Hidden Markov Models

Dr. Jianlin Cheng

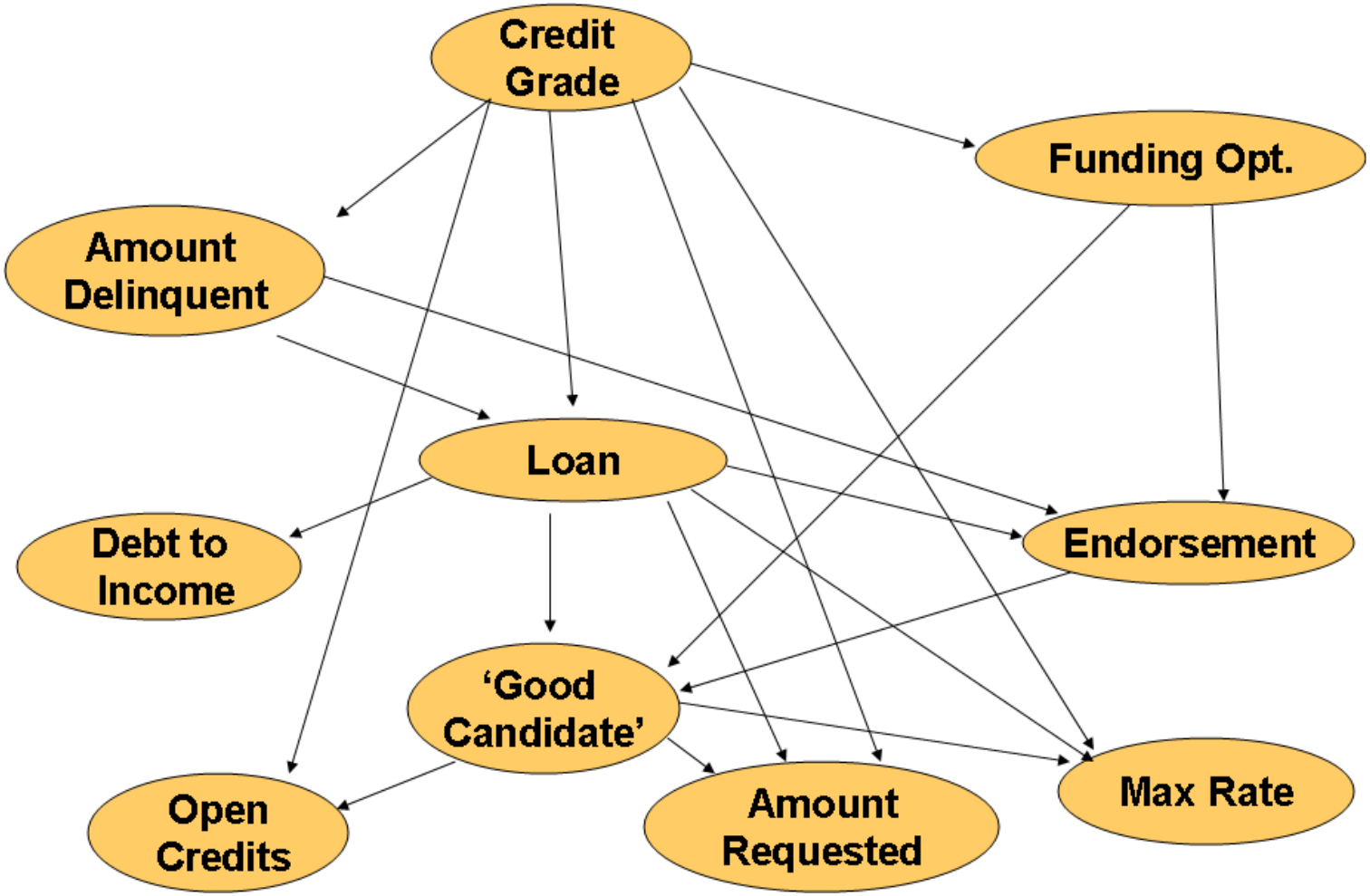
Computer Science Department
University of Missouri, Columbia
Fall, 2015

Slides Adapted from Book and CMU, Stanford Machine Learning Courses

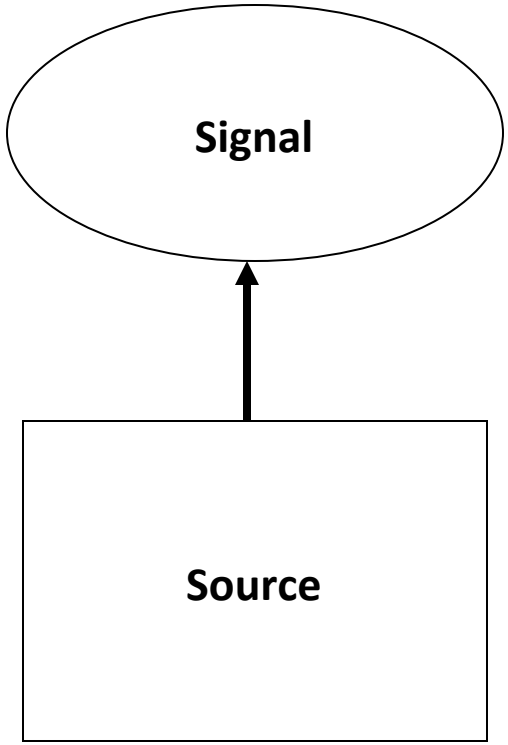
Graphical Model

- **Graphical model** is a probabilistic model for which a graph denotes dependency and independency relationships between random variables (e.g. disease – symptom)
- **Typical models**: hidden Markov models, Bayesian networks, Markov networks
- **Applications**: speech recognition, bioinformatics, image processing, medical informatics, robotics, computer vision

An Example


Credit Grade
Funding Opt.
Amount Delinquent
Loan
Debt to Income
Endorsement
'Good Candidate'
Max Rate
Open Credits
Amount Requested

Real World Process



Discrete signals: characters, nucleotides,…

Continuous signals: speech samples, music temperature measurements, music,…

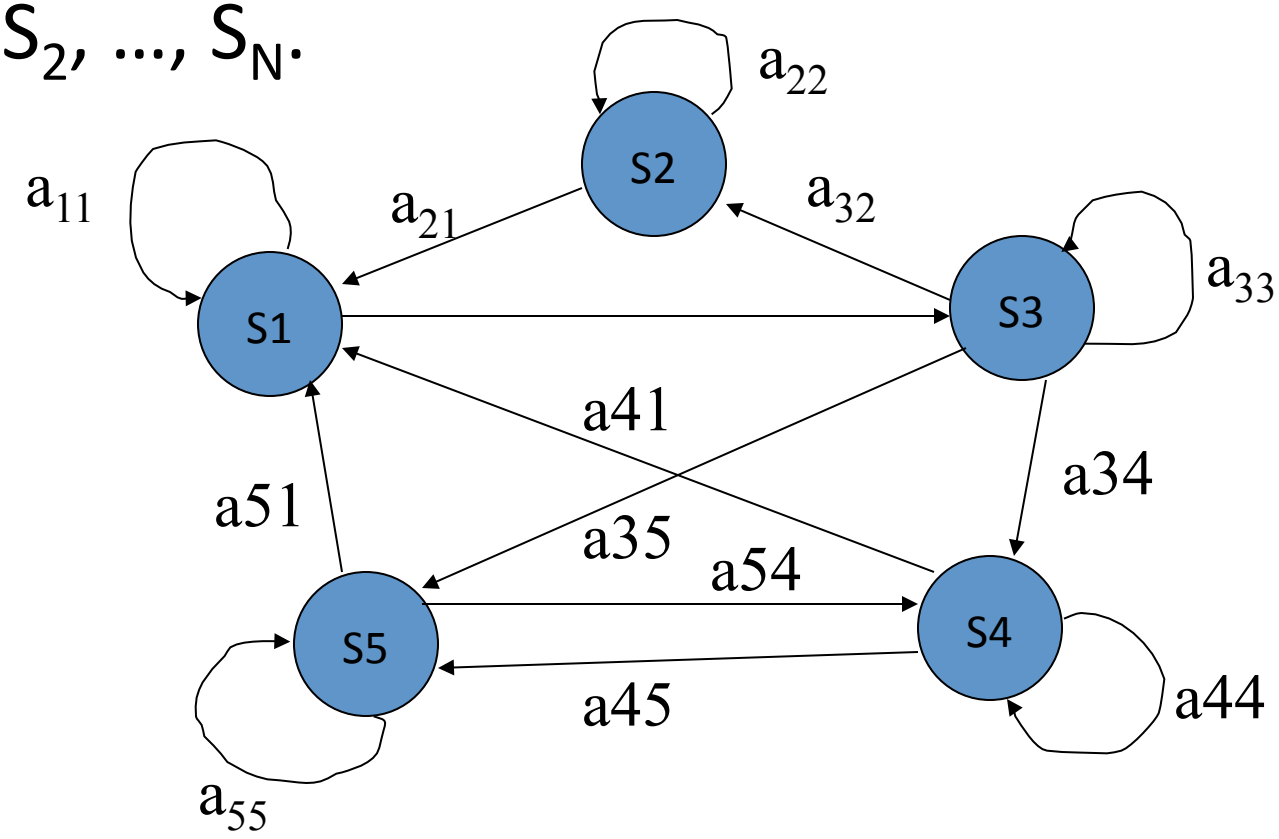
Signal can be pure (from a single source) or be corrupted from other sources (e.g Noise)

Stationary source: its statistical properties does not vary.

Nonstationary source: the signal properties vary over time.

Discrete Markov Process

- Consider a system described at any time as being in one of a set of N distinct states, S_1, S_2, …, S_N.



At regularly spaced discrete times, the system undergoes a change of state according to a set of probabilities.

Definition of Variables

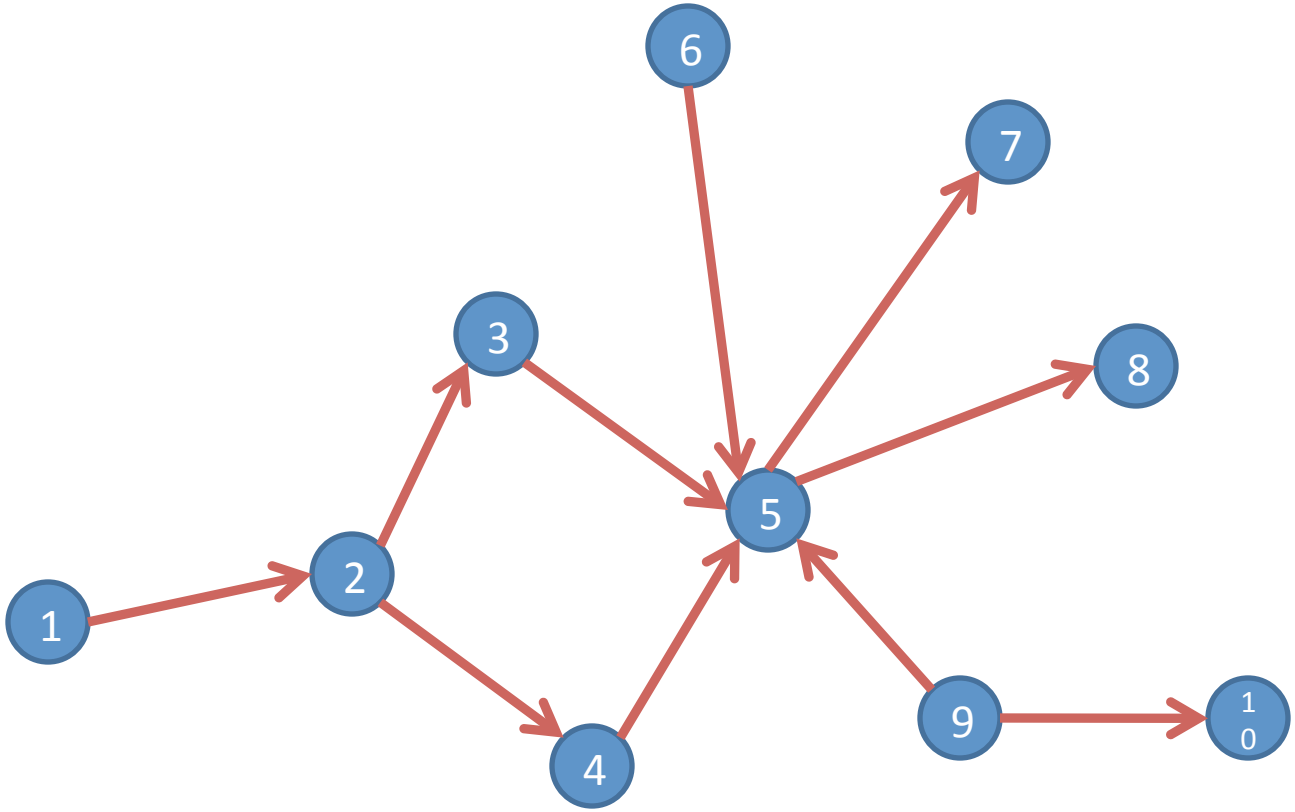
- Time instants associated with state changes as t = 1, 2, ….,T
- Observation: $O = x_1 x_2 \ldots x_T$
- Actual state at time t as y_t, $y_t \in \{S_1, S_2, \ldots, S_j, \ldots, S_M\}$
- A full probabilistic description of the system requires the specification of the current state, as well as all the predecessor states.
- The first order Makov chain (truncation):

  $P[y_t = S_j \mid y_{t-1} = S_i, y_{t-2} = S_k, \ldots] = P[y_t = S_j \mid y_{t-1} = S_i]$.
- Further simplification: state transition is independent of time.

  $a_{ij} = P[y_t = S_j \mid y_{t-1} = S_i]$, $1 <= i, j <= N$, subject to constraints:

  $a_{ij} >= 0$ and $\sum_{j=1}^{N} a_{ij} = 1$

Web as a Markov Model

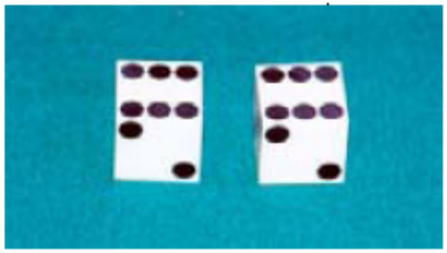


Hidden Markov Models

- Observation is a probabilistic function of the state.
- Doubly embedded process: Underlying stochastic process that is not observable (hidden), but only be observed through another set of stochastic processes that produce the sequence of observations.

A Dice Throwing Example

An experience in a casino



Game:

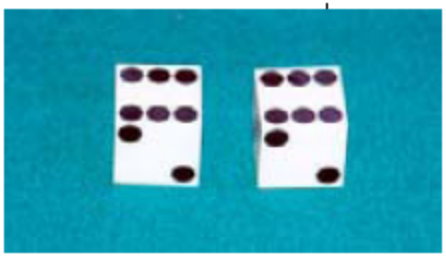
1. You bet $1
2. You roll (always with a fair die)
3. Casino player rolls (maybe with fair die, maybe with loaded die)
4. Highest number wins $2

Question:

12455264621461461361366616646616361636616361651561511514612356234 4

Which die is being used in each play?

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 about Dishonest Casino

GIVEN: A sequence of rolls by the casino player

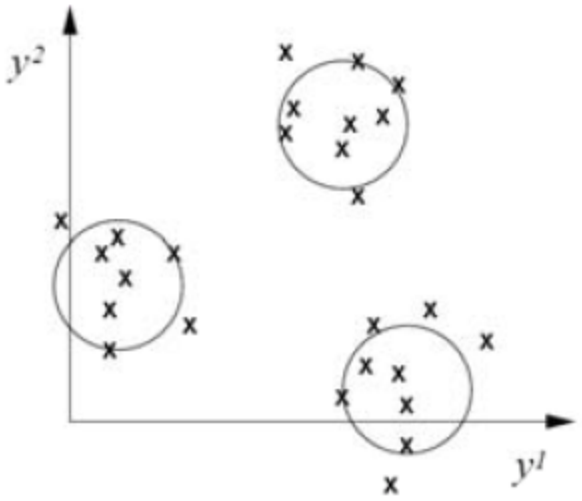
12455264621461461361366616646616366163661636161565151151461235623344

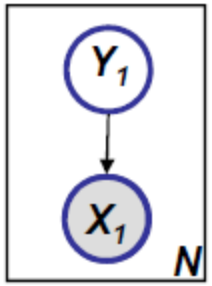
QUESTION

- 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

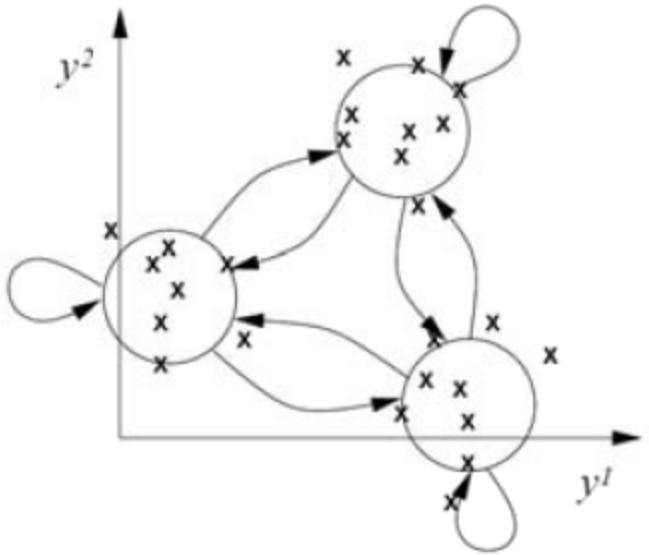
From Static to Dynamic Mixture Model

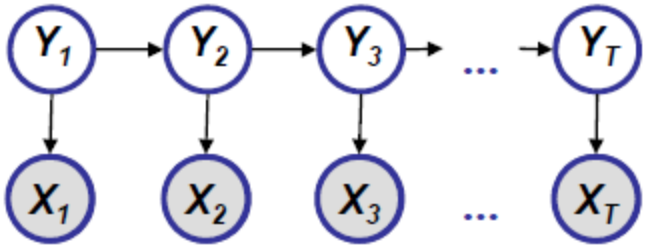
Static mixture





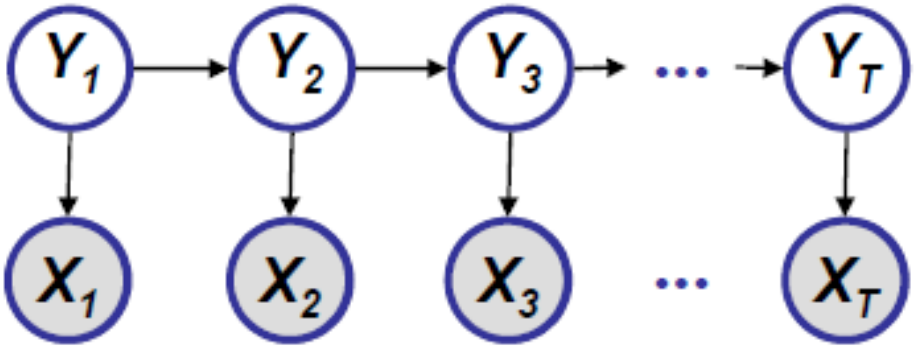
Dynamic mixture





Hidden Markov Models

The underlying source:



dice

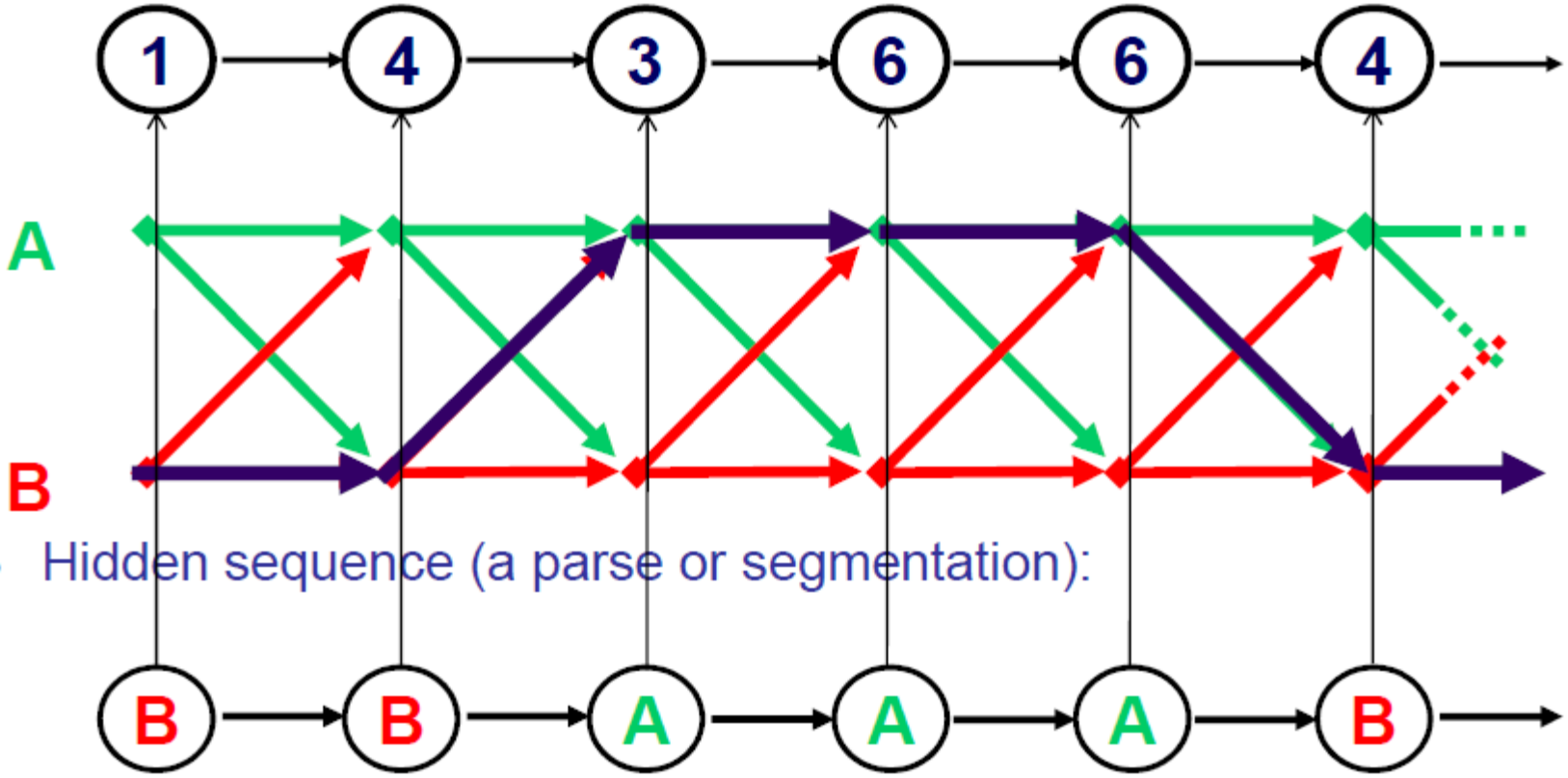
The sequence:

sequence of rolls,

Markov property:

An HMM is a Stochastic Generative Model

- Observed sequence:



- Hidden sequence (a parse or segmentation):

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

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

- Transition probabilities between any two states

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

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

- Start probabilities

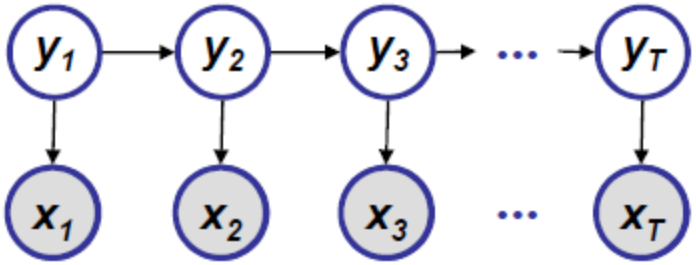
  $p(y_1) \sim \text{Multinomial}(\pi_1, \pi_2, \ldots, \pi_M)$.

- Emission probabilities associated with each state

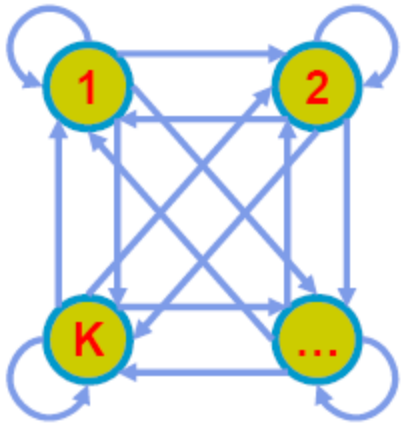
  $p(x_t \mid y_t^i = 1) \sim \text{Multinomial}(b_{i,1}, b_{i,2}, \ldots, b_{i,K}), \forall i \in \mathrm{I}.$

  or in general:

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

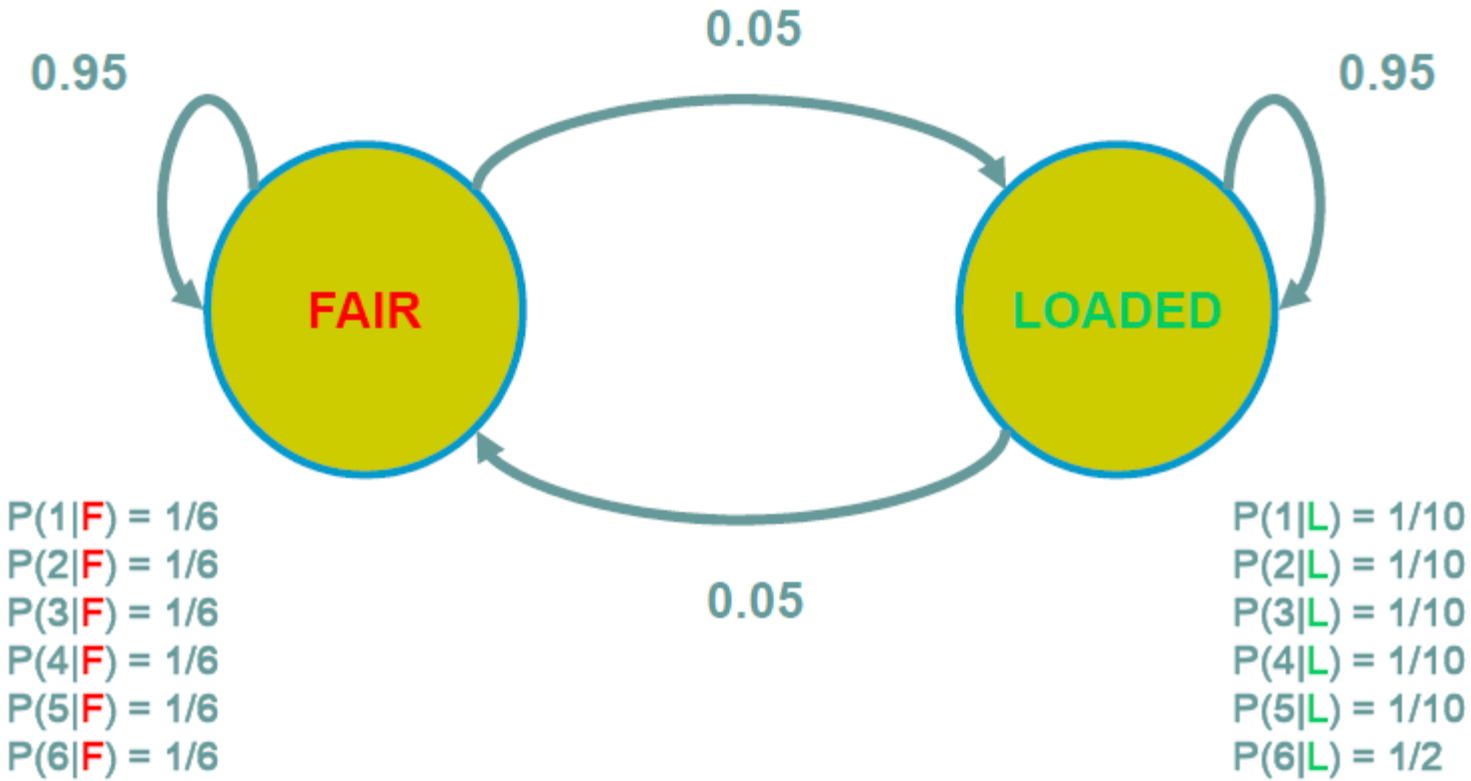


Graphical model



State automata

Dishonest Casino Model



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

Joint Probability

12455264621461461361366616646616366163661636161515615115146123562344

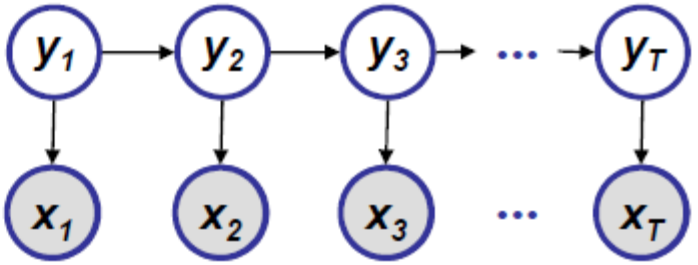
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

- 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)



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

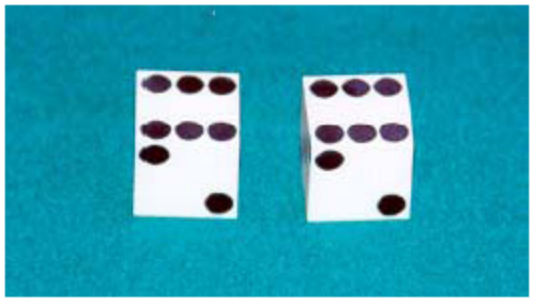
$= p(y_1)\, p(x_1 \mid y_1)\, p(y_2 \mid y_1)\, p(x_2 \mid y_2) \ldots p(y_T \mid y_{T-1})\, p(x_T \mid y_T)$

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

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

An Example: Dishonest Casino

- Let the sequence of rolls be:
  - x = 1, 2, 1, 5, 6, 2, 1, 6, 2, 4



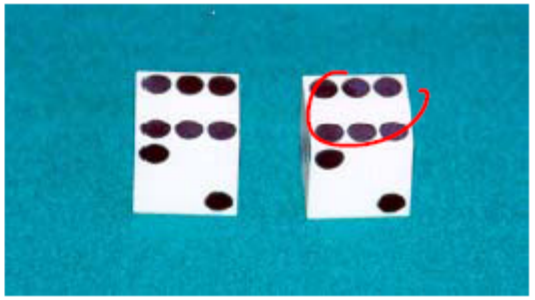
- Then, what is the likelihood of
  - y = Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair?
    (say initial probs $a_{0Fair} = ½$, $a_{oLoaded} = ½$)

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

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

An Example: 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?

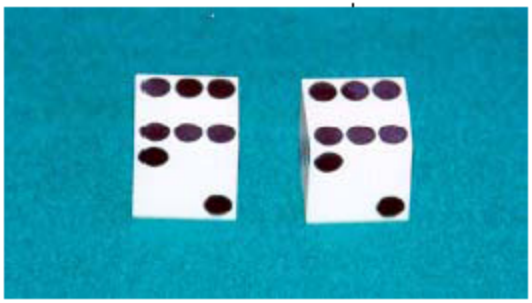
$\frac{1}{2} \times P(1 \mid \text{Loaded})\, P(\text{Loaded} \mid \text{Loaded}) \ldots P(4 \mid \text{Loaded}) =$

$\frac{1}{2} \times (1/10)^8 \times (1/2)^2\, (0.95)^9 = .000000000787811762 15 = 0.79 \times 10^{-9}$

- 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

An Example: Dishonest Casino

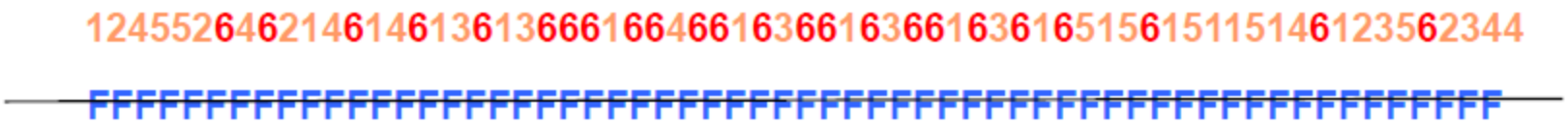
- Let the sequence of rolls be:
  - x = 1, 6, 6, 5, 6, 2, 6, 6, 3, 6
- Now, what is the likelihood π = F, F, …, F?
  - $\frac{1}{2} \times (1/6)^{10} \times (0.95)^9 = 0.5 \times 10^{-9}$, same as before
- What is the likelihood y = L, L, …, L?



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

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

Marginal Probability



- 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 $\mathbf{x}$:

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

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

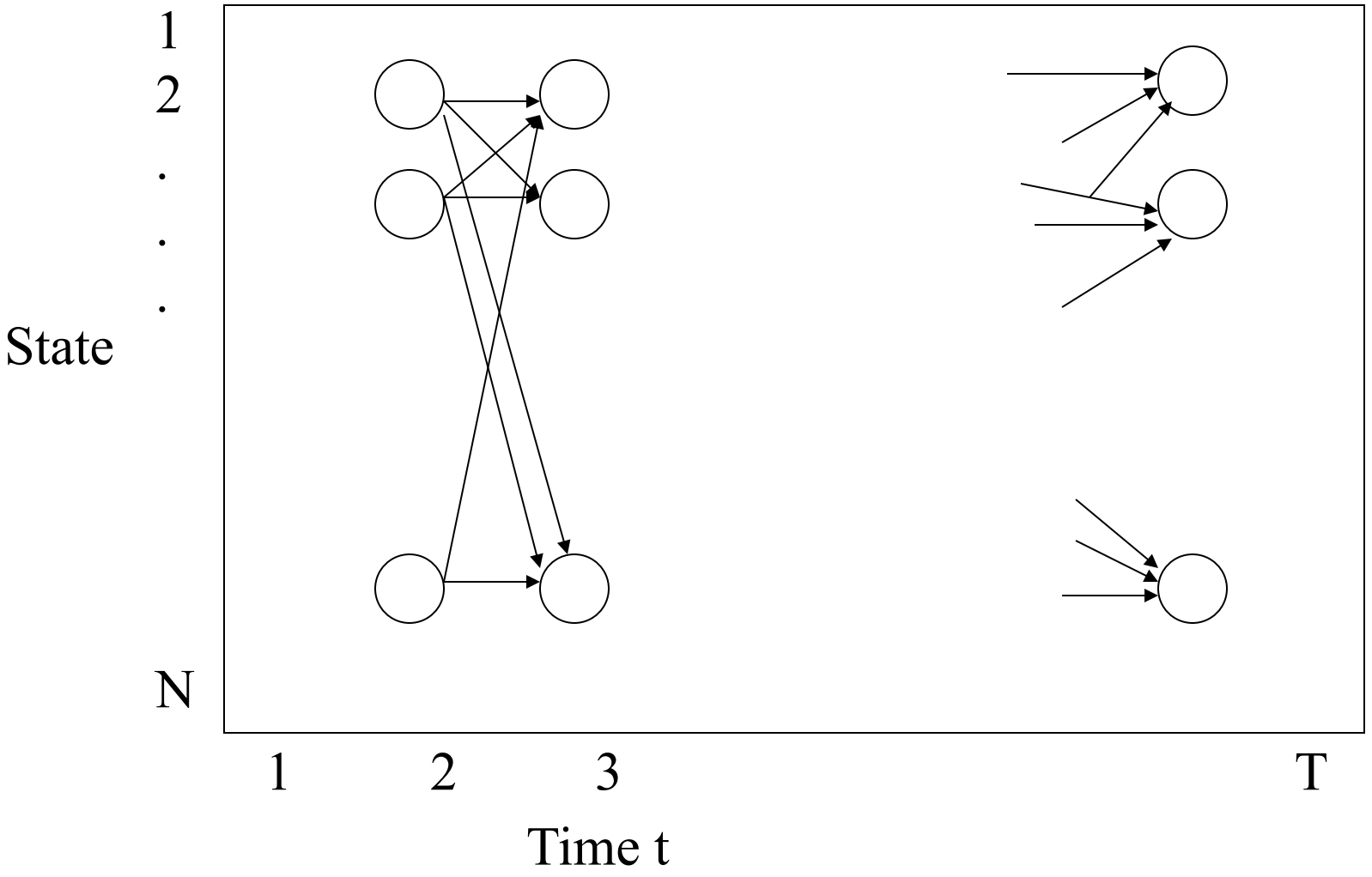
$$\alpha(y_t^k = 1) = \alpha_t^k \overset{\text{def}}{=} P(x_1, \ldots, x_t, y_t^k = 1)$$ (the forward probability)

  - The recursion:

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

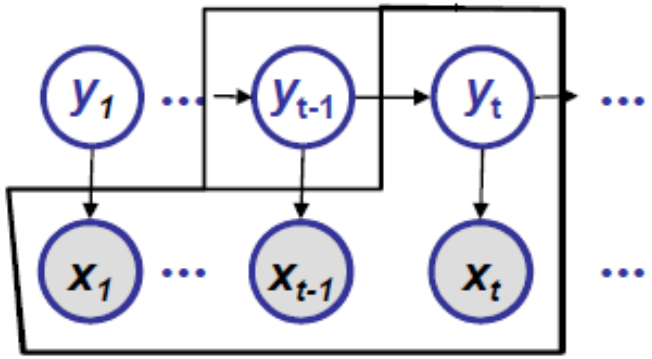
$$P(\mathbf{x}) = \sum_k \alpha_T^k$$

Lattice View



The Forward Algorithm - Derivation

- Compute the forward probability:



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

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

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

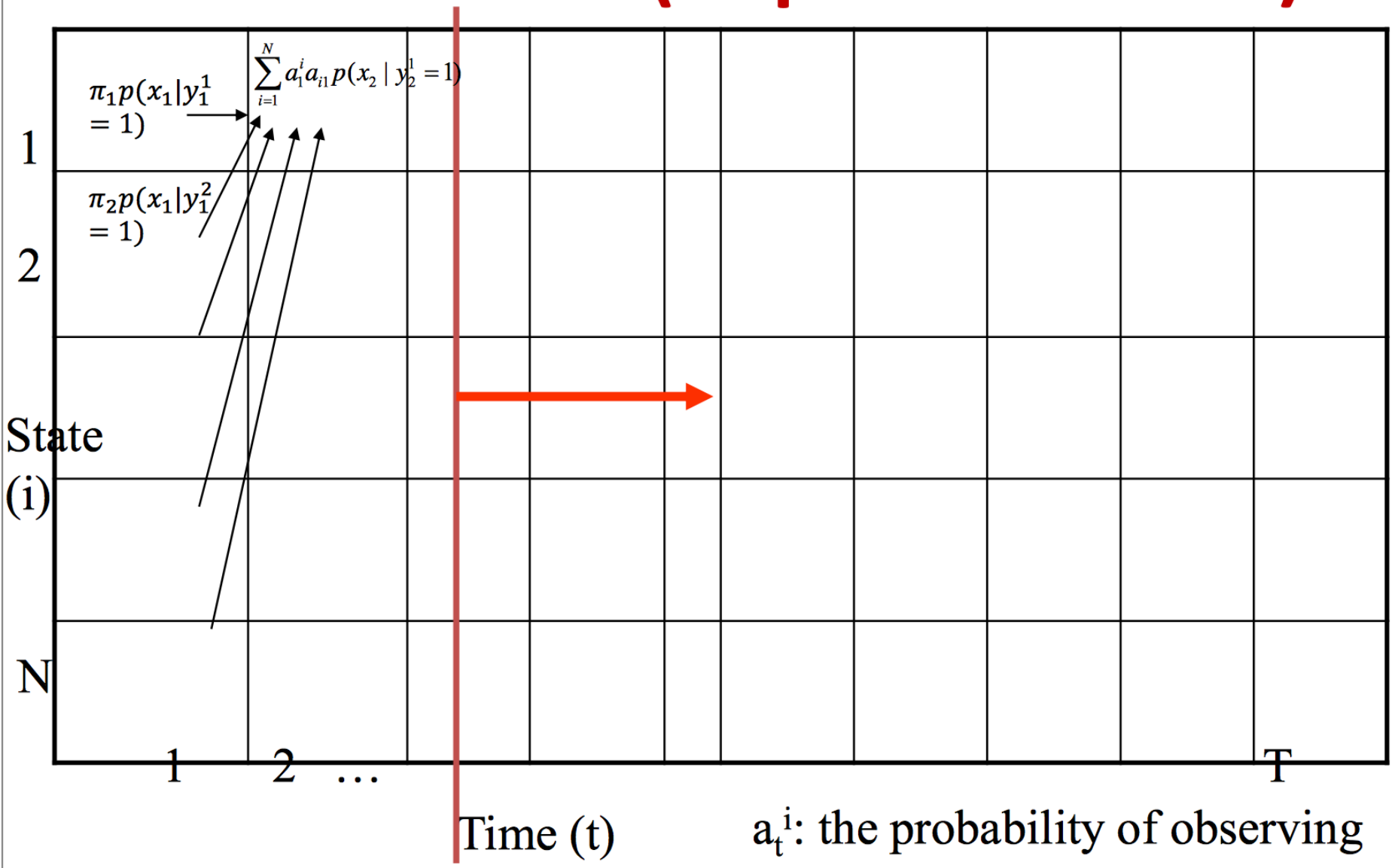
$$= \sum_{y_{t-1}} P(x_1,\ldots,x_{t-1},y_{t-1})P(y_t^k=1\mid y_{t-1})P(x_t\mid y_t^k=1)$$

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

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

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

Matrix View (Implementation)



a_t^i: the probability of observing $x_1 \ldots x_t$ and staying in state i.

Fill the matrix column by column.

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:

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

Time Complexity

- The computation is performed for all states i, for a given time t; the computation is then iterated for t=1,2,…T-1. Finally the desired is the sum of the terminal forward variable a_T^i. This is the case since $a_T^i = P(x_1 x_2 \ldots x_T, y_T^i = 1 | \lambda)$.
- Time complexity: M^2T.

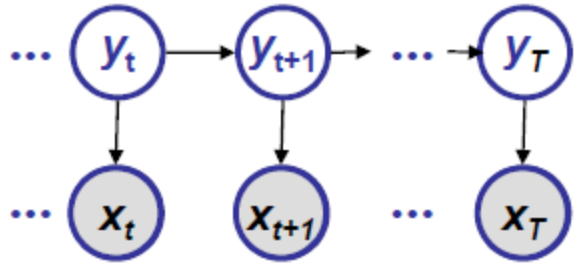
Insights

- Key: there are only M states at each time slot in the lattice, all the possible state sequences will merge into these M nodes, no matter how long the observation sequence.
- Similar to Dynamic Programming (DP trick).

The Backward Algorithm

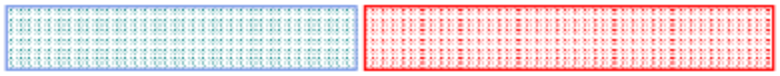
- We want to compute $P(y_t^k = 1 \mid \mathbf{x})$,

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



  - We start by computing

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

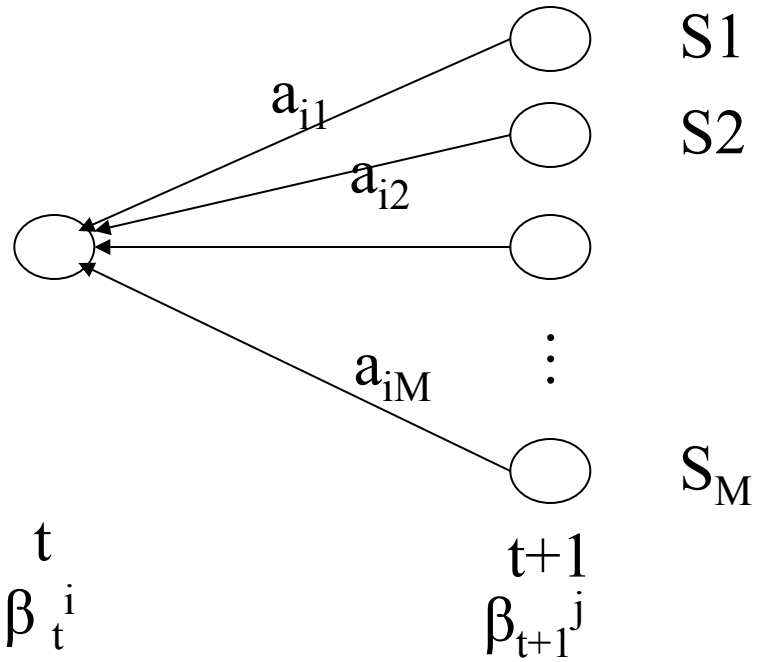


Forward, α_t^k Backward, $\beta_t^k = P(x_{t+1}, \ldots, x_T \mid y_t^k = 1)$

  - The recursion:

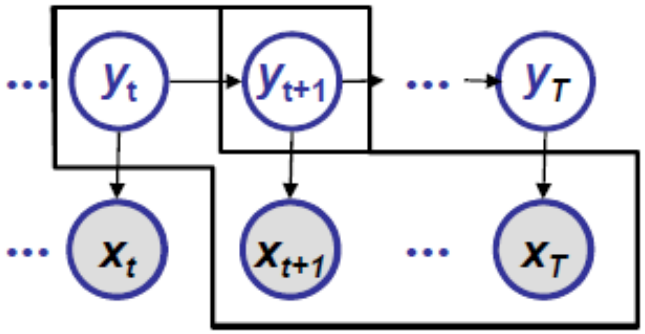
$$\beta_t^k = \sum_i a_{k,i} p(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

Backward Algorithm



The Backward Algorithm - Derivation

- Define the backward probability:



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

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

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

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

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

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:

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

An Example

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

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

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

An Example

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

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

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

$\pi_1 = \pi_2 = 0.5$

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

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

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

Implementation Issue: Scaling

- To compute a_t^i and b_t^i, Multiplication of a large number of terms (probability), value heads to 0 quickly, which exceed the precision range of any machine.
- The basic procedure is to multiply them by a scaling coefficient that is independent of i (i.e., it depends only on t). Logarithm cannot be used because of summation. But we can use $c_t = \frac{1}{\sum_{i=1}^{N} a_t^i}$

C_t will be stored for the time points when the scaling is performed. C_t is used for both a_t^i and b_t^i. The scaling factor will be canceled out for parameter estimation.

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 **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*

Viterbi Algorithm

- 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:

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

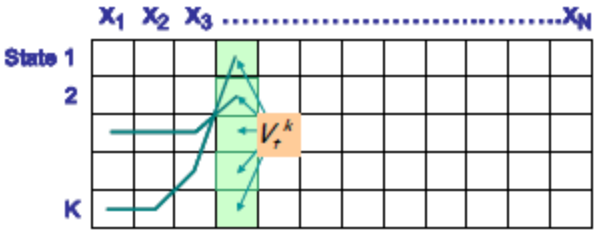
  - Let

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

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

Viterbi Algorithm - derivation

- Define the viterbi probability:

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

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

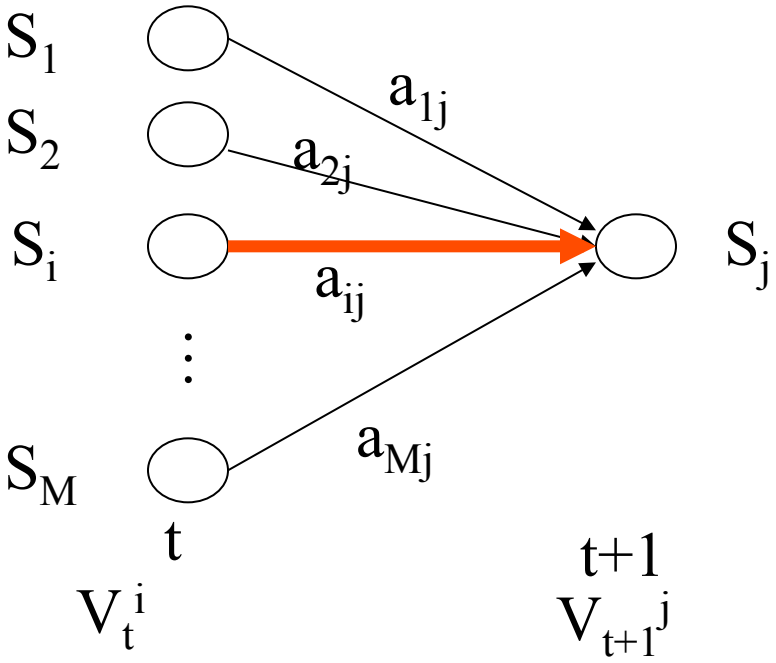
$$= \max_{\{y_1,\ldots y_t\}} P(x_{t+1},y_{t+1}^k = 1 \mid y_t) P(x_1,\ldots,x_{t-1},y_1,\ldots,y_{t-1},x_t,y_t)$$

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

$$= \max_i P(x_{t+1}, \mid y_{t+1}^k = 1) a_{i,k} V_t^i$$

$$= P(x_{t+1}, \mid y_{t+1}^k = 1) \max_i a_{i,k} V_t^i$$

Induction



Choose the transition step yielding the maximum probability.

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:

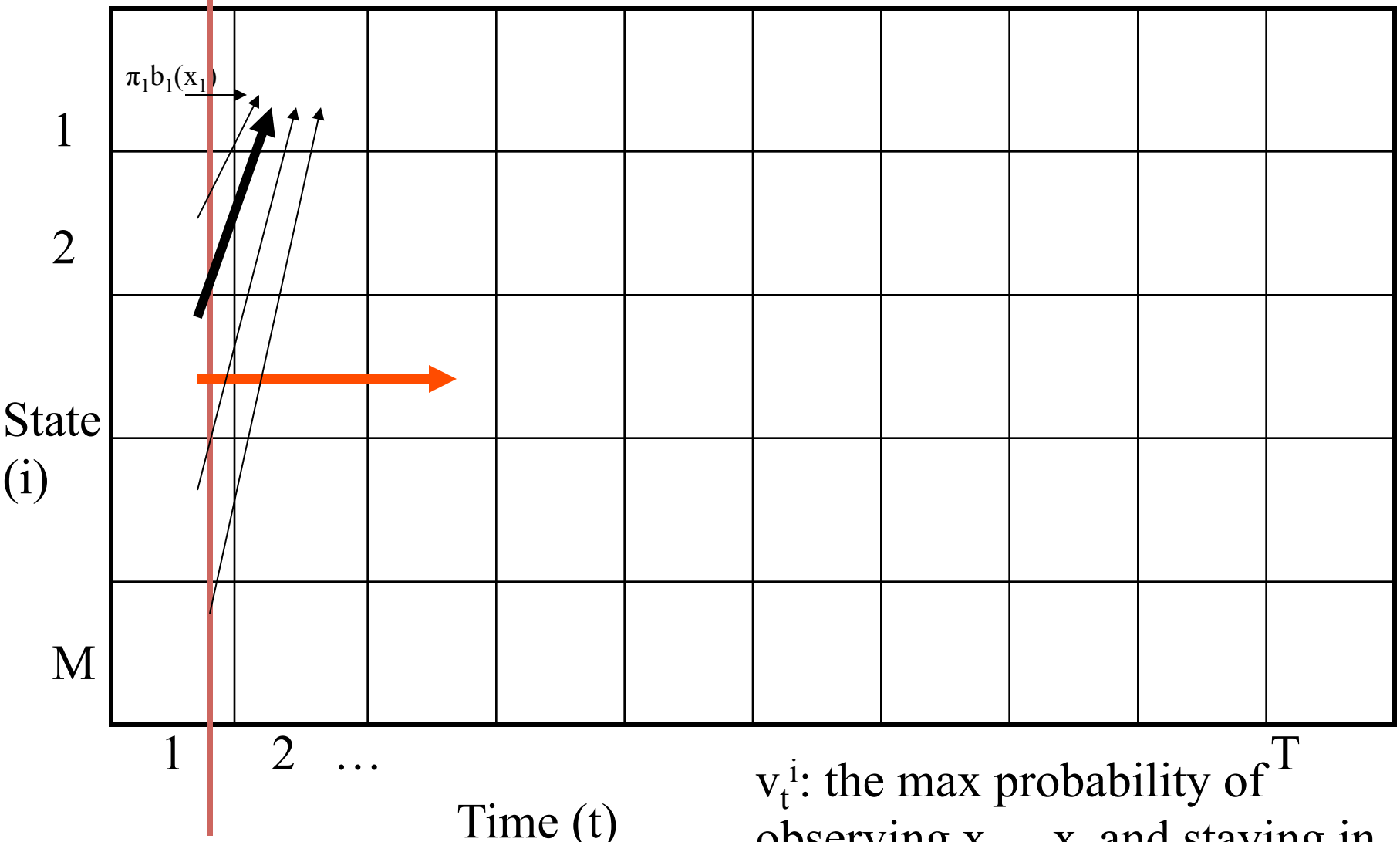
$$P(\mathbf{x}, \mathbf{y}^*) = \max_k V_T^k$$

TraceBack:

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

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

Matrix View (Implementation)



v_t^i: the max probability of observing $x_1 \ldots x_t$ and staying in state i.

Fill the matrix column by column

Viterbi VS. MPA

V_t^k (log)		$ptr(k,t)$		Seq	Veterbi	MPA	$p(y_t^k = 1 \mid x)$	
-2.4849	-1.3863	N/A		6	2	2	0.2733	0.7267
-4.0943	-4.1997	2	2	2	1	1	0.6040	0.3960
-6.3969	-7.0131	1	2	3	1	1	0.6538	0.3462
-8.6995	-9.6158	1	1	5	1	1	0.6062	0.3938
-11.0021	-10.3090	1	1	6	2	2	0.2861	0.7139
-13.0170	-13.1224	2	2	2	2	1	0.5342	0.4658
-15.3196	-14.3263	1	2	6	2	2	0.2734	0.7266
-17.0344	-17.1397	2	2	3	2	1	0.5226	0.4774
-19.3370	-18.3437	1	2	6	2	2	0.2252	0.7748
-21.0518	-19.5477	2	2	6	2	2	0.2159	0.7841

max

Same transition probabilities

Computational Complexity and Implementation Issues

- 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 Problems on HMMs

1. Evaluation

   GIVEN an HMM M, and a sequence x,
   FIND Prob (x | M)
   ALGO. Forward

2. Decoding

   GIVEN an HMM M, and a sequence x,
   FIND the sequence y of states that maximizes, e.g., P(y | x, 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 | 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 …**

Maximum Likelihood Problems

- **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: $a_{FF} = 1$; $a_{FL} = 0$
    $b_{F1} = b_{F3} = .2$;
    $b_{F2} = .3$; $b_{F4} = 0$; $b_{F5} = b_{F6} = .1$

Pseudo-counts

- 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**

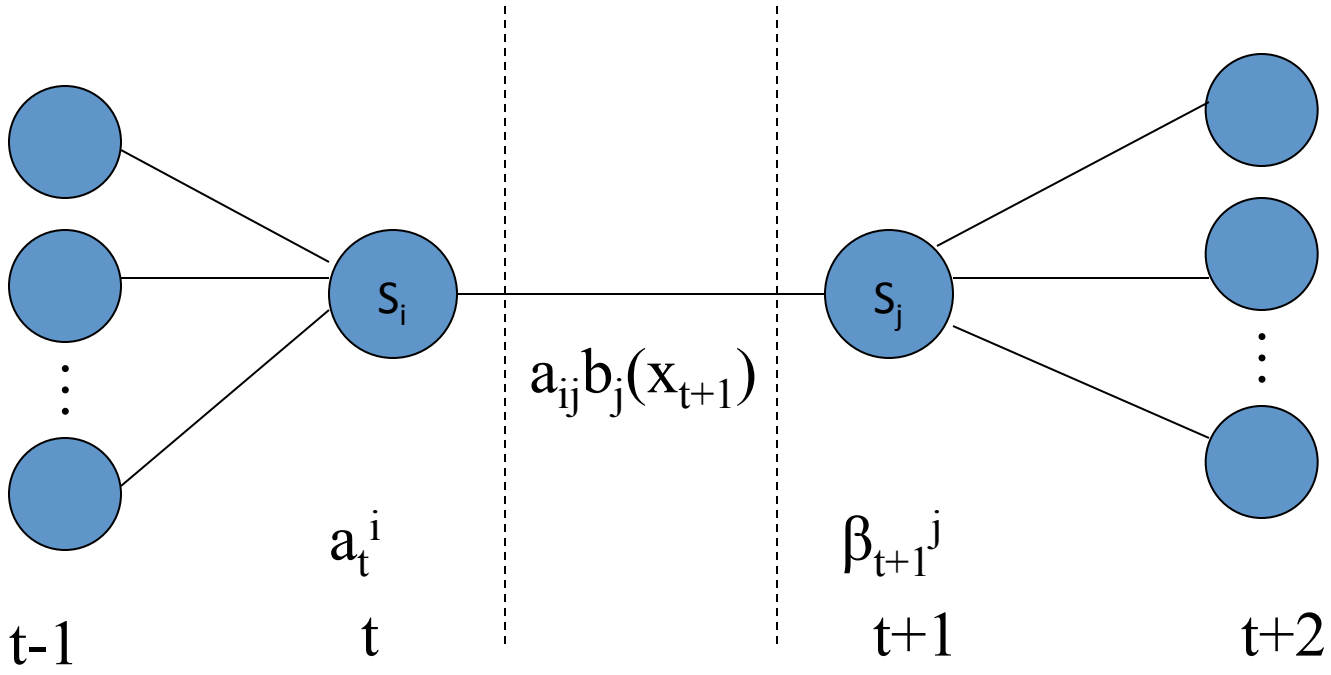
  0. Starting with our best guess of a model M, parameters θ.
  1. Estimate A_{ij} , B_{ik} in the training data
     - How? $A_{ij} = \sum_{n,t} \langle y^i_{n,t-1} y^j_{n,t} \rangle \quad B_{ik} = \sum_{n,t} \langle y^i_{n,t} \rangle x^k_{n,t}$,
     - Update θ according to A_{ij} , B_{ik}
     - Now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

Baum-Welch Algorithm

- Definition: $\xi_t(i,j)$, the probability of being in state S_i at time t and state S_j at time t+1, given the model and observation sequence, i.e. $\xi_t(i,j) = P(y_t^i=1, y_{t+1}^j=1 | X,\lambda)$



From the definitions of the forward and backward variables, we can write $\xi_t(i,j)$ in the form:

$$\xi_t(i,j) = \frac{a_t^i a_{ij} b_j(x_{t+1}) \beta_{t+1}^j}{P(X \mid \lambda)}$$

$$= \frac{\alpha_t^i a_{ij} b_j(x_{t+1}) \beta_{t+1}^j}{\sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_t^i a_{ij} b_j(x_{t+1}) \beta_{t+1}^j}$$

Important Quantities

- γ_t^i is the probability of being in state S_i at time t, given the observation and the model, hence we can relate γ_t^i to $\xi_t(i,j)$ by summing over j, giving

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

$$\sum_{1}^{T-1} \gamma_t(i) \quad = \text{expected number of transitions from } S_i.$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) \quad = \text{expected number of transitions from } S_i \text{ to } S_j.$$

Re-estimation of HMM parameters

A set of reasonable re-estimation formulas for π, A, and B are:

$\overline{\pi}_i$ = expected frequency (number of times) in state S_i at time (t=1) = $\gamma_1(i)$

$$\overline{a}_{ij} = \frac{\text{Expected number of transitions from state } S_i \text{ to state } S_j}{\text{Expected number of transitions from state } S_i.} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\overline{b}_j(k) = \frac{\text{expected number of times in state j and observing symbol } v_k}{\text{expected number of times in state j}} = \frac{\sum_{t=1, s.t. O_t = V_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Baum-Welch Algorithm

- **Initialize** model $\lambda = (A,B,\pi)$
- **Repeat**

E-step: Use forward/backward algorithm to expected frequencies (γ, ξ), given the current model λ and O.

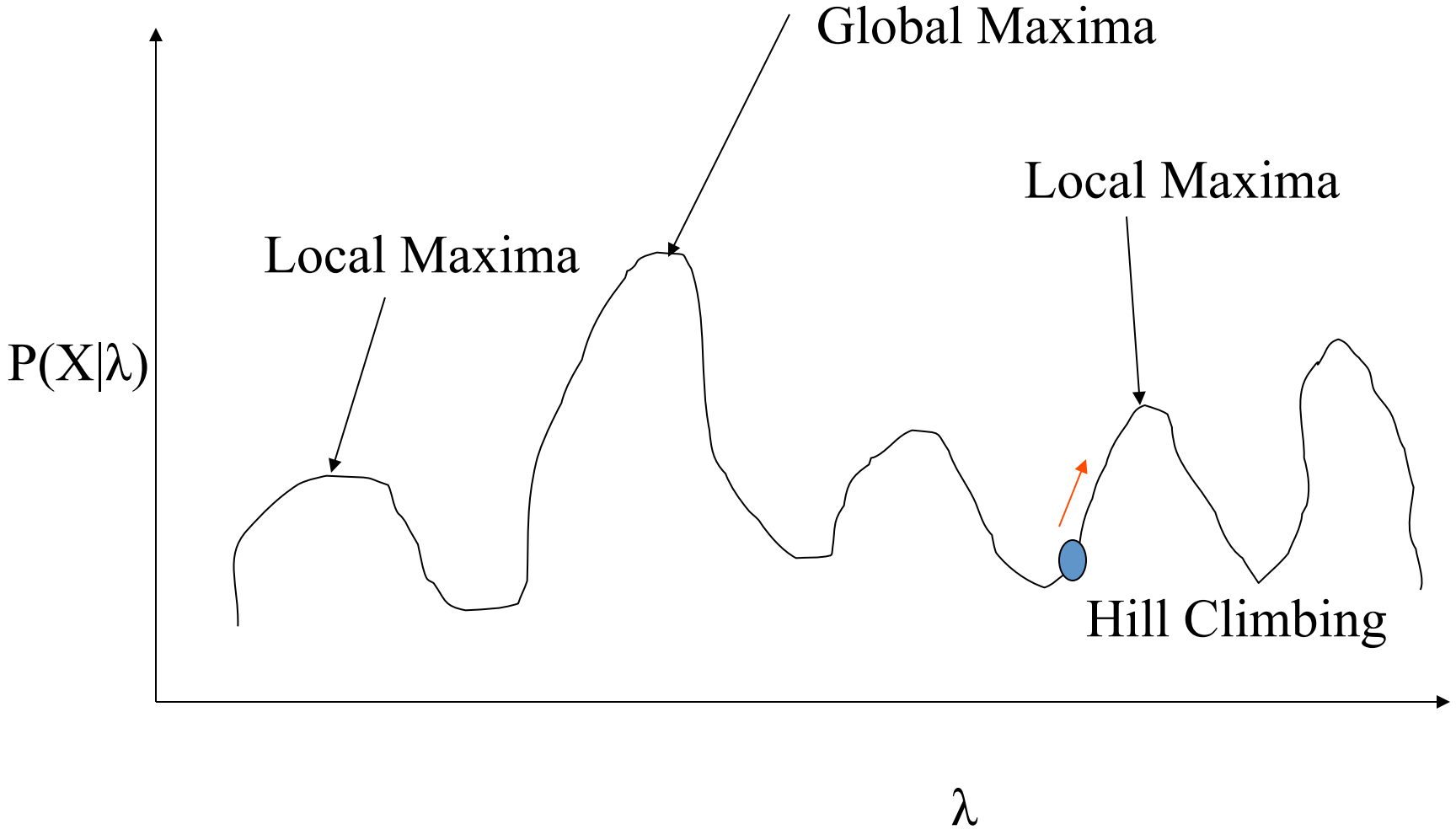
M-step: Use expected frequencies to compute the new model λ.

If ($\overline{\lambda} = \lambda$), stops, otherwise, set $\lambda = \overline{\lambda}$ and go to repeat.

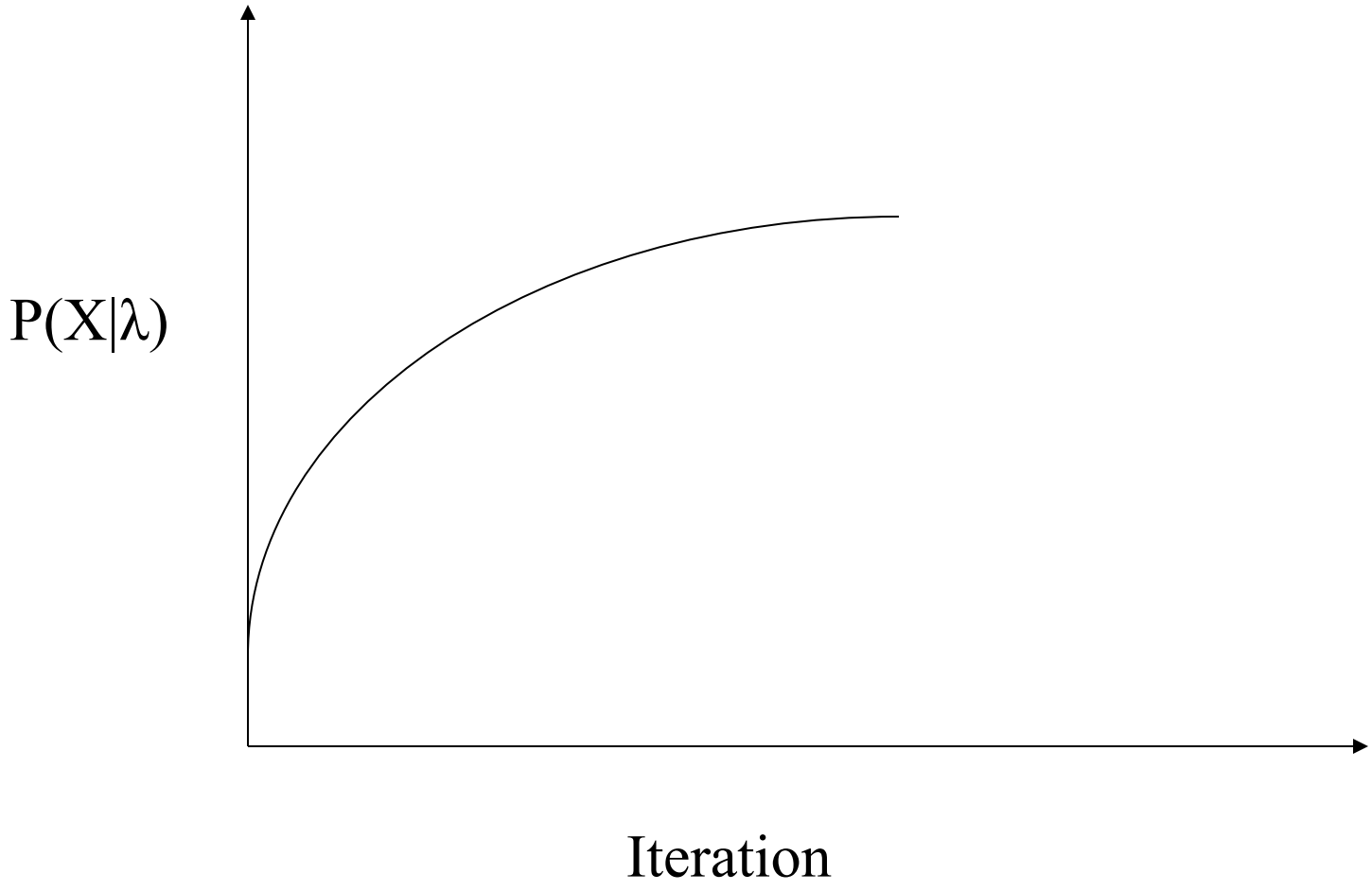
The final result of this estimation procedure is called a maximum likelihood estimate of the HMM. (local maxima)

Local Optimal Guaranteed

If we define the current model as $\lambda=(A,B,\pi)$ and use it to compute the new model $\lambda=(A,B,\pi)$, it has been proven by Baum et al. that either 1) initial model λ defines a critical point, in which case $\lambda = \lambda$; or 2) λ is more likely than λ in the sense that $P(O|\lambda) > P(O|\lambda)$, i.e., we have found a new model λ from which the observation sequence is more likely to have been produced.



Likelihood monotonically crease per iteration until it converges to local maxima.

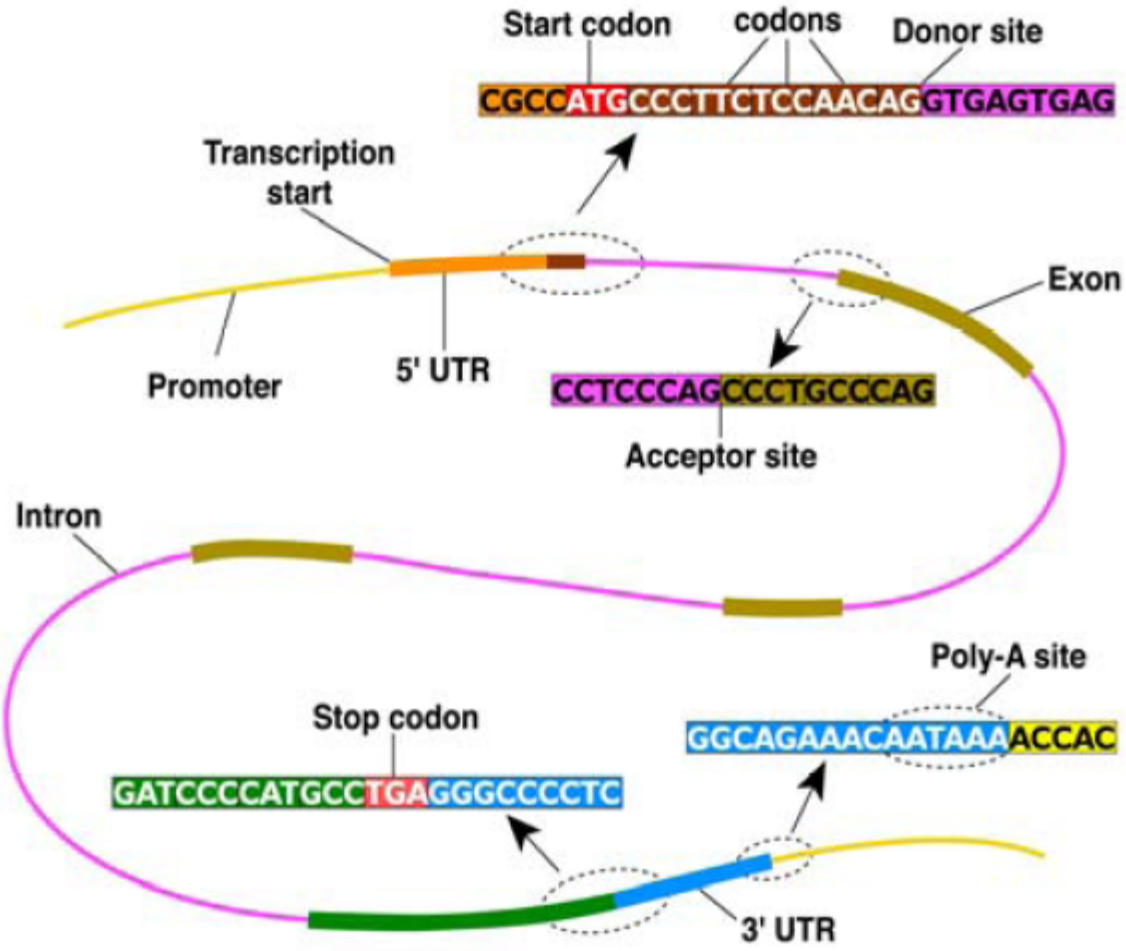


Likelihood monotonically increases per iteration until it converges to local maxima. It usually converges very fast in several iterations.

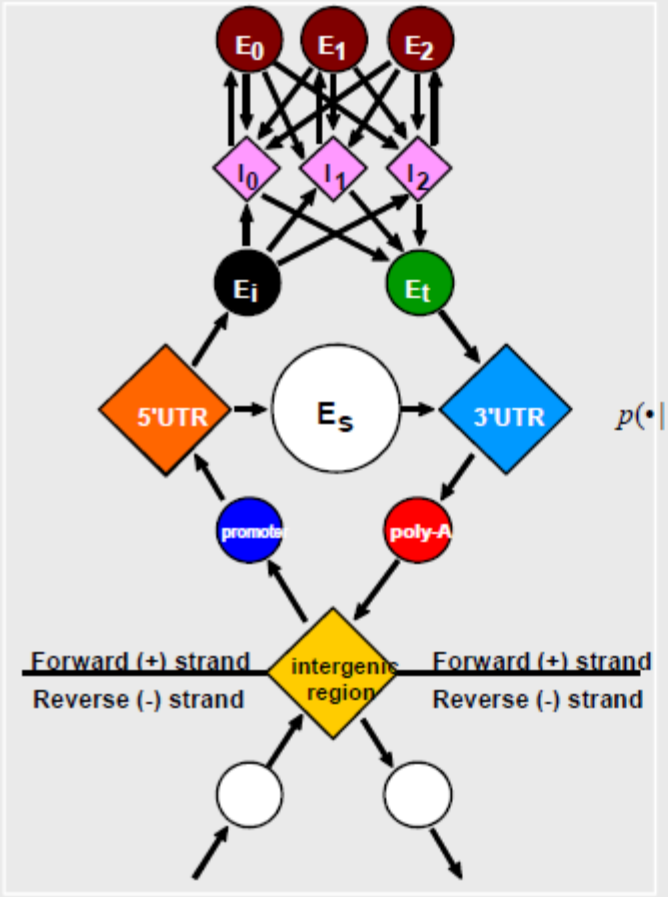
Applications of HMM

- **Some early applications of HMMs**
  - finance
  - 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

A Typical Gene Structure

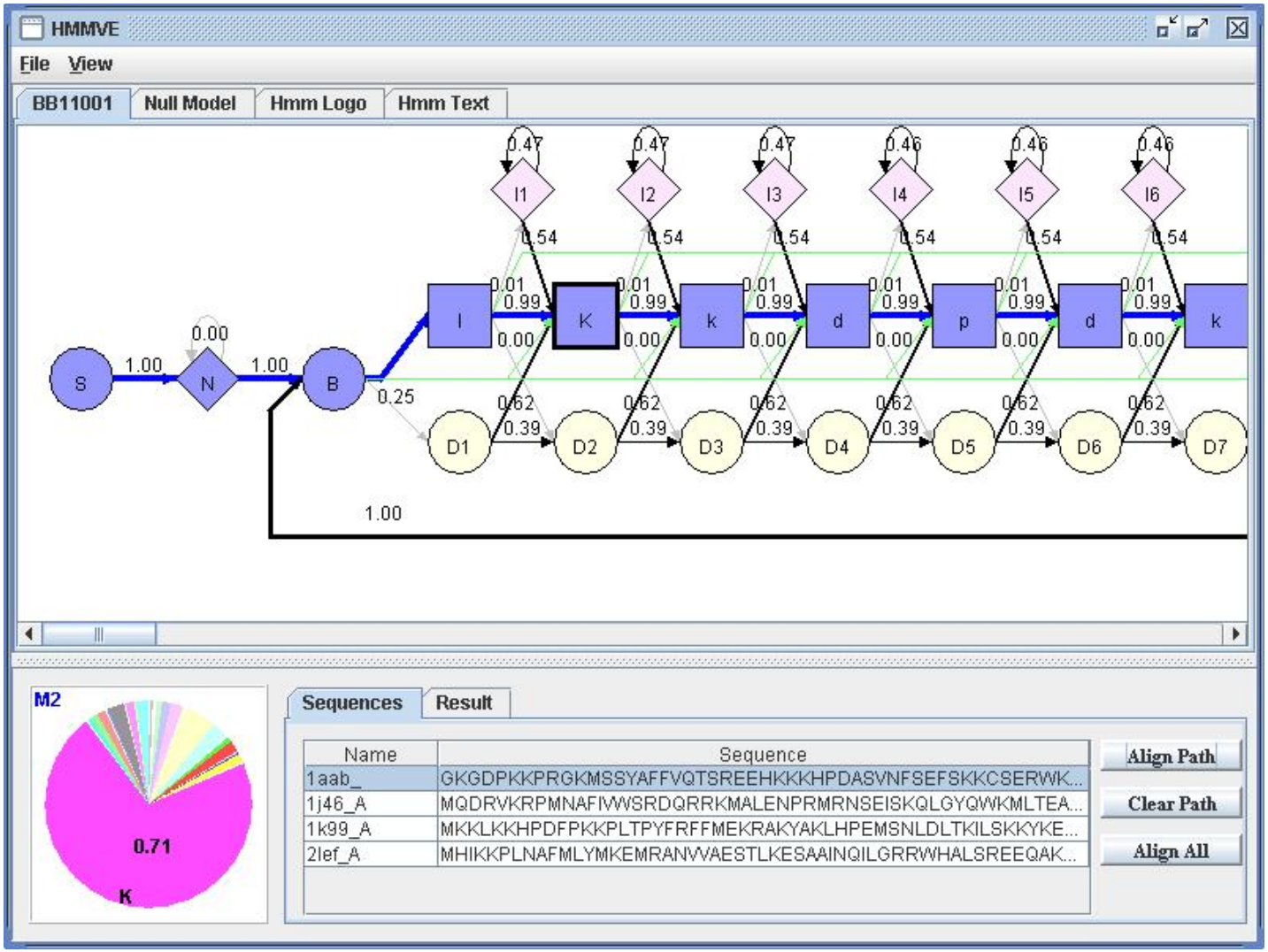


GeneScan (Burge and Karlin)



$$p(\bullet \mid y) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}$$

HMM Demo (HHMVE)



J. Dai, J. Cheng. HMMEditor: A Visual Editing Tool for Profile Hidden Makov Model. BMC Genomics. 2008