Lecture 10: Introduction to HMM (hidden Markov model)

DEEE725 음성신호처리실습

Instructor: 장길진

Original slides from:

Philip Jackson, Centre for Vision Speech & Signal Processing, University of Surrey

http://www.ee.surrey.ac.uk/Personal/P.Jackson/ISSPR/

Topics

- Markov model definition
- Hidden Markov model (HMM)
- Viterbi decoding
- Training HMM by segmental k-means
- Practical issues

Markov Models

- Set of states:

$$\{s_1, s_2, \ldots, s_N\}$$

- Process moves from one state to another generating a sequence of states :

$$s_{i1}, s_{i2}, \ldots, s_{ik}, \ldots$$

- Markov chain property:
  - probability of each subsequent state depends only on what was the previous state

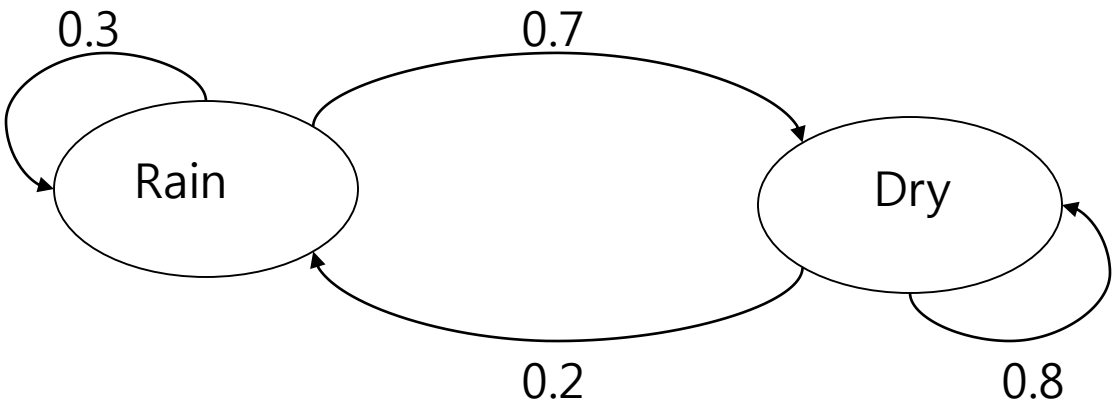
$$P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) \approx P(s_{ik} \mid s_{ik-1})$$

- To define Markov model, the following probabilities have to be specified:
  - transition probabilities
  - initial probabilities

$$a_{ij} = P(s_i \mid s_j)$$

$$\pi_i = P(s_i)$$

Example of Markov Model



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

Calculation of sequence probability

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

$$P(s_{i1}, s_{i2}, \ldots, s_{ik}) = P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) P(s_{i1}, s_{i2}, \ldots, s_{ik-1})$$
$$= P(s_{ik} \mid s_{ik-1}) P(s_{i1}, s_{i2}, \ldots, s_{ik-1}) = \ldots$$
$$= P(s_{ik} \mid s_{ik-1}) P(s_{ik-1} \mid s_{ik-2}) \ldots P(s_{i2} \mid s_{i1}) P(s_{i1})$$

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

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

What if some information is HIDDEN?

- Markov model assumes that
  - We know the current STATUS of the phenomena that we are interested in, so that the exact probability is calculated
- What if we are not aware of the current STATUS?
  - The status becomes HIDDEN (latent)
  - Can we infer it?

HIDDEN MARKOV MODELS

Hidden Markov Models

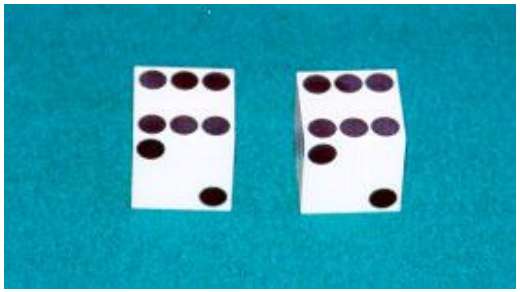
- Assumptions:
  - **Observed** variables (**emitted** symbols)
    - We are only given limited information
  - Hidden variables
    - Current status of a system
  - Relationships between them
    - Represented by a graph with transition probabilities

- **Goal:** Find the most likely explanation of a system given the observed variables only

The occasionally dishonest casino

- Suppose that you are playing a game in a casino
  - A dealer rolls a dice, and you bet on a number from your GUESS
  - Unfortunately, the dices are not always fair
    - Sometimes the dealer uses fair dice, and changes it with a LOADED dice which has different probabilities of the numbers
    - Because guests are watching the dealer, changing the dice occurs very rarely
- Your goal is to make guesses on the future number as exactly as possible from the history of the numbers only, keeping in mind that the dealer can change dice without being noticed

Example: The dishonest casino



A casino has two dice:

- Fair die

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

- Loaded die

  P(1) = P(2) = P(3) = P(4) = P(5) = 1/10

  P(6) = 1/2

Casino player switches between fair and loaded die with probability 1/20 at each turn

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

dishonest casino: assumptions

- **Case 1: assume it is known that**
  - **The number of dices**
  - **How LOADED the dices are: probabilities of 6 different numbers**
  - **How often the dices are changed, but cannot detect the exact moment of the change**
- Case 2:
  - Only the number of dices are known
- Case 3:
  - No information except the observed numbers
- Common assumption:
  - Transitions between dices obey a **first-order Markov process**, i.e. depending on the previous choice of the dices

The occasionally dishonest casino

- How LOADED: already knows the probabilities
  - Fair die: Prob(1) = Prob(2) = . . . = Prob(6) = 1/6
  - Loaded die: Prob(1) = Prob(2) = . . . = Prob(5) = 1/10, Prob(6) = ½
  - These are called the ***emission*** probabilities

- How OFTEN the change occurs: a casino uses a fair die most of the time, but occasionally switches to a loaded one
  - Prob(Fair $\rightarrow$ Loaded) = 0.01
  - Prob(Loaded $\rightarrow$ Fair) = 0.2
  - Called ***transition probabilities*** (1st order Markovian)

Question # 1: Decoding

GIVEN

A sequence of rolls by the casino player

QUESTION

What portion of the sequence was generated with the fair die, and what portion with the loaded die?

This is the **DECODING** question in HMMs

Question # 2: Evaluation

GIVEN

A sequence of rolls by the casino player

12455264621461461361366616646616366163661636165156151151461235623 44

Prob = 1.3×10^{-35}

QUESTION

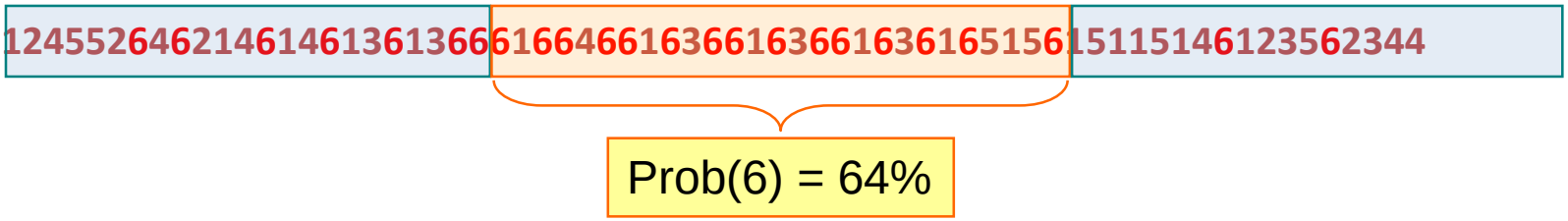
How likely is this sequence, given our model of how the casino works?

This is the **EVALUATION** problem in HMMs

Question # 3: Learning

GIVEN

A sequence of rolls by the casino player

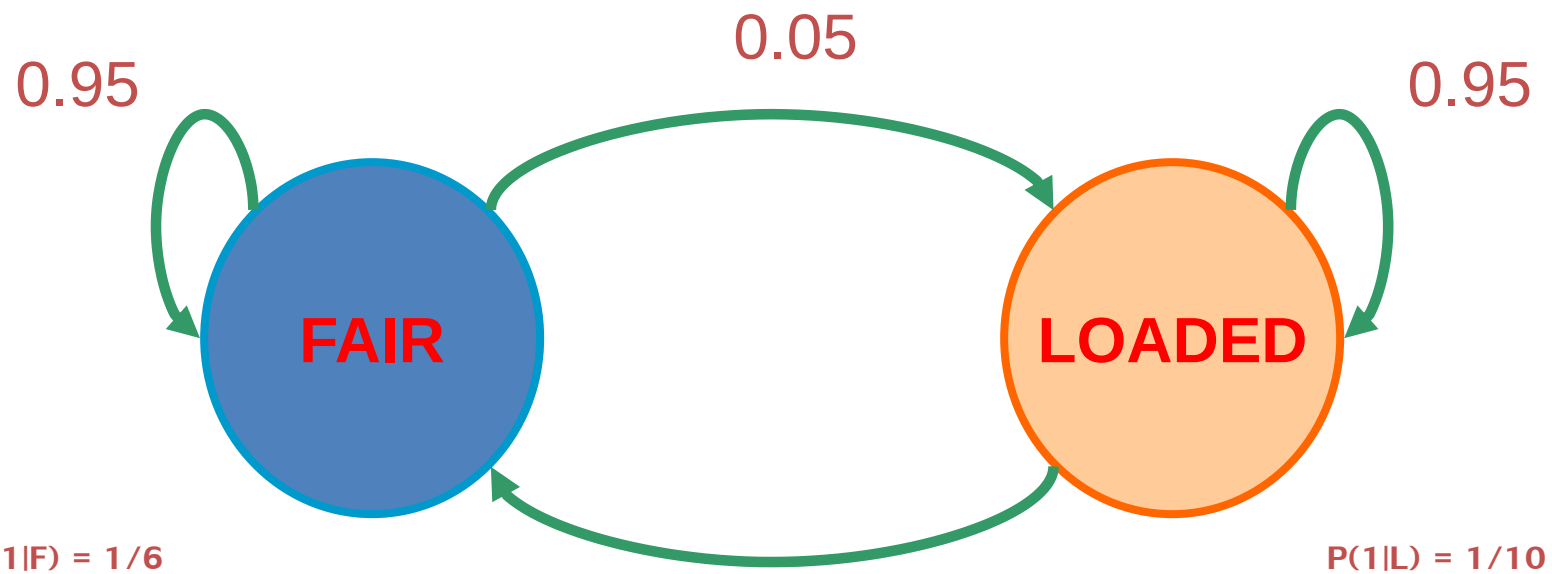


QUESTION

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 in HMMs

The dishonest casino model



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

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

The occasionally dishonest casino

- Known:
  - The observation probabilities
  - The transition probabilities
- Hidden: What the casino did
  - FFFFFLLLLLLLFFFF…
- Observable: The series of die tosses
  - 3415256664666153…
- What we must infer:
  - When was a fair die used?
  - When was a loaded one used?
    - The answer is a sequence FFFFFFFLLLLLLFFF…

Making the inference

- Model assigns a probability to each explanation of the observation:

  $P(326|FFL)$
  $= P(3|F) \cdot P(F \rightarrow F) \cdot P(2|F) \cdot P(F \rightarrow L) \cdot P(6|L)$
  $= 1/6 \cdot 0.99 \cdot 1/6 \cdot 0.01 \cdot ½$

- **Maximum Likelihood:** Determine which explanation is most likely
  - Find the path *most likely* to have produced the observed sequence
- **Total probability:** Determine probability that observed sequence was produced by the HMM
  - Consider *all* paths that could have produced the observed sequence

Notation

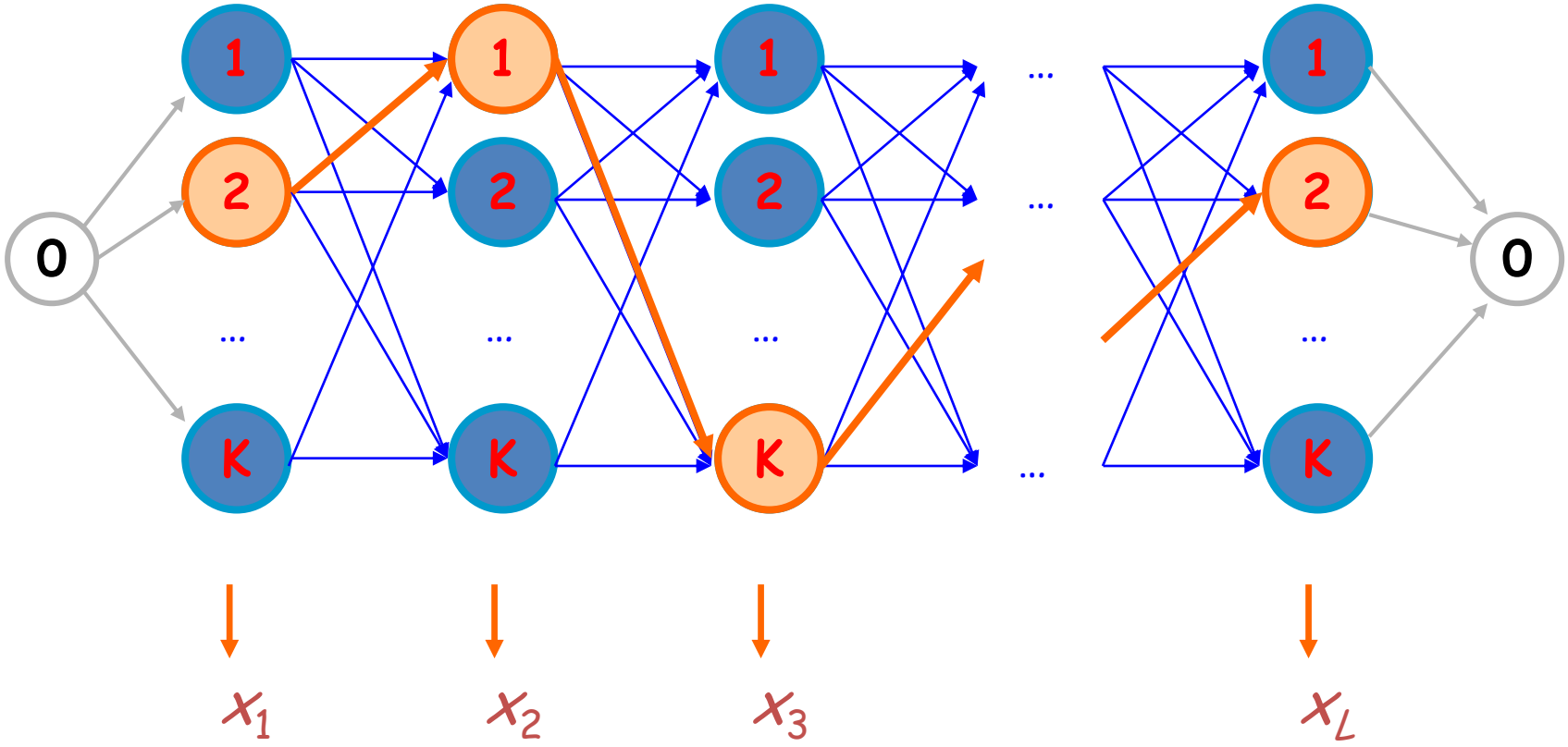
- x is the sequence of symbols emitted by model
  - x_i is the symbol emitted at time i
- A ***path***, π, is a sequence of states
  - The i-th state in π is π_i
- a_{kr} is the probability of making a transition from state k to state r:

$$a_{kr} = \Pr(\pi_i = r \mid \pi_{i-1} = k)$$

- $e_k(b)$ is the probability that symbol b is emitted when in state k

$$e_k(b) = \Pr(x_i = b \mid \pi_i = k)$$

A PARSE of a sequence



$$\Pr(x,\pi) = a_{0\pi_1} \prod_{i=1}^{L} e_{\pi_i}(x_i) \cdot a_{\pi_i \pi_{i+1}}$$

The occasionally dishonest casino

$$x = \langle x_1, x_2, x_3 \rangle = \langle 6,2,6 \rangle$$

$$\pi^{(1)} = FFF$$

$$\begin{aligned}\Pr(x, \pi^{(1)}) &= a_{0F} e_F(6) a_{FF} e_F(2) a_{FF} e_F(6) \\ &= 0.5 \times \frac{1}{6} \times 0.99 \times \frac{1}{6} \times 0.99 \times \frac{1}{6} \\ &\approx 0.00227\end{aligned}$$

$$\pi^{(2)} = LLL$$

$$\begin{aligned}\Pr(x, \pi^{(2)}) &= a_{0L} e_L(6) a_{LL} e_L(2) a_{LL} e_L(6) \\ &= 0.5 \times 0.5 \times 0.8 \times 0.1 \times 0.8 \times 0.5 \\ &= 0.008\end{aligned}$$

$$\pi^{(3)} = LFL$$

$$\begin{aligned}\Pr(x, \pi^{(3)}) &= a_{0L} e_L(6) a_{LF} e_F(2) a_{FL} e_L(6) a_{L0} \\ &= 0.5 \times 0.5 \times 0.2 \times \frac{1}{6} \times 0.01 \times 0.5 \\ &\approx 0.0000417\end{aligned}$$

Application: predicting the next symbol

$$\pi^{(2)} = LLL$$

$$\begin{aligned}\Pr(x, \pi^{(2)}) &= a_{0L} e_L(6) a_{LL} e_L(2) a_{LL} e_L(6) \\ &= 0.5 \times 0.5 \times 0.8 \times 0.1 \times 0.8 \times 0.5 \\ &= 0.008\end{aligned}$$

$$\Pr(x_t \mid \pi_{t-1}^{(2)} = L) = a_{LL} e_L(x_t) + a_{LF} e_F(x_t)$$

$$\Pr(x_t \in \{1,\ldots,5\} \mid \pi_{t-1}^{(2)} = L) = 0.8 \cdot \frac{1}{10} + 0.2 \cdot \frac{1}{6} = 0.11333$$

$$\Pr(x_t = 6 \mid \pi_{t-1}^{(2)} = L) = 0.8 \cdot \frac{1}{2} + 0.2 \cdot \frac{1}{6} = 0.4333$$

$$x_t^* = \arg\max_{x_t} \Pr(x_t \mid \pi_{t-1}^{(2)} = L) = 6$$

Other cases

- Case 2: only the number of dices are known
  - Observation probabilities and transition probabilities should be TRAINED
  - Most speech recognition problems
- Case 3: no information
  - Infer the number of states: very hard
    - Based on experts' experience (speech recognition)
    - Cross validation
    - Statistical methods
  - Kalman filters: continuous state

HMM FOR SPEECH RECOGNITION

Formulation of Fundamental Equation

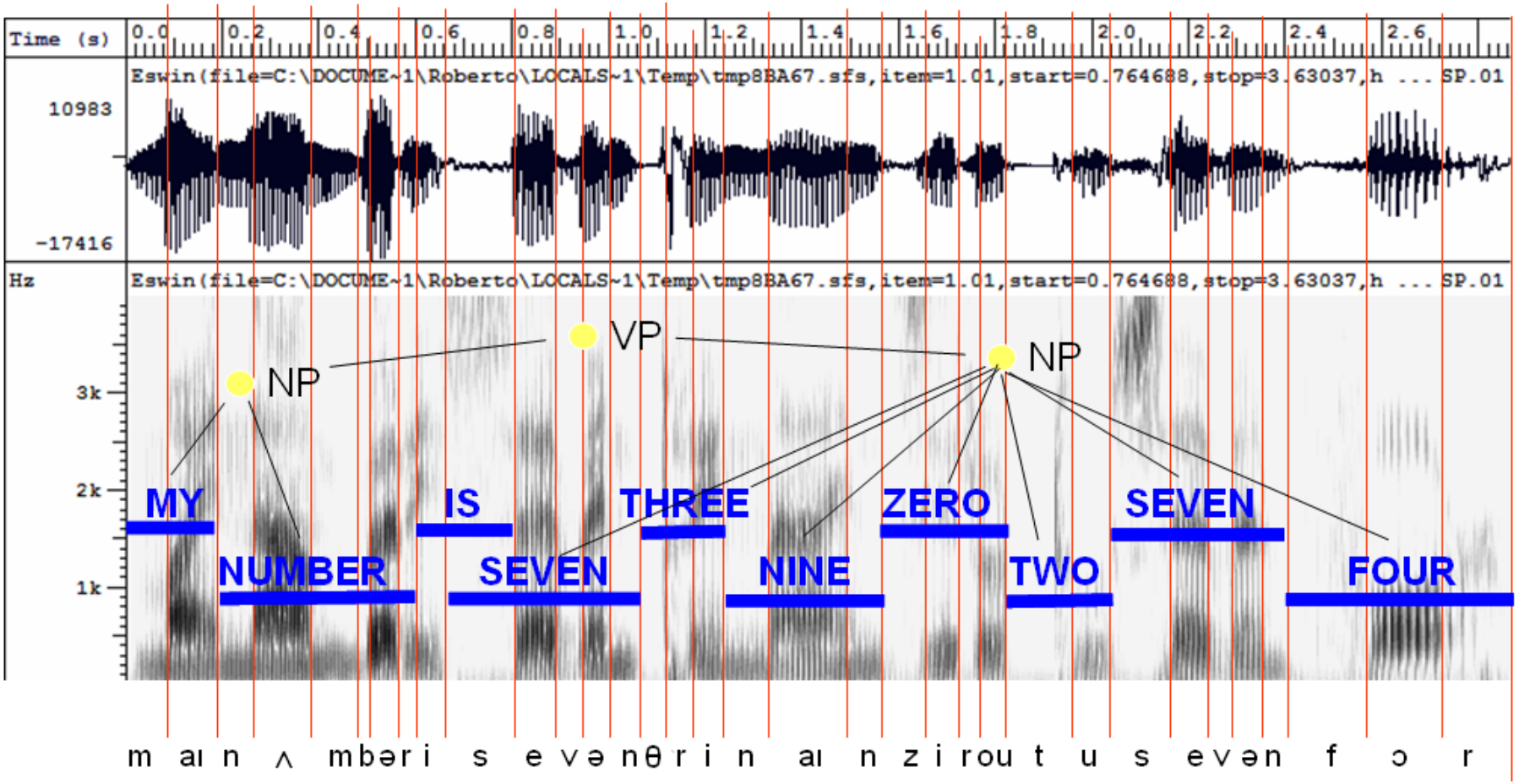
- **O** is an acoustical 'Observation'
  - Usually, MFCC matrix
- *w* is a 'word' we are trying to recognize
- Mathematical formulation

$$w^* = \arg\max_w P(w|\mathbf{O})$$

  - $P(w|\mathbf{O})$ is unknown so by Bayes' rule:

$$P(w|\mathbf{O}) = \frac{P(\mathbf{O}|w)P(w)}{P(\mathbf{O})}$$

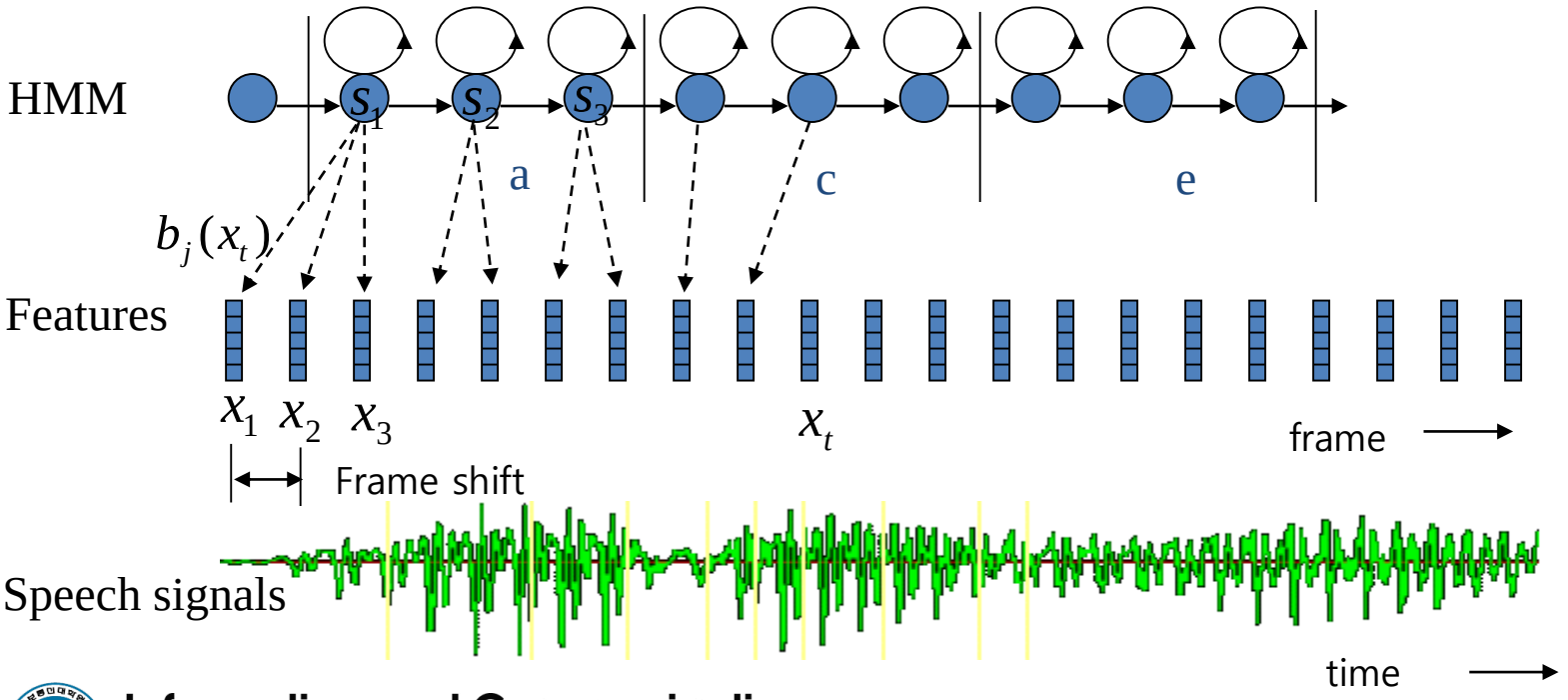
Results of Speech Recognition



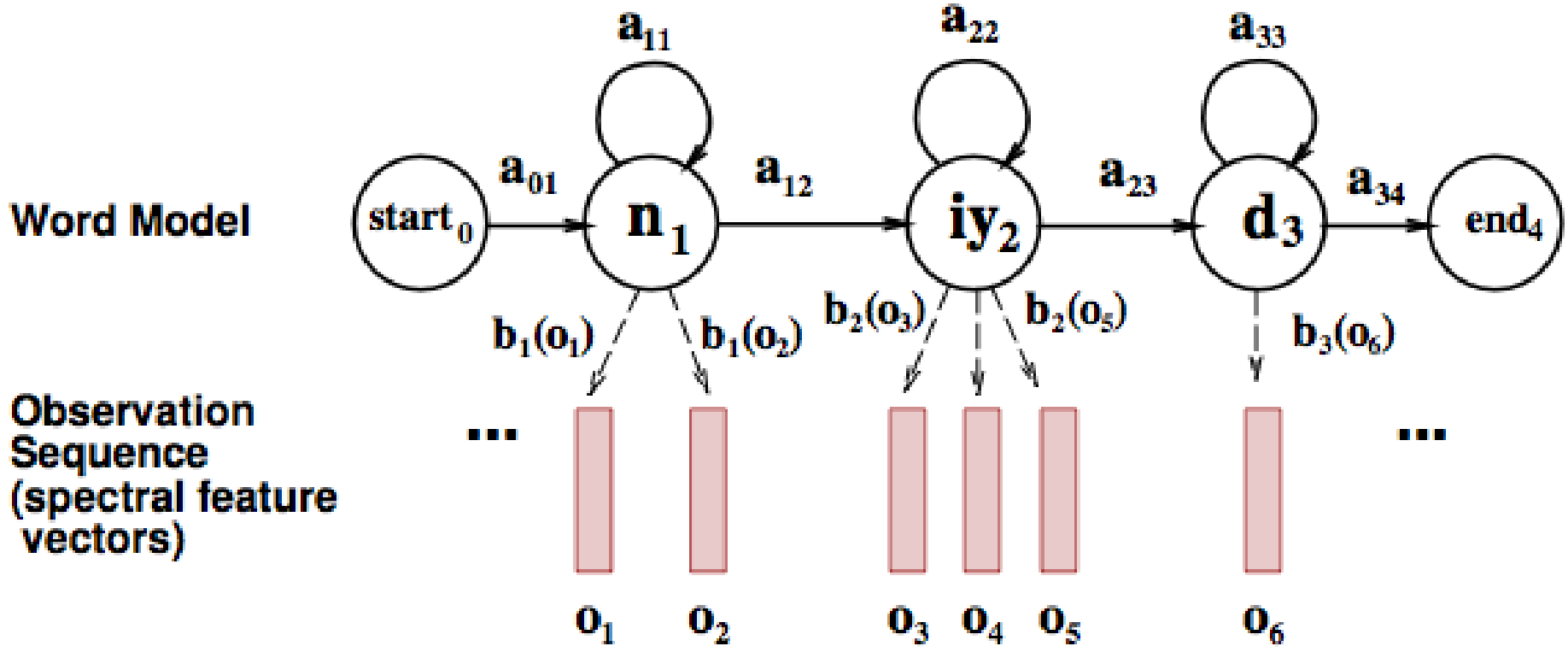
- Identify words / individual phonemes with varying length
- How to match both class label and timing info?

Application of HMM to Speech Recognition

- Assume speech signals are generated by HMM
- Find model parameters from training data
- Compute probability of test speech using the HMM and select the model having maximum likelihood
- Left-to-right model: easy to model signal whose properties change over time in a successive manner – e.g., speech

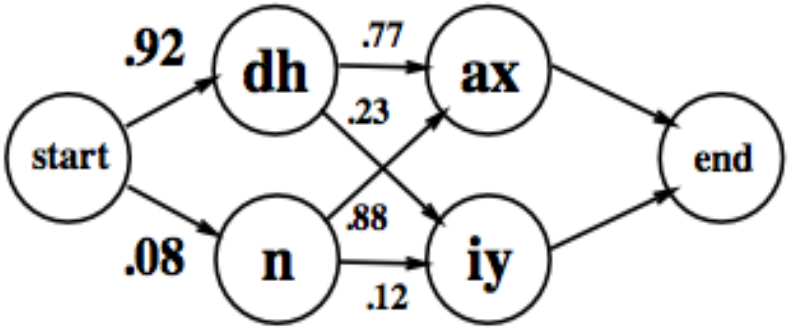


Word HMM – detailed view

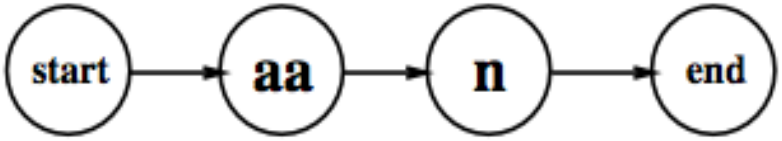


- Usually one phoneme is modeled by connected, two or three states to model the transition between one phoneme to the other

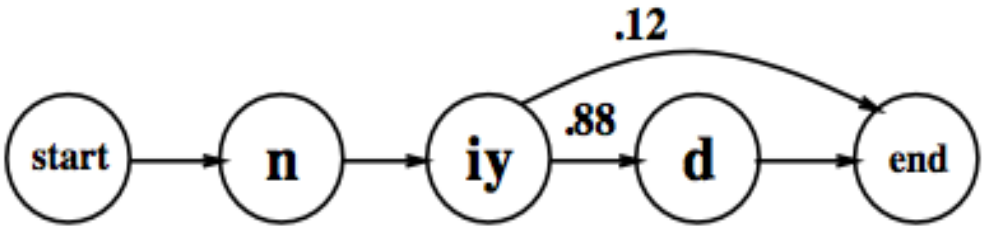
More Word Models



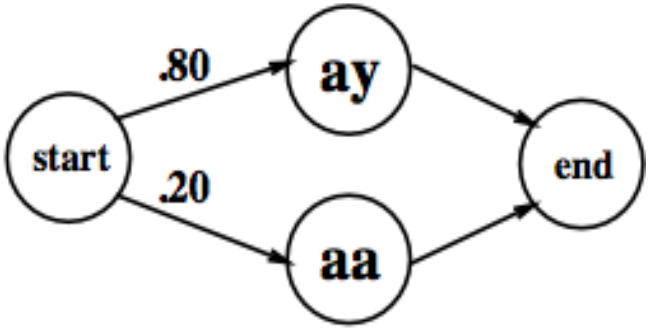
Word model for "the"



Word model for "on"



Word model for "need"



Word model for "I"

Elements of a discrete HMM

1. Number of different states N, $x \in \{1, \ldots, N\}$;

2. Number of different events K, $k \in \{1, \ldots, K\}$;

3. Initial-state probabilities,
   $\pi = \{\pi_i\} = \{P(x_1 = i)\}$ for $1 \leq i \leq N$;

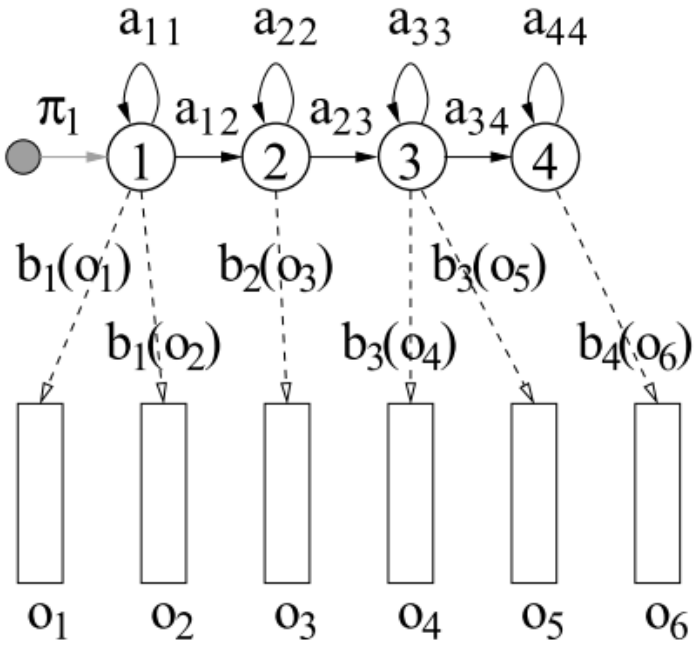
4. State-transition probabilities,
   $A = \{a_{ij}\} = \{P(x_t = j | x_{t-1} = i)\}$ for $1 \leq i, j \leq N$;

5. Discrete output probabilities,
   $B = \{b_i(k)\} = \{P(o_t = k | x_t = i)\}$ for $1 \leq i \leq N$ and $1 \leq k \leq K$.

Probability of state i generating a discrete observation $\mathbf{o}_t$

Will be explained in detail in the next class

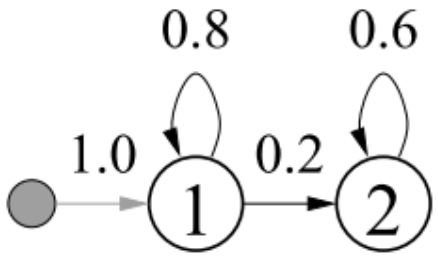
Illustration of HMM as a generator



The state sequence $X = \{1, 1, 2, 3, 3, 4\}$ produces the set of observations $\mathcal{O} = \{o_1, o_2, \ldots, o_6\}$:

$$
\begin{aligned}
P(X|\lambda) &= \pi_1\, a_{11}\, a_{12}\, a_{23}\, a_{33}\, a_{34} \\
P(\mathcal{O}|X,\lambda) &= b_1(o_1)\, b_1(o_2)\, b_2(o_3)\, b_3(o_4)\, b_3(o_5)\, b_4(o_6) \\
P(\mathcal{O},X|\lambda) &= P(\mathcal{O}|X,\lambda)P(X|\lambda) \\
&= \pi_1 b_1(o_1)\, a_{11} b_1(o_2)\, a_{12} b_2(o_3) \ldots \qquad (22)
\end{aligned}
$$

HMM: Toy Example



Transition probabilities:

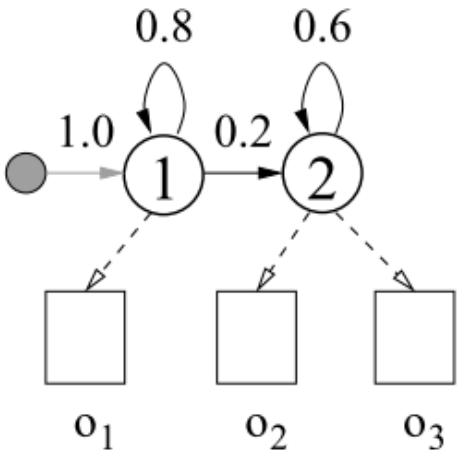
$$\pi = \{\pi_i\} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \text{ and}$$

$$A = \{a_{ij}\} = \begin{bmatrix} 0.8 & 0.2 \\ 0 & 0.6 \end{bmatrix}.$$

Probability of a certain state sequence, $X = \{1, 2, 2\}$:

$$\begin{aligned} P(X|\mathcal{M}) &= \pi_1 \, a_{12} \, a_{22} \\ &= 1 \times \\ &= \end{aligned} \tag{23}$$

HMM: toy example with 3 observations



Output probabilities: $B = \begin{bmatrix} b_1(k) \\ b_2(k) \end{bmatrix} = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0 & 0.9 & 0.1 \end{bmatrix}$.

Probability with a certain state sequence, $X = \{1, 2, 2\}$:

$$\begin{aligned} P(\mathcal{O}, X|\lambda) &= P(\mathcal{O}|X, \lambda)P(X|\lambda) \\ &= \pi_1 b_1(o_1)\, a_{12} b_2(o_2)\, a_{22} b_2(o_3) \\ &= 1 \times \\ &= \\ &\approx \end{aligned}$$

How to find the best, optimal path given input X and HMM parameters

VITERBI DECODING

Finding the best path

Given observations $\mathcal{O} = \{o_1, \ldots, o_T\}$, find the HMM state sequence $X = \{x_1, \ldots, x_T\}$ that has greatest likelihood

$$X^* = \arg\max_X P(\mathcal{O}, X|\lambda), \tag{30}$$

where

$$\begin{aligned} P(\mathcal{O}, X|\lambda) &= P(\mathcal{O}|X, \lambda)P(X|\lambda) \\ &= \pi_{x_1} b_{x_1}(o_1). \prod_{t=2}^{T} a_{x_{t-1}x_t} b_{x_t}(o_t). \end{aligned} \tag{31}$$

The *Viterbi algorithm* is an inductive method for finding the optimal state sequence X^* efficiently.

Decoding – main idea

Induction: Given that for all states k, and for a fixed position i,

$$\delta_k(i) = \max_{\{\pi 1 \ldots \pi i-1\}} P[x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-1}, x_i, \pi_i = k]$$

What is $\delta_l(i+1)$?

From definition,

$$V_l(i+1) = \max_{\{\pi 1 \ldots \pi i\}} P[\, x_1 \ldots x_i, \pi_1, \ldots, \pi_i, x_{i+1}, \pi_{i+1} = l \,]$$

$$= \max_{\{\pi 1 \ldots \pi i\}} P(x_{i+1}, \pi_{i+1} = l \mid x_1 \ldots x_i, \pi_1, \ldots, \pi_i)\, P[x_1 \ldots x_i, \pi_1, \ldots, \pi_i]$$

$$= \max_{\{\pi 1 \ldots \pi i\}} P(x_{i+1}, \pi_{i+1} = l \mid \pi_i)\, P[x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-1}, x_i, \pi_i]$$

$$= \max_k \left[P(x_{i+1}, \pi_{i+1} = l \mid \pi_i = k) \max_{\{\pi 1 \ldots \pi i-1\}} P[x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-1}, x_i, \pi_i = k] \right]$$

$$= \max_k \left[P(x_{i+1} \mid \pi_{i+1} = l)\, P(\pi_{i+1} = l \mid \pi_i = k)\, \delta_k(i) \right]$$

$$= e_l(x_{i+1}) \max_k a_{kl}\, \delta_k(i)$$

Viterbi algorithm

Similar recurrence to DTW, with template is replaced with an HMM, distance with observation probabilities

- Same path constraints can be applied (local and global)
- Usually, the number of states is much smaller than the number of frames

1. Initialise,

$$\delta_1(i) = \pi_i b_i(o_1)$$
$$\psi_1(i) = 0$$

for $1 \leq i \leq N$;

The function $\delta_t(j)$ and $\psi_t(j)$, is the probability and state index of the best path that symbol t is at state j.
The probability of the total input is $\delta_T(N)$ for the simple left-to-right, monotonic HMM.

2. Recur for $t = 2, \ldots, T$,

$$\delta_t(j) = \max_i \left[\delta_{t-1}(i) a_{ij}\right] b_j(o_t)$$
$$\psi_t(j) = \arg\max_i \left[\delta_{t-1}(i) a_{ij}\right]$$

for $1 \leq j \leq N$;

3. Finalise,

$$P(\mathcal{O}, X^* | \lambda) = \max_i \left[\delta_T(i)\right]$$
$$x_T^* = \arg\max_i \left[\delta_T(i)\right]$$

4. Trace back, for $t = T-1, T-2, \ldots, 1$,

$$x_t^* = \psi_{t+1}\left(x_{t+1}^*\right), \quad \text{and} \quad X^* = \{x_1^*, x_2^*, \ldots, x_T^*\}.$$

Illustration of the Viterbi Algorithm

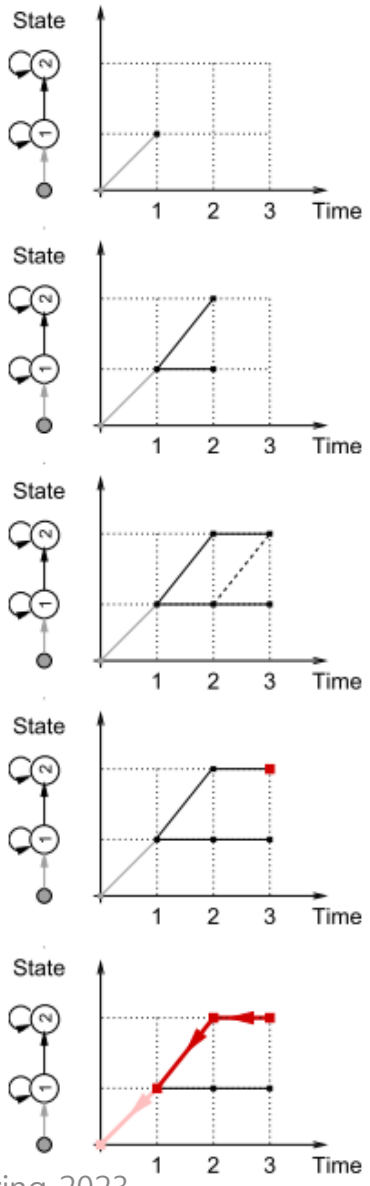
1. Initialise,
   $\delta_1(i) = \pi_i b_i(o_1)$
   $\psi_1(i) = 0$

2. Recur for $t = 2$,
   $\delta_2(j) = \max_i \left[\delta_1(i)a_{ij}\right] b_j(o_2)$
   $\psi_2(j) = \arg\max_i \left[\delta_1(i)a_{ij}\right]$

   Recur for $t = 3$,
   $\delta_3(j) = \max_i \left[\delta_2(i)a_{ij}\right] b_j(o_3)$
   $\psi_3(j) = \arg\max_i \left[\delta_2(i)a_{ij}\right]$

3. Finalise,
   $P(\mathcal{O}, X^*|\lambda) = \max_i [\delta_3(i)]$
   $x_3^* = \arg\max_i [\delta_3(i)]$

4. Trace back for t=2..1,
   $x_2^* = \psi_3\left(x_3^*\right)$
   $x_1^* = \psi_2\left(x_2^*\right)$
   $X^* = \{x_1^*, x_2^*, x_3^*\}$.



Viterbi Algorithm – a practical detail

Underflows are a significant problem

$$P[x_1,\ldots., x_i, \pi_1, \ldots, \pi_i] = a_{0\pi 1} a_{\pi 1\pi 2}\ldots\ldots a_{\pi i} e_{\pi 1}(x_1)\ldots\ldots e_{\pi i}(x_i)$$

These numbers become extremely small – underflow

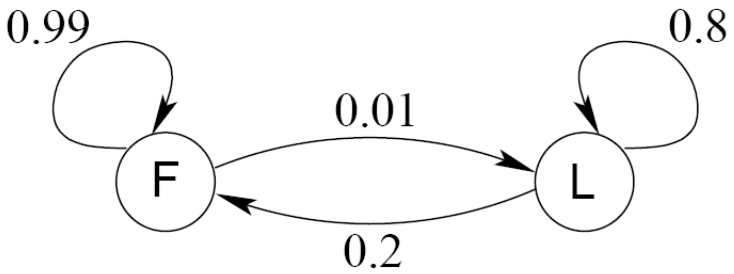
Solution: Take the logs of all values

$$\delta_l(i) = \log e_k(x_i) + \max_k [\delta_k(i-1) + \log a_{kl}]$$

Viterbi: Loaded Dice Example

π		ε	6	2	6
				x	
	B	1	0	0	0
	F	0	(1/6)×(1/2) = 1/12	(1/6)×max{(1/12)×0.99, (1/4)×0.2} = 0.01375	(1/6)×max{0.01375×0.99, 0.02×0.2} = 0.00226875
	L	0	(1/2)×(1/2) = 1/4	(1/10)×max{(1/12)×0.01, (1/4)×0.8} = 0.02	(1/2)×max{0.01375×0.01, 0.02×0.8} = 0.08

$$\delta_k(i) = e_k(x_i)\max_r\left(\delta_r(i-1)a_{rk}\right)$$



Viterbi gets it right more often than not

```
Rolls     315116246446644245321131631164152133625144543631656626566666
Die       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi   FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLL

Rolls     651166453132651245636664631636663162326455235266666625151631
Die       LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLFFFFFFFFF
Viterbi   LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF

Rolls     222555441666566563564324364131513465146353411126414626253356
Die       FFFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL
Viterbi   FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls     366163666466232534413661661163252562462255265252266435353336
Die       LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi   LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls     233121625364414432335163243633665562466662632666612355245242
Die       FFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
Viterbi   FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
```

A **simple** algorithm to train the model parameters

- State transition probabilities
- Observation probabilities

SEGMENTAL K-MEANS ALGORITHM TO TRAIN HMM

Learning Problem

- **Learning problem.** Given some training observation sequences $O=o_1 o_2 \ldots o_K$ 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 $P(O \mid M)$.

As a preliminary approach, let us use the optimal path X^* computed by the Viterbi algorithm with some initial model parameters $\lambda = \{\pi, A, B\}$. In so doing, we approximate the total likelihood of the observations:

$$P(\mathcal{O}, X|\lambda) = \sum_{\text{all } X} P(\mathcal{O}, X|\lambda) \approx P(\mathcal{O}, X^*|\lambda). \qquad (34)$$

Using X^*, we can make a hard binary decision about the occupation of state i, $q_t(i) \in \{0, 1\}$, and train the parameters of our model accordingly.

Viterbi training (hard state assignment)

Model parameters can be re-estimated using the Viterbi state alignment (aka. segmental k-means training):

(a) Initial-state probabilities,

$$\hat{\pi}_i = q_1(i) \qquad \text{for } 1 \le i \le N;$$

where state indicator $q_t(i) = \begin{cases} 1 & \text{for } i = x_t; \\ 0 & \text{otherwise.} \end{cases}$

(b) State-transition probabilities,

$$\hat{a}_{ij} = \frac{\sum_{t=2}^{T} q_{t-1}(i) q_t(j)}{\sum_{t=2}^{T} q_{t-1}(i)} \qquad \text{for } 1 \le i, j \le N;$$

(c) Discrete output probabilities,

$$\hat{b}_j(k) = \frac{\sum_{t=1}^{T} \omega_t(k) q_t(j)}{\sum_{t=1}^{T} q_t(j)} \qquad \text{for } 1 \le j \le N;$$

and $1 \le k \le K$.

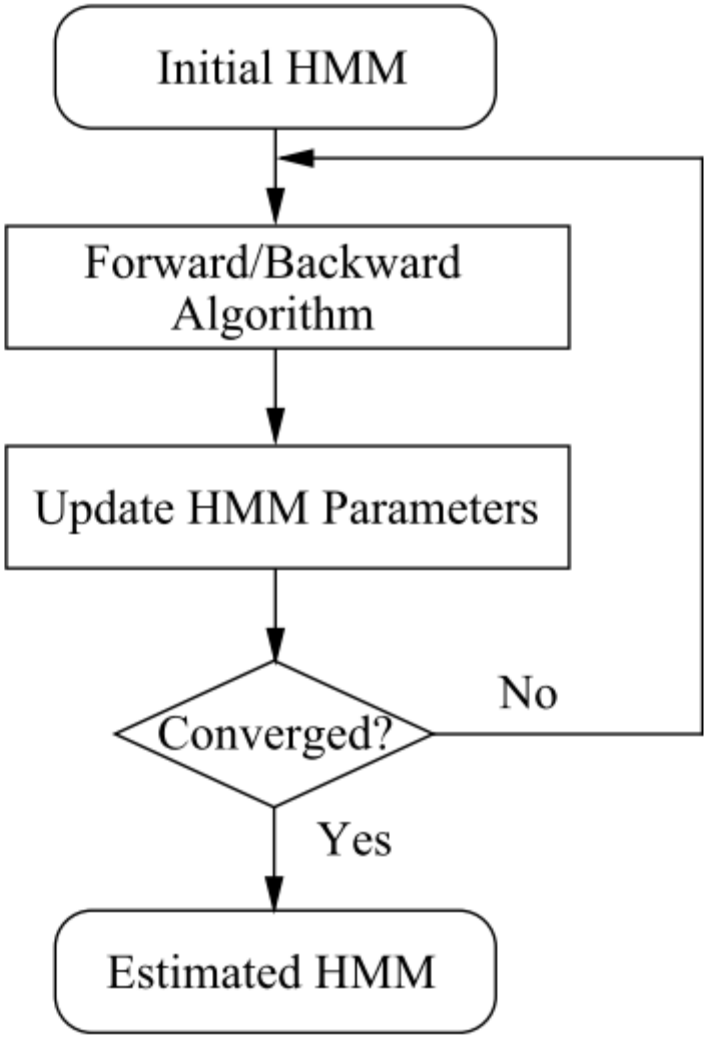
where event indicator $\omega_t(k) = \begin{cases} 1 & \text{for } k = o_t; \\ 0 & \text{otherwise.} \end{cases}$

Learning problem (2)

$$a_{ij} = P(s_i \mid s_j) = \frac{\text{Number of transitions from state } S_j \text{ to state } S_i}{\text{Number of transitions out of state } S_j}$$

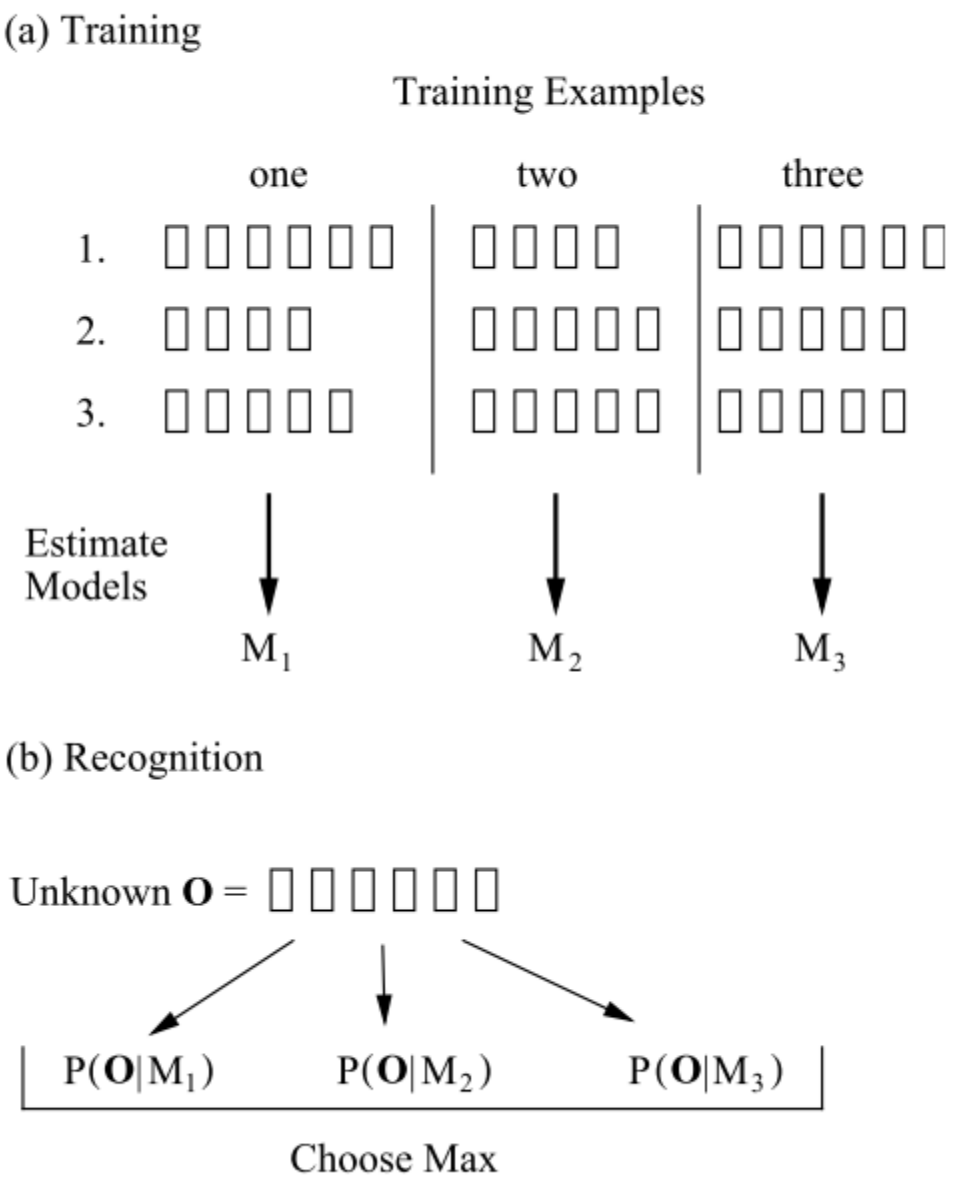
$$b_i(v_m) = P(v_m \mid s_i) = \frac{\#\text{times observation } V_m \text{ occurs in state } S_i}{\text{Number of times in state } S_i}$$

Overview of the training procedure



Parameter re-estimation based on a single training example (Young et al., 2002).

Use of HMMs with training and test data



Training and isolated-word recognition (Young et al., 2002).