

LECTURE 16:  
STRUCTURE LEARNING

Wed 10 Nov 2004

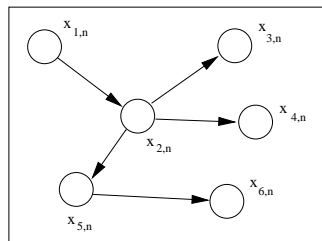
DIRECTED TREE GRAPHICAL MODELS (K&F 14.4.1)

- Directed trees are DAGMs in which each variable  $x_i$  has exactly one other variable as its parent  $x_{\pi_i}$  except the “root”  $x_{\text{root}}$  which has no parents. Thus, the probability of a variable taking on a certain value depends only on the value of its parent:

$$p(\mathbf{x}) = p(x_{\text{root}}) \prod_{i \neq \text{root}} p(x_i | x_{\pi_i})$$

- Trees are the next step up from assuming independence. Instead of considering variables in isolation, consider them in pairs.

NB: each node (except root) has exactly one parent, but nodes may have more than one child.



MAXIMIZING THE SCORE (K&F 14.4.3)

- Consider the family of DAGs  $G_d$  with maximum fan-in (number of parents) equal to  $d$ .
- Theorem 14.4.3: It is NP-hard to find

$$G^* = \arg \max_{G \in G_d} \text{score}(G, D)$$

for any  $d \geq 2$ .

- For  $d \leq 1$  (i.e., trees), we can solve the problem in  $O(n^2)$  time using max spanning tree.
- In general, we need to use heuristic local search.

UNDIRECTED TREE GRAPHICAL MODELS

- Undirected trees are connected, acyclic graphs with exactly  $(D-1)$  edges if there are  $D$  nodes (variables).
- For undirected trees, the cliques are all pairs of connected nodes.

$$p(\mathbf{x}) = \frac{1}{Z} \prod_i \psi_i(x_i, x_{\pi_i})$$

where we can make  $Z = 1$  with the choice  $\psi_i = p(x_i | x_{\pi_i})$  except for one clique involving the root:  $\psi_j = p(x_r) p(x_j | x_{\pi_j})$

- Trees have no “explaining-away” (converging arrows). Therefore, d-separation and regular separation are equivalent.
- Directed and undirected trees are equivalent and the choice of root is arbitrary (for fully observed models).
- Another characterization of trees: there is exactly one path between any pair of nodes (without doubling back).

## LIKELIHOOD FUNCTION

- Notation:

$\mathbf{y}_i \equiv$  a node  $x_i$  and its single parent  $x_{\pi_i}$ .

$\mathbf{V}_i \equiv$  set of joint configurations of node  $i$  and its parent  $x_{\pi_i}$

( $\mathbf{y}_{\text{root}} \equiv x_{\text{root}}$  and  $\mathbf{V}_{\text{root}} \equiv \mathbf{v}_{\text{root}}$ )

- Directed model likelihood:

$$\begin{aligned} \ell(\theta; \mathcal{D}) &= \sum_n \log p(\mathbf{x}^n) = \sum_n \left[ \log p_r(x_r^n) + \sum_{i \neq r} \log p(x_i^n | x_{\pi_i}^n) \right] \\ &= \sum_n \sum_i \sum_{\mathbf{v} \in \mathbf{V}_i} [\mathbf{y}_i^n = \mathbf{v}] \log p_i(\mathbf{v}) \quad \text{indicator trick} \\ &= \sum_i \sum_{\mathbf{v} \in \mathbf{V}_i} N_i(\mathbf{v}) \log p_i(\mathbf{v}) \end{aligned}$$

where  $N_i(\mathbf{v}) = \sum_n [\mathbf{y}_i^n = \mathbf{v}]$  and  $p_i(\mathbf{v}_i) = p(x_i | x_{\pi_i})$ .

## MAXIMUM LIKELIHOOD PARAMETERS GIVEN STRUCTURE

- Trees are just a special case of fully observed graphical models.
- For discrete data  $x_i$  with values  $v_i$ , each node stores a conditional probability table (CPT) over its values given its parent's value. The ML parameter estimates are just the empirical histograms of each node's values given its parent:

$$p^*(x_i = v_i | x_{\pi_i} = v_j) = \frac{N(x_i = v_i, x_{\pi_i} = v_j)}{\sum_{\mathbf{v}_i} N(x_i = v_i, x_{\pi_i} = v_j)} = \frac{N_i(\mathbf{y}_i)}{N_{\pi_i}(v_j)}$$

except for the root which uses marginal counts  $N_r(v_r)/N$ .

- For continuous data, the most common model is a two-dimensional Gaussian at each node. The ML parameters are just to set the mean of  $p_i(\mathbf{y}_i)$  to be the sample mean of  $[x_i; x_{\pi_i}]$  and the covariance matrix to the sample covariance.
- In practice we should use some kind of smoothing/regularization.

## MORE ON THE LIKELIHOOD FUNCTION

- Undirected model likelihood:

$$\begin{aligned} \ell(\theta; \mathcal{D}) &= \sum_n \log \prod_i \psi_i(\mathbf{y}_i^n) \\ &= \sum_n \sum_i \sum_{\mathbf{v} \in \mathbf{V}_i} [\mathbf{y}_i^n = \mathbf{v}] \log \psi_i(\mathbf{v}) \\ &= \sum_i \sum_{\mathbf{v} \in \mathbf{V}_i} N_i(\mathbf{v}) \log \psi_i(\mathbf{v}) \end{aligned}$$

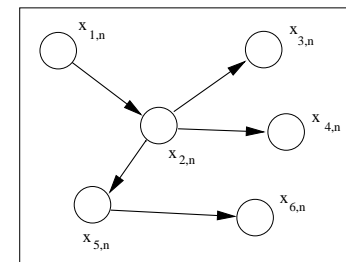
where  $N_i(\mathbf{y}) = \sum_n [\mathbf{y}_i^n = \mathbf{y}]$  and  $\psi_i(\mathbf{y}_i) = p(x_i | x_{\pi_i})$ .

(Except for one clique involving the root:  $\psi_j = p(x_r)p(x_j | x_{\pi_j})$ )

- Directed and undirected likelihoods are the same!
- Trees are in the exponential family with  $\mathbf{y}_i$  as sufficient statistics.

## STRUCTURE LEARNING

- What about the tree structure (links)?  
How do we know which nodes to make parents of which?



- Bold idea: how can we also *learn* the optimal structure?  
In principle, we could search all combinatorial structures, for each compute the ML parameters, and take the best one.
- But is there a better way? Yes. It turns out that structure learning in tree models can be converted to a good old computer science problem: maximum weight spanning tree.

## OPTIMAL STRUCTURE

- Let us rewrite the likelihood function:

$$\begin{aligned} \ell(\theta; \mathcal{D}) &= \sum_{\mathbf{x} \in \mathbf{V}_{\text{all}}} N(\mathbf{x}) \log p(\mathbf{x}) \\ &= \sum_{\mathbf{x}} N(\mathbf{x}) \left( \log p(\mathbf{x}_r) + \sum_{i \neq r} \log p(x_i | x_{\pi_i}) \right) \end{aligned}$$

- ML parameters, are equal to the observed frequency counts  $q(\cdot)$ :

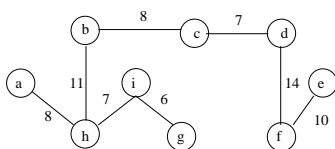
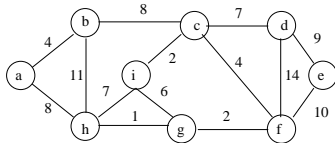
$$\begin{aligned} \frac{\ell^*}{N} &= \sum_{\mathbf{x} \in \mathbf{V}_{\text{all}}} q(\mathbf{x}) \left( \log q(\mathbf{x}_r) + \sum_{i \neq r} \log q(x_i | x_{\pi_i}) \right) \\ &= \sum_{\mathbf{x}} q(\mathbf{x}) \left( \log q(\mathbf{x}_r) + \sum_{i \neq r} \log \frac{q(x_i, x_{\pi_i})}{q(x_{\pi_i})} \right) \\ &= \sum_{\mathbf{x}} q(\mathbf{x}) \sum_{i \neq r} \log \frac{q(x_i, x_{\pi_i})}{q(x_i)q(x_{\pi_i})} + \sum_{\mathbf{x}} q(\mathbf{x}) \sum_i \log q(\mathbf{x}_i) \end{aligned}$$

- NB: second term does not depend on structure.

## KRUSKAL'S ALGORITHM (GREEDY SEARCH)

- To find the maximum weight spanning tree  $A$  on a graph with nodes  $U$  and weighted edges  $E$ :

- $A \leftarrow$  empty
- Sort edges  $E$  by nonincreasing weight:  $e_1, e_2, \dots, e_K$ .
- for  $k = 1$  to  $K$   $\{A += e_k$  unless doing so creates a cycle $\}$



## EDGE WEIGHTS

- Each term in sum  $i \neq r$  corresponds to an edge from  $i$  to its parent.

$$\begin{aligned} \frac{\ell^*}{N} &= \sum_{\mathbf{x}} q(\mathbf{x}) \sum_{i \neq r} \log \frac{q(x_i, x_{\pi_i})}{q(x_i)q(x_{\pi_i})} + C \\ &= \sum_{i \neq r} \sum_{x_i, x_{\pi_i}} q(x_i, x_{\pi_i}) \log \frac{q(x_i, x_{\pi_i})}{q(x_i)q(x_{\pi_i})} + C \\ &= \sum_{i \neq r} \sum_{y_i} q(y_i) \log \frac{q(y_i)}{q(x_i)q(x_{\pi_i})} + C \\ &= \sum_{i \neq r} W(i; \pi_i) + C \end{aligned}$$

where the edge weights  $W$  are defined by *mutual information*:

$$W(i; j) = \sum_{x_i, x_j} q(x_i, x_j) \log \frac{q(x_i, x_j)}{q(x_i)q(x_j)}$$

- So overall likelihood is sum of weights on edges that we use. We need the maximum weight spanning tree.

## MAXIMUM LIKELIHOOD TREES

We can now completely solve the tree learning problem:

- Compute the marginal counts  $q(x_i)$  for each node and pairwise counts  $q(x_i, x_j)$  for all pairs of nodes.
- Set the weights to the mutual informations:

$$W(i; j) = \sum_{x_i, x_j} q(x_i, x_j) \log \frac{q(x_i, x_j)}{q(x_i)q(x_j)}$$

- Find the maximum weight spanning tree  $A = \text{MWST}(W)$ .
- Using the undirected tree  $A$  chosen by MWST, pick a root arbitrarily and orient the edges away from the root. Set the conditional functions to the observed frequencies:

$$p(x_i | x_{\pi_i}) = \frac{q(x_i, x_{\pi_i})}{\sum_{x_i} q(x_i, x_{\pi_i})} = \frac{q(x_i, x_{\pi_i})}{q(x_{\pi_i})}$$

- Any directed tree consistent with the undirected tree found by the algorithm above will assign the same likelihood to any dataset.
- Amazingly, as far as likelihood goes, the root is arbitrary. We can just pick one node and orient the edges away from it. Or we can work with undirected models.
- For continuous nodes (e.g. Gaussian), the situation is similar, except that computing the mutual information requires an integral.
- Mutual information is the *Kullback-Leibler* divergence (cross-entropy) between a distribution and the product of its marginals. Measures how far from independent the joint distribution is.

$$W(i; j) = I[x_i; x_j] = \text{KL}[q(x_i, x_j) || q(x_i)q(x_j)]$$

---

### BAYESIAN MODEL AVERAGING (K&F 14.5)

---

- So far, we have just tried to find the mode of  $P(G|D)$ , i.e., the best scoring network.
- But the mode may be untypical of the distribution: most of the mass may be elsewhere.
- Suppose we are trying to determine if there is an edge  $X \rightarrow Y$  in the “true” model.
- We can compute features like this using

$$P(f|D) = \sum_G f(G)P(G|D)$$

where  $P(G|D) \propto P(D|G)P(G) \propto \prod_i \exp \text{FamScore}(D(X_i, \Pi_i))$ .

- The main problem is that there are  $2^{\Theta(n^2)}$  DAGs on  $n$  nodes.
- Even if we restrict indegree to  $\leq d$ , there are still  $2^{\Theta(dn \log n)}$  DAGs.

- Mixtures of trees - add hidden variables
- General graphs - local search

---

### MCMC FOR FEATURE PROBABILITY

---

- Suppose we can find a set  $G'$  of high-scoring networks. Then

$$P(f|D) \approx \frac{\sum_{G \in G'} P(G|D)f(G)}{\sum_{G \in G'} P(G|D)}$$

- If we can uniformly sample graphs from  $P(G|D)$ , we can approximate this using

$$P(f|D) \approx \frac{1}{T} f(G_t)$$

where  $G_k$  is the  $k$ 'th sample.

- Markov chain Monte Carlo (MCMC) provides a way of sampling from complex distributions such as this.

## MCMC

---

- We define a Markov chain on graph structures (in this case) with transition probability given by the Metropolis-Hastings rule

$$P(G'|G) = \min \left( 1, \frac{P(G'|D)Q(G|G')}{P(G|D)Q(G'|G)} \right)$$

where  $Q(G'|G)$  is the *proposal probability* and the ratio is the *acceptance probability*.

- The proposal  $Q$  has to be such that the Markov chain is ergodic, i.e., we can get to any state from any other state.
- We start the chain off in some initial state and then perform a random walk according to the above dynamics.
- Theory shows the stationary distribution of such a Markov chain is  $P(G|D)$ .

## MCMC FOR DAG STRUCTURE

---

- Suppose the proposal  $Q$  picks randomly from the following operators (where legal): add an edge, delete an edge, reverse an edge.
- The MH acceptance probability requires computing the Bayes factor  $P(G'|D)/P(G|D)$ , which is efficient for decomposable scores.
- However, small changes to the graph can result in large changes to the score, resulting in a jagged landscape.
- So the chain does not mix rapidly (it gets stuck in local optima).

## MCMC CONVERGENCE

---

- The *mixing time* is how long it takes the chain to converge from a random starting point.
- Once the chain has converged (after the *burnin*), we can draw (correlated) samples from  $P(G|D)$ .
- We can diagnose convergence by running the chain from multiple starting points and comparing the results. (Diagnosing convergence is an open problem.)

## RAO-BLACKWELLISED MCMC

---

- An alternative idea is to do MCMC sampling in the space of node orderings  $\prec$ , which “only” has size  $n!$ .
- Given an ordering, we can sum over all graphs efficiently (see below). Hence

$$P(f|D) \approx \frac{1}{T} P(f|D, \prec_t)$$

- This combination of sampling and exact integration/ marginalization is called Rao-Blackwellised sampling.
- This is named after the Rao-Blackwell theorem, which says (roughly) that variance is reduced if you sample in a smaller space:

$$\text{Var} E [E[f(G) | \prec]] \leq \text{Var} E[f(G)]$$

## MCMC OVER ORDERINGS

- We use Metropolis-Hastings as before.
- One proposal is to flip 2 variables in the order, leaving the rest unchanged:  
 $(X_{i_1}, \dots, \mathbf{X}_{i_j}, \dots, \mathbf{X}_{i_k}, \dots, X_{i_n}) \rightarrow (X_{i_1}, \dots, \mathbf{X}_{i_k}, \dots, \mathbf{X}_{i_j}, \dots, X_{i_n})$
- Using score decomposability, only family scores for nodes inside the bold range need to be recomputed.
- This is much more expensive than MCMC in DAG space, but each move is much more powerful, and the space is much smaller.

## PROB. FEATURE GIVEN KNOWN NODE ORDERING

- Given a sampled ordering, we can compute the probability of a parent set

$$P(\pi_i^G = U | D, \prec) = \frac{\exp \text{FamScore}(D(X_i, U))}{\sum_{U' \in U_{i, \prec}} \exp \text{FamScore}(D(X_i, U'))}$$

- From this, we can sample parents and hence graphs compatible with  $\prec$ .
- From this, we can compute probability of features such as “There is a directed path from  $X_i$  to  $X_j$ ”.
- Useful for determining features of biological networks from small sets of data.

## MARGINAL LIKELIHOOD GIVEN KNOWN NODE ORDERING

- If we know the ordering (eg. temporal), we have

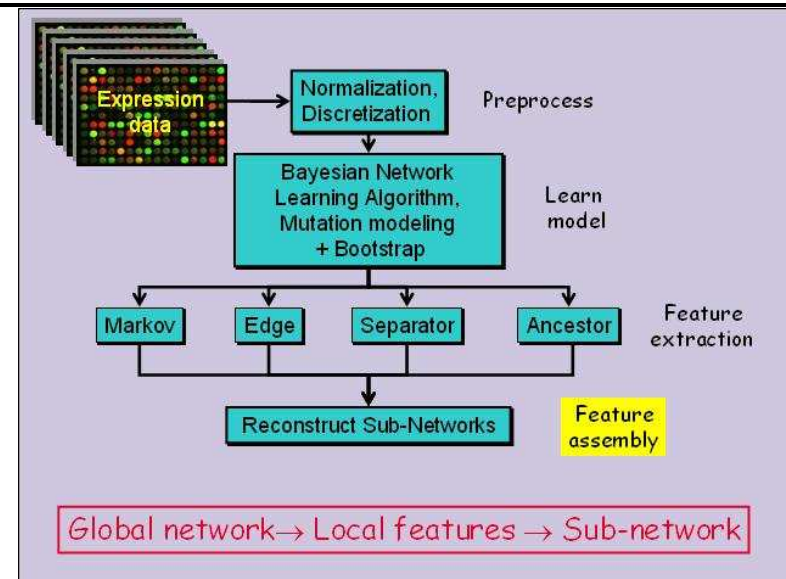
$$P(D | \prec) = \sum_{G \in G_{d, \prec}} P(G | \prec) P(D | G)$$

- Given  $\prec$ , we can pick the parents for each node independently. Let  $U_{i, \prec} = \{U : U \prec X_i, |U| \leq d\}$ . Assuming  $P(G | \prec)$  is uniform for legal graphs,

$$\begin{aligned} P(D | \prec) &= \sum_{G \in G_{d, \prec}} \prod_i \exp \text{FamScore}(D(X_i, \pi_i)) \\ &= \prod_i \sum_{U_i \in U_{i, \prec}} \exp \text{FamScore}(D(X_i, \pi_i)) \end{aligned}$$

- We marginalize out parameters  $\theta$  and graph structures  $G$ .
- This is what we need to evaluate the MH acceptance probability.

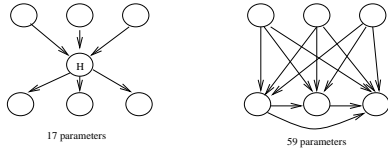
## LEARNING GENE REGULATORY PATHWAYS



(Slide from Nir Friedman)

## HIDDEN VARIABLES (K&F 15.7)

- So far, we have assumed all variables have been observed.
- In this case, we can compute the Bayesian score (evidence) exactly.
- But hidden variables can simplify a model a lot eg. mixture models, HMMs.



- Can still run local search to pick best model.
- But hidden variables raise various problems:
  - Efficiently computing the score from partially observed data.
  - Detecting the presence of latent (confounding) factors.
  - Inferring the dimensionality/ cardinality of latent factors.

## STRUCTURAL EM ALGORITHM (K&F 15.6)

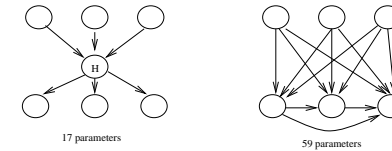
- Assume the number of hidden variables is given. Let  $y$  be the observed nodes,  $s$  be hidden, and  $z = (x, y)$  be all nodes.
- We can compute the BIC score for each candidate structure  $G'$  by applying EM to each one:

$$\text{score}_{BIC}(G'|y) = \log P(y|G', \hat{\theta}) - \frac{d(G)}{2} \log N$$

- But this is very expensive.
- Idea of structural EM: use current model  $(G, \theta)$  to compute the expected sufficient statistics (ESS) needed to evaluate each neighbor  $G'$ , i.e., compute the expected BIC score.
- This requires computing ESS for nodes and potentially new parents; such sets may not reside inside a clique of the jtree for  $G$ .
- Application: phylogenetic trees.

## DETECTING PRESENCE OF HIDDEN VARIABLES

- One idea is to look for dense semi-cliques.



- Then insert a hidden variable “in the middle”, and let the search algorithm figure out the detailed “wiring”.
- Unfortunately, many scoring criteria (e.g., BIC) produce very sparse graphs, which makes such semi-cliques rare.
- Constraint-based methods sometimes can be used to detect confounding.
- In general, this is an open problem.

## APPROXIMATING THE EVIDENCE IN LATENT VARIABLE MODELS

- When there are hidden variables, the parameter posterior has an exponential number of modes.
- Hence computing the marginal likelihood is intractable.
- There are various possible approximations:
  - Laplace
  - BIC
  - Cheeseman-Stutz (CS) lower bound
  - Variational Bayes EM lower bound
  - Sampling

## EXPECTATION-MAXIMIZATION (EM) ALGORITHM

- EM is an optimization strategy for objective functions that can be interpreted as likelihoods in the presence of missing data.
- It is much simpler than gradient methods:
  - No need to choose step size.
  - Enforces constraints automatically.
  - Calls inference and fully observed learning as subroutines.
- EM is an iterative algorithm with two linked steps:
  - E-step: fill-in hidden values using inference,  $p(\mathbf{z}|\mathbf{x}, \theta^t)$ .
  - M-step: update parameters  $\theta^{t+1}$  using standard MLE/MAP method applied to completed data
- We will prove that this procedure monotonically improves  $\ell$  (or leaves it unchanged). Thus it always converges to a local optimum of the likelihood.

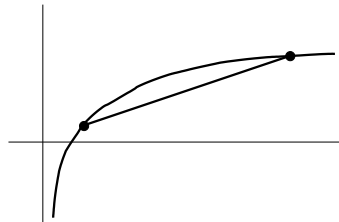
### EXPECTED COMPLETE LOG LIKELIHOOD

- For any distribution  $q(\mathbf{z})$  define *expected complete log likelihood*:

$$\ell_q(\theta; \mathbf{x}) = \langle \ell_c(\theta; \mathbf{x}, \mathbf{z}) \rangle_q \equiv \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{x}, \mathbf{z}|\theta)$$

- Amazing fact:  $\ell(\theta) \geq \ell_q(\theta) + \mathcal{H}(q)$  because of concavity of log:

$$\begin{aligned} \ell(\theta; \mathbf{x}) &= \log p(\mathbf{x}|\theta) \\ &= \log \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}|\theta) \\ &= \log \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x})} \\ &\geq \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x})} \end{aligned}$$



- Where the inequality is called *Jensen's inequality*.  
(It is only true for distributions:  $\sum q(\mathbf{z}) = 1$ ;  $q(\mathbf{z}) > 0$ .)

## COMPLETE & INCOMPLETE LOG LIKELIHOODS

- Observed variables  $\mathbf{x}$ , latent variables  $\mathbf{z}$ , parameters  $\theta$ :

$$\ell_c(\theta; \mathbf{x}, \mathbf{z}) = \log p(\mathbf{x}, \mathbf{z}|\theta)$$

is the *complete log likelihood*.

- Usually optimizing  $\ell_c(\theta)$  given both  $\mathbf{z}$  and  $\mathbf{x}$  is straightforward. (e.g. class conditional Gaussian fitting, linear regression)
- With  $\mathbf{z}$  unobserved, we need the log of a marginal probability:

$$\ell(\theta; \mathbf{x}) = \log p(\mathbf{x}|\theta) = \log \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}|\theta)$$

which is the *incomplete log likelihood*.

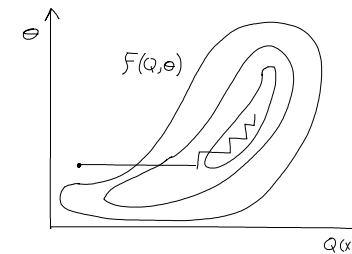
### LOWER BOUNDS AND FREE ENERGY

- For fixed data  $\mathbf{x}$ , define a functional called the *free energy*:

$$F(q, \theta) \equiv \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x})} \leq \ell(\theta)$$

- The EM algorithm is coordinate-ascent on  $F$ :

$$\begin{aligned} \mathbf{E}\text{-step:} & \quad q^{t+1} = \operatorname{argmax}_q F(q, \theta^t) \\ \mathbf{M}\text{-step:} & \quad \theta^{t+1} = \operatorname{argmax}_{\theta} F(q^{t+1}, \theta^t) \end{aligned}$$



## M-STEP: MAXIMIZATION OF EXPECTED $\ell_c$

- Note that the free energy breaks into two terms:

$$\begin{aligned} F(q, \theta) &= \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x})} \\ &= \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{x}, \mathbf{z}|\theta) - \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log q(\mathbf{z}|\mathbf{x}) \\ &= \ell_q(\theta; \mathbf{x}) + \mathcal{H}(q) \end{aligned}$$

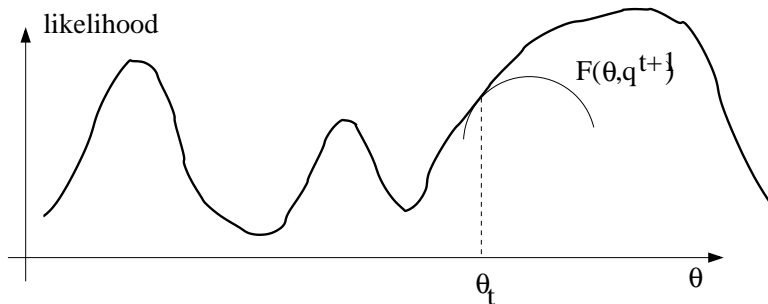
(this is where its name comes from)

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

$$\theta^{t+1} = \operatorname{argmax}_{\theta} \ell_q(\theta; \mathbf{x}) = \operatorname{argmax}_{\theta} \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{x}, \mathbf{z}|\theta)$$

## EM CONSTRUCTS SEQUENTIAL CONVEX LOWER BOUNDS

- Consider the likelihood function and the function  $F(q^{t+1}, \cdot)$ .



## E-STEP: INFERRING LATENT POSTERIOR

- Claim: the optimum setting of  $q$  in the E-step is:

$$q^{t+1} = p(\mathbf{z}|\mathbf{x}, \theta^t)$$

- 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 saturates the bound  $\ell(\theta; \mathbf{x}) \geq F(q, \theta)$

$$\begin{aligned} F(p(\mathbf{z}|\mathbf{x}, \theta^t), \theta^t) &= \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}, \theta^t) \log \frac{p(\mathbf{z}|\mathbf{x}, \theta^t)p(\mathbf{x}|\theta^t)}{p(\mathbf{z}|\mathbf{x}, \theta^t)} \\ &= \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}, \theta^t) \log p(\mathbf{x}|\theta^t) \\ &= \log p(\mathbf{x}|\theta^t) \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}, \theta^t) \\ &= \ell(\theta; \mathbf{x}) \cdot 1 \end{aligned}$$

- Can also show this result using variational calculus or the fact that  $\ell(\theta) - F(q, \theta) = \text{KL}[q||p(\mathbf{z}|\mathbf{x}, \theta)]$

## RECAP: EM ALGORITHM

- A way of maximizing likelihood function for latent variable models. Finds ML parameters when the original (hard) problem can be broken up into two (easy) pieces:
  - Estimate some “missing” or “unobserved” data from observed data and current parameters.
  - Using this “complete” data, find the maximum likelihood parameter estimates.
- Alternate between filling in the latent variables using our best guess (posterior) and updating the parameters based on this guess:
  - E-step:**  $q^{t+1} = p(\mathbf{z}|\mathbf{x}, \theta^t)$
  - M-step:**  $\theta^{t+1} = \operatorname{argmax}_{\theta} \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{x}, \mathbf{z}|\theta)$
- In the M-step we optimize a lower bound on the likelihood. In the E-step we close the gap, making bound=likelihood.

## VARIATIONAL BAYES EM (VBEM) ALGORITHM

- Latent variables are now  $x$  and parameters  $\theta$ , observations are  $y$ .
- Goal: Maximize lower bound on marginal likelihood  $P(y)$ .
- Key assumption: assume a factorized posterior  $q(x, \theta) \approx q_x(x)q_\theta(\theta)$ :

$$\log p(y) \geq \int q_x(x)q_\theta(\theta) \log \frac{p(y, x, \theta)}{q_x(x)q_\theta(\theta)} dx d\theta \stackrel{\text{def}}{=} F(q_x(x), q_\theta(\theta), y)$$

- Replaces stochastic dependence between  $x$  and  $\theta$  with deterministic constraints on moments.

- VB E step:

$$q_x^{t+1}(x) \propto \exp \left[ \int \log p(x, y|\theta) q_\theta^t(\theta) d\theta \right]$$

- VB M step:

$$q_\theta^{t+1}(\theta) \propto p(\theta) \exp \left[ \int \log p(x, y|\theta) q_x^{t+1}(x) dx \right]$$

## VBEM IN PRACTICE

- E-step: Do inference as usual, but use parameters  $\tilde{\theta}$  s.t.  $\phi(\tilde{\theta}) = \bar{\phi}$  (expected natural parameters)
- M-step: update hyper-parameters using expected sufficient statistics.
- The normalizing constant of inference is a lower bound on  $p(y)$ .
- Examples: HMMs, factor analysis (PCA), linear dynamical systems
- Variational message passing (VMP) is a way of implementing VBEM for any conjugate-exponential model, but makes the additional mean-field approximation that  $q(x) = \prod_k q(x_k)$ .

## CONJUGATE EXPONENTIAL MODELS

- Assumption 1: the complete-data log-likelihood is that of an exponential family:

$$p(x, y|\theta) = f(x, y)g(\theta) \exp(\phi(\theta)^T u(x, y))$$

- Assumption 2: the parameter prior is conjugate to the likelihood:

$$p(\theta|\eta, \nu) = h(\eta, \nu)g(\theta)^\eta \exp(\phi(\theta)^T \nu)$$

- Thm: at every step of VBEM, the parameter posterior is

$$q(\theta|\eta + n, \nu + \sum_{i=1}^n \bar{u}(y_i)), \text{ where } \bar{u}(y_i) = E_{q_{x_i}} u(x_i, y_i)$$

and the latent variable posterior is  $q_x(x) = \prod_i q_{x_i}(x_i)$  where

$$q_{x_i}(x_i) = p(x_i|y, \bar{\phi}) \propto f(x_i, y_i) \exp \left[ \bar{\phi}^T u(x_i, y_i) \right] \text{ where } \bar{\phi} = E_{q_\theta} \phi(\theta)$$

## VBEM FOR MODEL SELECTION

