ISyE 6416: Computational Statistics
Spring 2017

Lecture 8: Hidden Markov Model

Prof. Yao Xie

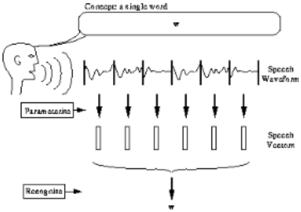
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Outline

- Motivating applications
- Set-up
- Forward-backward algorithm
- Viterbi algorithm
- Baum-Welch algorithm for model estimation

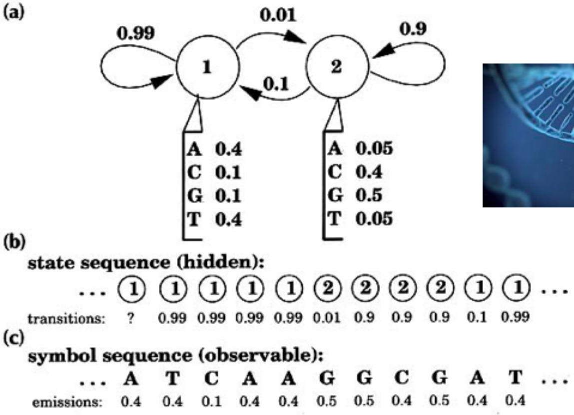
Speech recognition

Let each spoken word be represented by a sequence of speech signals.



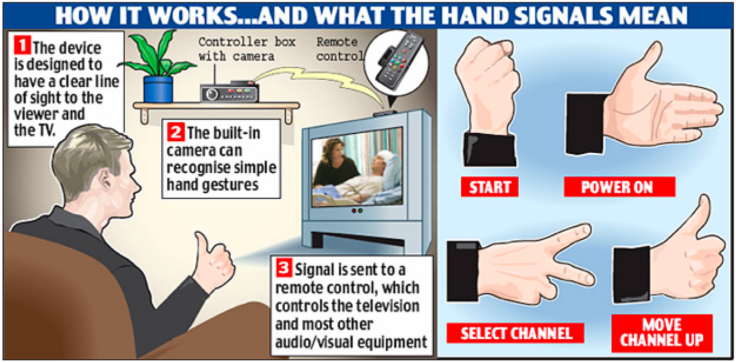
One speaks of an HMM 'generating' a sequence. The HMM is composed of a number of states. Each state 'emits' symbols (residues) according to symbol-emission probabilities, and the states are interconnected by state-transition probabilities. Start from some initial state, a sequence of states is generated by moving from state to state according to the state transition probabilities until an end state is reached. Each state then emits symbols according to that state's emission probability distribution, creating an observable sequence of symbols. - L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE, 1989

Genetics



For a given observed DNA sequence, we are interested in inferring the hidden state sequence that 'generated' it, that is, whether this position is in a CG-rich segment or an AT-rich segment.

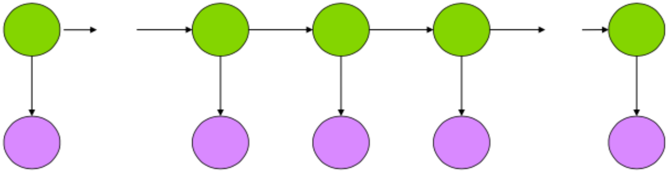
Gesture recognition



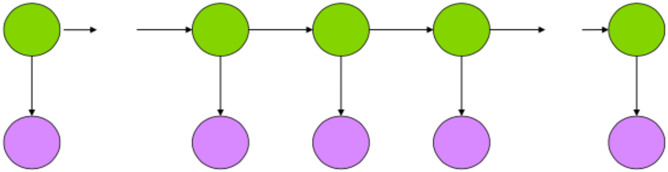
Gesture recognition is a topic in computer science and language technology with the goal of interpreting human gestures via mathematical algorithms. Gestures can originate from any bodily motion or state but commonly originate from the face or hand, e.g., kinetic user interface.

Hidden Markov Model

- HMM is a Markov chain, where at each time, the hidden state determines a observation.
- Goal is to infer the hidden state from the sequence of observations.
- Its special structure enables efficiently statistical estimation: a special case of *graphical model*
- HMM useful to model dependence in time sequence



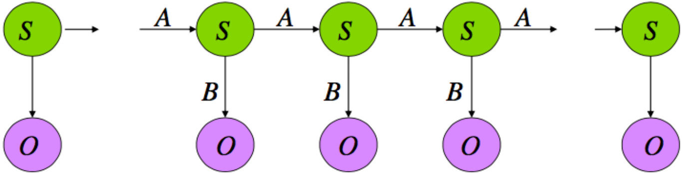
Specifications



- Circles: states
- arrows: probabilistic dependencies between states.
- Green: **hidden states**. Each state only depends on previous state.
- Purple: **observations**. Only depends on their corresponding hidden states.

The past is independent of the future given the present.

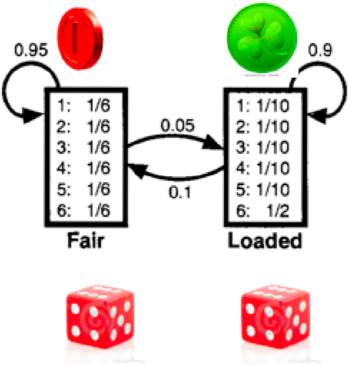
Formalization



- Time horizon $t = 1, \ldots, T$, number of possible states K
- $\{S, O, \Pi, A, B\}$
- $\Pi = \{\pi_i\}$ are the initial state probabilities
- $A = \{a_{ij}\}$ are the state transition probabilities, $i, j = 1, \ldots, K$
- $B = \{b_{k\ell}\}$ are the observation state probabilities, $k = 1, \ldots, K$, $\ell = 1, \ldots, |O|$

Example: occasionally dishonest casino

Model that generates the sequence



Consider a Markov model with two states and six possible emissions

A weighted red coin, for which the probability of heads is .9 and the probability of tails is .1.

A weighted green coin, for which the probability of heads is .95 and the probability of tails is .05.

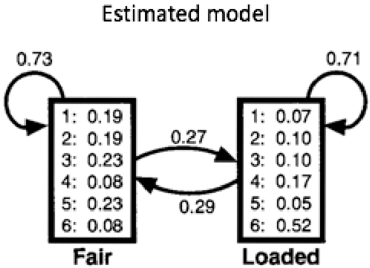
```
Rolls   315116246446644245311321631164152133625144543631656626566666
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
```

- $K = 2$
- Hidden states $S_t \in \{F, L\}$
- Observations $O_t \in \{1, 2, \ldots, 6\}$
- Initial state probability $\pi_1 = \pi_2 = 1/2$
- Transition matrix

$$A = \begin{bmatrix} 0.9 & 0.1 \\ 0.05 & 0.95 \end{bmatrix}$$

- Emission matrix

$$B = \begin{bmatrix} 1/6 & 1/6 & \cdots & 1/6 \\ 1/2 & 1/10 & \cdots & 1/10 \end{bmatrix}$$

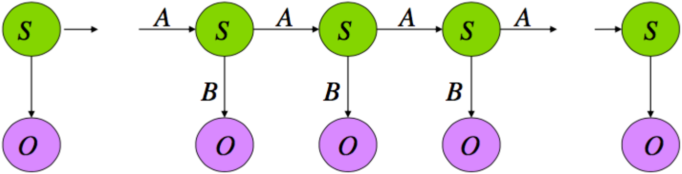


Inference in an HMM

- **Decoding**: Given an observation sequence and a model, compute the most likely hidden state sequence: **Forward-Backward algorithm** and **Viterbi algorithm**

- **Baum-Welch algorithm**: Given observation sequence, estimating model parameters; based on **EM algorithm**.

Probability calculation for HMM



(there is also an initial state S_0)

- Joint state and observation

$$\mathbb{P}(o_1, \ldots, o_T, s_0, s_1, \ldots, s_T)$$
$$= \pi_{s_0} a_{s_0,s_1} b_{s_1,o_1} a_{s_1,s_2} b_{s_2,o_2} \cdots a_{S_{T-1},S_T} b_{S_T,o_T}$$

Forward-Backward (FB) algorithm

- The Forward-Backward (FB) algorithm is used to compute the probabilities efficiently

$$\mathbb{P}(S_t = i | o_1, \ldots, o_T) \quad \text{“most likely state at any time”}$$

$$\mathbb{P}(S_t = i, S_{t+1} = j | o_1, \ldots, o_T) \quad \text{“most likely transition at any time”}$$

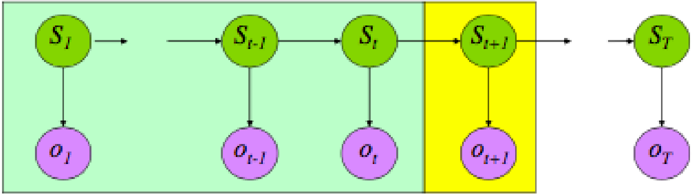
- **Strategy**: Break the sequence into past and future

$$\mathbb{P}(S_t = i | o_1, \cdots, o_T) \propto \underbrace{\mathbb{P}(S_t = i, o_1, \cdots, o_t)}_{\alpha_i(t)} \cdot \underbrace{\mathbb{P}(o_{t+1}, \cdots, o_T | S_t = i)}_{\beta_i(t)}$$

$$\mathbb{P}(S_t = i, S_{t+1} = j | o_1, \ldots, o_T) \propto \alpha_i(t) \beta_j(t+1) a_{i,j} b_{j,o_{t+1}}$$

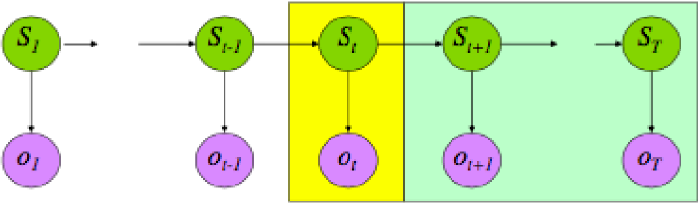
- Brute-force computation $\mathcal{O}(TK^T)$
- FB complexity $\mathcal{O}(K^2T)$

Forward recursion



- Special structure gives an efficient solution using *dynamic programming*
  Define
  $$\alpha_i(t) = \mathbb{P}(o_1, \ldots, o_t, S_t = i), \quad \alpha_i(0) = \pi_i$$
  Recursion due to Markov structure
  $$\alpha_j(t+1) = b_{jo_{t+1}} \sum_{i=1}^{K} \alpha_i(t) a_{ij}, \quad t = 0, \ldots, T-1$$
  Proof

Backward recursion



Define

$$\beta_i(T) = 1$$

$$\beta_i(t) = \mathbb{P}(o_{t+1}, \ldots, o_T | S_t = i)$$

Can show recursion due to Markov

$$\beta_i(t) = \sum_{j=1}^{K} a_{ij} b_{j o_{t+1}} \beta_j(t+1), \quad t = 0, \ldots, T-1$$

Forward recursion

$$
\begin{aligned}
\alpha_j(t+1) &= \mathbb{P}(o_1, \ldots, o_{t+1}, S_{t+1} = j) \\
&= \mathbb{P}(o_1, \ldots, o_{t+1} | S_{t+1} = j)\mathbb{P}(S_{t+1} = j) \\
&= \mathbb{P}(o_1, \ldots, o_t | S_{t+1} = j)\mathbb{P}(o_{t+1} | S_{t+1} = j)\mathbb{P}(S_{t+1} = j) \\
&= \mathbb{P}(o_1, \ldots, o_t, S_{t+1} = j)\mathbb{P}(o_{t+1} | S_{t+1} = j) \\
&= \sum_{i=1,\ldots,K} \mathbb{P}(o_1, \ldots, o_t, S_t = i, S_{t+1} = j)\mathbb{P}(o_{t+1} | S_{t+1} = j) \\
&= \sum_{i=1,\ldots,K} \mathbb{P}(o_1, \ldots, o_t, S_{t+1} = j | S_t = i)\mathbb{P}(S_t = i) \\
&\quad \mathbb{P}(o_{t+1} | S_{t+1} = j) \\
&= \sum_{i=1,\ldots,K} \mathbb{P}(o_1, \ldots, o_t, S_t = i)\mathbb{P}(S_{t+1} = j | S_t = i) \\
&\quad \mathbb{P}(o_{t+1} | S_{t+1} = j) \\
&= b_{j o_{t+1}} \sum_{i=1}^{K} \alpha_i(t) a_{ij}.
\end{aligned}
$$

Backward recursion

$$
\begin{aligned}
\beta_i(t) &= \mathbb{P}(o_{t+1}, \ldots, o_T | S_t = i) \\
&= \sum_{j=1,\ldots,K} \mathbb{P}(o_{t+1}, \ldots, o_T, S_{t+1} = j | S_t = i) \\
&= \sum_{j=1,\ldots,K} \mathbb{P}(o_{t+1}, \ldots, o_T | S_{t+1} = j, S_t = i) \mathbb{P}(S_{t+1} = j | S_t = i) \\
&= \sum_{j=1,\ldots,K} \mathbb{P}(o_{t+1} | S_{t+1} = j) \mathbb{P}(o_{t+2}, \ldots, o_T | S_{t+1} = j) \\
&\quad \mathbb{P}(S_{t+1} = j | S_t = i) \\
&= \sum_{j=1}^{K} a_{ij} b_{j o_{t+1}} \beta_j(t+1)
\end{aligned}
$$

Smoothing

- FB algorithm can be used to compute the most likely state for any point in time

$$\mathbb{P}(S_t = k | o_1, \ldots, o_T) = \frac{\alpha_k(t)\beta_k(t)}{\mathbb{P}(o_1, \ldots, o_T)}$$

$$\mathbb{P}(o_1, \ldots, o_T) = \sum_{k=1}^{K} \alpha_k(t)\beta_k(t), \quad \forall t$$

- BUT FB cannot be used to find the most likely **sequence of states**, have to be done through the **Viterbi algorithm**

Example: Rain man

- We would like to infer the weather given observation of a man either carrying nor not carrying an umbrella
- two possible states for the weather: state 1 = rain, state 2 = no rain
- the weather has a 70% chance of staying the same each day and a 30% chance of changing

Russell & Norvig 2010 Chapter 15 pp. 566

Example: Rain man (cont)

- transition probability

$$A = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix} \to \{a_{ij}\}$$

- assume each state generates 2 events: event $1 =$ umbrella, event $2 =$ no umbrella.
- emission probability: the conditional probabilities for these events occurring in each state

$$B = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix} \to \{b_{ij}\}$$

- observe a sequence of events: $\{$umbrella, umbrella, no umbrella, umbrella, umbrella$\}$
- what's the weather like?

Matrix-vector forms

- Define observation matrix

$$O_j = \text{diag}(b_{*,o_j})$$

Example:

events: {umbrella, umbrella, no umbrella, umbrella, umbrella}

$$\mathbf{O_1} = \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix} \mathbf{O_2} = \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix} \mathbf{O_3} = \begin{pmatrix} 0.1 & 0.0 \\ 0.0 & 0.8 \end{pmatrix} \mathbf{O_4} = \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix} \mathbf{O_5} = \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix}$$

- Initial state vector π_0

- Forward probabilities

$$\alpha_i(t) = \mathbb{P}(o_1, \ldots, o_t, S_t = i)$$

$$f_{0:t} = [\alpha_1(t), \ldots, \alpha_K(t)]^T$$

- Forward recursion

$$\alpha_i(0) = \pi_i, \quad \alpha_j(t+1) = b_{j o_{t+1}} \sum_{i=1}^{K} \alpha_i(t) a_{ij}$$

Forward recurision

$$f_{0:0} = [\pi_1, \ldots, \pi_K]^T, \quad f_{0:t+1} = O_{t+1} A^T f_{0:t}, \quad t = 0, \ldots, T-1$$

then scale each vector to sum up to 1 since $\sum_{k=1}^{K} \alpha_k(t) = 1$

- Backward probabilities

$$\beta_i(t) = \mathbb{P}(o_{t+1}, \ldots, o_T | S_t = i)$$

$$r_{t:T} = [\beta_1(t), \ldots, \beta_K(t)]^T, \quad r_{T:T} = [1, 1, \ldots, 1]^T$$

- Backward recursion

$$\beta_i(t) = \sum_{j=1}^{K} a_{ij} b_{j o_{t+1}} \beta_j(t+1), \quad t = 0, \ldots, T-1$$

can be written as

Backward recursion

$$r_{T:T} = [1, 1, \ldots, 1]^T, \quad r_{t:T} = A O_{t+1} r_{t+1:T}, \quad t = 0, \ldots, T-1$$

then scale each vector to sum up to 1 since $\sum_{i=1}^{K} \beta_i(t) = 1$

Rain man: Computation - forward

events: {umbrella, umbrella, no umbrella, umbrella, umbrella}

$$\mathbf{O}_1 = \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix} \ \mathbf{O}_2 = \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix} \ \mathbf{O}_3 = \begin{pmatrix} 0.1 & 0.0 \\ 0.0 & 0.8 \end{pmatrix} \ \mathbf{O}_4 = \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix} \ \mathbf{O}_5 = \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix}$$

$$f_{0:0} = (0.5, 0.5)^T$$

$$(\hat{\mathbf{f}}_{0:1})^T = c_1^{-1} \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix} \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \begin{pmatrix} 0.5000 \\ 0.5000 \end{pmatrix} = c_1^{-1} \begin{pmatrix} 0.4500 \\ 0.1000 \end{pmatrix} = \begin{pmatrix} 0.8182 \\ 0.1818 \end{pmatrix}$$

$$(\hat{\mathbf{f}}_{0:2})^T = c_2^{-1} \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix} \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \begin{pmatrix} 0.8182 \\ 0.1818 \end{pmatrix} = c_2^{-1} \begin{pmatrix} 0.5645 \\ 0.0745 \end{pmatrix} = \begin{pmatrix} 0.8834 \\ 0.1166 \end{pmatrix}$$

$$(\hat{\mathbf{f}}_{0:3})^T = c_3^{-1} \begin{pmatrix} 0.1 & 0.0 \\ 0.0 & 0.8 \end{pmatrix} \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \begin{pmatrix} 0.8834 \\ 0.1166 \end{pmatrix} = c_3^{-1} \begin{pmatrix} 0.0653 \\ 0.2772 \end{pmatrix} = \begin{pmatrix} 0.1907 \\ 0.8093 \end{pmatrix}$$

$$(\hat{\mathbf{f}}_{0:4})^T = c_4^{-1} \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix} \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \begin{pmatrix} 0.1907 \\ 0.8093 \end{pmatrix} = c_4^{-1} \begin{pmatrix} 0.3386 \\ 0.1247 \end{pmatrix} = \begin{pmatrix} 0.7308 \\ 0.2692 \end{pmatrix}$$

$$(\hat{\mathbf{f}}_{0:5})^T = c_5^{-1} \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix} \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \begin{pmatrix} 0.7308 \\ 0.2692 \end{pmatrix} = c_5^{-1} \begin{pmatrix} 0.5331 \\ 0.0815 \end{pmatrix} = \begin{pmatrix} 0.8673 \\ 0.1327 \end{pmatrix}$$

Rain man: Computation - backward

$$b_{5:5} = (1.0, 1.0)^T$$

$$\hat{\mathbf{b}}_{4:5} = \alpha \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix} \begin{pmatrix} 1.0000 \\ 1.0000 \end{pmatrix} = \alpha \begin{pmatrix} 0.6900 \\ 0.4100 \end{pmatrix} = \begin{pmatrix} 0.6273 \\ 0.3727 \end{pmatrix}$$

$$\hat{\mathbf{b}}_{3:5} = \alpha \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix} \begin{pmatrix} 0.6273 \\ 0.3727 \end{pmatrix} = \alpha \begin{pmatrix} 0.4175 \\ 0.2215 \end{pmatrix} = \begin{pmatrix} 0.6533 \\ 0.3467 \end{pmatrix}$$

$$\hat{\mathbf{b}}_{2:5} = \alpha \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \begin{pmatrix} 0.1 & 0.0 \\ 0.0 & 0.8 \end{pmatrix} \begin{pmatrix} 0.6533 \\ 0.3467 \end{pmatrix} = \alpha \begin{pmatrix} 0.1289 \\ 0.2138 \end{pmatrix} = \begin{pmatrix} 0.3763 \\ 0.6237 \end{pmatrix}$$

$$\hat{\mathbf{b}}_{1:5} = \alpha \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix} \begin{pmatrix} 0.3763 \\ 0.6237 \end{pmatrix} = \alpha \begin{pmatrix} 0.2745 \\ 0.1889 \end{pmatrix} = \begin{pmatrix} 0.5923 \\ 0.4077 \end{pmatrix}$$

$$\hat{\mathbf{b}}_{0:5} = \alpha \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix} \begin{pmatrix} 0.5923 \\ 0.4077 \end{pmatrix} = \alpha \begin{pmatrix} 0.3976 \\ 0.2170 \end{pmatrix} = \begin{pmatrix} 0.6469 \\ 0.3531 \end{pmatrix}$$

Rain man: Computation - smoothing

$$\mathbb{P}(S_t = k | o_1, \dots, o_T) = \frac{\alpha_k(t)\beta_k(t)}{\mathbb{P}(o_1, \dots, o_T)}$$

$$(\gamma_0)^T = \alpha \begin{pmatrix} 0.5000 \\ 0.5000 \end{pmatrix} \circ \begin{pmatrix} 0.6469 \\ 0.3531 \end{pmatrix} = \alpha \begin{pmatrix} 0.3235 \\ 0.1765 \end{pmatrix} = \begin{pmatrix} 0.6469 \\ 0.3531 \end{pmatrix}$$

$$(\gamma_1)^T = \alpha \begin{pmatrix} 0.8182 \\ 0.1818 \end{pmatrix} \circ \begin{pmatrix} 0.5923 \\ 0.4077 \end{pmatrix} = \alpha \begin{pmatrix} 0.4846 \\ 0.0741 \end{pmatrix} = \begin{pmatrix} 0.8673 \\ 0.1327 \end{pmatrix}$$

$$(\gamma_2)^T = \alpha \begin{pmatrix} 0.8834 \\ 0.1166 \end{pmatrix} \circ \begin{pmatrix} 0.3763 \\ 0.6237 \end{pmatrix} = \alpha \begin{pmatrix} 0.3324 \\ 0.0728 \end{pmatrix} = \begin{pmatrix} 0.8204 \\ 0.1796 \end{pmatrix}$$

$$(\gamma_3)^T = \alpha \begin{pmatrix} 0.1907 \\ 0.8093 \end{pmatrix} \circ \begin{pmatrix} 0.6533 \\ 0.3467 \end{pmatrix} = \alpha \begin{pmatrix} 0.1246 \\ 0.2806 \end{pmatrix} = \begin{pmatrix} 0.3075 \\ 0.6925 \end{pmatrix}$$

$$(\gamma_4)^T = \alpha \begin{pmatrix} 0.7308 \\ 0.2692 \end{pmatrix} \circ \begin{pmatrix} 0.6273 \\ 0.3727 \end{pmatrix} = \alpha \begin{pmatrix} 0.4584 \\ 0.1003 \end{pmatrix} = \begin{pmatrix} 0.8204 \\ 0.1796 \end{pmatrix}$$

$$(\gamma_5)^T = \alpha \begin{pmatrix} 0.8673 \\ 0.1327 \end{pmatrix} \circ \begin{pmatrix} 1.0000 \\ 1.0000 \end{pmatrix} = \alpha \begin{pmatrix} 0.8673 \\ 0.1327 \end{pmatrix} = \begin{pmatrix} 0.8673 \\ 0.1327 \end{pmatrix}$$

events: {umbrella, umbrella, no umbrella, umbrella, umbrella}

estimate: {rain, rain, not rain, rain, rain}

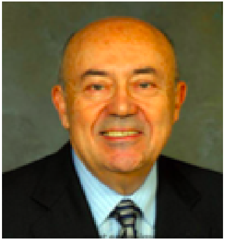
example from wikipedia

Viterbi Algorithm

- Forward-backward algorithm finds posterior probability of a *single* state at any time $\mathbb{P}(S_t = k | o_1, \ldots, o_T)$
- Viterbi Algorithm finds the most likely **sequence** of states by

$$\max_{S_0, S_1, \ldots, S_T} \mathbb{P}(S_0, S_1, \ldots, S_T | o_1, \ldots, o_T)$$

- Developed by Andrew Viterbi, 1966
- Solve using *dynamic programming*
- Exploit the Markov structure of the problem to beat the "curse-of-dimensionality" and lead to structured solution



Andrew Viterbi

Derivation of Viterbi algorithm

- Note that the likelihood function can be written as

$$\mathbb{P}(S_0, S_1, \ldots, S_T | o_1, \ldots, o_T) = \frac{\mathbb{P}(S_0, S_1, \ldots, S_T, o_1, \ldots, o_T)}{\mathbb{P}(o_1, \ldots, o_T)}$$

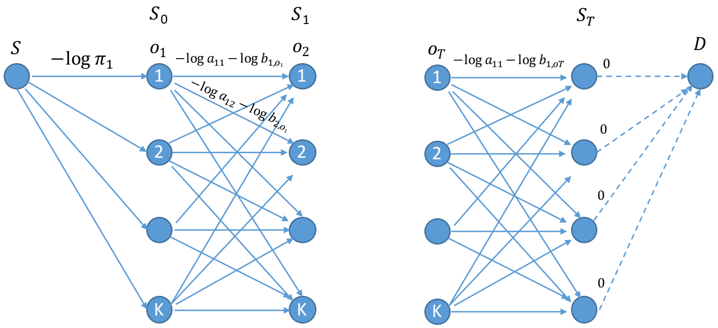
- Then we can show

$$\mathbb{P}(S_0, \ldots, S_T, o_1, \ldots, o_T) = \pi_{S_0} \prod_{k=1}^{T} a_{S_{k-1},S_k} b_{S_k,o_k}$$

- Taking negative log, one aims to find $\{S_0, S_1, \ldots, S_T\}$ to minimize

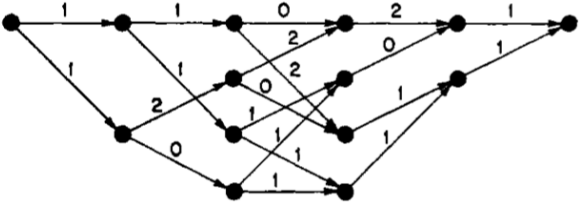
$$-\log \pi_{S_0} + \underbrace{\sum_{k=1}^{T} [-\log a_{S_{k-1},S_k} - \log b_{S_k,o_k}]}_{\text{decouple in time}}$$

- Convert the problem of finding the most likely sequence to the problem of finding the shortest path
- Find shortest path: first find a shortest path from $S \to$ step 1, and then use the distance to calculate $S \to$ step 2



$$-\log \pi_{S_0} + \sum_{k=1}^{T} [-\log a_{S_{k-1},S_k} - \log b_{S_k,o_k}]$$

- Solving $\max_{S_0, S_1, \ldots, S_T} \mathbb{P}(S_0, S_1, \ldots, S_T | o_1, \ldots, o_T)$ by brute-force (enumerate all possible paths) complexity is K^T
- Viterbi has complexity $\mathcal{O}(K^2 T)$, memory requirement is $\mathcal{O}(KT)$

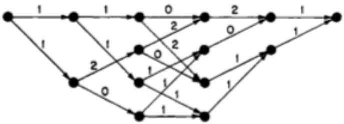


(a)

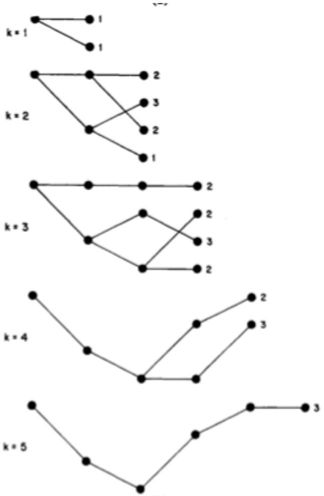
From "The Viterbi Algorithm", by D. Forney, 1973

An example Trellis from "The Viterbi Algorithm" by D. Forney 1973

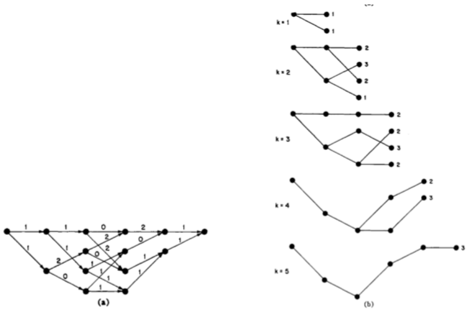
Example: path elimination



(a)



(b)

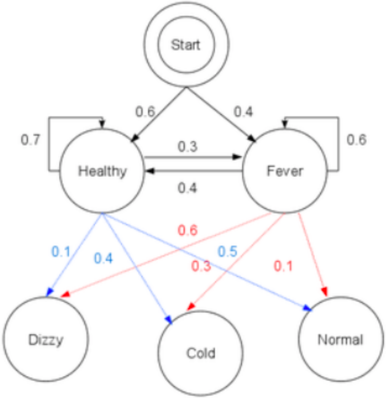


- Shortest path segment is called the *survivor* for a node
- important observation: the shortest complete path must begin with one of the survivors
- each stage only need to store K survivor paths

Example: Doctor's decision

- Consider the following model for a patient
- The patient visit 3 days in a row, and her symptoms are
  {normal, cold, dizzy}

What's the most likely sequence of states of the patients in 3 days? Draw a diagram for Viterbi algorithm and calculate weights on each edge.



Estimating Gaussian HMM Model

- Consider Gaussian emission probability

$$b_k(o) = \mathcal{N}(\mu_k, \Sigma_k)$$

- Model parameters a_{ij}, $i = 1, \ldots, K$, $j = 1, \ldots, K$
- Initial distribution π_i, $i = 1, \ldots, K$
- Emission probability parameters μ_i, Σ_i, $i = 1, \ldots, K$

Baum-Welch algorithm: EM for HMM

- **E-step**
  Compute $L_i(t)$ and $H_{i,j}(t)$ (from forward-backward algorithm)

$$L_i(t) = \mathbb{P}(S_t = i | o_1, \ldots, o_T)$$

$$H_{i,j}(t) = \mathbb{P}(S_t = i, S_{t+1} = j | o_1, \ldots, o_T)$$

- **M-step**: update parameters

$$\mu_i = \frac{\sum_{t=0}^{T} L_i(t) o_t}{\sum_{t=0}^{T} L_i(t)}, \quad \Sigma_i = \frac{\sum_{t=0}^{T} L_i(t)(o_t - \mu_i)(o_t - \mu_i)^T}{\sum_{t=0}^{T} L_i(t)}$$

$$a_{ij} = \frac{\sum_{t=0}^{T-1} H_{i,j}(t)}{\sum_{t=0}^{T-1} L_i(t)}, \quad \pi_i \propto \sum_{t=0}^{T} L_i(t)$$

Derivation of EM

- Compute $Q(\theta|\theta')$ function

$$\log f(S_0, \ldots, S_T, o_1, \ldots, o_T|\theta)$$

$$= \log \pi_{S_0} + \sum_{k=1}^{T} \log a_{S_{k-1},S_k} + \log b_{S_k,o_k}$$

$$\mathbb{E}[\log f(S_0, \ldots, S_T, o_1, \ldots, o_T|\theta)|o_1, \ldots, o_T, \theta']$$

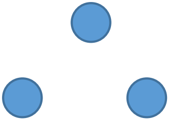
$$= \sum_{\mathbf{s}} \mathbb{P}(\mathbf{s}|\mathbf{o}, \theta') \left[\log \pi_{S_0} + \sum_{k=1}^{T} \log a_{S_{k-1},S_k} + \log b_{S_k,o_k}\right]$$

$$= \sum_{i=1}^{K} L_i(0) \log(\pi_i) + \sum_{t=0}^{t-1} \sum_{i=1}^{K} \sum_{j=1}^{K} H_{ij}(t) \log a_{ij}$$

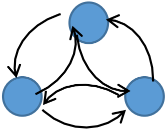
$$+ \sum_{t=0}^{T} \sum_{i=1}^{K} L_i(t) \log \mathbb{P}(o_t|\mu_i, \Sigma_i)$$

Comparison with GMM estimation

- $L_i(t)$ plays a similar role as the posterior probability of a component (state) given observation:
  HMM: $L_i(t) = \mathbb{P}(S_t = i | o_1, \ldots, o_T)$
  GMM: $p_{i,t} = \mathbb{P}(S_t = i | o_t)$
- view a mixture model as a special HMM independent states



GMM



HMM