Readings:
- K&F: 11.3, 11.5
- Yedidia et al. paper from the class website
- Chapter 9 – Jordan
- K&F: 16.2

Unifying Variational and GBP
Learning Parameters of MNs
EM for BNs

Graphical Models – 10708
Carlos Guestrin
Carnegie Mellon University

November 15th, 2006

Revisiting Mean-Fields

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

constant

- Choice of Q: $Q(X) = \prod_i Q_i(x_i)$
- Optimization problem:

decompose

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

$$\text{s.t.} \quad \sum_{x_i} Q_i(x_i) = 1$$

$$Q_i(x_i) \geq 0$$

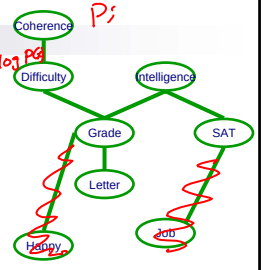
mean-fields solution is a stationary point...

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

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: $\ln Z = F[P_{\mathcal{F}}, Q] + D(Q||P_{\mathcal{F}})$
- View problem as an approximation of entropy term:

Entropy of a tree distribution



- Entropy term: $H_P(X) = -E_P[\log P(X)] = -\sum_x P(x) \log P(x)$
- Joint distribution: $P(X) = \frac{1}{Z} \prod_{ij} \psi_{ij}(X_i, X_j)$
- Decomposing entropy term:

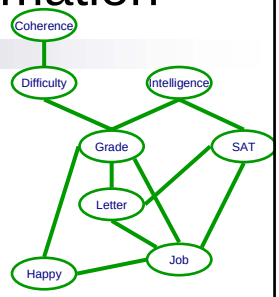
$$P(X) = \frac{P(CD) \cdot P(DG) \cdot P(GI) \cdot P(IS)\, P(GL)}{P(D)\, P(G)\, P(G) \cdot P(I)}$$

$$= \sum_{\text{cliques}} H(C_i) - \sum_{\text{seps}} H(S_{ij})$$

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

- 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

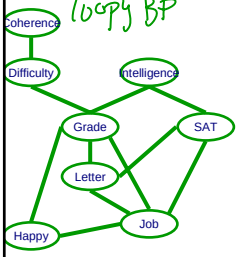


- Energy functional: $F[P_{\mathcal{F}}, Q] = \sum_{\phi \in \mathcal{F}} E_Q[\ln \phi] + H_Q(\mathcal{X})$
- **Bethe approximation of Free Energy**:
  - use entropy for trees, but loopy graphs:

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

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

GBP & Kikuchi approximation



- Exact Free energy: Junction Tree

$$F[P_{\mathcal{F}}, Q] = \sum_{(i,j)\in E} E_{\pi_{ij}}[\ln \phi_{ij}] + \sum_i H_{\pi_{\mathbf{C}_i}}(\mathbf{C}_i) - \sum_{(i,j)\in \mathcal{T}} H_{\pi_{\mathbf{S}_{ij}}}(\mathbf{S}_{ij})$$

- Bethe Free energy:

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

- Kikuchi approximation: Generalized cluster graph
  - spectrum from Bethe to exact
  - entropy terms weighted by counting numbers
  - see Yedidia et al.
- **Theorem**: If GBP converges, resulting $\pi_{\mathbf{C}_i}$ 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

Announcements

- Tomorrow's recitation
  - Khalid on learning Markov Networks

Learning Parameters of a BN

- Log likelihood decomposes:

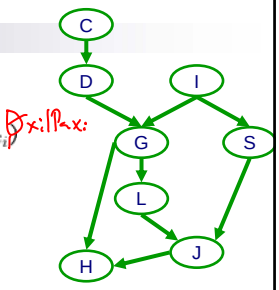
$$\ell(\mathcal{D}:\theta) = \log P(\mathcal{D} \mid \theta) = m \sum_i \sum_{x_i, \mathbf{Pa}_{x_i}} \hat{P}(x_i, \mathbf{Pa}_{x_i}) \log P(x_i \mid \mathbf{Pa}_{x_i})$$

$\theta_{x_i|\mathbf{Pa}_{x_i}}$

empirical dist

$$\theta_{X_i=t|\mathbf{Pa}_{X_i}=u} = \frac{\text{Count}(X_i=t, \mathbf{Pa}_{X_i}=u)}{\text{Count}(\mathbf{Pa}_{X_i}=u)}$$

- Learn each CPT independently
- Use counts



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

Log Likelihood for MN

$$Z = \sum_x \prod_i \psi_i(x)$$

- Log likelihood of the data:

i.i.d. m data points

$$\ell(D:\theta) = \log P(D|\theta) = \log \prod_{j \in D} \frac{1}{Z} \prod_i \psi_i(c_i = c_i^{(j)})$$

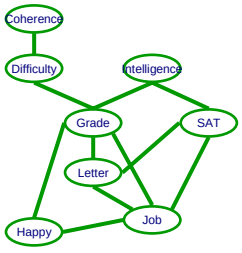
$$= \sum_{j \in D} \sum_i \log \psi_i(c_i = c_i^{(j)}) - \sum_{j \in D} \log Z$$

$m \log Z$

$$= \sum_i \sum_{j \in D} \log \psi_i(c_i = c_i^{(j)}) - m \log Z$$

$$= \sum_i \sum_{c_i} \text{Count}(c_i = c_i) \log \psi_i(c_i = c_i) - m \log Z$$

$$= m \sum_i \sum_{c_i} \hat{P}(c_i = c_i) \log \psi_i(c_i = c_i) - m \log Z(\psi_1 \dots \psi_n)$$

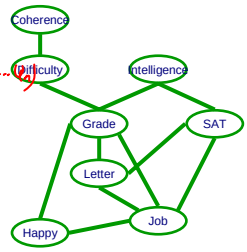


Log Likelihood doesn't decompose for MNs

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

- Log likelihood:

$$\ell(\mathcal{D} : \theta) = \log P(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \log \psi_i(\mathbf{c}_i) - m \log Z(\psi_1, \ldots, \psi_k)$$



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

Derivative of Log Likelihood for MNs

$$\frac{\partial}{\partial x} \log(x) = \frac{1}{x} \quad ; \quad \frac{\partial}{\partial x} \log Z(x) = \frac{Z'(x)}{Z(x)}$$

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

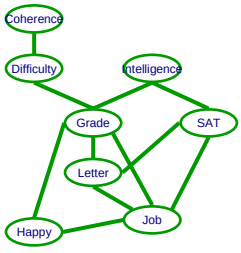
$$\ell(\mathcal{D} : \theta) = \log P(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \log \psi_i(\mathbf{c}_i) - m \log Z$$

$$\frac{\partial \ell}{\partial \psi_i(c_i)} = \frac{\partial}{\partial \psi_i(c_i)} \left[m \sum_i \sum_{c_i} \hat{P}(c_i) \log \psi_i(c_i) \right] - \frac{\partial}{\partial \psi_i(c_i)} \log Z$$

$$\hat{P}(c_i) \frac{\partial}{\partial \psi_i(c_i)} \log \psi_i(c_i) = \frac{\hat{P}(c_i)}{\psi_i(c_i)}$$

$$Z = \sum_{\mathbf{x}} \prod_i \psi_i(c_i)$$

$$\frac{\partial}{\partial \psi_i(c_i)} \log Z = \frac{\frac{\partial Z}{\partial \psi_i(c_i)}}{Z} \quad \ldots$$



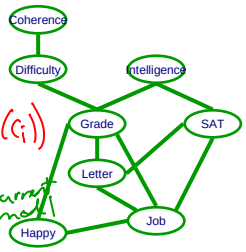
Derivative of Log Likelihood for MNs

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

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

- Derivative:

$$\frac{\partial \ell}{\partial \psi_i(\mathbf{c}_i)} = \frac{m\hat{P}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} - \frac{mP_{\mathcal{F}}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} = \frac{m}{\psi_i(c_i)} \left(\hat{P}(c_i) - P_{\mathcal{F}}(c_i)\right)$$

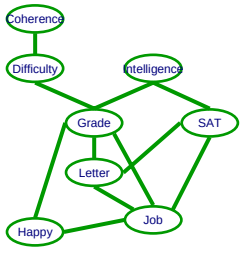


- Setting derivative to zero
- Can optimize using gradient ascent
  - Let's look at a simpler solution

Iterative Proportional Fitting (IPF)

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

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



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

Log-linear Markov network (most common representation)

- **Feature** is some function $\phi[\mathbf{D}]$ for some subset of variables **D**
  - e.g., indicator function
- **Log-linear model** over a Markov network *H*:
  - a set of features $\phi_1[\mathbf{D}_1],\ldots, \phi_k[\mathbf{D}_k]$
    - each $\mathbf{D}_i$ is a subset of a clique in *H*
    - two ϕ's can be over the same variables
  - a set of weights $w_1,\ldots,w_k$
    - usually learned from data
  - $P(X_1,\ldots,X_n) = \frac{1}{Z}\exp\left[\sum_{i=1}^{k} w_i\phi_i(\mathbf{D}_i)\right]$

$\psi(G,I,D) = \begin{cases} 1, & \text{if } G=t, I=t, D=f \\ 0, & \text{otherwise} \end{cases}$ ...

learn these

in Jordan Chapter 9: $w_i = \theta_i$, $\phi_i = f_i$

Learning params for log linear models 1 – Generalized Iterative Scaling

$$P(X_1,\ldots,X_n) = \frac{1}{Z}\exp\left[\sum_{i=1}^{k} w_i\phi_i(\mathbf{D}_i)\right]$$

$\sum_x \hat{P}(x)\,\phi_i(x)$ = expected value of ϕ_i wrt. data

e.g., $\frac{1}{m}\sum_{j\in D}\phi_i(x^{(j)})$

- IPF generalizes easily if:
  - $\phi_i(\mathbf{x}) \geq 0$
  - $\sum_i \phi_i(\mathbf{x}) = 1$
- Update rule: $w_i^{(t+1)} = w_i^{(t)} + \log\left(\frac{\sum_{\mathbf{x}} \hat{P}(\mathbf{x})\phi_i(\mathbf{x})}{\sum_{\mathbf{x}} P^{(t)}(\mathbf{x})\phi_i(\mathbf{x})}\right)$

$\sum_x P^{(t)}(x)\,\phi_i(x) = E_{P^{(t)}}[\phi_i]$ = expected v. of ϕ_i wrt. current model

- Must compute: $\text{Scope}[\phi_i] = D_i$

if $\phi_i[D_i]$: inference $P_F^{(t)}(D_i)$

$$E_{P_F^{(t)}}[\phi_i] = \sum_{d_i} P_F^{(t)}(d_i)\,\phi_i(d_i)$$

$\sum \phi_i(x) = 1 \quad \forall x$ not easy...

- If conditions violated, equations are not so simple…
  - c.f., Improved Iterative Scaling [Berger '97]

Learning params for log linear models 2 – Gradient Ascent

$$P(X_1,\ldots,X_n) = \frac{1}{Z}\exp\left[\sum_{i=1}^{k} w_i\phi_i(\mathbf{D}_i)\right]$$

- Log-likelihood of data:

$$\ell(D:w) = \log P(D|w) = \log \prod_{j\in D} \frac{1}{Z} e^{\sum_{i=1}^{k} w_i\phi_i(x^{(j)})}$$
$$= \log e^{\sum_j \sum_i w_i\phi_i(x^{(j)})} - \log Z$$
$$= \sum_{j\in D}\sum_i w_i\phi_i(x_i^{(j)}) - m\log Z$$

- Compute derivative & optimize
  - usually with conjugate gradient ascent
  - You will do an example in your homework! ☺

w_i's are free!!

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
  - applies to both tabular representations and log-linear models

Thus far, fully supervised learning

- We have assumed fully supervised learning:

- Many real problems have missing data:

The general learning problem with missing data

- Marginal likelihood – **x** is observed, **z** is missing:

$$\begin{aligned}\ell(\theta : \mathcal{D}) &= \log \prod_{j=1}^{m} P(\mathbf{x}_j \mid \theta) \\ &= \sum_{j=1}^{m} \log P(\mathbf{x}_j \mid \theta) \\ &= \sum_{j=1}^{m} \log \sum_{\mathbf{z}} P(\mathbf{x}_j, \mathbf{z} \mid \theta)\end{aligned}$$

E-step

- **x** is observed, **z** is missing
- Compute probability of missing data given current choice of θ
  - Q(**z**|$\mathbf{x}_j$) for each $\mathbf{x}_j$
    - e.g., probability computed during classification step
    - corresponds to "classification step" in K-means

$$Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \;=\; P(\mathbf{z} \mid \mathbf{x}_j, \theta^{(t)})$$

Jensen's inequality

$$\ell(\theta : \mathcal{D}) \;=\; \sum_{j=1}^{m} \log \sum_{\mathbf{z}} P(\mathbf{z} \mid \mathbf{x}_j) P(\mathbf{x}_j \mid \theta)$$

- **Theorem**: log $\sum_{\mathbf{z}}$ P(**z**) f(**z**) $\geq \sum_{\mathbf{z}}$ P(**z**) log f(**z**)

Applying Jensen's inequality

- Use: $\log \sum_{\mathbf{z}} P(\mathbf{z})\, f(\mathbf{z}) \geq \sum_{\mathbf{z}} P(\mathbf{z}) \log f(\mathbf{z})$

$$\ell(\theta^{(t)} : \mathcal{D}) = \sum_{j=1}^{m} \log \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \frac{P(\mathbf{z}, \mathbf{x}_j \mid \theta^{(t)})}{Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j)}$$

The M-step maximizes lower bound on weighted data

- Lower bound from Jensen's:

$$\ell(\theta^{(t)} : \mathcal{D}) \geq \sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta^{(t)}) + H(Q^{(t+1)})$$

- Corresponds to weighted dataset:
  - $\langle \mathbf{x}_1, \mathbf{z}=1 \rangle$ with weight $Q^{(t+1)}(\mathbf{z}=1|\mathbf{x}_1)$
  - $\langle \mathbf{x}_1, \mathbf{z}=2 \rangle$ with weight $Q^{(t+1)}(\mathbf{z}=2|\mathbf{x}_1)$
  - $\langle \mathbf{x}_1, \mathbf{z}=3 \rangle$ with weight $Q^{(t+1)}(\mathbf{z}=3|\mathbf{x}_1)$
  - $\langle \mathbf{x}_2, \mathbf{z}=1 \rangle$ with weight $Q^{(t+1)}(\mathbf{z}=1|\mathbf{x}_2)$
  - $\langle \mathbf{x}_2, \mathbf{z}=2 \rangle$ with weight $Q^{(t+1)}(\mathbf{z}=2|\mathbf{x}_2)$
  - $\langle \mathbf{x}_2, \mathbf{z}=3 \rangle$ with weight $Q^{(t+1)}(\mathbf{z}=3|\mathbf{x}_2)$
  - …

The M-step

$$\ell(\theta^{(t)} : \mathcal{D}) \geq \sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta^{(t)}) + H(Q^{(t+1)})$$

- Maximization step:

$$\theta^{(t+1)} \leftarrow \arg\max_{\theta} \sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta)$$

- Use expected counts instead of counts:
  - If learning requires Count(**x**,**z**)
  - Use $E_{Q(t+1)}$[Count(**x**,**z**)]

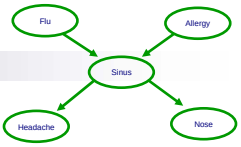
Convergence of EM

- Define potential function F(θ,Q):

$$\ell(\theta : \mathcal{D}) \geq F(\theta, Q) = \sum_{j=1}^{m} \sum_{\mathbf{z}} Q(\mathbf{z} \mid \mathbf{x}_j) \log \frac{P(\mathbf{z}, \mathbf{x}_j \mid \theta)}{Q(\mathbf{z} \mid \mathbf{x}_j)}$$

- EM corresponds to coordinate ascent on F
  - Thus, maximizes lower bound on marginal log likelihood
  - As seen in Machine Learning class last semester

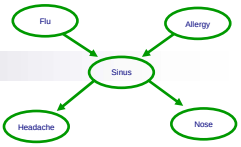
Data likelihood for BNs



- Given structure, log likelihood of fully observed data:

$$\log P(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G})$$

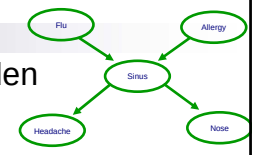
Marginal likelihood



- What if S is hidden?

$$\log P(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G})$$

Log likelihood for BNs with hidden data

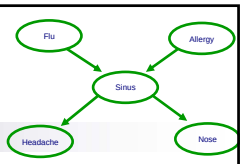


- Marginal likelihood – **O** is observed, **H** is hidden

$$\ell(\theta : \mathcal{D}) = \sum_{j=1}^{m} \log P(\mathbf{o}^{(j)} \mid \theta)$$

$$= \sum_{j=1}^{m} \log \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{o}^{(j)} \mid \theta)$$

E-step for BNs

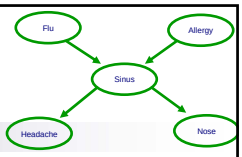


- E-step computes probability of hidden vars **h** given **o**

$$Q^{(t+1)}(\mathbf{x} \mid \mathbf{o}) = P(\mathbf{x} \mid \mathbf{o}, \theta^{(t)})$$

- Corresponds to inference in BN

The M-step for BNs

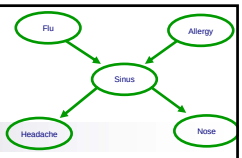


- Maximization step:

$$\theta^{(t+1)} \leftarrow \arg\max_{\theta} \sum_{\mathbf{x}} Q^{(t+1)}(\mathbf{h} \mid \mathbf{o}) \log P(\mathbf{h}, \mathbf{o} \mid \theta)$$

- Use expected counts instead of counts:
  - If learning requires Count(**h**,**o**)
  - Use $E_{Q(t+1)}$[Count(**h**,**o**)]

M-step for each CPT



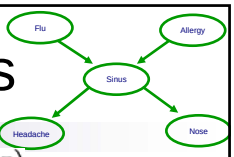
- M-step decomposes per CPT
  - Standard MLE:

$$P(X_i = x_i \mid \mathbf{Pa}_{X_i} = \mathbf{z}) = \frac{\text{Count}(X_i = x_i, \mathbf{Pa}_{X_i} = \mathbf{z})}{\text{Count}(\mathbf{Pa}_{X_i} = \mathbf{z})}$$

  - M-step uses expected counts:

$$P(X_i = x_i \mid \mathbf{Pa}_{X_i} = \mathbf{z}) = \frac{\text{ExCount}(X_i = x_i, \mathbf{Pa}_{X_i} = \mathbf{z})}{\text{ExCount}(\mathbf{Pa}_{X_i} = \mathbf{z})}$$

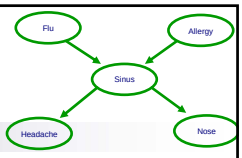
Computing expected counts



$$P(X_i = x_i \mid \mathbf{Pa}_{X_i} = \mathbf{z}) = \frac{\text{ExCount}(X_i = x_i, \mathbf{Pa}_{X_i} = \mathbf{z})}{\text{ExCount}(\mathbf{Pa}_{X_i} = \mathbf{z})}$$

- M-step requires expected counts:
  - For a set of vars **A**, must compute ExCount(**A**=**a**)
  - Some of **A** in example *j* will be observed
    - denote by $\mathbf{A_O} = \mathbf{a_O}^{(j)}$
  - Some of **A** will be hidden
    - denote by $\mathbf{A_H}$
- Use inference (E-step computes expected counts):
  - $\text{ExCount}^{(t+1)}(\mathbf{A_O} = \mathbf{a_O}^{(j)}, \mathbf{A_H} = \mathbf{a_H}) \leftarrow P(\mathbf{A_H} = \mathbf{a_H} \mid \mathbf{A_O} = \mathbf{a_O}^{(j)}, \boldsymbol{\theta}^{(t)})$

Data need not be hidden in the same way



- When data is fully observed
  - A data point is
- When data is partially observed
  - A data point is
- But unobserved variables can be different for different data points
  - e.g.,
- Same framework, just change definition of expected counts
  - $\text{ExCount}^{(t+1)}(\mathbf{A_O} = \mathbf{a_O}^{(j)}, \mathbf{A_H} = \mathbf{a_H}) \leftarrow P(\mathbf{A_H} = \mathbf{a_H} \mid \mathbf{A_O} = \mathbf{a_O}^{(j)}, \boldsymbol{\theta}^{(t)})$

What you need to know

- EM for Bayes Nets
- E-step: inference computes expected counts
  - Only need expected counts over X_i and $\mathbf{Pa}_{xi}$
- M-step: expected counts used to estimate parameters
- Hidden variables can change per datapoint

- Use labeled and unlabeled data $\rightarrow$ some data points are complete, some include hidden variables