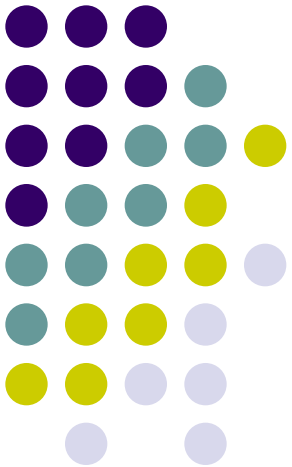
Machine Learning

Mixture Model, HMM, and Expectation Maximization

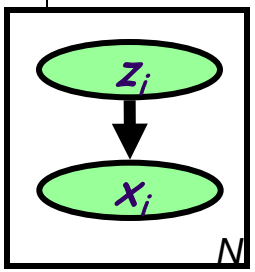


Eric Xing

Lecture 9, August 14, 2010

Reading:

Gaussian Discriminative Analysis



- 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

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

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

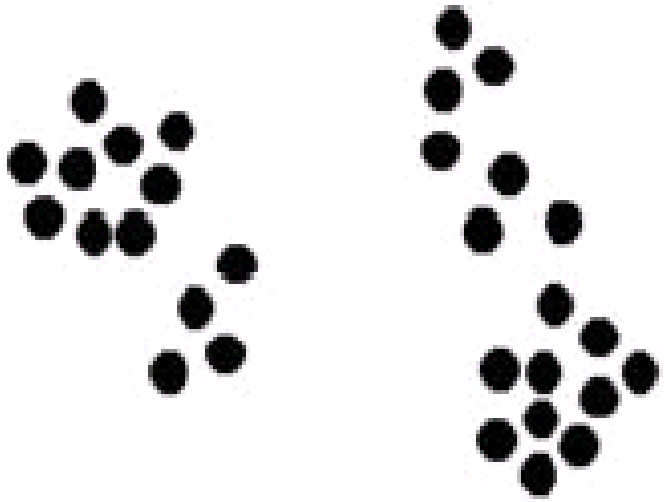
$$\hat{\sigma}_{k,MLE} = \arg\max_{\sigma} \ell(\boldsymbol{\theta}; D)$$

$$\hat{\mu}_{k,MLE} = \frac{\sum_n z_n^k x_n}{\sum_n z_n^k}$$

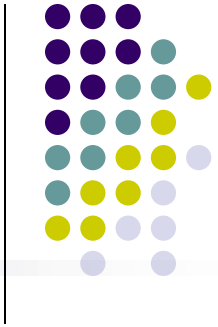
$$p(y_n^k = 1 \mid \mathbf{x}_n, \mu, \sigma) = \frac{\pi_k \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{x}_n - \mu_k)^2\right\}}{\sum_{k'} \pi_{k'} \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{x}_n - \mu_{k'})^2\right\}}$$

- What if we do not know z_n?

Clustering



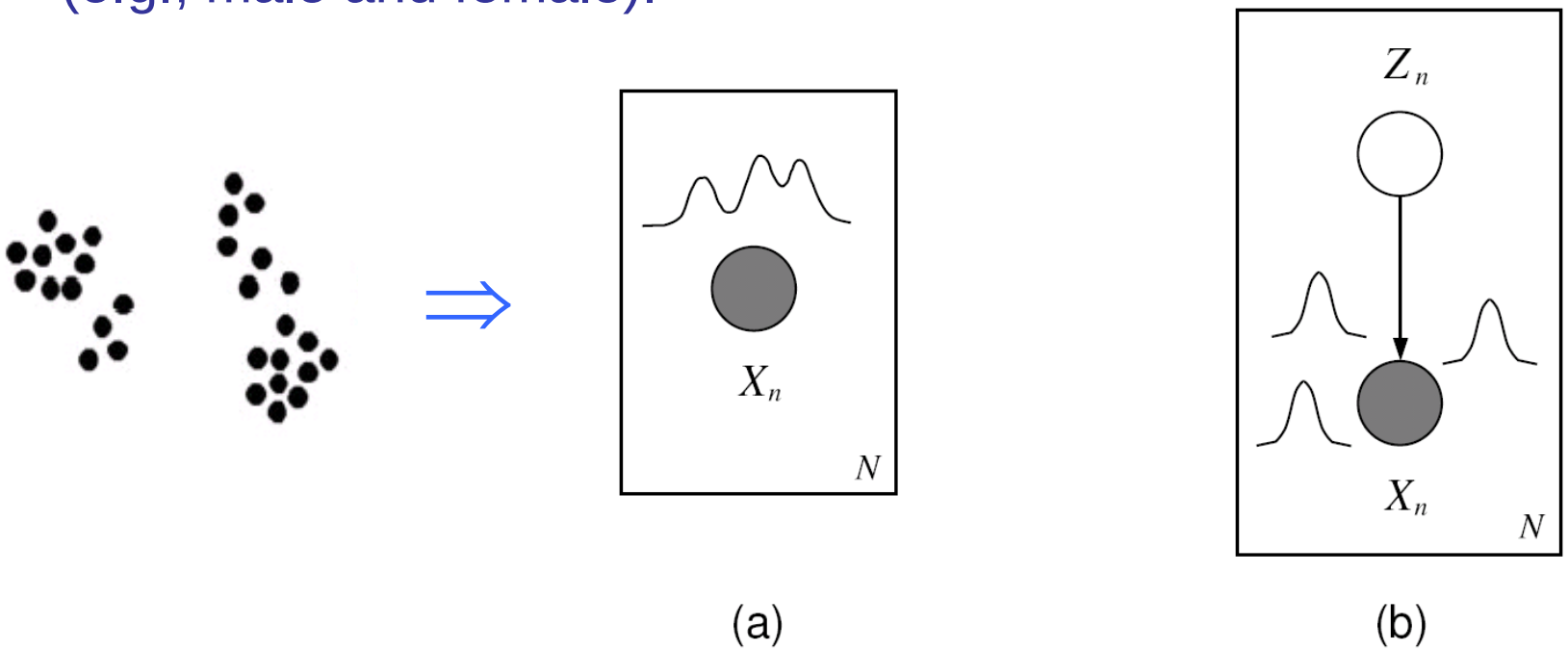
Unobserved Variables



- 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

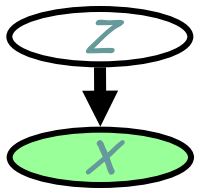
- A density model $p(x)$ may be multi-modal.
- We may be able to model it as a mixture of uni-modal distributions (e.g., Gaussians).
- Each mode may correspond to a different sub-population (e.g., male and female).



Gaussian Mixture Models (GMMs)

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

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



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

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

  - The likelihood of a sample:

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

mixture component

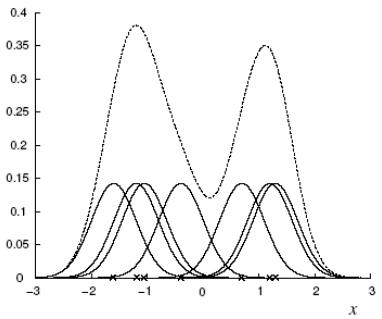
mixture proportion

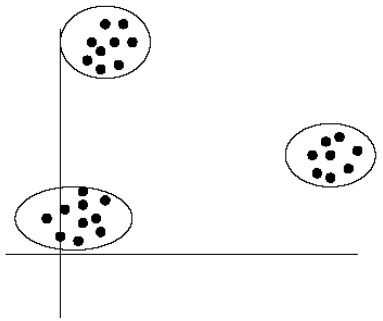
Gaussian Mixture Models (GMMs)

- Consider a mixture of *K* Gaussian components:

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

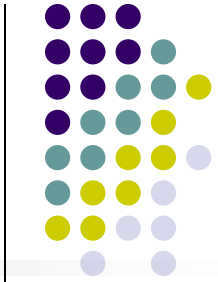
mixture proportion mixture component





- 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.

Learning mixture models



- Given data

$$\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^{N}$$

- Likelihood:

$$L(\pi, \mu, \Sigma; D) = \prod_n p(x_n | \pi, \mu, \Sigma) = \prod_n \left(\sum_k \pi_k N(x, | \mu_k, \Sigma_k) \right)$$

$$\{\pi^*, \mu^*, \Sigma^*\} = \arg\max L(\pi, \mu, \Sigma, D)$$

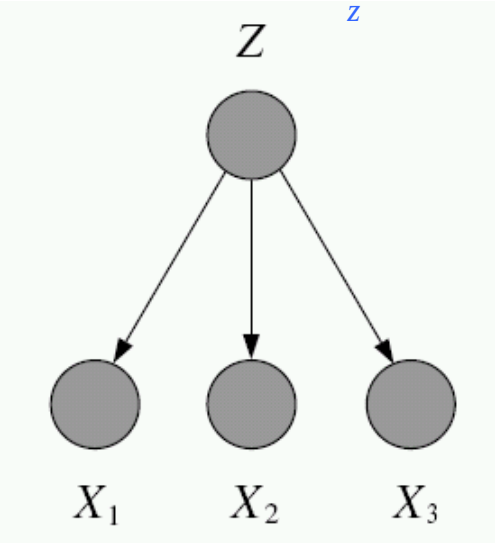
Why is Learning Harder?

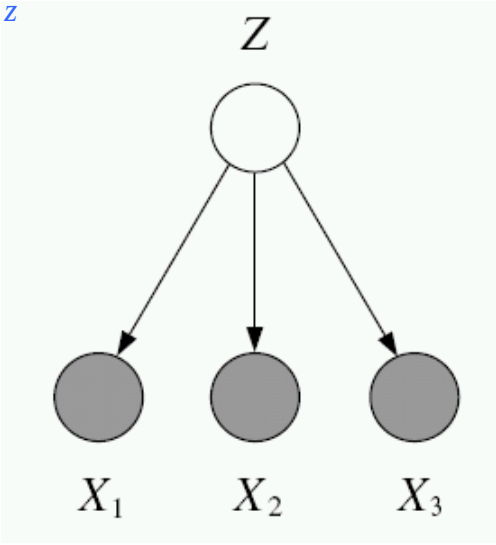
- In fully observed iid settings, the log likelihood decomposes into a sum of local terms.

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

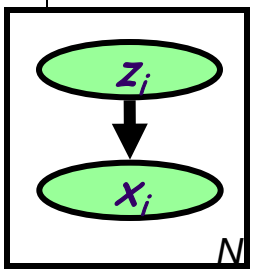
- With latent variables, all the parameters become coupled together via *marginalization*

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





Toward the EM algorithm



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

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

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

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

- MLE

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

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

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

$$\square \quad \hat{\mu}_{k,MLE} = \frac{\sum_n z_n^k x_n}{\sum_n z_n^k}$$

- What if we do not know z_n?

Recall K-means

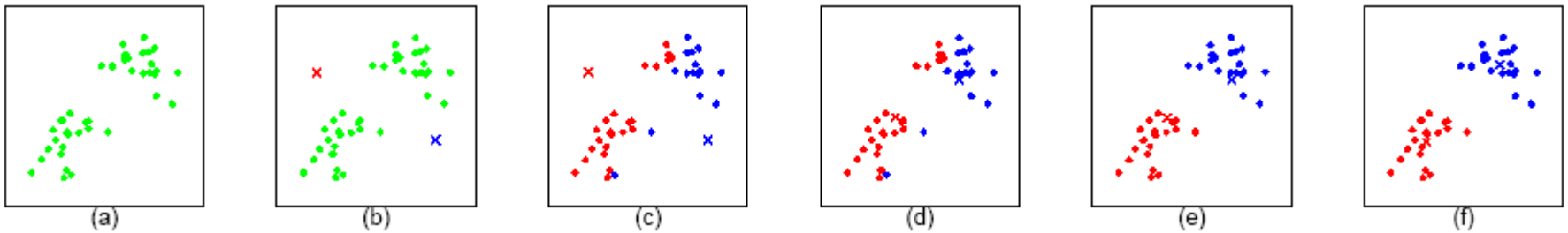
- Start:
  - "Guess" the centroid μ_k and coveriance Σ_k of each of the K clusters
- Loop
  - For each point n=1 to N,

    compute its cluster label:

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

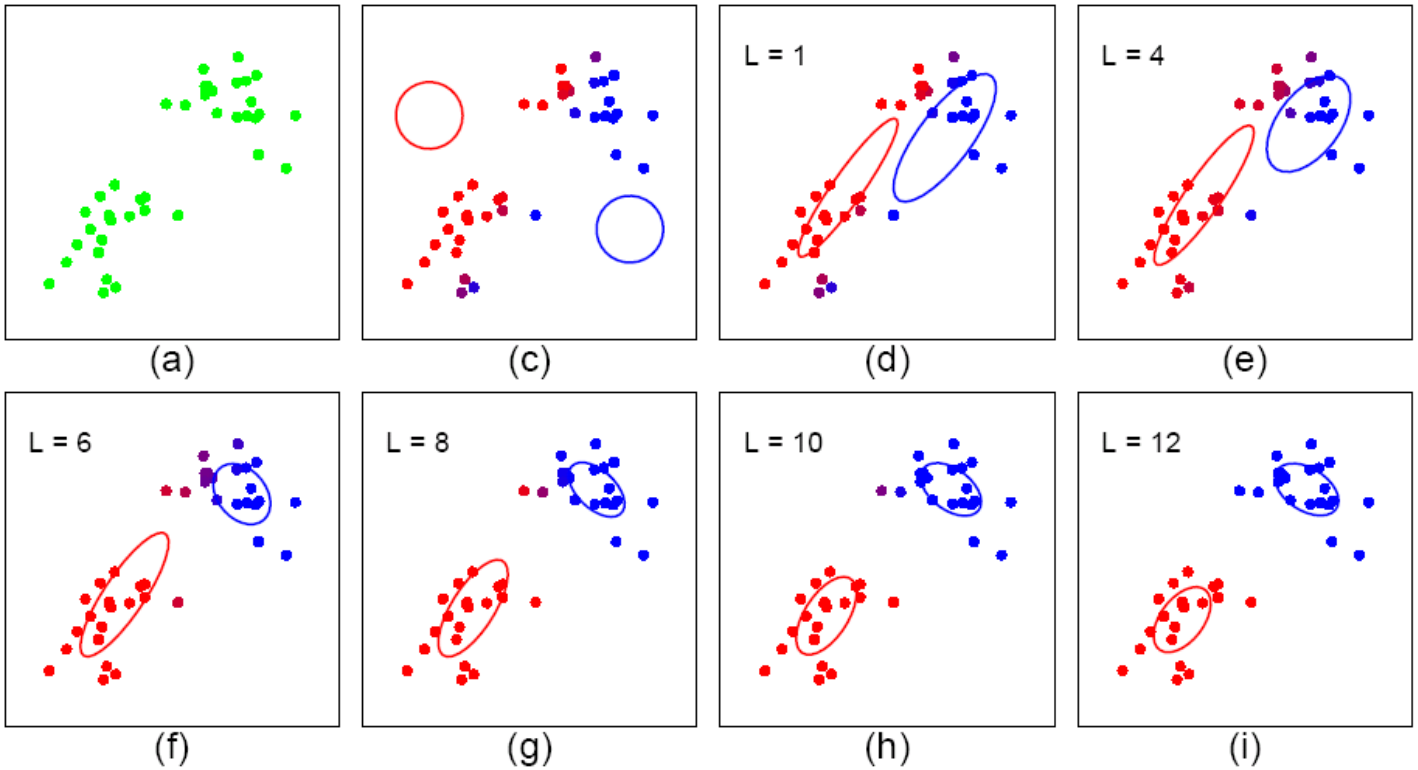
  - For each cluster k=1:K

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

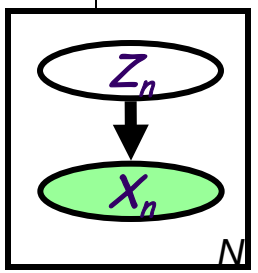


Expectation-Maximization

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



E-step

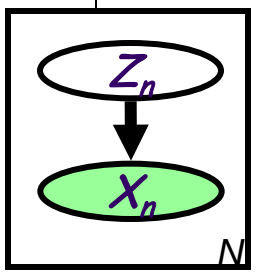


- Expectation step: computing the expected value of the sufficient statistics of the hidden variables (i.e., z) given current est. of the parameters (i.e., π and μ).

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

  - Here we are essentially doing **inference**

M-step



- Maximization step: compute the parameters under current results of the expected value of the hidden variables

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

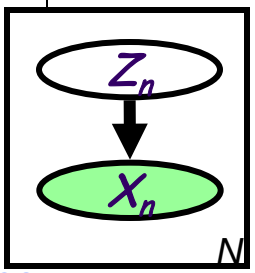
$$\square \ \pi_k^* = \sum_n \langle z_n^k \rangle_{q^{(t)}} / N = \sum_n \tau_n^{k(t)} / N = \langle n_k \rangle / N$$

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

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

- 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**")

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

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

  - The likelihood of a sample:

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

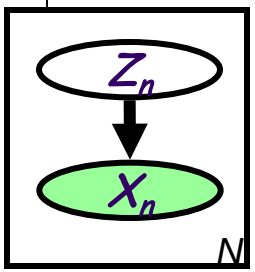
- The "complete" likelihood

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

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

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

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

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

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

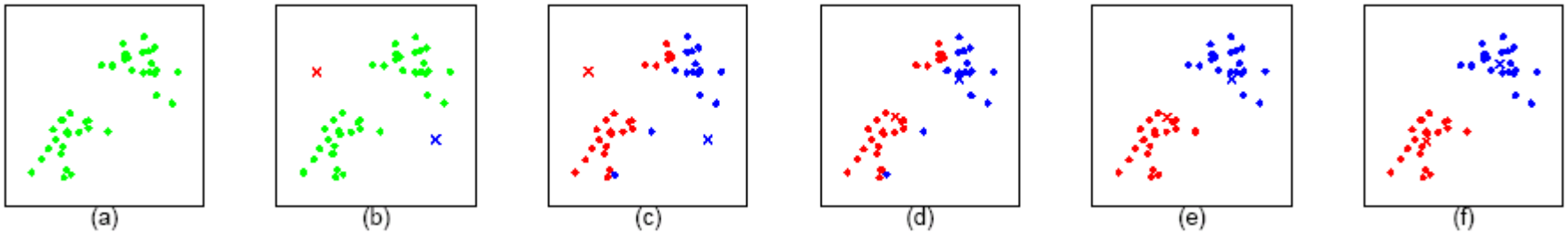
Compare: K-means

- 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:

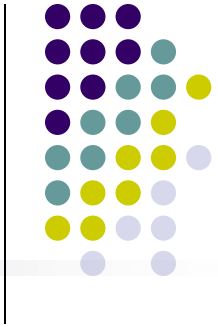
$$z_n^{(t)} = \arg\max_k (x_n - \mu_k^{(t)})^T \Sigma_k^{-1(t)} (x_n - \mu_k^{(t)}) \qquad \left(\tau_n^{k(t)} = \left\langle z_n^k \right\rangle_{q^{(t)}} \right)$$

- 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:

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



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.

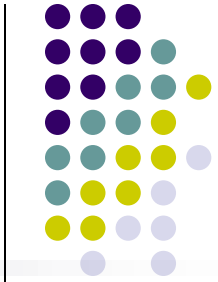
- But we do not observe z, so computing

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

  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**.

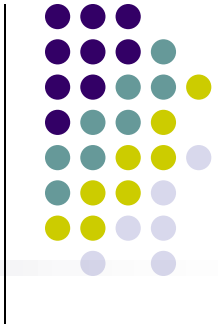
- Incomplete log likelihood

  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



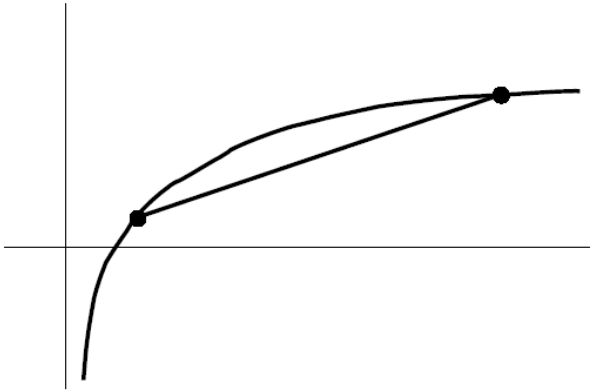
- For ***any*** distribution $q(z)$, define *expected complete log likelihood*:

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

  - 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

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



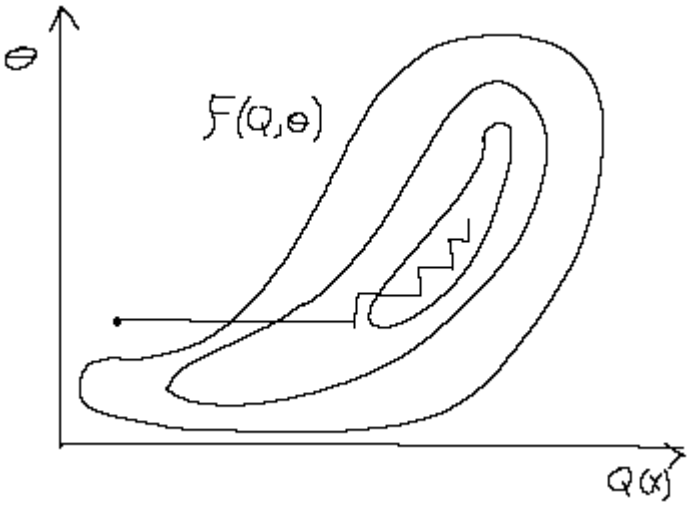
$$\Rightarrow \quad \ell(\theta; x) \geq \left\langle \ell_c(\theta; x, z) \right\rangle_q + H_q$$

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:

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

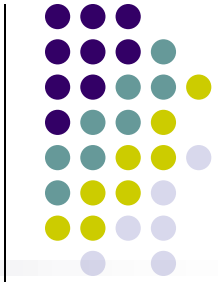
  - This is the posterior distribution over the latent variables given the data and the parameters. Often we need this at test time anyway (e.g. to perform classification).

- 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

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

$$= \sum_i \theta_i^t \left\langle f_i(x,z)\right\rangle_{q(z|x,\theta^t)} - A(\theta)$$

$$\overset{p\sim\text{GLIM}}{=} \sum_i \theta_i^t \left\langle \eta_i(z)\right\rangle_{q(z|x,\theta^t)} \xi_i(x) - A(\theta)$$

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

- Note that the free energy breaks into two terms:

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

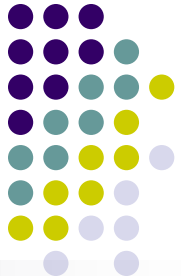
  - The first term is the expected complete log likelihood (energy) and the second term, which does not depend on θ, is the entropy.

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

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

  - 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



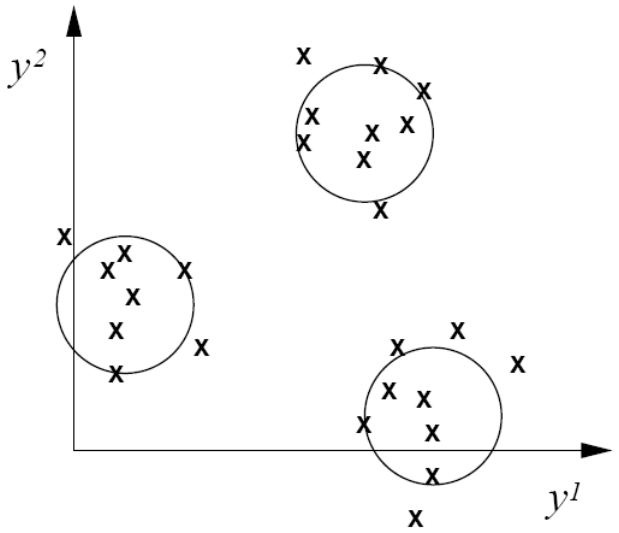
- A way of maximizing likelihood function for latent variable models. Finds MLE of parameters when the original (hard) problem can be broken up into two (easy) pieces:
  1. Estimate some "missing" or "unobserved" data from observed data and current parameters.
  2. Using this "complete" data, find the 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: $\theta^{t+1} = \arg\max_{\theta} F(q^{t+1}, \theta^t)$

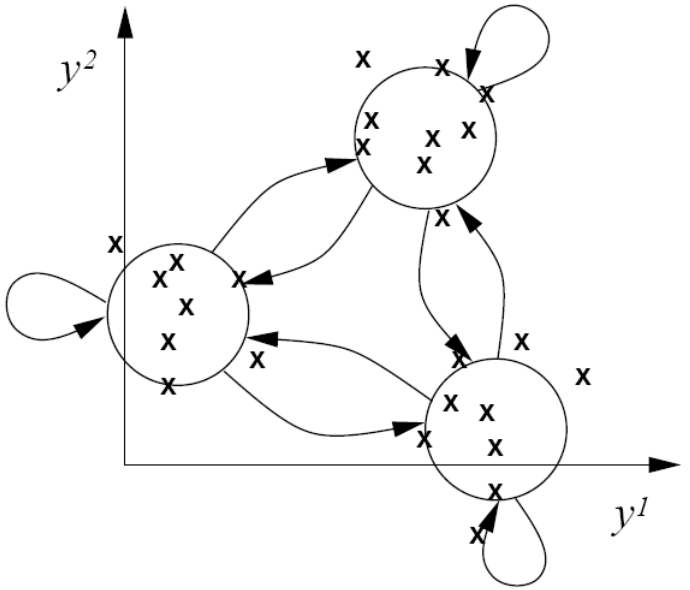
- In the M-step we optimize a lower bound on the likelihood. In the E-step we close the gap, making bound=likelihood.

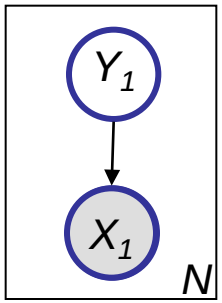
From static to dynamic mixture models

Static mixture



Dynamic mixture





The underlying source:

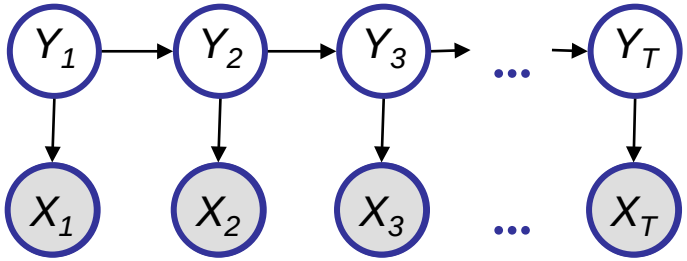
Speech signal,

dice,

The sequence:

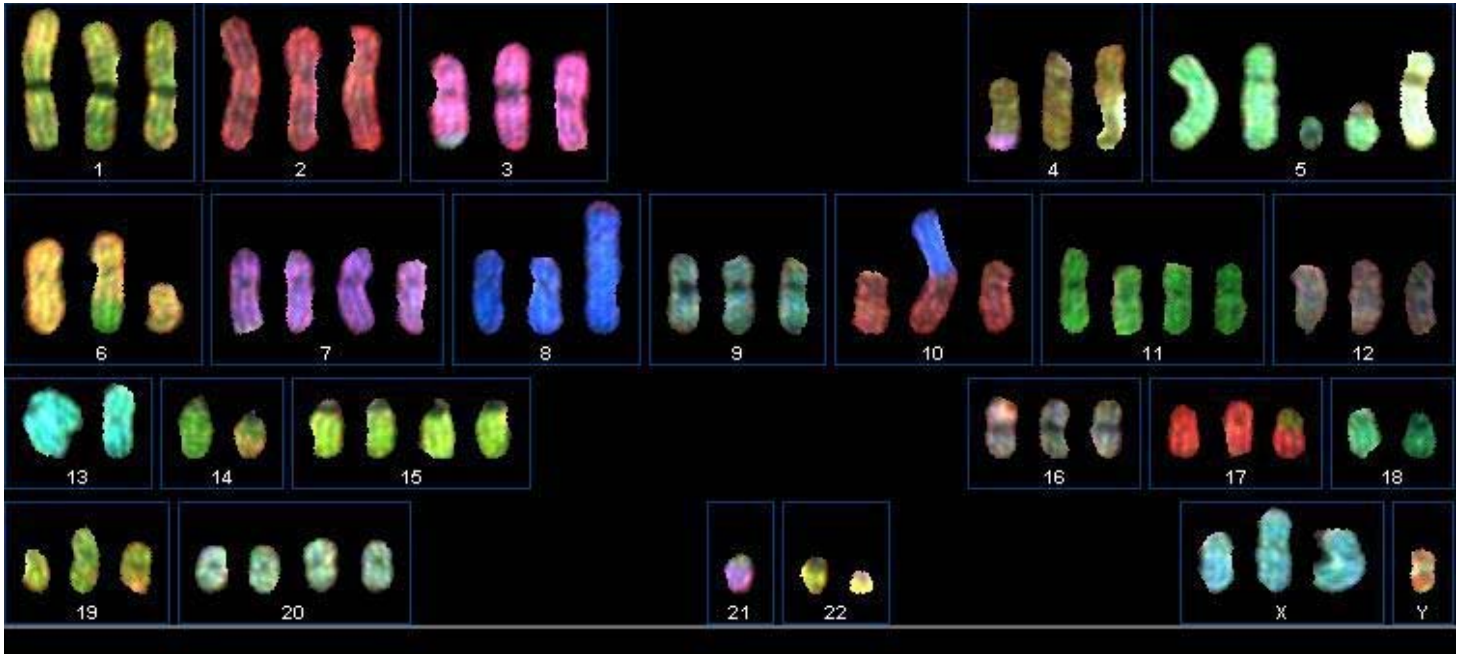
Phonemes,

sequence of rolls,



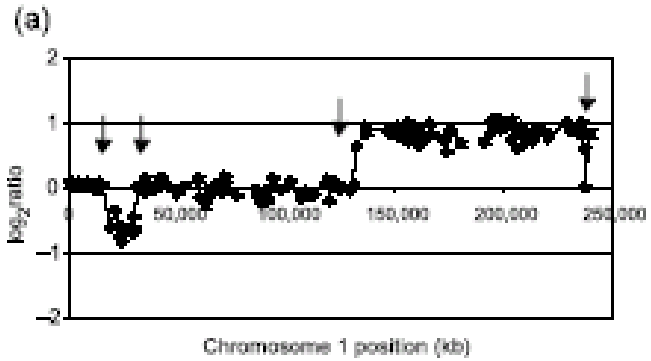
Predicting Tumor Cell States

Chromosomes of tumor cell:

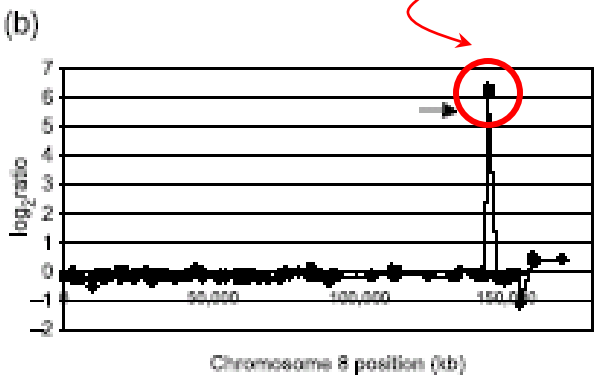


DNA Copy number aberration types in breast cancer

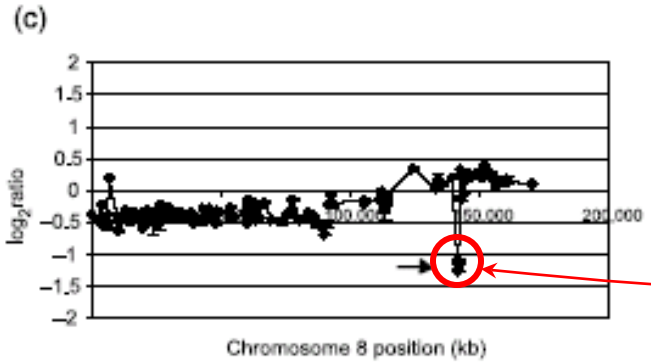
60-70 fold amplification of CMYC region



Copy number profile for chromosome 1 from 600 MPE cell line



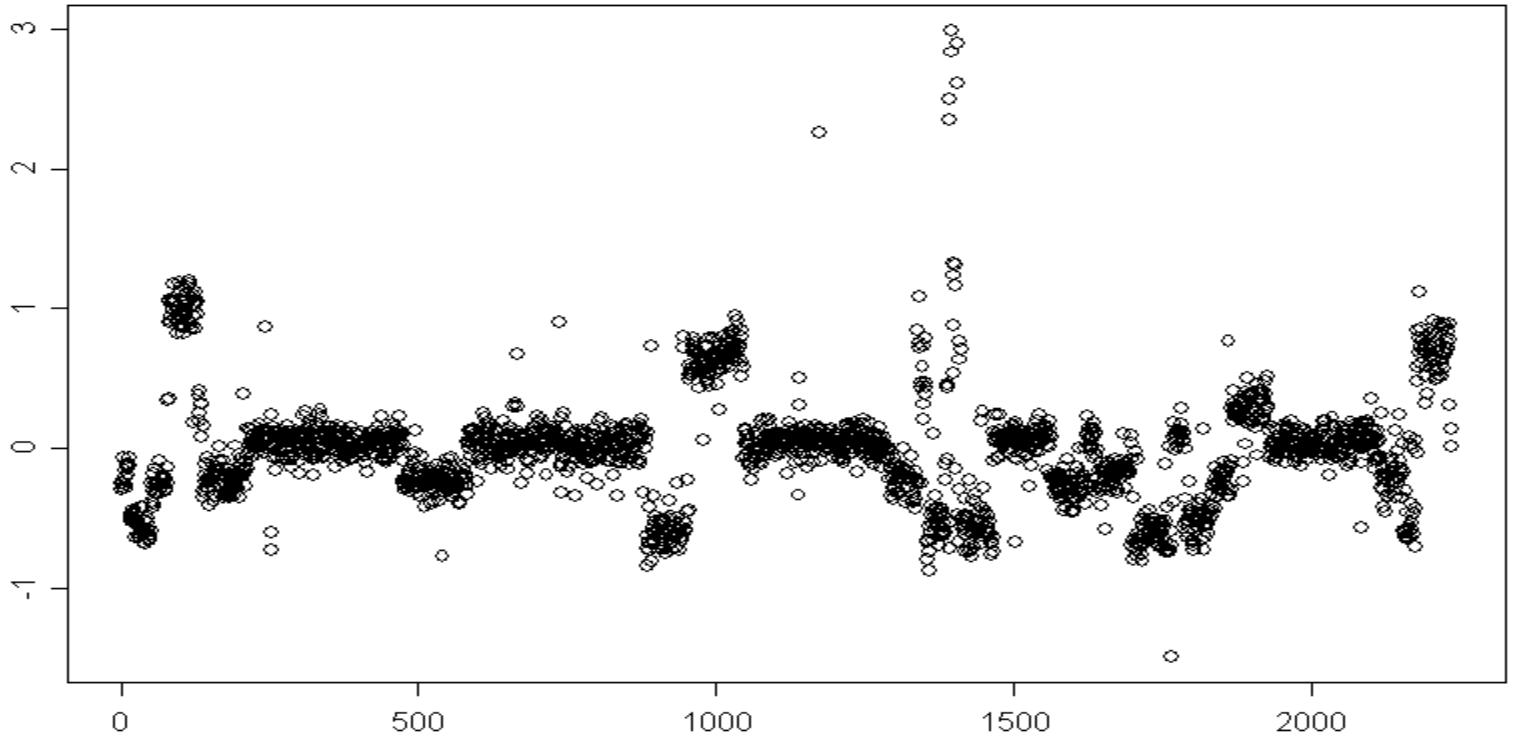
Copy number profile for chromosome 8 from COLO320 cell line



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

deletion

A real CGH run



Hidden Markov Model

- Observation space

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

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

- Index set of hidden states

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

- Transition probabilities between any two states

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

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

- Start probabilities

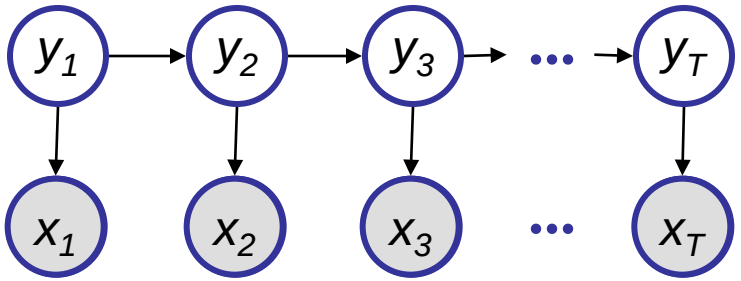
  $p(y_1) \sim \text{Multinomia}\ l(\pi_1, \pi_2, \ldots, \pi_M).$

- Emission probabilities associated with each state

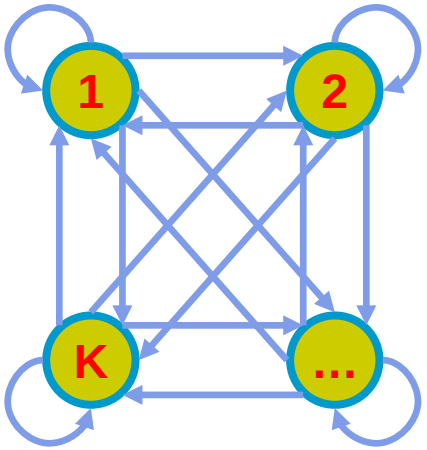
  $p(x_t \mid y_t^i = 1) \sim \text{Multinomia}\ l(b_{i,1}, b_{i,2}, \ldots, b_{i,K}), \forall i \in \mathbb{I}.$

  or in general:

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

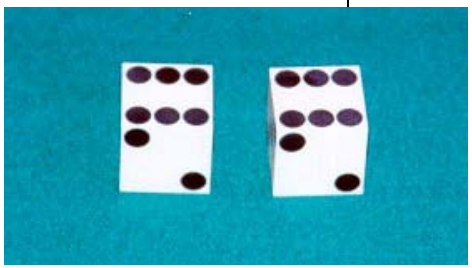


Graphical model



State automata

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

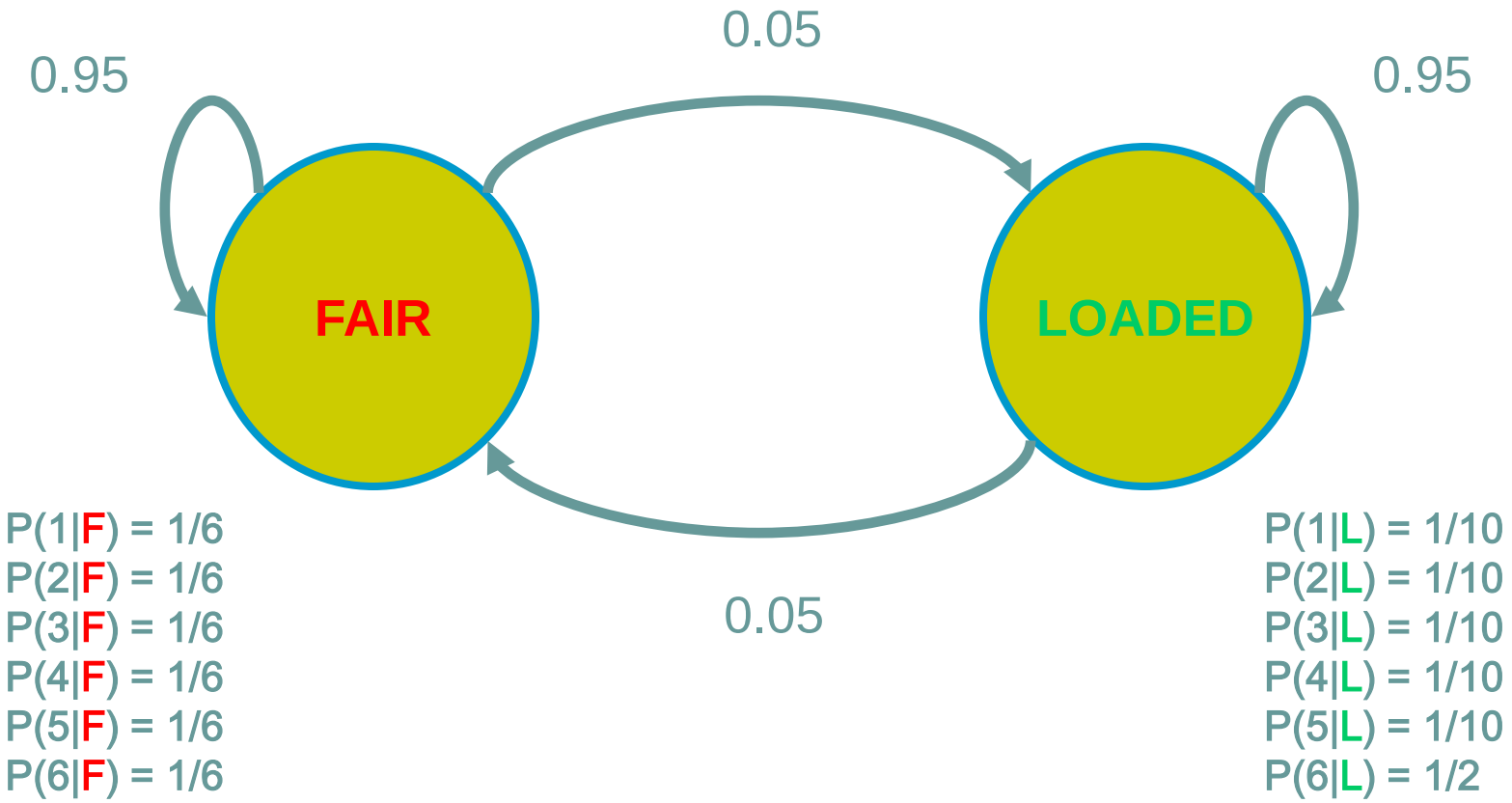
  P(6) = 1/2

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

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

The Dishonest Casino Model



Puzzles Regarding the Dishonest Casino

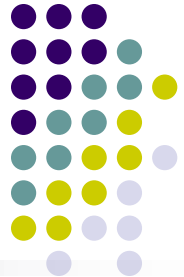
GIVEN: A sequence of rolls by the casino player

1245526462146146136136661664661636616366163616515615115146123562344

QUESTION

- How likely is this sequence, given our model of how the casino works?
  - This is the **EVALUATION** problem in HMMs
- What portion of the sequence was generated with the fair die, and what portion with the loaded die?
  - This is the **DECODING** question in HMMs
- 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 in HMMs

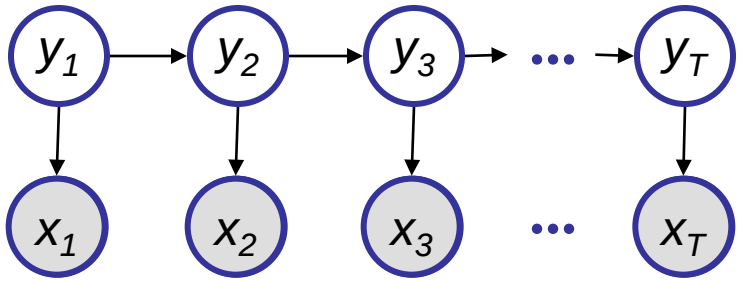
Joint Probability



1245526462146146136136661664661636616366163616515615115146123562344

Probability of a Parse

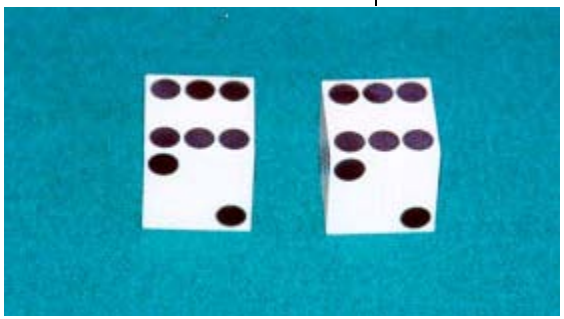
- Given a sequence $\mathbf{x} = x_1 \ldots\ldots x_T$ and a parse $\mathbf{y} = y_1, \ldots\ldots, y_T$,
- To find how likely is the parse: (given our HMM and the sequence)



$$p(\mathbf{x}, \mathbf{y}) = p(x_1 \ldots\ldots x_T, y_1, \ldots\ldots, y_T) \quad \text{(Joint probability)}$$
$$= p(y_1)\, p(x_1 \mid y_1)\, p(y_2 \mid y_1)\, p(x_2 \mid y_2) \ldots p(y_T \mid y_{T-1})\, p(x_T \mid y_T)$$
$$= p(y_1)\, \mathrm{P}(y_2 \mid y_1) \ldots p(y_T \mid y_{T-1}) \times p(x_1 \mid y_1)\, p(x_2 \mid y_2) \ldots p(x_T \mid y_T)$$

- Marginal probability: $p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}) = \sum_{y_1} \sum_{y_2} \cdots \sum_{y_N} \pi_{y_1} \prod_{t=2}^{T} a_{y_{t-1}, y_t} \prod_{t=1}^{T} p(x_t \mid y_t)$
- Posterior probability: $p(\mathbf{y} \mid \mathbf{x}) = p(\mathbf{x}, \mathbf{y}) / p(\mathbf{x})$

Example: the Dishonest Casino



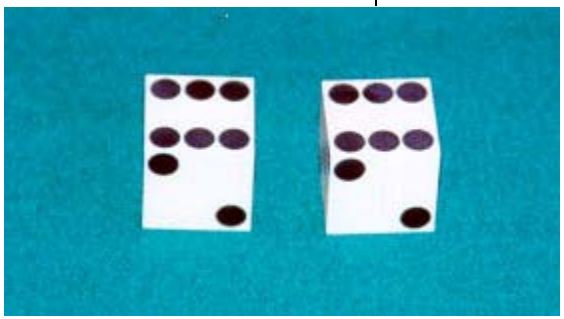
- Let the sequence of rolls be:
  - x = 1, 2, 1, 5, 6, 2, 1, 6, 2, 4
- Then, what is the likelihood of
  - y = Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair?
    (say initial probs $a_{0Fair} = ½$, $a_{oLoaded} = ½$)

½ × P(1 | Fair) P(Fair | Fair) P(2 | Fair) P(Fair | Fair) … P(4 | Fair) =

$½ \times (\mathbf{1/6})^{10} \times (\mathbf{0.95})^{9} = .00000000521158647211 = 5.21 \times 10^{-9}$

Example: the Dishonest Casino

- So, the likelihood the die is fair in all this run is just 5.21×10^{-9}



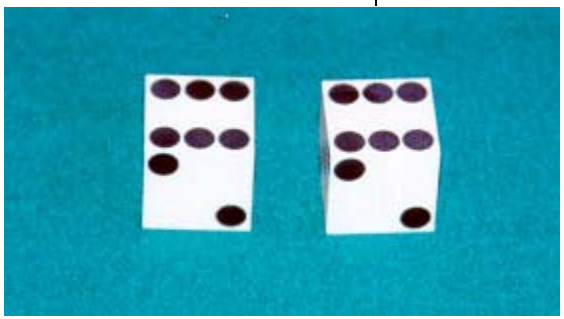
- OK, but what is the likelihood of
  - π = Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded?

$\frac{1}{2} \times P(1 \mid Loaded)\, P(Loaded \mid Loaded) \ldots P(4 \mid Loaded) =$

$\frac{1}{2} \times (\mathbf{1/10})^8 \times (\mathbf{1/2})^2\, (\mathbf{0.95})^9 = .000000000787817621 5 = 0.79 \times 10^{-9}$

- Therefore, it is after all 6.59 times more likely that the die is fair all the way, than that it is loaded all the way

Example: the Dishonest Casino



- Let the sequence of rolls be:
  - x = 1, 6, 6, 5, 6, 2, 6, 6, 3, 6
- Now, what is the likelihood π = F, F, …, F?
  - $\frac{1}{2} \times (1/6)^{10} \times (0.95)^9 = 0.5 \times 10^{-9}$, same as before
- What is the likelihood y = L, L, …, L?

$\frac{1}{2} \times (1/10)^4 \times (1/2)^6 (0.95)^9 = .00000049238235134735 = 5 \times 10^{-7}$

- So, it is 100 times more likely the die is loaded

Three Main Questions on HMMs

1. Evaluation

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

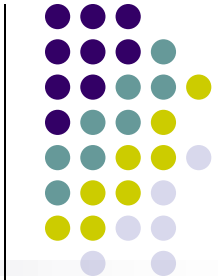
2. Decoding

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

3. Learning

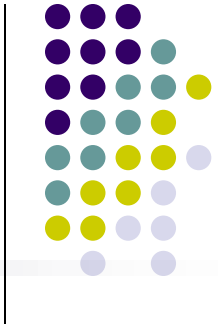
GIVEN an HMM $\boldsymbol{M}$, with unspecified transition/emission probs., and a sequence $\boldsymbol{x}$,
FIND parameters $\theta = (\pi_i, a_{ij}, \eta_{ik})$ that maximize $P(\boldsymbol{x} \mid \theta)$
ALGO. Baum-Welch (EM)

Applications of HMMs



- **Some early applications of HMMs**
  - finance, but we never saw them
  - speech recognition
  - modelling ion channels
- In the mid-late 1980s HMMs entered genetics and molecular biology, and they are now firmly entrenched.
- **Some current applications of HMMs to biology**
  - mapping chromosomes
  - aligning biological sequences
  - predicting sequence structure
  - inferring evolutionary relationships
  - finding genes in DNA sequence

The Forward Algorithm



- We want to calculate $P(\mathbf{x})$, the likelihood of $\mathbf{x}$, given the HMM
  - Sum over all possible ways of generating $\mathbf{x}$:

$$p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}) = \sum_{y_1} \sum_{y_2} \cdots \sum_{y_N} \pi_{y_1} \prod_{t=2}^{T} a_{y_{t-1}, y_t} \prod_{t=1}^{T} p(x_t \mid y_t)$$

  - To avoid summing over an exponential number of paths $\mathbf{y}$, define

$$\alpha(y_t^k = 1) = \alpha_t^k \overset{\text{def}}{=} P(x_1, \ldots, x_t, y_t^k = 1)$$ (the forward probability)

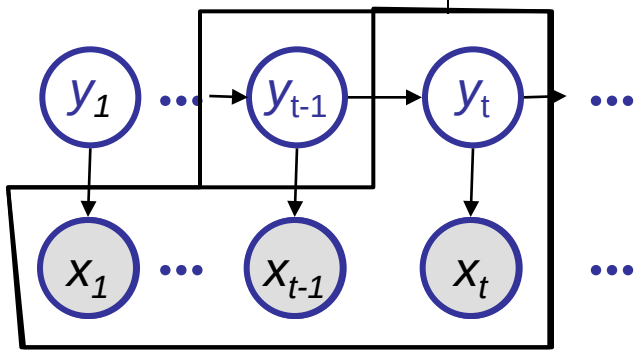
  - The recursion:

$$\alpha_t^k = p(x_t \mid y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k}$$

$$P(\mathbf{x}) = \sum_k \alpha_T^k$$

The Forward Algorithm – derivation

- Compute the forward probability:



$$\alpha_t^k = P(x_1,\ldots,x_{t-1},x_t,y_t^k=1)$$

$$= \sum_{y_{t-1}} P(x_1,\ldots,x_{t-1},y_{t-1})P(y_t^k=1\mid y_{t-1},x_1,\ldots,x_{t-1})P(x_t\mid y_t^k=1,x_1,\ldots,x_{t-1},y_{t-1})$$

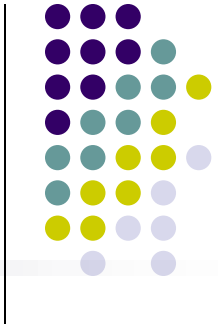
$$= \sum_{y_{t-1}} P(x_1,\ldots,x_{t-1},y_{t-1})P(y_t^k=1\mid y_{t-1})P(x_t\mid y_t^k=1)$$

$$= P(x_t\mid y_t^k=1)\sum_i P(x_1,\ldots,x_{t-1},y_{t-1}^i=1)P(y_t^k=1\mid y_{t-1}^i=1)$$

$$= P(x_t\mid y_t^k=1)\sum_i \alpha_{t-1}^i a_{i,k}$$

Chain rule : $P(A,B,C)=P(A)P(B\mid A)P(C\mid A,B)$

The Forward Algorithm



- We can compute α_t^k for all k, t, using dynamic programming!

Initialization:

$$\alpha_1^k = P(x_1 \mid y_1^k = 1)\pi_k$$

$$\begin{aligned}\alpha_1^k &= P(x_1, y_1^k = 1) \\ &= P(x_1 \mid y_1^k = 1)P(y_1^k = 1) \\ &= P(x_1 \mid y_1^k = 1)\pi_k\end{aligned}$$

Iteration:

$$\alpha_t^k = P(x_t \mid y_t^k = 1)\sum_i \alpha_{t-1}^i a_{i,k}$$

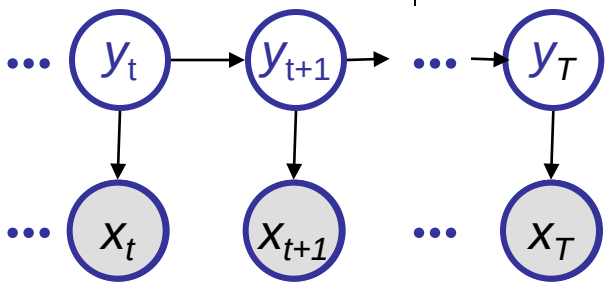
Termination:

$$P(\mathbf{x}) = \sum_k \alpha_T^k$$

The Backward Algorithm

- We want to compute $P(y_t^k = 1 \mid \mathbf{x})$,

  the posterior probability distribution on the t^{th} position, given $\mathbf{x}$



  - We start by computing

$$P(y_t^k = 1, \mathbf{x}) = P(x_1, \ldots, x_t, y_t^k = 1, x_{t+1}, \ldots, x_T)$$
$$= P(x_1, \ldots, x_t, y_t^k = 1) P(x_{t+1}, \ldots, x_T \mid x_1, \ldots, x_t, y_t^k = 1)$$
$$= P(x_1 \ldots x_t, y_t^k = 1) P(x_{t+1} \ldots x_T \mid y_t^k = 1)$$

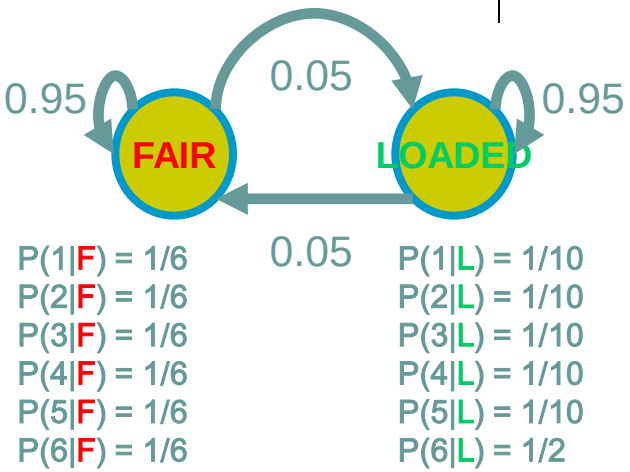
Forward, α_t^k Backward, $\beta_t^k = P(x_{t+1}, \ldots, x_T \mid y_t^k = 1)$

  - The recursion:

$$\beta_t^k = \sum_i a_{k,i}\, p(x_{t+1} \mid y_{t+1}^i = 1)\, \beta_{t+1}^i$$

Example:

x = 1, 2, 1, 5, 6, 2, 1, 6, 2, 4

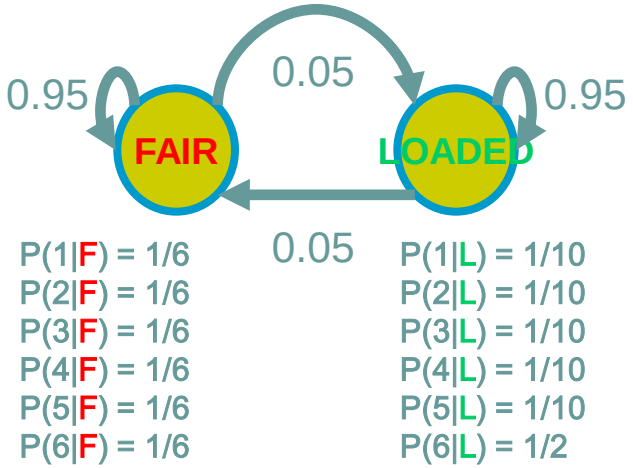


$$\alpha_t^k = P(x_t \mid y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k}$$

$$\beta_t^k = \sum_i a_{k,i} P(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

x = 1, 2, 1, 5, 6, 2, 1, 6, 2, 4

Alpha (actual)		Beta (actual)	
0.0833	0.0500	0.0000	0.0000
0.0136	0.0052	0.0000	0.0000
0.0022	0.0006	0.0000	0.0000
0.0004	0.0001	0.0000	0.0000
0.0001	0.0000	0.0001	0.0001
0.0000	0.0000	0.0007	0.0006
0.0000	0.0000	0.0045	0.0055
0.0000	0.0000	0.0264	0.0112
0.0000	0.0000	0.1633	0.1033
0.0000	0.0000	1.0000	1.0000

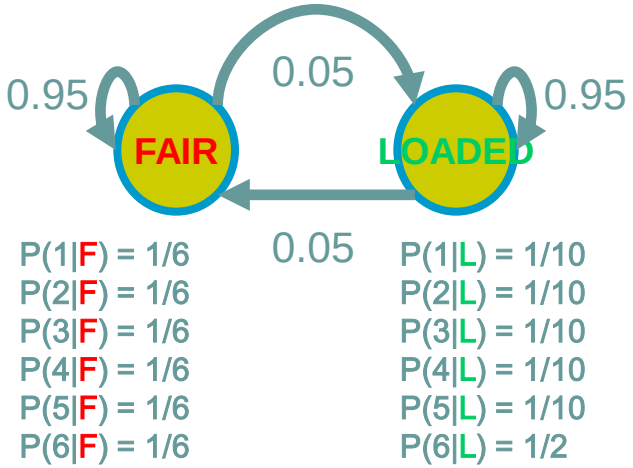


$$\alpha_t^k = P(x_t \mid y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k}$$

$$\beta_t^k = \sum_i a_{k,i} P(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

x = 1, 2, 1, 5, 6, 2, 1, 6, 2, 4

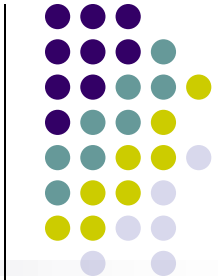
Alpha (logs)		Beta (logs)	
-2.4849	-2.9957	-16.2439	-17.2014
-4.2969	-5.2655	-14.4185	-14.9922
-6.1201	-7.4896	-12.6028	-12.7337
-7.9499	-9.6553	-10.8042	-10.4389
-9.7834	-10.1454	-9.0373	-9.7289
-11.5905	-12.4264	-7.2181	-7.4833
-13.4110	-14.6657	-5.4135	-5.1977
-15.2391	-15.2407	-3.6352	-4.4938
-17.0310	-17.5432	-1.8120	-2.2698
-18.8430	-19.8129	0	0



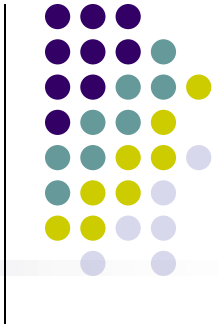
$$\alpha_t^k = P(x_t \mid y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k}$$

$$\beta_t^k = \sum_i a_{k,i} P(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

What is the probability of a hidden state prediction?



Posterior decoding



- We can now calculate

$$P(y_t^k = 1 \mid \mathbf{x}) = \frac{P(y_t^k = 1, \mathbf{x})}{P(\mathbf{x})} = \frac{\alpha_t^k \beta_t^k}{P(\mathbf{x})}$$

- Then, we can ask
  - What is the most likely state at position *t* of sequence **x**:

$$k_t^* = \arg\max_k P(y_t^k = 1 \mid \mathbf{x})$$

  - Note that this is an MPA of a single hidden state, what if we want to a MPA of a whole hidden state sequence?
  - Posterior Decoding: $\left\{ y_t^{k_t^*} = 1 : t = 1 \cdots T \right\}$
  - This is different from MPA of a whole sequence states
  - This can be understood as *bit error rate* vs. *word error rate*

Example:
MPA of *X* ?
MPA of *(X, Y)* ?

of hidden

x	y	P(x,y)
0	0	0.35
0	1	0.05
1	0	0.3
1	1	0.3

Viterbi decoding

- GIVEN $\mathbf{x} = x_1, \ldots, x_T$, we want to find $\mathbf{y} = y_1, \ldots, y_T$, such that $P(\mathbf{y}|\mathbf{x})$ is maximized:

$$\mathbf{y}^* = \text{argmax}_{\mathbf{y}}\, P(\mathbf{y}|\mathbf{x}) = \text{argmax}_{\pi}\, P(\mathbf{y},\mathbf{x})$$

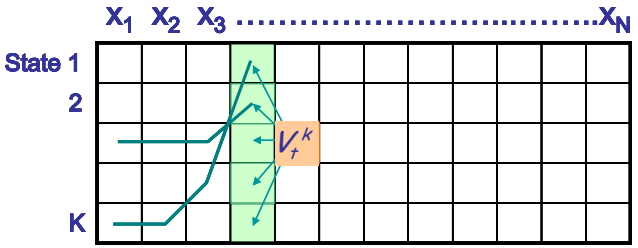
  - Let

$$V_t^k = \max_{\{y_1,\ldots y_{t-1}\}} P(x_1,\ldots,x_{t-1},y_1,\ldots,y_{t-1},x_t,y_t^k = 1)$$

    = Probability of most likely **sequence of states** ending at state $y_t = k$

  - The recursion:

$$V_t^k = p(x_t \mid y_t^k = 1) \max_i a_{i,k} V_{t-1}^i$$



  - Underflows are a significant problem

$$p(x_1,\ldots,x_t,y_1,\ldots,y_t) = \pi_{y_1} a_{y_1,y_2} \cdots a_{y_{t-1},y_t} b_{y_1,x_1} \cdots b_{y_t,x_t}$$

    - These numbers become extremely small – underflow
    - Solution: Take the logs of all values: $V_t^k = \log p(x_t \mid y_t^k = 1) + \max_i \left(\log\left(a_{i,k}\right) + V_{t-1}^i\right)$

Computational Complexity and implementation details

- What is the running time, and space required, for Forward, and Backward?

$$\alpha_t^k = p(x_t \mid y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k}$$

$$\beta_t^k = \sum_i a_{k,i}\, p(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

$$V_t^k = p(x_t \mid y_t^k = 1) \max_i a_{i,k} V_{t-1}^i$$

Time: $O(K^2N)$; Space: $O(KN)$.

- Useful implementation technique to avoid underflows
  - Viterbi: sum of logs
  - Forward/Backward: rescaling at each position by multiplying by a constant

Learning HMM

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

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

  - **Define:**

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

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

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

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

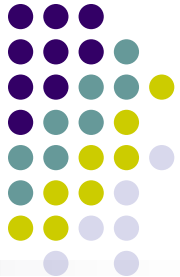
$$b_{ik}^{ML} = \frac{\#(i \to k)}{\#(i \to \bullet)} = \frac{\sum_n \sum_{t=1}^{T} y_{n,t}^{i} x_{n,t}^{k}}{\sum_n \sum_{t=1}^{T} y_{n,t}^{i}} = \frac{B_{ik}}{\sum_{k'} B_{ik'}}$$

(Homework!)

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

(Homework!)

Unsupervised ML estimation



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

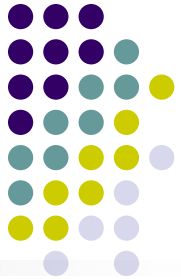
  - **EXPECTATION MAXIMIZATION**

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

  This is called the Baum-Welch Algorithm

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

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

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

- EM
  - The **E** step

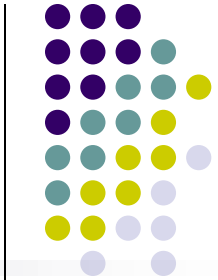
$$\gamma^i_{n,t} = \left\langle y^i_{n,t} \right\rangle = p(y^i_{n,t} = 1 \mid \mathbf{x}_n)$$

$$\xi^{i,j}_{n,t} = \left\langle y^i_{n,t-1} y^j_{n,t} \right\rangle = p(y^i_{n,t-1} = 1, y^j_{n,t} = 1 \mid \mathbf{x}_n)$$

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

$$\pi_i^{ML} = \frac{\sum_n \gamma^i_{n,1}}{N} \qquad a_{ij}^{ML} = \frac{\sum_n \sum_{t=2}^{T} \xi^{i,j}_{n,t}}{\sum_n \sum_{t=1}^{T-1} \gamma^i_{n,t}} \qquad b_{ik}^{ML} = \frac{\sum_n \sum_{t=1}^{T} \gamma^i_{n,t} x^k_{n,t}}{\sum_n \sum_{t=1}^{T-1} \gamma^i_{n,t}}$$

Summary



- Modeling hidden transitional trajectories (in discrete state space, such as cluster label, DNA copy number, dice-choice, etc.) underlying observed sequence data (discrete, such as dice outcomes; or continuous, such as CGH signals)
- Useful for parsing, segmenting sequential data
- Important HMM computations:
  - The joint likelihood of a parse and data can be written as a product to local terms (i.e., initial prob, transition prob, emission prob.)
  - Computing marginal likelihood of the observed sequence: forward algorithm
  - Predicting a single hidden state: forward-backward
  - Predicting an entire sequence of hidden states: viterbi
  - Learning HMM parameters: an EM algorithm known as Baum-Welch