)
- Complexity of computing MAP query is NP-complete
- Complexity of marginal MAP is even higher (NP^{PP})
- MAP Problem is NP-hard even for a polytree

VE for MAP

- Example 13.1. Simple network A → B, no evidence variables
- $\max_{a,b} P(a,b) = \max_{a,b} P(a)\, P(b|a)$

  $= \max_a \max_b P(a)\, P(b|a)$

  $= \max_a P(a) \max_b P(b|a)$
- Let $\phi(a)$ denote $\max_b P(b|a)$

a^0	a^1
0.4	0.6

A	b^0	b^1
a^0	0.1	0.9
a^1	0.55	0.45.

$$\phi(a^1) = \max_b P(b \mid a^1) = 0.55 \textit{ and } \phi(a^0) = \max_b P(b \mid a^0) = 0.9.$$

$$\max_a P(a)\phi(a) = \max\,[0.4 \cdot 0.9, 0.6 \cdot 0.55] = 0.36.$$

- How to find the MAP assignments?
  - When we computed $\phi(A)$, we could not know what value of B would lead to MAP solution
  - At the last step, we know that A= a^1 is selected, so B = b^0

Factor Maximization

- Definition 13.2

Let $\boldsymbol{X}$ be a set of variables, and $Y \notin \boldsymbol{X}$ a variable. Let $\phi(\boldsymbol{X}, Y)$ be a factor. We define the factor maximization of Y in ϕ to be factor ψ over $\boldsymbol{X}$ such that:

$$\psi(\boldsymbol{X}) = \max_{Y} \phi(\boldsymbol{X}, Y).$$

■

Max-marginalization over B

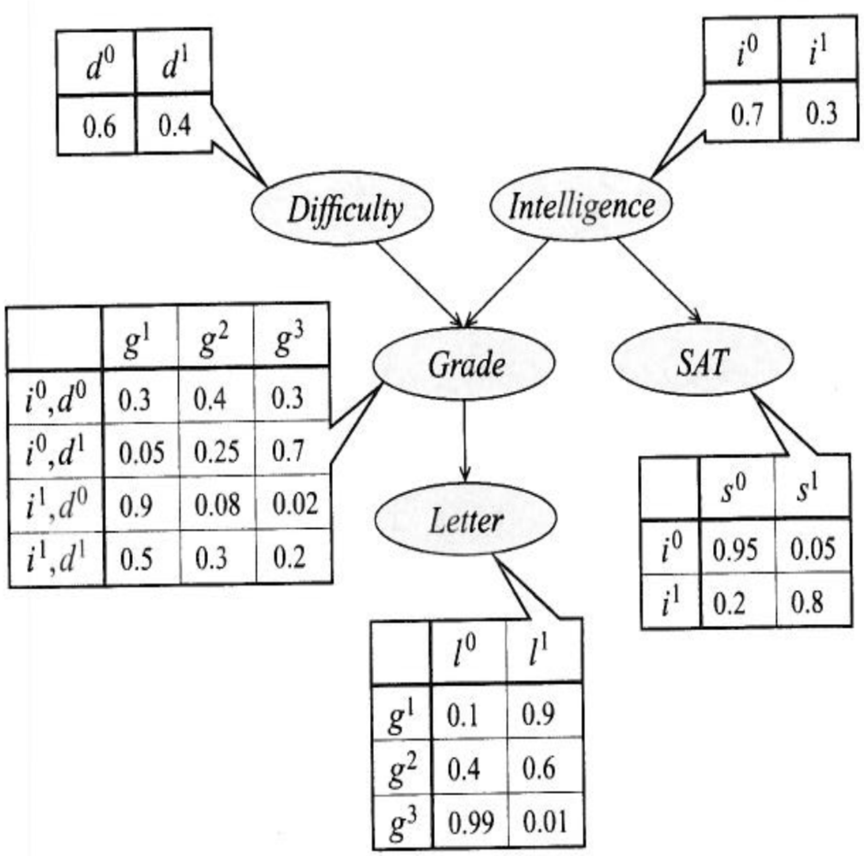
a^1	b^1	c^1	0.25
a^1	b^1	c^2	0.35
a^1	b^2	c^1	0.08
a^1	b^2	c^2	0.16
a^2	b^1	c^1	0.05
a^2	b^1	c^2	0.07
a^2	b^2	c^1	0
a^2	b^2	c^2	0
a^3	b^1	c^1	0.15
a^3	b^1	c^2	0.21
a^3	b^2	c^1	0.09
a^3	b^2	c^2	0.18

a^1	c^1	0.25
a^1	c^2	0.35
a^2	c^1	0.05
a^2	c^2	0.07
a^3	c^1	0.15
a^3	c^2	0.21

Rules for Factor Maximization

- $\sum_x (\phi_1 . \phi_2) = \phi_1 . \sum_x \phi_2$, if X is not in domain of ϕ_1
- $\max_x (\phi_1 . \phi_2) = \phi_1 . \max_x \phi_2$, X is not in domain of ϕ_1
- $\max_x (\phi_1 + \phi_2) = \phi_1 + \max_x \phi_2$, X is not in domain of ϕ_1

Max Product VE: Example



Eliminate variables in order S,I,D,L,G

Max instead of sum at each step

τ_1 (I)= $\max_s \phi_s(I,s)$

τ_1 (i^0)= .95 τ_1 (i^1)= .8

Note different values of S for the two values above

See table on next slide for other variables.

Example

Step	Variable eliminated	Factors used	Intermediate factor	New factor
1	S	$\phi_S(I,S)$	$\psi_1(I,S)$	$\tau_1(I)$
2	I	$\phi_I(I)$, $\phi_G(G,I,D)$, $\tau_1(I)$	$\psi_2(G,I,D)$	$\tau_2(G,D)$
3	D	$\phi_D(D)$, $\tau_2(G,D)$	$\psi_3(G,D)$	$\tau_3(G)$
4	L	$\phi_L(L,G)$	$\psi_4(L,G)$	$\tau_4(G)$
5	G	$\tau_4(G)$, $\tau_3(G)$	$\psi_5(G)$	$\tau_5(\emptyset)$

Note that this procedure does not provide the values of eliminated variables that give rise to the maximum

Consider elimination of S, we don't consider what value of S gave us τ (I)

TraceBack

- At the last elimination, we do have access to the argument that maximizes the corresponding factor (ψ_5), this gives a value for g^*
- Given g^* we can compute $l^* = \text{argmax}_l\ \psi_4(g^*, l)$
- Continue, one variable at a time
- $d^* = \text{argmax}_d\ \psi_3(g^*, d)$

  $i^* = \text{argmax}_d\ \psi_2(g^*, i, d^*)$

  $s^* = \text{argmax}_s\ \psi_1(i^*, s)$

- Pseudo-code on following slides
  - Identical to sum-product except that "sum" is replaced by "max" and *traceback* function is added

Procedure Max-Product-VE (
  Φ, // Set of factors over $\boldsymbol{X}$
  $\prec$ // Ordering on $\boldsymbol{X}$
)
  Let $X_1, \ldots, X_k$ be an ordering of $\boldsymbol{X}$ such that
    $X_i \prec X_j$ iff $i < j$
  for $i = 1, \ldots, k$
    $(\Phi, \phi_{X_i}) \leftarrow$ Max-Product-Eliminate-Var(Φ, X_i)
  $\boldsymbol{x}^* \leftarrow$ Traceback-MAP$(\{\phi_{X_i} : i = 1, \ldots, k\})$
  return $\boldsymbol{x}^*, \Phi$ // Φ contains the probability of the MAP

Procedure Max-Product-Eliminate-Var (
  Φ, // Set of factors
  Z // Variable to be eliminated
)
  $\Phi' \leftarrow \{\phi \in \Phi : Z \in Scope[\phi]\}$
  $\Phi'' \leftarrow \Phi - \Phi'$
  $\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
  $\tau \leftarrow \max_Z \psi$
  return $(\Phi'' \cup \{\tau\}, \psi)$

Procedure Traceback-MAP (
$\{\phi_{X_i} \; : \; i = 1, \ldots, k\}$
)
for $i = k, \ldots, 1$
$\boldsymbol{u}_i \leftarrow \; (x^*_{i+1}, \ldots, x^*_k)\langle \mathit{Scope}[\phi_{X_i}] - \{X_i\}\rangle$
// The maximizing assignment to the variables eliminated after X_i
$x^*_i \leftarrow \; \arg\max_{x_i} \phi_{X_i}(x_i, \boldsymbol{u}_i)$
// x^*_i is chosen so as to maximize the corresponding entry in the factor, relative to the previous choices $\boldsymbol{u}_i$
return x^*

VE with a Marginal MAP

- We want to find max over some set **Y** but don't care about the rest (set **W**)
- $\boldsymbol{y}^{\text{m-map}} = \arg\max_{y} P_{\phi}(\boldsymbol{y}) = \arg\max_{\boldsymbol{y}} \ \Sigma_{\mathbf{W}} P_{\phi}^{\sim}(\boldsymbol{y}, \mathbf{W})$
- What we want to compute is a max sum product

$$\max_{\mathbf{Y}} \sum_{\mathbf{W}} \prod_{\phi \varepsilon \Phi} \phi.$$

- More difficult: as we must compute both sum and max functions. Can be shown that the sum computations must be done first and max later: the two are not commutative
  - Less freedom in choosing the elimination order
- Example on next slide

$$\max_{S,L} \sum_{G,I,D} P(I, D, G, S, L).$$

$$\begin{aligned}
\psi_1(I,G,D) &= \phi_D(D) \cdot \phi_G(G,I,D) \\
\tau_1(I,G) &= \sum_D \psi_1(I,G,D)
\end{aligned}$$

$$\begin{aligned}
\psi_2(S,G,I) &= \phi_I(I) \cdot \phi_S(S,I) \cdot \tau_1(I,G) \\
\tau_2(S,G) &= \sum_I \psi_2(S,G,I)
\end{aligned}$$

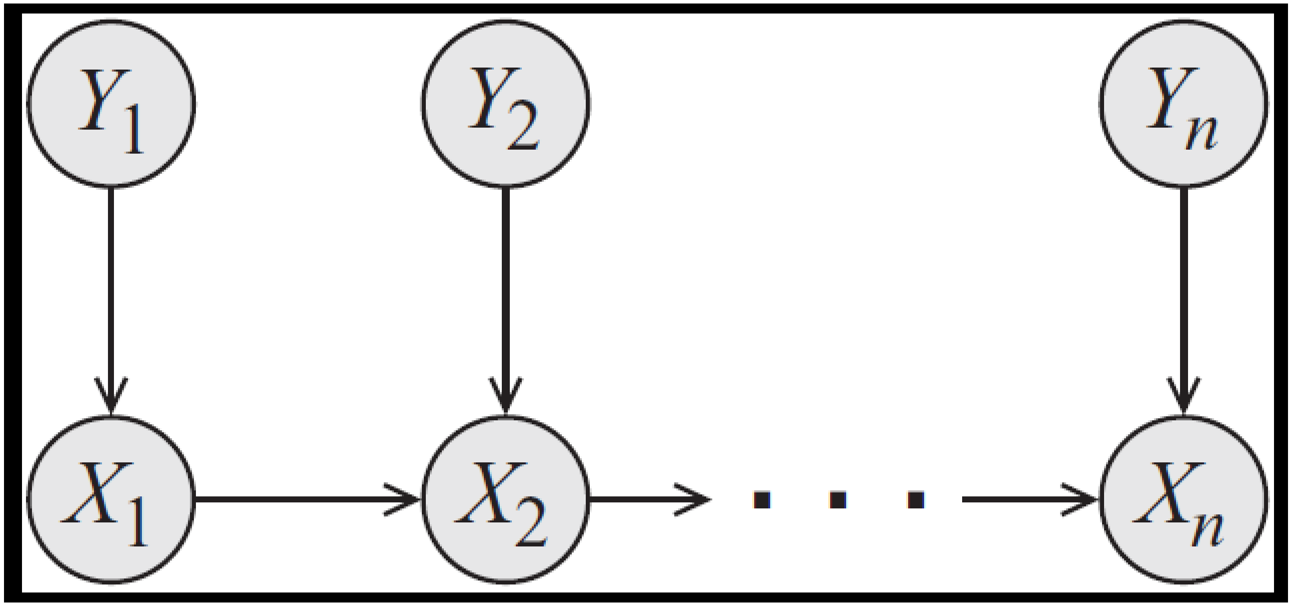
$$\begin{aligned}
\psi_3(S,G,L) &= \tau_2(S,G) \cdot \phi_L(L,G) \\
\tau_3(S,L) &= \sum_G \psi_3(S,G,L)
\end{aligned}$$

$$\begin{aligned}
\psi_4(S,L) &= \tau_3(S,L) \\
\tau_4(L) &= \max_S \psi_4(S,L)
\end{aligned}$$

$$\begin{aligned}
\psi_5(L) &= \tau_4(L) \\
\tau_5(\emptyset) &= \max_L \psi_5(L).
\end{aligned}$$

Computationally Complex Example

- Goal is to compute $\boldsymbol{y}^{\text{m-map}} = \arg\max_{\mathrm{Y}} \Sigma_{\mathbf{X}} \mathrm{P}(\mathbf{Y}, \mathbf{X})$



- First eliminate X_i variables in order
- Resulting factor will contain all of the Y_{i} variables, so exponential in n
- Find max over Y_i variables

Max-Product in Clique Trees

- Compute max-marginals over cliques instead of sum-marginals
- Advantage of this algorithm over VE is less clear as we are not seeking distributions over all variables (just over one assignment)
- May show other assignments that have nearly the same probability as the max assignment
- Method generalizes to loopy BP calculations

Algorithm 13.2 Max-product message computation for MAP

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

$\psi(\boldsymbol{C}_i) \leftarrow \psi_i \cdot \prod_{k \in (\mathrm{Nb}_i - \{j\})} \delta_{k \to i}$

$\tau(\boldsymbol{S}_{i,j}) \leftarrow \max_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi(\boldsymbol{C}_i)$

return $\tau(\boldsymbol{S}_{i,j})$

Definition: Max-marginal of a Function

- Max-marginal of a function f, relative to a set of variables $\mathbf{Y}$ is said to be:

$$MaxMarg_f(\boldsymbol{y}) = \max_{\xi\langle \boldsymbol{Y}\rangle = \boldsymbol{y}} f(\xi), \qquad \textit{for any assignment } \boldsymbol{y} \in Val(\boldsymbol{Y})$$

- Consider $\text{MaxMarg}_{\tilde{P}_\Phi}$ (Y) : not a single number but a factor
  - For each assignment of values y to Y, the MaxMarg function corresponds to max of $\tilde{P}_\Phi$ (y); note that P may contain other variables whose values are being chosen to maximize the expression

Max-marginal

- After convergence, in a clique tree, beliefs in each clique will be max-marginals

$$\beta_i(\mathbf{c}_i) = MaxMarg_{\tilde{P}_\Phi}(\mathbf{c}_i).$$

- For each assignment c_i to variables in the clique C_i , belief is the unnormalized max of all assignments elsewhere in the graph, consistent with the clique assignments, c_i .
- Max-marginals agree over the variables in the sepset, as is the case for the product sum algorithm, i.e. clique tree is *max-calibrated*
- Surprisingly, the resulting beliefs still are a reparameterization of the probability function (multiply all and divide by product of beliefs on sepset variables gives the original, unnormalized distribution)

Decoding the Max-Marginals

- More complex than in the VE algorithm
- Find assignments, ξ^* that are locally optimal (for one clique)
- Can be shown that set of such assignments over all the cliques defines the global MAP
  - Theorem 13.6, Lemma 13.1; rather difficult to follow
- In case of multiple assignments having the same value, choose one and *extend* it
  - Find a locally optimal assignment for one clique, for the neighboring clique, use this assignment for the common variables and iterate

Max Product BP in Loopy Graphs

- Similar to the Sum Product BP algorithm but Sum is replaced by Max
- Convergence is not guaranteed
- Max-marginals at convergence are not necessarily the correct max-marginals of the distribution: call them *pseudo max-marginals*
- Recovering assignments is more complex as the local optimality property may not be satisfied (i.e. globally optimal assignment may not be locally optimal)
  - A globally consistent solution may not exist
  - Can not just extend the locally optimal assignments
  - Search to find an assignment that satisfies all local optimal conditions
    - Equivalent to a constraint satisfaction problem
  - Fallback: take assignments to variables that are unambiguous, do exact inference on rest if needed

Next Class

- Read section 11.5.1 of the KF book