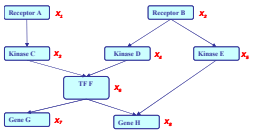
School of Computer Science
Carnegie Mellon

Graphical Models (III)

Learning



Eric Xing

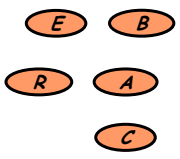
Carnegie Mellon University

June 4, 2007

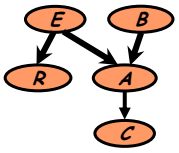
Learning Graphical Models

The goal:

Given set of independent samples (***assignments*** of random variables), find the ***best*** (the most likely?) graphical model (both the graph and the CPDs)



(B,E,A,C,R)=(T,F,F,T,F)
(B,E,A,C,R)=(T,F,T,T,F)
……..
(B,E,A,C,R)=(F,T,T,T,F)

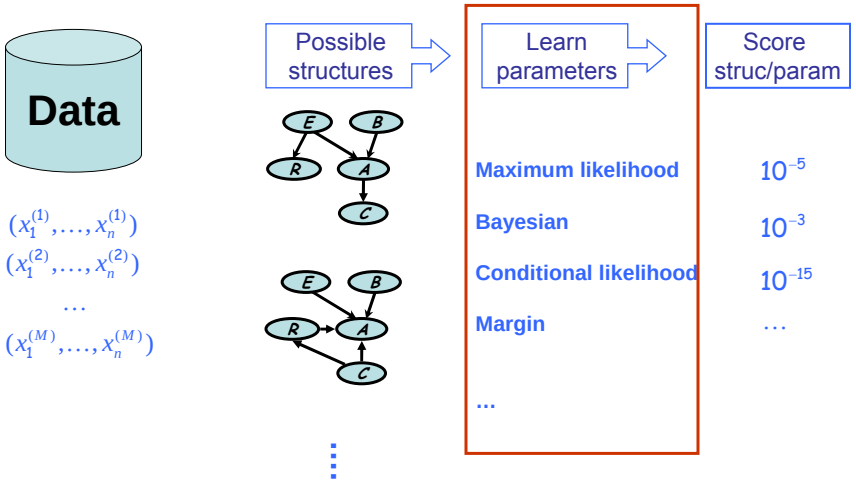


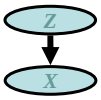
E	B	P(A | E,B)	
e	b	0.9	0.1
e	$\bar{b}$	0.2	0.8
$\bar{e}$	b	0.9	0.1
$\bar{e}$	$\bar{b}$	0.01	0.99

Learning Graphical Models

- Scenarios:
  - completely observed GMs
    - directed √
    - undirected √
  - partially observed GMs
    - directed √
    - undirected (an open research topic)
- Estimation principles:
  - Maximal likelihood estimation (MLE) √
  - Bayesian estimation
  - Maximal conditional likelihood
  - Maximal "Margin"
- We use **learning** as a name for the process of estimating the parameters, and in some cases, the topology of the network, from data.

Score-based approach



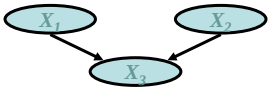


ML Parameter Est. for completely observed GMs of given structure

- The data:

  $\{(z^{(1)},x^{(1)}), (z^{(2)},x^{(2)}), (z^{(3)},x^{(3)}), \ldots (z^{(N)},x^{(N)})\}$

The basic idea underlying MLE



- Likelihood
  (for now let's assume that the structure is given):

  $L(\boldsymbol{\theta} \mid X) = p(X \mid \boldsymbol{\theta}) = p(X_1 \mid \theta_1)\, p(X_2 \mid \theta_2)\, p(X_3 \mid X_3, X_3; \theta_3)$

- Log-Likelihood:

  $l(\boldsymbol{\theta} \mid X) = \log p(X \mid \boldsymbol{\theta}) = \log p(X_1 \mid \theta_1) + \log p(X_2 \mid \theta_2) + \log p(X_3 \mid X_3, X_3, \theta_3)$

- Data log-likelihood

  $l(\boldsymbol{\theta} \mid DATA) = \log \prod_n p(X_n \mid \boldsymbol{\theta})$

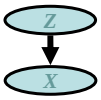
  $= \sum_n \log p(X_{n,1} \mid \theta_1) + \sum_n \log p(X_{n,2} \mid \theta_2) + \sum_n \log p(X_{n,3} \mid X_{n,1} X_{n,2}, \theta_3)$

- MLE

  $\{\theta_1, \theta_2, \theta_3\}_{MLE} = \arg\max l(\boldsymbol{\theta} \mid DATA)$

$\theta_1^* = \arg\max \sum_n \log p(X_{n,1} \mid \theta_1), \quad \theta_2^* = \arg\max \sum_n \log p(X_{n,2} \mid \theta_2), \quad \theta_3^* = \arg\max \sum_n \log p(X_{n,3} \mid X_{n,1} X_{n,2}, \theta_3)$

Example 1: conditional Gaussian



- The completely observed model:
  - Z is a class indicator vector

$$Z = \begin{bmatrix} Z^1 \\ Z^2 \\ \vdots \\ Z^M \end{bmatrix}, \quad \text{where } Z^m = [0,1], \text{ and } \sum_m Z^m = 1$$

and a datum is in class i w.p. π_i

$$p(z^i = 1 \mid \pi) = \pi_i = \pi_1^{z^1} \times \pi_2^{z^2} \times \ldots \times \pi_M^{z^M}$$

All except one of these terms will be one

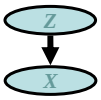
$$p(z) = \prod_m \pi_m^{z^m}$$

  - X is a conditional Gaussian variable with a class-specific mean

$$p(x \mid z^m = 1, \mu, \sigma) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{ -\frac{1}{2\sigma^2}(x - \mu_m)^2 \right\}$$

$$p(x \mid z, \mu, \sigma) = \prod_m N(x \mid \mu_m, \sigma)^{z^m}$$

Example 1: conditional Gaussian



- Data log-likelihood

$$l(\boldsymbol{\theta} \mid D) = \log \prod_n p(z_n, x_n) = \log \prod_n p(z_n \mid \pi) p(x_n \mid z_n, \mu, \sigma)$$

$$= \sum_n \log p(z_n \mid \pi) + \sum_n \log p(x_n \mid z_n, \mu, \sigma)$$

$$= \sum_n \log \prod_m \pi_m^{z_n^m} + \sum_n \log \prod_m N(x_n \mid \mu_m, \sigma)^{z_n^m}$$

$$= \sum_n \sum_m z_n^m \log \pi_m - \sum_n \sum_m z_n^m \frac{1}{2\sigma^2}(x_n - \mu_m)^2 + C$$

- MLE

$$\pi_m^* = \arg\max l(\boldsymbol{\theta} \mid D), \quad \Rightarrow \frac{\partial}{\partial \pi_m} l(\boldsymbol{\theta} \mid D) = 0, \forall m, \quad \text{s.t. } \sum_m \pi_m = 1$$

$$\Rightarrow \pi_m^* = \sum_n z_n^m / N = n_m / N$$

the fraction of samples of class m

$$\mu_m^* = \arg\max l(\boldsymbol{\theta} \mid D), \quad \Rightarrow \mu_m^* = \frac{\sum_n z_n^m x_n}{\sum_n z_n^m} = \frac{\sum_n z_n^m x_n}{n_m}$$

the average of samples of class m

Example 2: HMM: two scenarios

- **Supervised learning**: estimation when the "right answer" is known
  - **Examples:**

    GIVEN: a genomic region $x = x_1 \ldots x_{1,000,000}$ where we have good (experimental) annotations of the CpG islands

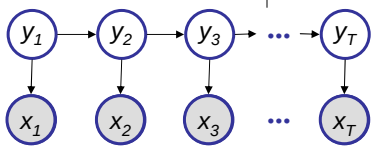
    GIVEN: the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls
- **Unsupervised learning**: estimation when the "right answer" is unknown
  - **Examples:**

    GIVEN: the porcupine genome; we don't know how frequent are the CpG islands there, neither do we know their composition

    GIVEN: 10,000 rolls of the casino player, but we don't see when he changes dice
- **QUESTION:** Update the parameters θ of the model to maximize $P(x|\theta)$ -- Maximal likelihood (ML) estimation

Recall definition of HMM

- Transition probabilities between any two states



$$p(y_t^j = 1 \mid y_{t-1}^i = 1) = a_{i,j},$$

or $$p(y_t \mid y_{t-1}^i = 1) \sim \text{Multinomial}(a_{i,1}, a_{i,2}, \ldots, a_{i,M}), \forall i \in \mathbb{I}.$$

- Start probabilities

$$p(y_1) \sim \text{Multinomial}(\pi_1, \pi_2, \ldots, \pi_M).$$

- Emission probabilities associated with each state

$$p(x_t \mid y_t^i = 1) \sim \text{Multinomial}(b_{i,1}, b_{i,2}, \ldots, b_{i,K}), \forall i \in \mathbb{I}.$$

or in general: $$p(x_t \mid y_t^i = 1) \sim \text{f}(\cdot \mid \theta_i), \forall i \in \mathbb{I}.$$

Supervised ML estimation

- Given $x = x_1 \ldots x_N$ for which the true state path $y = y_1 \ldots y_N$ is known,
  - **Define:**

    A_{ij} = # times state transition $i \to j$ occurs in $\mathbf{y}$

    B_{ik} = # times state i in $\mathbf{y}$ emits k in $\mathbf{x}$

  - **We can show that the maximum likelihood parameters θ are:**

$$a_{ij}^{ML} = \frac{\#(i \to j)}{\#(i \to \bullet)} = \frac{\sum_n \sum_{t=2}^{T} y_{n,t-1}^i y_{n,t}^j}{\sum_n \sum_{t=2}^{T} y_{n,t-1}^i} = \frac{A_{ij}}{\sum_{j'} A_{ij'}}$$

$$b_{ik}^{ML} = \frac{\#(i \to k)}{\#(i \to \bullet)} = \frac{\sum_n \sum_{t=1}^{T} y_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^{T} y_{n,t}^i} = \frac{B_{ik}}{\sum_{k'} B_{ik'}}$$

  - **What if x is continuous? We can treat $\{(x_{n,t}, y_{n,t}) : t = 1:T, n = 1:N\}$ as $N \times T$ observations of, e.g., a Gaussian, and apply learning rules for Gaussian ...**

Supervised ML estimation, ctd.

- **Intuition:**
  - When we know the underlying states, the best estimate of θ is the average frequency of transitions & emissions that occur in the training data
- **Drawback:**
  - Given little data, there may be **overfitting**:
    - $P(x|\theta)$ is maximized, but θ is unreasonable

      0 probabilities – VERY BAD
- **Example:**
  - Given 10 casino rolls, we observe

    `x = 2, 1, 5, 6, 1, 2, 3, 6, 2, 3`

    `y = F, F, F, F, F, F, F, F, F, F`
  - Then: $a_{FF} = 1$; $a_{FL} = 0$

    $b_{F1} = b_{F3} = .2$;

    $b_{F2} = .3$; $b_{F4} = 0$; $b_{F5} = b_{F6} = .1$

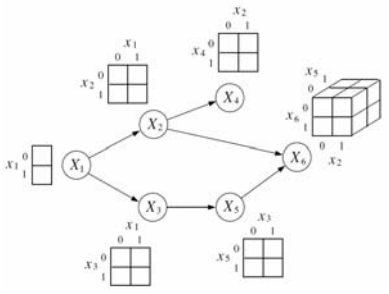
Pseudocounts

- Solution for small training sets:
  - Add pseudocounts
    A_{ij} = # times state transition $i \to j$ occurs in $\mathbf{y}$ + R_{ij}
    B_{ik} = # times state i in $\mathbf{y}$ emits k in $\mathbf{x}$ + S_{ik}
  - R_{ij}, S_{ij} are pseudocounts representing our prior belief
  - Total pseudocounts: $R_i = \Sigma_j R_{ij}$, $S_i = \Sigma_k S_{ik}$,
    - --- "strength" of prior belief,
    - --- total number of imaginary instances in the prior
- Larger total pseudocounts $\Rightarrow$ strong prior belief
- Small total pseudocounts: just to avoid 0 probabilities --- smoothing
- This is equivalent to Bayesian est. under a uniform prior with "parameter strength" equals to the pseudocounts

MLE for general BNs

- If we assume the parameters for each CPD are globally independent, and all nodes are fully observed, then the log-likelihood function decomposes into a sum of local terms, one per node:

$$\ell(\theta; D) = \log p(D \mid \theta)$$
$$= \log \prod_n \left(\prod_i p(x_{n,i} \mid \mathbf{x}_{\pi_i}, \theta_i) \right)$$
$$= \sum_i \left(\sum_n \log p(x_{n,i} \mid \mathbf{x}_{\pi_i}, \theta_i) \right)$$

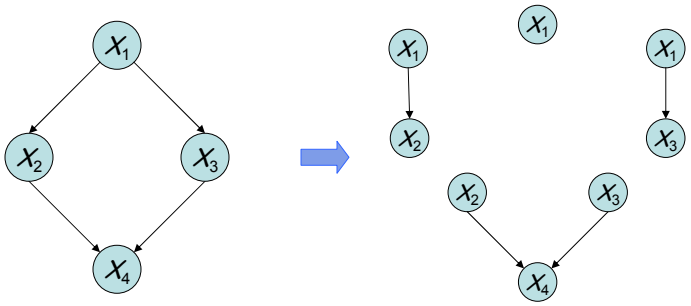


Example: A directed model

- Consider the distribution defined by the directed acyclic GM:

$$p(x \mid \theta) = p(x_1 \mid \theta_1)p(x_2 \mid x_1, \theta_1)p(x_3 \mid x_1, \theta_3)p(x_4 \mid x_2, x_3, \theta_1)$$

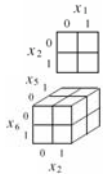
- This is exactly like learning four separate small BNs, each of which consists of a node and its parents.



MLE for BNs with tabular CPDs

- Assume each CPD is represented as a table (multinomial) where

$$\theta_{ijk} \overset{\text{def}}{=} p(X_i = j \mid X_{\pi_i} = k)$$



  - Note that in case of multiple parents, $\mathbf{X}_{\pi_i}$ will have a composite state, and the CPD will be a high-dimensional table
  - The sufficient statistics are counts of family configurations

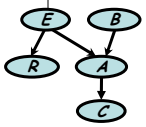
$$n_{ijk} \overset{\text{def}}{=} \sum_n x_{n,i}^j x_{n,\pi_i}^k$$

- The log-likelihood is

$$\ell(\theta; D) = \log \prod_{i,j,k} \theta_{ijk}^{n_{ijk}} = \sum_{i,j,k} n_{ijk} \log \theta_{ijk}$$

- Using a Lagrange multiplier to enforce $\sum_j \theta_{ijk} = 1$, we get:

$$\theta_{ijk}^{ML} = \frac{n_{ijk}}{\sum_{i,j',k} n_{ij'k}}$$



ML Structural Learning for completely observed GMs

$(x_1^{(1)},\ldots,x_n^{(1)})$
$(x_1^{(2)},\ldots,x_n^{(2)})$
…
$(x_1^{(M)},\ldots,x_n^{(M)})$

Information Theoretic Interpretation of ML

$$
\begin{aligned}
\ell(\theta_G, G; D) &= \log p(D \mid \theta_G, G) \\
&= \log \prod_n \left(\prod_i p(x_{n,i} \mid \mathbf{x}_{n,\pi_i(G)}, \theta_{i\mid\pi_i(G)}) \right) \\
&= \sum_i \left(\sum_n \log p(x_{n,i} \mid \mathbf{x}_{n,\pi_i(G)}, \theta_{i\mid\pi_i(G)}) \right) \\
&= M \sum_i \left(\sum_{x_i, \mathbf{x}_{\pi_i(G)}} \frac{count(x_i, \mathbf{x}_{\pi_i(G)})}{M} \log p(x_i \mid \mathbf{x}_{\pi_i(G)}, \theta_{i\mid\pi_i(G)}) \right) \\
&= M \sum_i \left(\sum_{x_i, \mathbf{x}_{\pi_i(G)}} \hat{p}(x_i, \mathbf{x}_{\pi_i(G)}) \log p(x_i \mid \mathbf{x}_{\pi_i(G)}, \theta_{i\mid\pi_i(G)}) \right)
\end{aligned}
$$

From sum over data points to sum over count of variable states

Information Theoretic Interpretation of ML (con'd)

$$
\begin{aligned}
\ell(\theta_G, G; D) &= \log \hat{p}(D \mid \theta_G, G) \\
&= M \sum_i \left(\sum_{x_i, \mathbf{x}_{\pi_i(G)}} \hat{p}(x_i, \mathbf{x}_{\pi_i(G)}) \log \hat{p}(x_i \mid \mathbf{x}_{\pi_i(G)}, \theta_{i|\pi_i(G)}) \right) \\
&= M \sum_i \left(\sum_{x_i, \mathbf{x}_{\pi_i(G)}} \hat{p}(x_i, \mathbf{x}_{\pi_i(G)}) \log \frac{\hat{p}(x_i, \mathbf{x}_{\pi_i(G)}, \theta_{i|\pi_i(G)})}{\hat{p}(\mathbf{x}_{\pi_i(G)})} \frac{\hat{p}(x_i)}{\hat{p}(x_i)} \right) \\
&= M \sum_i \left(\sum_{x_i, \mathbf{x}_{\pi_i(G)}} \hat{p}(x_i, \mathbf{x}_{\pi_i(G)}) \log \frac{\hat{p}(x_i, \mathbf{x}_{\pi_i(G)}, \theta_{i|\pi_i(G)})}{\hat{p}(\mathbf{x}_{\pi_i(G)}) \hat{p}(x_i)} \right) - M \sum_i \left(\sum_{x_i} \hat{p}(x_i) \log p(x_i) \right) \\
&= M \sum_i \hat{I}(x_i, \mathbf{x}_{\pi_i(G)}) - M \sum_i \hat{H}(x_i)
\end{aligned}
$$

Decomposable score and a function of the graph structure

Structural Search

- How many graphs over *n* nodes? $O(2^{n^2})$
- How many trees over *n* nodes? $O(n!)$
- But it turns out that we can find exact solution of an optimal tree (under MLE)!
  - Trick: in a tree each node has only one parent!
  - Chow-liu algorithm

Chow-Liu tree learning algorithm

- Objection function:

$$\ell(\theta_G, G; D) = \log \hat{p}(D \mid \theta_G, G) = M \sum_i \hat{I}(x_i, \mathbf{x}_{\pi_i(G)}) - M \sum_i \hat{H}(x_i) \quad \Rightarrow \quad \boxed{C(G) = M \sum_i \hat{I}(x_i, \mathbf{x}_{\pi_i(G)})}$$

- Chow-Liu:
  - For each pair of variable x_i and x_j
    - Compute empirical distribution: $\hat{p}(X_i, X_j) = \frac{count(x_i, x_j)}{M}$
    - Compute mutual information: $\hat{I}(X_i, X_j) = \sum_{x_i, x_j} \hat{p}(x_i, x_j) \log \frac{\hat{p}(x_i, x_j)}{\hat{p}(x_i)\hat{p}(x_j)}$
  - Define a graph with node $x_1, \ldots, x_n$
    - Edge (I,j) gets weight $\hat{I}(X_i, X_j)$

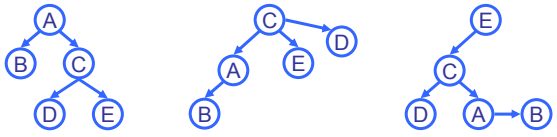
Chow-Liu algorithm (con'd)

- Objection function:

$$\ell(\theta_G, G; D) = \log \hat{p}(D \mid \theta_G, G) = M \sum_i \hat{I}(x_i, \mathbf{x}_{\pi_i(G)}) - M \sum_i \hat{H}(x_i) \quad \Rightarrow \quad \boxed{C(G) = M \sum_i \hat{I}(x_i, \mathbf{x}_{\pi_i(G)})}$$

- Chow-Liu:

  Optimal tree BN
  - Compute maximum weight spanning tree
  - Direction in BN: pick any node as root, do breadth-first-search to define directions
  - I-equivalence:



$$C(G) = I(A,B) + I(A,C) + I(C,D) + I(C,E)$$

Structure Learning for general graphs

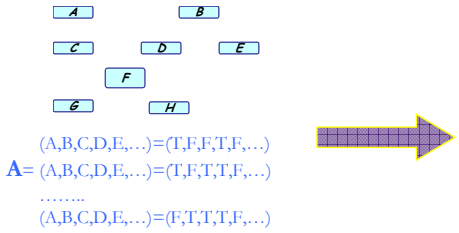
- Theorem:
  - The problem of learning a BN structure with at most *d* parents is NP-hard for any (fixed) *d*≥2
- Most structure learning approaches use heuristics
  - Exploit score decomposition
  - Two heuristics that exploit decomposition in different ways
    - Greedy search through space of node-orders
    - Local search of graph structures

Order search versus graph search

- Order search advantages
  - For fixed order, optimal BN –more "global"optimization
  - Space of orders much smaller than space of graphs
- Graph search advantages
  - Not restricted to k parents
  - Especially if exploiting CPD structure, such as CSI
  - Cheaper per iteration
  - Finer moves within a graph

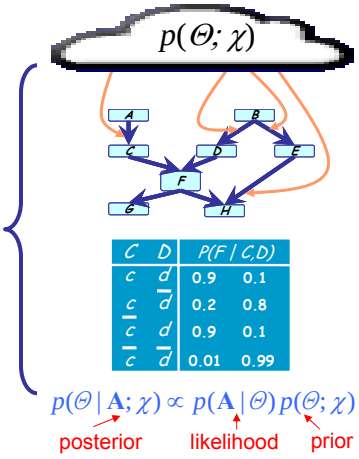
Bayesian model averaging

- Probabilistic statements of Θ is conditioned on the values of the observed variables $\mathbf{A}_{obs}$ and prior $p(\,|\chi)$



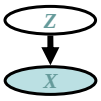
$\mathbf{A}$= (A,B,C,D,E,…)=(T,F,F,T,F,…)
(A,B,C,D,E,…)=(T,F,T,T,F,…)
……..
(A,B,C,D,E,…)=(F,T,T,T,F,…)

$$\Theta_{Bayes} = \int \Theta \; p(\Theta \mid \mathbf{A}, \chi)\, d\Theta$$



$$p(\Theta \mid \mathbf{A};\chi) \propto p(\mathbf{A} \mid \Theta)\, p(\Theta;\chi)$$

posterior likelihood prior



Learning partially observed GMs

- The data:

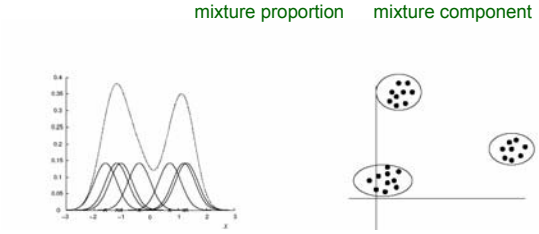
$\{(x^{(1)}), (x^{(2)}), (x^{(3)}), \ldots (x^{(N)})\}$

Gaussian Mixture Models (GMMs)

- Consider a mixture of *K* Gaussian components:

$$p(x_n \mid \mu, \Sigma) = \sum_k \pi_k N(x, \mid \mu_k, \Sigma_k)$$

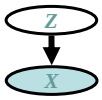
mixture proportion mixture component



- This model can be used for unsupervised clustering.
  - This model (fit by AutoClass) has been used to discover new kinds of stars in astronomical data, etc.

Gaussian Mixture Models (GMMs)

- Consider a mixture of *K* Gaussian components:
  - *Z* is a latent class indicator vector:



$$p(z_n) = \text{multi}(z_n : \pi) = \prod_k (\pi_k)^{z_n^k}$$

  - *X* is a conditional Gaussian variable with a class-specific mean/covariance

$$p(x_n \mid z_n^k = 1, \mu, \Sigma) = \frac{1}{(2\pi)^{m/2} \left|\Sigma_k\right|^{1/2}} \exp\left\{-\tfrac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k)\right\}$$

  - The likelihood of a sample:

mixture component

mixture proportion

$$p(x_n \mid \mu, \Sigma) = \sum_k p(z^k = 1 \mid \pi) p(x, \mid z^k = 1, \mu, \Sigma)$$
$$= \sum_{z_n} \prod_k \left((\pi_k)^{z_n^k} N(x_n : \mu_k, \Sigma_k)^{z_n^k} \right) = \sum_k \pi_k N(x, \mid \mu_k, \Sigma_k)$$

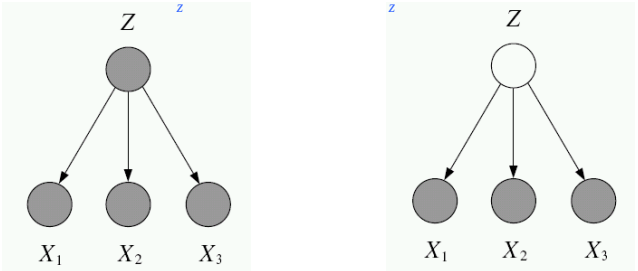
Why is Learning Harder?

- In fully observed iid settings, the log likelihood decomposes into a sum of local terms (at least for directed models).

$$\ell_c(\theta; D) = \log p(x, z \mid \theta) = \log p(z \mid \theta_z) + \log p(x \mid z, \theta_x)$$

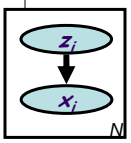
- With latent variables, all the parameters become coupled together via marginalization

$$\ell_c(\theta; D) = \log \sum_z p(x, z \mid \theta) = \log \sum_z p(z \mid \theta_z)\, p(x \mid z, \theta_x)$$



Toward the EM algorithm

- Recall MLE for completely observed data



- Data log-likelihood

$$\ell(\boldsymbol{\theta}; D) = \log \prod_n p(z_n, x_n) = \log \prod_n p(z_n \mid \pi) p(x_n \mid z_n, \mu, \sigma)$$

$$= \sum_n \log \prod_k \pi_k^{z_n^k} + \sum_n \log \prod_k N(x_n; \mu_k, \sigma)^{z_n^k}$$

$$= \sum_n \sum_k z_n^k \log \pi_k - \sum_n \sum_k z_n^k \tfrac{1}{2\sigma^2} (x_n - \mu_k)^2 + C$$

- MLE

$$\hat{\pi}_{k,MLE} = \arg\max_\pi \ell(\boldsymbol{\theta}; D),$$

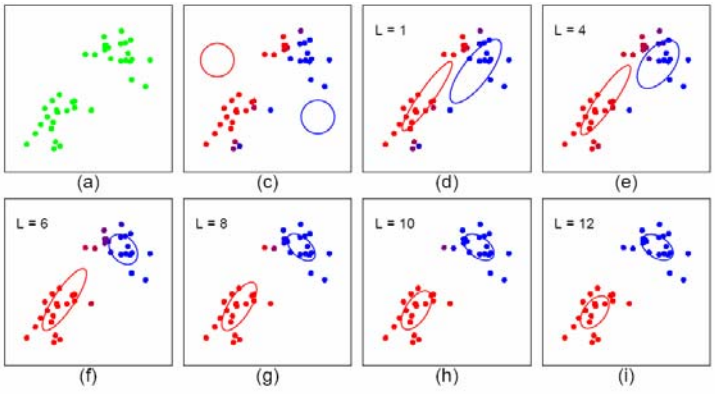
$$\hat{\mu}_{k,MLE} = \arg\max_\mu \ell(\boldsymbol{\theta}; D) \quad \Rightarrow \quad \hat{\mu}_{k,MLE} = \frac{\sum_n z_n^k x_n}{\sum_n z_n^k}$$

$$\hat{\sigma}_{k,MLE} = \arg\max_\sigma \ell(\boldsymbol{\theta}; D)$$

- What if we do not know z_n?

Expectation-Maximization

- Start:
  - "Guess" the centroid μ_k and coveriance Σ_k of each of the K clusters
- Loop



Example: Gaussian mixture model

- A mixture of K Gaussians:
  - Z is a latent class indicator vector

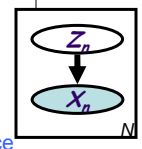
$$p(z_n) = \text{multi}(z_n : \pi) = \prod_k (\pi_k)^{z_n^k}$$

  - X is a conditional Gaussian variable with class-specific mean/covariance

$$p(x_n \mid z_n^k = 1, \mu, \Sigma) = \frac{1}{(2\pi)^{m/2}|\Sigma_k|^{1/2}} \exp\left\{-\tfrac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1}(x_n - \mu_k)\right\}$$

  - The likelihood of a sample:

$$p(x_n \mid \mu, \Sigma) = \sum_k p(z^k = 1 \mid \pi) p(x, \mid z^k = 1, \mu, \Sigma)$$
$$= \sum_{z_n} \prod_k \left((\pi_k)^{z_n^k} N(x_n : \mu_k, \Sigma_k)^{z_n^k}\right) = \sum_k \pi_k N(x, \mid \mu_k, \Sigma_k)$$



- The expected complete log likelihood

$$\langle \ell_c(\theta; x, z) \rangle = \sum_n \langle \log p(z_n \mid \pi) \rangle_{p(z \mid x)} + \sum_n \langle \log p(x_n \mid z_n, \mu, \Sigma) \rangle_{p(z \mid x)}$$
$$= \sum_n \sum_k \langle z_n^k \rangle \log \pi_k - \frac{1}{2} \sum_n \sum_k \langle z_n^k \rangle \left((x_n - \mu_k)^T \Sigma_k^{-1}(x_n - \mu_k) + \log|\Sigma_k| + C\right)$$

E-step

- We maximize $\langle l_c(\theta) \rangle$ iteratively using the following iterative procedure:
  - Expectation step: computing the expected value of the sufficient statistics of the hidden variables (i.e., z) given current est. of the parameters (i.e., π and μ).

$$\tau_n^{k(t)} = \langle z_n^k \rangle_{q^{(t)}} = p(z_n^k = 1 \mid x, \mu^{(t)}, \Sigma^{(t)}) = \frac{\pi_k^{(t)} N(x_n, \mid \mu_k^{(t)}, \Sigma_k^{(t)})}{\sum_i \pi_i^{(t)} N(x_n, \mid \mu_i^{(t)}, \Sigma_i^{(t)})}$$

  - Here we are essentially doing **inference**

M-step

- We maximize $\langle l_c(\theta) \rangle$ iteratively using the following iterative procudure:
  - Maximization step: compute the parameters under current results of the expected value of the hidden variables

$$\pi_k^* = \arg\max \langle l_c(\theta) \rangle, \quad \Rightarrow \frac{\partial}{\partial \pi_k} \langle l_c(\theta) \rangle = 0, \forall k, \quad \text{s.t.} \sum_k \pi_k = 1$$

$$\Rightarrow \pi_k^* = \sum_n \langle z_n^k \rangle_{q^{(t)}} / N = \sum_n \tau_n^{k(t)} / N = \langle n_k \rangle / N$$

$$\mu_k^* = \arg\max \langle l(\theta) \rangle, \quad \Rightarrow \mu_k^{(t+1)} = \frac{\sum_n \tau_n^{k(t)} x_n}{\sum_n \tau_n^{k(t)}}$$

$$\Sigma_k^* = \arg\max \langle l(\theta) \rangle, \quad \Rightarrow \Sigma_k^{(t+1)} = \frac{\sum_n \tau_n^{k(t)} (x_n - \mu_k^{(t+1)})(x_n - \mu_k^{(t+1)})^T}{\sum_n \tau_n^{k(t)}}$$

Fact:

$$\frac{\partial \log |A^{-1}|}{\partial A^{-1}} = A^T$$

$$\frac{\partial x^T A x}{\partial A} = x x^T$$

  - This is isomorphic to **MLE** except that the variables that are hidden are replaced by their expectations (in general they will by replaced by their corresponding "**sufficient statistics**")

Theory underlying EM

- What are we doing?
- Recall that according to MLE, we intend to learn the model parameter that would have maximize the likelihood of the data.
- But we do not observe z, so computing

$$\ell_c(\theta; D) = \log \sum_z p(x, z \mid \theta) = \log \sum_z p(z \mid \theta_z)\, p(x \mid z, \theta_x)$$

  is difficult!

- What shall we do?

Complete & Incomplete Log Likelihoods

- Complete log likelihood

  Let X denote the observable variable(s), and Z denote the latent variable(s).

  If Z could be observed, then

$$\ell_c(\theta; x, z) \overset{\text{def}}{=} \log p(x, z \mid \theta)$$

  - Usually, optimizing $\ell_c()$ given both z and x is straightforward (c.f. MLE for fully observed models).
  - Recalled that in this case the objective for, e.g., MLE, decomposes into a sum of factors, the parameter for each factor can be estimated separately.
  - **But given that Z is not observed, $\ell_c()$ is a random quantity, cannot be maximized directly**.

- Incomplete log likelihood

  With z unobserved, our objective becomes the log of a marginal probability:

$$\ell_c(\theta; x) = \log p(x \mid \theta) = \log \sum_z p(x, z \mid \theta)$$

  - **This objective won't decouple**

Expected Complete Log Likelihood

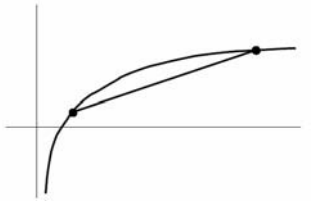
- For ***any*** distribution $q(z)$, define *expected complete log likelihood*:

$$\langle \ell_c(\theta; x, z) \rangle_q \stackrel{\text{def}}{=} \sum_z q(z \mid x, \theta) \log p(x, z \mid \theta)$$

  - A deterministic function of θ
  - Linear in $\ell_c()$ --- inherit its factorizabiility
  - Does maximizing this surrogate yield a maximizer of the likelihood?

- Jensen's inequality

$$\begin{aligned} \ell(\theta; x) &= \log p(x \mid \theta) \\ &= \log \sum_z p(x, z \mid \theta) \\ &= \log \sum_z q(z \mid x) \frac{p(x, z \mid \theta)}{q(z \mid x)} \\ &\geq \sum_z q(z \mid x) \log \frac{p(x, z \mid \theta)}{q(z \mid x)} \end{aligned} \quad \Rightarrow \quad \ell(\theta; x) \geq \langle \ell_c(\theta; x, z) \rangle_q + H_q$$

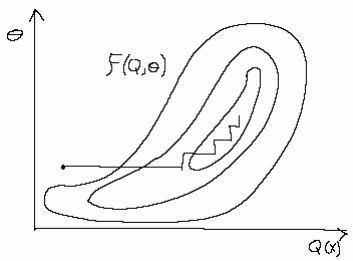


Lower Bounds and Free Energy

- For fixed data x, define a functional called the free energy:

$$F(q, \theta) \stackrel{\text{def}}{=} \sum_z q(z \mid x) \log \frac{p(x, z \mid \theta)}{q(z \mid x)} \leq \ell(\theta; x)$$

- The EM algorithm is coordinate-ascent on F:
  - **E-step:** $q^{t+1} = \arg\max_q F(q, \theta^t)$
  - **M-step:** $\theta^{t+1} = \arg\max_\theta F(q^{t+1}, \theta^t)$



E-step: maximization of expected ℓ_c w.r.t. q

- Claim:
$$q^{t+1} = \arg\max_q F(q, \theta^t) = p(z \mid x, \theta^t)$$
  - This is the posterior distribution over the latent variables given the data and the parameters. Often we need this at test time anyway (e.g. to perform classification).
- Proof (easy): this setting attains the bound $\ell(\theta; x) \geq F(q, \theta)$
$$\begin{aligned} F(p(z|x,\theta^t),\theta^t) &= \sum_z p(z|x,\theta^t) \log \frac{p(x,z \mid \theta^t)}{p(z|x,\theta^t)} \\ &= \sum_z q(z \mid x) \log p(x \mid \theta^t) \\ &= \log p(x \mid \theta^t) = \ell(\theta^t; x) \end{aligned}$$
- Can also show this result using variational calculus or the fact that $\ell(\theta; x) - F(q, \theta) = \mathrm{KL}(q \,\|\, p(z \mid x, \theta))$

E-step ≡ plug in posterior expectation of latent variables

- Without loss of generality: assume that $p(x,z|\theta)$ is a generalized exponential family distribution:
$$p(x,z|\theta) = \frac{1}{Z(\theta)} h(x,z) \exp\left\{\sum_i \theta_i f_i(x,z)\right\}$$
  - Special cases: if $p(X|Z)$ are GLIMs, then $f_i(x,z) = \eta_i^T(z)\xi_i(x)$
- The expected complete log likelihood under $q^{t+1} = p(z \mid x, \theta^t)$ is
$$\begin{aligned} \left\langle \ell_c(\theta^t; x, z) \right\rangle_{q^{t+1}} &= \sum_z q(z \mid x, \theta^t) \log p(x, z \mid \theta^t) - A(\theta) \\ &= \sum_i \theta_i^t \left\langle f_i(x,z) \right\rangle_{q(z|x,\theta^t)} - A(\theta) \\ &\overset{p\sim\text{GLIM}}{=} \sum_i \theta_i^t \left\langle \eta_i(z) \right\rangle_{q(z|x,\theta^t)} \xi_i(x) - A(\theta) \end{aligned}$$

M-step: maximization of expected ℓ_c w.r.t. θ

- Note that the free energy breaks into two terms:

$$F(q,\theta) = \sum_z q(z \mid x) \log \frac{p(x,z \mid \theta)}{q(z \mid x)}$$
$$= \sum_z q(z \mid x) \log p(x,z \mid \theta) - \sum_z q(z \mid x) \log q(z \mid x)$$
$$= \langle \ell_c(\theta; x, z) \rangle_q + H_q$$

  - The first term is the expected complete log likelihood (energy) and the second term, which does not depend on θ, is the entropy.
- Thus, in the M-step, maximizing with respect to θ for fixed q we only need to consider the first term:

$$\theta^{t+1} = \arg\max_\theta \langle \ell_c(\theta; x, z) \rangle_{q^{t+1}} = \arg\max_\theta \sum_z q(z \mid x) \log p(x,z \mid \theta)$$

  - Under optimal q^{t+1}, this is equivalent to solving a standard MLE of fully observed model $p(x,z|\theta)$, with the sufficient statistics involving z replaced by their expectations w.r.t. $p(z|x,\theta)$.

Example: HMM

- **Supervised learning**: estimation when the "right answer" is known
  - **Examples:**
    GIVEN: a genomic region $x = x_1 \ldots x_{1,000,000}$ where we have good (experimental) annotations of the CpG islands
    GIVEN: the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls
- **Unsupervised learning**: estimation when the "right answer" is unknown
  - **Examples:**
    GIVEN: the porcupine genome; we don't know how frequent are the CpG islands there, neither do we know their composition
    GIVEN: 10,000 rolls of the casino player, but we don't see when he changes dice
- **QUESTION:** Update the parameters θ of the model to maximize $P(x|\theta)$ -- Maximal likelihood (ML) estimation

The Baum Welch algorithm

- The complete log likelihood

$$\ell_c(\boldsymbol{\theta};\mathbf{x},\mathbf{y}) = \log p(\mathbf{x},\mathbf{y}) = \log \prod_n \left(p(y_{n,1}) \prod_{t=2}^{T} p(y_{n,t} \mid y_{n,t-1}) \prod_{t=1}^{T} p(x_{n,t} \mid x_{n,t}) \right)$$

- The expected complete log likelihood

$$\langle \ell_c(\boldsymbol{\theta};\mathbf{x},\mathbf{y}) \rangle = \sum_n \left(\langle y_{n,1}^i \rangle_{p(y_{n,1}|\mathbf{x}_n)} \log \pi_i \right) + \sum_n \sum_{t=2}^{T} \left(\langle y_{n,t-1}^i y_{n,t}^j \rangle_{p(y_{n,t-1},y_{n,t}|\mathbf{x}_n)} \log a_{i,j} \right) + \sum_n \sum_{t=1}^{T} \left(x_{n,t}^k \langle y_{n,t}^i \rangle_{p(y_{n,t}|\mathbf{x}_n)} \log b_{i,k} \right)$$

- EM
  - The **E** step

$$\gamma_{n,t}^i = \langle y_{n,t}^i \rangle = p(y_{n,t}^i = 1 \mid \mathbf{x}_n)$$

$$\xi_{n,t}^{i,j} = \langle y_{n,t-1}^i y_{n,t}^j \rangle = p(y_{n,t-1}^i = 1, y_{n,t}^j = 1 \mid \mathbf{x}_n)$$

  - The **M** step ("symbolically" identical to MLE)

$$\pi_i^{ML} = \frac{\sum_n \gamma_{n,1}^i}{N} \qquad a_{ij}^{ML} = \frac{\sum_n \sum_{t=2}^{T} \xi_{n,t}^{i,j}}{\sum_n \sum_{t=1}^{T-1} \gamma_{n,t}^i} \qquad b_{ik}^{ML} = \frac{\sum_n \sum_{t=1}^{T} \gamma_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^{T-1} \gamma_{n,t}^i}$$

Unsupervised ML estimation

- Given $x = x_1 \ldots x_N$ for which the true state path $y = y_1 \ldots y_N$ is unknown,
  - **EXPECTATION MAXIMIZATION**

0. Starting with our best guess of a model M, parameters θ:
1. Estimate A_{ij} , B_{ik} in the training data
   - How? $A_{ij} = \sum_{n,t} \langle y_{n,t-1}^i y_{n,t}^j \rangle$ $B_{ik} = \sum_{n,t} \langle y_{n,t}^i \rangle x_{n,t}^k$,
2. Update θ according to A_{ij} , B_{ik}
   - Now a "supervised learning" problem
3. Repeat 1 & 2, until convergence

This is called the Baum-Welch Algorithm

We can get to a provably more (or equally) likely parameter set θ each iteration

EM for general BNs

while not converged

% E-step

for each node i

$ESS_i = 0$ % reset expected sufficient statistics

for each data sample n

do inference with $X_{n,H}$

for each node i

$$ESS_i \mathrel{+}= \left\langle SS_i(x_{n,i}, x_{n,\pi_i}) \right\rangle_{p(x_{n,H} | x_{n,-H})}$$

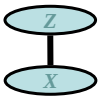
% M-step

for each node i

θ_i := MLE(ESS_i)

Summary: EM Algorithm

- A way of maximizing likelihood function for latent variable models. Finds MLE of parameters when the original (hard) problem can be broken up into two (easy) pieces:
  1. Estimate some "missing" or "unobserved" data from observed data and current parameters.
  2. Using this "complete" data, find the maximum likelihood parameter estimates.
- Alternate between filling in the latent variables using the best guess (posterior) and updating the parameters based on this guess:
  - E-step: $q^{t+1} = \arg\max_q F(q, \theta^t)$
  - M-step: $\theta^{t+1} = \arg\max_\theta F(q^{t+1}, \theta^t)$
- In the M-step we optimize a lower bound on the likelihood. In the E-step we close the gap, making bound=likelihood.



Learning completely observed undirected GMs

- The data:

  $\{(z^{(1)},x^{(1)}), (z^{(2)},x^{(2)}), (z^{(3)},x^{(3)}), \ldots (z^{(N)},x^{(N)})\}$

MLE for undirected graphical models

- For directed graphical models, the log-likelihood decomposes into a sum of terms, one per family (node plus parents).
- For undirected graphical models, the log-likelihood **does not** decompose, because the normalization constant Z is a function of **all** the parameters

$$P(x_1,\ldots,x_n) = \frac{1}{Z}\prod_{c\in\mathcal{C}}\psi_c(\mathbf{x}_c) \qquad Z = \sum_{x_1,\ldots,x_n}\prod_{c\in\mathcal{C}}\psi_c(\mathbf{x}_c)$$

- In general, we will need to do inference (i.e., marginalization) to learn parameters for undirected models, even in the fully observed case.

Feature-based Clique Potentials

- So far we have discussed the most general form of an undirected graphical model in which cliques are parameterized by general potential functions $\psi_c(\mathbf{x}_c)$.

$$P(x_1,\ldots,x_n) = \frac{1}{Z}\prod_{c\in C}\psi_c(\mathbf{x}_c)$$

- But for large cliques these general potentials are exponentially costly for inference and have exponential numbers of parameters that we must learn from limited data.
- One solution: change the graphical model to make cliques smaller. But this changes the dependencies, and may force us to make more independence assumptions than we would like.
- Another solution: keep the same graphical model, but use a less general parameterization of the clique potentials.
- This is the idea behind feature-based models.

Features

- Consider a clique $\mathbf{x}_c$ of random variables in a UGM, e.g. three consecutive characters $c_1c_2c_3$ in a string of English text.
- How would we build a model of $p(c_1c_2c_3)$?
  - If we use a single clique function over $c_1c_2c_3$, the full joint clique potential would be huge: 26^3-1 parameters.
  - However, we often know that some particular joint settings of the variables in a clique are quite likely or quite unlikely. e.g. ing, ate, ion, ?ed, qu?, jkx, zzz,...
- A "feature" is a function which is vacuous over all joint settings except a few particular ones on which it is high or low.
  - For example, we might have $f_{ing}(c_1c_2c_3)$ which is 1 if the string is 'ing' and 0 otherwise, and similar features for '?ed', etc.
- We can also define features when the inputs are continuous. Then the idea of a cell on which it is active disappears, but we might still have a compact parameterization of the feature.

Features as Micropotentials

- By exponentiating them, each feature function can be made into a "micropotential". We can multiply these micropotentials together to get a **clique potential**.
- Example: a clique potential $\psi(c_1 c_2 c_3)$ could be expressed as:

$$\psi_c(c_1,c_2,c_3) = e^{\theta_{\text{ing}} f_{\text{ing}}} \times e^{\theta_{?\text{ed}} f_{?\text{ed}}} \times \ldots$$
$$= \exp\left\{\sum_{k=1}^{K} \theta_k f_k(c_1,c_2,c_3)\right\}$$

- This is still a potential over 26^3 possible settings, but only uses K parameters if there are K features.
  - By having one indicator function per combination of $\mathbf{x}_c$, we recover the standard tabular potential.

Combining Features

- Each feature has a weight θ_k which represents the numerical strength of the feature and whether it increases or decreases the probability of the clique.
- The marginal over the clique is a generalized exponential family distribution, actually, a GLIM:

$$p(c_1,c_2,c_3) \propto \exp\left\{\begin{array}{l}\theta_{\text{ing}} f_{\text{ing}}(c_1,c_2,c_3) + \theta_{?\text{ed}} f_{?\text{ed}}(c_1,c_2,c_3) + \\ \theta_{\text{qu?}} f_{\text{qu?}}(c_1,c_2,c_3) + \theta_{\text{zzz}} f_{\text{zzz}}(c_1,c_2,c_3) + \cdots\end{array}\right\}$$

- In general, the features may be overlapping, unconstrained indicators or any function of any subset of the clique variables:

$$\psi_c(\mathbf{x}_c) \overset{\text{def}}{=} \exp\left\{\sum_{i \in \mathcal{I}_c} \theta_k f_k(\mathbf{x}_{c_i})\right\}$$

- How can we combine feature into a probability model?

Feature Based Model

- We can multiply these clique potentials as usual:

$$p(\mathbf{x}) = \frac{1}{Z(\theta)} \prod_c \psi_c(\mathbf{x}_c) = \frac{1}{Z(\theta)} \exp\left\{ \sum_c \sum_{i \in \mathcal{I}_c} \theta_k f_k(\mathbf{x}_{c_i}) \right\}$$

- However, in general we can forget about associating features with cliques and just use a simplified form:

$$p(\mathbf{x}) = \frac{1}{Z(\theta)} \exp\left\{ \sum_i \theta_i f_i(\mathbf{x}_{c_i}) \right\}$$

- This is just our friend the exponential family model, with the features as sufficient statistics

MLE of Feature Based UGMs

- Scaled likelihood function

$$\tilde{\ell}(\theta; D) = \ell(\theta; D)/N = \frac{1}{N} \sum_n \log p(x_n \mid \theta)$$
$$= \sum_x \tilde{p}(x) \log p(x \mid \theta)$$
$$= \sum_x \tilde{p}(x) \sum_i \theta_i f_i(x) - \log Z(\theta)$$

- Instead of optimizing this objective directly, we attack its lower bound
  - The logarithm has a linear upper bound …

$$\log Z(\theta) \le \mu Z(\theta) - \log \mu - 1$$

  - This bound holds for all μ, in particular, for $\mu = Z^{-1}(\theta^{(t)})$

  - Thus we have

$$\tilde{\ell}(\theta; D) \ge \sum_x \tilde{p}(x) \sum_i \theta_i f_i(x) - \frac{Z(\theta)}{Z(\theta^{(t)})} - \log Z(\theta^{(t)}) + 1$$

Generalized Iterative Scaling (GIS)

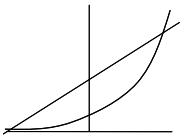
- Lower bound of scaled loglikelihood

$$\tilde{\ell}(\theta;D) \geq \sum_x \tilde{p}(x)\sum_i \theta_i f_i(x) - \frac{Z(\theta)}{Z(\theta^{(t)})} - \log Z(\theta^{(t)}) + 1$$

- Define $\Delta\theta_i^{(t)} \overset{\text{def}}{=} \theta_i - \theta_i^{(t)}$

$$\tilde{\ell}(\theta;D) \geq \sum_x \tilde{p}(x)\sum_i \theta_i f_i(x) - \frac{1}{Z(\theta^{(t)})}\sum_x \exp\left\{\sum_i \theta_i f_i(x)\right\} - \log Z(\theta^{(t)}) + 1$$

- Relax again
  - Assume $f_i(x) \geq 0, \quad \sum_i f_i(x) = 1$
  - Convexity of exponential: $\exp\left(\sum_i \pi_i x_i\right) \leq \sum_i \pi_i \exp(x_i)$



- We have:

$$\tilde{\ell}(\theta;D) \geq \sum_i \theta_i \sum_x \tilde{p}(x) f_i(x) - \sum_x p(x \mid \theta^{(t)}) \sum_i f_i(x) \exp\left(\Delta\theta_i^{(t)}\right) - \log Z(\theta^{(t)}) + 1 \overset{\text{def}}{=} \Lambda(\theta)$$

GIS

- Lower bound of scaled loglikelihood

$$\tilde{\ell}(\theta;D) \geq \sum_i \theta_i \sum_x \tilde{p}(x) f_i(x) - \sum_x p(x \mid \theta^{(t)}) \sum_i f_i(x) \exp\left(\Delta\theta_i^{(t)}\right) - \log Z(\theta^{(t)}) + 1 \overset{\text{def}}{=} \Lambda(\theta)$$

- Take derivative: $\frac{\partial \Lambda}{\partial \theta_i} = \sum_x \tilde{p}(x) f_i(x) - \exp\left(\Delta\theta_i^{(t)}\right) \sum_x p(x \mid \theta^{(t)}) f_i(x)$
- Set to zero

$$e^{\Delta\theta_i^{(t)}} = \frac{\sum_x \tilde{p}(x) f_i(x)}{\sum_x p(x \mid \theta^{(t)}) f_i(x)} = \frac{\sum_x \tilde{p}(x) f_i(x)}{\sum_x p^{(t)}(x) f_i(x)} Z(\theta^{(t)})$$

  - where $p^{(t)}(x)$ is the unnormalized version of $p(x|\theta^{(t)})$

- Update

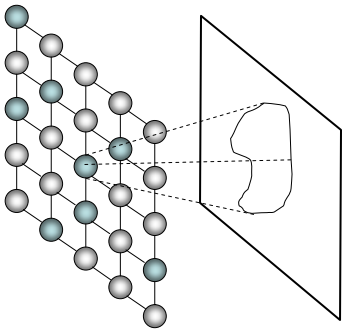
$$\theta_i^{(t+1)} = \theta_i^{(t)} + \Delta\theta_i^{(t)} \Rightarrow p^{(t+1)}(x) = p^{(t)}(x) e^{\Delta\theta_i^{(t)} f_i(x)}$$

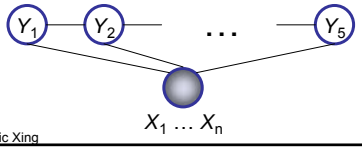
$$p^{(t+1)}(x) = \frac{p^{(t)}(x)}{Z(\theta^{(t)})} \prod_i \left(\frac{\sum_x \tilde{p}(x) f_i(x)}{\sum_x p^{(t)}(x) f_i(x)} Z(\theta^{(t)})\right)^{f_i(x)}$$

$$\Rightarrow = \frac{p^{(t)}(x)}{Z(\theta^{(t)})} \prod_i \left(\frac{\sum_x \tilde{p}(x) f_i(x)}{\sum_x p^{(t)}(x) f_i(x)}\right)^{f_i(x)} \left(Z(\theta^{(t)})\right)^{\sum_i f_i(x)}$$

$$= p^{(t)}(x) \prod_i \left(\frac{\sum_x \tilde{p}(x) f_i(x)}{\sum_x p^{(t)}(x) f_i(x)}\right)^{f_i(x)}$$

Example: Conditional Random Fields





$$p_\theta(y \mid x) = \frac{1}{Z(\theta, x)} \exp\left\{\sum_c \theta_c f_c(x, y_c)\right\}$$

- Allow arbitrary dependencies on input
- Clique dependencies on labels
- Use approximate inference for general graphs

Alternative Learning Strategy

- Recall that in CRF
  - We predict based on:

$$y^* \mid x = \arg\max_y p_\theta(y \mid x) = \frac{1}{Z(\theta, x)} \exp\left\{\sum_c \theta_c f_c(x, y_c)\right\}$$

  - And we learn based on:

$$\theta_c^* \mid \{y_n, x_n\} = \arg\max_{\theta_c} \prod_n p_\theta(y_n \mid x_n) = \prod_n \frac{1}{Z(\theta, x_n)} \exp\left\{\sum_c \theta_c f_c(x_n, y_{n,c})\right\}$$

- MaxMargin:
  - We predict based on:

$$y^* \mid x = \arg\max_y \sum_c \theta_c f_c(x, y_c) = \arg\max_y w^T F(x, y)$$

  - And we learn based on:

$$w^* \mid \{y_n, x_n\} = \arg\max_w \left(\max_{y'_n \neq y_n, \forall n} w^T \left(F(y_n, x_n) - F(y'_n, x_n)\right)\right)$$

Max-Margin Learning

$$\max \quad \frac{1}{2}\|w\| - \sum_n \xi_n$$

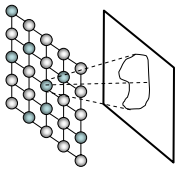
$$\text{s.t.} \quad w^T\left(F(y_n, x_n) - F(y'_n, x_n)\right) \geq \xi_n + \Delta(y'_n, y_n) \qquad \forall n, y'_n \in \mathcal{Y}_n \setminus y_n$$

$$\xi_n \geq 0$$

- Solutions:
  - Convex optimization (akin to SVM) with exponentially many constrains
  - Many algorithms and heuristics exist
    - Interior-point methods
    - Iterative active-support elimination
    - Inference based on GM
    - …

Open Problems

- Unsupervised CRF learning and MaxMargin Learning
  - We want to recognize a pattern that is maximally different from the rest!



  - What does margin or conditional likelihood mean in these cases? Given only $\{X_n\}$, how can we define the cost function?

$$p_\theta(y \mid x) = \frac{1}{Z(\theta, x)} \exp\left\{\sum_c \theta_c f_c(x, y_c)\right\}$$

$$\text{margin} = w^T\left(F(y_n, x_n) - F(y'_n, x_n)\right)$$

  - Algorithmic challenge