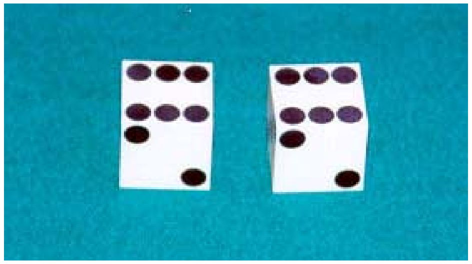
Machine Learning

10-701/15-781, Fall 2012

Hidden Markov Model: dynamic clustering

Eric Xing

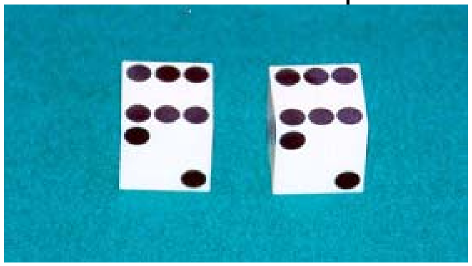


Lecture 12, October 24, 2012

Reading: Chap. 13 CB

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

1245526462146146136136661664661636616366163616515615115146123562344

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

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

- 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

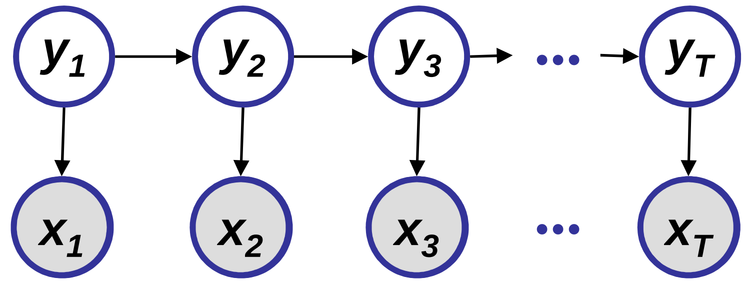
$$p(y_1) \sim \text{Multinomial}(\pi_1, \pi_2, \ldots, \pi_M).$$

- Emission probabilities associated with each state

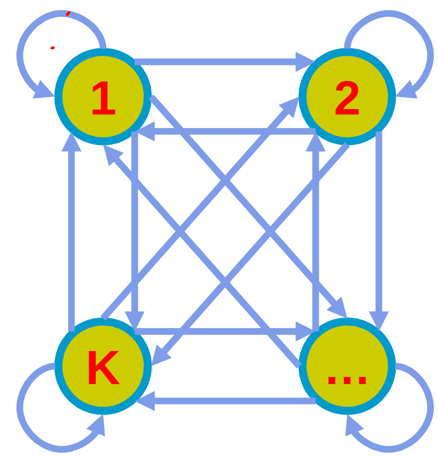
$$p(x_t \mid y_t^i = 1) \sim \text{Multinomial}(b_{i,1}, b_{i,2}, \ldots, b_{i,K}), \forall i \in \mathbb{I}.$$

or in general:

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

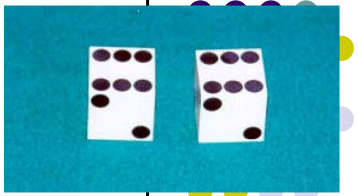


Graphical model

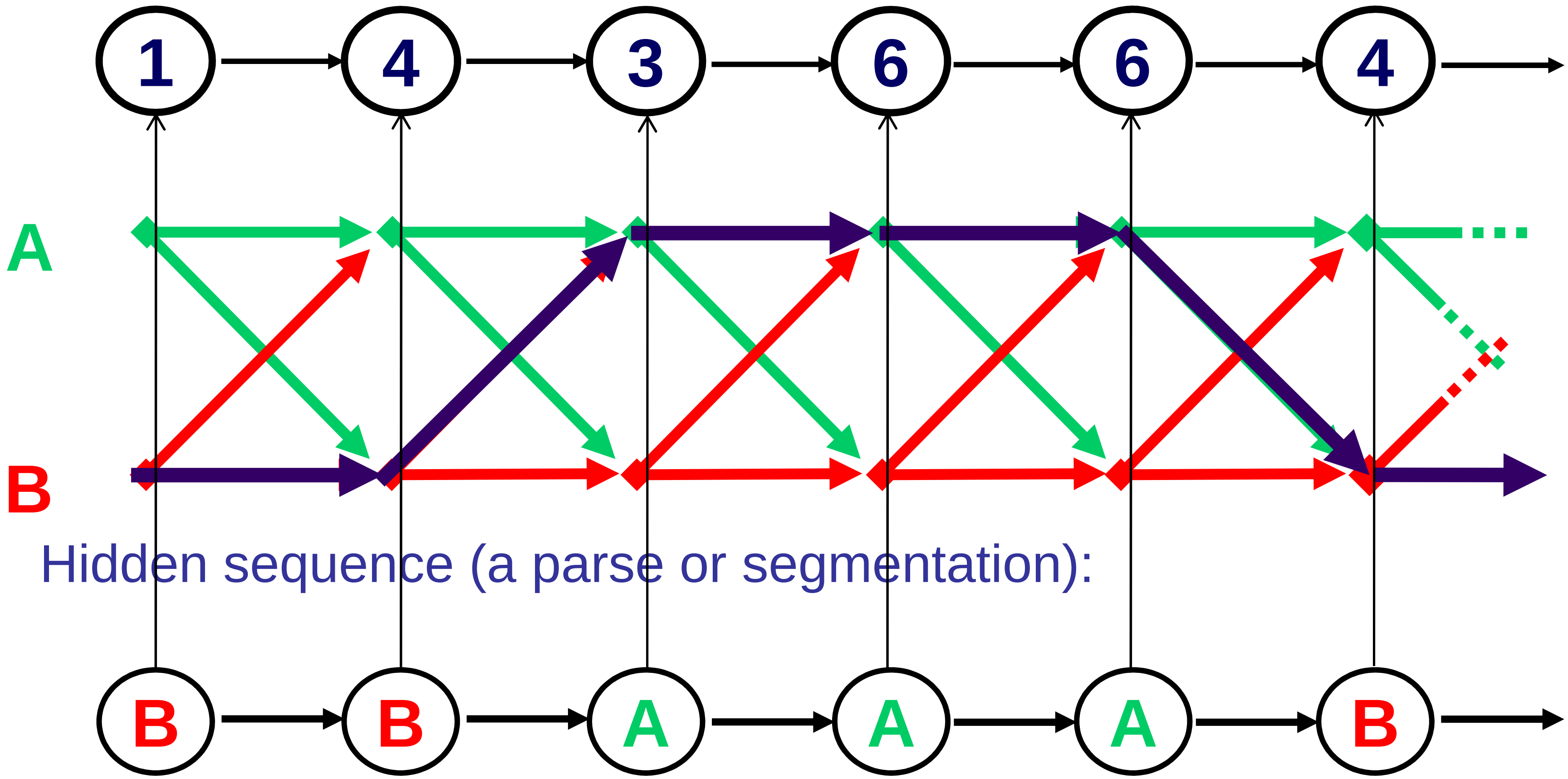


State automata

An HMM is a Stochastic Generative Model

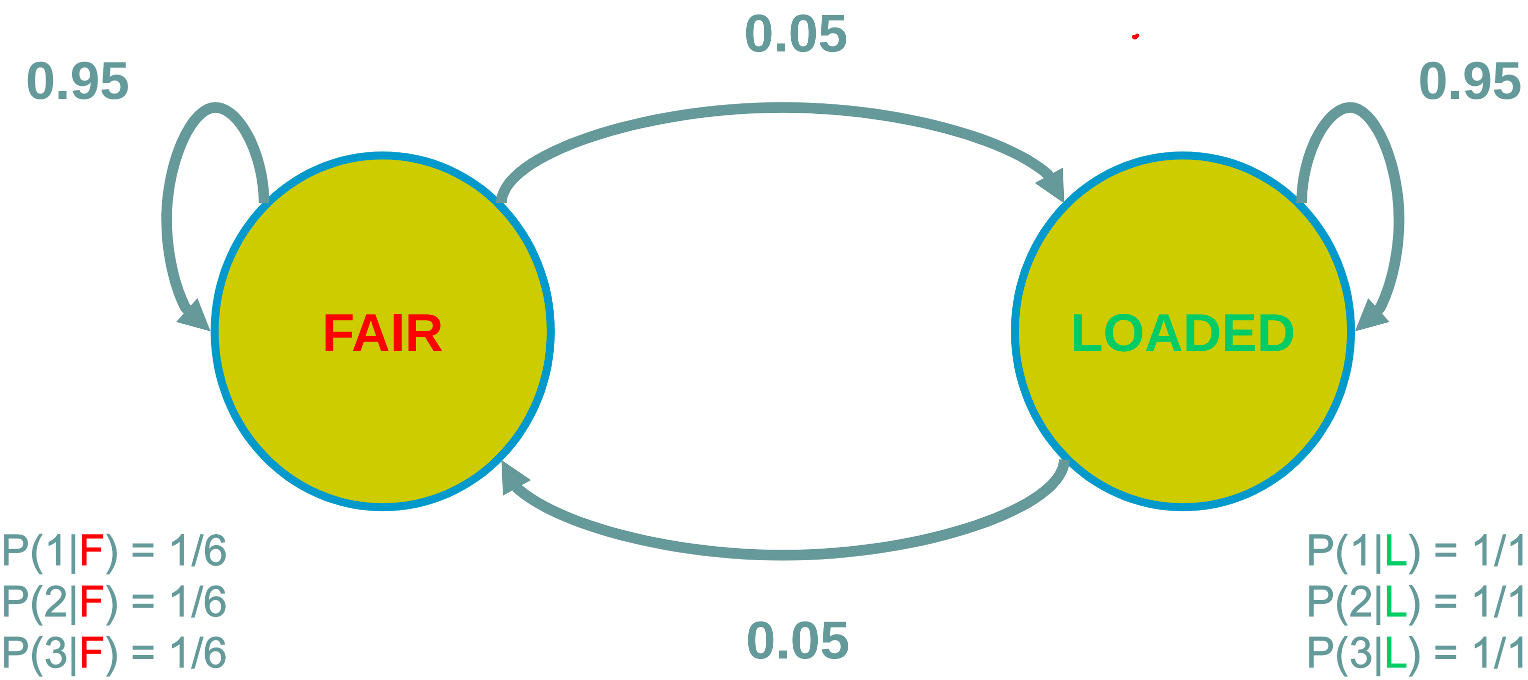


- Observed sequence:



- Hidden sequence (a parse or segmentation):

The Dishonest Casino Model



P(1|F) = 1/6
P(2|F) = 1/6
P(3|F) = 1/6
P(4|F) = 1/6
P(5|F) = 1/6
P(6|F) = 1/6

P(1|L) = 1/10
P(2|L) = 1/10
P(3|L) = 1/10
P(4|L) = 1/10
P(5|L) = 1/10
P(6|L) = 1/2

Three Main Questions on HMMs

1. Evaluation

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

2. Decoding

GIVEN an HMM $\boldsymbol{M}$, and a sequence $\boldsymbol{x}$,
FIND the sequence $\mathbf{y}$ of states that maximizes, e.g., P($\boldsymbol{y}$ | $\boldsymbol{x}$, $\boldsymbol{M}$),
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($\boldsymbol{x}$ | θ)
ALGO. Baum-Welch (EM)

Joint Probability

1245526462146146136136661664661636616366163616515615115146123562344

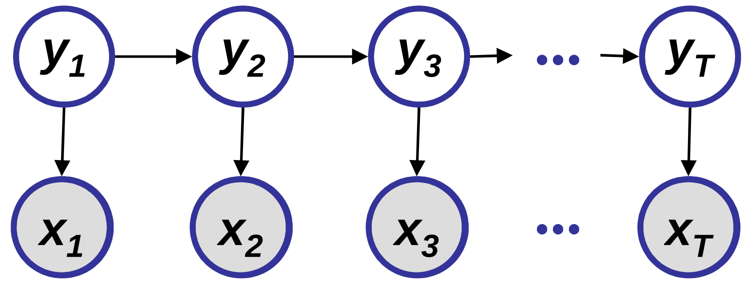
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

- When the state-labeling is known, this is easy …

$$P(\mathbf{X}, \mathbf{Y}) \quad ?$$

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)

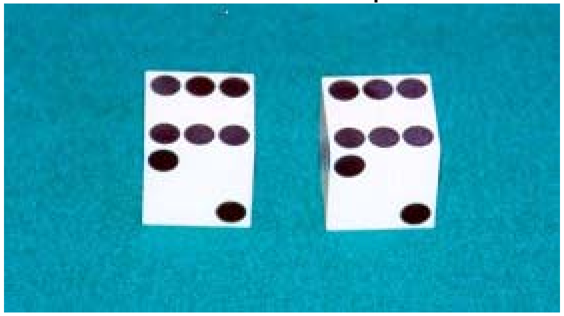


$$p(\mathbf{x}, \mathbf{y}) = p(x_1 \ldots\ldots x_T, y_1, \ldots\ldots, y_T) \quad \text{(Joint probability)}$$

$$= p(y_1)\, p(x_1 \mid y_1)\, p(y_2 \mid y_1)\, p(x_2 \mid y_2) \ldots p(y_T \mid y_{T-1})\, p(x_T \mid y_T)$$

$$= p(y_1)\, P(y_2 \mid y_1) \ldots p(y_T \mid y_{T-1}) \times p(x_1 \mid y_1)\, p(x_2 \mid y_2) \ldots p(x_T \mid y_T)$$

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?

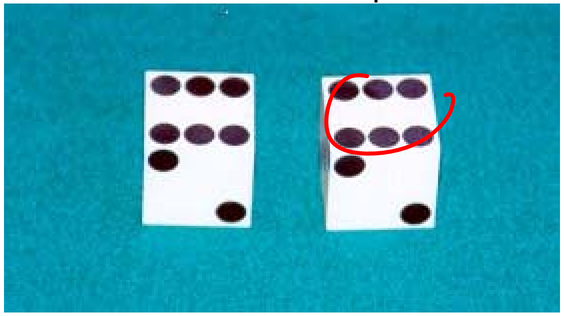
    (say initial probs $a_{0Fair} = ½$, $a_{oLoaded} = ½$)

$½ \times P(1 \mid Fair)\, P(Fair \mid Fair)\, P(2 \mid Fair)\, P(Fair \mid Fair) \ldots P(4 \mid Fair) =$

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

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?

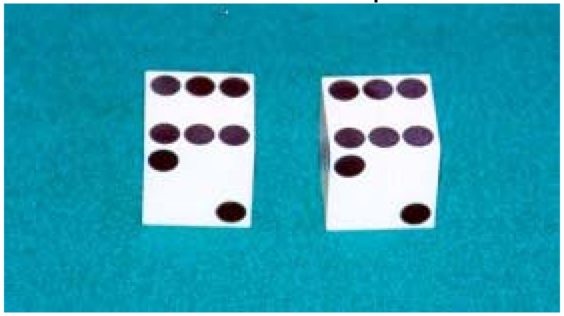
$\frac{1}{2} \times$ P(1 | Loaded) P(Loaded | Loaded) … P(4 | Loaded) =

$\frac{1}{2} \times (\mathbf{1/10})^8 \times (\mathbf{1/2})^2 (\mathbf{0.95})^9 = .00000000078781176215 = 0.79 \times 10^{-9}$

- 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

Marginal Probability

124552646214614613613666166466163661636616361651561511514612356 2344

~~FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF~~

- What if state-labeling Y is not observed

$$P(\mathbf{X}) \quad ?$$

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

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

(the forward probability)

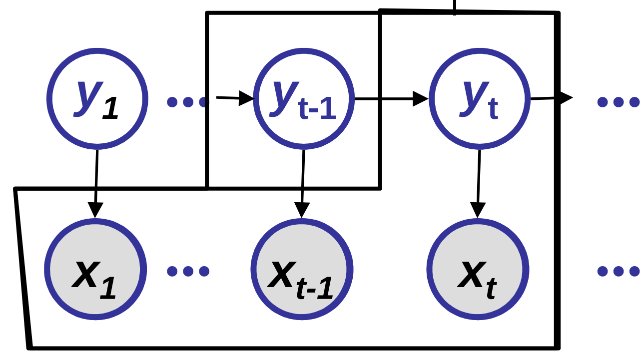
  - The recursion:

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

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

The Forward Algorithm – derivation

- Compute the forward probability:



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

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

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

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

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

$$\text{Chain rule:} \quad P(A,B,C) = P(A)P(B \mid A)P(C \mid A,B)$$

The Forward Algorithm

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

Initialization:

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

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

Iteration:

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

Termination:

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

Three Main Questions on HMMs

1. Evaluation

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

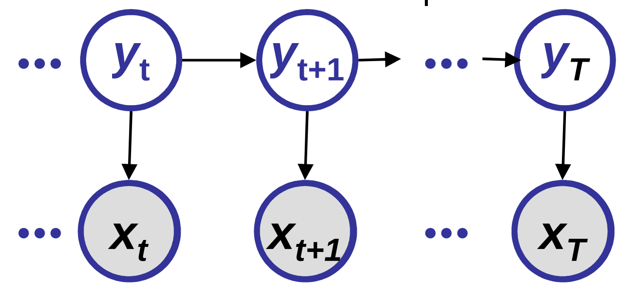
2. Decoding

GIVEN an HMM $\boldsymbol{M}$, and a sequence $\boldsymbol{x}$,
FIND the sequence $\mathbf{y}$ of states that maximizes, e.g., P($\boldsymbol{y}$ | $\boldsymbol{x}$, $\boldsymbol{M}$),
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($\boldsymbol{x}$ | θ)
ALGO. Baum-Welch (EM)

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

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

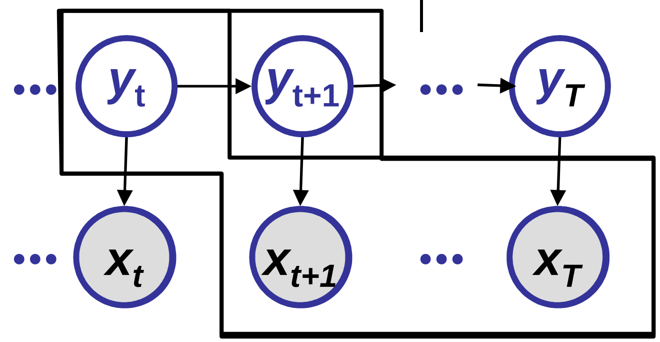
Forward, α_t^k Backward, $\beta_t^k = P(x_{t+1}, \ldots, x_T \mid y_t^k = 1)$

  - The recursion:

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

The Backward Algorithm – derivation

- Define the backward probability:



$$\beta_t^k = P(x_{t+1},\dots,x_T \mid y_t^k = 1)$$

$$= \sum_{y_{t+1}} P(, y_{t+1}, x_{t+1},\dots,x_T \mid y_t^k = 1)$$

$$= \sum_i P(y_{t+1}^i = 1 \mid y_t^k = 1) p(x_{t+1} \mid y_{t+1}^i = 1, y_t^k = 1) P(x_{t+2},\dots,x_T \mid x_{t+1}, y_{t+1}^i = 1, y_t^k = 1)$$

$$= \sum_i P(y_{t+1}^i = 1 \mid y_t^k = 1) p(x_{t+1} \mid y_{t+1}^i = 1) P(x_{t+2},\dots,x_T \mid y_{t+1}^i = 1)$$

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

$$\text{Chain rule:} \quad P(A,B,C \mid \alpha) = P(A \mid \alpha) P(B \mid A,\alpha) P(C \mid A,B,\alpha)$$

The Backward Algorithm

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

Initialization:

$$\beta_T^k = 1, \ \forall k$$

Iteration:

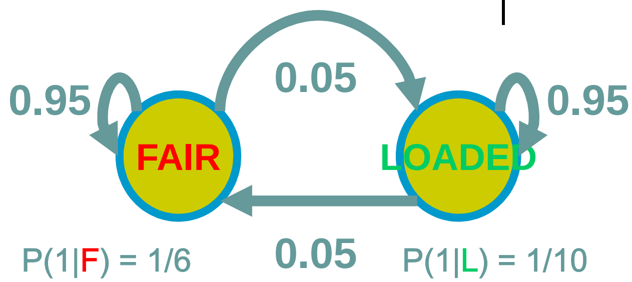
$$\beta_t^k = \sum_i a_{k,i} P(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

Termination:

$$P(\mathbf{x}) = \sum_k \alpha_1^k \beta_1^k$$

Example:

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



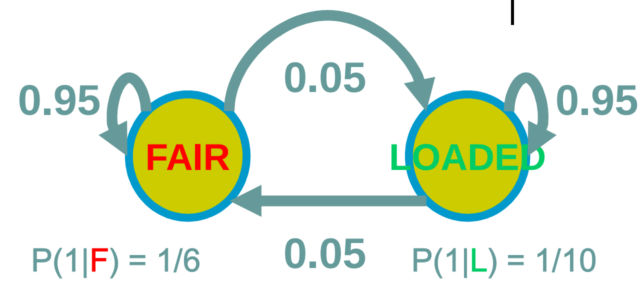
P(1|F) = 1/6
P(2|F) = 1/6
P(3|F) = 1/6
P(4|F) = 1/6
P(5|F) = 1/6
P(6|F) = 1/6

P(1|L) = 1/10
P(2|L) = 1/10
P(3|L) = 1/10
P(4|L) = 1/10
P(5|L) = 1/10
P(6|L) = 1/2

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

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

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



P(1|F) = 1/6
P(2|F) = 1/6
P(3|F) = 1/6
P(4|F) = 1/6
P(5|F) = 1/6
P(6|F) = 1/6

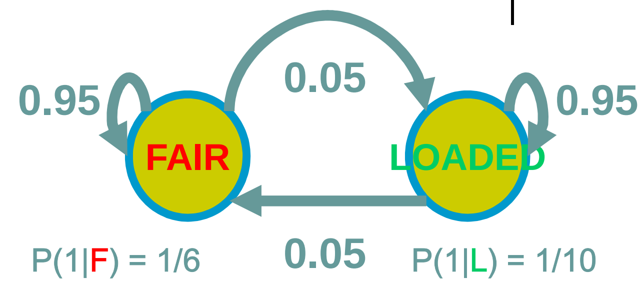
P(1|L) = 1/10
P(2|L) = 1/10
P(3|L) = 1/10
P(4|L) = 1/10
P(5|L) = 1/10
P(6|L) = 1/2

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



P(1|F) = 1/6	P(1|L) = 1/10
P(2|F) = 1/6	P(2|L) = 1/10
P(3|F) = 1/6	P(3|L) = 1/10
P(4|F) = 1/6	P(4|L) = 1/10
P(5|F) = 1/6	P(5|L) = 1/10
P(6|F) = 1/6	P(6|L) = 1/2

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

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

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

What is the probability of a hidden state prediction?

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

What is the probability of a hidden state prediction?

- A single state:

$$P(y_t|\mathbf{X})$$

- What about a hidden state sequence ?

$$P(y_1, \ldots, y_T|\mathbf{X})$$

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 $\mathbf{x}$:

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

  - 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 of hidden states
  - This can be understood as *bit error rate* vs. *word error rate*

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

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:

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

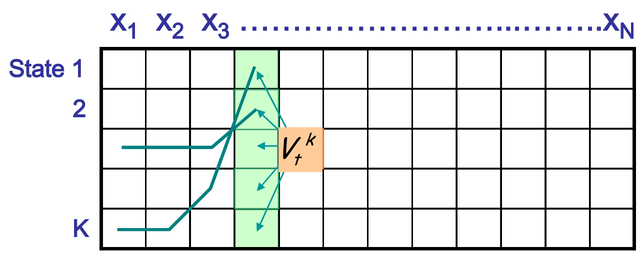
- Let

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

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

The Viterbi Algorithm – derivation

- Define the viterbi probability:

$$
\begin{aligned}
V_{t+1}^k &= \max_{\{y_1,\dots y_t\}} P(x_1,\dots,x_t, y_1,\dots,y_t, x_{t+1,} y_{t+1}^k = 1) \\
&= \max_{\{y_1,\dots y_t\}} P(x_{t+1,} y_{t+1}^k = 1 \mid x_1,\dots,x_t, y_1,\dots,y_t) P(x_1,\dots,x_t, y_1,\dots,y_t) \\
&= \max_{\{y_1,\dots y_t\}} P(x_{t+1,} y_{t+1}^k = 1 \mid y_t) P(x_1,\dots,x_{t-1}, y_1,\dots,y_{t-1}, x_t, y_t) \\
&= \max_i P(x_{t+1,} y_{t+1}^k = 1 \mid y_t^i = 1) \max_{\{y_1,\dots y_{t-1}\}} P(x_1,\dots,x_{t-1}, y_1,\dots,y_{t-1}, x_t, y_t^i = 1) \\
&= \max_i P(x_{t+1,} \mid y_{t+1}^k = 1) a_{i,k} V_t^i \\
&= P(x_{t+1,} \mid y_{t+1}^k = 1) \max_i a_{i,k} V_t^i
\end{aligned}
$$

The Viterbi Algorithm

- Input: $\mathbf{x} = x_1, \ldots, x_T,$

Initialization:

$$V_1^k = P(x_1 \mid y_1^k = 1)\pi_k$$

Iteration:

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

$$\mathrm{Ptr}(k,t) = \arg\max_i a_{i,k} V_{t-1}^i$$

Termination:

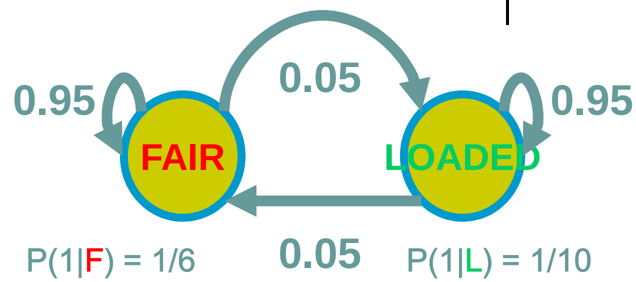
$$P(\mathbf{x}, \mathbf{y}^*) = \max_k V_T^k$$

TraceBack:

$$y_T^* = \arg\max_k V_T^k$$

$$y_{t-1}^* = \mathrm{Ptr}(y_t^*, t)$$

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



P(1|F) = 1/6	P(1|L) = 1/10
P(2|F) = 1/6	P(2|L) = 1/10
P(3|F) = 1/6	P(3|L) = 1/10
P(4|F) = 1/6	P(4|L) = 1/10
P(5|F) = 1/6	P(5|L) = 1/10
P(6|F) = 1/6	P(6|L) = 1/2

$$V_t^k$$

0.6250	**0.3750**
0.7353	**0.2647**
0.8224	**0.1776**
0.8853	**0.1147**
0.7200	**0.2800**
0.8108	**0.1892**
0.8772	**0.1228**
0.7043	**0.2957**
0.7988	**0.2012**
0.8687	**0.1313**

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

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

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

Viterbi Vs. Posterior Decoding (individual)

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

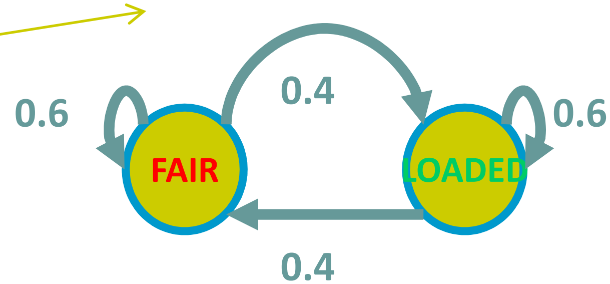
V_t^k		$ptr(k,t)$		Veterbi	PD	$p(y_t^k = 1 \mid x)$	
0.6250	0.3750	N/A		1	1	0.8128	0.1872
0.7353	0.2647	1	2	1	1	0.8238	0.1762
0.8224	0.1776	1	2	1	1	0.8176	0.1824
0.8853	0.1147	1	2	1	1	0.7925	0.2075
0.7200	0.2800	1	2	1	1	0.7415	0.2585
0.8108	0.1892	1	2	1	1	0.7505	0.2495
0.8772	0.1228	1	2	1	1	0.7386	0.2614
0.7043	0.2957	1	2	1	1	0.7027	0.2973
0.7988	0.2012	1	2	1	1	0.7251	0.2749
0.8687	0.1313	1	2	1	1	0.7251	0.2749

Another Example

X = 6, 2, 3, 5, 6, 2, 6, 3, 6, 6

V_t^k		$ptr(k,t)$		Veterbi	PD	$p(y_t^k = 1 \mid x)$	
0.2500	0.7500	N/A		2	2	0.2733	0.7267
0.5263	0.4737	2	2	1	1	0.6040	0.3960
0.6494	0.3506	1	2	1	1	0.6538	0.3462
0.7143	0.2857	1	1	1	1	0.6062	0.3938
0.3333	0.6667	1	1	2	2	0.2861	0.7139
0.5263	0.4737	2	2	2	1	0.5342	0.4658
0.2703	0.7297	1	2	2	2	0.2734	0.7266
0.5263	0.4737	2	2	2	1	0.5226	0.4774
0.2703	0.7297	1	2	2	2	0.2252	0.7748
0.1818	0.8182	2	2	2	2	0.2159	0.7841

max



Same transition probabilities

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

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

Three Main Questions on HMMs

1. Evaluation

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

3. Learning

GIVEN an HMM $\mathcal{M}$, with unspecified transition/emission probs., and a sequence $\boldsymbol{x}$,
FIND parameters $\theta = (\pi_i, a_{ij}, \eta_{ik})$ that maximize P($\boldsymbol{x}$ | θ)
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

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

    A_{ij} = # times state transition $i \rightarrow j$ 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:**

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

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

  - **What if 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 …**

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 $i \rightarrow j$ 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? $A_{ij} = \sum_{n,t} \langle y^i_{n,t-1} y^j_{n,t} \rangle$ $\quad B_{ik} = \sum_{n,t} \langle y^i_{n,t} \rangle x^k_{n,t}$,
   - 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

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

- EM
  - The **E** step

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

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

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

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

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

Summary: the HMM algorithms

Questions:

- **Evaluation**: What is the probability of the observed sequence? Forward
- **Decoding**: What is the probability that the state of the 3rd roll is loaded, given the observed sequence? Forward-Backward
- **Decoding**: What is the most likely die sequence? Viterbi
- **Learning**: Under what parameterization are the observed sequences most probable? Baum-Welch (EM)