

LECTURE 12:

PARAMETER LEARNING FOR BNS WITH HIDDEN NODES

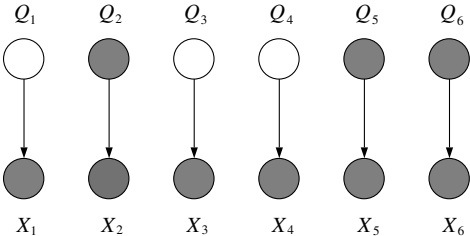
Kevin Murphy

Wed 27 Oct 2004

- Latent variable models
- Maximum likelihood learning
  - Gradient ascent
  - EM
- (Bayesian learning)

UNOBSERVED VARIABLES

- Certain variables  $Q$  in our models may be *unobserved*, either some of the time or always, either at training time or at test time.



Graphically, we will use shading to indicate observation.

- We shall assume values are *missing at random*; if not, we need to model the *data censoring mechanism* explicitly.
- e.g., probability we observe a medical record may depend on values of the other attributes.

PARTIALLY UNOBSERVED (MISSING) VARIABLES

- If variables are occasionally unobserved they are *missing data*. e.g. undefined inputs, missing class labels, erroneous target values
- Variables which are always unobserved are called *latent variables*.
- Now we maximize the likelihood of the observed data; we have to *sum out* or *marginalize* the missing values at training or test time:

$$\begin{aligned} \ell(\theta; \mathcal{D}) &= \sum_{\text{complete}} \log p(\mathbf{x}^c, \mathbf{y}^c | \theta) + \sum_{\text{missing}} \log p(\mathbf{x}^m | \theta) \\ &= \sum_{\text{complete}} \log p(\mathbf{x}^c, \mathbf{y}^c | \theta) + \sum_{\text{missing}} \log \sum_{\mathbf{z}} p(\mathbf{x}^m, \mathbf{z} | \theta) \end{aligned}$$

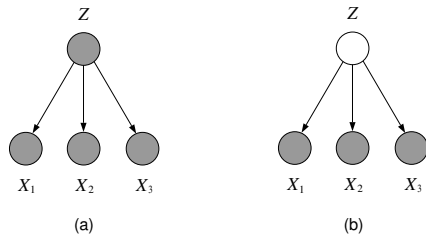
## WHY IS LEARNING HARDER?

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

$$\ell(\theta; \mathcal{D}) = \log p(\mathbf{x}, \mathbf{z}|\theta) = \log p(\mathbf{z}|\theta_z) + \log p(\mathbf{x}|\mathbf{z}, \theta_x)$$

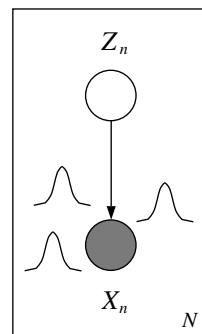
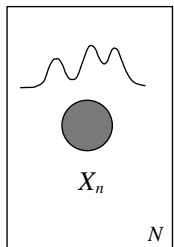
- With latent variables, all the parameters become coupled together via  $\log \sum()$ :

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



## MIXTURE MODELS

- A density model  $p(x)$  may be multi-modal.
- We may be able to model it as a mixture of uni-modal distributions (e.g., Gaussians).
- Each mode may correspond to a different sub-population (e.g., male and female).

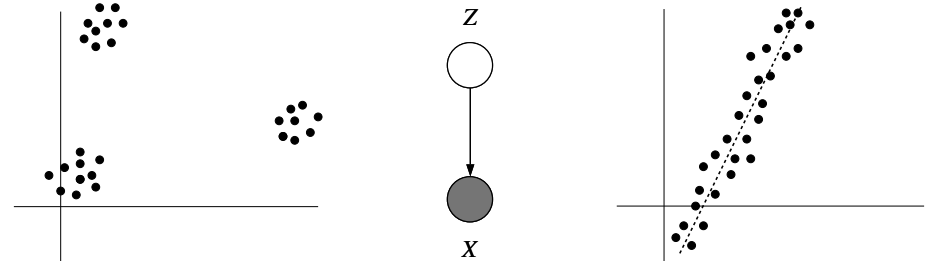


(a)

(b)

## WHERE DO LATENT VARIABLES COME FROM?

- Missing values can arise because something wasn't measured, because of faulty sensors, etc.
- But we may also *intentionally* introduce latent variables to simplify a model.
- Discrete latent variables can be used to partition/ cluster data into sub-groups.
- Continuous latent variables (factors) can be used for dimensionality reduction (PCA, etc).

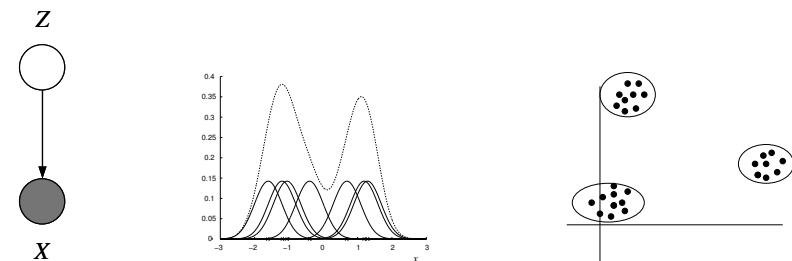


## GAUSSIAN MIXTURE MODELS (GMMs)

- Consider a mixture of  $K$  Gaussian components:

$$p(Z = k) = \alpha_k \text{ mixing weights}$$

$$p(\mathbf{x}|z = k) = \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k) \text{ class-conditional densities}$$



- This 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

- The posterior probability that a data point  $x$  is assigned to cluster  $k$  is given by

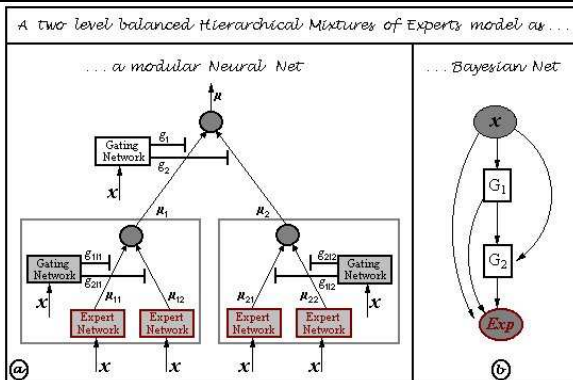
$$p(z = k | \mathbf{x}, \theta) = \frac{\alpha_k p_k(\mathbf{x} | \theta_k)}{\sum_j \alpha_j p_j(\mathbf{x} | \theta_j)} = r_k(x)$$

- These quantities are called *responsibilities* or *soft assignments*.
- Vector quantization (VQ)* = hard assignment of  $x$  to the most probable cluster.
- The log-likelihood is

$$\ell(\theta; \mathcal{D}) = \sum_n \log p(\mathbf{x}^n) = \sum_n \log \sum_k \alpha_k \mathcal{N}(\mathbf{x}^n | \mu_k, \Sigma_k)$$

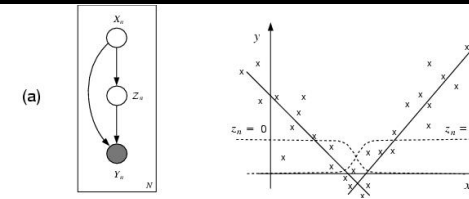
- We will discuss how to maximize this later.

## HIERARCHICAL MIXTURE OF EXPERTS



- This is like a soft version of a depth-2 classification/ regression tree.
- $P(Y|X, Z_1, Z_2)$  can be modelled as a GLIM, with parameters dependent on the values of  $Z_1$  and  $Z_2$  (which specify the path to a given leaf in the tree).

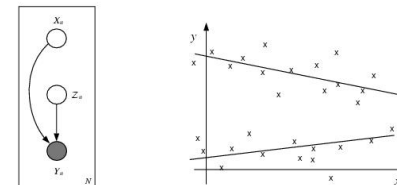
## CONDITIONAL MIXTURE MODEL: MIXTURE OF EXPERTS



- We will model  $P(Y|X)$  using different experts, each responsible for different regions of the input space.
- Latent variable  $Z$  chooses expert using softmax gating function:  
 $P(Z = k | x) = \sigma(\theta_k^T x)$ .
- Each expert can be a linear regression model:  
 $p(y|x, Z = k) = \mathcal{N}(y; \beta_k^T x, \sigma_k^2)$ .
- The posterior expert responsibilities are

$$p(z = k | \mathbf{x}, \mathbf{y}, \theta) = \frac{\alpha_k(\mathbf{x}) p_k(\mathbf{y} | \mathbf{x}, \theta_k)}{\sum_j \alpha_j(\mathbf{x}) p_j(\mathbf{y} | \mathbf{x}, \theta_j)}$$

## MIXTURE OF OVERLAPPING EXPERTS



- By removing the  $X \rightarrow Z$  arc, we can make the partitions independent of the input, thus allowing overlap.
- This is a mixture of linear regressors; each subpopulation has a different conditional mean.

- We can learn mixture densities using gradient descent on the log likelihood. The gradients are quite interesting:

$$\begin{aligned} \ell(\theta) &= \log p(\mathbf{x}|\theta) = \log \sum_k \alpha_k p_k(\mathbf{x}|\theta_k) \\ \frac{\partial \ell}{\partial \theta} &= \frac{1}{p(\mathbf{x}|\theta)} \sum_k \alpha_k \frac{\partial p_k(\mathbf{x}|\theta_k)}{\partial \theta} \\ &= \sum_k \alpha_k \frac{1}{p(\mathbf{x}|\theta)} p_k(\mathbf{x}|\theta_k) \frac{\partial \log p_k(\mathbf{x}|\theta_k)}{\partial \theta} \\ &= \sum_k \alpha_k \frac{p_k(\mathbf{x}|\theta_k)}{p(\mathbf{x}|\theta)} \frac{\partial \ell_k}{\partial \theta_k} = \sum_k r_k \frac{\partial \ell_k}{\partial \theta_k} \end{aligned}$$

- In other words, the gradient is the *responsibility weighted sum* of the individual log likelihood gradients.
- Can pass this to a conjugate gradient routine.

LOGSUM

- You may encounter numerical problems when computing the log-likelihood:

$$\ell(\theta) = \log p(\mathbf{x}|\theta) = \log \sum_k \alpha_k p_k(\mathbf{x}|\theta_k)$$

since  $p_k(\mathbf{x}|\theta_k)$  may be extremely small.

- The class conditional *log* likelihoods are well-behaved:  
 $b_k = \log p_k(\mathbf{x}|\theta_k)$ .
- But the following will underflow:

$$\log \sum_k e^{b_k}$$

- Often we have constraints on the parameters, e.g.  $\sum_k \alpha_k = 1$ ,  $\Sigma$  symmetric positive definite (hence  $\Sigma_{ii} > 0$ ).
- We can use constrained optimization, or we can reparameterize in terms of unconstrained values.
- For discrete variables, use the softmax transform:  $\alpha_k = \frac{\exp(q_k)}{\sum_j \exp(q_j)}$
- For covariance matrices, use the Cholesky decomposition:

$$\Sigma^{-1} = A^T A$$

where  $A$  is upper diagonal with positive diagonal:

$$A_{ii} = \exp(r_i) > 0 \quad A_{ij} = a_{ij} \quad (j > i) \quad A_{ij} = 0 \quad (j < i)$$

- The variables  $q_i, r_i, a_{ij} \in \mathbb{R}$  are unconstrained.
- Use chain rule to compute  $\frac{\partial \ell}{\partial \alpha}, \frac{\partial \ell}{\partial A}$ .

LOGSUM

- You should use

$$\begin{aligned} \log \sum_k e^{b_k} &= \log \left[ \left( \sum_k e^{b_k} \right) e^{-B} e^B \right] \\ &= \log \left[ \left( \sum_k e^{b_k - B} \right) e^B \right] \\ &= \left[ \log \left( \sum_k e^{b_k - B} \right) \right] + B \end{aligned}$$

where  $B = \max_k b_k$

- Example

$$\log(e^{-120} + e^{-121}) = \log \left( e^{-120} (e^0 + e^{-1}) \right) = \log(e^0 + e^{-1}) - 120$$

## GRADIENT LEARNING FOR BNs WITH TABULAR CPDs

- Let  $\theta_{ijk} = P(X_i = j | X_{\pi_i} = k)$ .
- For a fully observed case, the gradient of the likelihood is

$$\begin{aligned} \frac{\partial}{\partial \theta_{ijk}} P(x_{1:N}) &= \frac{\partial}{\partial \theta_{ijk}} \prod_{i'} \theta_{i', x_{i'}, x_{\pi_{i'}}} \\ &= \delta(x_i = j, x_{\pi_i} = k) \theta_{ijk} \prod_{i' \neq i} \theta_{i', x_{i'}, x_{\pi_{i'}}} \\ &= \delta(x_i = j, x_{\pi_i} = k) \frac{\prod_{i'} \theta_{i', x_{i'}, x_{\pi_{i'}}}}{\theta_{ijk}} \\ &= \delta(x_i = j, x_{\pi_i} = k) \frac{P(x_{1:N})}{\theta_{ijk}} \end{aligned}$$

## GRADIENT LEARNING FOR BNs WITH GENERAL CPDs

- Consider CPD  $P(X = x | X_{\pi} = u; \phi)$ .
- We have

$$\frac{\partial P(e)}{\partial P(X = x | X_{\pi} = u)} = \frac{P(X = x, X_{\pi} = u, e)}{P(X = x | X_{\pi} = u)}$$

- By the chain rule

$$\frac{\partial P(e)}{\partial \phi} = \sum_{x, u} \frac{\partial P(e)}{\partial P(X = x, X_{\pi} = u)} \frac{\partial P(X = x | X_{\pi} = u)}{\partial \phi}$$

- This can be used to learn noisy-OR, sigmoids, etc.

## GRADIENT LEARNING FOR BNs WITH TABULAR CPDs

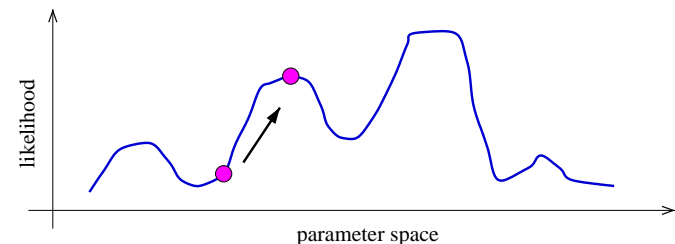
- For a partially observed case,

$$\begin{aligned} \frac{\partial}{\partial \theta_{ijk}} P(e) &= \frac{\partial}{\partial \theta_{ijk}} \sum_{x: x(E)=e} P(x) \\ &= \sum_x \delta(x_E = e) \delta(x_i = j, x_{\pi_i} = k) \frac{P(x_{1:N})}{\theta_{ijk}} \\ &= \frac{P(x_i = j, x_{\pi_i} = k, e)}{\theta_{ijk}} \end{aligned}$$

- A more complex expression can be used if  $\theta_{ijk} = 0$ .
- The numerator can be computed using probabilistic inference.
- A junction tree will always contain  $X_i$  and  $X_{\pi_i}$  in the same clique.
- For batch learning, we need to do inference for every training case for every iteration of gradient ascent (slow!).

## IDENTIFIABILITY

- A mixture model induces a multi-modal likelihood.
- Hence gradient ascent can only find a local maximum.
- Mixture models are unidentifiable, since we can always switch the hidden labels without affecting the likelihood.
- Hence we should be careful in trying to interpret the “meaning” of latent variables.



## 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.

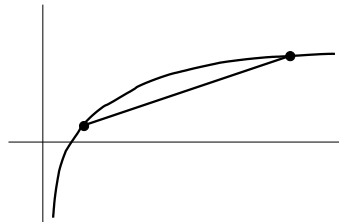
### EXPECTED COMPLETE LOG LIKELIHOOD

- 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*.

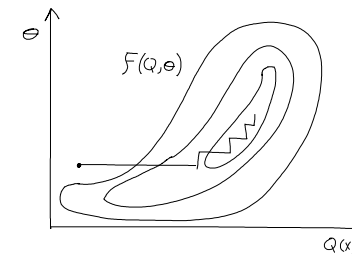
### LOWER BOUNDS AND FREE ENERGY

- 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:

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

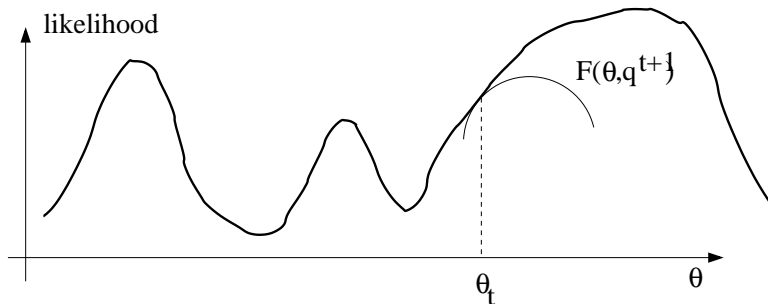
(this is where its name comes from)

- 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:
  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 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.

## EXAMPLE: MIXTURES OF GAUSSIANS

- Recall: a mixture of  $K$  Gaussians:

$$p(\mathbf{x}|\theta) = \sum_k \alpha_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

$$\ell(\theta; \mathcal{D}) = \sum_n \log \sum_k \alpha_k \mathcal{N}(\mathbf{x}^n|\mu_k, \Sigma_k)$$

- Learning with EM algorithm:

**E – step :**  $p_{kn}^t = \mathcal{N}(\mathbf{x}^n|\mu_k^t, \Sigma_k^t)$

$$q_{kn}^{t+1} = p(z=k|\mathbf{x}^n, \theta^t) = \frac{\alpha_k^t p_{kn}^t}{\sum_j \alpha_j^t p_{jn}^t}$$

**M – step :**

$$\mu_k^{t+1} = \frac{\sum_n q_{kn}^{t+1} \mathbf{x}^n}{\sum_n q_{kn}^{t+1}}$$

$$\Sigma_k^{t+1} = \frac{\sum_n q_{kn}^{t+1} (\mathbf{x}^n - \mu_k^{t+1})(\mathbf{x}^n - \mu_k^{t+1})^\top}{\sum_n q_{kn}^{t+1}}$$

$$\alpha_k^{t+1} = \frac{1}{M} \sum_n q_{kn}^{t+1}$$

## DERIVATION OF M-STEP

- Expected complete log likelihood  $\ell_q(\theta; \mathcal{D})$ :

$$\sum_n \sum_k q_{kn} \left[ \log \alpha_k - \frac{1}{2} (\mathbf{x}^n - \mu_k^{t+1})^\top \Sigma_k^{-1} (\mathbf{x}^n - \mu_k^{t+1}) - \frac{1}{2} \log |2\pi \Sigma_k| \right]$$

- For fixed  $q$  we can optimize the parameters:

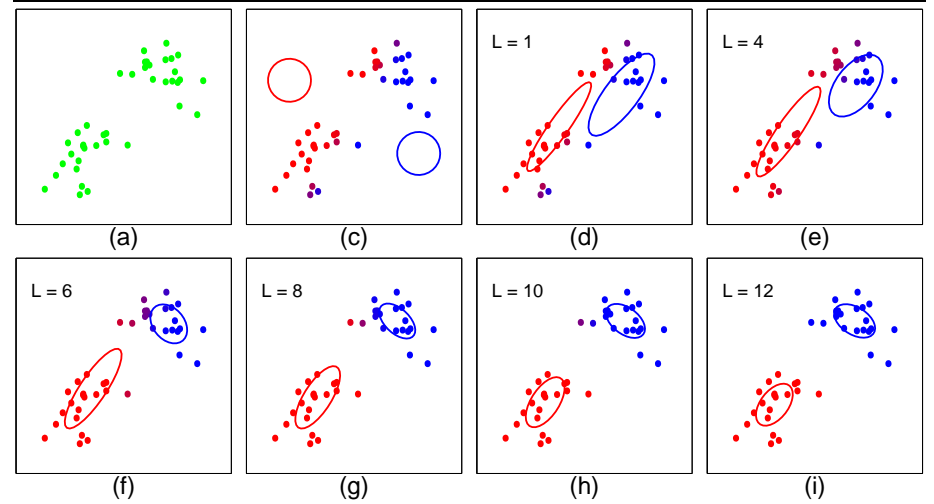
$$\frac{\partial \ell_q}{\partial \mu_k} = \Sigma_k^{-1} \sum_n q_{kn} (\mathbf{x}^n - \mu_k)$$

$$\frac{\partial \ell_q}{\partial \Sigma_k^{-1}} = \frac{1}{2} \sum_n q_{kn} \left[ \Sigma_k^\top - (\mathbf{x}^n - \mu_k^{t+1})(\mathbf{x}^n - \mu_k^{t+1})^\top \right]$$

$$\frac{\partial \ell_q}{\partial \alpha_k} = \frac{1}{\alpha_k} \sum_n q_{kn} - \lambda \quad (\lambda = M)$$

- Fact:  $\frac{\partial \log |A^{-1}|}{\partial A^{-1}} = A^\top$  and  $\frac{\partial \mathbf{x}^\top A \mathbf{x}}{\partial A} = \mathbf{x} \mathbf{x}^\top$

## EM FOR MOG



## COMPARE: K-MEANS

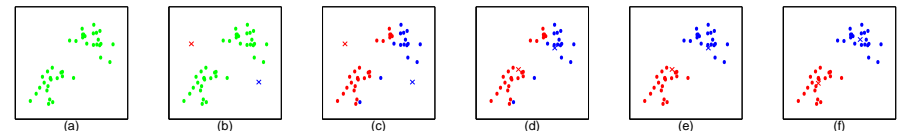
- The EM algorithm for mixtures of Gaussians is just like a soft version of the K-means algorithm.

- In the K-means “E-step” we do hard assignment:

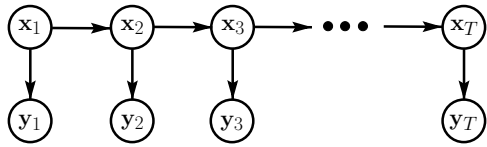
$$c_n^{t+1} = \operatorname{argmin}_k (\mathbf{x}^n - \mu_k^t)^\top \Sigma_k^{-1} (\mathbf{x}^n - \mu_k^t)$$

- In the K-means “M-step” we update the means as the weighted sum of the data, but now the weights are 0 or 1:

$$\mu_k^{t+1} = \frac{\sum_n [c_k^{t+1} = n] \mathbf{x}^n}{\sum_n [c_k^{t+1} = n]}$$



## REMINDER: HMM GRAPHICAL MODEL



- Hidden states  $\{x_t\}$ , outputs  $\{y_t\}$   
Joint probability factorizes:

$$\begin{aligned} P(\{\mathbf{x}\}, \{\mathbf{y}\}) &= \prod_{t=1}^T P(x_t | \mathbf{x}_{t-1}) P(y_t | x_t) \\ &= \pi_{\mathbf{x}_1} \prod_{t=1}^{T-1} S_{x_t, x_{t+1}} \prod_{t=1}^T A_{x_t}(y_t) \end{aligned}$$

- We saw efficient recursions for computing  
 $L = P(\{\mathbf{y}\}) = \sum_{\{\mathbf{x}\}} P(\{\mathbf{x}\}, \{\mathbf{y}\})$  and  $\gamma_i(t) = P(x_t = i | \{\mathbf{y}\})$ .

## PARAMETER ESTIMATION USING EM

- $S_{ij}$  are transition probs; state  $j$  has output distribution  $B_j(\mathbf{y})$

$$\begin{aligned} P(x_{t+1} = j | x_t = i) &= S_{ij} & P(x_1 = j) &= \pi_j \\ P(\mathbf{y}_t = y | x_t = j) &= B_j(y) \end{aligned}$$

- Complete log likelihood:

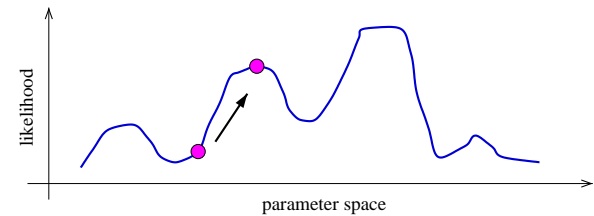
$$\begin{aligned} \log p(x, y) &= \log \left\{ \pi_{\mathbf{x}_1} \prod_{t=1}^{T-1} S_{x_t, x_{t+1}} \prod_{t=1}^T B_{x_t}(\mathbf{y}_t) \right\} \\ &= \log \left\{ \prod_i \pi_i^{[x_1^i]} \prod_{t=1}^{T-1} \prod_{ij} S_{ij}^{[x_t^i, x_{t+1}^j]} \prod_{t=1}^T \prod_k B_k(\mathbf{y}_t)^{[x_t^k]} \right\} \\ &= \sum_i [x_1^i] \log \pi_i + \sum_{t=1}^{T-1} \sum_{ij} [x_t^i, x_{t+1}^j] \log S_{ij} + \sum_{t=1}^T \sum_k [x_t^k] \log B_k(\mathbf{y}_t) \end{aligned}$$

where the indicator  $[x_t^i] = 1$  if  $x_t = i$  and 0 otherwise

- For EM, we need to compute the *expected complete log likelihood*.

## BAUM-WELCH ALGORITHM: EM TRAINING

1. Intuition: if only we *knew* the true state path then ML parameter estimation would be trivial (MM1 on  $x$ , conditional on  $y$ ).
2. But: can *estimate* state path using inference recursions.
3. *Baum-Welch algorithm* (special case of EM): estimate the states, then compute params, then re-estimate states, and so on ...
4. This works and we can *prove* that it always improves likelihood.
5. However: finding the ML parameters is NP complete, so initial conditions matter a lot and convergence is hard to tell.



## STATE EXPECTATIONS REQUIRED FROM THE E-STEP

- The expected complete log likelihood requires  
 $\gamma_i(t) = \langle [x_t^i] \rangle$  and  $\xi_{ij}(t) = \langle [x_t^i, x_{t+1}^j] \rangle$
- So in the E-step we need to compute both  
 $\gamma_i(t) = p(x_t = i | \{\mathbf{y}\})$  and  $\xi_{ij}(t) = p(x_t = i, x_{t+1} = j | \{\mathbf{y}\})$ .
- We can use the forwards-backwards (Shafer-Shenoy) or Lauritzen-Spiegelhalter algorithms.

## M-STEP: NEW PARAMETERS ARE JUST RATIOS OF FREQUENCY COUNTS

---

- Initial state distribution: expected #times in state  $i$  at time 1:

$$\hat{\pi}_i = \gamma_i(1)$$

- Expected #transitions from state  $i$  to  $j$  which begin at time  $t$ :

$$\xi_{ij}(t) = \alpha_i(t)S_{ij}B_j(\mathbf{y}_{t+1})\beta_j(t+1)/L$$

so the estimated transition probabilities are:

$$\hat{S}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

- The output distributions are the expected number of times we observe a particular symbol in a particular state:

$$\hat{B}_j(y) = \frac{\sum_{t:\mathbf{y}_t=y} \gamma_j(t)}{\sum_{t=1}^T \gamma_j(t)}$$

---

## EM FOR GENERAL BNS

while not converged

// E-step

for each node  $i$

$ESS_i = 0$  // reset expected sufficient statistics

for each case  $m$

do inference with  $e(m)$

for each node  $i$

$ESS_{i+} = SS(P(X_i, X_{\pi_i}|e_m))$

// M-step

for each node  $i$

$\theta_i := MLE(ESS_i)$

---

## PARTIALLY HIDDEN DATA

- Multiple observation sequences: can be dealt with by averaging numerators and averaging denominators in the ratios given above.
- Initialization: mixtures of Naive Bayes or mixtures of Gaussians for the output (observation) models, left-to-right for the transition matrix (if appropriate).
- Numerical scaling: the probability values  $P(X_t, y_{1:t})$  can get very small, so normalize to get  $P(X_t|y_{1:t})$  or use logsum trick.

- Of course, we can learn when there are missing (hidden) variables on some cases and not on others.
- In this case the cost function was:
$$\ell(\theta; \mathcal{D}) = \sum_{\text{complete}} \log p(\mathbf{x}^c, \mathbf{y}^c | \theta) + \sum_{\text{missing}} \log \sum_{\mathbf{y}} \log p(\mathbf{x}^m, \mathbf{y} | \theta)$$
- Now you can think of this in a new way: in the E-step we estimate the hidden variables on the incomplete cases only.
- The M-step optimizes the log likelihood on the complete data plus the expected likelihood on the incomplete data using the E-step.

- Sparse EM:  
Do not recompute exactly the posterior probability on each data point under all models, because it is almost zero. Instead keep an “active list” which you update every once in a while.
- Generalized (Incomplete) EM: It might be hard to find the ML parameters in the M-step, even given the completed data. We can still make progress by doing an M-step that improves the likelihood a bit (e.g. gradient step).

- Some good things about EM:
  - no learning rate (step-size) parameter
  - automatically enforces parameter constraints
  - very fast for low dimensions
  - each iteration guaranteed to improve likelihood
- Some bad things about EM:
  - can get stuck in local minima
  - can be slower than conjugate gradient (especially near convergence)
  - requires expensive inference step
  - is a maximum likelihood/MAP method