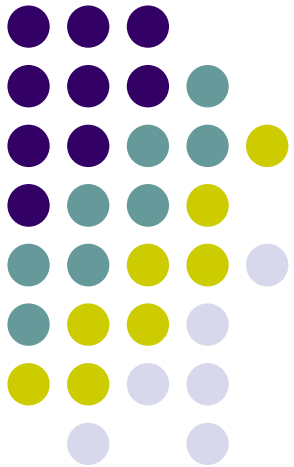
Machine Learning

10-701/15-781, Fall 2012

Expectation Maximization: Mixture model and HMM

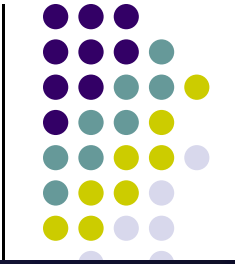


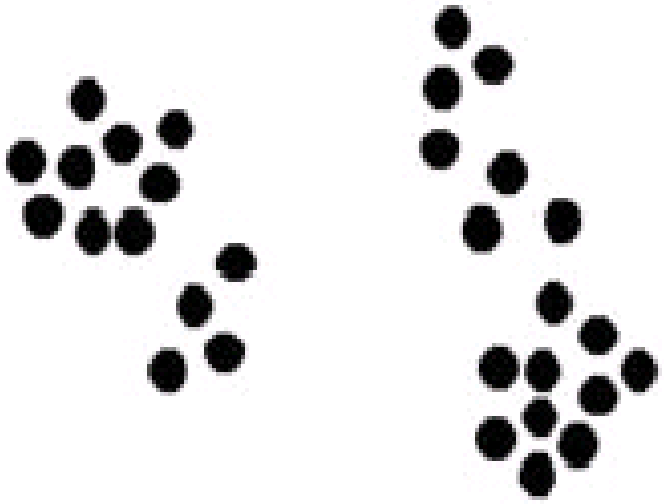
Eric Xing

Lecture 12, October 22, 2012

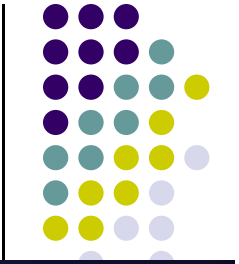
Reading: Chap. 9, 13, C.B book

Clustering and partially observable probabilistic models





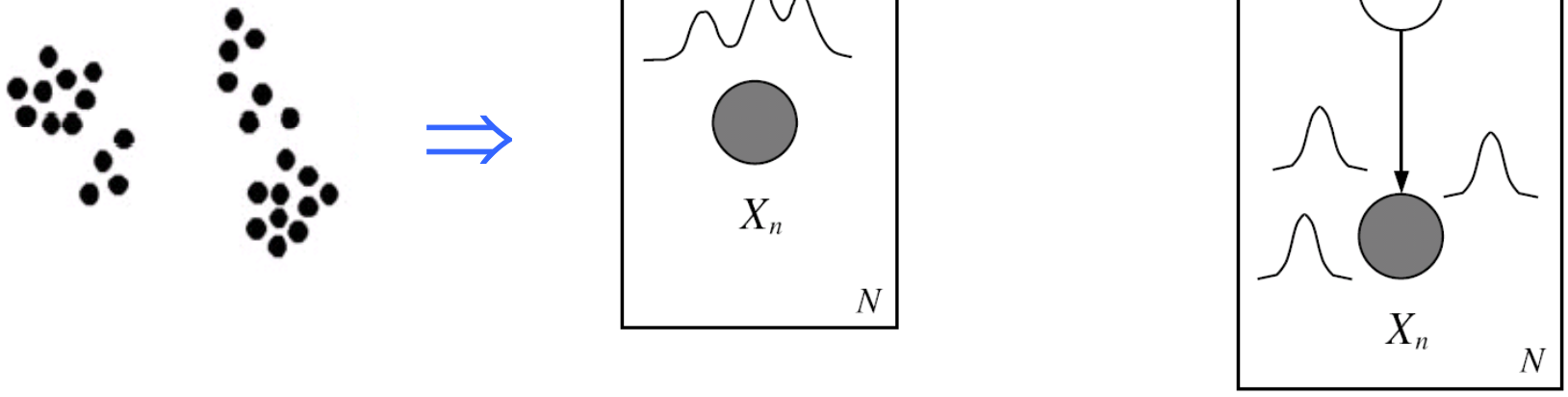
Unobserved Variables



- A variable can be unobserved (latent) because:
  - it is an imaginary quantity meant to provide some simplified and abstractive view of the data generation process
    - e.g., speech recognition models, mixture models …
  - it is a real-world object and/or phenomena, but difficult or impossible to measure
    - e.g., the temperature of a star, causes of a disease, evolutionary ancestors …
  - it is a real-world object and/or phenomena, but sometimes wasn't measured, because of faulty sensors; or was measure with a noisy channel, etc.
    - e.g., traffic radio, aircraft signal on a radar screen,
- Discrete latent variables can be used to partition/cluster data into sub-groups (mixture models, forthcoming).
- Continuous latent variables (factors) can be used for dimensionality reduction (factor analysis, etc., later lectures).

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).



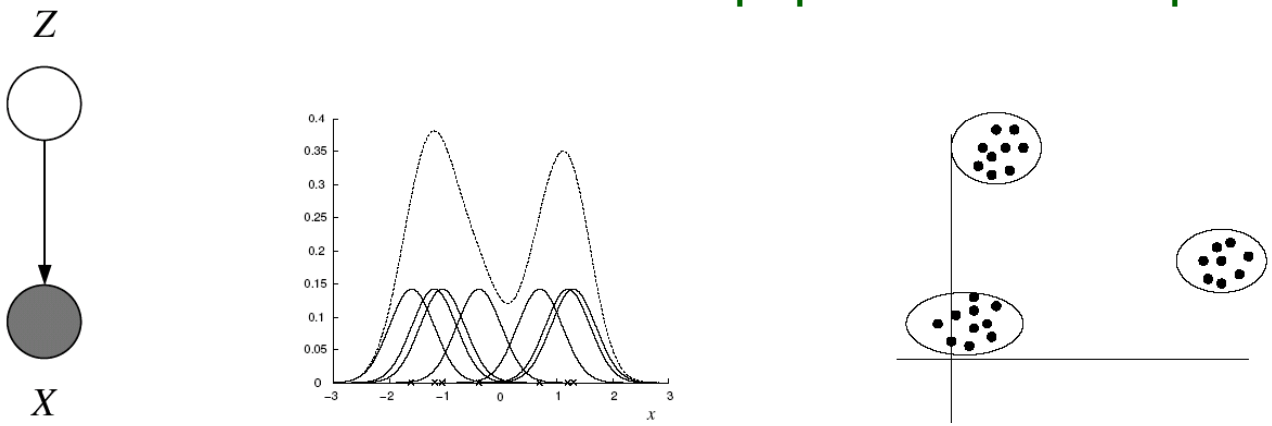
(a) (b)

Gaussian Mixture Models (GMMs)

- Consider a mixture of *K* Gaussian components:

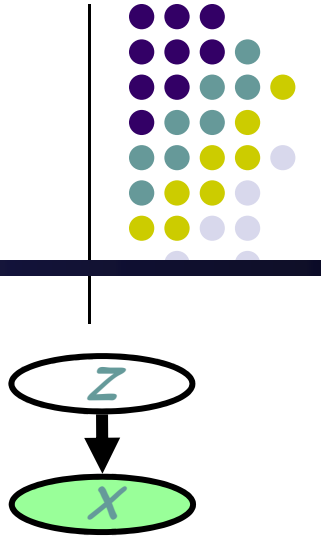
$$p(x_n \mid \mu, \Sigma) = \sum_k \pi_k N(x, \mid \mu_k, \Sigma_k)$$

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.

GGM derivations



- 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

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

  - The likelihood of a sample:

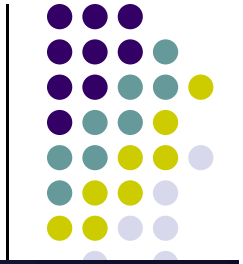
$$p(x_n \mid \mu, \Sigma) = \sum_k p(z^k = 1 \mid \pi) p(x, \mid z^k = 1, \mu, \Sigma)$$

$$= \sum_{z_n} \prod_k \left((\pi_k)^{z_n^k} \mathcal{N}(x_n : \mu_k, \Sigma_k)^{z_n^k} \right) = \sum_k \pi_k \mathcal{N}(x, \mid \mu_k, \Sigma_k)$$

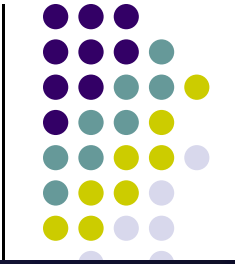
mixture component

mixture proportion

Learning mixture models



Why is Learning Harder?

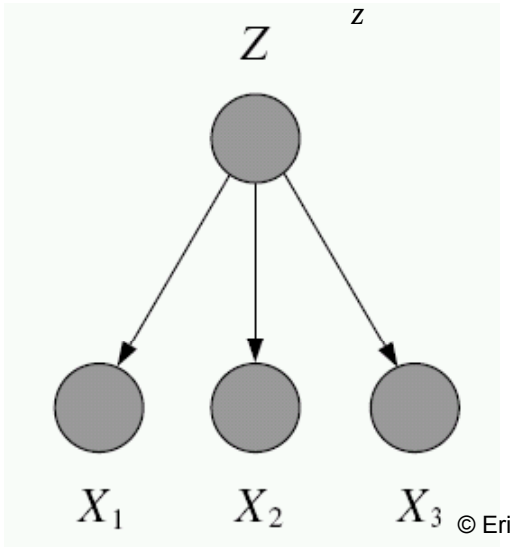


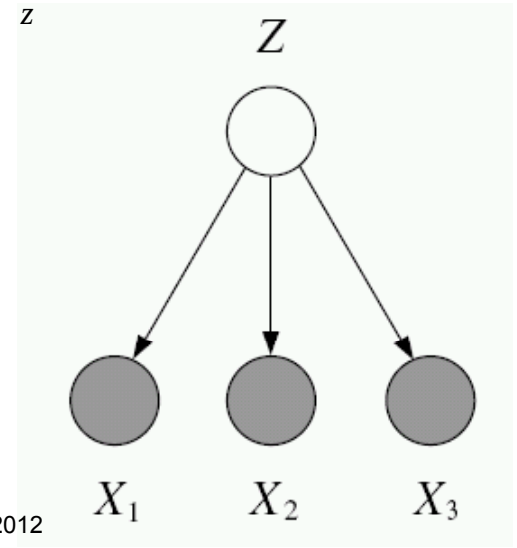
- In fully observed iid settings, the log likelihood decomposes into a sum of local terms.

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

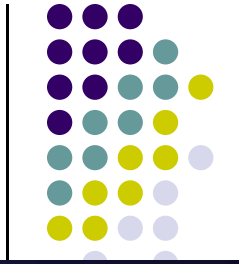
- With latent variables, all the parameters become coupled together via *marginalization*

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





Gradient Learning for mixture models



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

$$\ell(\theta) = \log p(\mathrm{x} \mid \theta) = \log \sum_k \pi_k p_k(\mathrm{x}|\theta_k)$$

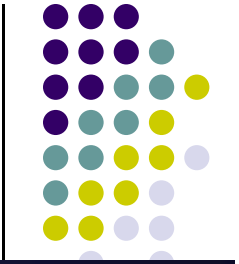
$$\frac{\partial \ell}{\partial \theta} = \frac{1}{p(\mathrm{x} \mid \theta)} \sum_k \pi_k \frac{\partial p_k(\mathrm{x}|\theta_k)}{\partial \theta}$$

$$= \sum_k \frac{\pi_k}{p(\mathrm{x} \mid \theta)} p_k(\mathrm{x}|\theta_k) \frac{\partial \log p_k(\mathrm{x}|\theta_k)}{\partial \theta}$$

$$= \sum_k \pi_k \frac{p_k(\mathrm{x}|\theta_k)}{p(\mathrm{x} \mid \theta)} \frac{\partial \log p_k(\mathrm{x}|\theta_k)}{\partial \theta_k} = \sum_k r_k \frac{\partial \ell_k}{\partial \theta_k}$$

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

Parameter Constraints



- Often we have constraints on the parameters, e.g. $\Sigma_k \pi_k = 1$, Σ being symmetric positive definite (hence $\Sigma_{ii} > 0$).
- We can use constrained optimization, or we can reparameterize in terms of unconstrained values.
  - For normalized weights, use the softmax transform:
  - For covariance matrices, use the Cholesky decomposition:

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

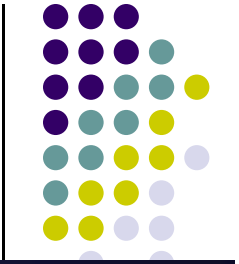
where A is upper diagonal with positive diagonal:

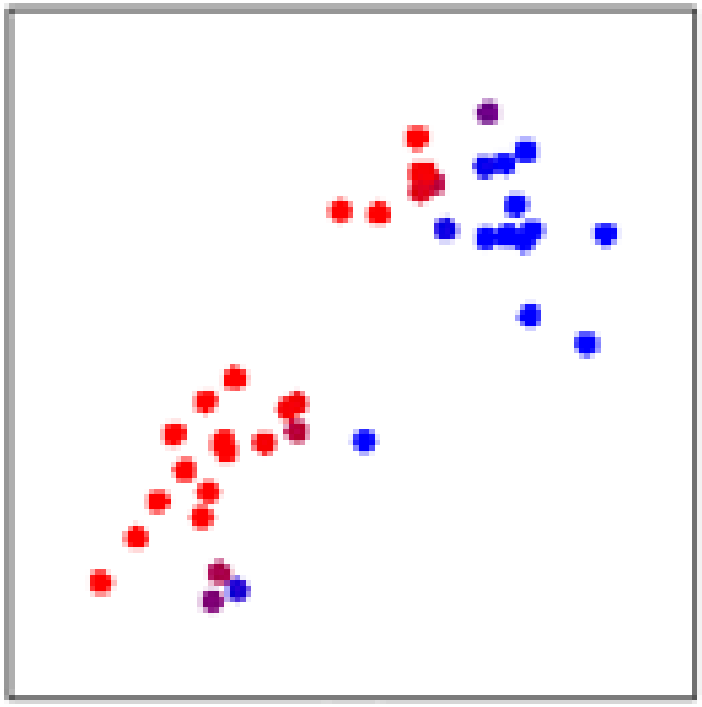
$$\mathbf{A}_{ii} = \exp(\lambda_i) > 0 \quad \mathbf{A}_{ij} = \eta_{ij} \ \ (j > i) \quad \mathbf{A}_{ij} = 0 \ \ (j < i)$$

the parameters $\gamma_i, \lambda_i, \eta_{ij} \in \mathbb{R}$ are unconstrained.

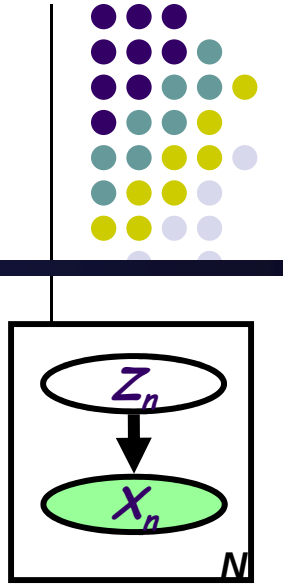
  - Use chain rule to compute $\frac{\partial \ell}{\partial \pi}, \frac{\partial \ell}{\partial \mathbf{A}}$.

The Expectation-Maximization (EM) Algorithm





EM algorithm for GMM



- E.g., A mixture of K Gaussians:
  - Z is a latent class indicator vector

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

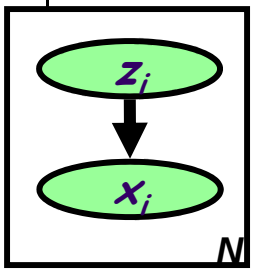
  - X is a conditional Gaussian variable with a class-specific mean/covariance

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

  - The likelihood of a sample:

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

EM algorithm for GMM



- **Recall MLE for completely observed data**
- **Data log-likelihood**

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

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

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

- **MLE**

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

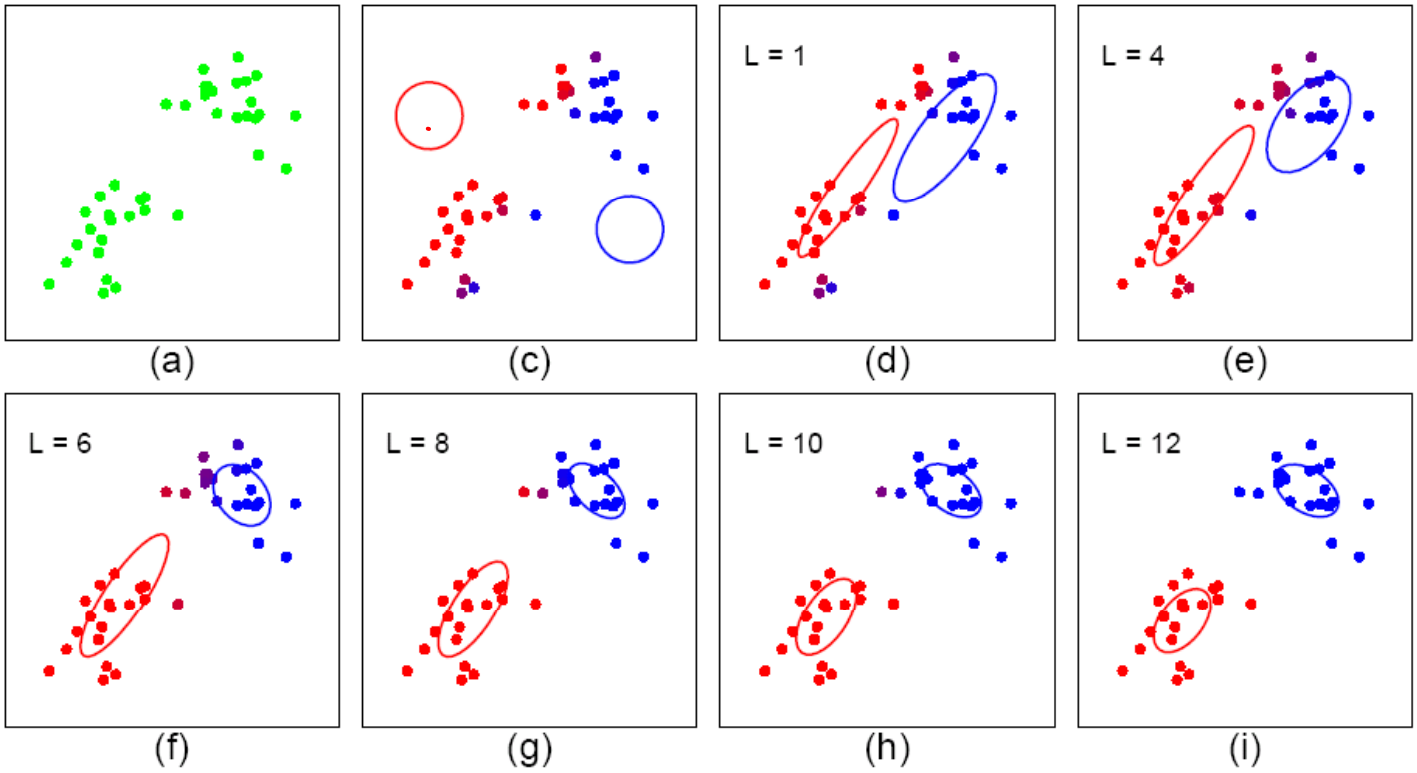
$$\hat{\mu}_{k,MLE} = \arg\max_\mu \ell(\boldsymbol{\theta}; D) \quad \Rightarrow \hat{\mu}_{k,MLE} = \frac{\sum_n z_n^k x_n}{\sum_n z_n^k}$$

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

- **What if we do not know z_n?** $\quad z_n \rightarrow p(z_n^k = 1 \mid x, \mu^{(t)}, \Sigma^{(t)})$

EM algorithm for GMM

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



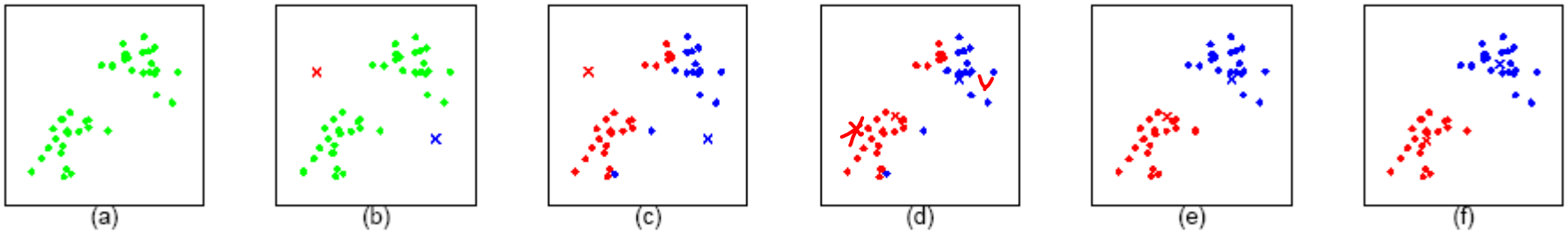
Comparing to K-means

- Start:
  - "Guess" the centroid μ_k and coveriance Σ_k of each of the K clusters
- Loop
  - For each point n=1 to N,
    compute its cluster label:

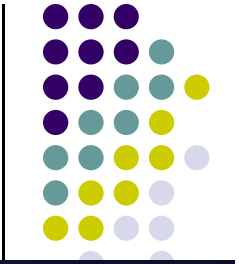
$$z_n^{(t)} = \arg\max_k (x_n - \mu_k^{(t)})^T \Sigma_k^{-1(t)} (x_n - \mu_k^{(t)})$$

  - For each cluster k=1:K

$$\mu_k^{(t+1)} = \frac{\sum_n \delta(z_n^{(t)}, k) x_n}{\sum_n \delta(z_n^{(t)}, k)} \qquad \Sigma_k^{(t+1)} = \ldots$$

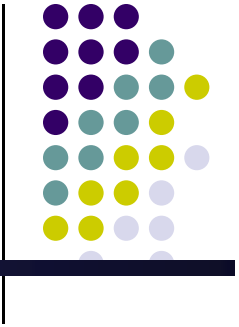


Notes on EM Algorithm

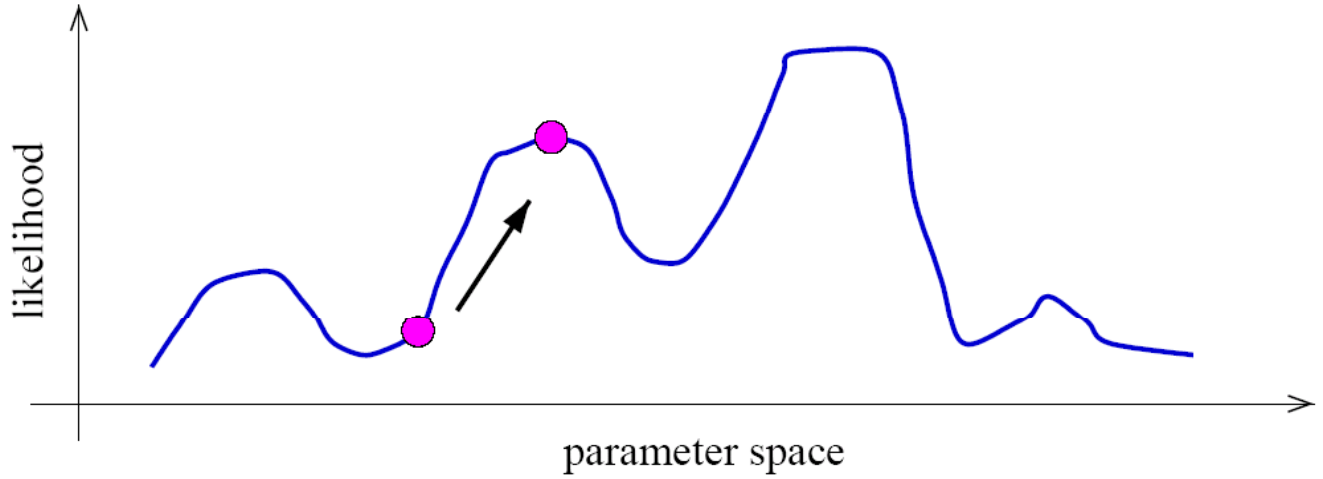


- EM is an optimization strategy for objective functions that can be interpreted as likelihoods in the presence of missing data.
- It is much simpler than gradient methods:
  - No need to choose step size.
  - Enforces constraints automatically.
  - Calls inference and fully observed learning as subroutines.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

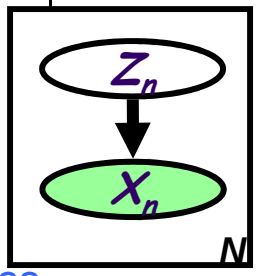
Identifiability



- 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.



How is EM derived?



- 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 a 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:

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

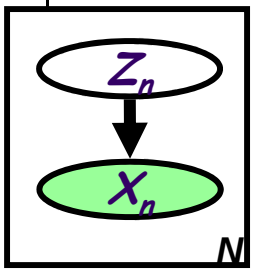
- The "complete" likelihood

$$p(x_n, z_n^k = 1|\mu,\Sigma) = p(z_n^k = 1|\pi)\, p(x,|\, z_n^k = 1, \mu, \Sigma) = \pi_k N(x,|\, \mu_k, \Sigma_k)$$

$$p(x_n, z_n|\mu,\Sigma) = \prod_k \left[\pi_k N(x,|\, \mu_k, \Sigma_k)\right]^{z_n^k}$$

But this is itself a random variable! Not good as objective function

How is EM derived?



- The complete log likelihood:

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

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

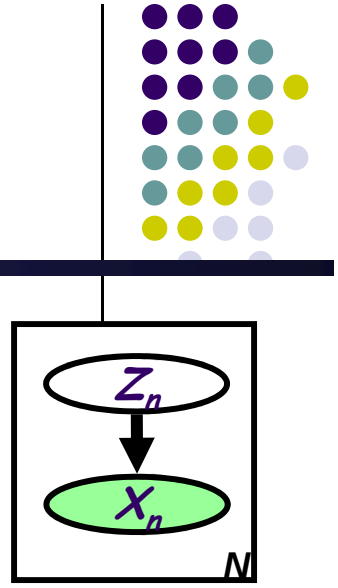
$$= \sum_n \sum_k z_n^k \log \pi_k - \sum_n \sum_k z_n^k \frac{1}{2\sigma^2} (x_n - \mu_k)^2 + C$$

- The expected complete log likelihood

$$\langle \ell_c(\theta; x, z) \rangle = \sum_n \langle \log p(z_n \mid \pi) \rangle_{p(z|x)} + \sum_n \langle \log p(x_n \mid z_n, \mu, \Sigma) \rangle_{p(z|x)}$$

$$= \sum_n \sum_k \langle z_n^k \rangle \log \pi_k - \frac{1}{2} \sum_n \sum_k \langle z_n^k \rangle \left((x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k) + \log |\Sigma_k| + C \right)$$

E-step



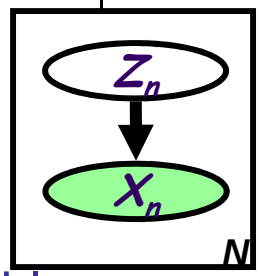
- We maximize $\langle l_c(\theta) \rangle$ iteratively using the following iterative procedure:

  – 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).

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

  - Here we are essentially doing **inference**

M-step



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

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

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

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

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

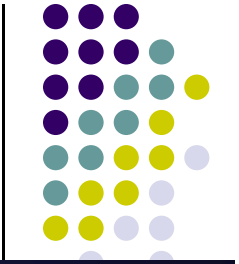
Fact :

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

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

- 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**")

Compare: K-means

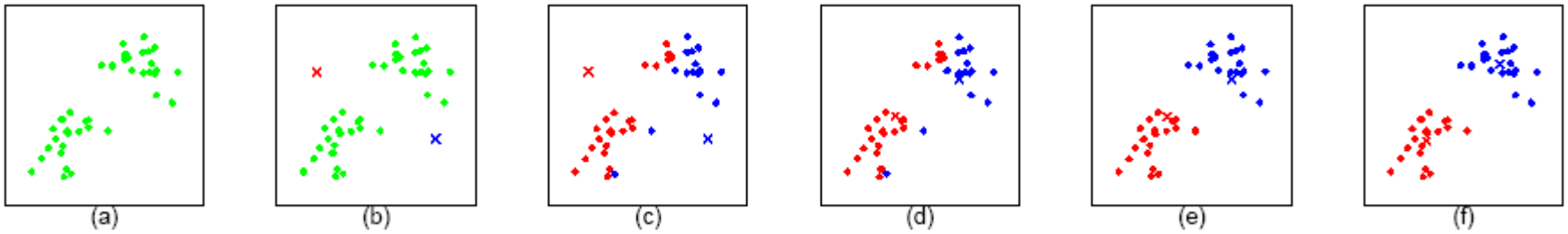


- The EM algorithm for mixtures of Gaussians is like a "soft version" of the K-means algorithm.
- In the K-means “E-step” we do hard assignment:

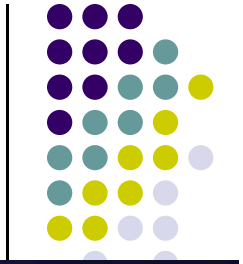
$$z_n^{(t)} = \arg\max_k (x_n - \mu_k^{(t)})^T \Sigma_k^{-1(t)} (x_n - \mu_k^{(t)})$$

- 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:

$$\mu_k^{(t+1)} = \frac{\sum_n \delta(z_n^{(t)}, k) x_n}{\sum_n \delta(z_n^{(t)}, k)}$$



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.

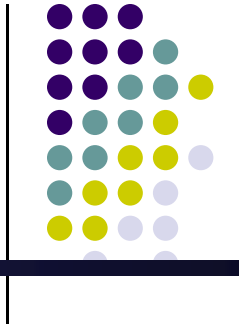
- But we do not observe z, so computing

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

  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**.

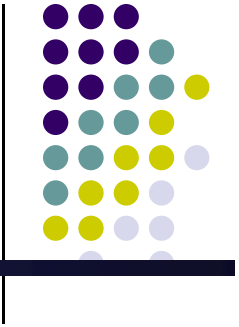
- Incomplete log likelihood

  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



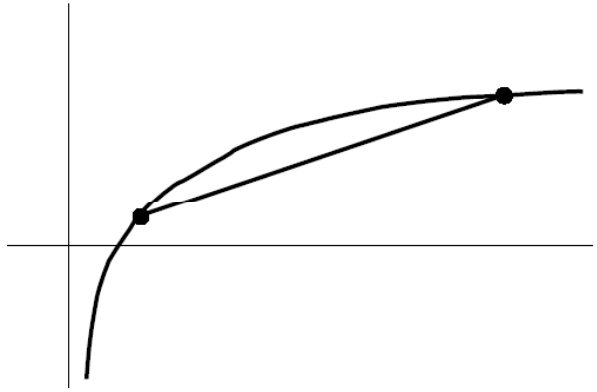
- For ***any*** distribution $q(z)$, define *expected complete log likelihood*:

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

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

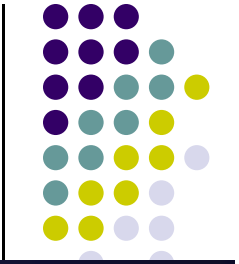
- Jensen's inequality

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



$$\Rightarrow \quad \ell(\theta; x) \geq \left\langle \ell_c(\theta; x, z) \right\rangle_q + H_q$$

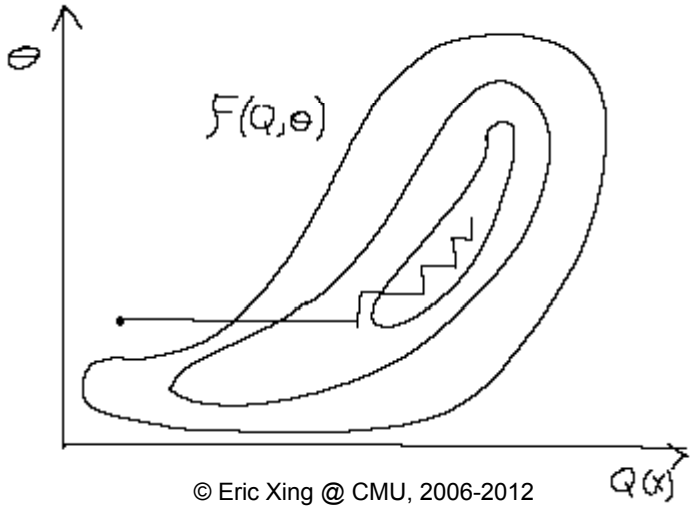
Lower Bounds and Free Energy



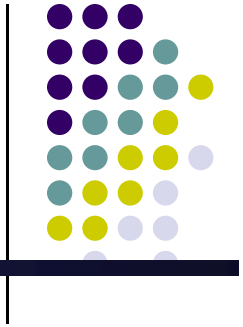
- For fixed data x, define a functional called the free energy:

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

- 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:

$$q^{t+1} = \arg\max_{q} F(q, \theta^t) = p(z \mid x, \theta^t)$$

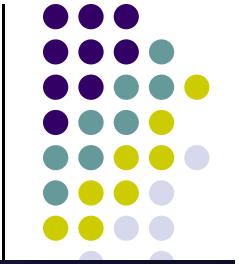
  - This is the posterior distribution over the latent variables given the data and the parameters. Often we need this at test time anyway (e.g. to perform classification).

- Proof (easy): this setting attains the bound $\ell(\theta; x) \geq F(q, \theta)$

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

- 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:

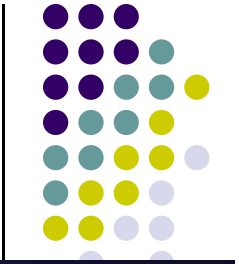
$$p(x,z|\theta) = \frac{1}{Z(\theta)} h(x,z) \exp\left\{\sum_i \theta_i f_i(x,z)\right\}$$

  - 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:

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

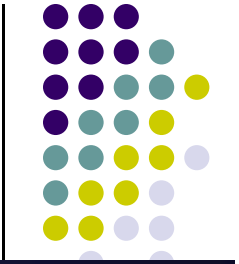
  - The first term is the expected complete log likelihood (energy) and the second term, which does not depend on θ, is the entropy.

- Thus, in the M-step, maximizing with respect to θ for fixed q we only need to consider the first term:

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

  - 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. $p(z|x,\theta)$.

Summary: EM Algorithm

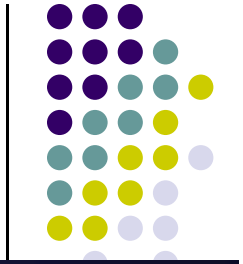


- A way of maximizing likelihood function for latent variable models. Finds MLE of parameters when the original (hard) problem can be broken up into two (easy) pieces:
  1. Estimate some "missing" or "unobserved" data from observed data and current parameters.
  2. Using this "complete" data, find the 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.

EM Variants



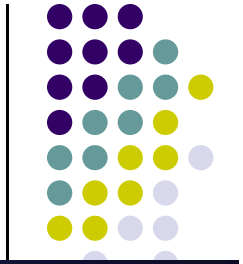
- Sparse EM:

  Do not re-compute 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). Recall the IRLS step in the mixture of experts model.

A Report Card for EM

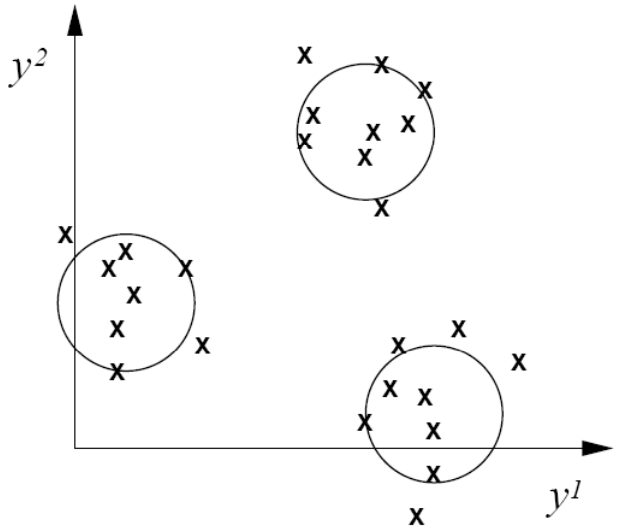


- Some good things about EM:
  - no learning rate (step-size) parameter
  - automatically enforces parameter constraints
  - very fast for low dimensions
  - each iteration guaranteed to improve likelihood

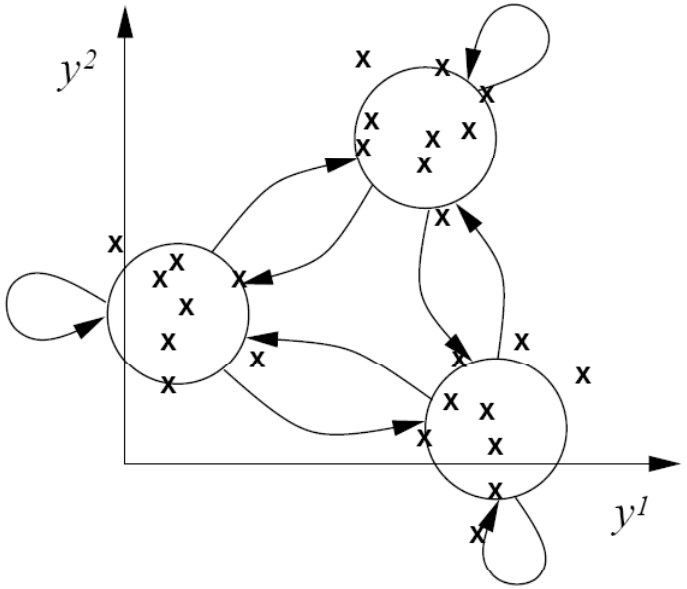
- Some bad things about EM:
  - can get stuck in local minima
  - can be slower than conjugate gradient (especially near convergence)
  - requires expensive inference step
  - is a maximum likelihood/MAP method

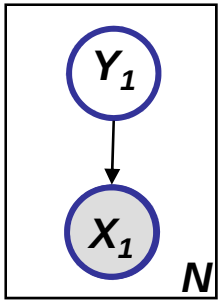
From static to dynamic mixture models

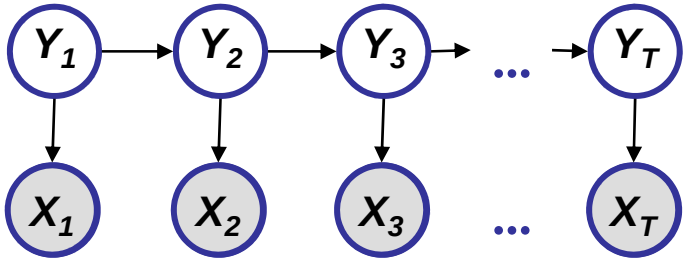
Static mixture



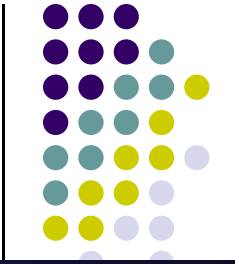
Dynamic mixture







Hidden Markov Models



The underlying source:

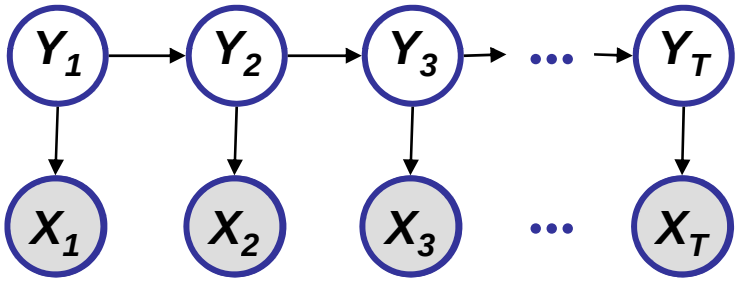
genomic entities,

dice,

The sequence:

CGH signal,

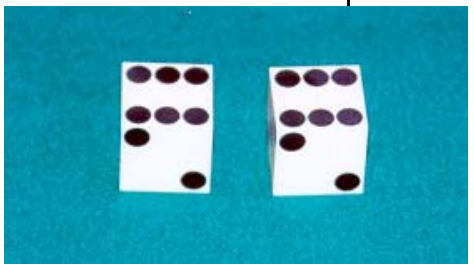
sequence of rolls,



Markov property:

This problem in IMPORTANT!!! - ☺

An experience in a casino



Game:

1. You bet $1
2. You roll (always with a fair die)
3. Casino player rolls (maybe with fair die, maybe with loaded die)
4. Highest number wins $2

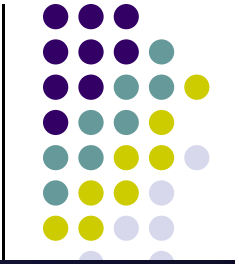
Question:

12455264621461461361366616646616366163661636616515615115146123562344

Which die is being used in each play?

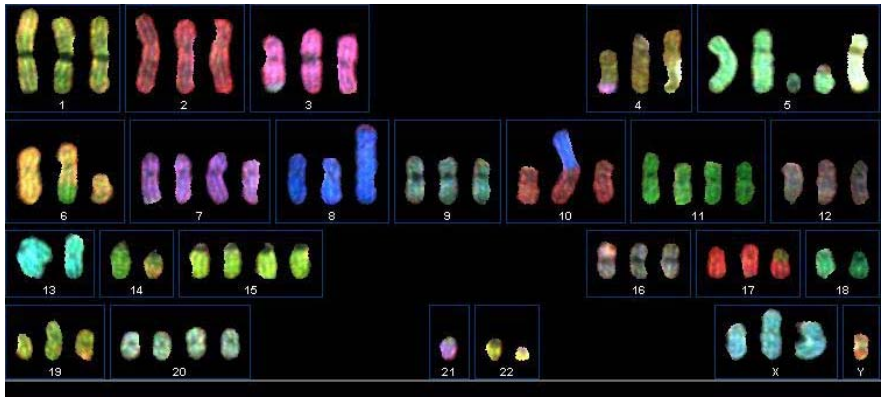
A more serious question ...

☹



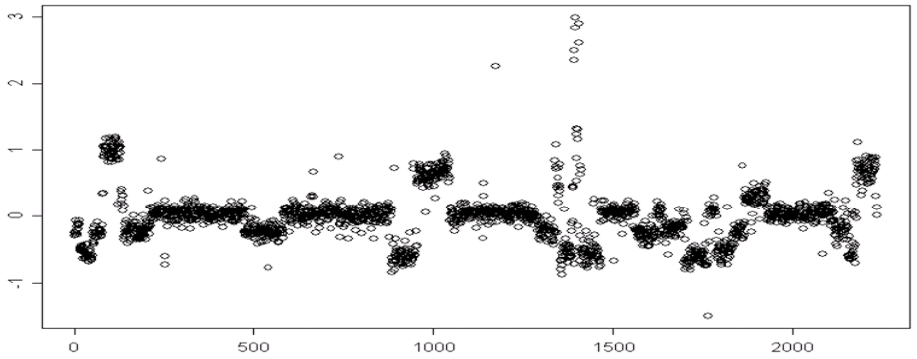
- Naturally, data points arrive one at a time
  - Does the ordering index carry (additional) clustering information besides the data value itself ?
  - Example:

    Chromosomes of tumor cell:

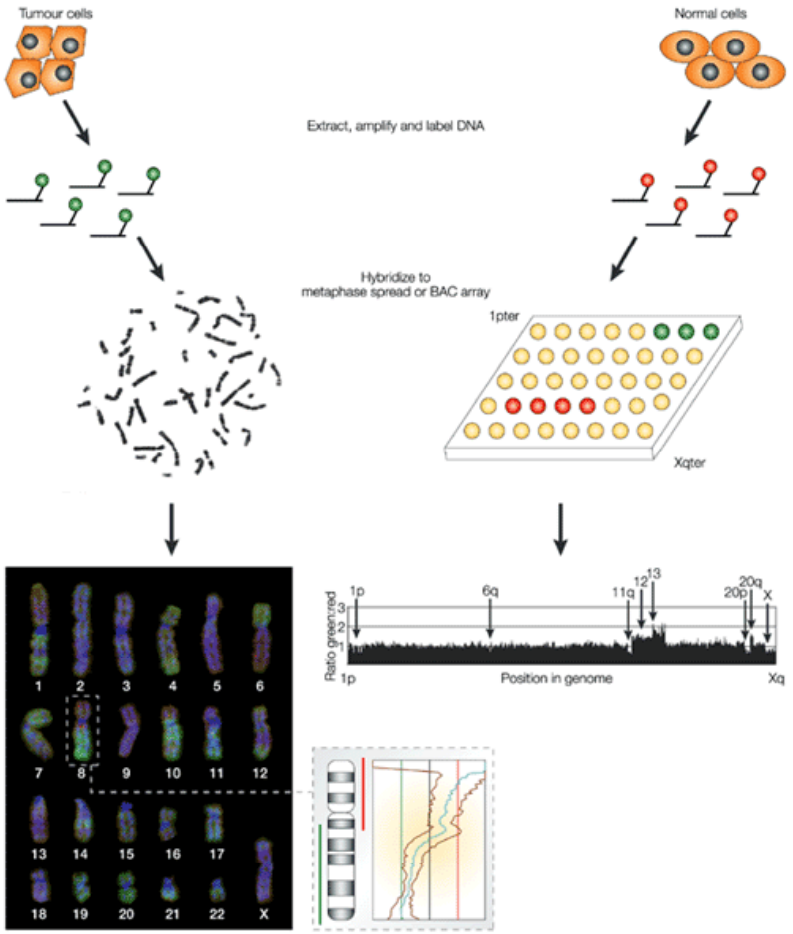


Copy number measurements

(known as CGH)



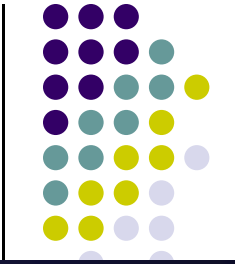
Array CGH (comparative genomic hybridization)



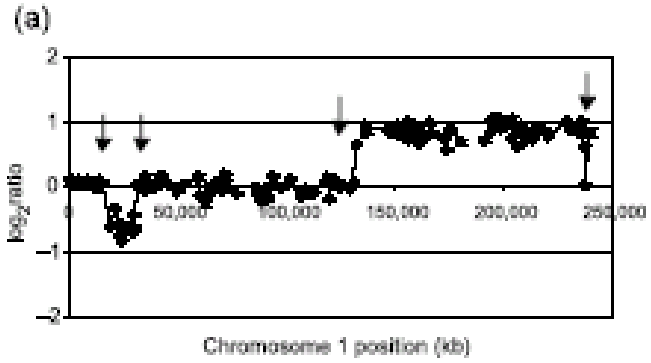
Nature Reviews | Genetics

- The basic assumption of a CGH experiment is that the ratio of the binding of test and control DNA is proportional to the ratio of the copy numbers of sequences in the two samples.
- But various kinds of noises make the true observations less easy to interpret …

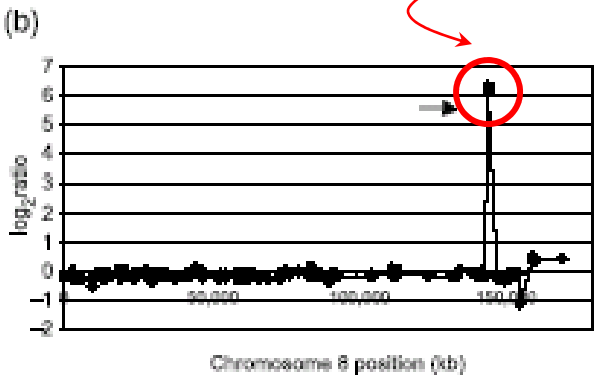
DNA Copy number aberration types in breast cancer



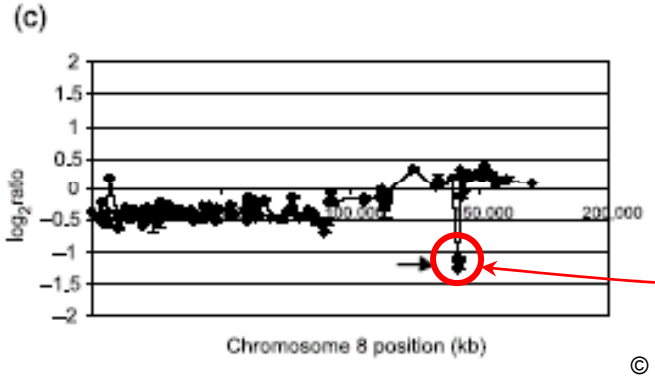
60-70 fold amplification of CMYC region



Copy number profile for chromosome 1 from 600 MPE cell line



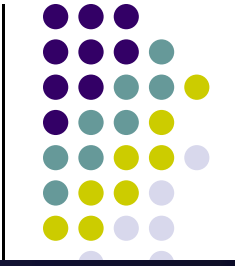
Copy number profile for chromosome 8 from COLO320 cell line



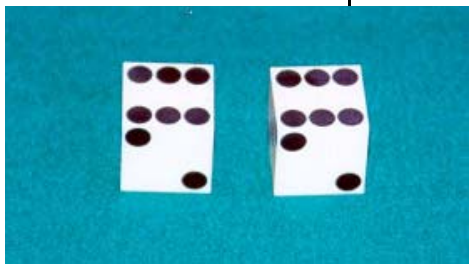
Copy number profile for chromosome 8 in MDA-MB-231 cell line

deletion

Suppose you were told about the following story before heading to Vegas…



The Dishonest Casino !!!



A casino has two dice:

- **Fair die**

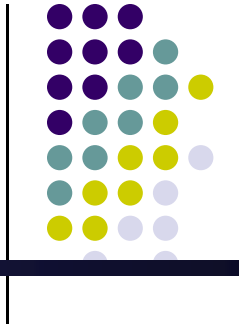
  P(1) = P(2) = P(3) = P(5) = P(6) = 1/6
- **Loaded die**

  P(1) = P(2) = P(3) = P(5) = 1/10

  P(6) = 1/2

Casino player switches back-&-forth between fair and loaded die once every 20 turns

Puzzles Regarding the Dishonest Casino



GIVEN: A sequence of rolls by the casino player

12455264621461461361366616646616366163661636165156151151461235623 44

QUESTION

- How likely is this sequence, given our model of how the casino works?
  - This is the **EVALUATION** problem
- What portion of the sequence was generated with the fair die, and what portion with the loaded die?
  - This is the **DECODING** question
- How "loaded" is the loaded die? How "fair" is the fair die? How often does the casino player change from fair to loaded, and back?
  - This is the **LEARNING** question

Definition (of HMM)

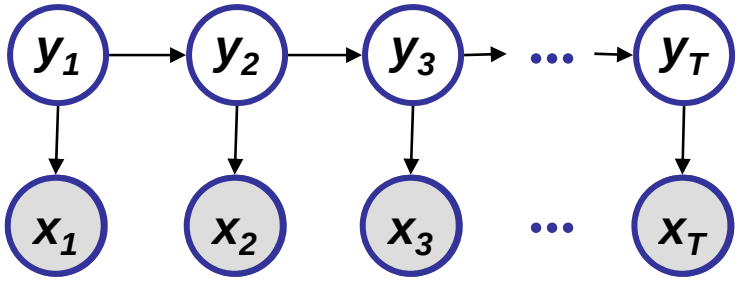
- Observation space

  Alphabetic set: $\mathbb{C} = \{c_1, c_2, \cdots, c_K\}$

  Euclidean space: $\mathbb{R}^d$

- Index set of hidden states

  $\mathbb{I} = \{1, 2, \cdots, M\}$



Graphical model

- 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

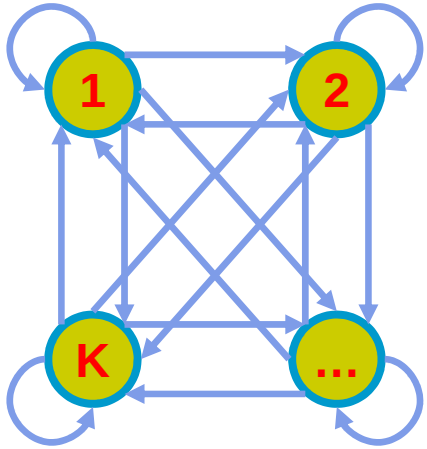
  $p(y_1) \sim \text{Multinomial}(\pi_1, \pi_2, \ldots, \pi_M)$.

- Emission probabilities associated with each state

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

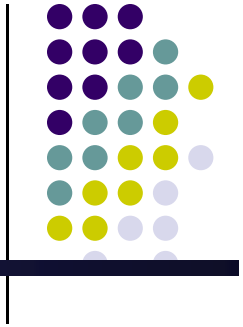
  or in general:

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



State automata

Three Main Questions on HMMs



1. Evaluation

   GIVEN an HMM $\mathcal{M}$, and a sequence x,
   FIND Prob (x | $\mathcal{M}$)
   ALGO. Forward

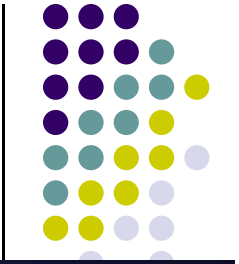
2. Decoding

   GIVEN an HMM $\mathcal{M}$, and a sequence x,
   FIND the sequence y of states that maximizes, e.g., P(y | x, $\mathcal{M}$),
   or the most probable subsequence of states
   ALGO. Viterbi, Forward-backward

3. Learning

   GIVEN an HMM M, with unspecified transition/emission probs.,
   and a sequence x,
   FIND parameters $\theta = (\pi_i, a_{ij}, \eta_{ik})$ that maximize $P(x \mid \theta)$
   ALGO. Baum-Welch (EM)

Learning 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

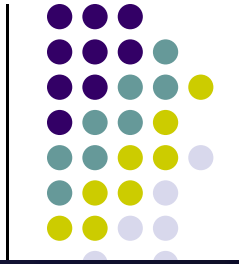
- **Unsupervised learning**: estimation when the "right answer" is unknown
  - **Examples:**

    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

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

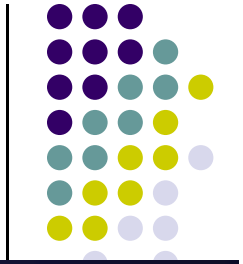
    x = 2, 1, 5, 6, 1, 2, 3, 6, 2, 3

    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



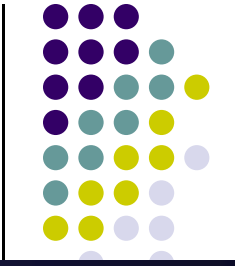
- Solution for small training sets:
  - Add pseudocounts

    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

Unsupervised ML estimation



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

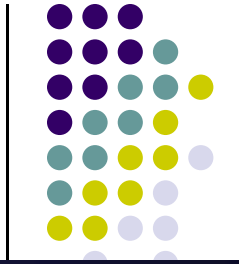
  - **EXPECTATION MAXIMIZATION**

  0. Starting with our best guess of a model M, parameters θ:
  1. Estimate A_{ij} , B_{ik} in the training data
     - How? $A_{ij} = \sum_{n,t} \left\langle y^i_{n,t-1} y^j_{n,t} \right\rangle \quad B_{ik} = \sum_{n,t} \left\langle y^i_{n,t} \right\rangle x^k_{n,t}$,
     - Update θ according to A_{ij} , B_{ik}
     - Now a "supervised learning" problem
  2. 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

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

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

- EM
  - The **E** step

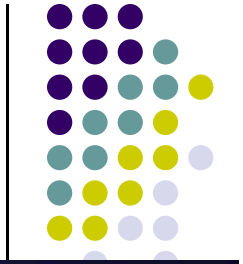
$$\gamma_{n,t}^i = \left\langle y_{n,t}^i \right\rangle = p(y_{n,t}^i = 1 \mid \mathbf{x}_n)$$

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

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

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

The Baum-Welch algorithm -- comments



Time Complexity:

iterations $\times$ $O(K^2N)$

- Guaranteed to increase the log likelihood of the model
- Not guaranteed to find globally best parameters
- Converges to local optimum, depending on initial conditions
- Too many parameters / too large model: Overt-fitting