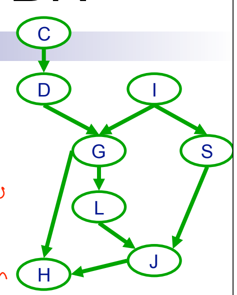
Readings:
K&F: 19.1, 19.2, 19.3, 19.4

Parameter learning in Markov nets

Graphical Models – 10708
Carlos Guestrin
Carnegie Mellon University
November 17th, 2008

Learning Parameters of a BN



- Log likelihood decomposes:

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

parameters I want to learn

- Learn each CPT independently
- Use counts

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

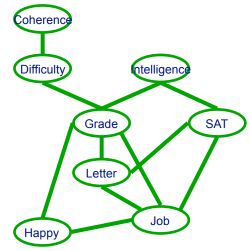
MLE:

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

Log Likelihood for MN

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

- Log likelihood of the data:



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

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

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

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

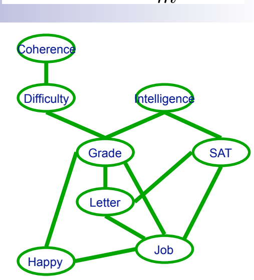
$= m \underbrace{\sum_{ij} \sum_{x_i x_j} \hat{P}(X_i = x_i, X_j = x_j) \log \phi_{ij}(X_i = x_i, X_j = x_j | \theta_{ij})}_{\text{decomposes nicely over factors just like BN}} - \underbrace{m \log Z_\theta}_{\text{doesn't decompose}}$

Log Likelihood doesn't decompose for MNs

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

- Log likelihood:

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

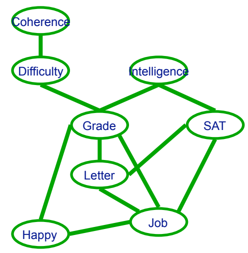


- A concave problem
  - Can find global optimum!!
- Term log Z doesn't decompose!!

Derivative of Log Likelihood for MNs

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

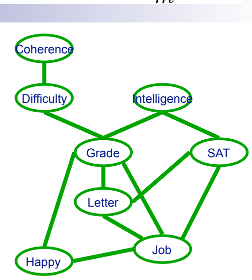
$$\ell(\mathcal{D}:\theta) = \log P(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \log \psi_i(\mathbf{c}_i) - m \log Z$$



Derivative of Log Likelihood for MNs 2

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

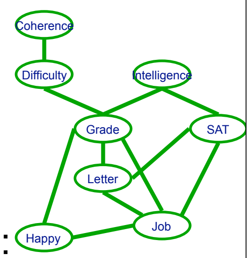
$$\ell(\mathcal{D}:\theta) = \log P(\mathcal{D} \mid \theta, \mathcal{G}) = m \sum_i \sum_{\mathbf{c}_i} \hat{P}(\mathbf{c}_i) \log \psi_i(\mathbf{c}_i) - m \log Z$$



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:

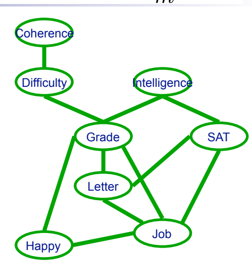
$$\frac{\partial \ell}{\partial \psi_i(\mathbf{c}_i)} = \frac{m\hat{P}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} - \frac{m P^{\psi}_{\mathcal{F}}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)}$$

- 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)

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

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



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

Log-linear Markov network (most common representation)

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

Learning params for log linear models – 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:

- Compute derivative & optimize
  - usually with conjugate gradient ascent

Derivative of log-likelihood 1 – log-linear models

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

$$\ell(\mathcal{D} : \mathbf{w}) = \log P(\mathcal{D} \mid \mathbf{w}, \mathcal{G}) = \sum_{j=1}^{m} \log \frac{1}{Z} \exp\left[\sum_{i=1}^{k} w_i \phi_i(\mathbf{d}_i^{(j)})\right]$$

Derivative of log-likelihood 2 – log-linear models

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

$$\frac{\partial \ell(\mathcal{D} : \mathbf{w})}{\partial w_i} = m \sum_{\mathbf{d}_i} \hat{P}(\mathbf{d}_i)\phi_i(\mathbf{d}_i) - m\frac{\partial \log Z}{\partial w_i}$$

Learning log-linear models with gradient ascent

- Gradient: $$\frac{\partial \ell(\mathcal{D} : \mathbf{w})}{\partial w_i} = m \sum_{\mathbf{d}_i} \hat{P}(\mathbf{d}_i)\phi_i(\mathbf{d}_i) - m \sum_{\mathbf{d}_i} P(\mathbf{d}_i \mid \mathbf{w})\phi_i(\mathbf{d}_i)$$
- Requires one inference computation per
- Theorem: **w** is maximum likelihood solution iff
  -
- Usually, must regularize
  - E.g., L_2 regularization on parameters

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

Readings:
K&F: 19.3.2

Conditional Random Fields

Graphical Models – 10708
Carlos Guestrin
Carnegie Mellon University
November 17th, 2008

Generative v. Discriminative classifiers – A review

- **Want to Learn**: h:**X** ↦ Y
  - **X** – features
  - Y – target classes
- **Bayes optimal classifier** – P(Y|**X**)
- **Generative classifier**, e.g., Naïve Bayes:
  - Assume some **functional form for P(X|Y), P(Y)**
  - Estimate parameters of P(**X**|Y), P(Y) directly from training data
  - Use Bayes rule to calculate P(Y|**X= x**)
  - This is a **'*generative*' model**
    - **Indirect** computation of P(Y|**X**) through Bayes rule
    - But, **can generate a sample of the data**, P(**X**) = $\sum_y$ P(y) P(**X**|y)
- **Discriminative classifiers**, e.g., Logistic Regression:
  - Assume some **functional form for P(**Y|**X)**
  - Estimate parameters of P(Y|**X**) directly from training data
  - This is the **'*discriminative*' model**
    - Directly learn P(Y|**X**)
    - But **cannot obtain a sample of the data**, because P(**X**) is not available

Log-linear CRFs (most common representation)

- **Graph *H*:** only over hidden vars $Y_1,..,Y_n$
  - **No assumptions about dependency on observed vars *X***
  - You must always observe all of **X**
- **Feature** is some function $\phi[\mathbf{D}]$ for some subset of variables **D**
  - e.g., indicator function,
- **Log-linear model** over a CRF *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
  - $P(Y_1, \ldots, Y_n \mid x) = \frac{1}{Z(x)} \exp\left[\sum_{i=1}^{k} w_i \phi_i(\mathbf{D}_i, x)\right]$

Learning params for log linear CRFs – Gradient Ascent

$$P(Y_1, \ldots, Y_n \mid x) = \frac{1}{Z(x)} \exp\left[\sum_{i=1}^{k} w_i \phi_i(\mathbf{D}_i, x)\right]$$

- Log-likelihood of data:

- Compute derivative & optimize
  - usually with conjugate gradient ascent

Learning log-linear CRFs with gradient ascent

- Gradient: $$\frac{\partial \ell(\mathcal{D}:\mathbf{w})}{\partial w_i} = \sum_{j=1}^{m}\left[\phi_i(\mathbf{d}_i^{(j)}, x^{(j)}) - \sum_{\mathbf{d}_i} P(\mathbf{d}_i \mid x^{(j)}, \mathbf{w})\phi_i(\mathbf{d}_i, x^{(j)})\right]$$

- Requires one inference computation per

- Usually, must regularize
  - E.g., L_2 regularization on parameters

What you need to know about CRFs

- Discriminative learning of graphical models
  - Fewer assumptions about distribution ➔ often performs better than "similar" MN
  - Gradient computation requires inference per datapoint ➔ Can be really slow!!

Readings: 18.1, 18.2

EM for BNs

Graphical Models – 10708
Carlos Guestrin
Carnegie Mellon University

November 17th, 2008

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:

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

E-step

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

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

Jensen's inequality

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

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

Applying Jensen's inequality

- Use: log $\sum_{\mathbf{z}}$ P(**z**) f(**z**) ≥ $\sum_{\mathbf{z}}$ P(**z**) log f(**z**)

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

The M-step maximizes lower bound on weighted data

- Lower bound from Jensen's:

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

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

The M-step

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

- Maximization step:

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

- 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):

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

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