



Hidden Markov Models

COSI 114 – Computational Linguistics
James Pustejovsky

February 10, 2015
Brandeis University

Slides thanks to David Blei

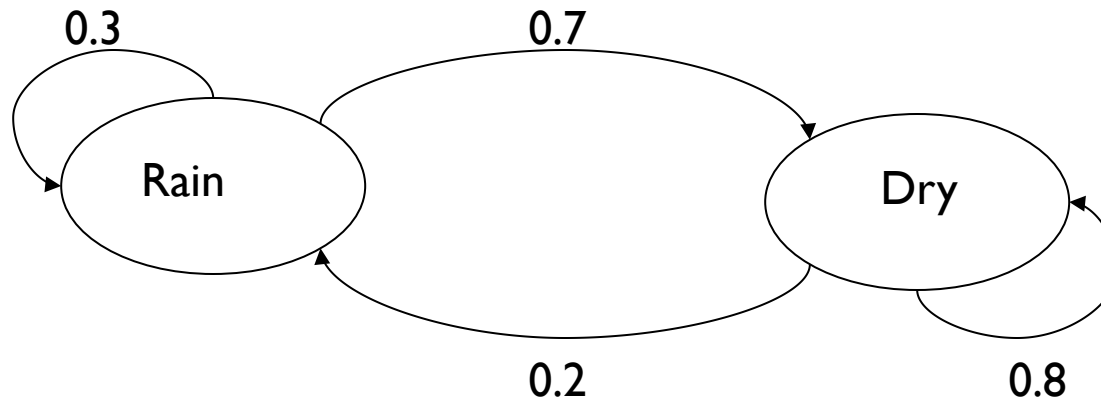
Markov Models

- Set of states: $\{s_1, s_2, \dots, s_N\}$
- Process moves from one state to another generating a sequence of states : $s_{i1}, s_{i2}, \dots, s_{ik}, \dots$
- Markov chain property: probability of each subsequent state depends only on what was the previous state:

$$P(s_{ik} \mid s_{i1}, s_{i2}, \dots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$

- To define Markov model, the following probabilities have to be specified: transition probabilities $a_{ij} = P(s_i \mid s_j)$ and initial probabilities $\pi_i = P(s_i)$

Example of Markov Model



- Two states : 'Rain' and 'Dry' .
- Transition probabilities: $P(\text{'Rain' } | \text{'Rain'}) = 0.3$, $P(\text{'Dry' } | \text{'Rain'}) = 0.7$,
 $P(\text{'Rain' } | \text{'Dry'}) = 0.2$, $P(\text{'Dry' } | \text{'Dry'}) = 0.8$
- Initial probabilities: say $P(\text{'Rain'}) = 0.4$, $P(\text{'Dry'}) = 0.6$.

Calculation of sequence probability

- By Markov chain property, probability of state sequence can be found by the formula:

$$\begin{aligned}P(s_{i1}, s_{i2}, \dots, s_{ik}) &= P(s_{ik} \mid s_{i1}, s_{i2}, \dots, s_{ik-1}) P(s_{i1}, s_{i2}, \dots, s_{ik-1}) \\&= P(s_{ik} \mid s_{ik-1}) P(s_{i1}, s_{i2}, \dots, s_{ik-1}) = \dots \\&= P(s_{ik} \mid s_{ik-1}) P(s_{ik-1} \mid s_{ik-2}) \dots P(s_{i2} \mid s_{i1}) P(s_{i1})\end{aligned}$$

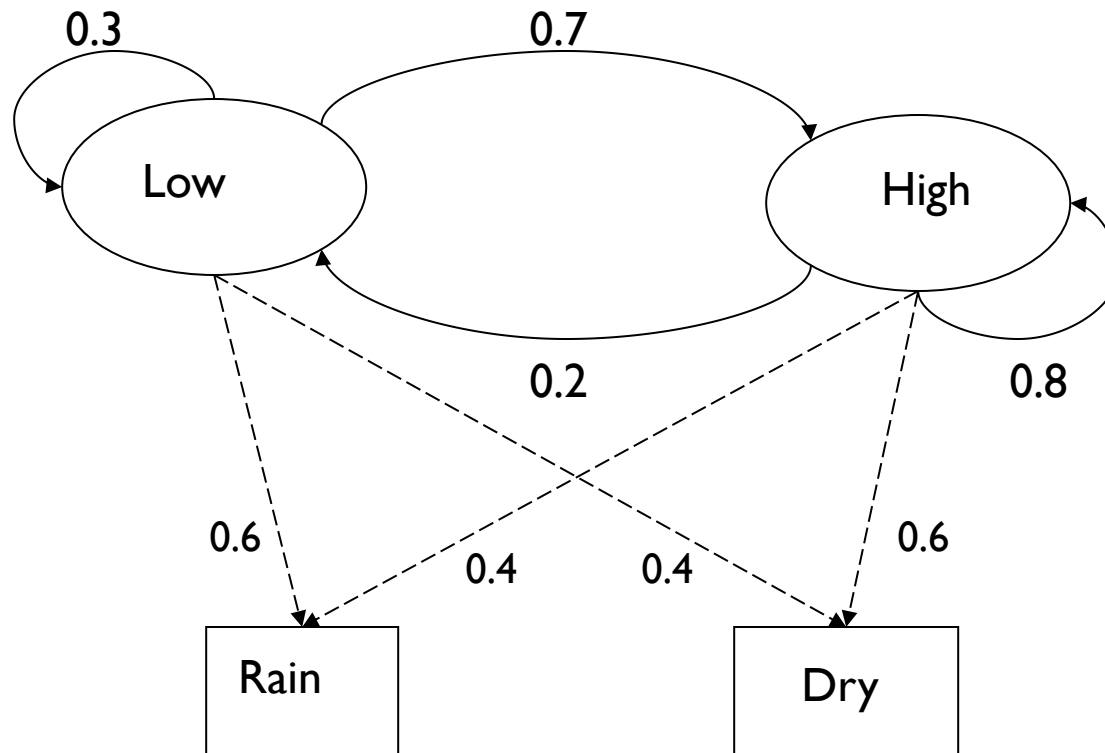
- Suppose we want to calculate a probability of a sequence of states in our example, { 'Dry' , ' Dry' , ' Rain' ,Rain' }.

$$\begin{aligned}&P(\{ \text{'Dry' , ' Dry' , ' Rain' ,Rain' } \}) = \\&P(\text{'Rain' } \mid \text{' Rain' }) P(\text{'Rain' } \mid \text{' Dry' }) P(\text{'Dry' } \mid \text{' Dry' }) \\&P(\text{'Dry' }) = \\&= 0.3 * 0.2 * 0.8 * 0.6\end{aligned}$$

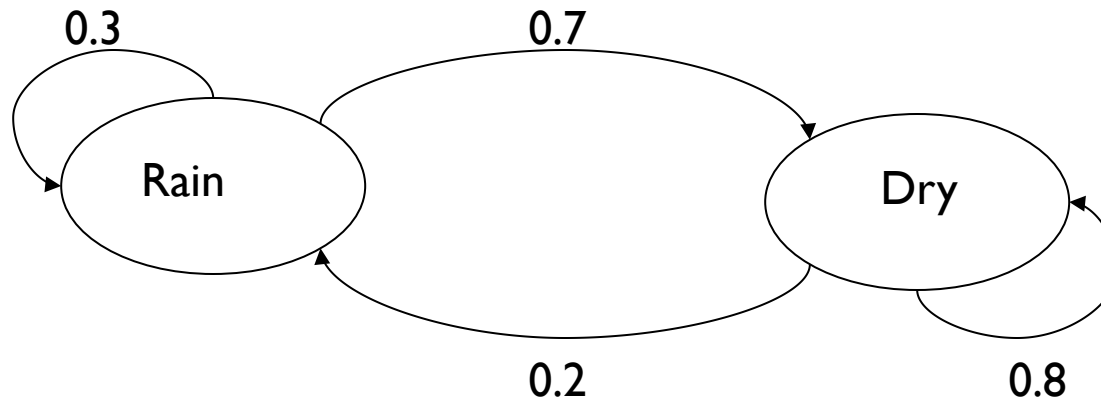
Hidden Markov models

- Set of states: $\{s_1, s_2, \dots, s_N\}$
- Process moves from one state to another generating a sequence of states :
 $s_{i1}, s_{i2}, \dots, s_{ik}, \dots$
- Markov chain property: probability of each subsequent state depends only on what was the previous state:
$$P(s_{ik} \mid s_{i1}, s_{i2}, \dots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$
- States are not visible, but each state randomly generates one of M observations (or visible states) $\{v_1, v_2, \dots, v_M\}$
- To define hidden Markov model, the following probabilities have to be specified: matrix of transition probabilities $A=(a_{ij})$, $a_{ij} = P(s_i \mid s_j)$, matrix of observation probabilities $B=(b_i(v_m))$, $b_i(v_m) = P(v_m \mid s_i)$ and a vector of initial probabilities $\pi=(\pi_i)$, $\pi_i = P(s_i)$. Model is represented by $M=(A, B, \pi)$.

Example of Hidden Markov Model



Example of Markov Model



- Two states : 'Rain' and 'Dry' .
- Transition probabilities: $P(\text{'Rain' } | \text{'Rain'}) = 0.3$, $P(\text{'Dry' } | \text{'Rain'}) = 0.7$,
 $P(\text{'Rain' } | \text{'Dry'}) = 0.2$, $P(\text{'Dry' } | \text{'Dry'}) = 0.8$
- Initial probabilities: say $P(\text{'Rain'}) = 0.4$, $P(\text{'Dry'}) = 0.6$.

Calculation of sequence probability

- By Markov chain property, probability of state sequence can be found by the formula:

$$\begin{aligned}P(s_{i1}, s_{i2}, \dots, s_{ik}) &= P(s_{ik} \mid s_{i1}, s_{i2}, \dots, s_{ik-1}) P(s_{i1}, s_{i2}, \dots, s_{ik-1}) \\&= P(s_{ik} \mid s_{ik-1}) P(s_{i1}, s_{i2}, \dots, s_{ik-1}) = \dots \\&= P(s_{ik} \mid s_{ik-1}) P(s_{ik-1} \mid s_{ik-2}) \dots P(s_{i2} \mid s_{i1}) P(s_{i1})\end{aligned}$$

- Suppose we want to calculate a probability of a sequence of states in our example, { 'Dry' , ' Dry' , ' Rain' ,Rain' }.

$$\begin{aligned}&P(\{ \text{'Dry' , ' Dry' , ' Rain' ,Rain' } \}) = \\&P(\text{'Rain' } \mid \text{' Rain' }) P(\text{'Rain' } \mid \text{' Dry' }) P(\text{'Dry' } \mid \text{' Dry' }) \\&P(\text{'Dry' }) = \\&= 0.3 * 0.2 * 0.8 * 0.6\end{aligned}$$

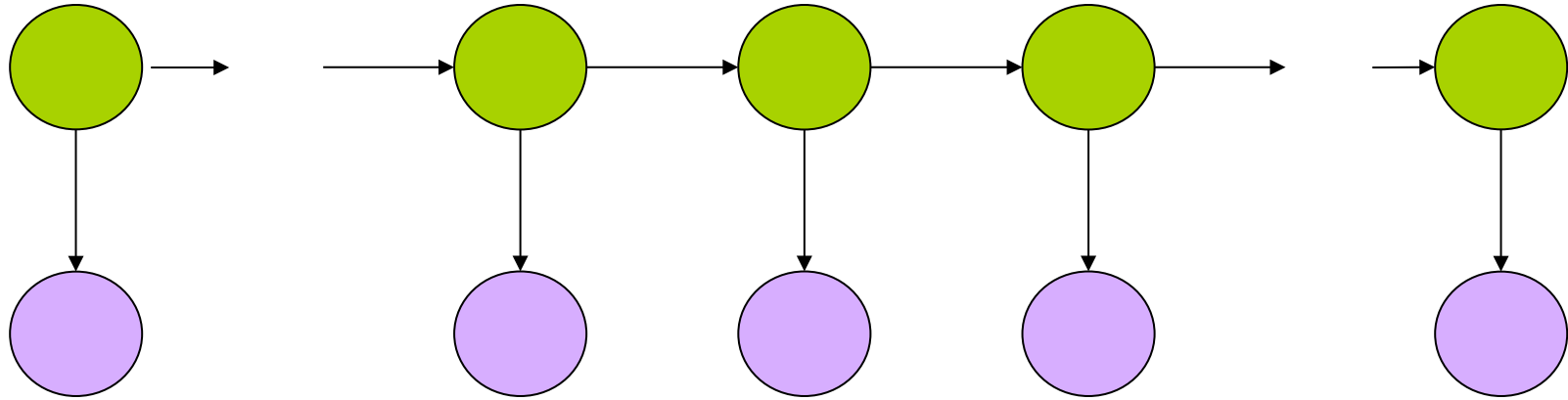
Hidden Markov models

- Set of states: $\{s_1, s_2, \dots, s_N\}$
- Process moves from one state to another generating a sequence of states :
 $s_{i1}, s_{i2}, \dots, s_{ik}, \dots$
- Markov chain property: probability of each subsequent state depends only on what was the previous state:
$$P(s_{ik} \mid s_{i1}, s_{i2}, \dots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$
- States are not visible, but each state randomly generates one of M observations (or visible states) $\{v_1, v_2, \dots, v_M\}$
- To define hidden Markov model, the following probabilities have to be specified: matrix of transition probabilities $A=(a_{ij})$, $a_{ij} = P(s_i \mid s_j)$, matrix of observation probabilities $B=(b_i(v_m))$, $b_i(v_m) = P(v_m \mid s_i)$ and a vector of initial probabilities $\pi=(\pi_i)$, $\pi_i = P(s_i)$. Model is represented by $M=(A, B, \pi)$.

Example of Hidden Markov Model

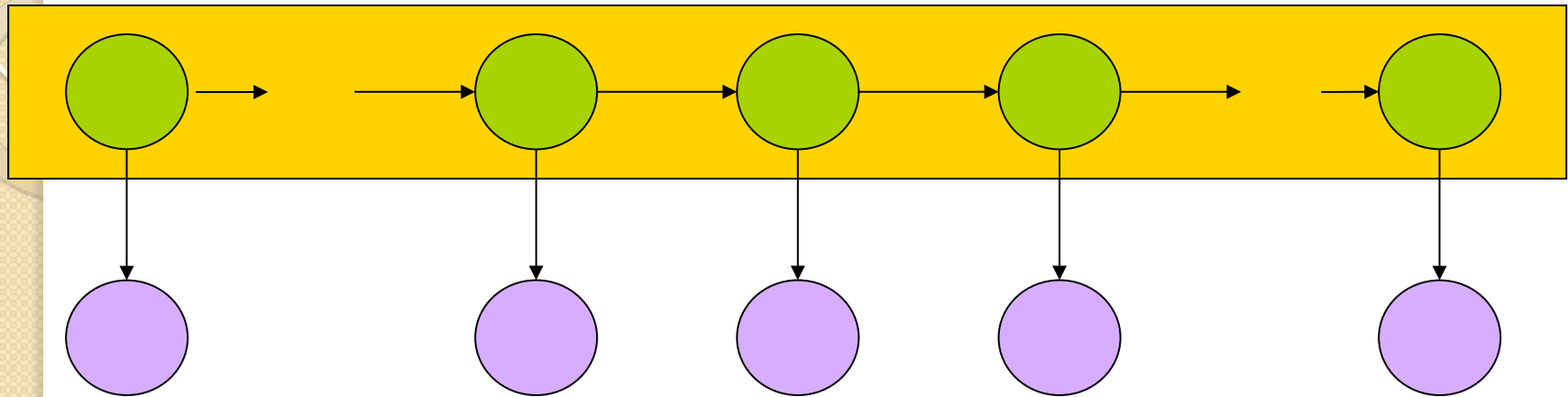
- Two states : 'Low' and 'High' atmospheric pressure.
- Two observations : 'Rain' and 'Dry' .
- Transition probabilities: $P(\text{'Low' | 'Low'})=0.3$, $P(\text{'High' | 'Low'})=0.7$,
 $P(\text{'Low' | 'High'})=0.2$, $P(\text{'High' | 'High'})=0.8$
- Observation probabilities : $P(\text{'Rain' | 'Low'})=0.6$, $P(\text{'Dry' | 'Low'})=0.4$,
 $P(\text{'Rain' | 'High'})=0.4$, $P(\text{'Dry' | 'High'})=0.3$.
- Initial probabilities: say $P(\text{'Low'})=0.4$, $P(\text{'High'})=0.6$.

What is an HMM?



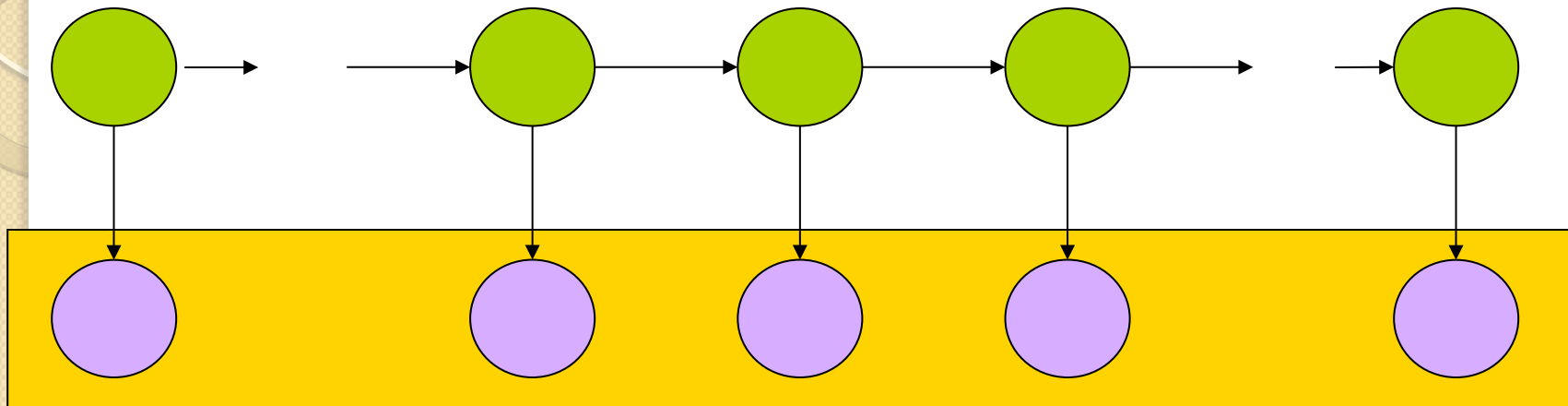
- Graphical Model
- Circles indicate states
- Arrows indicate probabilistic dependencies between states

What is an HMM?



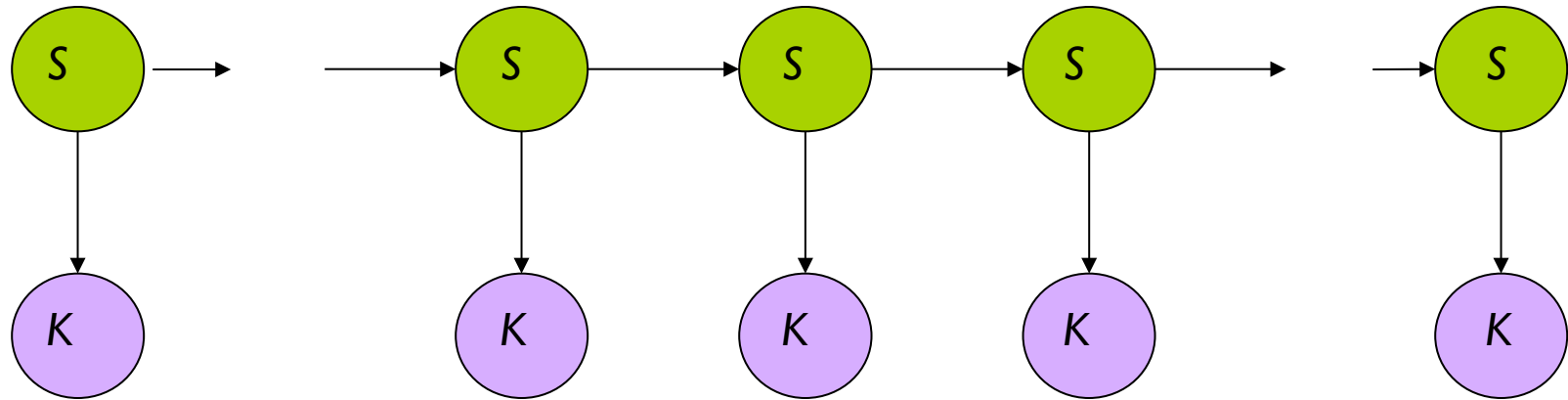
- Green circles are ***hidden states***
- Dependent only on the previous state
- “The past is independent of the future given the present.”

What is an HMM?



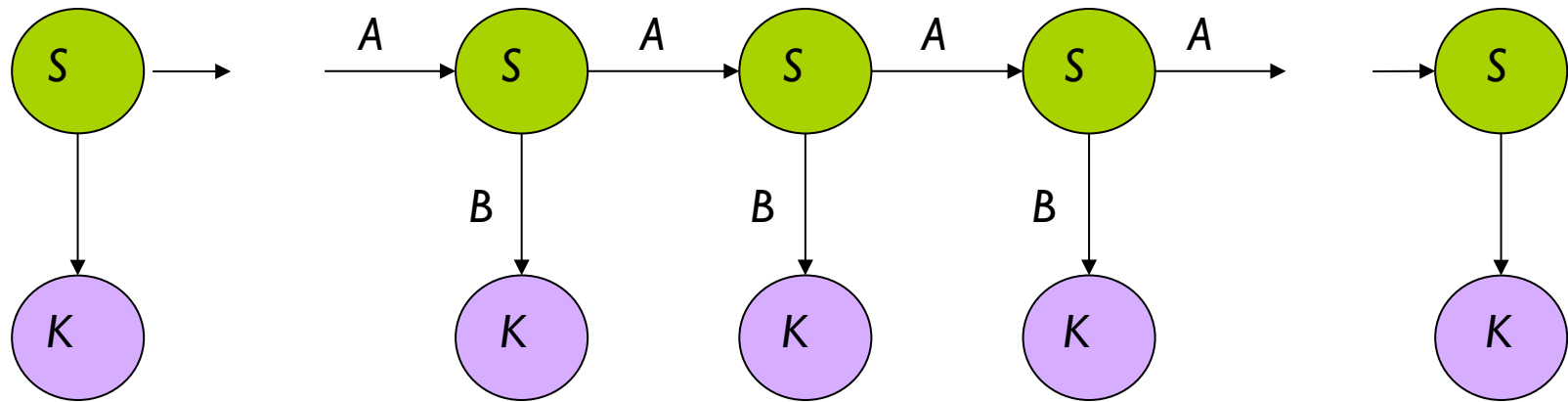
- Purple nodes are ***observed states***
- Dependent only on their corresponding hidden state

HMM Formalism



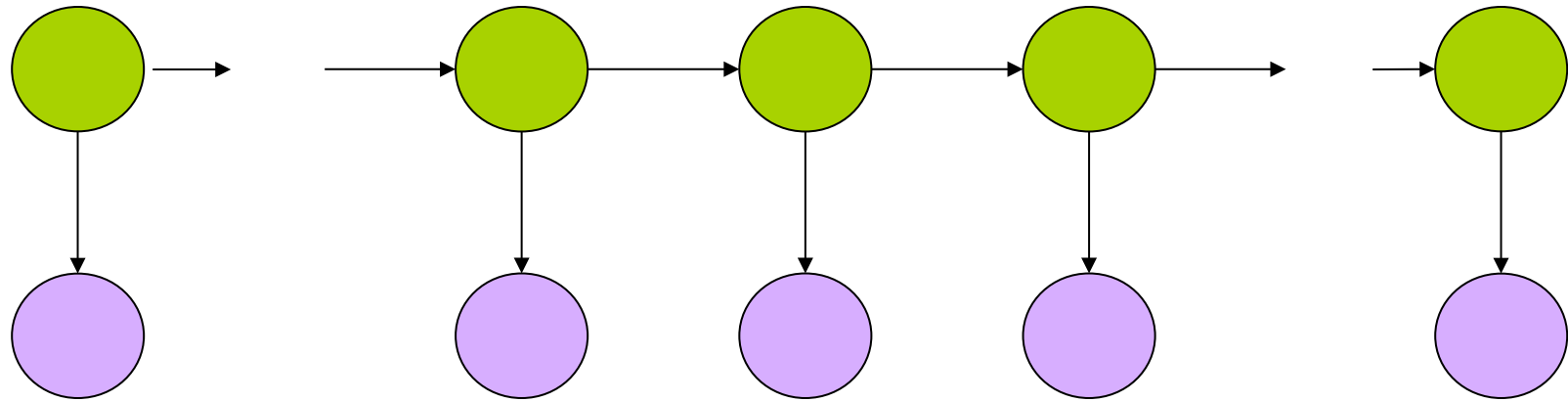
- $\{S, K, \Pi, A, B\}$
- $S : \{s_1 \dots s_N\}$ are the values for the hidden states
- $K : \{k_1 \dots k_M\}$ are the values for the observations

HMM Formalism



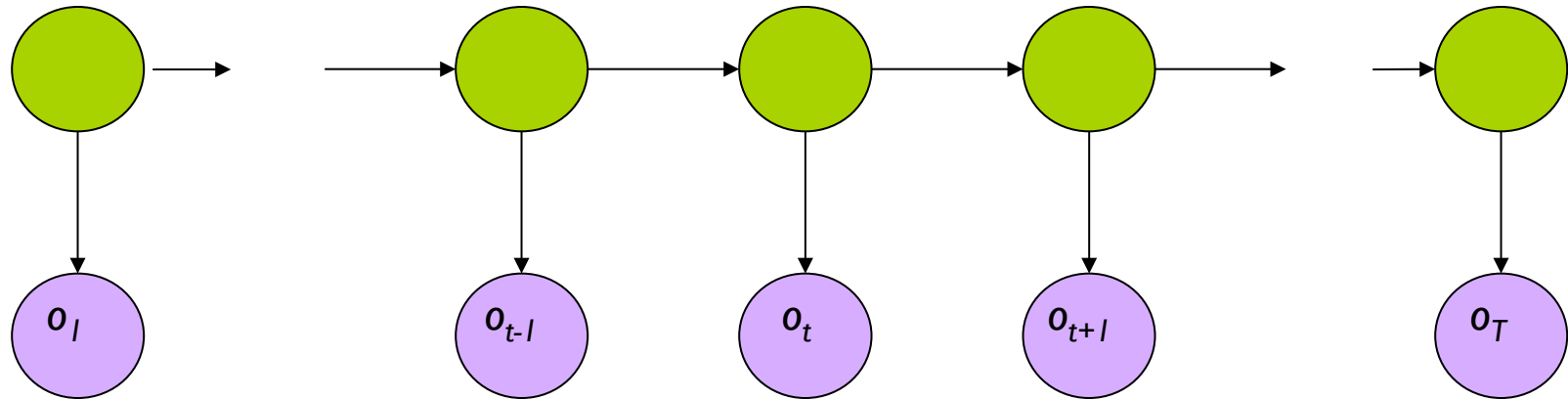
- $\{S, K, \Pi, A, B\}$
- $\Pi = \{\pi_i\}$ are the initial state probabilities
- $A = \{a_{ij}\}$ are the state transition probabilities
- $B = \{b_{ik}\}$ are the observation state probabilities

Inference in an HMM



- Compute the probability of a given observation sequence
- Given an observation sequence, compute the most likely hidden state sequence
- Given an observation sequence and set of possible models, which model most closely fits the data?

Decoding

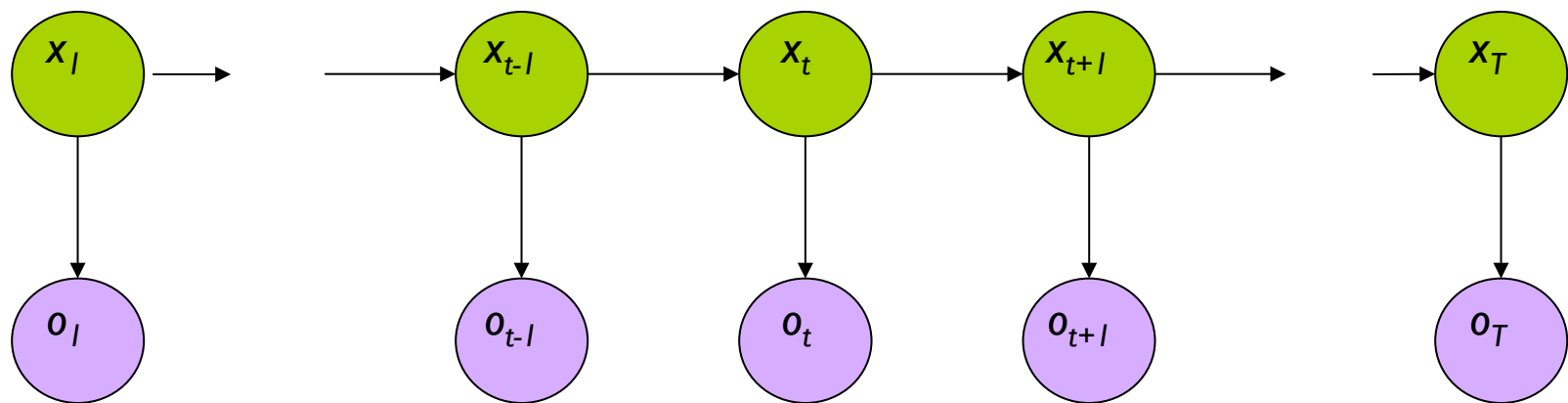


Given an observation sequence and a model,
compute the probability of the observation
sequence

$$O = (o_1 \dots o_T), \mu = (A, B, \Pi)$$

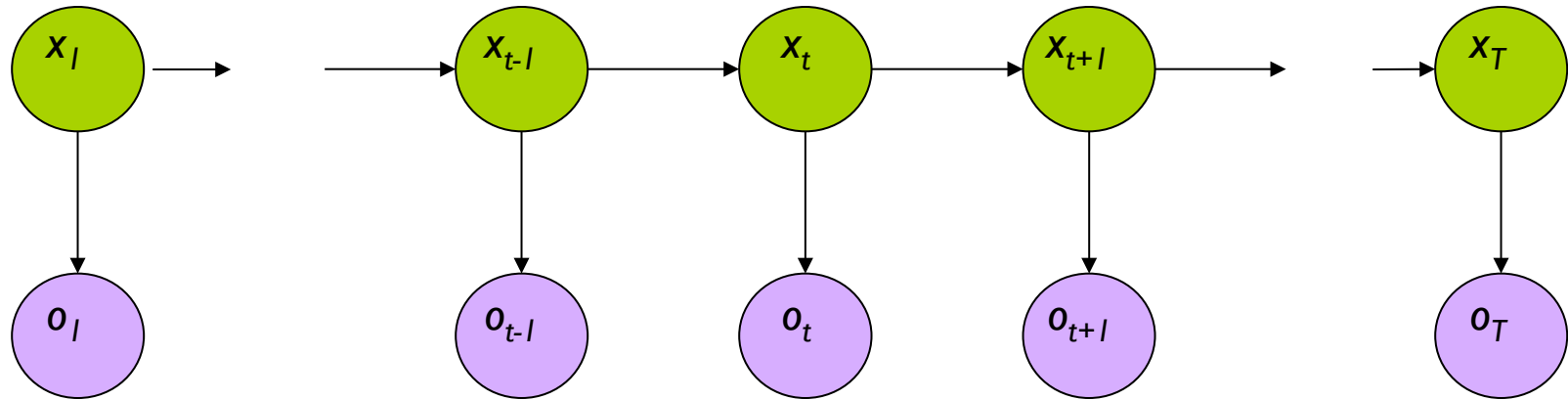
Compute $P(O \mid \mu)$

Decoding



$$P(O | X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \dots b_{x_T o_T}$$

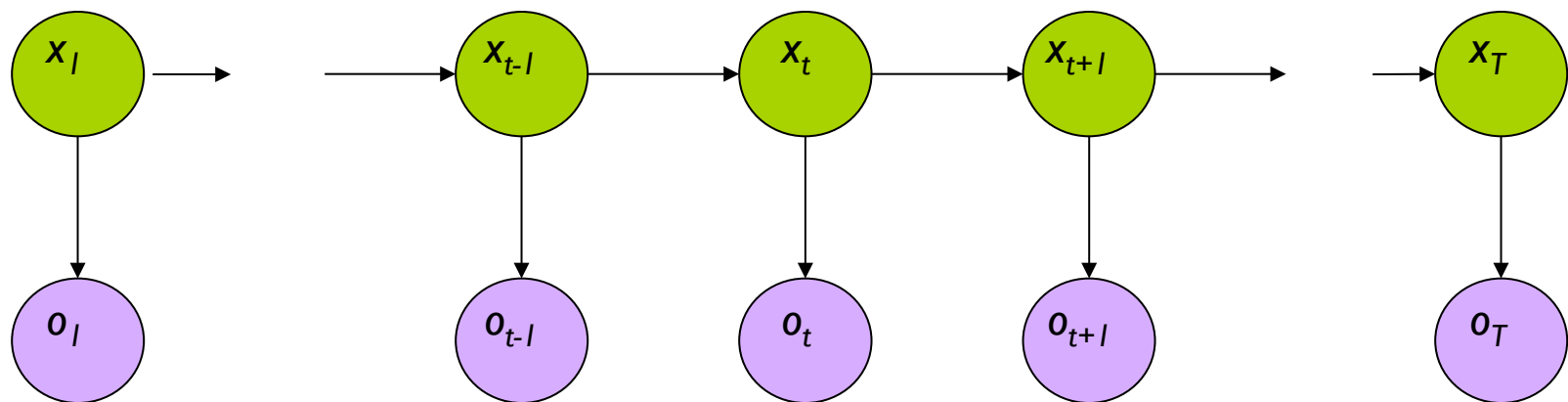
Decoding



$$P(O | X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \dots b_{x_T o_T}$$

$$P(X | \mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \dots a_{x_{T-1} x_T}$$

Decoding

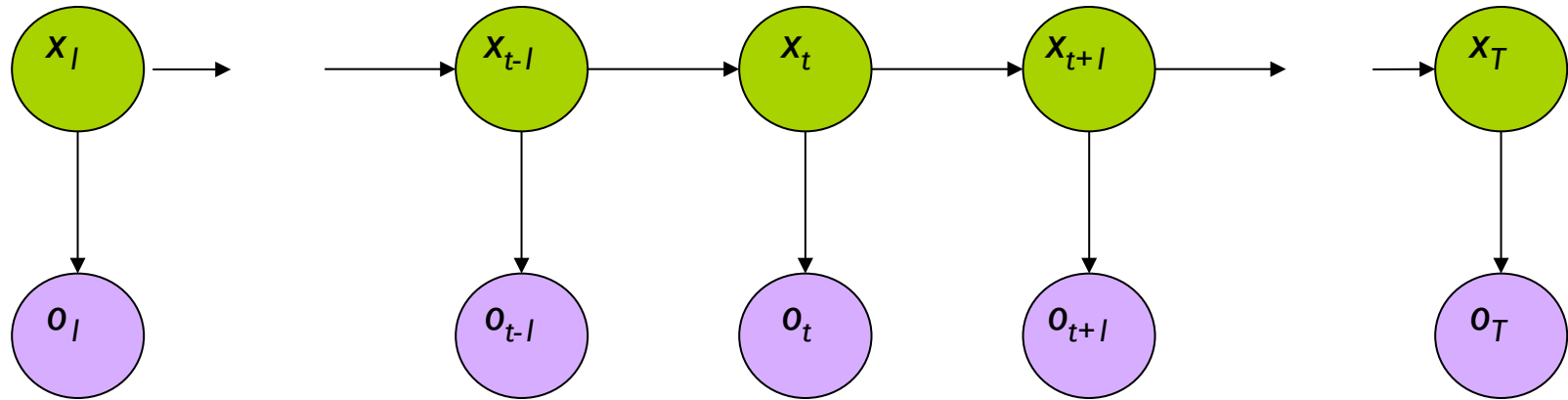


$$P(O | X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \dots b_{x_T o_T}$$

$$P(X | \mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \dots a_{x_{T-1} x_T}$$

$$P(O, X | \mu) = P(O | X, \mu) P(X | \mu)$$

Decoding



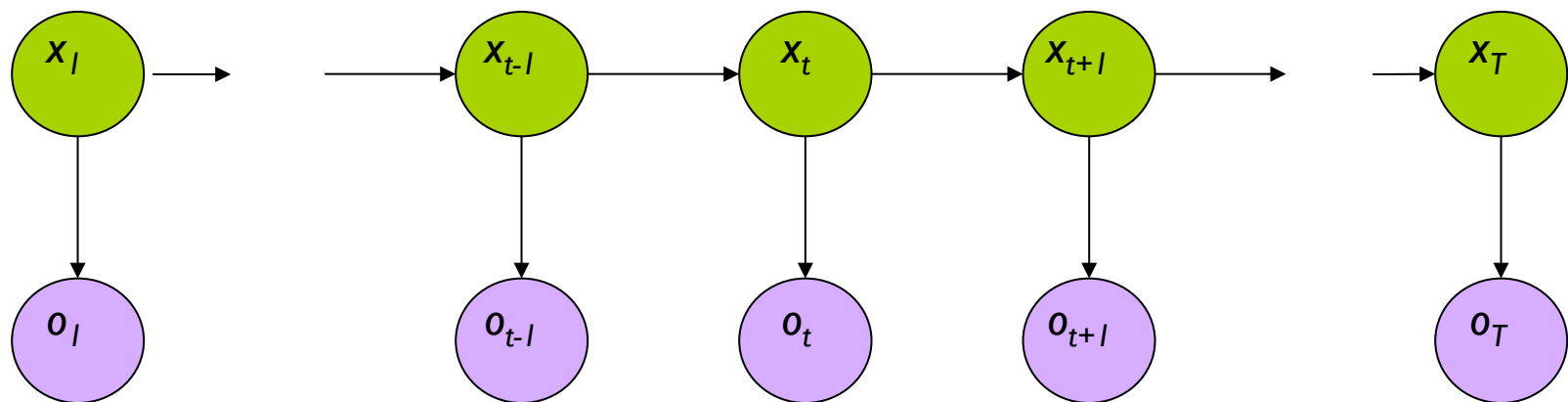
$$P(O | X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \dots b_{x_T o_T}$$

$$P(X | \mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \dots a_{x_{T-1} x_T}$$

$$P(O, X | \mu) = P(O | X, \mu) P(X | \mu)$$

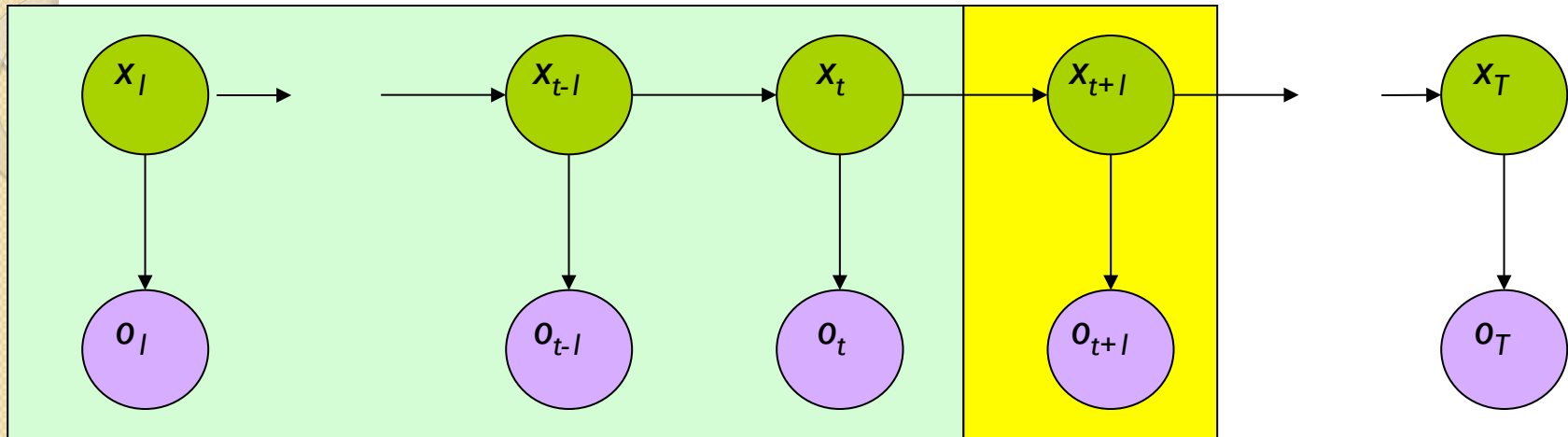
$$P(O | \mu) = \sum_X P(O | X, \mu) P(X | \mu)$$

Decoding



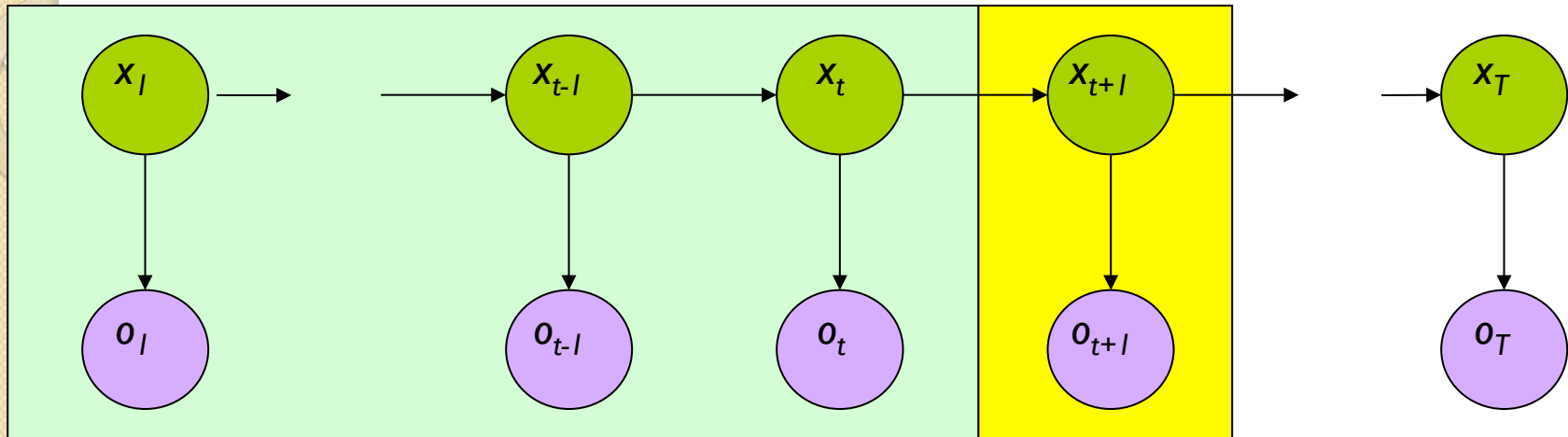
$$P(O \mid \mu) = \sum_{\{x_1 \dots x_T\}} \pi_{x_1} b_{x_1 o_1} \prod_{t=1}^{T-1} a_{x_t x_{t+1}} b_{x_{t+1} o_{t+1}}$$

Forward Procedure



- Special structure gives us an efficient solution using *dynamic programming*.
- **Intuition:** Probability of the first t observations is the same for all possible $t+1$ length state sequences.
- **Define:** $\alpha_i(t) = P(o_1 \dots o_t, x_t = i \mid \mu)$

Forward Procedure



$$\alpha_j(t+1)$$

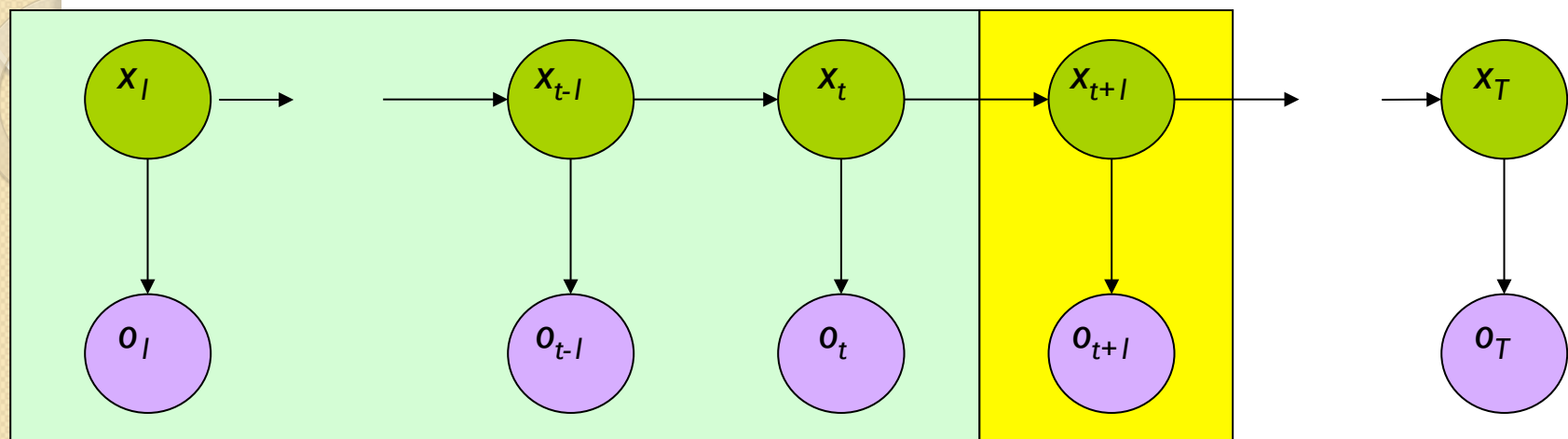
$$= P(o_1 \dots o_{t+1}, x_{t+1} = j)$$

$$= P(o_1 \dots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \dots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \dots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

Forward Procedure



$$\alpha_j(t+1)$$

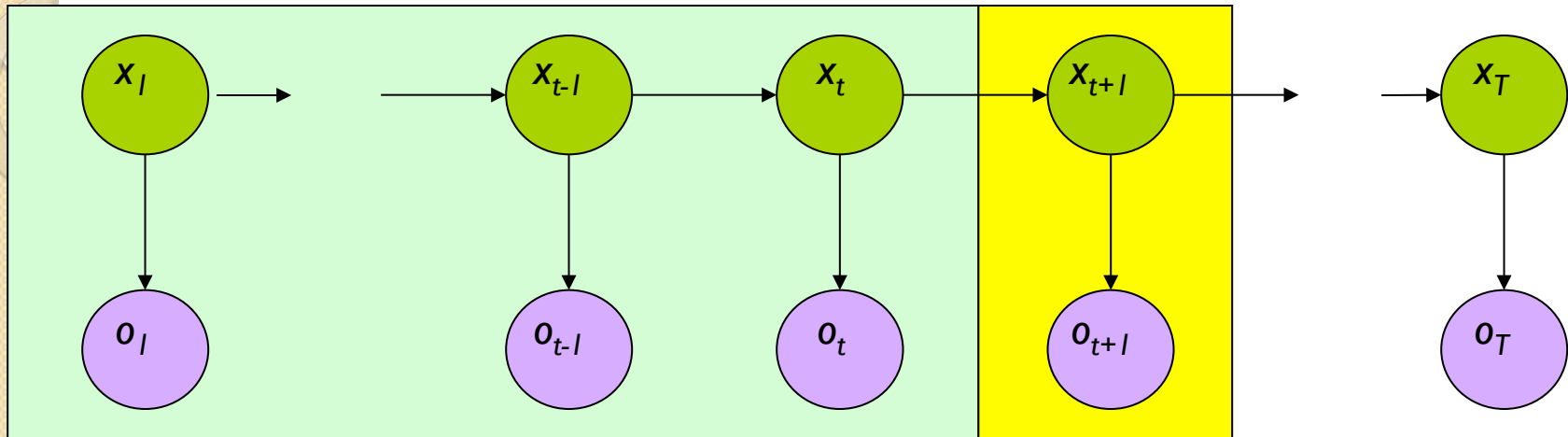
$$= P(o_1 \dots o_{t+1}, x_{t+1} = j)$$

$$= P(o_1 \dots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \dots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \dots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

Forward Procedure



$$\alpha_j(t+1)$$

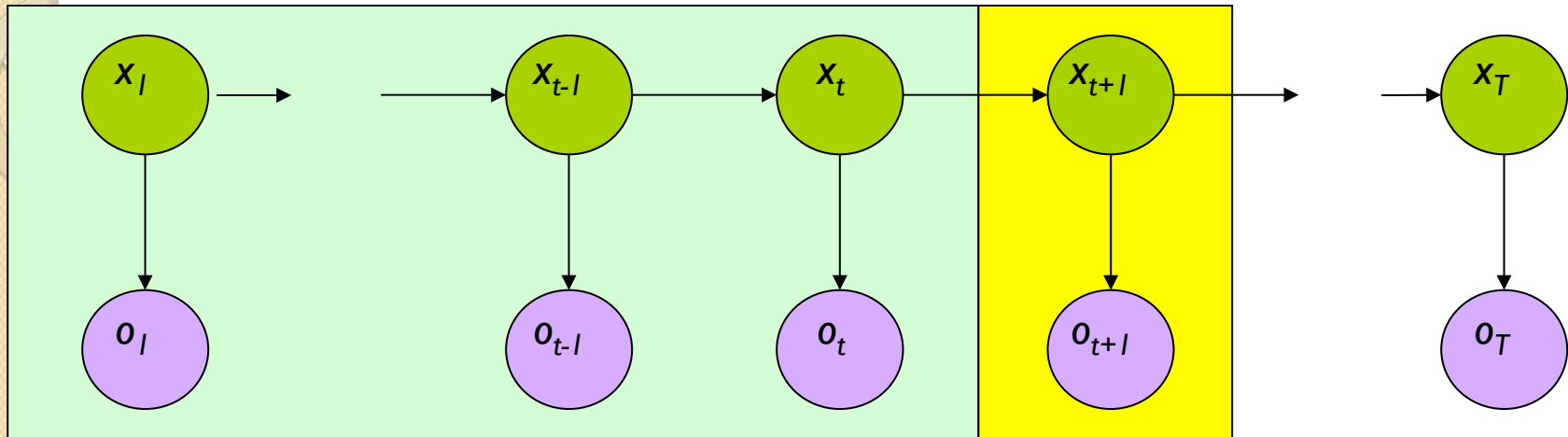
$$= P(o_1 \dots o_{t+1}, x_{t+1} = j)$$

$$= P(o_1 \dots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \dots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \dots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

Forward Procedure



$$\alpha_j(t+1)$$

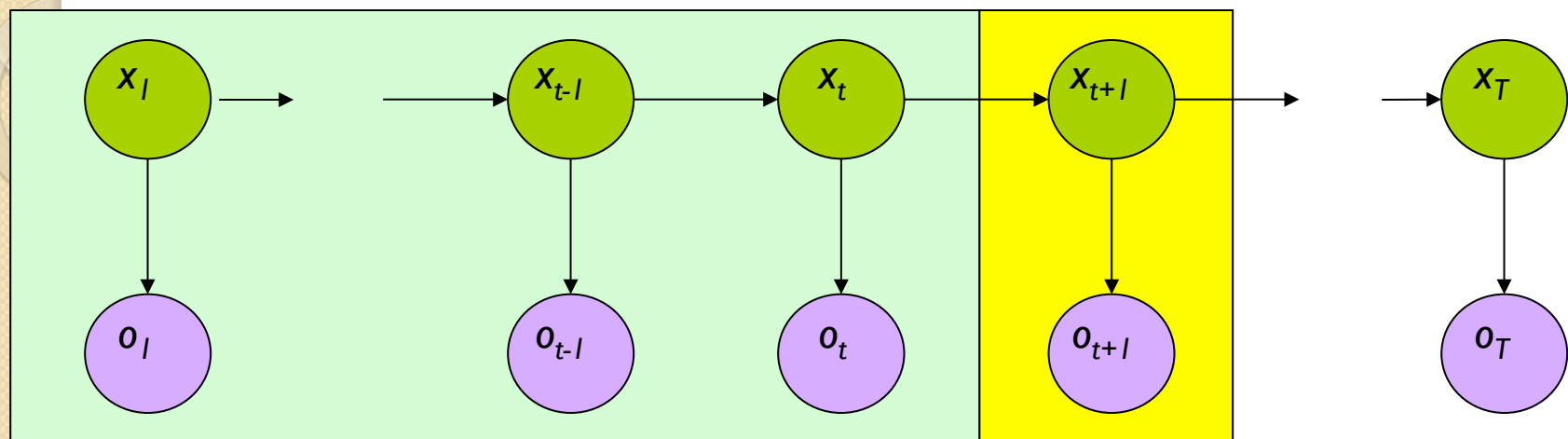
$$= P(o_1 \dots o_{t+1}, x_{t+1} = j)$$

$$= P(o_1 \dots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \dots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \dots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

Forward Procedure



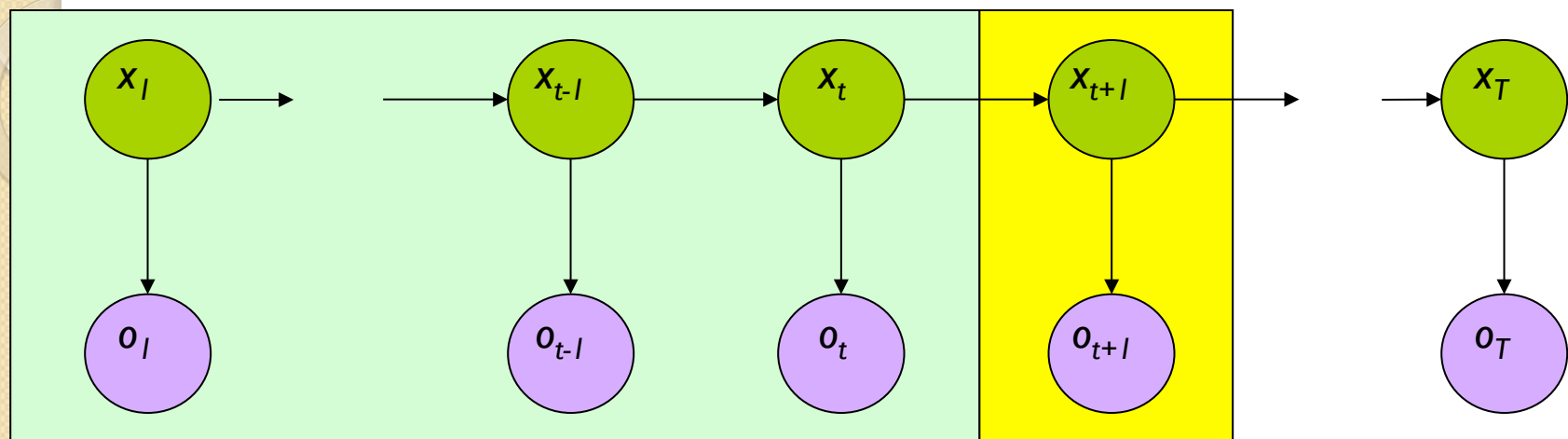
$$= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_t = i, x_{t+1} = j) P(o_{t+1} | x_{t+1} = j)$$

$$= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_{t+1} = j | x_t = i) P(x_t = i) P(o_{t+1} | x_{t+1} = j)$$

$$= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_t = i) P(x_{t+1} = j | x_t = i) P(o_{t+1} | x_{t+1} = j)$$

$$= \sum_{i=1 \dots N} \alpha_i(t) a_{ij} b_{jo_{t+1}}$$

Forward Procedure



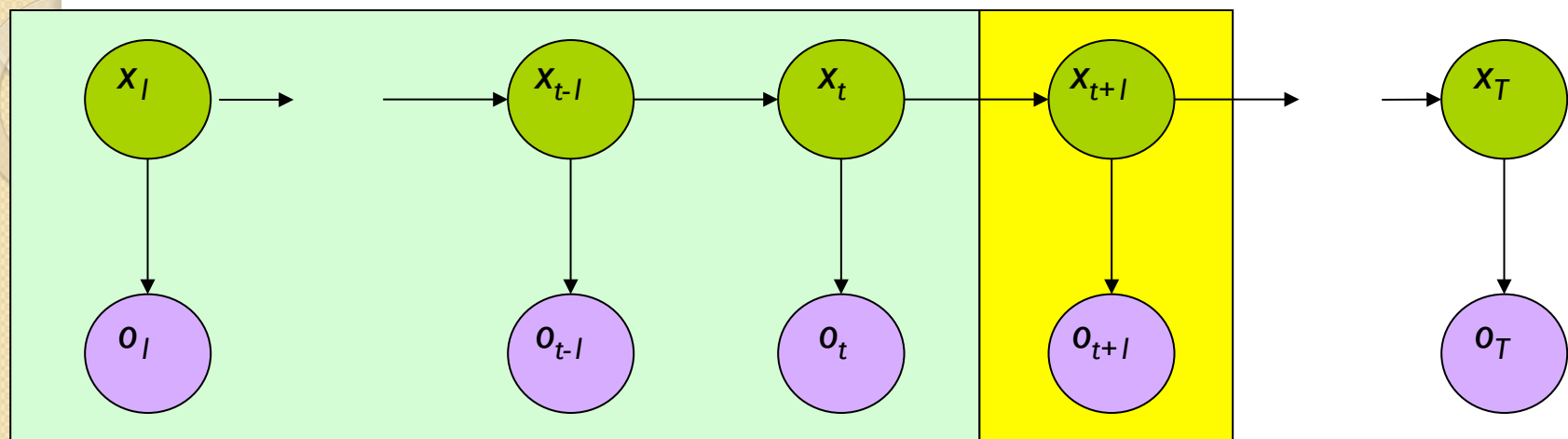
$$= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_t = i, x_{t+1} = j) P(o_{t+1} | x_{t+1} = j)$$

$$= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_{t+1} = j | x_t = i) P(x_t = i) P(o_{t+1} | x_{t+1} = j)$$

$$= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_t = i) P(x_{t+1} = j | x_t = i) P(o_{t+1} | x_{t+1} = j)$$

$$= \sum_{i=1 \dots N} \alpha_i(t) a_{ij} b_{jo_{t+1}}$$

Forward Procedure



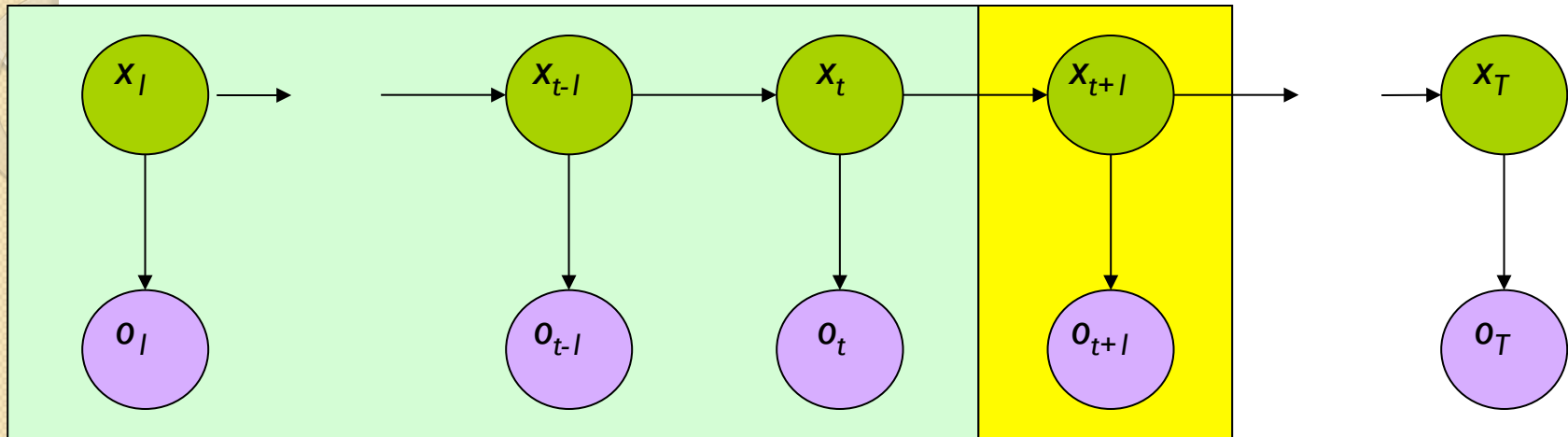
$$= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_t = i, x_{t+1} = j) P(o_{t+1} | x_{t+1} = j)$$

$$= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_{t+1} = j | x_t = i) P(x_t = i) P(o_{t+1} | x_{t+1} = j)$$

$$= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_t = i) P(x_{t+1} = j | x_t = i) P(o_{t+1} | x_{t+1} = j)$$

$$= \sum_{i=1 \dots N} \alpha_i(t) a_{ij} b_{jo_{t+1}}$$

Forward Procedure



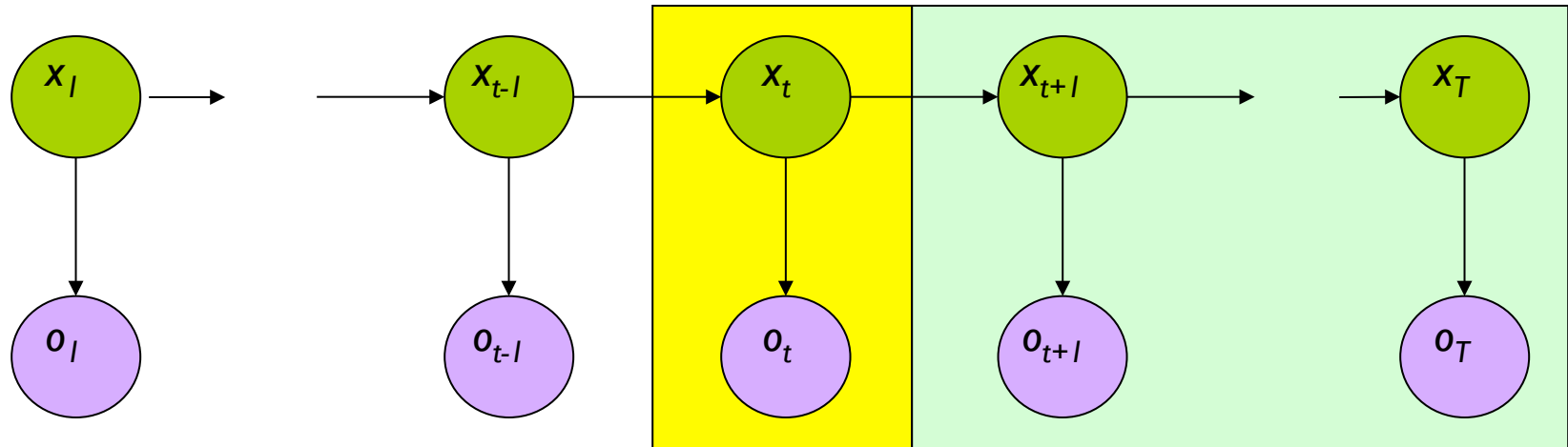
$$= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_t = i, x_{t+1} = j) P(o_{t+1} | x_{t+1} = j)$$

$$= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_{t+1} = j | x_t = i) P(x_t = i) P(o_{t+1} | x_{t+1} = j)$$

$$= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_t = i) P(x_{t+1} = j | x_t = i) P(o_{t+1} | x_{t+1} = j)$$

$$= \sum_{i=1 \dots N} \alpha_i(t) a_{ij} b_{jo_{t+1}}$$

Backward Procedure



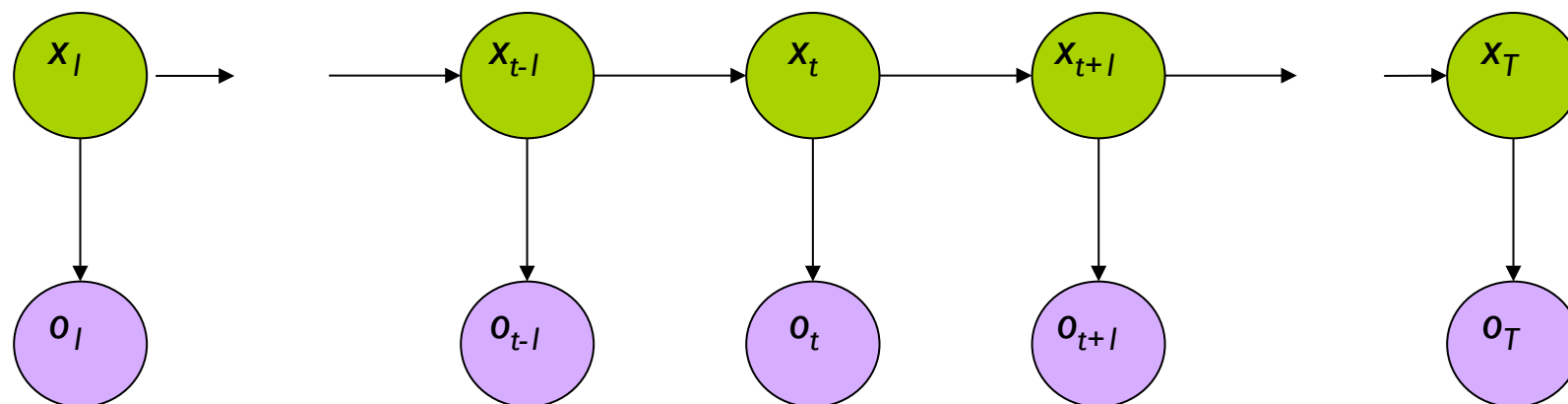
$$\beta_i(T+1) = 1$$

$$\beta_i(t) = P(o_t \dots o_T \mid x_t = i)$$

$$\beta_i(t) = \sum_{j=1 \dots N} a_{ij} b_{io_t} \beta_j(t+1)$$

Probability of the rest
of the states given the
first state

Decoding Solution



$$P(O | \mu) = \sum_{i=1}^N \alpha_i(T)$$

Forward Procedure

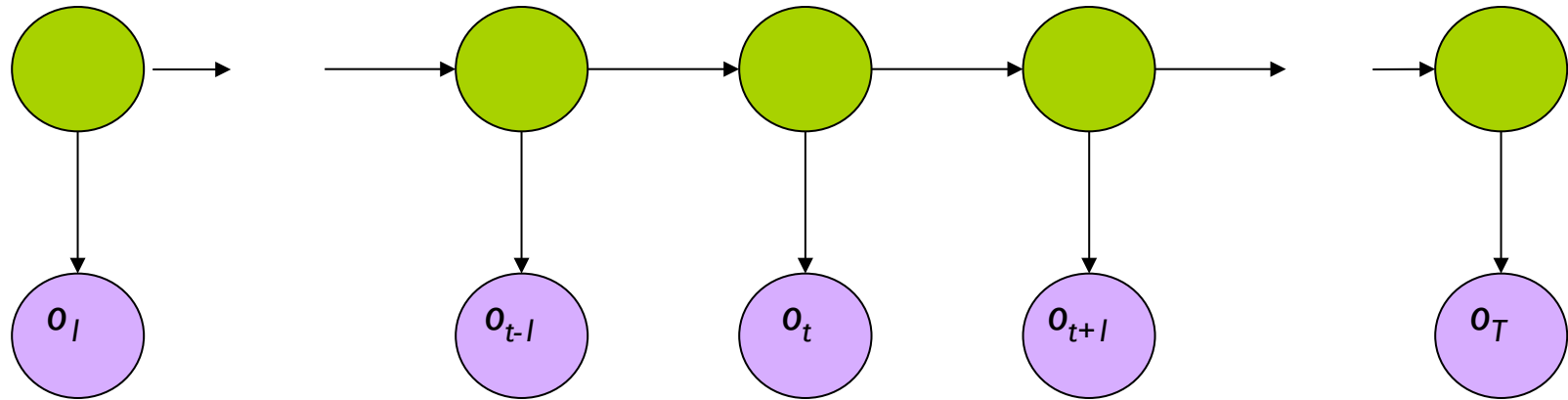
$$P(O | \mu) = \sum_{i=1}^N \pi_i \beta_i(1)$$

Backward Procedure

$$P(O | \mu) = \sum_{i=1}^N \alpha_i(t) \beta_i(t)$$

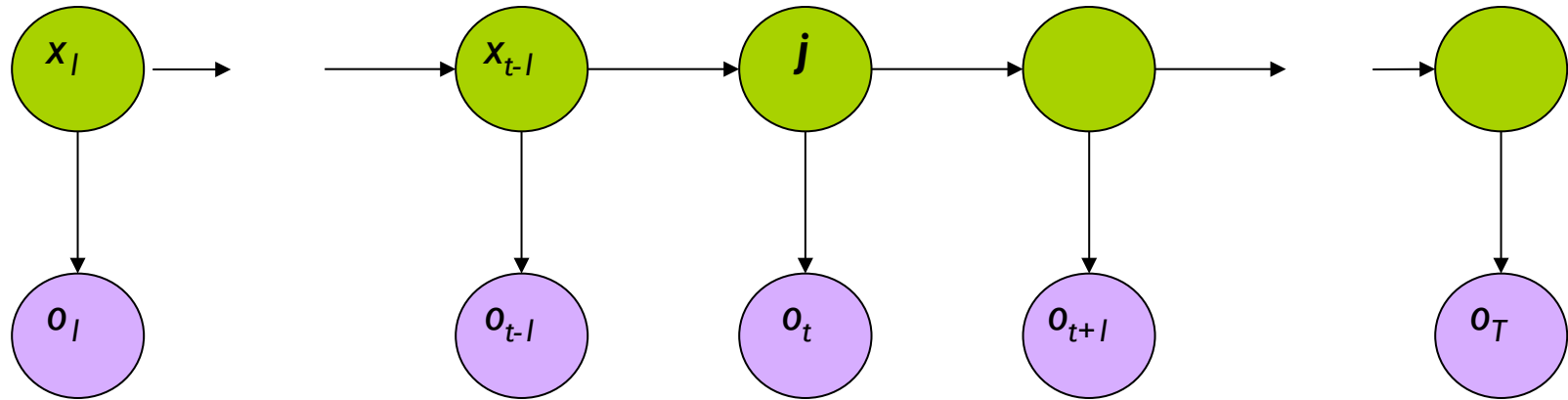
Combination

Best State Sequence



- Find the state sequence that best explains the observations
- **Viterbi** algorithm
- $\arg \max_X P(X | O)$

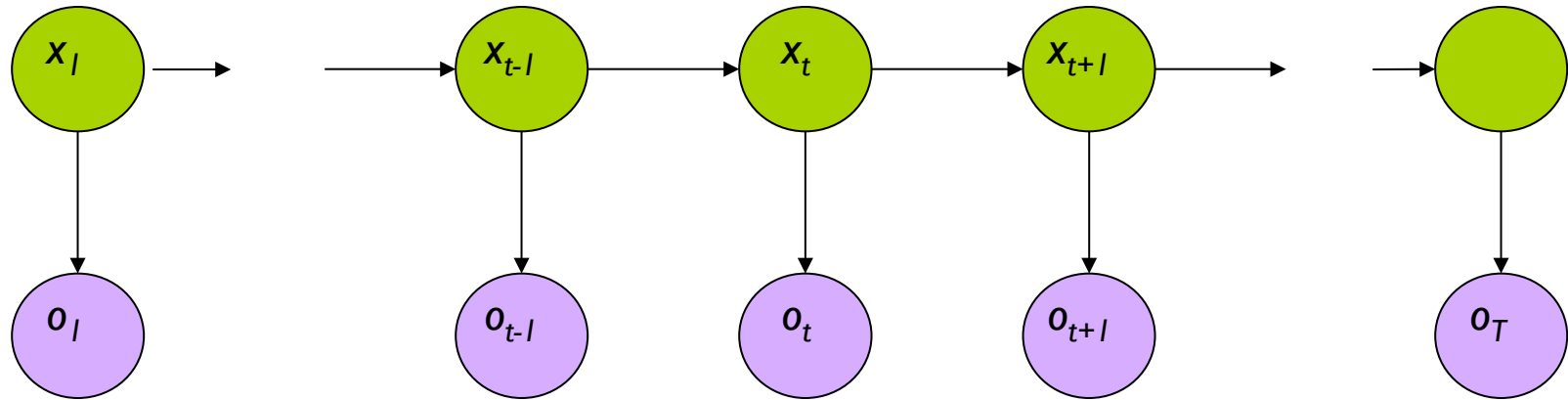
Viterbi Algorithm



$$\delta_j(t) = \max_{x_1 \dots x_{t-1}} P(x_1 \dots x_{t-1}, o_1 \dots o_{t-1}, x_t = j, o_t)$$

The state sequence which maximizes the probability of seeing the observations to time $t-1$, landing in state j , and seeing the observation at time t

Viterbi Algorithm



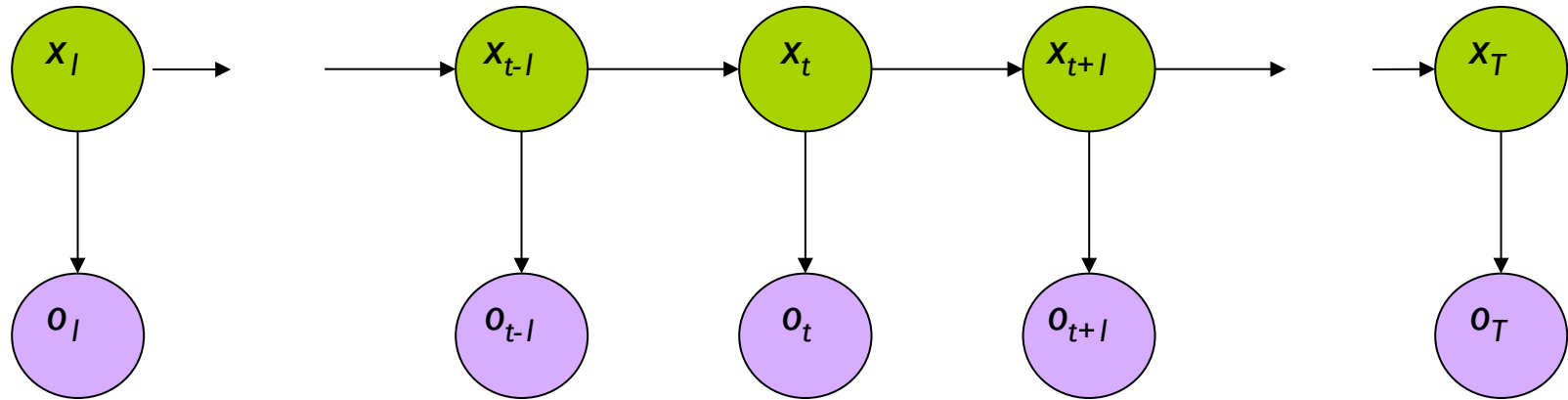
$$\delta_j(t) = \max_{x_1 \dots x_{t-1}} P(x_1 \dots x_{t-1}, o_1 \dots o_{t-1}, x_t = j, o_t)$$

$$\delta_j(t+1) = \max_i \delta_i(t) a_{ij} b_{jo_{t+1}}$$

$$\psi_j(t+1) = \arg \max_i \delta_i(t) a_{ij} b_{jo_{t+1}}$$

Recursive
Computation

Viterbi Algorithm



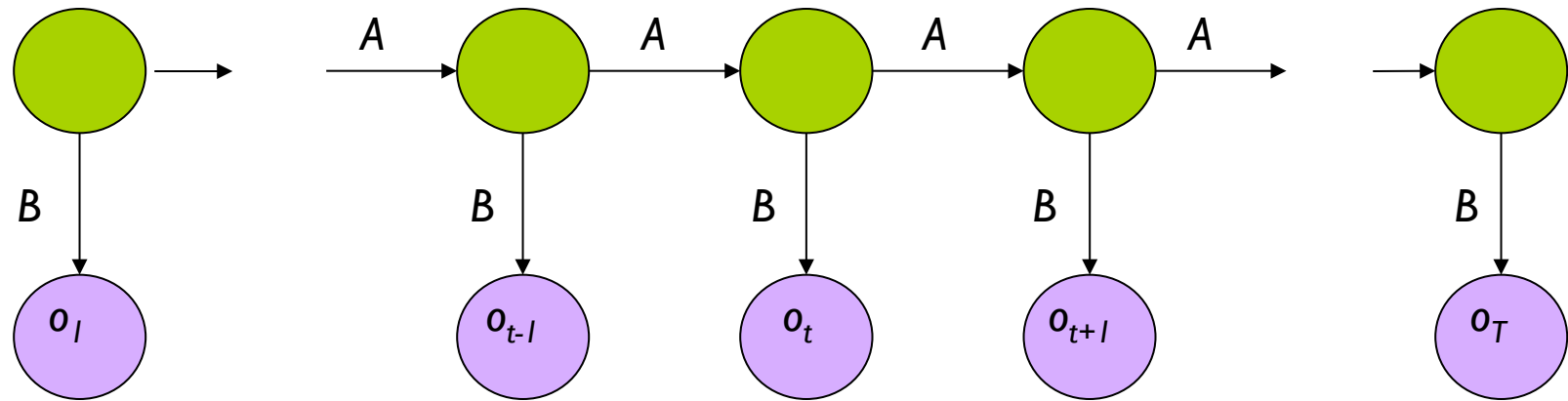
$$\hat{X}_T = \arg \max_i \delta_i(T)$$

$$\hat{X}_t = \psi_{\hat{X}_{t+1}}(t+1)$$

$$P(\hat{X}) = \arg \max_i \delta_i(T)$$

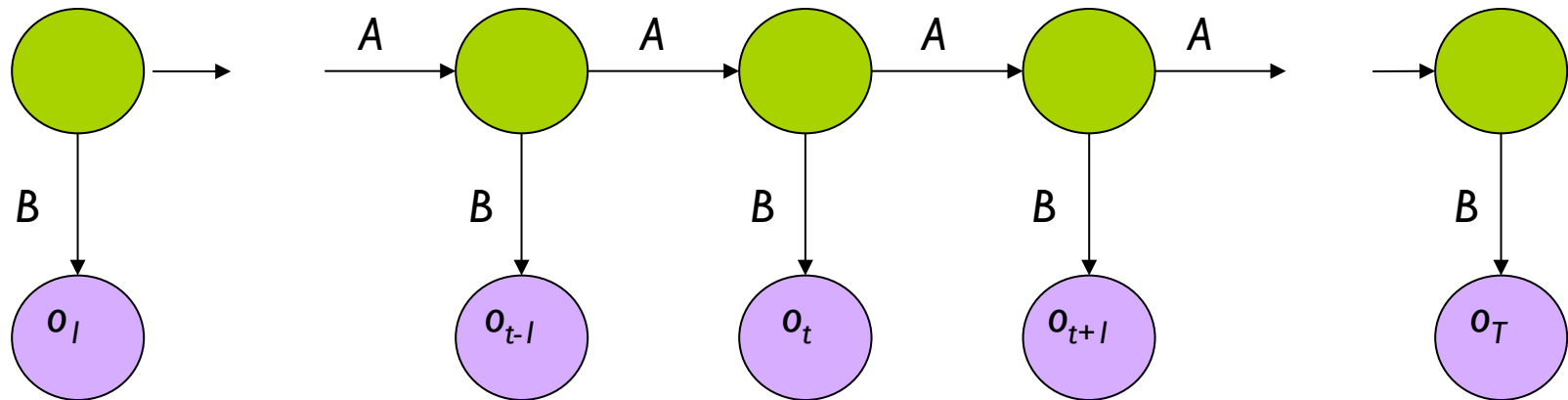
Compute the most likely state sequence by working backwards

Parameter Estimation



- Given an observation sequence, find the model that is most likely to produce that sequence.
- No analytic method
- Given a model and observation sequence, update the model parameters to better fit the observations.

Parameter Estimation



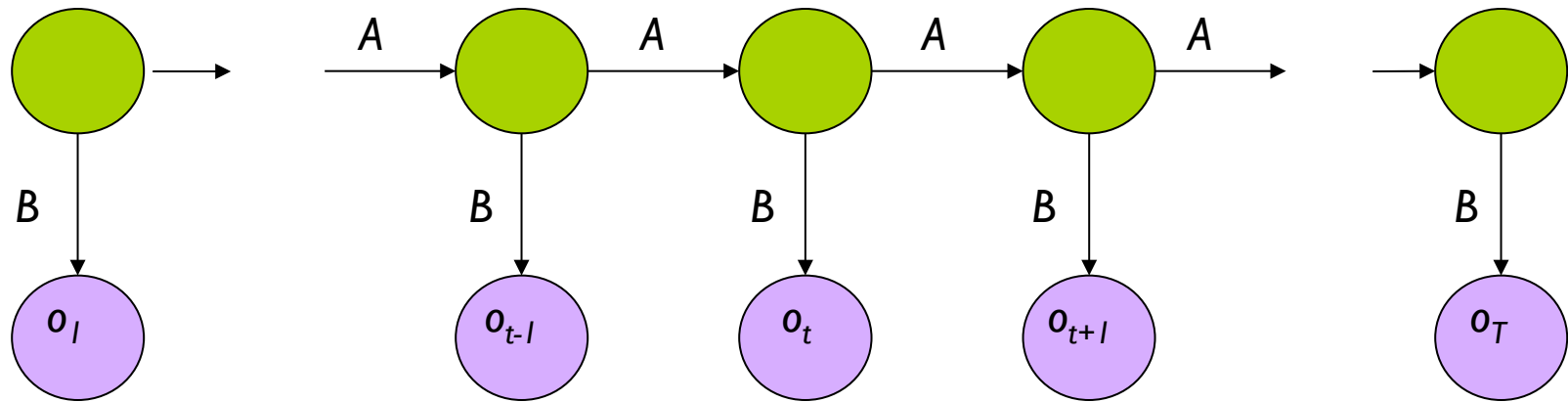
$$p_t(i, j) = \frac{\alpha_i(t) a_{ij} b_{j o_{t+1}} \beta_j(t+1)}{\sum_{m=1 \dots N} \alpha_m(t) \beta_m(t)}$$

Probability of
traversing an arc

$$\gamma_i(t) = \sum_{j=1 \dots N} p_t(i, j)$$

Probability of being
in state i

Parameter Estimation



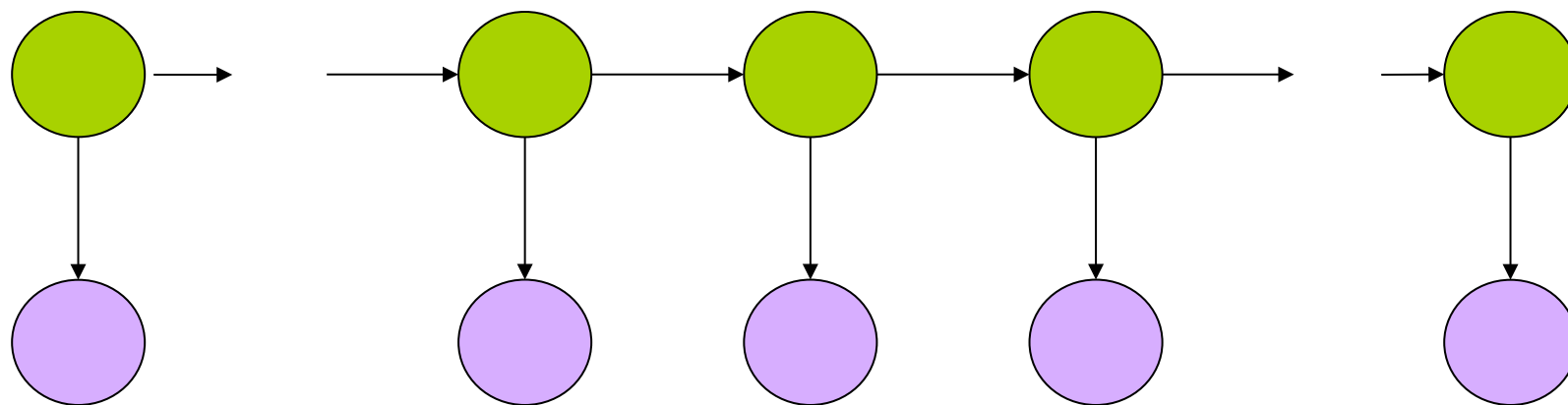
$$\hat{\pi}_i = \gamma_i(1)$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^T p_t(i, j)}{\sum_{t=1}^T \gamma_i(t)}$$

$$\hat{b}_{ik} = \frac{\sum_{\{t: o_t=k\}} \gamma_t(i)}{\sum_{t=1}^T \gamma_i(t)}$$

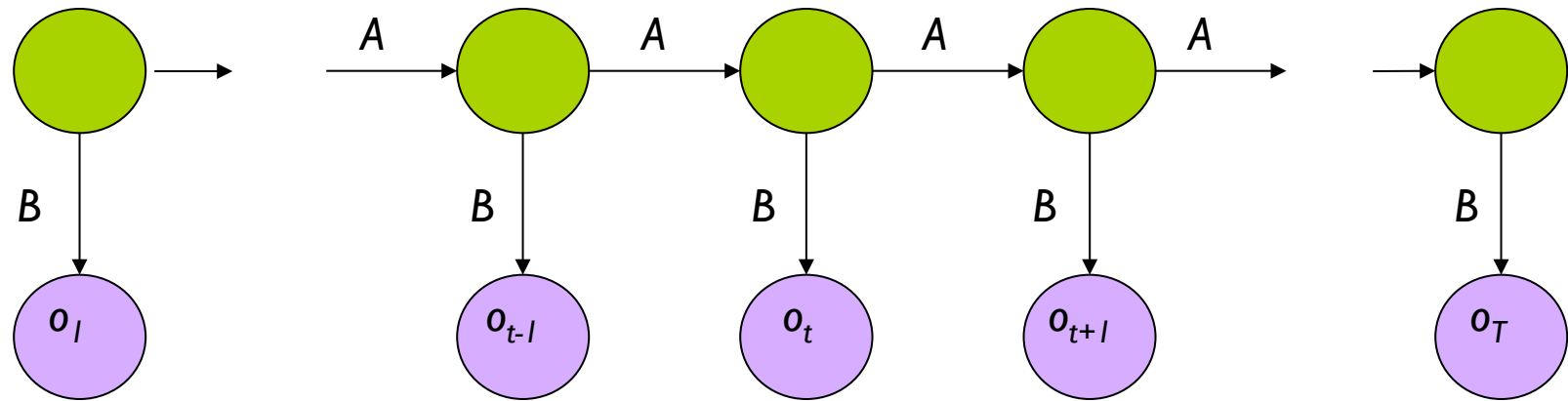
Now we can
compute the new
estimates of the
model parameters.

HMM Applications



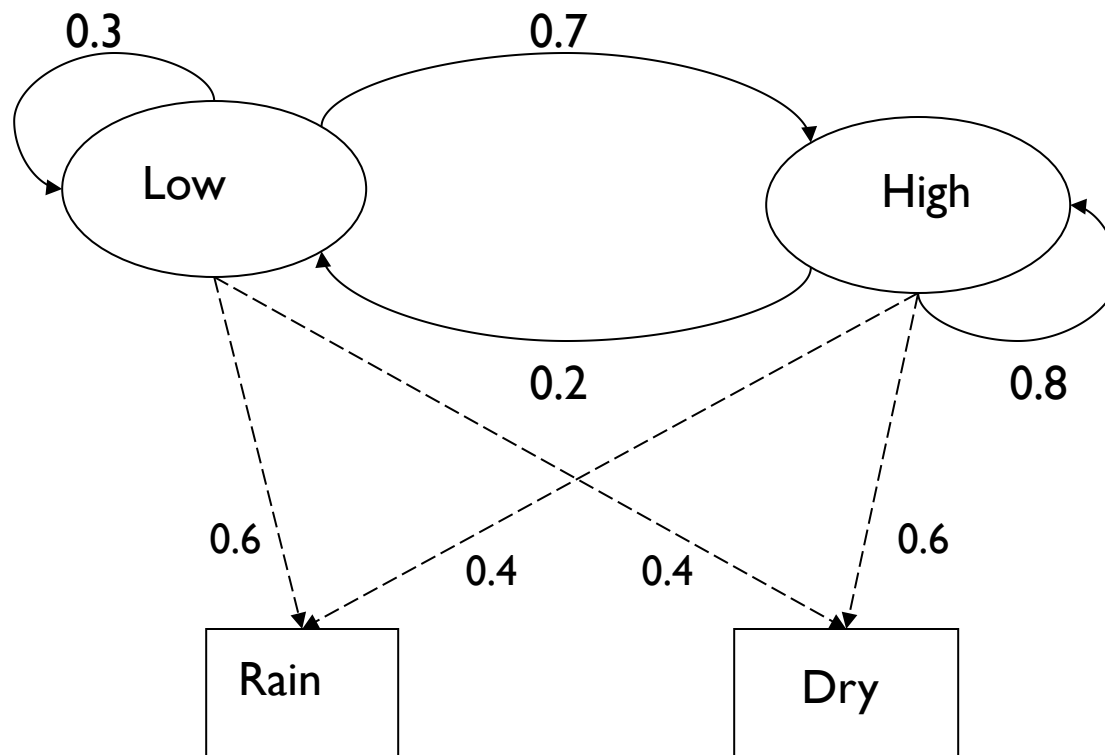
- Generating parameters for n-gram models
- Tagging speech
- Speech recognition

The Most Important Thing

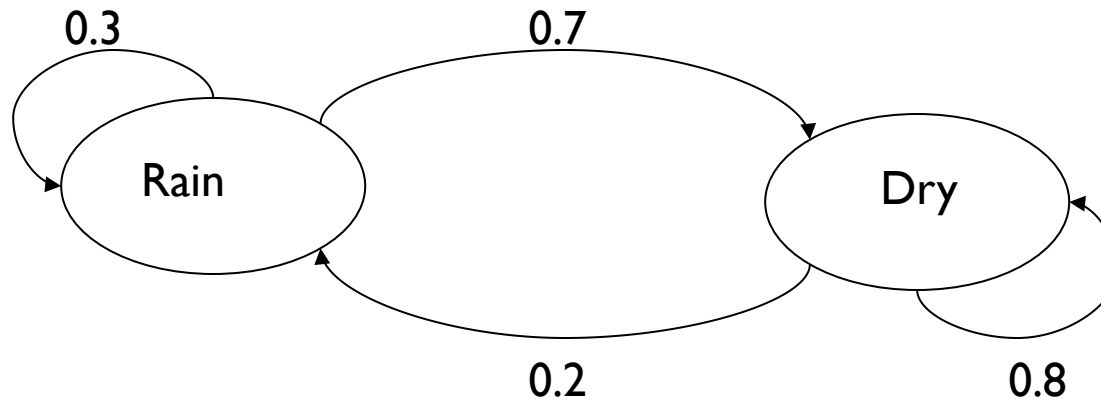


We can use the special structure of this model to do a lot of neat math and solve problems that are otherwise not solvable.

Example of Hidden Markov Model



Example of Markov Model



- Two states : 'Rain' and 'Dry' .
- Transition probabilities: $P(\text{'Rain' } | \text{'Rain'}) = 0.3$, $P(\text{'Dry' } | \text{'Rain'}) = 0.7$,
 $P(\text{'Rain' } | \text{'Dry'}) = 0.2$, $P(\text{'Dry' } | \text{'Dry'}) = 0.8$
- Initial probabilities: say $P(\text{'Rain'}) = 0.4$, $P(\text{'Dry'}) = 0.6$.

Calculation of sequence probability

- By Markov chain property, probability of state sequence can be found by the formula:

$$\begin{aligned}P(s_{i1}, s_{i2}, \dots, s_{ik}) &= P(s_{ik} \mid s_{i1}, s_{i2}, \dots, s_{ik-1}) P(s_{i1}, s_{i2}, \dots, s_{ik-1}) \\&= P(s_{ik} \mid s_{ik-1}) P(s_{i1}, s_{i2}, \dots, s_{ik-1}) = \dots \\&= P(s_{ik} \mid s_{ik-1}) P(s_{ik-1} \mid s_{ik-2}) \dots P(s_{i2} \mid s_{i1}) P(s_{i1})\end{aligned}$$

- Suppose we want to calculate a probability of a sequence of states in our example, { 'Dry' , ' Dry' , ' Rain' ,Rain' }.

$$\begin{aligned}&P(\{ \text{'Dry' , ' Dry' , ' Rain' ,Rain' } \}) = \\&P(\text{'Rain' } \mid \text{' Rain' }) P(\text{'Rain' } \mid \text{' Dry' }) P(\text{'Dry' } \mid \text{' Dry' }) \\&P(\text{'Dry' }) = \\&= 0.3 * 0.2 * 0.8 * 0.6\end{aligned}$$

Hidden Markov models

- Set of states: $\{s_1, s_2, \dots, s_N\}$
- Process moves from one state to another generating a sequence of states :
 $s_{i1}, s_{i2}, \dots, s_{ik}, \dots$
- Markov chain property: probability of each subsequent state depends only on what was the previous state:
$$P(s_{ik} \mid s_{i1}, s_{i2}, \dots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$
- States are not visible, but each state randomly generates one of M observations (or visible states) $\{v_1, v_2, \dots, v_M\}$
- To define hidden Markov model, the following probabilities have to be specified: matrix of transition probabilities $A=(a_{ij})$, $a_{ij} = P(s_i \mid s_j)$, matrix of observation probabilities $B=(b_i(v_m))$, $b_i(v_m) = P(v_m \mid s_i)$ and a vector of initial probabilities $\pi=(\pi_i)$, $\pi_i = P(s_i)$. Model is represented by $M=(A, B, \pi)$.

Example of Hidden Markov Model

- Two states : 'Low' and 'High' atmospheric pressure.
- Two observations : 'Rain' and 'Dry' .
- Transition probabilities: $P(\text{'Low' | 'Low'})=0.3$, $P(\text{'High' | 'Low'})=0.7$,
 $P(\text{'Low' | 'High'})=0.2$, $P(\text{'High' | 'High'})=0.8$
- Observation probabilities : $P(\text{'Rain' | 'Low'})=0.6$, $P(\text{'Dry' | 'Low'})=0.4$,
 $P(\text{'Rain' | 'High'})=0.4$, $P(\text{'Dry' | 'High'})=0.3$.
- Initial probabilities: say $P(\text{'Low'})=0.4$, $P(\text{'High'})=0.6$.

Calculation of observation sequence probability

- Suppose we want to calculate a probability of a sequence of observations in our example, { 'Dry' , ' Rain' }.
- Consider all possible hidden state sequences:

$$P(\{ \text{'Dry' , ' Rain' } \}) = P(\{ \text{'Dry' , ' Rain' } \}, \{ \text{'Low' , ' Low' } \}) + \\ P(\{ \text{'Dry' , ' Rain' } \}, \{ \text{'Low' , ' High' } \}) + P(\{ \text{'Dry' , ' Rain' } \}, \\ \{ \text{'High' , ' Low' } \}) + P(\{ \text{'Dry' , ' Rain' } \}, \{ \text{'High' , ' High' } \})$$

where first term is :

$$P(\{ \text{'Dry' , ' Rain' } \}, \{ \text{'Low' , ' Low' } \}) = \\ P(\{ \text{'Dry' , ' Rain' } \} | \{ \text{'Low' , ' Low' } \}) P(\{ \text{'Low' , ' Low' } \}) = \\ P(\text{'Dry' } | \text{'Low' }) P(\text{'Rain' } | \text{'Low' }) P(\text{'Low' }) P(\text{'Low' } | \text{'Low' }) \\ = 0.4 * 0.4 * 0.6 * 0.4 * 0.3$$

Main issues using HMMs :

Evaluation problem. Given the HMM $M=(A, B, \pi)$ and the observation sequence $O=o_1 o_2 \dots o_K$, calculate the probability that model M has generated sequence O .

• **Decoding problem.** Given the HMM $M=(A, B, \pi)$ and the observation sequence $O=o_1 o_2 \dots o_K$, calculate the most likely sequence of hidden states S_i that produced this observation sequence O .

• **Learning problem.** Given some training observation sequences $O=o_1 o_2 \dots o_K$ and general structure of HMM (numbers of hidden and visible states), determine HMM parameters $M=(A, B, \pi)$ that best fit training data.

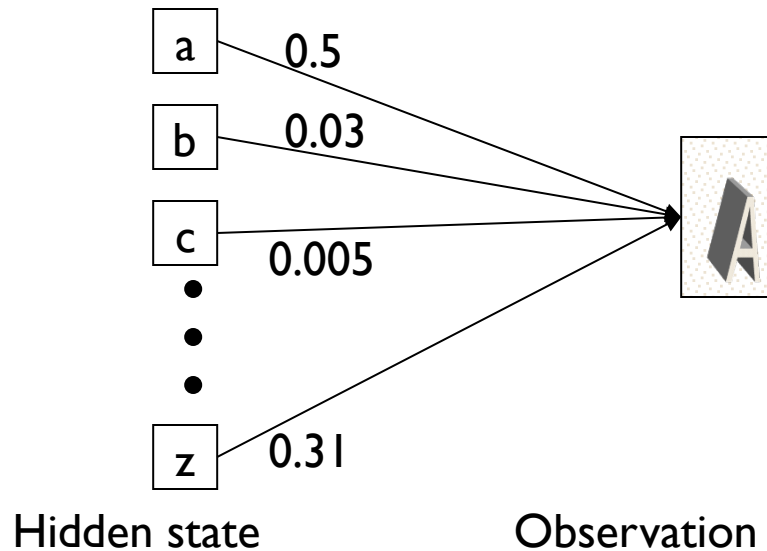
$O=o_1 \dots o_K$ denotes a sequence of observations $o_k \in \{v_1, \dots, v_M\}$.

Word recognition example(I).

- Typed word recognition, assume all characters are separated.



- Character recognizer outputs probability of the image being particular character, $P(\text{image}|\text{character})$.



Word recognition example(2).

- Hidden states of HMM = characters.
- Observations = typed images of characters segmented from the image v_α . Note that there is an infinite number of observations

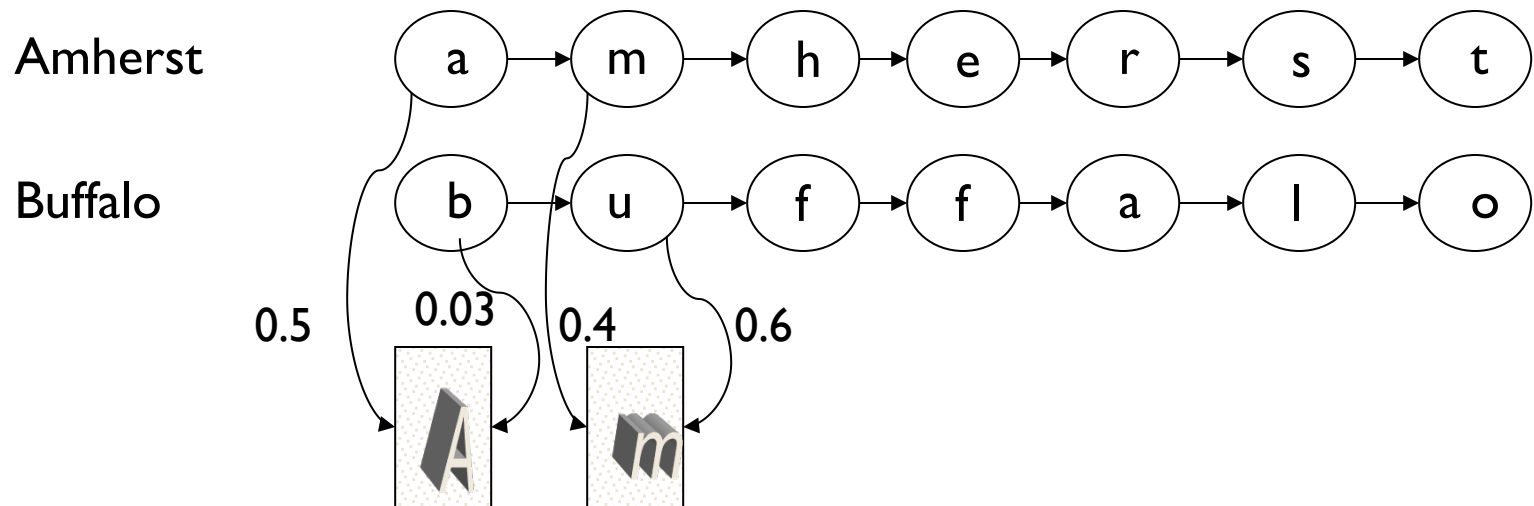
- Observation probabilities = character recognizer scores.

$$B = (b_i(v_\alpha)) = (P(v_\alpha | s_i))$$

- Transition probabilities will be defined differently in two subsequent models.

Word recognition example(3).

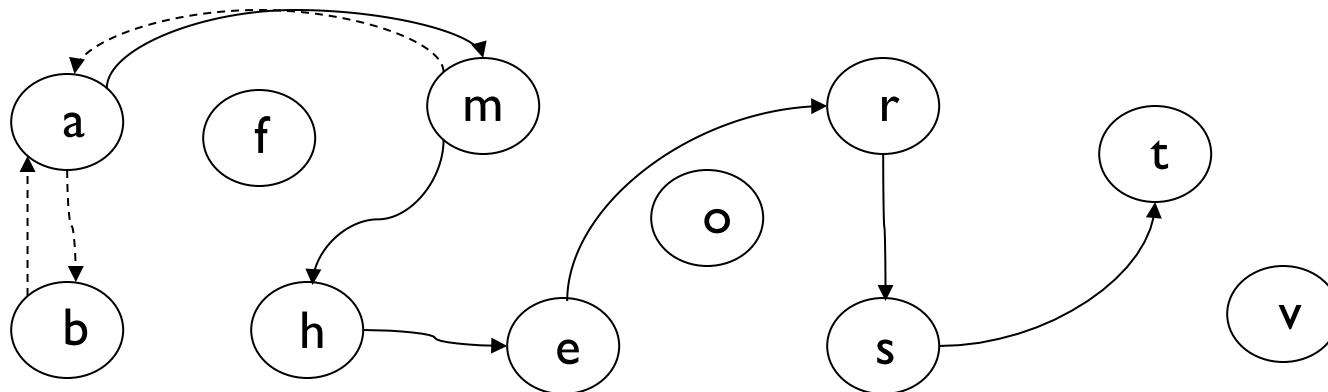
- If lexicon is given, we can construct separate HMM models for each lexicon word.



- Here recognition of word image is equivalent to the problem of evaluating few HMM models.
- This is an application of **Evaluation problem**.

Word recognition example(4).

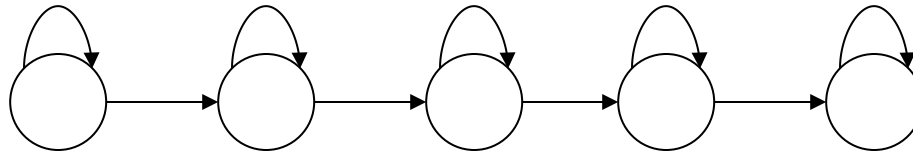
- We can construct a single HMM for all words.
- Hidden states = all characters in the alphabet.
- Transition probabilities and initial probabilities are calculated from language model.
- Observations and observation probabilities are as before.



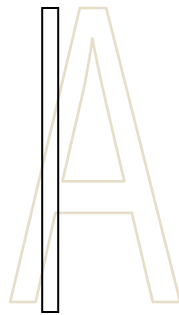
- Here we have to determine the best sequence of hidden states, the one that most likely produced word image.
- This is an application of **Decoding problem**.

Character recognition with HMM example.

- The structure of hidden states is chosen.



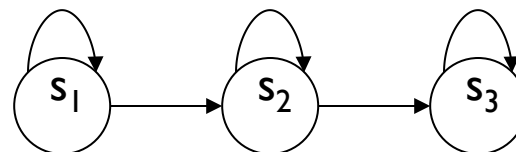
- Observations are feature vectors extracted from vertical slices.



- Probabilistic mapping from hidden state to feature vectors:
 1. use mixture of Gaussian models
 2. Quantize feature vector space.

Exercise: character recognition with HMM(I)

- The structure of hidden states:

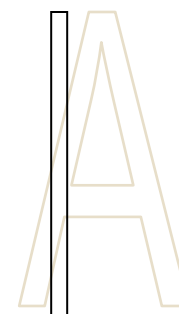


- Observation = number of islands in the vertical slice.

- HMM for character 'A' :

Transition probabilities: $\{a_{ij}\} = \begin{pmatrix} .8 & .2 & 0 \\ 0 & .8 & .2 \\ 0 & 0 & 1 \end{pmatrix}$

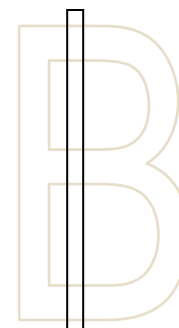
Observation probabilities: $\{b_{jk}\} = \begin{pmatrix} .9 & .1 & 0 \\ .1 & .8 & .1 \\ .9 & .1 & 0 \end{pmatrix}$



- HMM for character 'B' :

Transition probabilities: $\{a_{ij}\} = \begin{pmatrix} .8 & .2 & 0 \\ 0 & .8 & .2 \\ 0 & 0 & 1 \end{pmatrix}$

Observation probabilities: $\{b_{jk}\} = \begin{pmatrix} .9 & .1 & 0 \\ 0 & .2 & .8 \\ .6 & .4 & 0 \end{pmatrix}$



Exercise: character recognition with HMM(2)

- Suppose that after character image segmentation the following sequence of island numbers in 4 slices was observed:

$\{ 1, 3, 2, 1 \}$

- What HMM is more likely to generate this observation sequence , HMM for 'A' or HMM for 'B' ?

Exercise: character recognition with HMM(3)

Consider likelihood of generating given observation for each possible sequence of hidden states:

- HMM for character 'A' :

Hidden state sequence	Transition probabilities	Observation probabilities
$s_1 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3$	$.8 * .2 * .2$	$* .9 * 0 * .8 * .9 = 0$
$s_1 \rightarrow s_2 \rightarrow s_2 \rightarrow s_3$	$.2 * .8 * .2$	$* .9 * .1 * .8 * .9 = 0.0020736$
$s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_3$	$.2 * .2 * 1$	$* .9 * .1 * .1 * .9 = 0.000324$
Total = 0.0023976		

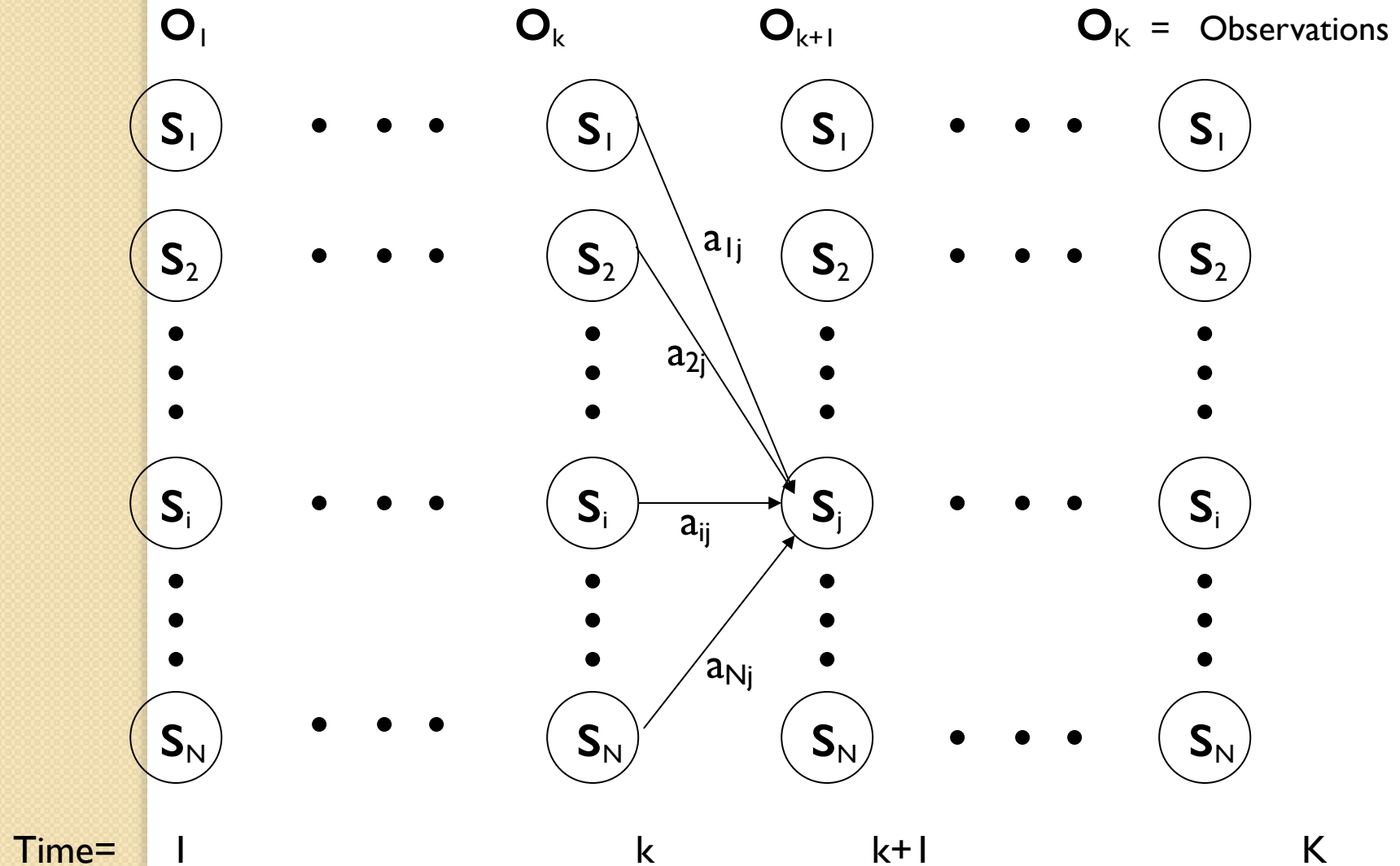
- HMM for character 'B' :

Hidden state sequence	Transition probabilities	Observation probabilities
$s_1 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3$	$.8 * .2 * .2$	$* .9 * 0 * .2 * .6 = 0$
$s_1 \rightarrow s_2 \rightarrow s_2 \rightarrow s_3$	$.2 * .8 * .2$	$* .9 * .8 * .2 * .6 = 0.0027648$
$s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_3$	$.2 * .2 * 1$	$* .9 * .8 * .4 * .6 = 0.006912$
Total = 0.0096768		

Evaluation Problem.

- **Evaluation problem.** Given the HMM $M=(A, B, \pi)$ and the observation sequence $O=o_1 o_2 \dots o_K$, calculate the probability that model M has generated sequence O .
- Trying to find probability of observations $O=o_1 o_2 \dots o_K$ by means of considering all hidden state sequences (as was done in example) is impractical:
 N^K hidden state sequences - exponential complexity.
- Use **Forward-Backward HMM algorithms** for efficient calculations.
- Define the forward variable $\alpha_k(i)$ as the joint probability of the partial observation sequence $o_1 o_2 \dots o_k$ and that the hidden state at time k is s_i : $\alpha_k(i) = P(o_1 o_2 \dots o_k, q_k = s_i)$

Trellis representation of an HMM



Forward recursion for HMM

- Initialization:

$$\alpha_1(i) = P(o_1, q_1 = s_i) = \pi_i b_i(o_1), \quad 1 \leq i \leq N.$$

- Forward recursion:

$$\begin{aligned} \alpha_{k+1}(j) &= P(o_1 o_2 \dots o_{k+1}, q_{k+1} = s_j) = \\ &= \sum_i P(o_1 o_2 \dots o_{k+1}, q_k = s_i, q_{k+1} = s_j) = \\ &= \sum_i P(o_1 o_2 \dots o_k, q_k = s_i) a_{ij} b_j(o_{k+1}) = \\ &= \left[\sum_i \alpha_k(i) a_{ij} \right] b_j(o_{k+1}), \quad 1 \leq j \leq N, 1 \leq k \leq K-1. \end{aligned}$$

- Termination:

$$P(o_1 o_2 \dots o_K) = \sum_i P(o_1 o_2 \dots o_K, q_K = s_i) = \sum_i \alpha_K(i)$$

- Complexity :
N²K operations.

Backward recursion for HMM

- Define the forward variable $\beta_k(i)$ as the joint probability of the partial observation sequence $\mathbf{o}_{k+1} \mathbf{o}_{k+2} \dots \mathbf{o}_K$ given that the hidden state at time k is \mathbf{s}_i : $\beta_k(i) = P(\mathbf{o}_{k+1} \mathbf{o}_{k+2} \dots \mathbf{o}_K | \mathbf{q}_k = \mathbf{s}_i)$

- Initialization:

$$\beta_K(i) = 1, \quad 1 \leq i \leq N.$$

- Backward recursion:

$$\begin{aligned} \beta_k(j) &= P(\mathbf{o}_{k+1} \mathbf{o}_{k+2} \dots \mathbf{o}_K | \mathbf{q}_k = \mathbf{s}_j) = \\ &\sum_i P(\mathbf{o}_{k+1} \mathbf{o}_{k+2} \dots \mathbf{o}_K, \mathbf{q}_{k+1} = \mathbf{s}_i | \mathbf{q}_k = \mathbf{s}_j) = \\ &\sum_i P(\mathbf{o}_{k+2} \mathbf{o}_{k+3} \dots \mathbf{o}_K | \mathbf{q}_{k+1} = \mathbf{s}_i) a_{ji} b_i(\mathbf{o}_{k+1}) = \\ &\sum_i \beta_{k+1}(i) a_{ji} b_i(\mathbf{o}_{k+1}), \quad 1 \leq j \leq N, 1 \leq k \leq K-1. \end{aligned}$$

- Termination:

$$\begin{aligned} P(\mathbf{o}_1 \mathbf{o}_2 \dots \mathbf{o}_K) &= \sum_i P(\mathbf{o}_1 \mathbf{o}_2 \dots \mathbf{o}_K, \mathbf{q}_1 = \mathbf{s}_i) = \\ &\sum_i P(\mathbf{o}_1 \mathbf{o}_2 \dots \mathbf{o}_K | \mathbf{q}_1 = \mathbf{s}_i) P(\mathbf{q}_1 = \mathbf{s}_i) = \sum_i \beta_1(i) b_i(\mathbf{o}_1) \pi_i \end{aligned}$$

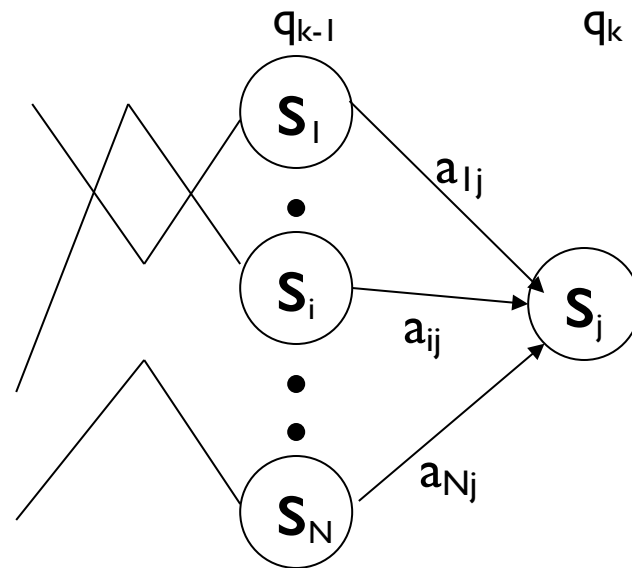
Decoding problem

- **Decoding problem.** Given the HMM $M=(A, B, \pi)$ and the observation sequence $O=o_1 o_2 \dots o_K$, calculate the most likely sequence of hidden states S_i that produced this observation sequence.
- We want to find the state sequence $Q=q_1 \dots q_K$ which maximizes $P(Q \mid o_1 o_2 \dots o_K)$, or equivalently $P(Q, o_1 o_2 \dots o_K)$.
- Brute force consideration of all paths takes exponential time. Use efficient **Viterbi algorithm** instead.
- Define variable $\delta_k(i)$ as the maximum probability of producing observation sequence $o_1 o_2 \dots o_k$ when moving along any hidden state sequence $q_1 \dots q_{k-1}$ and getting into $q_k = s_i$.
$$\delta_k(i) = \max P(q_1 \dots q_{k-1}, q_k = s_i, o_1 o_2 \dots o_k)$$
where max is taken over all possible paths $q_1 \dots q_{k-1}$.

Viterbi algorithm (I)

- General idea:

if best path ending in $q_k = s_j$ goes through $q_{k-1} = s_i$ then it should coincide with best path ending in $q_{k-1} = s_i$.



$$\delta_k(i) = \max P(q_1 \dots q_{k-1}, q_k = s_j, o_1 o_2 \dots o_k) = \max_i [a_{ij} b_j(o_k) \max P(q_1 \dots q_{k-1} = s_i, o_1 o_2 \dots o_{k-1})]$$

- To backtrack best path keep info that predecessor of s_j was s_i .

Viterbi algorithm (2)

- Initialization:

$$\delta_1(i) = \max P(q_1 = s_i, o_1) = \pi_i b_i(o_1), 1 \leq i \leq N.$$

- Forward recursion:

$$\begin{aligned} \delta_k(j) &= \max P(q_1 \dots q_{k-1}, q_k = s_j, o_1 o_2 \dots o_k) = \\ &= \max_i [a_{ij} b_j(o_k) \max P(q_1 \dots q_{k-1} = s_i, o_1 o_2 \dots o_{k-1})] = \\ &= \max_i [a_{ij} b_j(o_k) \delta_{k-1}(i)], \quad 1 \leq j \leq N, 2 \leq k \leq K. \end{aligned}$$

- Termination: choose best path ending at time K

$$\max_i [\delta_K(i)]$$

- Backtrack best path.

This algorithm is similar to the forward recursion of evaluation problem, with \sum replaced by max and additional backtracking.

Learning problem (I)

- **Learning problem.** Given some training observation sequences $O = o_1 o_2 \dots o_K$ and general structure of HMM (numbers of hidden and visible states), determine HMM parameters $M = (A, B, \pi)$ that best fit training data, that is maximizes $P(O | M)$.
- There is no algorithm producing optimal parameter values.
- Use iterative expectation-maximization algorithm to find local maximum of $P(O | M)$ - **Baum-Welch** algorithm.

Learning problem (2)

- If training data has information about sequence of hidden states (as in word recognition example), then use maximum likelihood estimation of parameters:

$$a_{ij} = P(s_i | s_j) =$$

$$\frac{\text{Number of transitions from state } \mathbf{S}_j \text{ to state } \mathbf{S}_i}{\text{Number of transitions out of state } \mathbf{S}_j}$$

$$b_i(\mathbf{v}_m) = P(\mathbf{v}_m | s_i) = \frac{\text{Number of times observation } \mathbf{V}_m \text{ occurs in state } \mathbf{S}_i}{\text{Number of times in state } \mathbf{S}_i}$$

Baum-Welch algorithm

General idea:

$$a_{ij} = P(s_i | s_j) = \frac{\text{Expected number of transitions from state } \mathbf{S}_j \text{ to state } \mathbf{S}_i}{\text{Expected number of transitions out of state } \mathbf{S}_j}$$

$$b_i(v_m) = P(v_m | s_i) = \frac{\text{Expected number of times observation } \mathbf{V}_m \text{ occurs in state } \mathbf{S}_i}{\text{Expected number of times in state } \mathbf{S}_i}$$

$$\pi_i = P(s_i) = \text{Expected frequency in state } \mathbf{S}_i \text{ at time } k=1.$$

Baum-Welch algorithm: expectation step(I)

- Define variable $\xi_k(i,j)$ as the probability of being in state S_i at time k and in state S_j at time $k+1$, given the observation sequence $O_1 O_2 \dots O_K$.

$$\xi_k(i,j) = P(q_k = s_i, q_{k+1} = s_j \mid o_1 o_2 \dots o_K)$$

$$\xi_k(i,j) = \frac{P(q_k = s_i, q_{k+1} = s_j, o_1 o_2 \dots o_K)}{P(o_1 o_2 \dots o_K)} =$$

$$\frac{P(q_k = s_i, o_1 o_2 \dots o_k) a_{ij} b_j(o_{k+1}) P(o_{k+2} \dots o_K \mid q_{k+1} = s_j)}{P(o_1 o_2 \dots o_K)} =$$

$$\frac{\alpha_k(i) a_{ij} b_j(o_{k+1}) \beta_{k+1}(j)}{\sum_i \sum_j \alpha_k(i) a_{ij} b_j(o_{k+1}) \beta_{k+1}(j)}$$

Baum-Welch algorithm: expectation step(2)

- Define variable $\gamma_k(i)$ as the probability of being in state S_i at time k , given the observation sequence $O_1 O_2 \dots O_K$.

$$\gamma_k(i) = P(q_k = s_i \mid o_1 o_2 \dots o_k)$$

$$\gamma_k(i) = \frac{P(q_k = s_i, o_1 o_2 \dots o_k)}{P(o_1 o_2 \dots o_k)} = \frac{\alpha_k(i) \beta_k(i)}{\sum_i \alpha_k(i) \beta_k(i)}$$

Baum-Welch algorithm: expectation step(3)

• We calculated $\xi_k(i,j) = P(q_k = s_i, q_{k+1} = s_j \mid o_1 o_2 \dots o_K)$
and $\gamma_k(i) = P(q_k = s_i \mid o_1 o_2 \dots o_K)$

• Expected number of transitions from state S_i to state $S_j =$

$$= \sum_k \xi_k(i,j)$$

• Expected number of transitions out of state $S_i = \sum_k \gamma_k(i)$

• Expected number of times observation V_m occurs in state $S_i =$

$$= \sum_k \gamma_k(i), k \text{ is such that } o_k = V_m$$

• Expected frequency in state S_i at time $k=1 : \gamma_1(i)$.

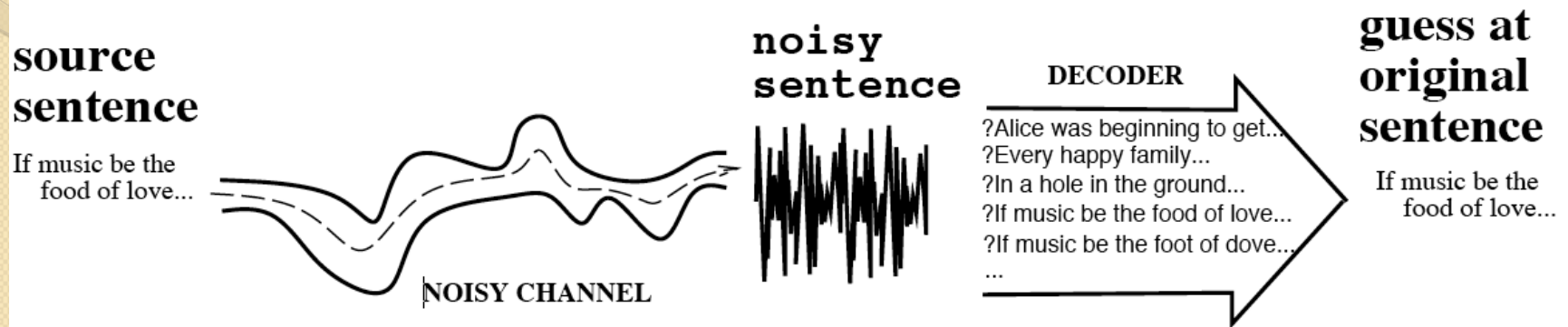
Baum-Welch algorithm: maximization step

$$a_{ij} = \frac{\text{Expected number of transitions from state } S_j \text{ to state } S_i}{\text{Expected number of transitions out of state } S_j} = \frac{\sum_k \xi_{k(i,j)}}{\sum_k \gamma_{k(i)}}$$

$$b_i(v_m) = \frac{\text{Expected number of times observation } v_m \text{ occurs in state } S_i}{\text{Expected number of times in state } S_i} = \frac{\sum_k \xi_{k(i,j)}}{\sum_{k, o_k = v_m} \gamma_{k(i)}}$$

$$\pi_i = (\text{Expected frequency in state } S_i \text{ at time } k=1) = \gamma_1(i).$$

The Noisy Channel Model



- Search through space of all possible sentences.
- Pick the one that is most probable given the waveform.

The Noisy Channel Model (II)

- What is the most likely sentence out of all sentences in the language L given some acoustic input O ?
- Treat acoustic input O as sequence of individual observations
 - $O = o_1, o_2, o_3, \dots, o_t$
- Define a sentence as a sequence of words:
 - $W = w_1, w_2, w_3, \dots, w_n$

Noisy Channel Model (III)

- Probabilistic implication: Pick the highest prob S:

$$\hat{W} = \arg \max_{W \in L} P(W | O)$$

- We can use Bayes rule to rewrite this:

$$\hat{W} = \arg \max_{W \in L} \frac{P(O | W)P(W)}{P(O)}$$

- Since denominator is the same for each candidate sentence W , we can ignore it for the argmax:

$$\hat{W} = \arg \max_{W \in L} P(O | W)P(W)$$

Noisy channel model

likelihood prior

↓ ↓

$$\hat{W} = \arg \max_{W \in L} P(O | W) P(W)$$

The noisy channel model

- Ignoring the denominator leaves us with two factors: $P(\text{Source})$ and $P(\text{Signal} | \text{Source})$

source sentence

If music be the food of love...



noisy sentence



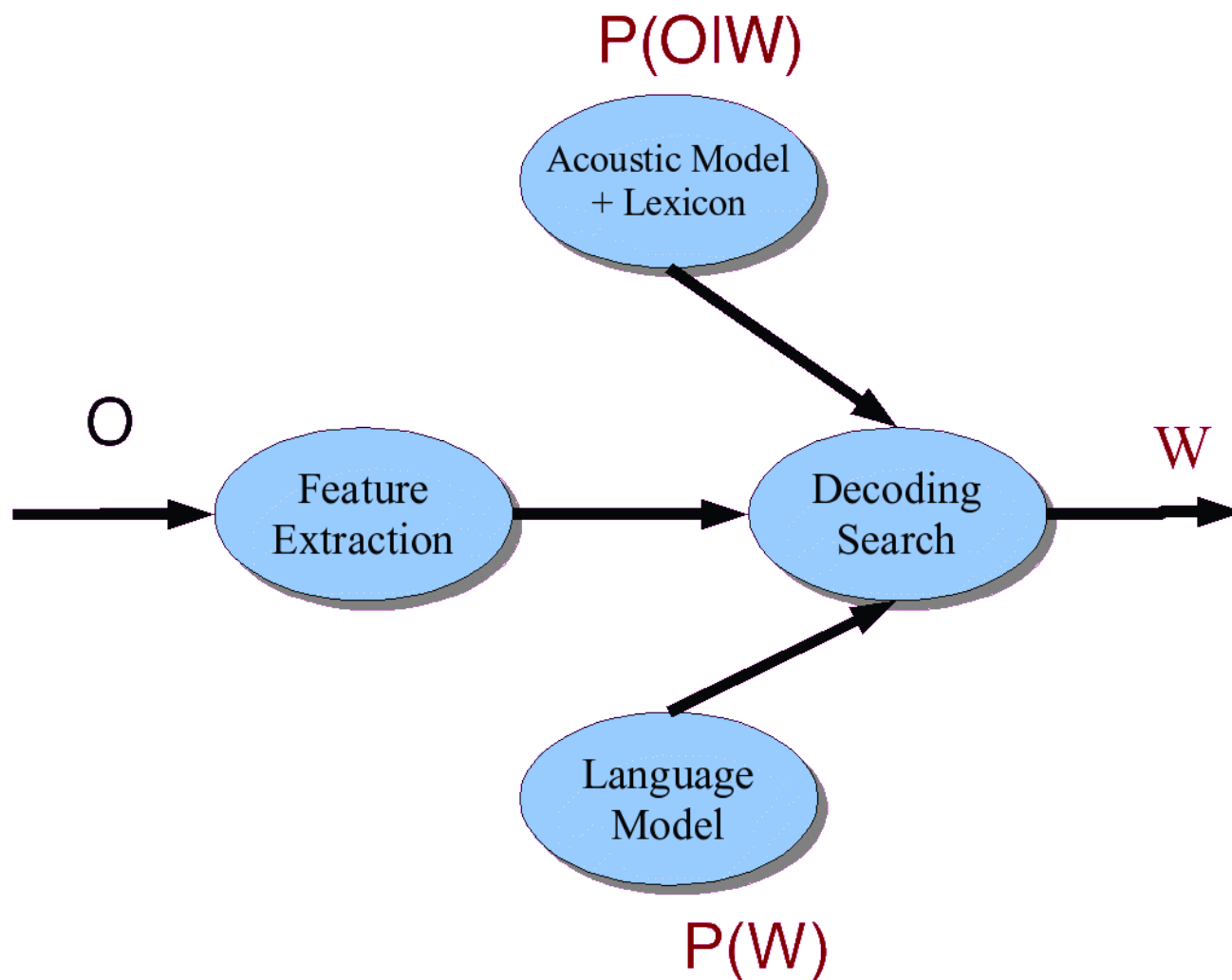
DECODER

?Alice was beginning to get...
?Every happy family...
?In a hole in the ground...
?If music be the food of love...
?If music be the foot of dove...
...

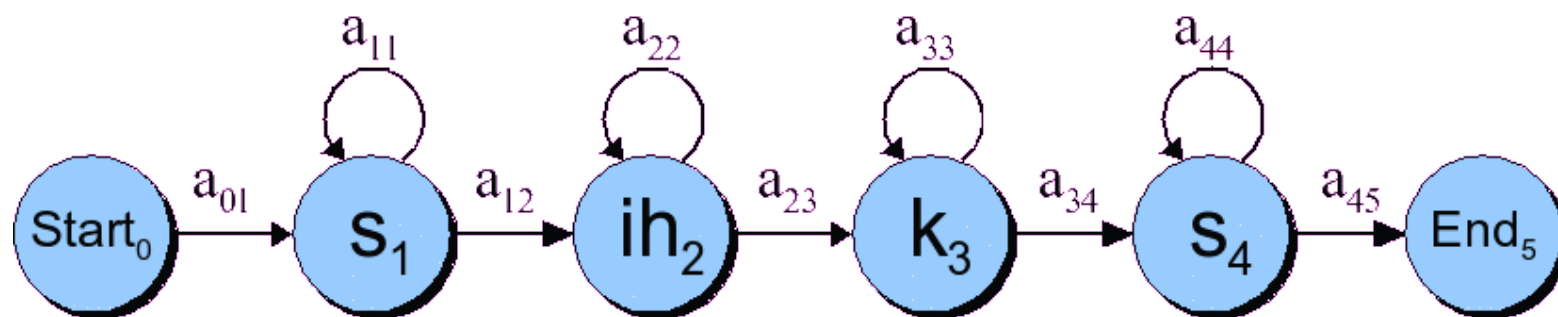
guess at original sentence

If music be the food of love...

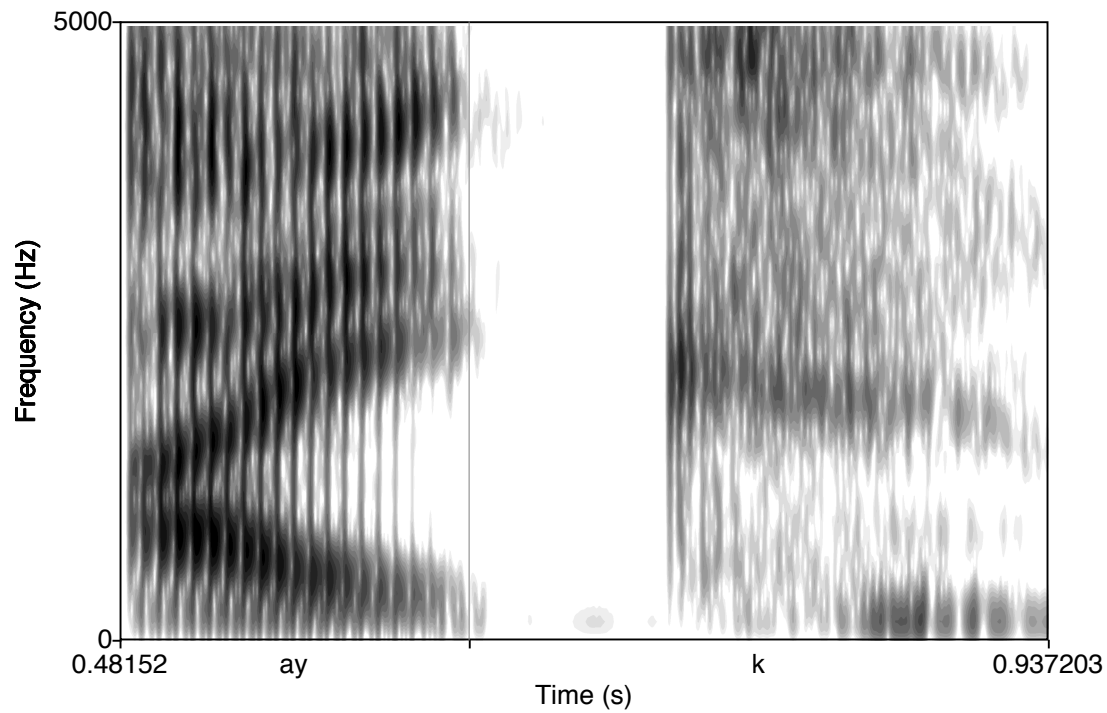
Speech Architecture meets Noisy Channel



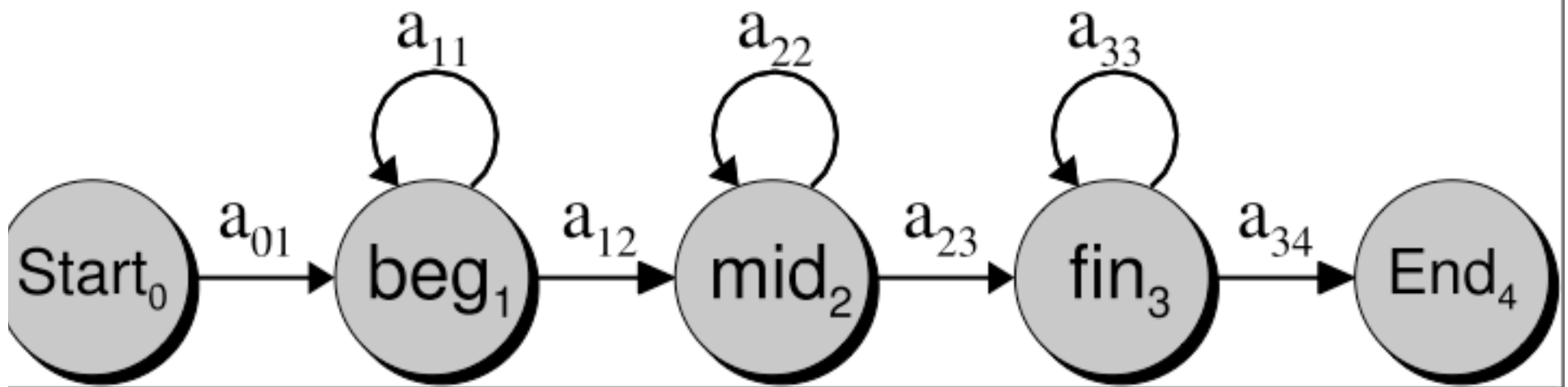
HMMs for speech



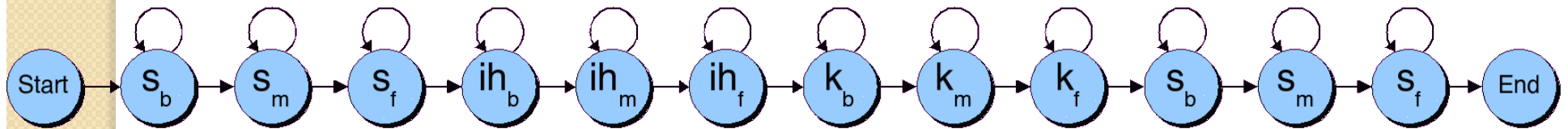
Phones are not homogeneous!



Each phone has 3 subphones



Resulting HMM word model for “six”



HMMs more formally

- Markov chains
- A kind of weighted finite-state automaton

$$Q = q_1 q_2 \dots q_N$$

a set of **states**

$$A = a_{01} a_{02} \dots a_{n1} \dots a_{nn}$$

a **transition probability matrix** A , each a_{ij} representing the probability of moving from state i to state j , s.t. $\sum_{j=1}^n a_{ij} = 1 \quad \forall i$

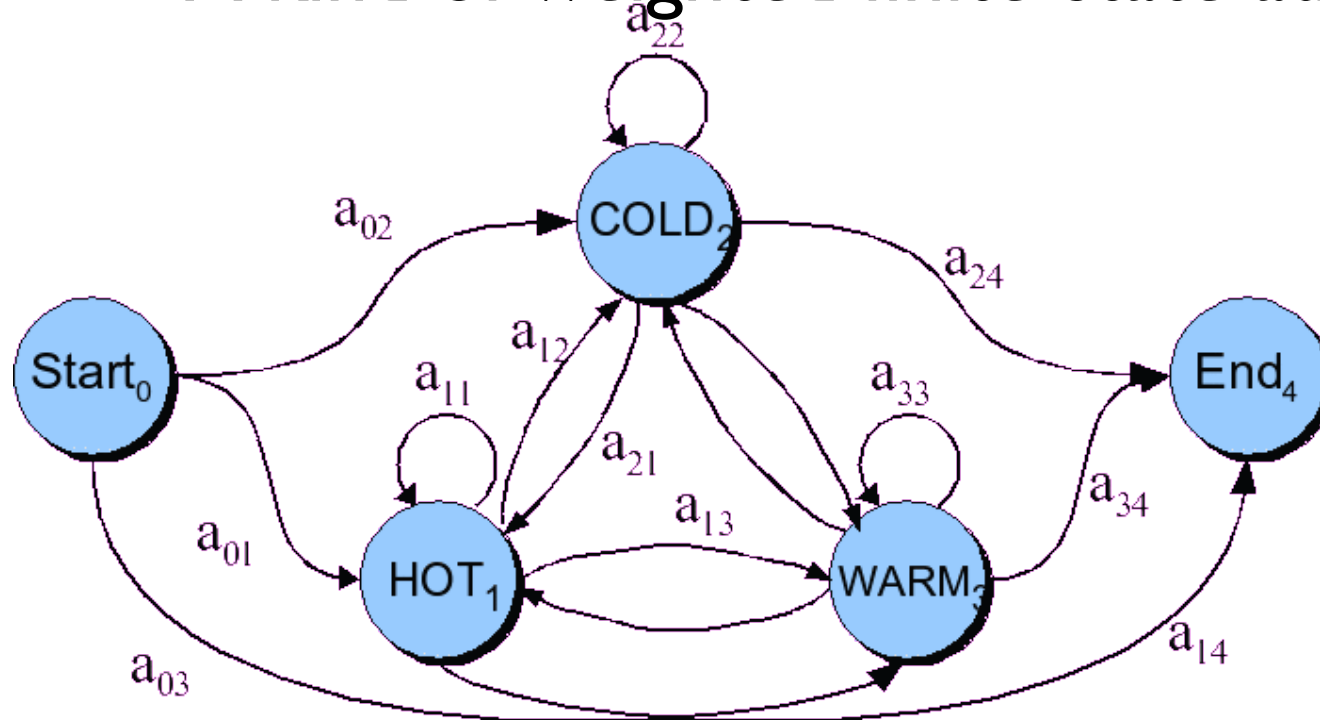
$$q_0, q_{end}$$

a special **start and end state** which are not associated with observations.

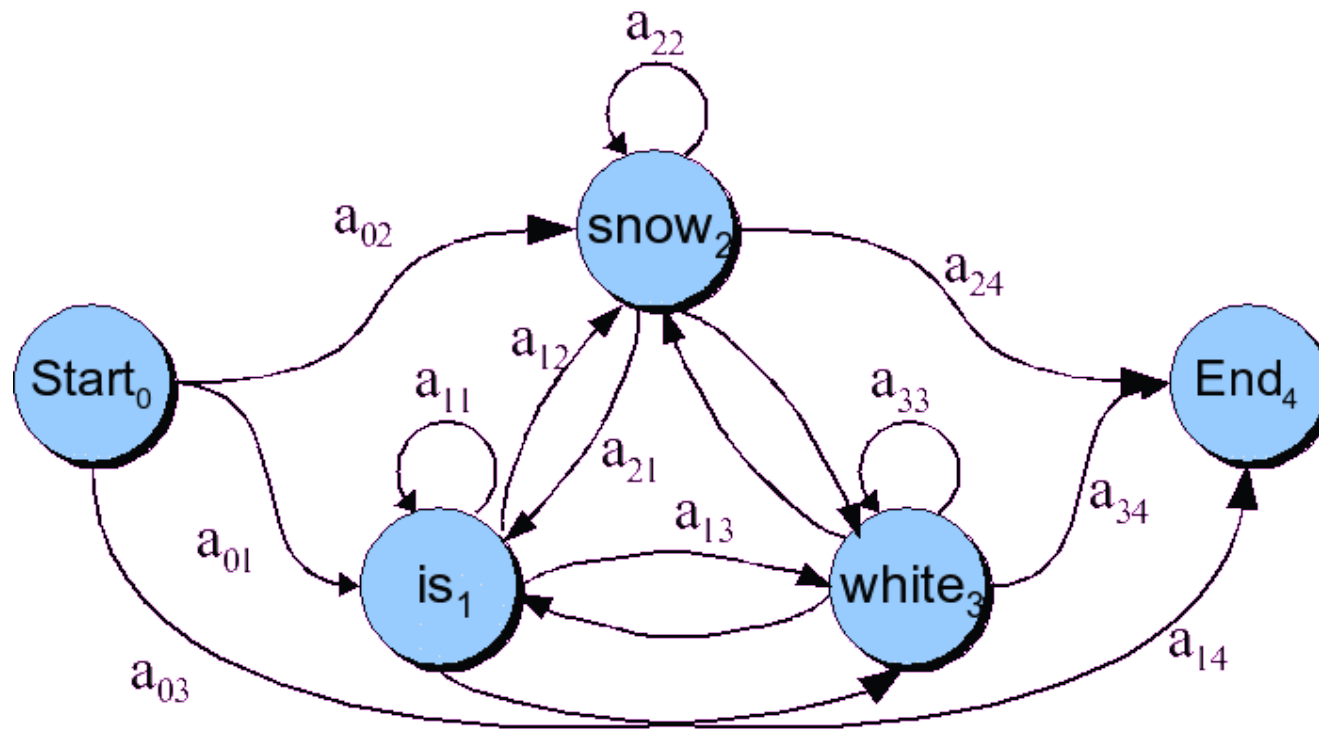
Markov Assumption: $P(q_i | q_1 \dots q_{i-1}) = P(q_i | q_{i-1})$

HMMs more formally

- Markov chains
- A kind of weighted finite-state automaton



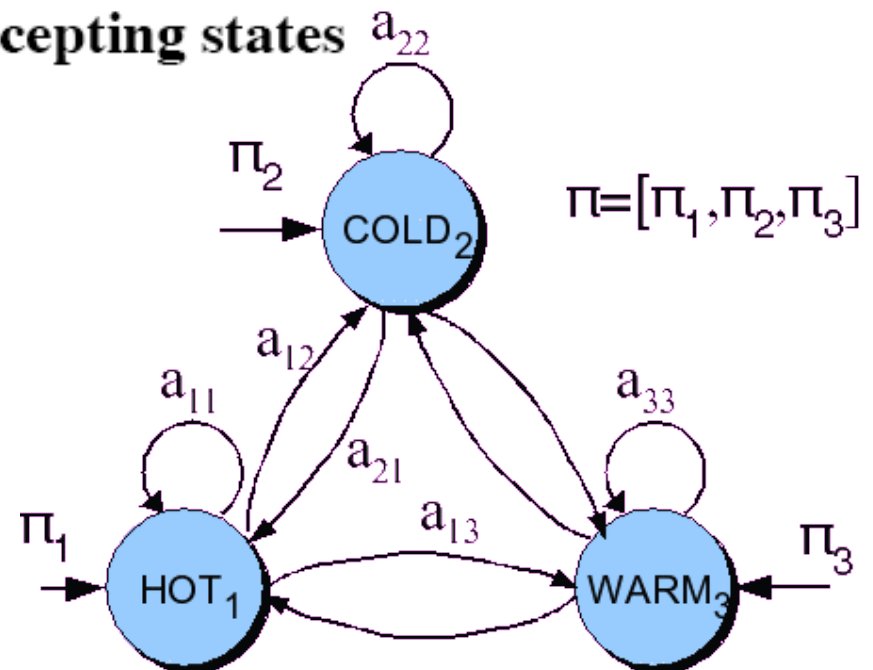
Another Markov chain



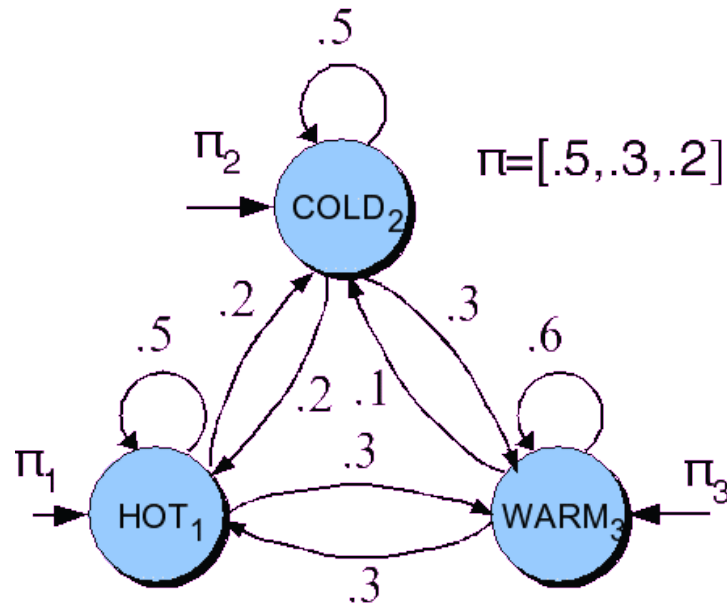
Another view of Markov chains

$\pi = \pi_1, \pi_2, \dots, \pi_N$ an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i . Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^n \pi_i = 1$

$QA = \{q_x, q_y, \dots\}$ a set $QA \subset Q$ of legal **accepting states**



An example with numbers:



- What is probability of:
 - Hot hot hot hot
 - Cold hot cold hot

Hidden Markov Models

$$Q = q_1 q_2 \dots q_N$$

a set of N states

$$A = a_{11} a_{12} \dots a_{n1} \dots a_{nn}$$

a **transition probability matrix** A , each a_{ij} representing the probability of moving from state i to state j , s.t. $\sum_{j=1}^n a_{ij} = 1 \quad \forall i$

$$O = o_1 o_2 \dots o_T$$

a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, \dots, v_V$.

$$B = b_i(o_t)$$

A sequence of **observation likelihoods**:, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i .

$$q_0, q_F$$

a special **start state** and **end (final) state** which are not associated with observations, together with transition probabilities $a_{01} a_{02} \dots a_{0n}$ out of the start state and $a_{1F} a_{2F} \dots a_{nF}$ into the end state.

Hidden Markov Models

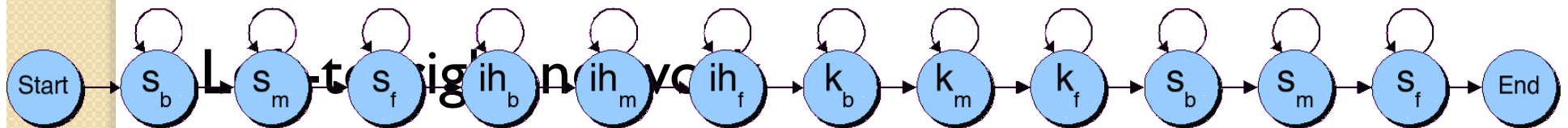
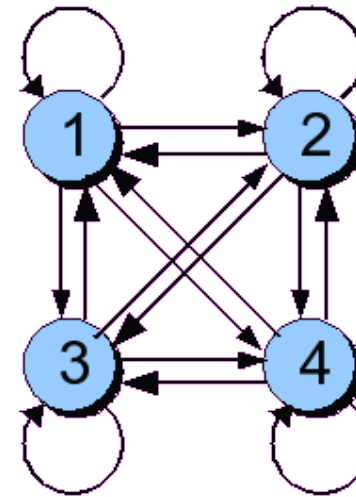
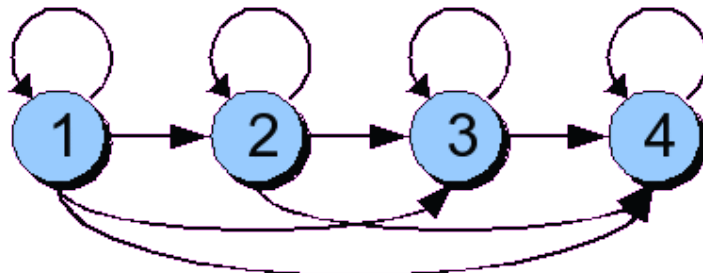
Markov Assumption: $P(q_i | q_1 \dots q_{i-1}) = P(q_i | q_{i-1})$

Output Independence Assumption: $P(o_i | q_1 \dots q_i, \dots, q_n, o_1, \dots, o_i, \dots, o_n) = P(o_i | q_i)$

Hidden Markov Models

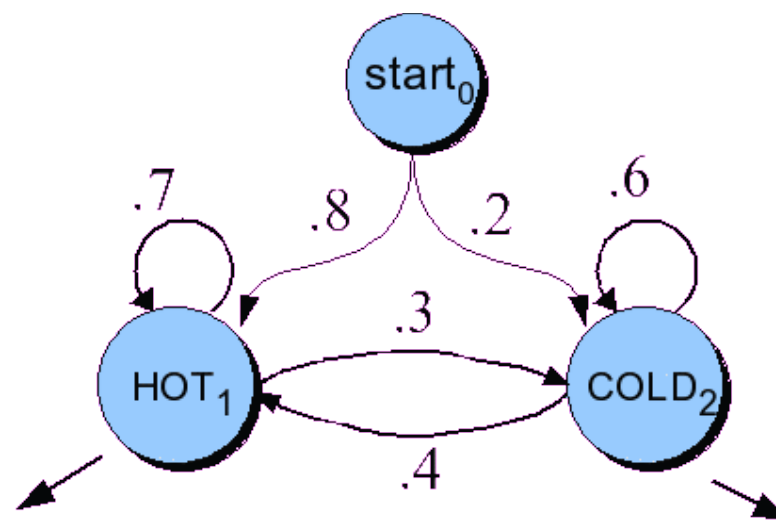
- Bakis network network

Ergodic (fully-connected)



The Jason Eisner task

- You are a climatologist in 2799 studying the history of global warming
- YOU can't find records of the weather in Baltimore for summer 2006
- But you do find Jason Eisner's diary
- Which records how many ice creams he ate each day.
- Can we use this to figure out the weather?
 - Given a sequence of observations O ,
 - each observation an integer = number of ice creams eaten
 - Figure out correct hidden sequence Q of weather states (H or C) which caused Jason to eat the ice cream



$$B_1 \quad \begin{bmatrix} P(1 \mid \text{HOT}) \\ P(2 \mid \text{HOT}) \\ P(3 \mid \text{HOT}) \end{bmatrix} = \begin{bmatrix} .2 \\ .4 \\ .4 \end{bmatrix}$$

$$B_2 \quad \begin{bmatrix} P(1 \mid \text{COLD}) \\ P(2 \mid \text{COLD}) \\ P(3 \mid \text{COLD}) \end{bmatrix} = \begin{bmatrix} .5 \\ .4 \\ .1 \end{bmatrix}$$

HMMs more formally

- Three fundamental problems
 - Jack Ferguson at IDA in the 1960s
 - 1) Given a specific HMM, determine likelihood of observation sequence.
 - 2) Given an observation sequence and an HMM, discover the best (most probable) hidden state sequence
 - 3) Given only an observation sequence, learn the HMM parameters (A, B matrix)

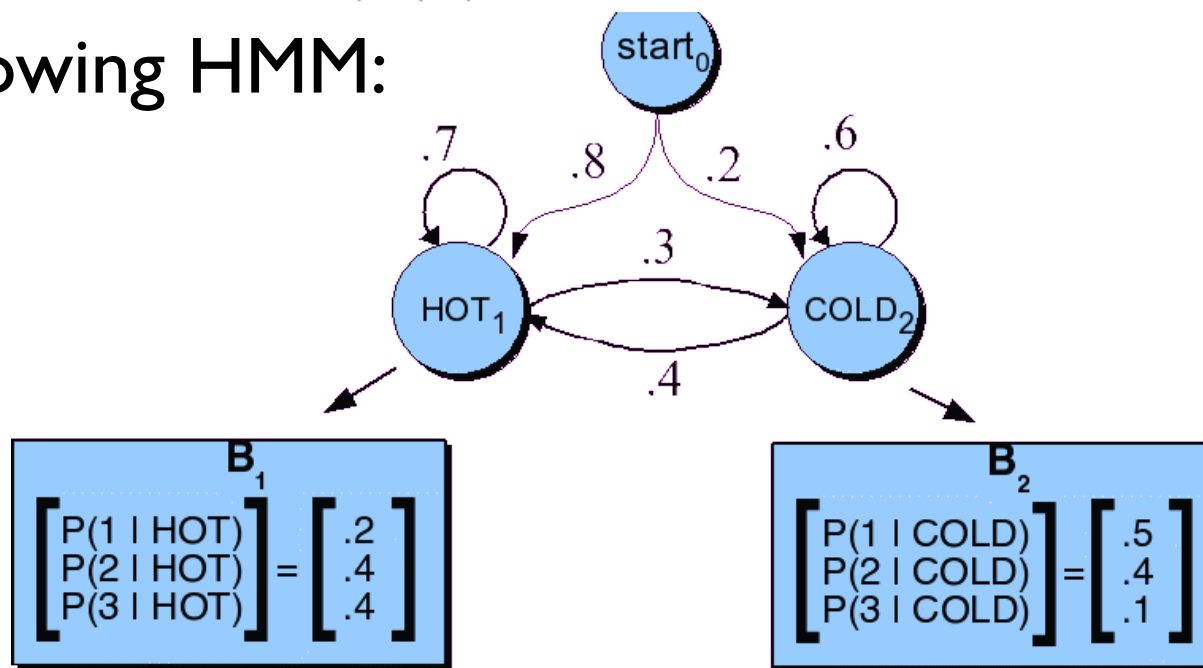
The Three Basic Problems for HMMs

- Problem 1 (**Evaluation**): Given the observation sequence $O=(o_1 o_2 \dots o_T)$, and an HMM model $\Phi = (A,B)$, **how do we efficiently compute $P(O | \Phi)$** , the probability of the observation sequence, given the model
- Problem 2 (**Decoding**): Given the observation sequence $O=(o_1 o_2 \dots o_T)$, and an HMM model $\Phi = (A,B)$, **how do we choose a corresponding state sequence $Q=(q_1 q_2 \dots q_T)$** that is optimal in some sense (i.e., best explains the observations)
- Problem 3 (**Learning**): **How do we adjust the model parameters $\Phi = (A,B)$ to maximize $P(O | \Phi)$** ?

Problem I: computing the observation likelihood

Computing Likelihood: Given an HMM $\lambda = (A, B)$ and an observation sequence O , determine the likelihood $P(O|\lambda)$.

- Given the following HMM:



- How likely is the sequence 3 | 3?

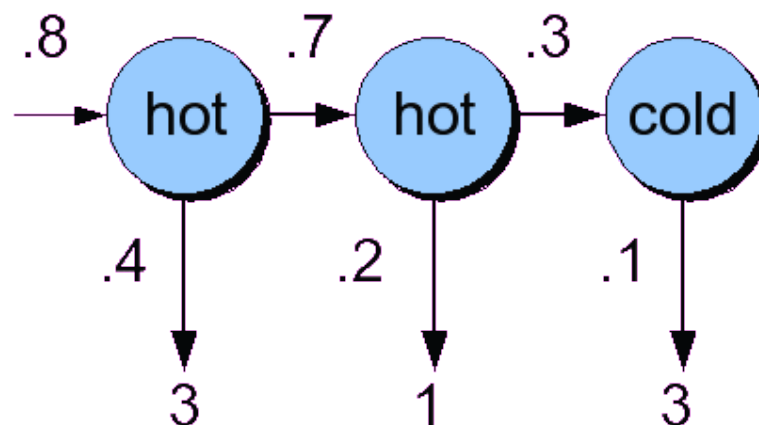
How to compute likelihood

- For a Markov chain, we just follow the states $3 \mid 3$ and multiply the probabilities
- But for an HMM, we don't know what the states are!
- So let's start with a simpler situation.
- Computing the observation likelihood for a **given** hidden state sequence
 - Suppose we knew the weather and wanted to predict how much ice cream Jason would eat.
 - I.e. $P(3 \mid 3 \mid H H C)$

Computing likelihood for I given hidden state sequence

$$P(O|Q) = \prod_{i=1}^n P(o_i|q_i) \times \prod_{i=1}^n P(q_i|q_{i-1})$$

$$P(3 \ 1 \ 3 | \text{hot hot cold}) = P(\text{hot} | \text{start}) \times P(\text{hot} | \text{hot}) \times P(\text{cold} | \text{hot}) \\ \times P(3 | \text{hot}) \times P(1 | \text{hot}) \times P(3 | \text{cold})$$



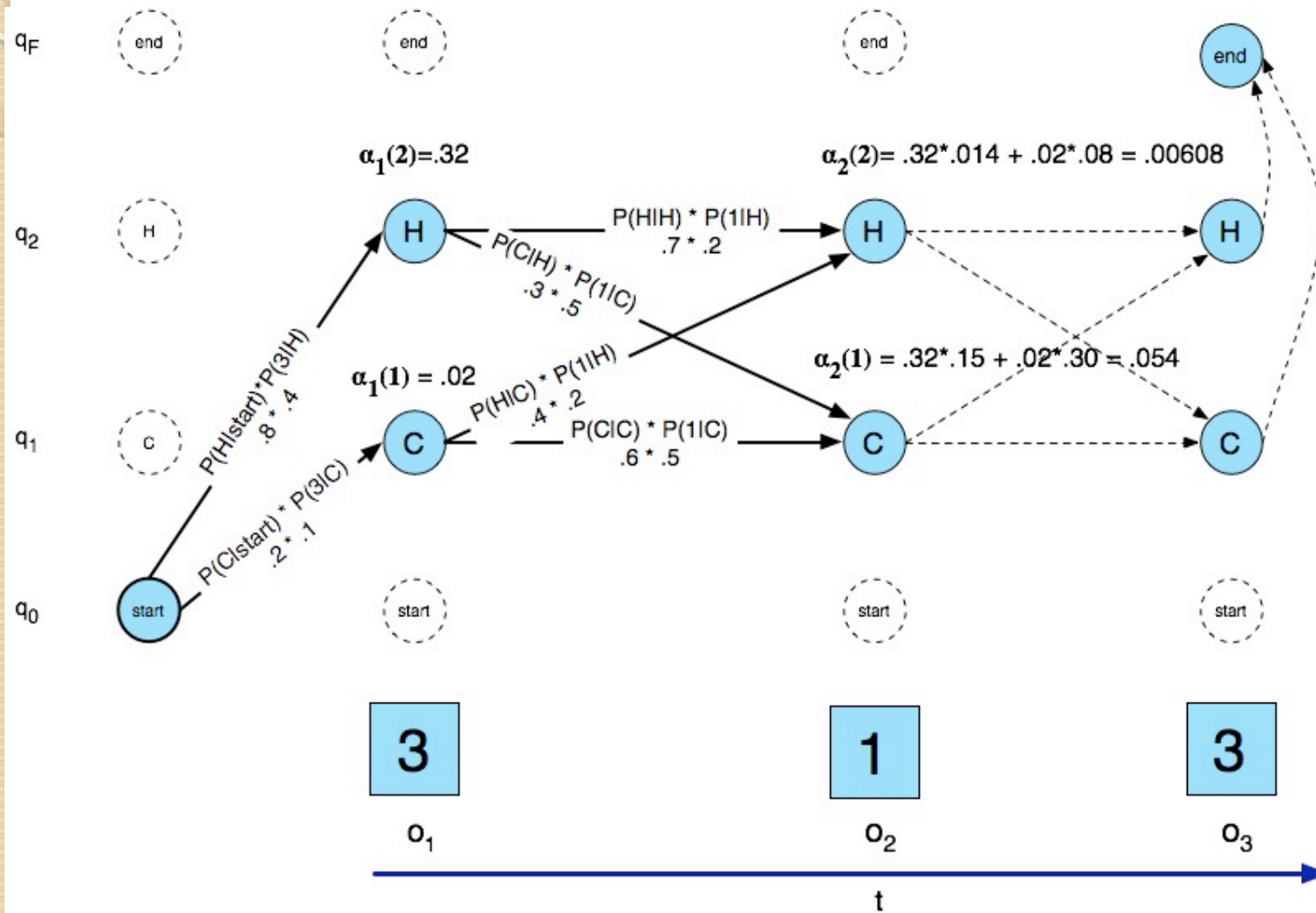
Computing total likelihood of 3 | 3

- We would need to sum over
 - Hot hot cold
 - Hot hot hot
 - Hot cold hot
 -
- How many possible hidden state sequences are there for this sequence?
- How about in general for an HMM with N hidden states and a sequence of T observations?
 - N^T
- So we can't just do separate computation for each hidden state sequence.

Instead: the Forward algorithm

- A kind of **dynamic programming** algorithm
 - Uses a table to store intermediate values
- Idea:
 - Compute the likelihood of the observation sequence
 - By summing over all possible hidden state sequences
 - But doing this efficiently
 - By folding all the sequences into a single **trellis**

The Forward Trellis



The forward algorithm

- Each cell of the forward algorithm trellis $\alpha_t(j)$
 - Represents the probability of being in state j
 - After seeing the first t observations

$$\alpha_t(j) = P(o_1, o_2 \dots o_t, q_t = j | \lambda)$$

probability

We update each cell

$\alpha_{t-1}(i)$

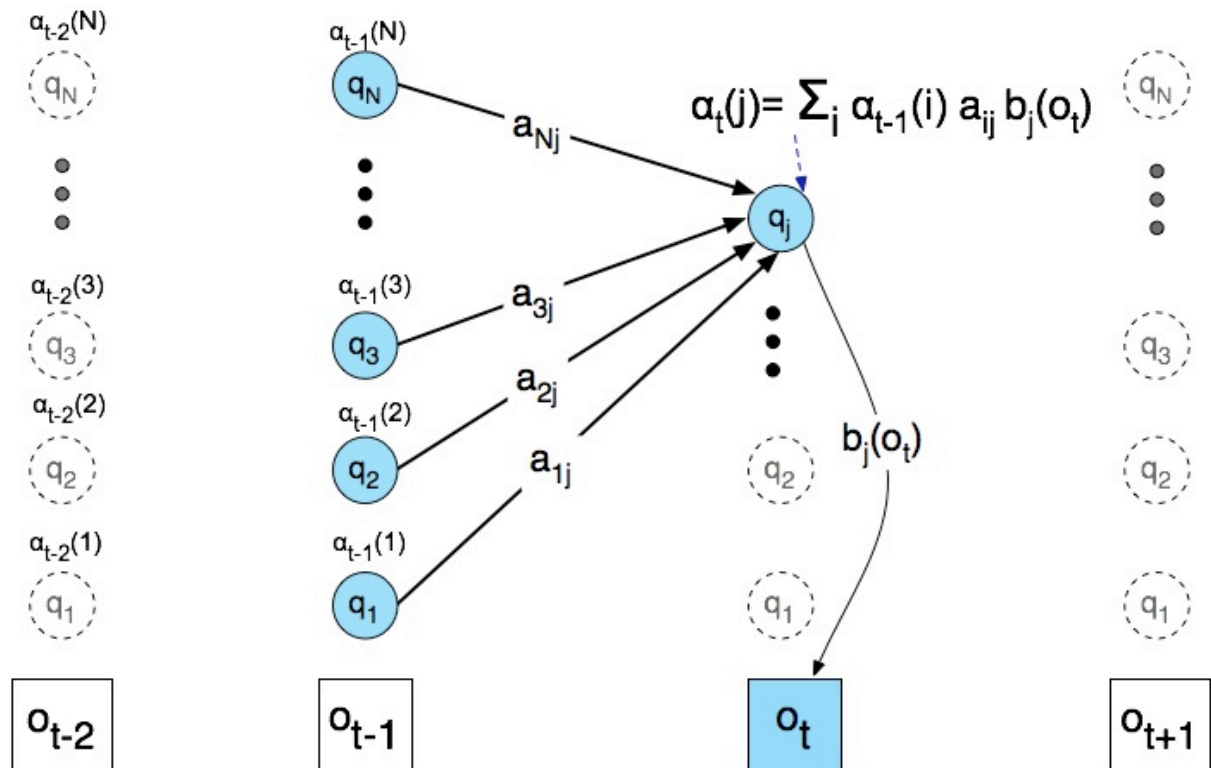
the **previous forward path probability** from the previous time step

a_{ij}

the **transition probability** from previous state q_i to current state q_j

$b_j(o_t)$

the **state observation likelihood** of the observation symbol o_t given the current state j





The Forward Recursion


1. Initialization:

$$\alpha_1(j) = a_{0j}b_j(o_1) \quad 1 \leq j \leq N$$

2. Recursion (since states 0 and F are non-emitting):

$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i)a_{ij}b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T$$

3. Termination:

$$P(O|\lambda) = \alpha_T(q_F) = \sum_{i=1}^N \alpha_T(i)a_{iF}$$


The Forward Algorithm

function FORWARD(*observations* of len T , *state-graph* of len N) **returns** *forward-prob*

create a probability matrix $forward[N+2, T]$

for each state s **from** 1 **to** N **do** ; initialization step

$forward[s, 1] \leftarrow a_{0,s} * b_s(o_1)$

for each time step t **from** 2 **to** T **do** ; recursion step

for each state s **from** 1 **to** N **do**

$forward[s, t] \leftarrow \sum_{s'=1}^N forward[s', t-1] * a_{s',s} * b_s(o_t)$

$forward[q_F, T] \leftarrow \sum_{s=1}^N forward[s, T] * a_{s,q_F}$; termination step

return $forward[q_F, T]$

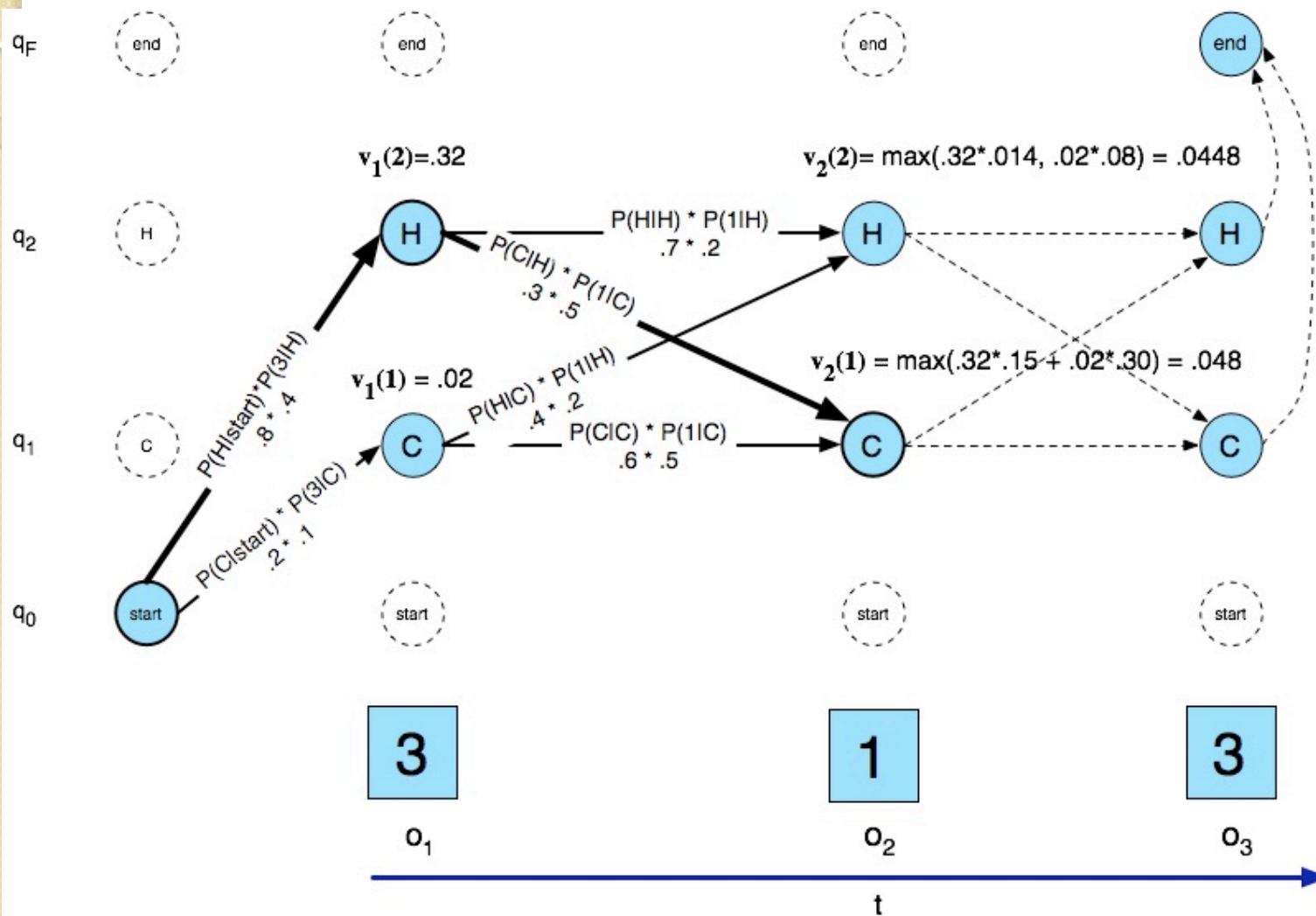
Decoding

- Given an observation sequence
 - 3 | 3
- And an HMM
- The task of the **decoder**
 - To find the best **hidden** state sequence
- Given the observation sequence $O=(o_1 o_2 \dots o_T)$, and an HMM model $\Phi = (A,B)$, **how do we choose a corresponding state sequence $Q=(q_1 q_2 \dots q_T)$** that is optimal in some sense (i.e., best explains the observations)

Decoding

- One possibility:
 - For each hidden state sequence
 - HHH, HHC, HCH,
 - Run the forward algorithm to compute $P(\Phi | O)$
- Why not?
 - N^T
- Instead:
 - The Viterbi algorithm
 - Is again a **dynamic programming** algorithm
 - Uses a similar trellis to the Forward algorithm

The Viterbi trellis



Viterbi intuition

- Process observation sequence left to right
- Filling out the trellis

$$v_t(i) = P(q_0, q_1 \dots q_{t-1}, o_1, o_2 \dots o_t, q_t = i | \lambda)$$

$$v_t(j) = \max_{i=1}^N v_{t-1}(i) a_{ij} b_j(o_t)$$

$v_{t-1}(i)$	the previous Viterbi path probability from the previous time step
a_{ij}	the transition probability from previous state q_i to current state q_j
$b_j(o_t)$	the state observation likelihood of the observation symbol o_t given the current state j

Viterbi Algorithm

function VITERBI(*observations* of len T , *state-graph* of len N) **returns** *best-path*

create a path probability matrix $viterbi[N+2, T]$

for each state s **from** 1 **to** N **do** ; initialization step

$viterbi[s, 1] \leftarrow a_{0,s} * b_s(o_1)$

$backpointer[s, 1] \leftarrow 0$

for each time step t **from** 2 **to** T **do** ; recursion step

for each state s **from** 1 **to** N **do**

$viterbi[s, t] \leftarrow \max_{s'=1}^N viterbi[s', t-1] * a_{s',s} * b_s(o_t)$

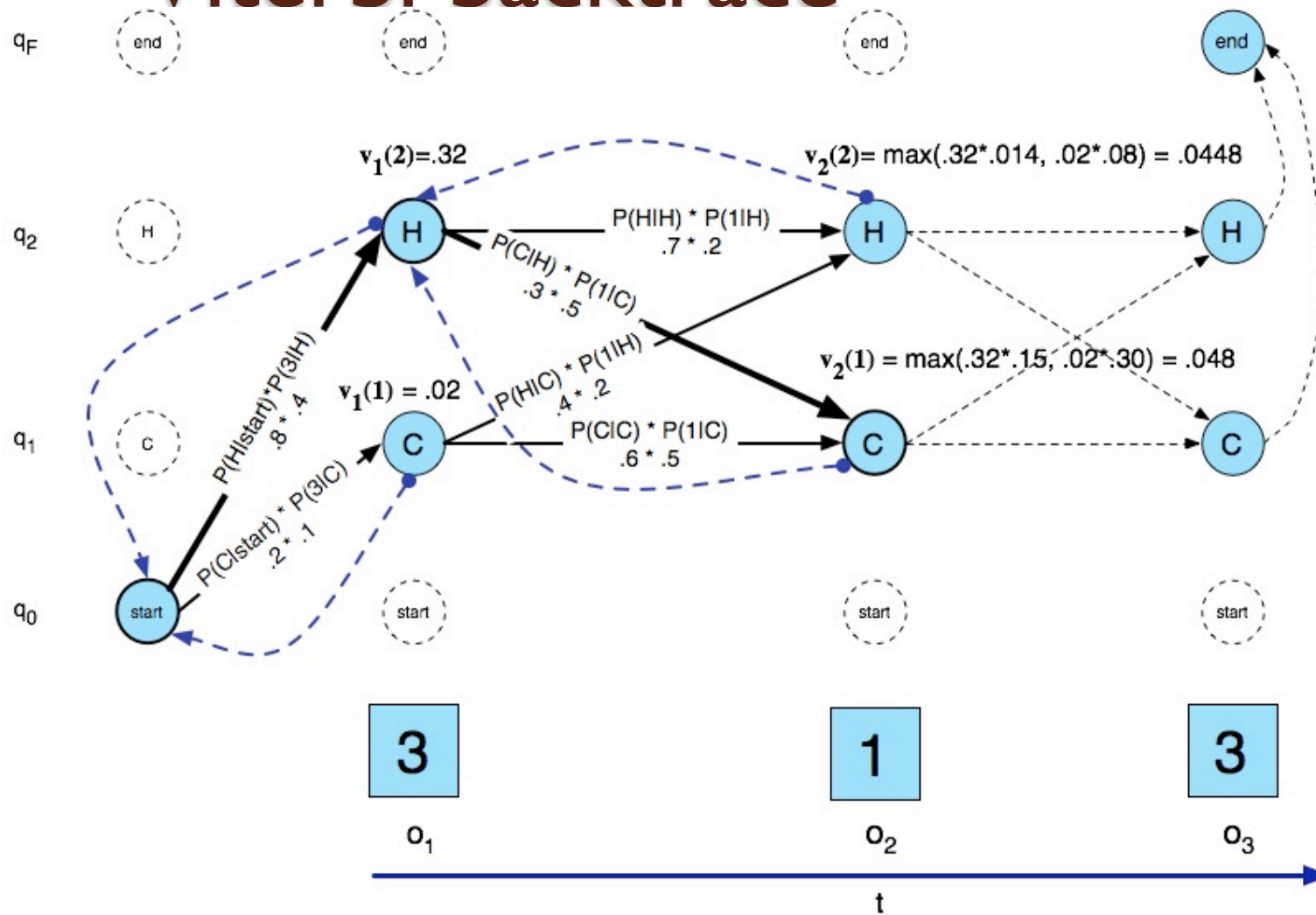
$backpointer[s, t] \leftarrow \operatorname{argmax}_{s'=1}^N viterbi[s', t-1] * a_{s',s}$

$viterbi[q_F, T] \leftarrow \max_{s=1}^N viterbi[s, T] * a_{s,q_F}$; termination step

$backpointer[q_F, T] \leftarrow \operatorname{argmax}_{s=1}^N viterbi[s, T] * a_{s,q_F}$; termination step

return the backtrace path by following backpointers to states back in time from $backpointer[q_F, T]$

Viterbi backtrace



Viterbi Recursion

1. Initialization:

$$\begin{aligned}v_1(j) &= a_{0j}b_j(o_1) \quad 1 \leq j \leq N \\bt_1(j) &= 0\end{aligned}$$

2. Recursion (recall states 0 and q_F are non-emitting):

$$\begin{aligned}v_t(j) &= \max_{i=1}^N v_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T \\bt_t(j) &= \operatorname{argmax}_{i=1}^N v_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T\end{aligned}$$

3. Termination:

$$\text{The best score: } P^* = v_T(q_F) = \max_{i=1}^N v_T(i) * a_{i,F}$$

$$\text{The start of backtrace: } q_T^* = bt_T(q_F) = \operatorname{argmax}_{i=1}^N v_T(i) * a_{i,F}$$

Why “Dynamic Programming”

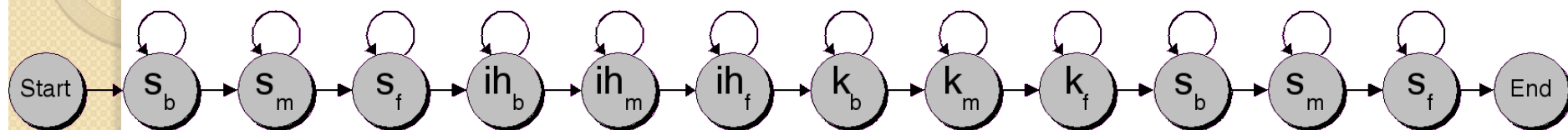
“I spent the Fall quarter (of 1950) at RAND. My first task was to find a name for multistage decision processes. An interesting question is, Where did the name, dynamic programming, come from? The 1950s were not good years for mathematical research. We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word, research. I’ m not using the term lightly; I’ m using it precisely. His face would suffuse, he would turn red, and he would get violent if people used the term, research, in his presence. You can imagine how he felt, then, about the term, mathematical. The RAND Corporation was employed by the Air Force, and the Air Force had Wilson as its boss, essentially. Hence, I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporation. What title, what name, could I choose? In the first place I was interested in planning, in decision making, in thinking. But planning, is not a good word for various reasons. I decided therefore to use the word, “programming” I wanted to get across the idea that this was dynamic, this was multistage, this was time-varying I thought, lets kill two birds with one stone. Lets take a word that has an absolutely precise meaning, namely dynamic, in the classical physical sense. It also has a very interesting property as an adjective, and that is its impossible to use the word, dynamic, in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. Its impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities.” Richard Bellman, “Eye of the Hurrican: an autobiography” 1984.

Thanks to Chen, Picheny, Eide, Nock

HMMs for Speech

- We haven't yet shown how to learn the A and B matrices for HMMs; we'll do that later today or possibly on Monday
- But let's return to think about speech

Reminder: a word looks like this:



$$Q = q_1 q_2 \dots q_N$$

$$A = a_{01} a_{02} \dots a_{n1} \dots a_{nn}$$

$$B = b_i(o_t)$$

a set of **states** corresponding to **subphones**

a **transition probability matrix** A , each a_{ij} representing the probability for each subphone of taking a **self-loop** or going to the next subphone. Together, Q and A implement a **pronunciation lexicon**, an HMM state graph structure for each word that the system is capable of recognizing.

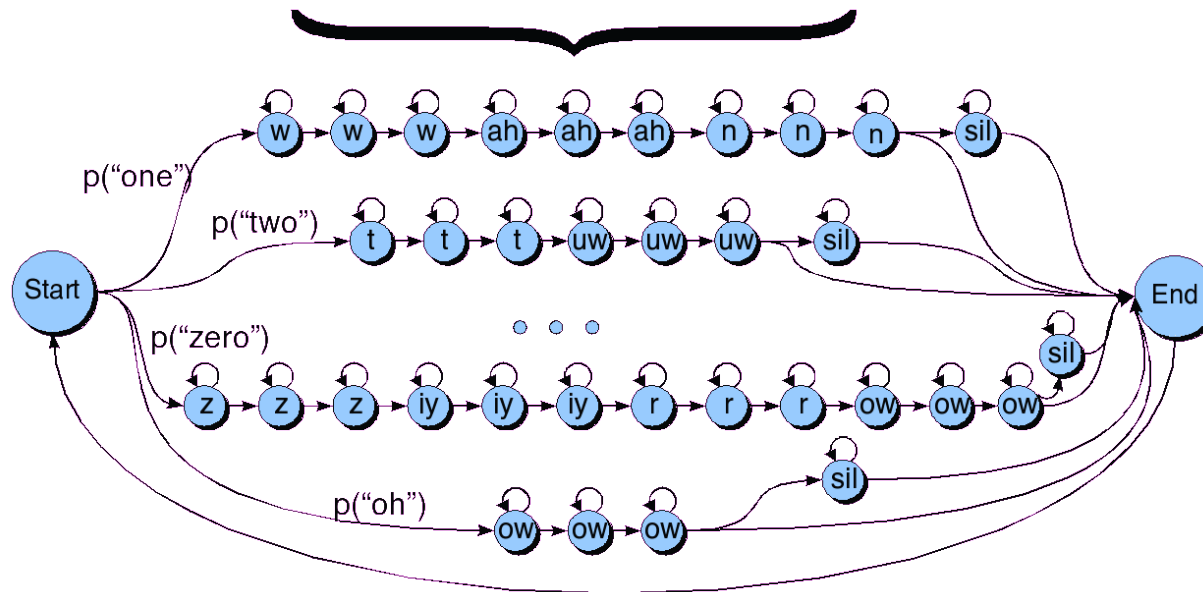
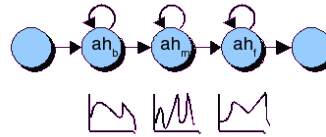
A set of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of a cepstral feature vector (observation o_t) being generated from subphone state i .

HMM for digit recognition task

Lexicon

one	w ah n
two	t uw
three	th r iy
four	f ao r
five	f ay v
six	s ih k s
seven	s eh v ax n
eight	ey t
nine	n ay n
zero	z iy r ow
oh	ow

Phone HMM



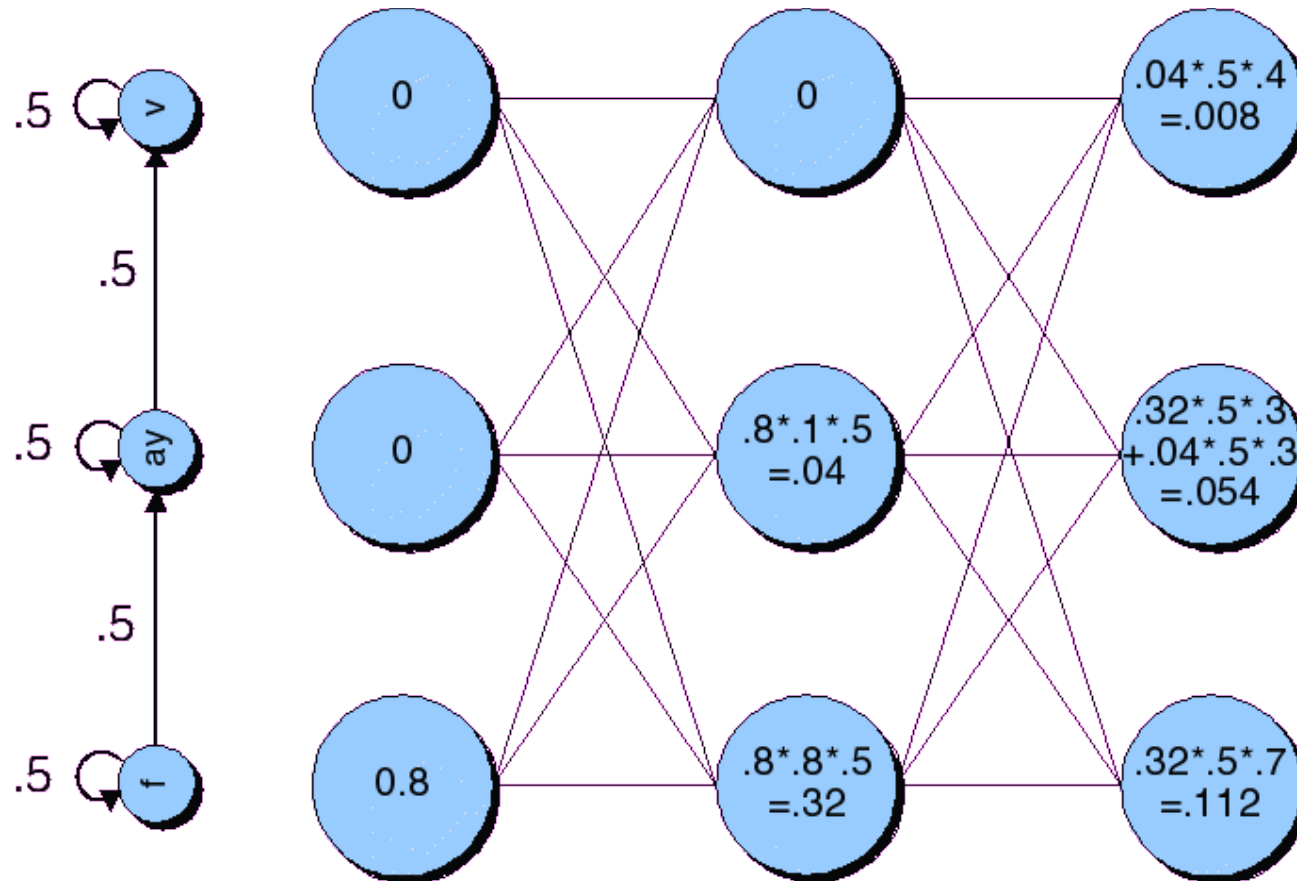
The Evaluation (forward) problem for speech

- The observation sequence O is a series of MFCC vectors
- The hidden states W are the phones and words
- For a given phone/word string W , our job is to evaluate $P(O|W)$
- Intuition: how likely is the input to have been generated by just that word string W

Evaluation for speech: Summing over all different paths!

- f ay ay ay ay v v v v
- f f ay ay ay ay v v v
- f f f f ay ay ay ay v
- f f ay ay ay ay ay ay v
- f f ay ay ay ay ay ay ay ay v
- f f ay v v v v v v v

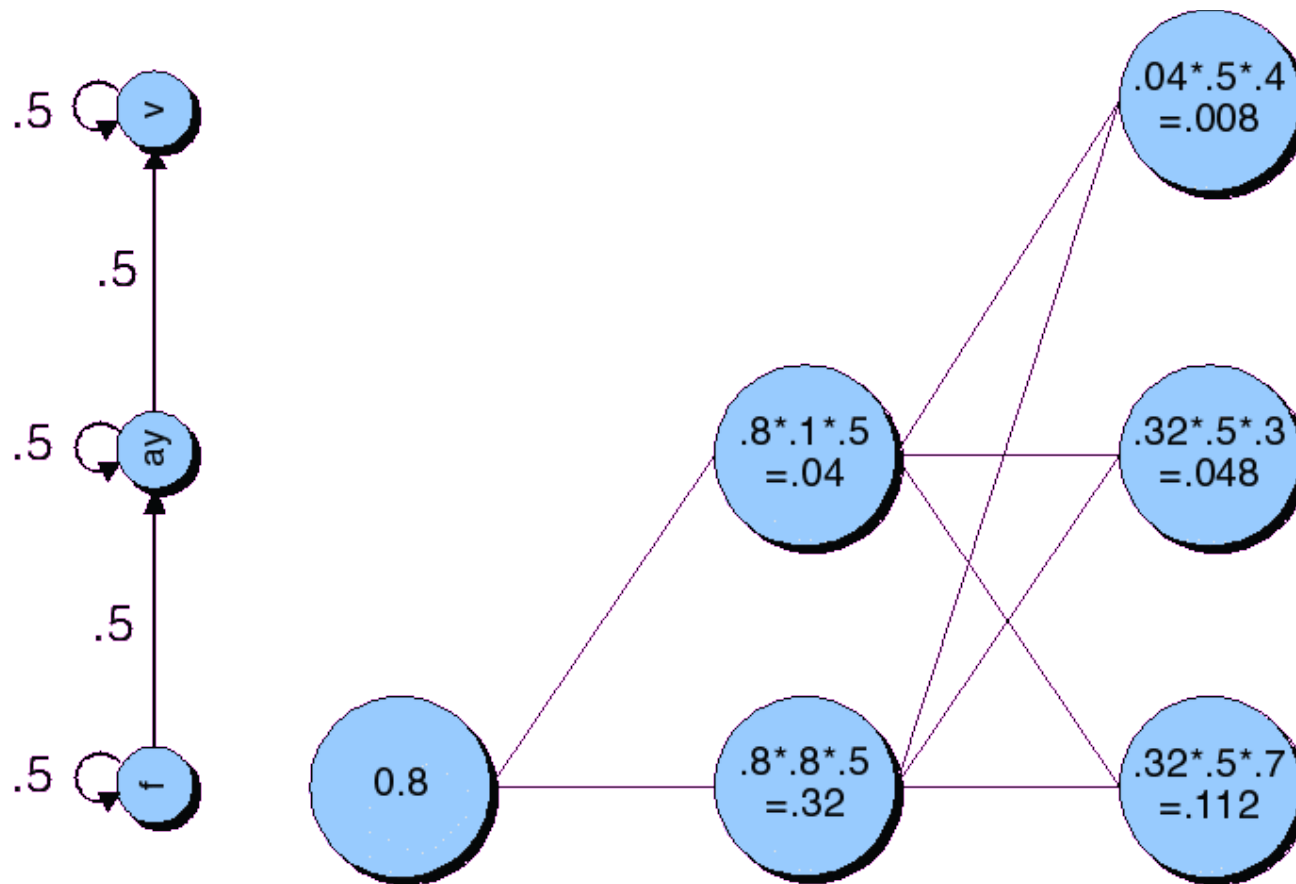
The forward lattice for “five”



The forward trellis for “five”

V	0	0	0.008	0.0093	0.0114	0.00703	0.00345	0.00306	0.00206	0.00117
AY	0	0.04	0.054	0.0664	0.0355	0.016	0.00676	0.00208	0.000532	0.000109
F	0.8	0.32	0.112	0.0224	0.00448	0.000896	0.000179	4.48e-05	1.12e-05	2.8e-06
Time	1	2	3	4	5	6	7	8	9	10
B	<i>f</i> 0.8	<i>f</i> 0.8	<i>f</i> 0.7	<i>f</i> 0.4	<i>f</i> 0.4	<i>f</i> 0.4	<i>f</i> 0.4	<i>f</i> 0.5	<i>f</i> 0.5	<i>f</i> 0.5
	<i>ay</i> 0.1	<i>ay</i> 0.1	<i>ay</i> 0.3	<i>ay</i> 0.8	<i>ay</i> 0.8	<i>ay</i> 0.8	<i>ay</i> 0.8	<i>ay</i> 0.6	<i>ay</i> 0.5	<i>ay</i> 0.4
	<i>v</i> 0.6	<i>v</i> 0.6	<i>v</i> 0.4	<i>v</i> 0.3	<i>v</i> 0.3	<i>v</i> 0.3	<i>v</i> 0.3	<i>v</i> 0.6	<i>v</i> 0.8	<i>v</i> 0.9
	<i>p</i> 0.4	<i>p</i> 0.4	<i>p</i> 0.2	<i>p</i> 0.1	<i>p</i> 0.1	<i>p</i> 0.1	<i>p</i> 0.1	<i>p</i> 0.1	<i>p</i> 0.3	<i>p</i> 0.3
	<i>iy</i> 0.1	<i>iy</i> 0.1	<i>iy</i> 0.3	<i>iy</i> 0.6	<i>iy</i> 0.6	<i>iy</i> 0.6	<i>iy</i> 0.6	<i>iy</i> 0.5	<i>iy</i> 0.5	<i>iy</i> 0.4

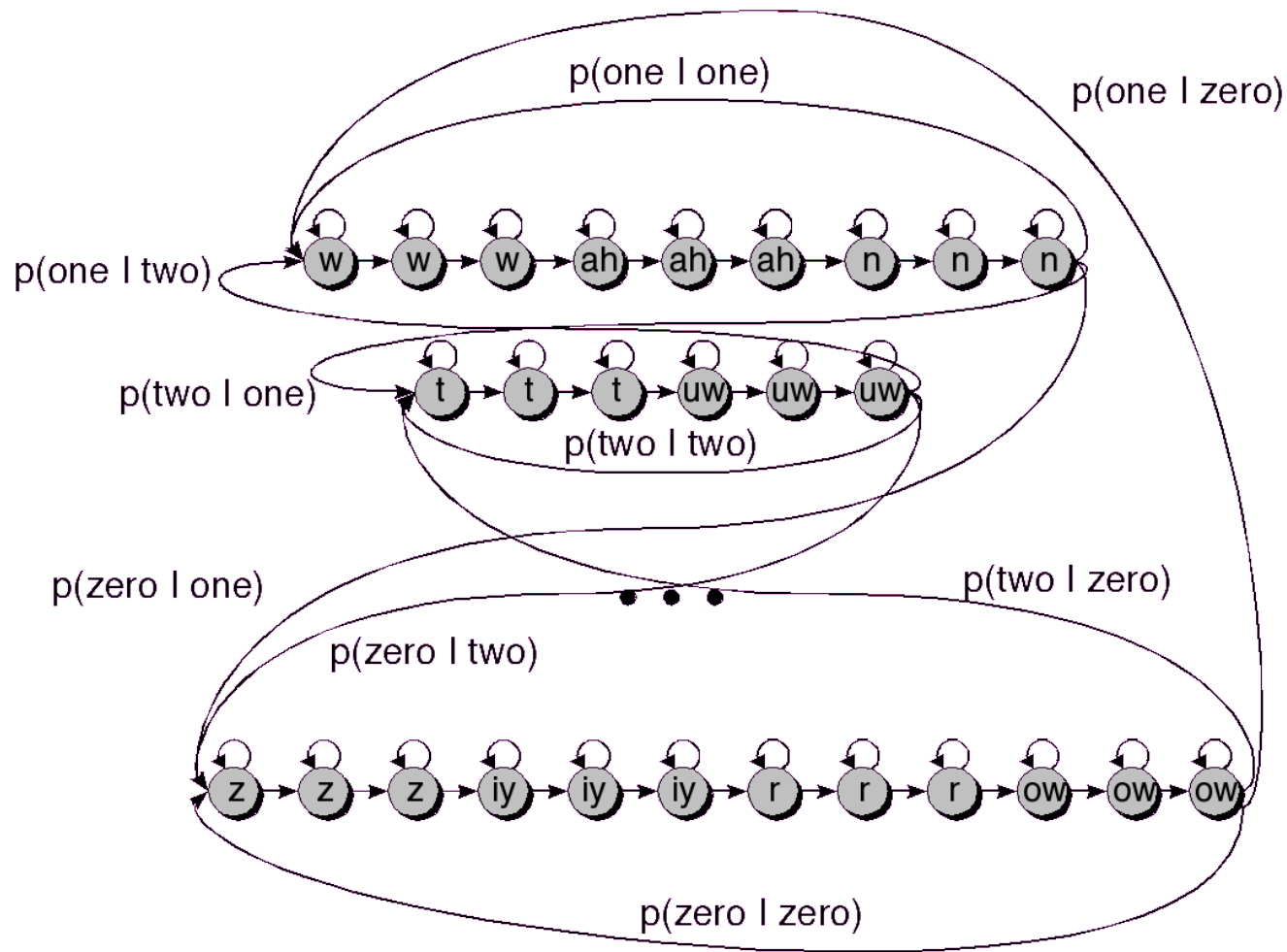
Viterbi trellis for “five”



Viterbi trellis for “five”

V	0	0	0.008	0.0072	0.00672	0.00403	0.00188	0.00161	0.000667	0.000493
AY	0	0.04	0.048	0.0448	0.0269	0.0125	0.00538	0.00167	0.000428	8.78e-05
F	0.8	0.32	0.112	0.0224	0.00448	0.000896	0.000179	4.48e-05	1.12e-05	2.8e-06
Time	1	2	3	4	5	6	7	8	9	10
B	<i>f</i> 0.8	<i>f</i> 0.8	<i>f</i> 0.7	<i>f</i> 0.4	<i>f</i> 0.4	<i>f</i> 0.4	<i>f</i> 0.4	<i>f</i> 0.5	<i>f</i> 0.5	<i>f</i> 0.5
	<i>ay</i> 0.1	<i>ay</i> 0.1	<i>ay</i> 0.3	<i>ay</i> 0.8	<i>ay</i> 0.8	<i>ay</i> 0.8	<i>ay</i> 0.8	<i>ay</i> 0.6	<i>ay</i> 0.5	<i>ay</i> 0.4
	<i>v</i> 0.6	<i>v</i> 0.6	<i>v</i> 0.4	<i>v</i> 0.3	<i>v</i> 0.3	<i>v</i> 0.3	<i>v</i> 0.3	<i>v</i> 0.6	<i>v</i> 0.8	<i>v</i> 0.9
	<i>p</i> 0.4	<i>p</i> 0.4	<i>p</i> 0.2	<i>p</i> 0.1	<i>p</i> 0.1	<i>p</i> 0.1	<i>p</i> 0.1	<i>p</i> 0.1	<i>p</i> 0.3	<i>p</i> 0.3
	<i>iy</i> 0.1	<i>iy</i> 0.1	<i>iy</i> 0.3	<i>iy</i> 0.6	<i>iy</i> 0.6	<i>iy</i> 0.6	<i>iy</i> 0.6	<i>iy</i> 0.5	<i>iy</i> 0.5	<i>iy</i> 0.4

Search space with bigrams



Viterbi backtrace

