Parameter learning in Markov nets

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

Readings:

Learning Parameters of a BN

- Log likelihood decomposes:
  \[ \ell(D : \theta) = \log P(D \mid \theta) = m \sum_i \sum x_i P_{x_i} \log P(x_i \mid P_{x_i}) \]

- Learn each CPT independently
- Use counts

\[ P(x_{i} \mid P_{x_i} = u) = \frac{\text{MLE} \; \text{count} (x_i = x ; P_{x_i} = u)}{\text{MLE} \; \text{count} (P_{x_i} = u)} \]
Log Likelihood for MN

\[ \ell(D; \theta) = \log P(D | \theta) = \sum_{i,j} \log \Phi_{ij}(x_i, x_j | \theta) \]

Log likelihood of the data:

\[ \ell(D; \theta) = \log P(D | \theta) = \sum_{i,j} \log \frac{1}{Z} \prod_{x_i, x_j} \Phi_{ij}(x_i, x_j | \theta) \]

\[ = \sum_{i,j} \log \sum_{x_i, x_j} \Phi_{ij}(x_i, x_j | \theta) - \sum_{x_i, x_j} \log Z \]

\[ = \sum_{i,j} \log \text{count}(x_i = x, x_j = x_j) \log \Phi_{ij}(x_i = x, x_j = x_j) - \sum_{x_i, x_j} \log Z \]

\[ = m \sum_{i,j} \log \Phi_{ij}(x_i = x, x_j = x_j) \log \Phi_{ij}(x_i = x, x_j = x_j) - m \log Z \]

Log Likelihood doesn’t decompose for MNs

\[ P(u) = \frac{\text{Count}(U = u)}{m} \]

- A concave problem
  - Can find global optimum!!

- Term log Z doesn’t decompose!!
Derivative of Log Likelihood for MNs

\[ \ell(D; \theta) = \log P(D \mid \theta, \mathcal{G}) = m \sum_i \sum_j P(c_i) \log \psi_j(c_i) - m \log Z \]

\[ P(u) = \frac{\text{Count}(U = u)}{m} \]
Derivative of Log Likelihood for MNs

\[ \ell(D: \theta) = \log P(D \mid \theta, G) = m \sum_i \sum_{c_i} \hat{P}(c_i) \log \psi_i(c_i) - m \log Z \]

- Derivative:
  \[ \frac{\partial \ell}{\partial \psi_i(c_i)} = \frac{m \hat{P}(c_i)}{\psi_i(c_i)} - \frac{m P^\psi(c_i)}{\psi_i(c_i)} \]

- 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)

\[ P(u) = \frac{\text{Count}(U = u)}{m} \]

- Setting derivative to zero:
  \[ \frac{\partial \ell}{\partial \psi_i(c_i)} = \frac{m \hat{P}(c_i)}{\psi_i(c_i)} - \frac{m P^\psi(c_i)}{\psi_i(c_i)} \]

- Fixed point equation:

- Iterate and converge to optimal parameters
  - Each iteration, must compute:
Log-linear Markov network
(most common representation)

- **Feature** is some function $\phi[D]$ for some subset of variables $D$
  - e.g., indicator function
- **Log-linear model** over a Markov network $H$:
  - a set of features $\phi_1[D_1], \ldots, \phi_k[D_k]$
    - each $D_i$ is a subset of a clique in $H$
    - two $\phi$'s can be over the same variables
  - a set of weights $w_1, \ldots, w_k$
    - usually learned from data
  - $P(X_1, \ldots, X_n) = \frac{1}{Z} \exp \left[ \sum_{i=1}^{k} w_i \phi_i(D_i) \right]$ 

Learning params for log linear models – Gradient Ascent

- $P(X_1, \ldots, X_n) = \frac{1}{Z} \exp \left[ \sum_{i=1}^{k} w_i \phi_i(D_i) \right]$
- Log-likelihood of data:
  - Compute derivative & optimize
    - usually with conjugate gradient ascent
Derivative of log-likelihood 1 – log-linear models

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

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

Derivative of log-likelihood 2 – log-linear models

\[ \frac{\partial \ell(D : w)}{\partial w_i} = m \sum_{d_i} \hat{P}(d_i) \phi_i(d_i) - m \frac{\partial \log Z}{\partial w_i} \]
Learning log-linear models with gradient ascent

- Gradient: 
  \[ \frac{\partial \ell(D : w)}{\partial w_i} = m \sum_{d_i} \hat{P}(d_i) \phi_i(d_i) - m \sum_{d_i} P(d_i \mid w) \phi_i(d_i) \]

  - 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
Generative v. Discriminative classifiers – A review

- **Want to Learn**: h: X \(\rightarrow\) 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 $Y_1, \ldots, Y_n$
  - No assumptions about dependency on observed vars $X$
  - You must always observe all of $X$

- **Feature** is some function $\phi[D]$ for some subset of variables $D$
  - e.g., indicator function,

- **Log-linear model** over a CRF $H$:
  - a set of features $\phi_1[D_1], \ldots, \phi_k[D_k]$
    - each $D_i$ is a subset of a clique in $H$
    - two $\phi$'s can be over the same variables
  - a set of weights $w_1, \ldots, w_k$
    - usually learned from data

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

Learning params for log linear CRFs –

Gradient Ascent

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

- Log-likelihood of data:
  - Compute derivative & optimize
    - usually with conjugate gradient ascent
Learning log-linear CRFs with gradient ascent

Gradient:

\[ \frac{\partial \ell(D: w)}{\partial w_i} = \sum_{j=1}^{m} \left[ \phi_i(d_i^{(j)}, x^{(j)}) - \sum_{d_i} P(d_i | x^{(j)}, w) \phi_i(d_i, x^{(j)}) \right] \]

- 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!!
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 – \( \mathbf{x} \) is observed, \( \mathbf{z} \) is missing:

\[
\ell(\mathcal{D} : \theta) = \log \prod_{j=1}^{m} P(x^{(j)} | \theta) \\
= \sum_{j=1}^{m} \log P(x^{(j)} | \theta) \\
= \sum_{j=1}^{m} \log \sum_{z} P(z, x^{(j)} | \theta)
\]

E-step

\( \mathbf{x} \) is observed, \( \mathbf{z} \) is missing

Compute probability of missing data given current choice of \( \theta \)

- \( Q(\mathbf{z}|\mathbf{x}^{(i)}) \) for each \( \mathbf{x}^{(i)} \)
  - e.g., probability computed during classification step
  - corresponds to “classification step” in K-means

\[
Q^{(t+1)}(z | x^{(j)}) = P(z | x^{(j)}, \theta^{(t)})
\]
Jensen’s inequality

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

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

Applying Jensen’s inequality

- **Use:** \( \log \sum_{z} P(z) f(z) \geq \sum_{z} P(z) \log f(z) \)

\[ \ell(\mathcal{D} : \theta^{(t)}) = \sum_{j=1}^{m} \log \sum_{z} Q_{(t+1)}^{(t+1)}(z | x^{(j)}) \frac{P(z, x^{(j)} | \theta^{(t)})}{Q_{(t+1)}^{(t+1)}(z | x^{(j)})} \]
The M-step maximizes lower bound on weighted data

- Lower bound from Jensen’s:
  \[ \ell(D : \theta^{(t)}) \geq \sum_{j=1}^{m} \sum_{z} Q^{(t+1)}(z \mid x^{(j)}) \log P(z, x^{(j)} \mid \theta^{(t)}) + H(Q^{(t+1)}) \]

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

The M-step

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

- 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 \( \text{Count}(x, z) \)
  - Use \( E_{Q^{(t+1)}}[\text{Count}(x, z)] \)
Convergence of EM

- Define potential function $F(\theta, Q)$:

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

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