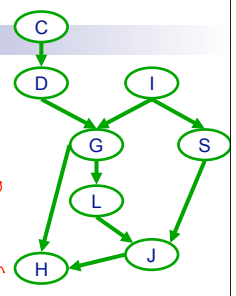
Readings:
K&F: 19.1, 19.2, 19.3, 19.4

Parameter learning in Markov nets

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

Learning Parameters of a BN



- Log likelihood decomposes:

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

parameters I want to learn

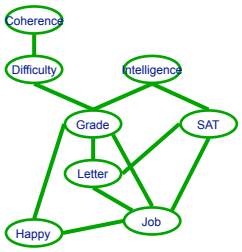
- Learn each CPT independently
- Use counts

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

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 likelihood of the data:



Log Likelihood doesn't decompose for MNs

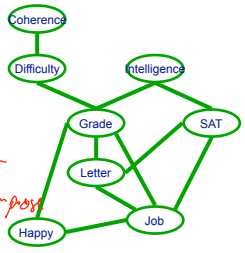
$$\hat{P}(\mathbf{u}) = \frac{\text{Count}(\mathbf{U} = \mathbf{u})}{m}$$

- Log likelihood:

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

decomposes nicely

doesn't decompose



- A concave problem
  - Can find global optimum!!

good news

- Term log Z doesn't decompose!!

bad news

Derivative of Log Likelihood for MNs

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

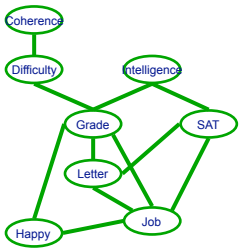
tabular

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

$$\frac{\partial \ell(\mathcal{D}:\theta)}{\partial \psi_i(c_i)} = m \hat{P}(c_i) \frac{\partial \log \psi_i(c_i)}{\partial \psi_i(c_i)} - m \frac{\partial \log Z}{\partial \psi_i(c_i)}$$

$$\frac{\partial \log \psi_i(c_i)}{\partial \psi_i(c_i)} = \frac{1}{\psi_i(c_i)}$$

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



Derivative of Log Likelihood for MNs 2

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

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

$$\frac{\partial \ell}{\partial \psi_i(c_i)} = m \frac{\hat{P}(c_i)}{\psi_i(c_i)} - m \frac{\partial \log Z}{\partial \psi_i(c_i)}$$

$$\frac{\partial \log Z}{\partial \psi_i(c_i)} = \frac{\partial Z}{\partial \psi_i(c_i)} = \left(\frac{1}{Z} \sum_{x \sim c_i} \prod_j \psi_j(c_j)\right) \frac{1}{\psi_i(c_i)}$$

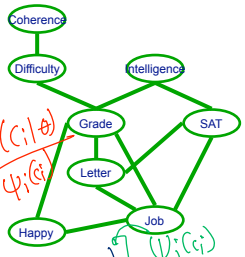
$$\frac{P(c_i \mid \theta)}{\psi_i(c_i)} = \left[\sum_{x \sim c_i} \prod_{j \neq i} \psi_j(c_j)\right] \frac{\psi_i(c_i)}{\psi_i(c_i)}$$

$$\frac{\partial Z}{\partial \psi_i(c_i)} = \frac{\partial}{\partial \psi_i(c_i)} \left(\sum_x \prod_j \psi_j(c_j)\right)$$

Z: ϕ if x not consistent with c_i

$$= \frac{\partial}{\partial \psi_i(c_i)} \sum_{x \sim c_i} \prod_j \psi_j(c_j)$$

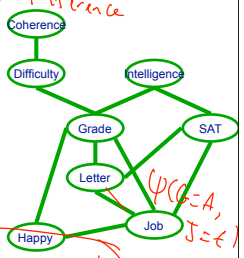
$x \sim c_i$: consistent with c_i



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:

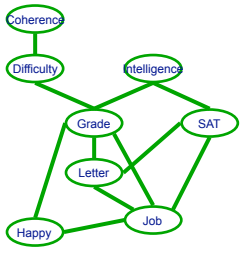
$$\frac{\partial \ell}{\partial \psi_i(\mathbf{c}_i)} = \frac{m\hat{P}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} - \frac{mP_{\mathcal{F}}^{\psi}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)}$$

- 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)

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

$$\frac{\partial \ell}{\partial \psi_i(\mathbf{c}_i)} = \frac{m\hat{P}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} - \frac{mP_{\mathcal{F}}^{\psi}(\mathbf{c}_i)}{\psi_i(\mathbf{c}_i)} = 0$$



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

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:

$$\ell(D:w) = \log \prod_{j=1}^{m} \frac{1}{Z}\exp\left[\sum_{i=1}^{k} w_i\phi_i(d_i^{(j)})\right]$$

$$= \sum_{j=1}^{m} \log \frac{1}{Z}\exp\left[\sum_{i=1}^{k} w_i\phi_i(d_i^{(j)})\right]$$

- Compute derivative & optimize
  - usually with conjugate gradient ascent

Derivative of log-likelihood 1 – log-linear models

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

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

$$\frac{\partial \ell}{\partial w_i} = \frac{\partial}{\partial w_i}\left[\sum_{j=1}^{m}\sum_{i=1}^{k} w_i \phi_i(d_i^{(j)}) - m \log Z\right]$$

$$= \sum_{j=1}^{m} \phi_i(d_i^{(j)}) - m \frac{\partial \log Z}{\partial w_i}$$

$m \sum_{d_i} \hat{P}(d_i)\, \phi_i(d_i)$ — will be expected feature value in model

$\hat{E}_{\mathcal{D}}[\phi_i]$ ← expected feature value in data

Derivative of log-likelihood 2 – log-linear models

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

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

$\frac{\partial}{\partial w_i} \log Z = \frac{\frac{\partial Z}{\partial w_i}}{Z}$

$\frac{\partial}{\partial w} e^{w\phi} = \phi e^{w\phi}$

$$\frac{\partial \log Z(w)}{\partial w_i} = \frac{1}{Z} \frac{\partial}{\partial w_i}\left[\sum_{x} e^{\sum_{j=1}^{k} w_j \phi_j[d_j]}\right]$$

$$= \frac{1}{Z} \sum_{x} \phi_i(d_i)\, e^{\sum_{j=1}^{k} w_j \phi_j[d_j]}$$

$$= \sum_{d_i} \phi_i(d_i)\, P(d_i \mid w) = E_{f,w}[\phi_i]$$

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
  - $E_{\hat{P}}[\phi_i] = E_{\mathbf{w}}[\phi_i]$
- 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

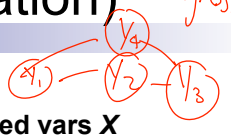
Graphical Models – 10708
Carlos Guestrin
Carnegie Mellon University
November 17th, 2008

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)

generalization of logistic regression



- **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 ϕ[**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

difference in MNs $P(X,Y)$ in CRFs $P(Y|X)$

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

- Log-likelihood of data:

$$\ell(D;w) = \log \prod_{j=1}^{m} P(y_1^{(j)},\ldots y_n^{(j)} \mid x^{(j)})$$

$$= \sum_{j=1}^{m} \log \frac{1}{Z(x^{(j)})} \exp\left[\sum_{i=1}^{k} w_i \phi_i(d_i^{(j)}, x^{(j)})\right]$$

- Compute derivative & optimize
  - usually with conjugate gradient ascent

Learning log-linear CRFs with gradient ascent

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

"from data part" "from model part"

$= \sum_{j=1}^{m} \phi_i(d_i^{(j)}, x^{(j)}) - \sum_{j=1}^{m} \sum_{d_i} P(d_i \mid x^{(j)}, w)\, \phi_i(d_i, x^{(j)})$

"expected value of feature"

probability depends on $x^{(j)}$

- Requires one inference computation per data point j, per feature i
- Usually, must regularize
  - E.g., L_2 regularization on parameters

especially important

What you need to know about CRFs

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