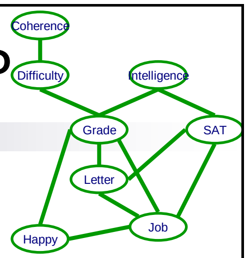
Readings:
K&F: 10.2, 10.3

Generalized Belief Propagation

Graphical Models – 10708
Carlos Guestrin
Carnegie Mellon University
November 12th, 2008

More details on Loopy BP



- Numerical problem:
  - messages < 1 get multiplied together as we go around the loops
  - numbers can go to zero
  - normalize messages to one:

$$\delta_{i\to j}(X_j) = \frac{1}{Z_{i\to j}} \sum_{x_i} \phi_i(x_i)\phi_{ij}(x_i, X_j) \prod_{k\in\mathcal{N}(i)-j} \delta_{k\to i}(x_i)$$

  - Z_{ij} doesn't depend on X_j, so doesn't change the answer
- Computing node "beliefs" (estimates of probs.):

$$b_i(x_i) = \widehat{P}(X_i) = \frac{1}{Z_i}\phi_i(X_i) \prod_{k\in\mathcal{N}(i)} \delta_{k\to i}(X_i)$$

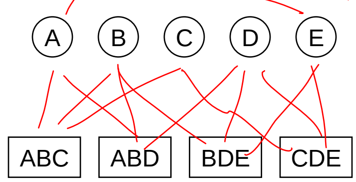
Sometimes important to compute in log space

$\log \delta_{i\to j}(X_j) =$ log

Loopy BP in Factor graphs

- From node *i* to factor *j*:
  - *F*(i) factors whose scope includes X_i

$$\delta_{i\to j}(X_i) \propto \prod_{k\in\mathcal{F}(i)-j} \delta_{k\to i}(X_i)$$

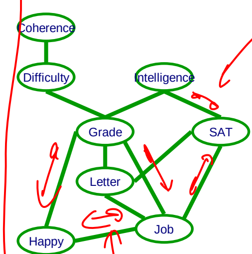


- From factor *j* to node *i*:
  - Scope[ϕ_j] = $\mathbf{Y}\cup\{X_i\}$

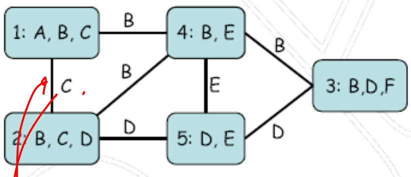
$$\delta_{j\to i}(X_i) \propto \sum_{\mathbf{y}} \phi_j(X_i,\mathbf{y}) \prod_{X_k\in \text{Scope}[\phi_j]-X_i} \delta_{k\to j}(x_k)$$

- Belief:
  - Node: $P(X_i) \approx b_i(x_i) \propto \prod_{\phi_j, X_i\in \text{Scope}[\phi_j]} \delta_{\phi_j\to X_i}(X_i)$
  - Factor: $P(Y) \approx b_{\phi_j}(Y) \propto \phi_j(Y) \prod_{X_i\in Y} \delta_{X_i\to\phi_j}(X_i)$

Loopy BP v. Clique trees: Two ends of a spectrum

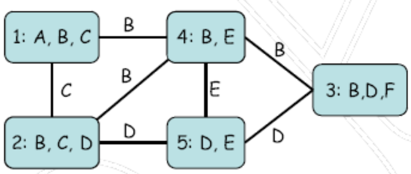


Generalize cluster graph



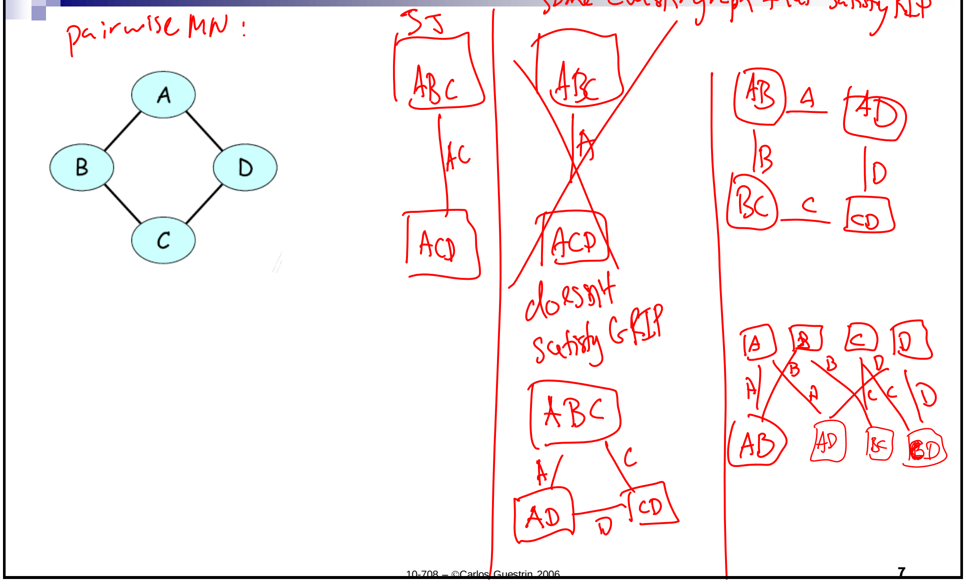
- **Generalized cluster graph**: For set of factors *F*
  - Undirected graph
  - Each node i associated with a cluster $\mathbf{C}_i$
  - *Family preserving*: for each factor $f_j \in F$, $\exists$ node i such that scope[f_j] $\subseteq \mathbf{C}_i$
  - Each edge i – j is associated with a set of variables $\mathbf{S}_{ij} \subseteq \mathbf{C}_i \cap \mathbf{C}_j$

Running intersection property

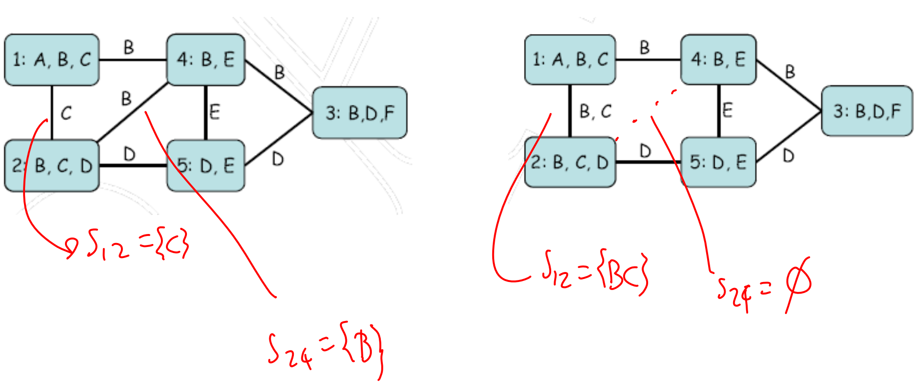


- **(Generalized) Running intersection property (RIP)**
  - Cluster graph satisfies RIP if whenever $X \in \mathbf{C}_i$ and $X \in \mathbf{C}_j$ then $\exists$ *one and only one path* from $\mathbf{C}_i$ to $\mathbf{C}_j$ where $X \in \mathbf{S}_{uv}$ for every edge (u,v) in the path

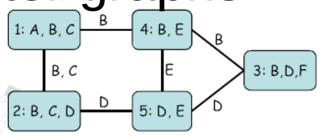
Examples of cluster graphs



Two cluster graph satisfying RIP with different edge sets



Generalized BP on cluster graphs satisfying RIP



- Initialization:
  - Assign each factor ϕ to a clique $\alpha(\phi)$, Scope$[\phi]\subseteq \mathbf{C}_{\alpha(\phi)}$
  - Initialize cliques: $\psi_i^0(\mathbf{C}_i) \propto \prod_{\phi:\alpha(\phi)=i} \phi$
  - Initialize messages: $\delta_{j\to i} = 1$
- While not converged, send messages:

$$\delta_{i\to j}(\mathbf{S}_{ij}) \propto \sum_{\mathbf{C}_i - \mathbf{S}_{ij}} \psi_i^0(\mathbf{C}_i) \prod_{k\in\mathcal{N}(i)-j} \delta_{k\to i}(\mathbf{S}_{ik})$$

- Belief:

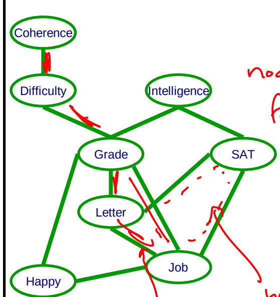
clique potential; all incoming messages from other nodes; normalize messages; marginalize all vars not in edge

or $P(C_i|e)$

$P(C_i) \approx b_i(C_i) \propto \psi_i^0(C_i) \prod_{k\in N(i)} \delta_{k\to i}(S_{ik})$

Cluster graph for (basic) Loopy BP

especial case of cluster graph w. GRIP

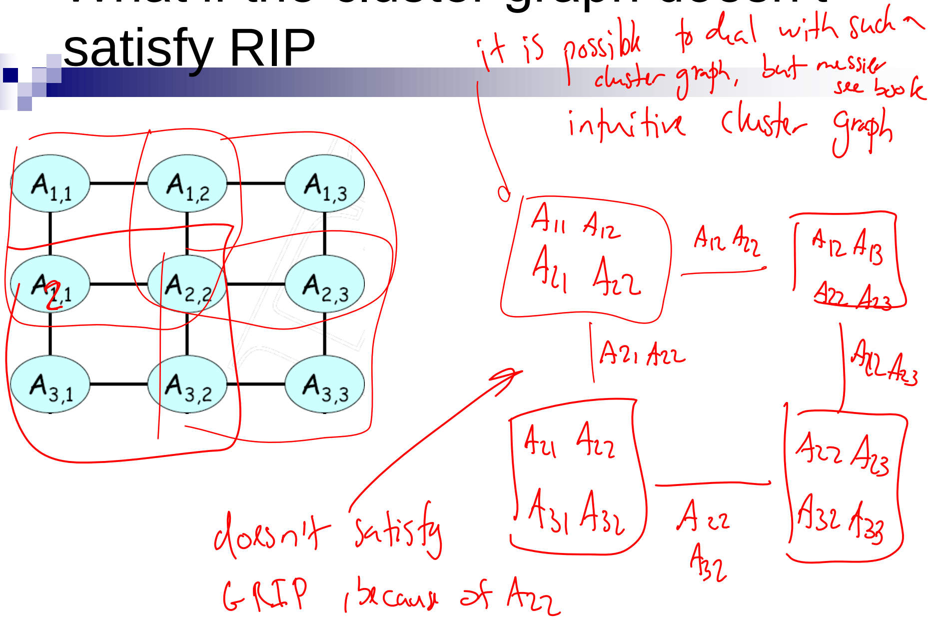


node per var; node per factor; triple factor

because each var forms a "tree"

message are about L or J independently; share L,J; think XOR; no joint message about L & J

What if the cluster graph doesn't satisfy RIP



Region graphs to the rescue

- Can address generalized cluster graphs that don't satisfy RIP using *region graphs*:
  - Book: 10.3
- Example in your homework! ☺

Revisiting Mean-Fields

lower bound

$\ln Z = F[P_{\mathcal{F}}, Q] + D(Q||P_{\mathcal{F}}) \quad F[P_{\mathcal{F}}, Q] = \sum_{\phi \in \mathcal{F}} E_Q[\ln \phi] + H_Q(\mathcal{X})$

- Choice of Q: *e.g.* $Q(X) = \prod_i Q_i(x_i)$
- Optimization problem: *= equal*

$\max_Q \sum_{\phi \in F} E_Q[\ln \phi] + \sum_i H_{Q_i}(X_i)$

$Q_i(x_i) \geq 0$

$\sum_{x_i} Q_i(x_i) = 1$

$Q \approx P$ intuitively as approx to $H_P(X)$

$$\max_Q F[P_{\mathcal{F}}, Q] = \sum_{\phi \in \mathcal{F}} E_Q[\ln \phi] + \sum_j H_{Q_j}(X_j), \quad \forall i, \sum_{x_i} Q_i(x_i) = 1$$

Announcements

- Recitation tomorrow
- HW5 out soon

you should go!!

- Will not cover relational models this semester
  - Instead, recommend Pedro Domingos' tutorial on Markov Logic
    - Markov logic is one example of a relational probabilistic model
    - November 14th from 1:00 pm to 3:30 pm in Wean 4623

Interpretation of energy functional

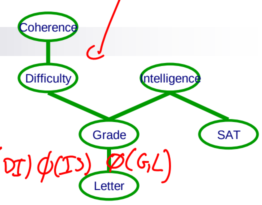
- Energy functional: $F[P_{\mathcal{F}}, Q] = \sum_{\phi \in \mathcal{F}} E_Q[\ln \phi] + H_Q(\mathcal{X})$
- ~~Exact if P=Q~~: $\ln Z = F[P_{\mathcal{F}}, Q] + D(Q||P_{\mathcal{F}})$ (constant)
- View problem as an approximation of entropy term:

$$H_Q(X) \approx H_P(X)$$

$$F[P_F, Q] = \sum_{\phi \in F} E_Q[\ln \phi] + H_Q(X) \approx \sum_{\phi \in F} E_Q[\ln \phi] + H_P(X)$$

Entropy of a tree distribution

P



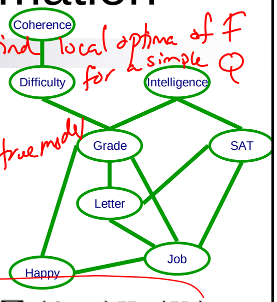
- Entropy term: $H_P(X)$
- Joint distribution: $P(X) = \frac{1}{Z}\phi(CD)\,\phi(DG)\,\phi(GI)\,\phi(IS)\,\phi(GL)$
- Decomposing entropy term:

$$H(X) = H(CD) + H(DG) + H(GI) + H(IS) + H(GL) - H(D) - 2H(G) - H(I)$$

for any tree MN

- More generally: $H_P(\mathbf{X}) = \sum_{(i,j) \in E} H(X_i, X_j) - \sum_i (d_i - 1) H(X_i)$
  - d_i number neighbors of X_i

Loopy BP & Bethe approximation



- Energy functional: $F[P_{\mathcal{F}}, Q] = \sum_{\phi \in \mathcal{F}} E_Q[\ln \phi] + H_Q(\mathcal{X})$

  Variational methods try to find local optima of F for a simple Q — *true model*

- **Bethe approximation of Free Energy**:
  - use entropy for trees, but loopy graphs: *≈ H_P(X)*

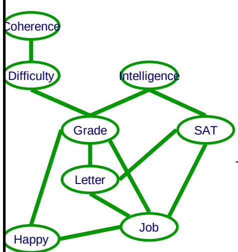
equation for

$$\tilde{F}[P_{\mathcal{F}}, Q] = \sum_{(i,j)\in E} E_{b_{ij}}[\ln \phi_{ij}] + \sum_{(i,j)\in E} H_{b_{ij}}(X_i, X_j) - \sum_i (d_i - 1) H_{b_i}(X_i)$$

all edges in loopy graph for p — *tree entropy eqn. for a loopy graph*

- **Theorem**: If Loopy BP converges, resulting b_{ij} & b_i are stationary point (usually local maxima) of Bethe Free energy! *belief*

GBP & Kikuchi approximation



- Exact Free energy: Junction Tree
  $F[P_{\mathcal{F}}, Q] = \sum_{(i,j)\in E} E_{\pi_{ij}}[\ln \phi_{ij}] + \sum_i H_{\pi_{\mathbf{C}_i}}(\mathbf{C}_i) - \sum_{(i,j)\in T} H_{\pi_{\mathbf{S}_{ij}}}(\mathbf{S}_{ij})$
- Bethe Free energy:
  $\tilde{F}[P_{\mathcal{F}}, Q] = \sum_{(i,j)\in E} E_{b_{ij}}[\ln \phi_{ij}] + \sum_{(i,j)\in E} H_{b_{ij}}(X_i, X_j) - \sum_i (d_i - 1) H_{b_i}(X_i)$
- Kikuchi approximation: Generalized cluster graph
  - spectrum from Bethe to exact
- **Theorem**: If GBP converges, resulting b_{C_i} are stationary point (usually local maxima) of Kikuchi Free energy!

What you need to know about GBP

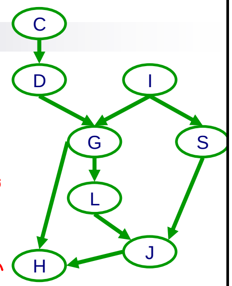
- Spectrum between Loopy BP & Junction Trees:
  - More computation, but typically better answers
- If satisfies RIP, equations are very simple
- General setting, slightly trickier equations, but not hard
- Relates to variational methods: Corresponds to local optima of approximate version of energy functional

Readings:
K&F: ~~10.2, 10.3~~

Parameter learning in Markov nets

Graphical Models – 10708
Carlos Guestrin
Carnegie Mellon University
November 12th, 2008

Learning Parameters of a BN



- Log likelihood decomposes:

$$\ell(\mathcal{D}:\theta) = \log P(\mathcal{D} \mid \theta) = m \sum_i \sum_{x_i, \mathbf{Pa}_{x_i}} \hat{P}(x_i, \mathbf{Pa}_{x_i}) \log P(x_i \mid \mathbf{Pa}_{x_i})$$

parameters I want to learn

- Learn each CPT independently
- Use counts

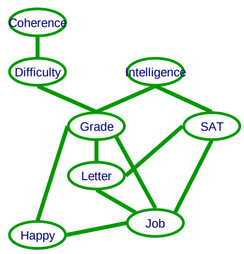
$$\hat{P}(\mathbf{u}) = \frac{\text{Count}(\mathbf{U}=\mathbf{u})}{m}$$

MLE:

$$P(X_i = x_i \mid Pa_{X_i} = u) \overset{MLE}{=} \frac{Count(X_i = x_i, Pa_{X_i} = u)}{Count(Pa_{X_i} = u)}$$

Log Likelihood for MN

$$\log Z_\theta = \log \sum_x \prod_{ij} \phi_{ij}(x_i, x_j \mid \theta_{ij})$$



- Log likelihood of the data:

$$\ell(D:\theta) = \log P(D \mid \theta) \overset{iid}{=} \sum_k \log P(x^{(k)} \mid \theta)$$

$$= \sum_k \log \frac{1}{Z} \prod_{ij} \phi_{ij}(x_i^{(k)}, x_j^{(k)} \mid \theta)$$

$$= \sum_k \sum_{ij} \log \phi_{ij}(x_i^{(k)}, x_j^{(k)}) - \underbrace{\sum_{k=1}^{m} \log Z}_{m \log Z}$$

$$= \sum_{ij} \sum_{x_i x_j} Count(X_i = x_i, X_j = x_j) \log \phi_{ij}(X_i = x_i, X_j = x_j) - m \log Z$$

$$= m \sum_{ij} \sum_{x_i x_j} \hat{P}(X_i = x_i, X_j = x_j) \log \phi_{ij}(X_i = x_i, X_j = x_j \mid \theta_{ij}) - m \log Z_\theta$$

decomposes nicely over factors just like BN

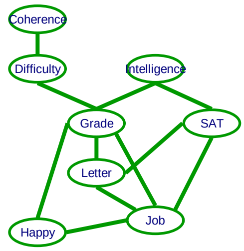
doesn't decompose

Log Likelihood doesn't decompose for MNs

$$\hat{P}(\mathbf{u}) = \frac{\text{Count}(\mathbf{U} = \mathbf{u})}{m}$$

- Log likelihood:

$$\ell(\mathcal{D} : \theta) = \log P(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \log \psi_i(\mathbf{c}_i) - m \log Z$$

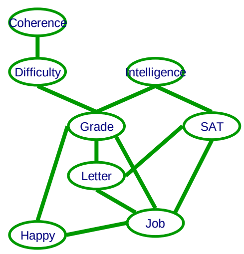


- A convex problem
  - Can find global optimum!!
- Term log Z doesn't decompose!!

Derivative of Log Likelihood for MNs

$$\hat{P}(\mathbf{u}) = \frac{\text{Count}(\mathbf{U} = \mathbf{u})}{m}$$

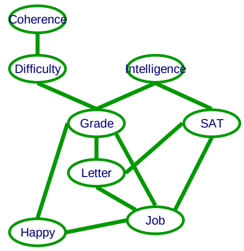
$$\ell(\mathcal{D} : \theta) = \log P(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \log \psi_i(\mathbf{c}_i) - m \log Z$$



Derivative of Log Likelihood for MNs 2

$$\hat{P}(\mathbf{u}) = \frac{\text{Count}(\mathbf{U}=\mathbf{u})}{m}$$

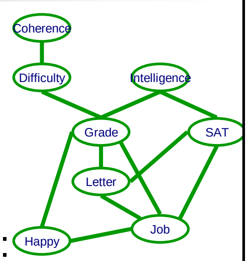
$$\ell(\mathcal{D}:\theta) = \log P(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \log \psi_i(\mathbf{c}_i) - m \log Z$$



Derivative of Log Likelihood for MNs

$$\hat{P}(\mathbf{u}) = \frac{\text{Count}(\mathbf{U}=\mathbf{u})}{m}$$

$$\ell(\mathcal{D}:\theta) = \log P(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \log \psi_i(\mathbf{c}_i) - m \log Z$$



- Derivative:

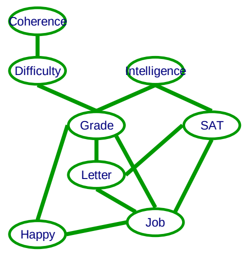
$$\frac{\partial \ell}{\partial \psi_i(\mathbf{c}_i)} = \frac{m\hat{P}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} - \frac{mP_{\mathcal{F}}^{\psi}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)}$$

- Computing derivative requires inference:
- Can optimize using gradient ascent
  - Common approach
  - Conjugate gradient, Newton's method,…
- Let's also look at a simpler solution

Iterative Proportional Fitting (IPF)

$$\hat{P}(\mathbf{u}) = \frac{\text{Count}(\mathbf{U} = \mathbf{u})}{m}$$

$$\frac{\partial \ell}{\partial \psi_i(\mathbf{c}_i)} = \frac{m\hat{P}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} - \frac{mP^{\psi}_{\mathcal{F}}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)}$$



- Setting derivative to zero:
- Fixed point equation:
- Iterate and converge to optimal parameters
  - Each iteration, must compute:

What you need to know about learning MN parameters?

- BN parameter learning easy
- MN parameter learning doesn't decompose!
- Learning requires inference!
- Apply gradient ascent or IPF iterations to obtain optimal parameters