隐马尔可夫模型
Hidden Markov Model

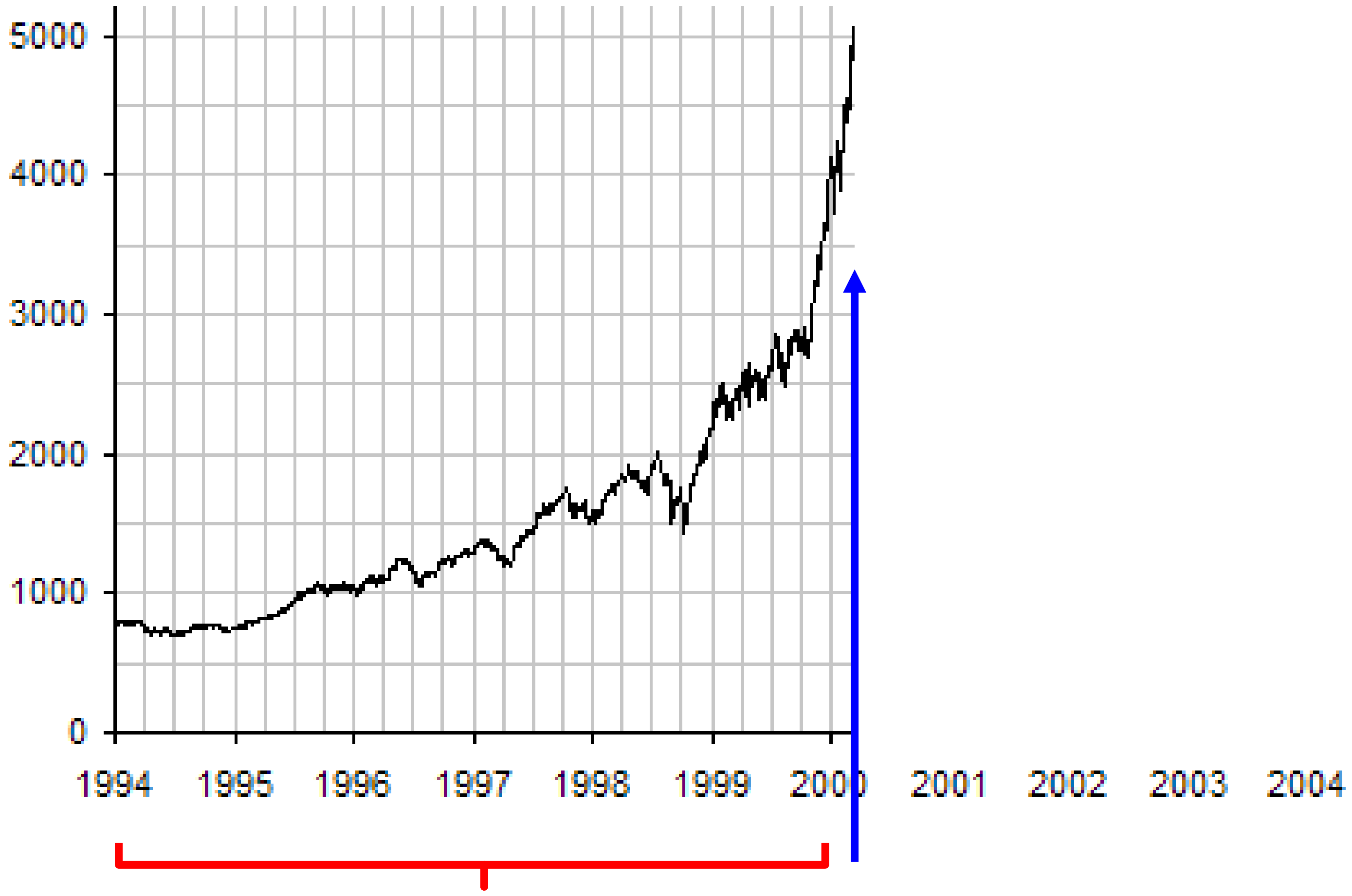
清华大学深圳研究生院
吴志勇
zywu@sz.tsinghua.edu.cn

Preview

- 隐马尔可夫模型（**Hidden Markov Model**）
  - HMM的基本概念与原理
    - $\lambda = f\,(\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$
  - HMM的三个基本问题
    - 概率计算
      - 已知系统输出 **Y** 及模型 $\lambda = f\,(\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$，计算产生 **Y** 的概率$P(\mathbf{Y}|\lambda)$
    - 参数估计：训练
      - 给定若干输出（训练样本）**Y**，确定模型 $\lambda = f\,(\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$ 的参数
    - 最优状态序列搜索：识别
      - 已知输出 **Y** 及模型 $\lambda = f\,(\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$，估计系统产生 **Y** 最可能的状态序列 **X**
  - HMM模型的概率计算
    - 前向概率
    - 后向概率

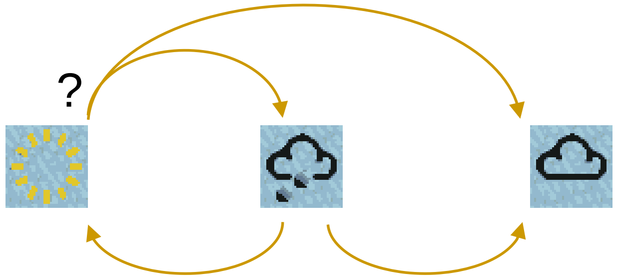
Before We Start …

Before We Start …

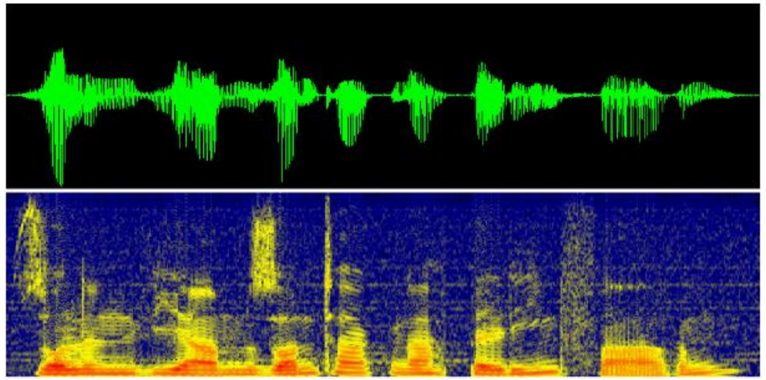


Before We Start …

Toss a coin: H H T H T T H T H H ? ? ? ?



The weather: sunny, rainy, cloudy, …



Speech: b aa m aa f o w …

Overview

Toss a coin: H H T H T T H T H H

The weather: sunny, rainy, cloudy

Speech: b aa m aa f o w

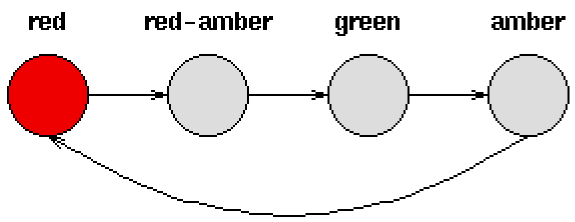
Sequence modeling

What are the underlying patterns?

Can we model such underlying patterns from observations?

Deterministic Patterns

- **The Example of Traffic Lights**



The sequence of lights is:
red – red/amber – green – amber – red

- The sequence of traffic lights is deterministic.
- Each state is dependent solely on the previous state.

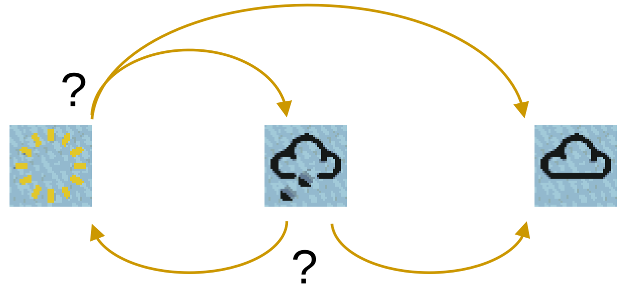
Deterministic State Machine

- **Deterministic State Machine**
  - A finite state machine where for each state, there is one and only one transition to a next state.
  - The transition between states is *deterministic*.

$$
\begin{array}{c|cccc}
 & red & red/amber & green & amber \\
\hline
red & 0 & 1 & 0 & 0 \\
red/amber & 0 & 0 & 1 & 0 \\
green & 0 & 0 & 0 & 1 \\
amber & 1 & 0 & 0 & 0
\end{array}
$$

Non-deterministic Patterns

- **The Weather Example**
  - Three states: sunny, cloudy, and rainy



The sequence of weather states *is NOT deterministic*

- **The Problem**
  - We *can not* expect these three weather states to follow each other *deterministically*.
  - But we still hope *to model the system* that generates a weather pattern.

Markov Assumption

- **Markov Assumption**
  - By assuming that the state of the model depends only upon the previous states of the model.

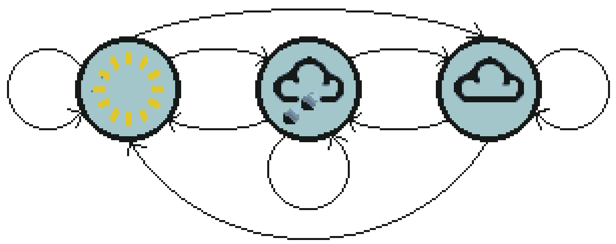
- **Constraints**
  - This may be a gross simplification, and much important *information may be lost* because of this assumption.

  - Nevertheless, since such simplified systems can be *subject to analysis*, we often accept the assumption in the knowledge that it may generate information that is not fully accurate.

Markov Process

- **Markov Process**

  - A Markov process is a process which moves from state to state *depending (only) on the previous n states*.
  - The process is called an *order n* model where *n* is the number of states affecting the choice of next state.
  - A *first order Markov process* is the simplest Markov process, where the choice of state is made purely on the basis of the *previous one* state.
  - The choice of the state transition is *to be made probabilistically*, not deterministically.

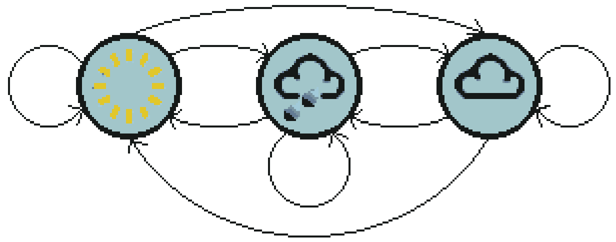


First Order Markov Process

- **First Order Markov Process**
  - A first order Markov process consists of
    - states
    - start probability
    - state transition matrix

States

- **The States**
  - All the possible states a Markov system could be for all the time.
  - For the weather example, the states are “{Sun, Could, Rain}”.



Sun Rain Could

Start Probability

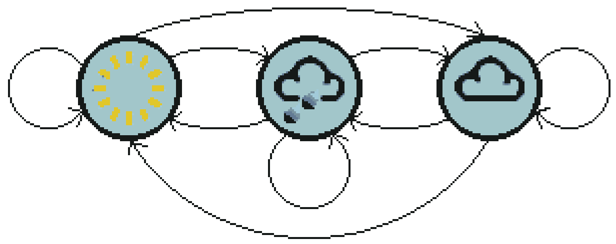
- **The Start State**
  - To initialize a Markov system, we need to state what the weather was (or probably was) on the day after creation.
  - A vector of *start probabilities*, called, the π *vector* is defined.

$$\begin{array}{ccc} \text{Sun} & \text{Cloud} & \text{Rain} \end{array}$$
$$\begin{pmatrix} 1.0 & 0.0 & 0.0 \end{pmatrix}$$

State Transition Probability /Matrix

- **State Transition Probability**
  - For a first order process with N states, there are N^2 transitions between states, as it is possible for any one state to follow another.
  - The *state transition probability* is associated with each transition, which is the probability of moving from one state to another

- **State Transition Matrix**
  - The N^2 probabilities may be collected together in an obvious way into a *state transition matrix*.



$$
\begin{array}{ll}
 & \quad Today \\
Yesterday & \begin{array}{l} \\ \text{sun} \\ \text{cloud} \\ \text{rain} \end{array}
\begin{bmatrix} \text{sun} & \text{cloud} & \text{rain} \\ 0.50 & 0.375 & 0.125 \\ 0.25 & 0.125 & 0.625 \\ 0.25 & 0.375 & 0.375 \end{bmatrix}
\end{array}
$$

Sum to 1

Assumption: The state transition probabilities do not vary in time.

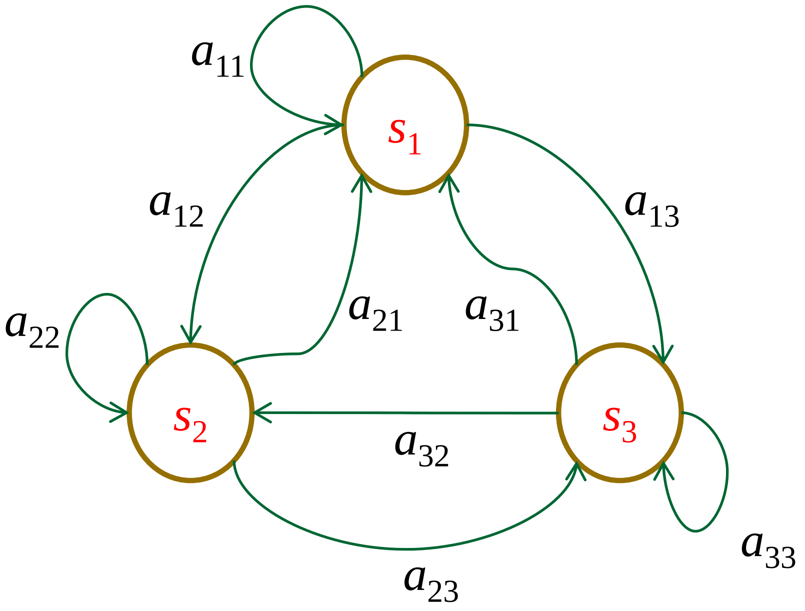
First Order Markov Process

- **First Order Markov Process**
  - A first order Markov process consists of
    - states
    - start probability
    - state transition matrix

Any system that can be described in the above manner is a Markov system.

A Markov System

- The system has N states: $\{s_1, s_2, \ldots, s_i, \ldots, s_N\}$.
- There are T discrete time steps, $t = 1, 2, \ldots, T$.
- On the t-th time step, the system is on exactly one of the states $s_i(t)$.
  - $s_i(t)$ means the system is on the state s_i at time t
  - The state sequence can be represented by $Q=\{q_1,q_2,\ldots,q_T\}$, where $q_t=s_i$
- Between two time steps, the next state is chosen probabilistically.
- The current state determines the probability for the next state.



Markov Property

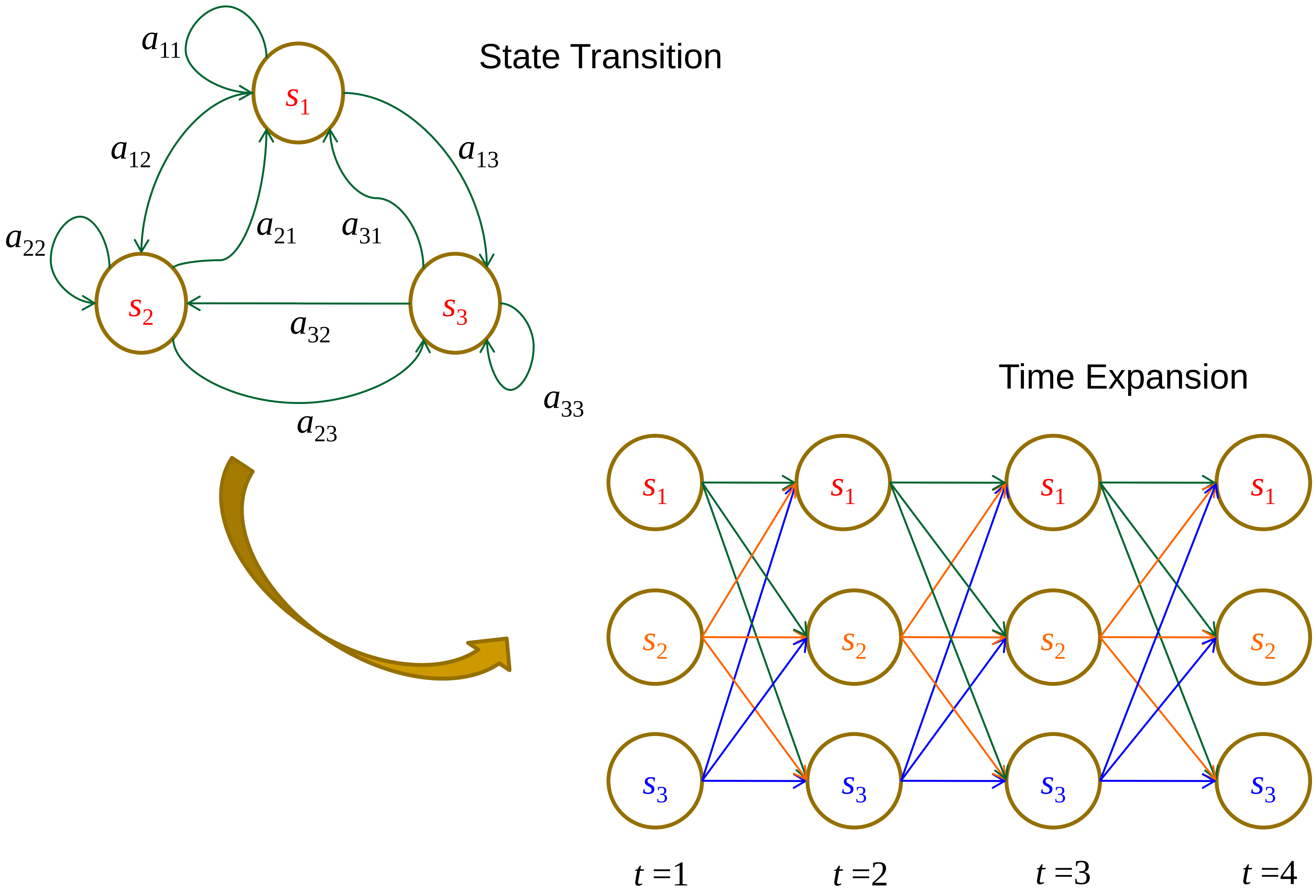
- State transition probability: $P[s_j(t+1) \mid s_i(t)] = P[q_{t+1}=s_j|q_t=s_i] = a_{ij}$
- a_{ij} is independent of time t.
- $s(t+1)$ is conditionally independent of $\{s(1), s(2), \dots, s(t-1)\}$ given $s(t)$.
  - $P[s_j(t+1) \mid s_i(t)] = P[s_j(t+1) \mid s_i(t)$, any earlier history$]$
- Start probability: $\boldsymbol{\pi} = [\pi_1, \pi_2, \dots, \pi_N]$
  - $\pi_n = P[s(1)=s_n] = P[q_1=s_n]$, $n = 1, 2, \dots, N$
- The probability of generating state sequence $\{s_1, s_3, s_2, s_2, s_1, s_3\}$
  - $P(s^6|\theta) = \pi_1\, a_{13}\, a_{32}\, a_{22}\, a_{21}\, a_{13}$

Sum to 1

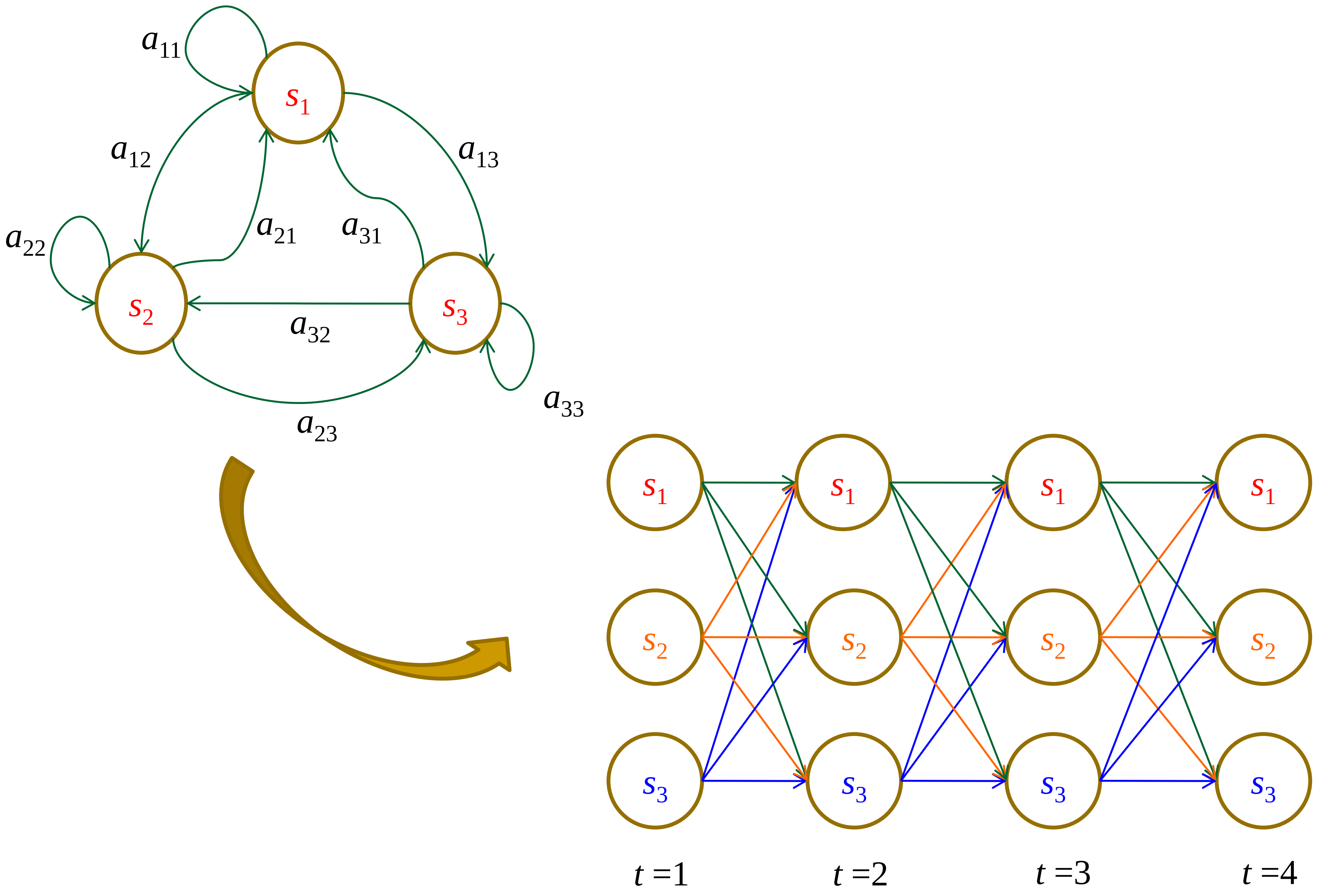
Transition Matrix

$$\begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix}$$

A Markov System: Cont.



Probability Calculation



Weather Example

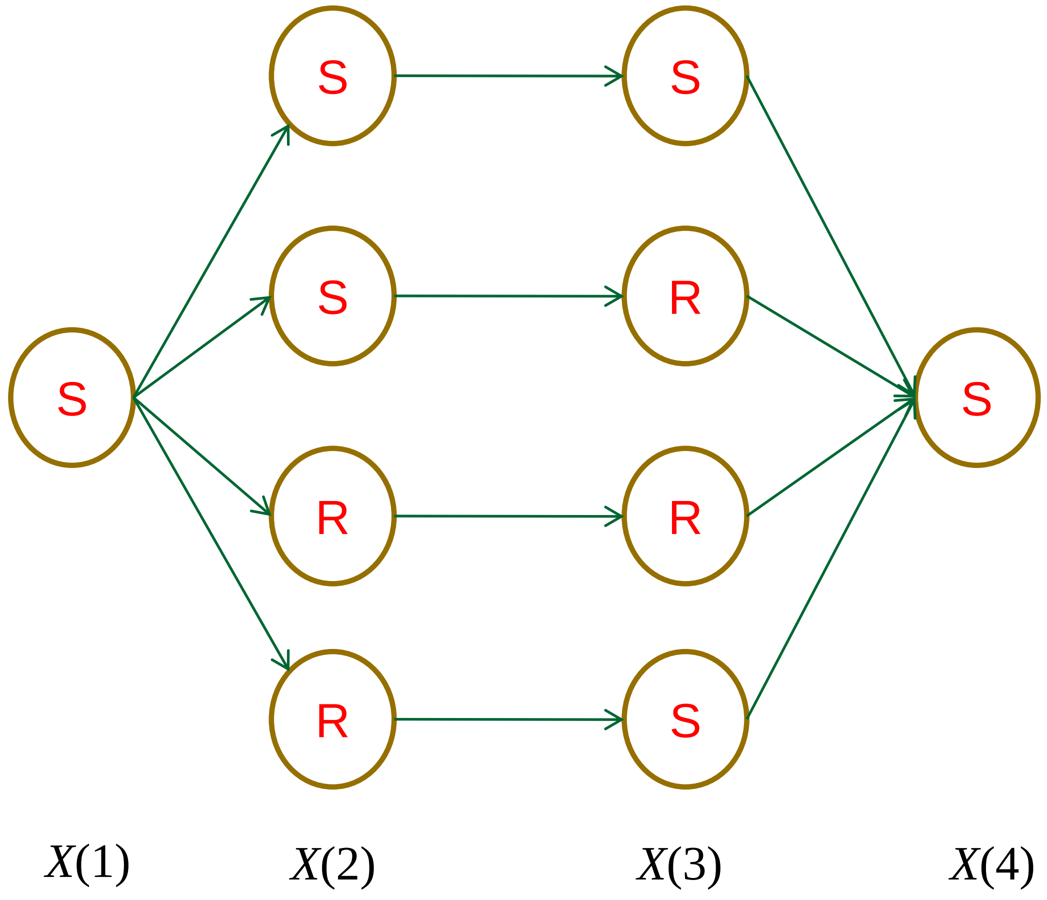
Q: What is the probability of being sunny at day 4?

$$X(1) = [1, 0]$$

$$P = \begin{array}{c|cc} & sunny & rainy \\ \hline sunny & 0.9 & 0.1 \\ rainy & 0.5 & 0.5 \end{array}$$

A Brute Force Method

Q: What is the probability of being sunny at day 4?



Complexity: $O(t \cdot N^t)$

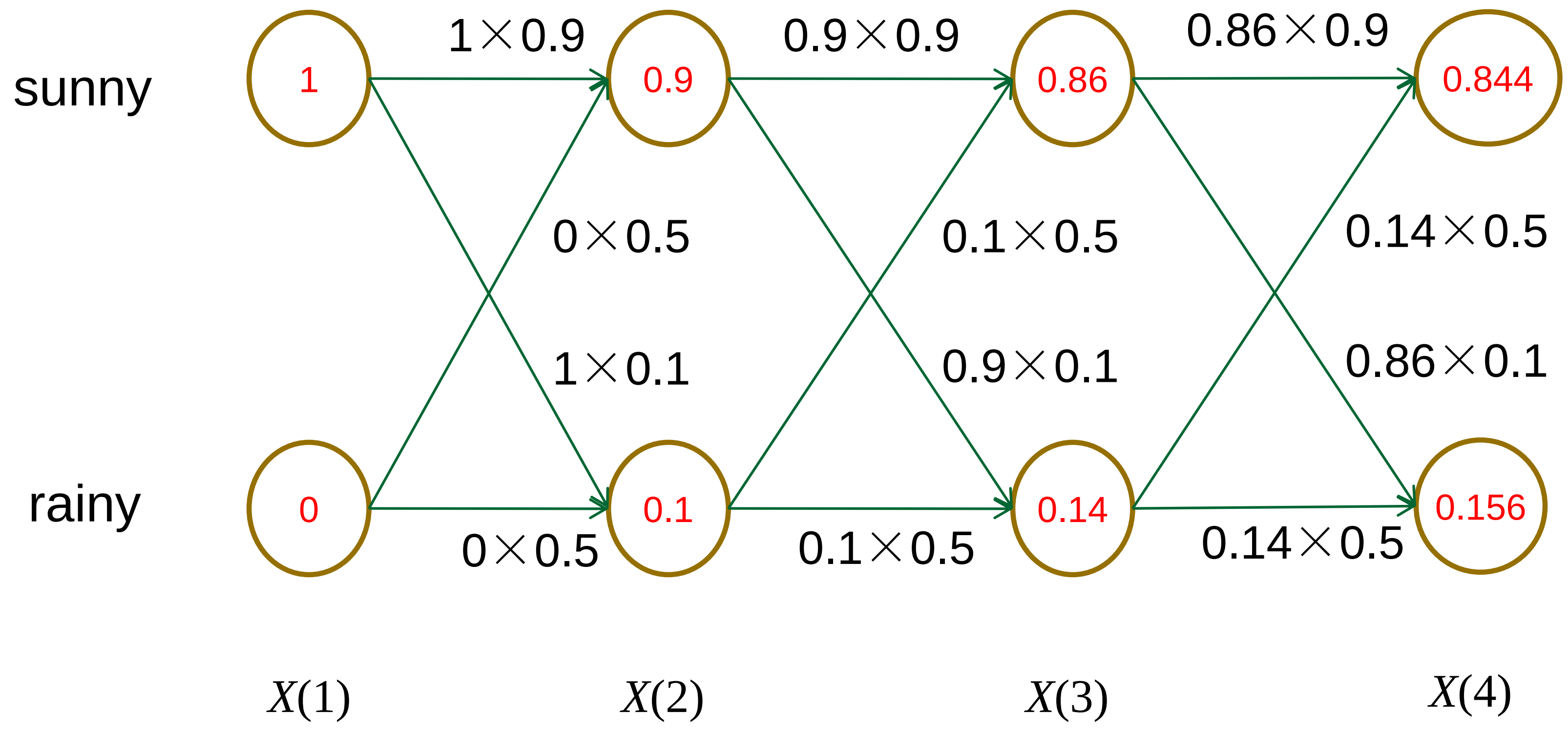
0.9×0.9×0.9=0.729
0.9×0.1×0.5=0.045
0.1×0.5×0.5=0.025
0.1×0.5×0.9=0.045

0.844

$P =$	sunny	rainy
sunny	0.9	0.1
rainy	0.5	0.5

A Smart Method

Q: What is the probability of being sunny at day 4?



Weather Example

Q: What is the probability of being sunny at day 4?

$$X(1) = [1, 0]$$

$$X(2) = X(1)P = [1, 0]\begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix} = [0.9, 0.1]$$

$$X(3) = X(2)P = [0.9, 0.1]\begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix} = [0.86, 0.14]$$

$$X(4) = X(3)P = X(2)P^2 = [0.9, 0.1]\begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix}\begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix} = [0.844, 0.156]$$

General rules for day t are:

$$X(t) = X(t-1)P$$

$$X(t) = X(1)P^{t-1}$$

$$P = \begin{array}{c|cc} & sunny & rainy \\ \hline sunny & 0.9 & 0.1 \\ rainy & 0.5 & 0.5 \end{array}$$

Complexity: $O(t \cdot N^2)$

Limitations of a Markov Process

The states of the Markov process might be hidden

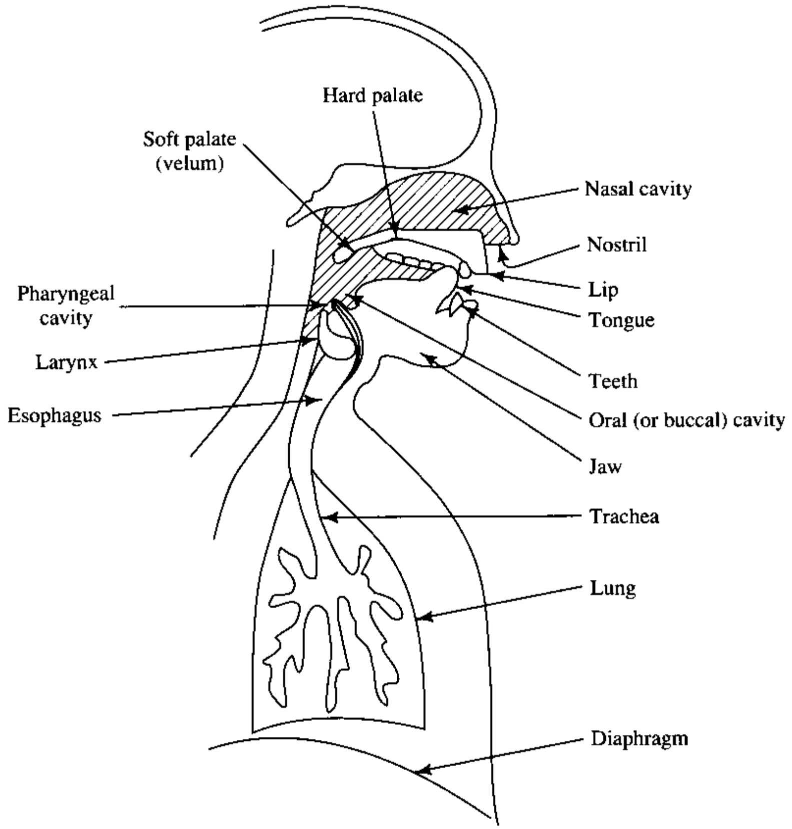
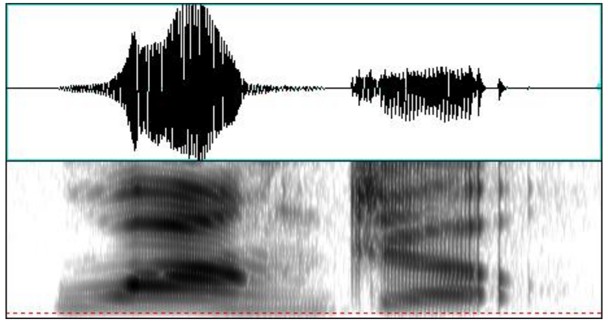
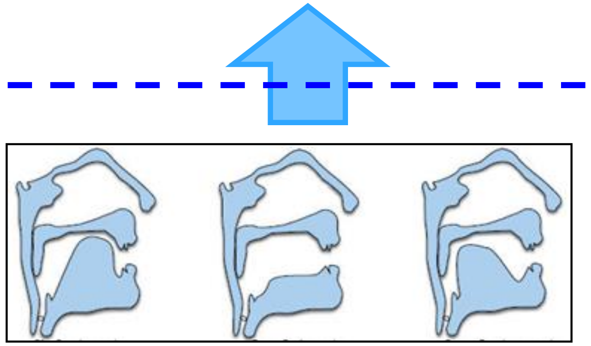


Diagram of the articulators (speech organs)

- Speech generation and recognition example



Observation
the
Heard Sound



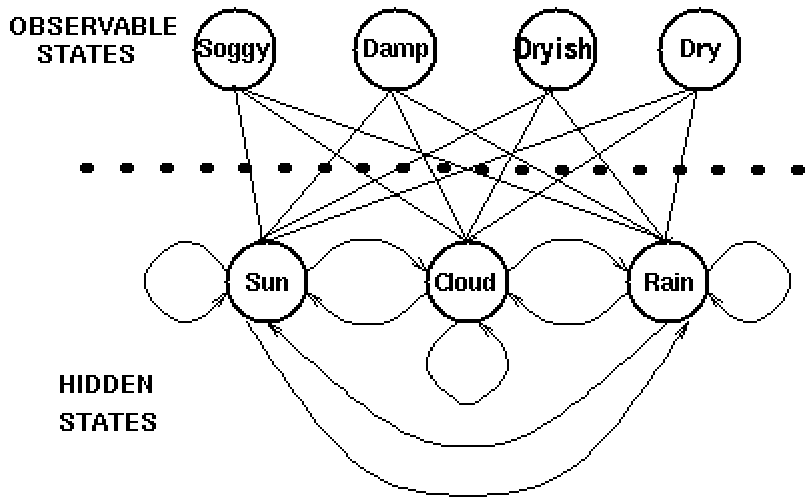
Hidden States
of the
Articulators

(from: http://clas.mq.edu.au/phonetics/phonetics/introduction/index.html)

The States May be Hidden…

- **The Weather Example**
  - We have two sets of states
    - the *observable states*, and
    - the *hidden states*

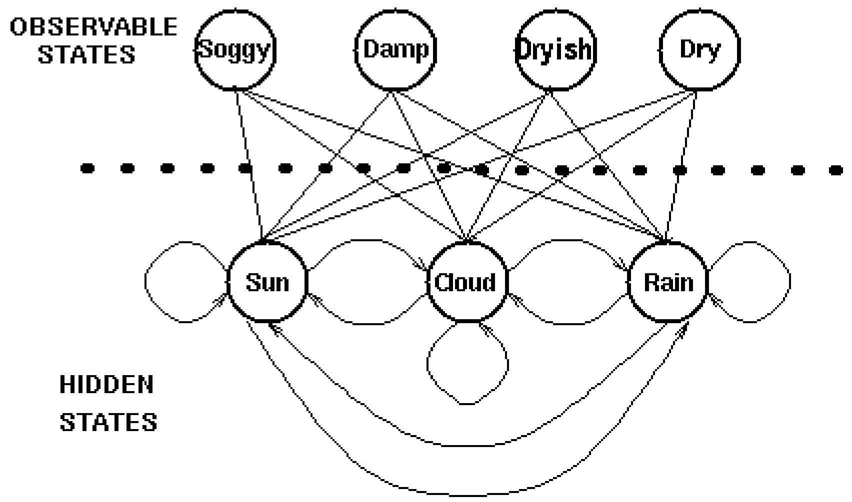
The number of the hidden states and the number of observable states might be different.



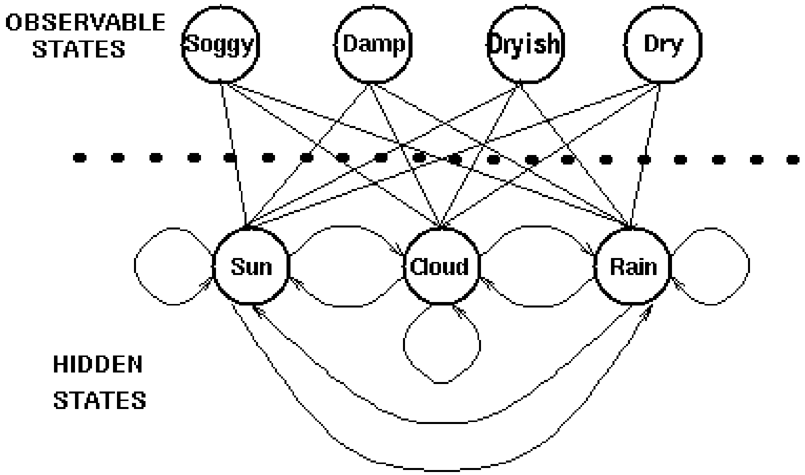
We wish to devise an algorithm to forecast weather (*the hidden states*) from the seaweed (*the observation states*) and the Markov assumption without actually ever seeing the weather.

HMM: Hidden Markov Model

- **HMM**
  - There is an underlying *hidden Markov process* changing over time.
  - The observable states are *probabilistically* related to the hidden states.



HMM: Parameter Description



- **Two sets of states**
  - Hidden states
  - Observable states

- **Two parts**
  - Hidden part
    - First order Markov process
      - Start probability
      - State transition probability
  - Observable part
    - Observations
      - Emission probability

HMM: Emission Probability / Matrix

- **Emission Probability**
  - The probabilities of the observable state v_k given a particular hidden state s_j:
    $P(v_k \mid s_j)$, $j=1,\ldots,N$; $k=1,\ldots,M$
  - N hidden states
  - M observable states

- **Emission Matrix**
  - The $N \times M$ probabilities may be collected together in an obvious way into a *emission matrix*.

Seaweed

weather		Dry	Dryish	Damp	Soggy
	Sun	0.60	0.20	0.15	0.05
	Cloud	0.25	0.25	0.25	0.25
	Rain	0.05	0.10	0.35	0.50

Sum to 1

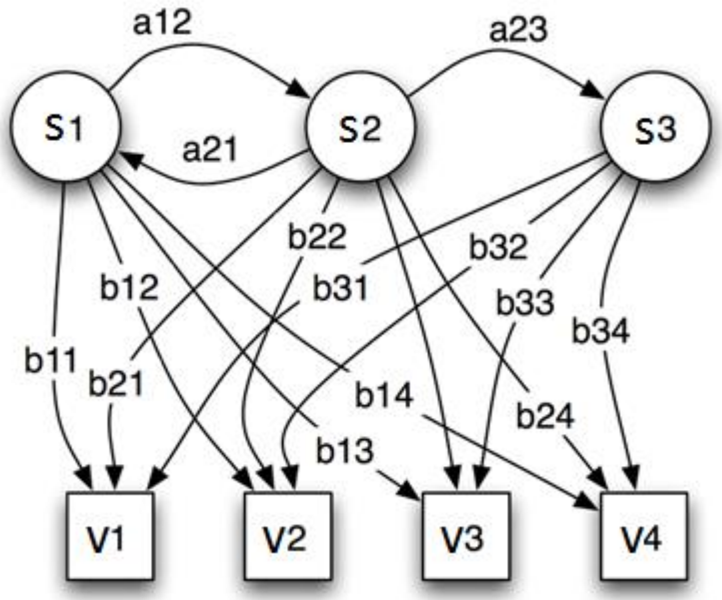
HMM: Probabilistic Parameters

- **Two sets of states**
  - hidden states

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

  - observations

$$\{v_1, v_2, \ldots, v_M\}$$



- **Three sets of probabilities**
  - start probability

$$\boldsymbol{\pi} = \{\pi_i, 1 \le i \le N\}$$

  - state transition probability

$$\mathbf{A} = \{a_{ij}\} \qquad a_{ij} = P\left(s_j(t) \mid s_i(t-1)\right) \qquad 1 \le i, j \le N$$

  - emission probability

$$\mathbf{B} = \{b_{jk}\} \qquad b_{jk} = P\left(v_k \mid s_j\right) \qquad 1 \le j \le N \quad 1 \le k \le M$$

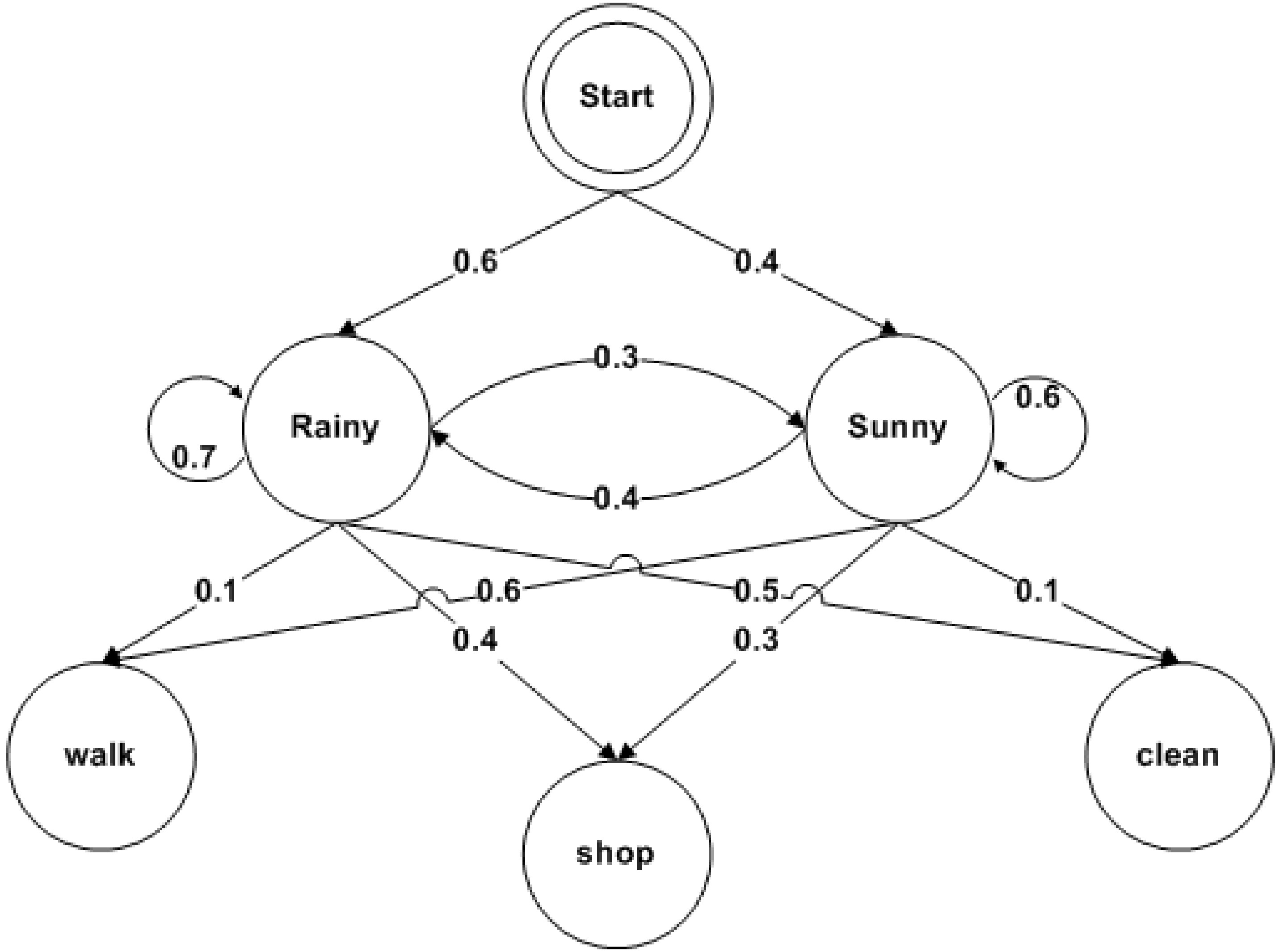
Each probability in the state transition matrix and emission matrix *is time independent*.

HMM: A Concrete Example

- **states = ('Rainy', 'Sunny')**
- **observations = ('walk', 'shop', 'clean')**
- **start probability = {'Rainy': 0.6, 'Sunny': 0.4}**
- **state transition probability = {**
  - 'Rainy' : {'Rainy': 0.7, 'Sunny': 0.3},
  - 'Sunny' : {'Rainy': 0.4, 'Sunny': 0.6},

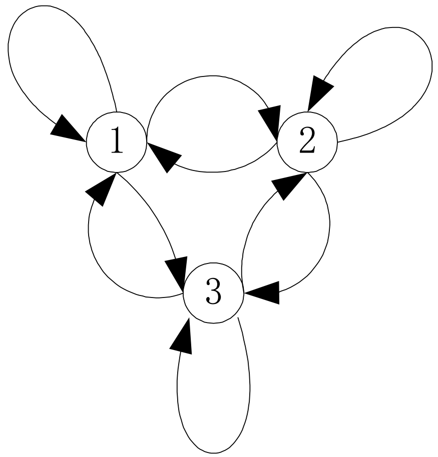
    }
- **emission probability = {**
  - 'Rainy' : {'walk': 0.1, 'shop': 0.4, 'clean': 0.5},
  - 'Sunny' : {'walk': 0.6, 'shop': 0.3, 'clean': 0.1},

    }

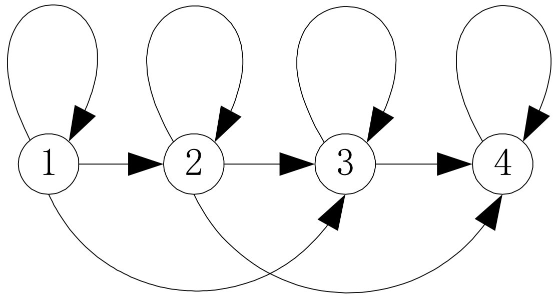


(Adapted from: Prof. Yuan Bo's lecture notes)

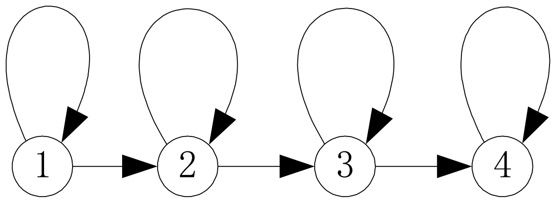
HMM Structures



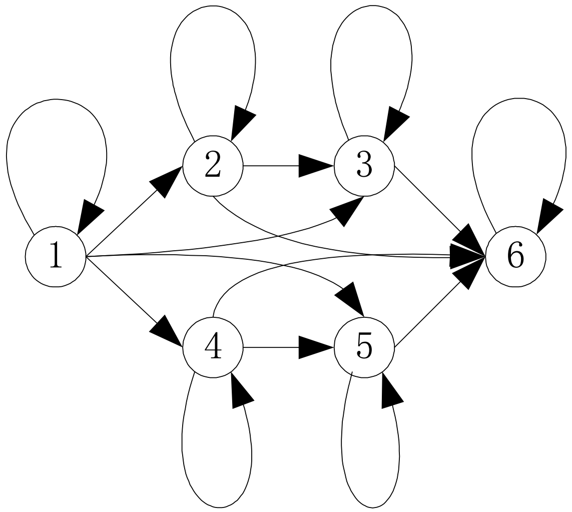
(a) 全连结模型



(c) 有跨越从左向右模型



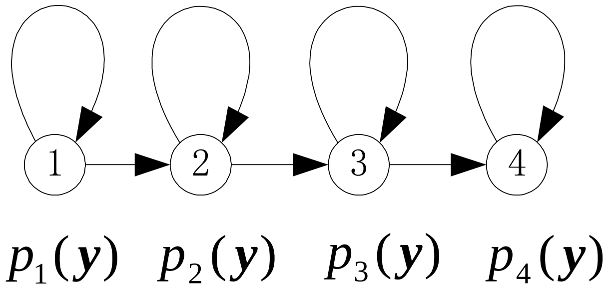
(b) 无跨越从左向右模型



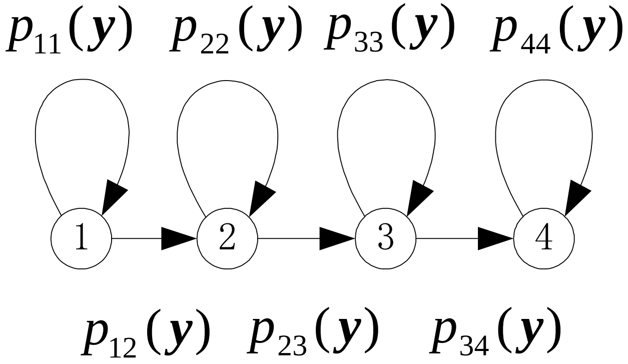
(d) 并行从左向右模型

HMM Types

- **State Emitting HMM**
- **State Transition Emitting HMM**

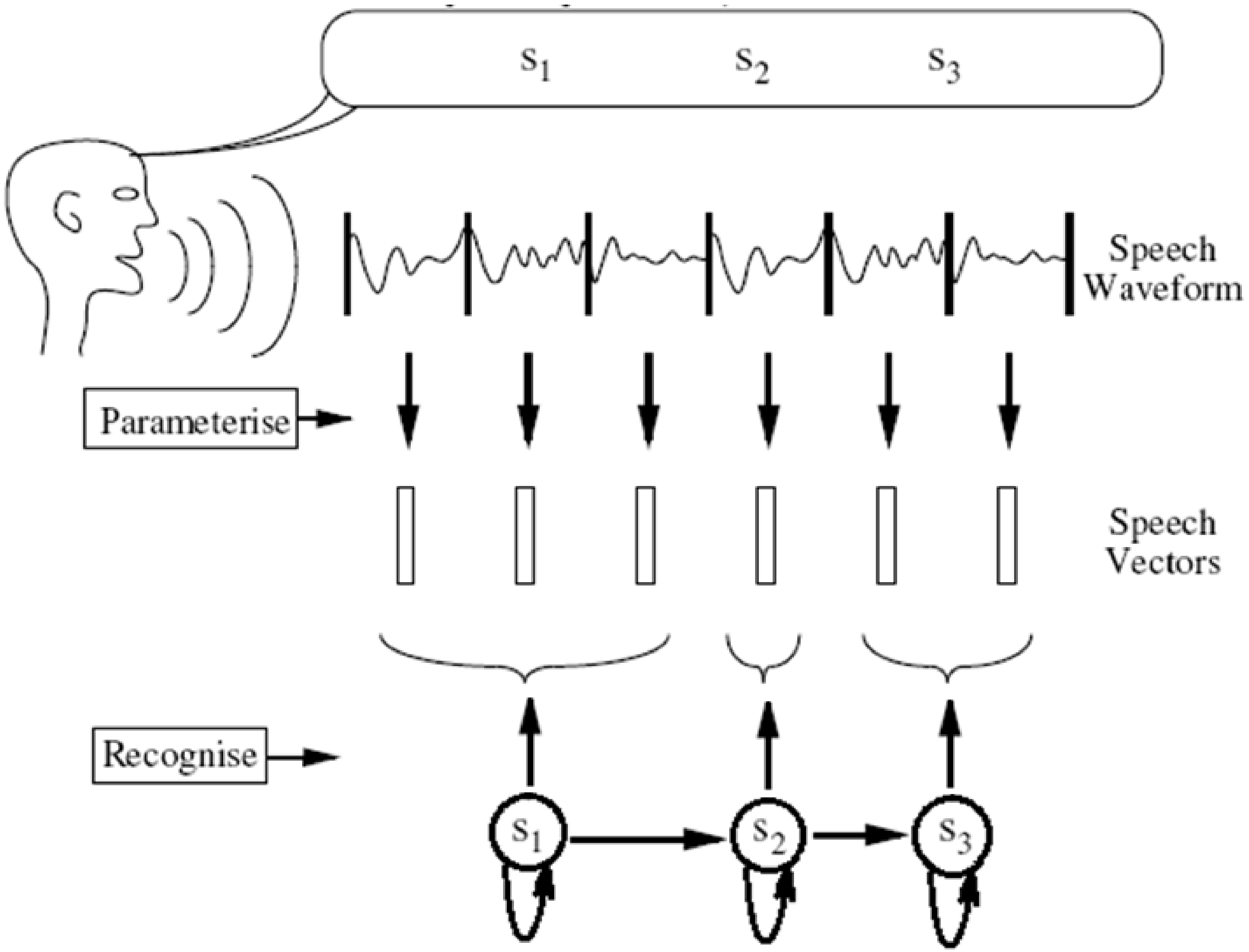


(a) 用4个状态输出概率描述



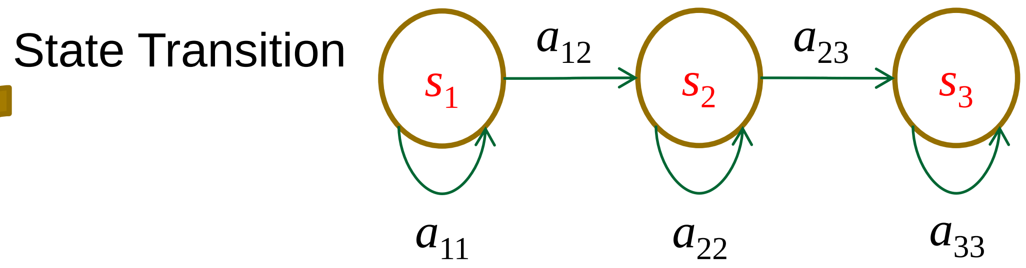
(b) 用7个转移弧输出概率描述

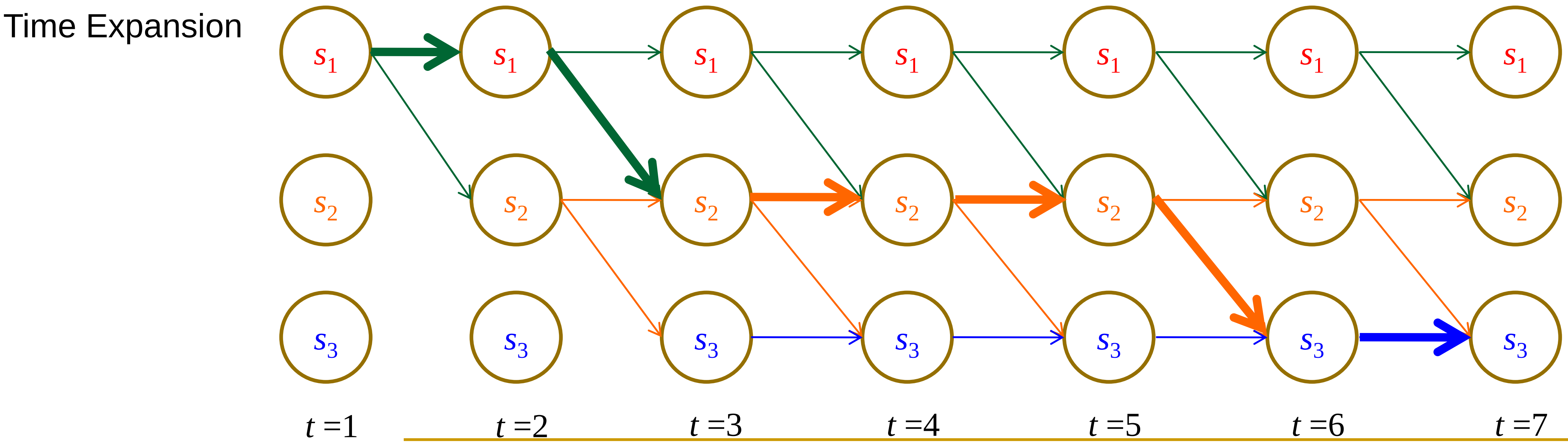
HMM: The Speech Example



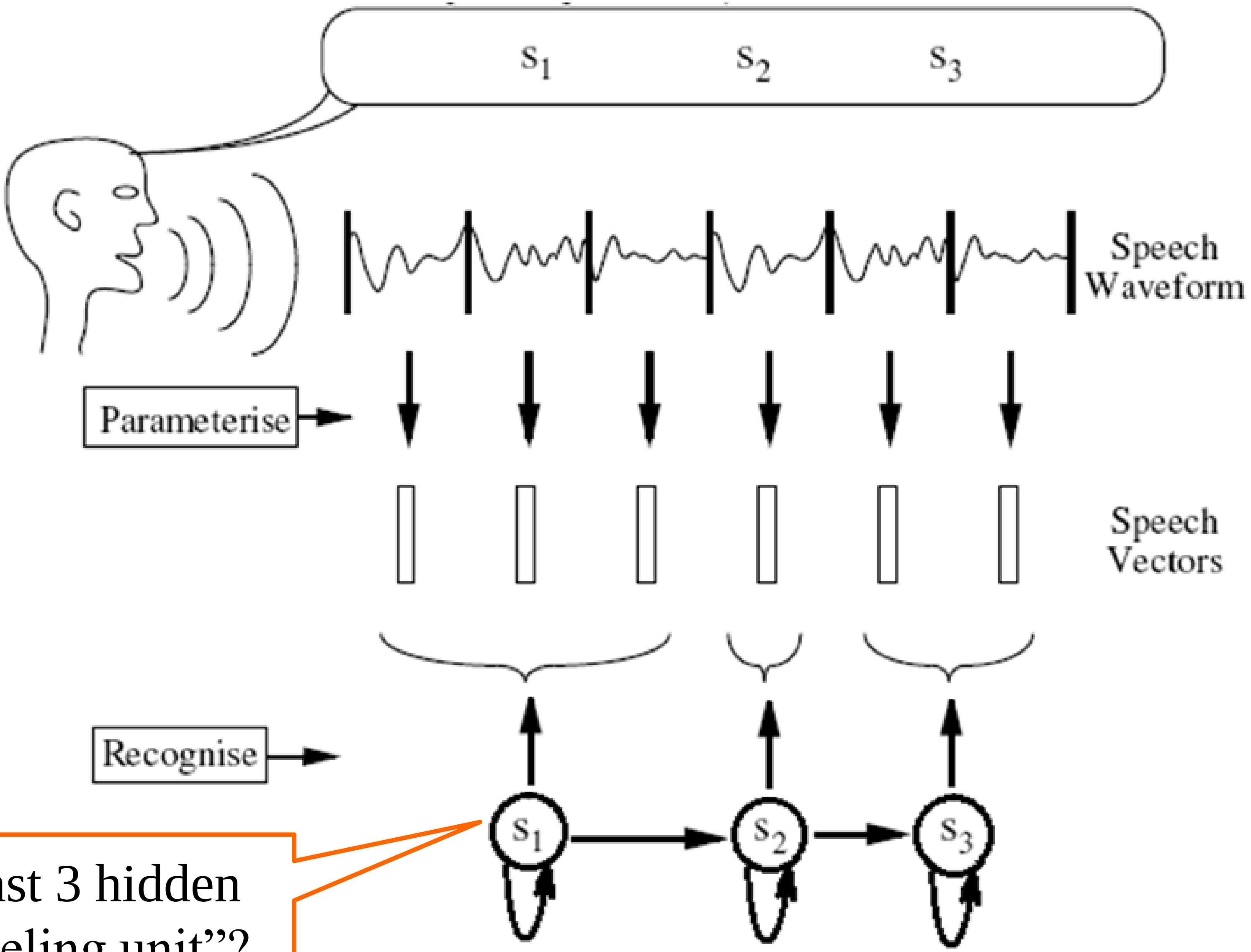
(Adapted from: HTK Book)

HMM: The Speech Example





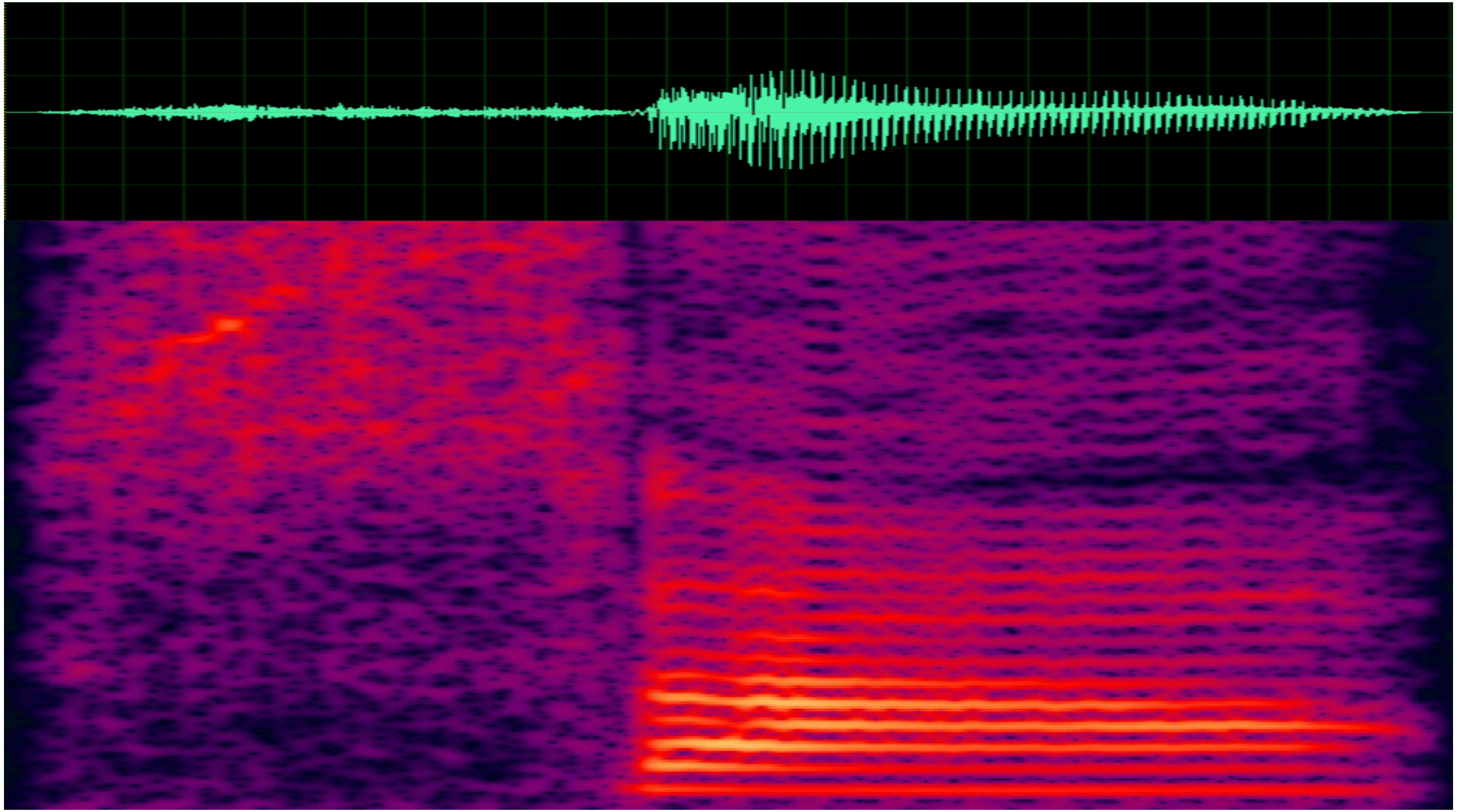
HMM: The Speech Example



Why using at least 3 hidden states for a "modeling unit"?

(Adapted from: HTK Book)

HMM: The Speech Example



Three Canonical Questions

- **Q1 (Evaluation):** Given the parameters of the model, compute the probability of a particular output sequence.
  - Given $O=o_1o_2\ldots o_T$, and $\lambda=(\pi, A, B)$, how to compute the probability $P(O|\lambda)$?

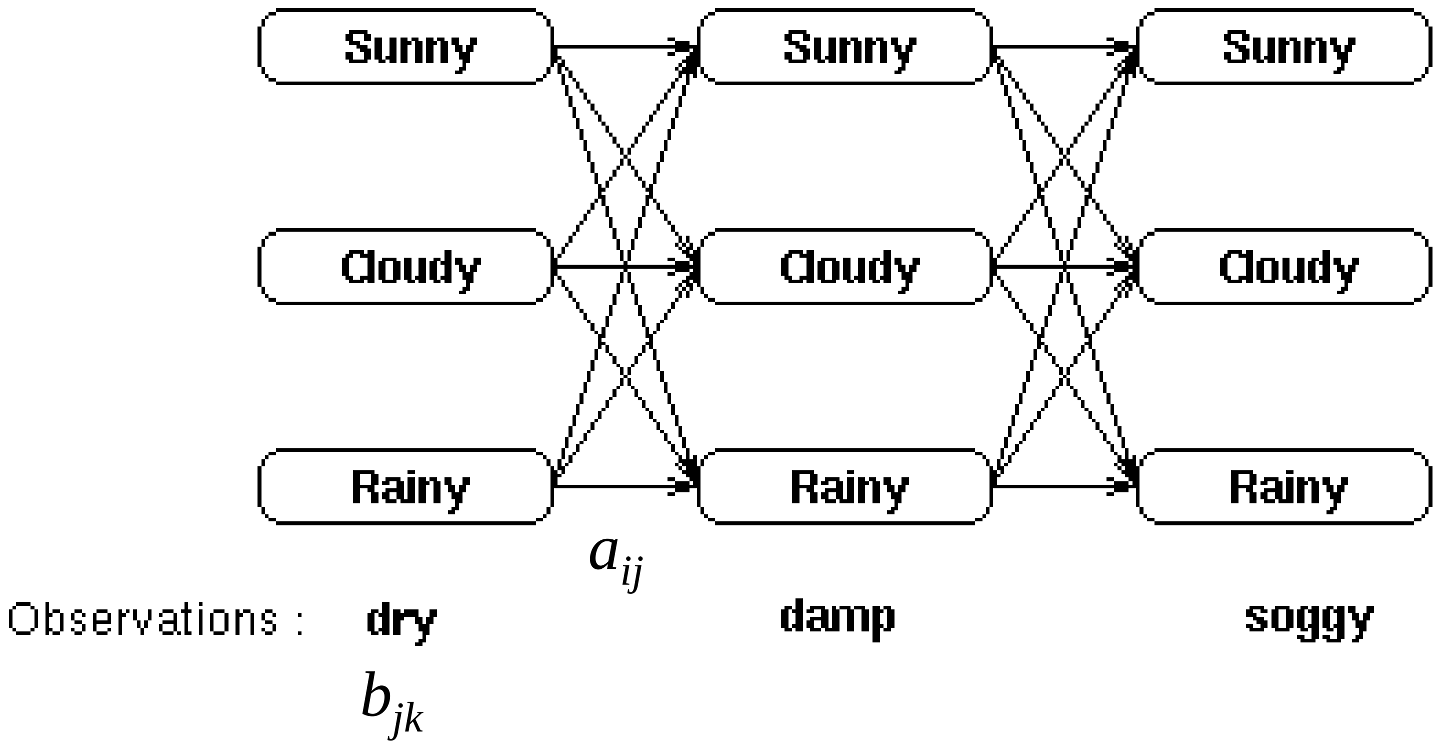
- **Q2 (Decoding):** Given the parameters of the model and a particular output sequence, find the state sequence that is most likely to have generated that output sequence.
  - Given $O=o_1o_2\ldots o_T$, and $\lambda=(\pi, A, B)$, how can the hidden states $Q=q_1q_2\ldots q_T$ be computed, which is optimal in some sense, i.e., best explains O?

- **Q3 (Learning):** Given a set of output sequences, estimate the parameters of the model that can most probably generate the output.
  - How to estimate the parameters of the HMM model?

- All questions require calculations over all possible state sequences.
- Fortunately, they can be solved more efficiently.

Q1: Exhaustive Search

- Given $O=o_1o_2\ldots o_T$, and $\lambda=(\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$, how to compute the probability $P(O|\lambda)$



Q1: Exhaustive Search

$$O = o_1 o_2 \ldots o_T\,,\ \lambda = \{\boldsymbol{\pi}, \mathbf{A}, \mathbf{B}\} \ \Rightarrow\ P(O \mid \lambda)\ ?$$

- Consider one fixed state sequence Q of length T:

$$Q = q_1 q_2 \ldots q_T$$

- The probability of such a Q is:

$$P(Q \mid \lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T}$$

- Assuming statistical independence of observations:

$$P(O \mid Q, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda) = b_{q_1}(o_1) b_{q_2}(o_2) \cdots b_{q_T}(o_T)$$

- Since the joint probability of O and Q is:

$$P(O, Q \mid \lambda) = P(O \mid Q, \lambda) P(Q \mid \lambda)$$

- The sum over all state sequence gives the solution:

$$P(O \mid \lambda) = \sum_{all\ Q} P(O, Q \mid \lambda) = \sum_{all\ Q} P(O \mid Q, \lambda) P(Q \mid \lambda)$$

$$= \sum_{q_1 q_2 \ldots q_T} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \cdots a_{q_{T-1} q_T} b_{q_T}(o_T)$$

Q1: Exhaustive Search

$$
\begin{aligned}
P(O|\lambda) &= \sum_{all\ Q} P(O,Q|\lambda) \\
&= \sum_{all\ Q} P(O\,|\,Q,\lambda)P(Q\,|\,\lambda) \\
&= \sum_{q_1 q_2 \ldots q_T} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \cdots a_{q_{T-1} q_T} b_{q_T}(o_T)
\end{aligned}
$$

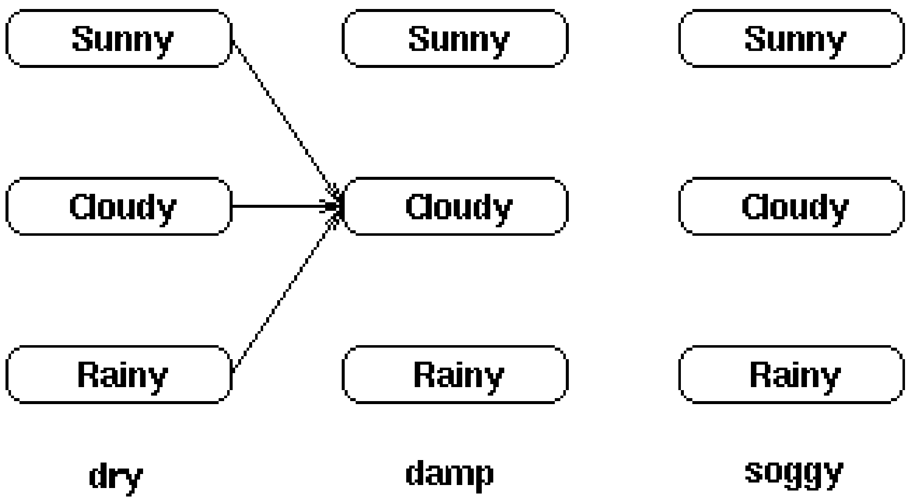
$$q_t \in \{s_1, s_2, \ldots, s_N\},\ t = 1, 2, \ldots, T$$

- **Computational Complexity**

$$O(2T \cdot N^T)$$

Q1: A Smart Way

- We can calculate the probability of reaching *an intermediate state* in the trellis as the sum of all possible paths to that state.
- The probability of being cloudy at t=2 is calculated from the paths:

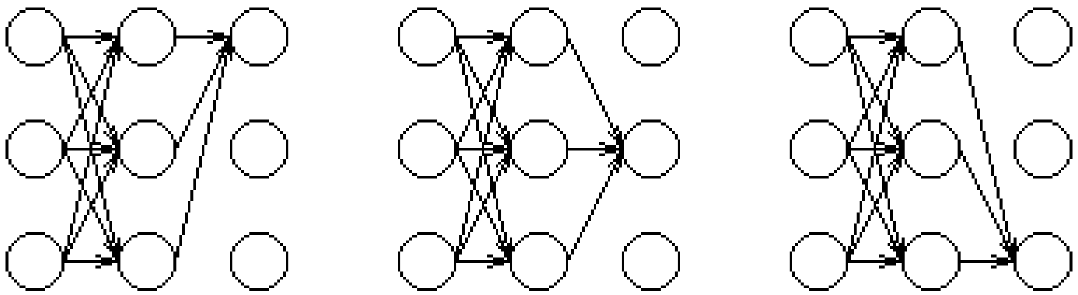


- Partial probability:

$$\alpha_t(j) = P(\text{observation} \mid \text{hidden state is } s_j) \\ \times P(\text{all paths to state } s_j \text{ at time } t)$$

Q1: A Smart Way

- The partial probabilities for *the final observation* hold the probability of reaching those states *going through all possible paths*.

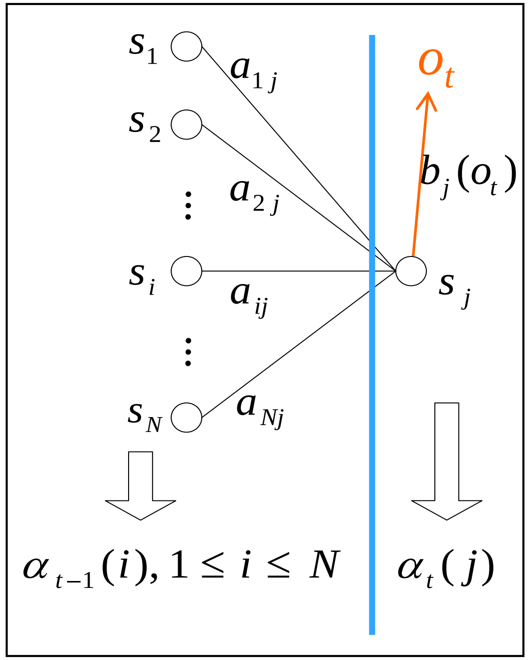


- The sum of these final partial probabilities is the probability of observing the observation sequence given the HMM.

Q1: Forward Probability

- **Forward Probability**
  - The probability of the partial observation sequence $o_1 o_2 \ldots o_t$ (time t included) and an intermediate state s_j at time step t, given the model:

$$\alpha_t(j) = P(o_1 o_2 \cdots o_{t-1} o_t, q_t = s_j \mid \lambda)$$



  - At time t=1:

$$\alpha_1(j) = \pi_j b_j(o_1), \ 1 \le j \le N$$

  - At time step t, $\alpha_t(j)$ can be solved inductively from time t-1:

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t), \ 1 \le j \le N, \ 2 \le t \le T$$

Q1: Forward Algorithm

- **The solution to Q1:**
  - Initialization: $\alpha_1(j) = \pi_j b_j(o_1),\ 1 \le j \le N$
  - Induction: $\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t),\ 1 \le j \le N,\ 2 \le t \le T$
  - Q1 solution: $P(O \mid \lambda) = \sum_{j=1}^{N} \alpha_T(j)$
- **Computational Complexity**

$$O(T \cdot N^2)$$

Q1: Forward Algorithm

```
function forward
begin
    // initialization
    t = 1;
    for j = 1 to N do begin
        α_t(j) = π_j b_j(o_t);
    end
    // induction
    for t = 2 to T do begin
        for j = 1 to N do begin
            α_t(j) = 0;
            for i = 1 to N do begin
                α_t(j) = α_t(j) + α_{t-1}(i) a_ij b_j(o_t);
            end
        end
    end
    // evaluation
    P = 0;
    for j = 1 to N do begin
        P = P + α_T(j);
    end
    return P;
end
```

Q1: The Weather Example

- Hidden states (weather) = {Sunny, Cloudy, Rainy}
- Observed states (seaweed) = {Dry, Dryish, Damp, Soggy}
- Start probability =

Sunny	Cloudy	Rainy
0.63	0.17	0.20

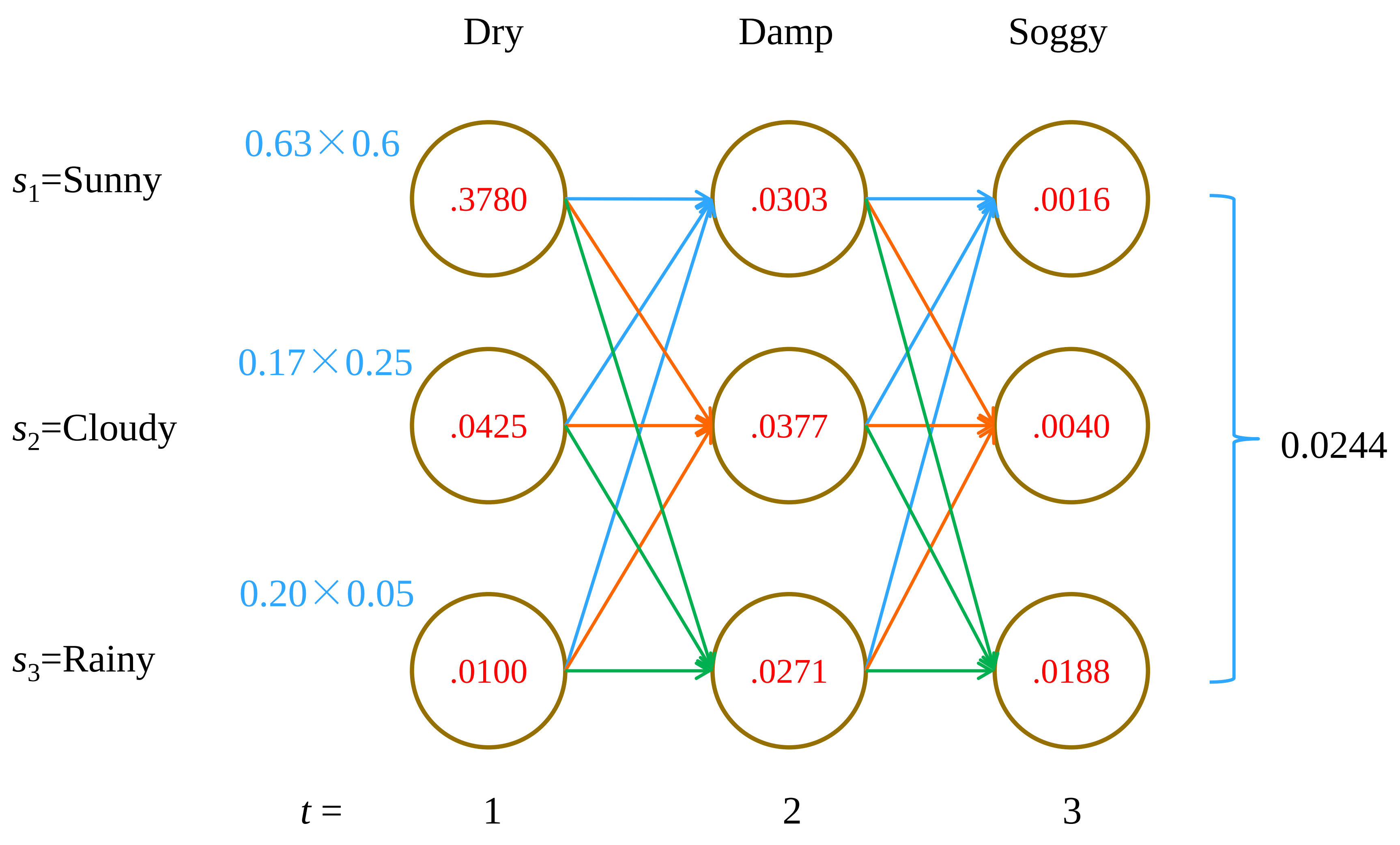
- State transition probability =

	Sunny	Cloudy	Rainy
Sunny	0.500	0.375	0.125
Cloudy	0.250	0.125	0.625
Rainy	0.250	0.375	0.375

- Emission probability =

	Dry	Dryish	Damp	Soggy
Sunny	0.60	0.20	0.15	0.05
Cloudy	0.25	0.25	0.25	0.25
Rainy	0.05	0.10	0.35	0.50

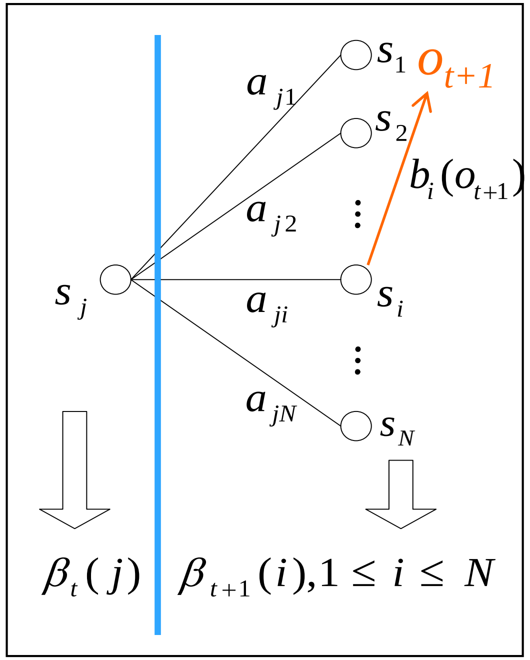
Q1: The Weather Example



Q1: Backward Probability

- **Another way to solve Q1**

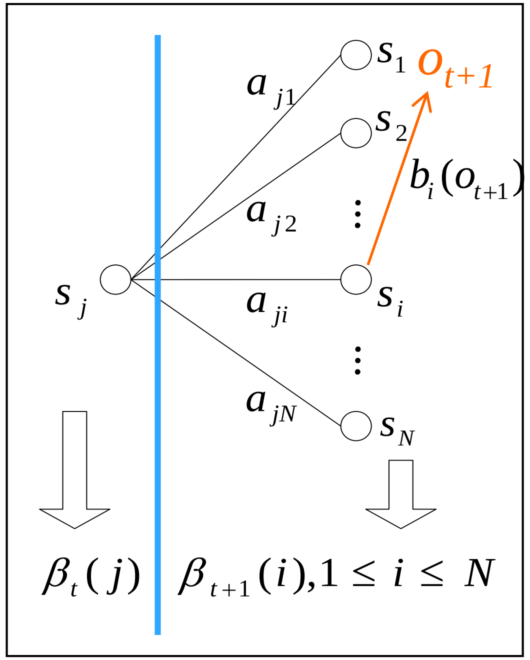
- **Backward probability**



Q1: Backward Probability

- **Backward Probability**
  - The probability of the partial observation sequence from t+1 to the end $o_{t+1}o_{t+2}\ldots o_T$ (time t excluded), given state s_j at time t and the model λ:

$$\beta_t(j) = P(o_{t+1}o_{t+2}\cdots o_T \mid q_t = s_j, \lambda)$$



  - At time t=T:

$$\beta_T(j) = 1,\ 1 \le j \le N$$

  - At time step t, $\beta_t(j)$ can be solved inductively from time t+1:

$$\beta_t(j) = \sum_{i=1}^{N} a_{ji} b_i(o_{t+1}) \beta_{t+1}(i),\ 1 \le j \le N,\ t = T-1, T-2, \ldots, 1$$

Q1: Backward Algorithm

- **Computing the Backward Probability:**

  - Initialization: $\beta_T(j) = 1, \ 1 \le j \le N$

  - Induction: $\beta_t(j) = \sum_{i=1}^{N} a_{ji} b_i(o_{t+1}) \beta_{t+1}(i),$

    $1 \le j \le N, \ t = T-1, T-2, \ldots, 1$

  - Induction part is the heart of the backward calculation
  - Can be computed in a trellis structure similar to that used for forward probability α's
  - However, α's are computed recursively from left to right, while β's from right to left

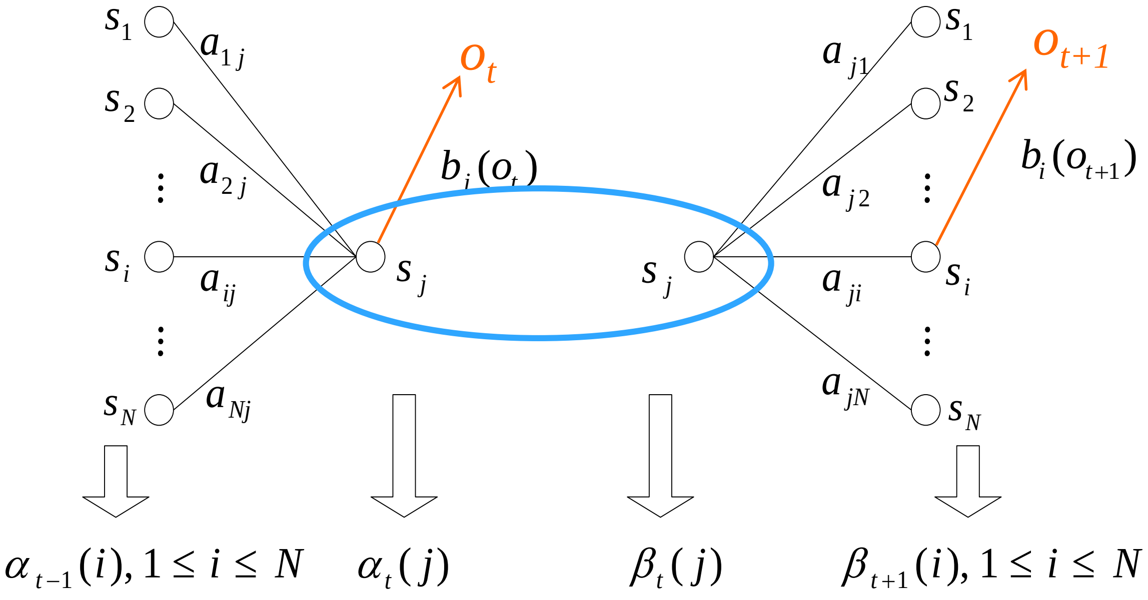
- **Computational Complexity**

$$O(T \cdot N^2)$$

Q1: Backward Algorithm

```
function backward
begin
    // initialization
    t = T;
    for j = 1 to N do begin
        β_t( j) = 1;
    end
    // induction
    for t = T-1 to 1 do begin
        for j = 1 to N do begin
            β_t( j) = 0;
            for i =1 to N do begin
                β_t( j) = β_t( j) + a_ji b_i (o_t+1) β_t+1 (i);
            end
        end
    end
    // returning the backward probabilities
    return β_t( j) (t = 1,…,T, j = 1,…,N);
end
```

Q1: Revisiting …



(a) Forward Probability

(b) Backward Probability

$P(O, q_t = s_j \mid \lambda)$

$$P(O \mid \lambda) = \sum_{j=1}^{N} P(O, q_t = s_j \mid \lambda)$$

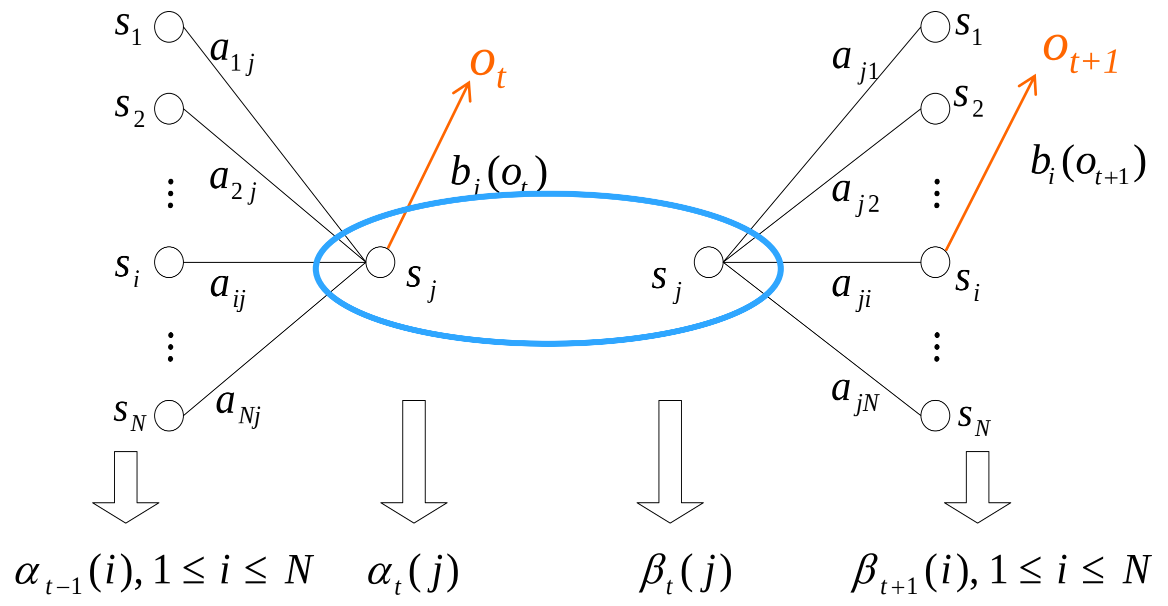
$\alpha_t(j)\beta_t(j)$

$$\alpha_t(j)\beta_t(j) = P(o_1 o_2 \cdots o_t, q_t = s_j \mid \lambda) P(o_{t+1} o_{t+2} \cdots o_T \mid q_t = s_j, \lambda)$$

$$P(O \mid \lambda) = \sum_{j=1}^{N} P(O, q_t = s_j \mid \lambda) = \sum_{j=1}^{N} \alpha_t(j)\beta_t(j)$$

Q1: Combining Forward Backward Probabilities

- Computing $P(O|\lambda)$ with Forward and Backward Probabilities



(a) Forward Probability

(b) Backward Probability

$$P(O|\lambda) = \sum_{j=1}^{N} \alpha_t(j)\beta_t(j) = \sum_{j=1}^{N} \alpha_{t-1}(j)\beta_{t-1}(j)$$

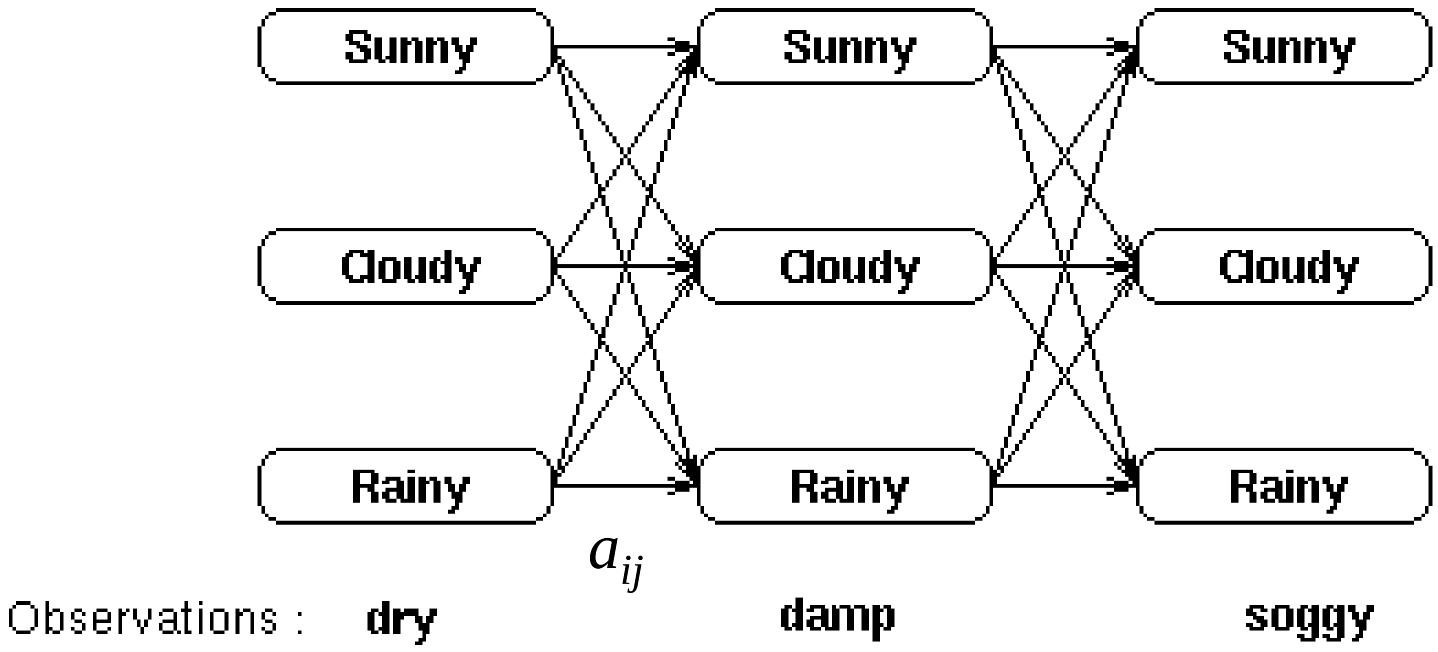
$$P(O|\lambda) = \sum_{j=1}^{N} \alpha_T(j)\beta_T(j) = \sum_{j=1}^{N} \alpha_T(j)$$

Computation is independent of time t

$$P(O|\lambda) = \sum_{j=1}^{N} \alpha_1(j)\beta_1(j) = \sum_{j=1}^{N} \pi_j b_j(o_1)\beta_1(j)$$

Q2: Exhaustive Search

- Given $O=o_1o_2\dots o_T$, and $\lambda=(\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$, how can the hidden states $Q=q_1q_2\dots q_T$ be computed, which is optimal in some sense, i.e., best explains O



Q2: Exhaustive Search

$$O = o_1 o_2 \ldots o_T\,,\ \lambda = \{\boldsymbol{\pi}, \mathbf{A}, \mathbf{B}\} \ \Rightarrow\ Q = q_1 q_2 \ldots q_T ?$$

- The best state sequence Q given the observation O and the model λ is:

$$\hat{Q} = \arg\max_{Q} P(Q|O,\lambda)$$

- Applying the Bayes' theorem, and considering the denominator is the same for all possible state sequence:

$$\arg\max_{Q} P(Q|O,\lambda) = \arg\max_{Q} \frac{P(Q,O\,|\,\lambda)}{P(O\,|\,\lambda)} = \arg\max_{Q} P(Q,O\,|\,\lambda)$$

- The P(O,Q| λ) can be computed as:

$$\begin{aligned} P(O,Q|\lambda) &= P(O\,|\,Q,\lambda)P(Q\,|\,\lambda) \\ &= \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \cdots a_{q_{T-1} q_T} b_{q_T}(o_T) \end{aligned}$$

- The best state sequence for the solution of Q2 is:

$$\begin{aligned} \hat{Q} &= \arg\max_{Q} P(O,Q\,|\,\lambda) \\ &= \arg\max_{Q} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \cdots a_{q_{T-1} q_T} b_{q_T}(o_T) \end{aligned}$$

Q2: Exhaustive Search

$$\hat{Q} = \arg\max_{all\ Q} P(O,Q|\lambda)$$
$$= \arg\max_{all\ Q} P(O \mid Q,\lambda)P(Q \mid \lambda)$$
$$= \arg\max_{q_1 q_2 \ldots q_T} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \cdots a_{q_{T-1} q_T} b_{q_T}(o_T)$$

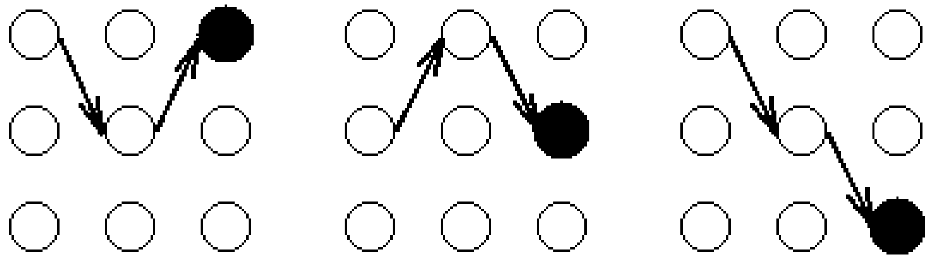
$$q_t \in \{s_1, s_2, \ldots, s_N\},\ t = 1, 2, \ldots, T$$

- **Computational Complexity**

$$O(2T \cdot N^T)$$

Q2: A Smart Way

- For each intermediate or terminating state in the trellis, there is *a most probable path* to that state.
  - Each of the three states at t = 3 will have a most probable path to it, perhaps like this:



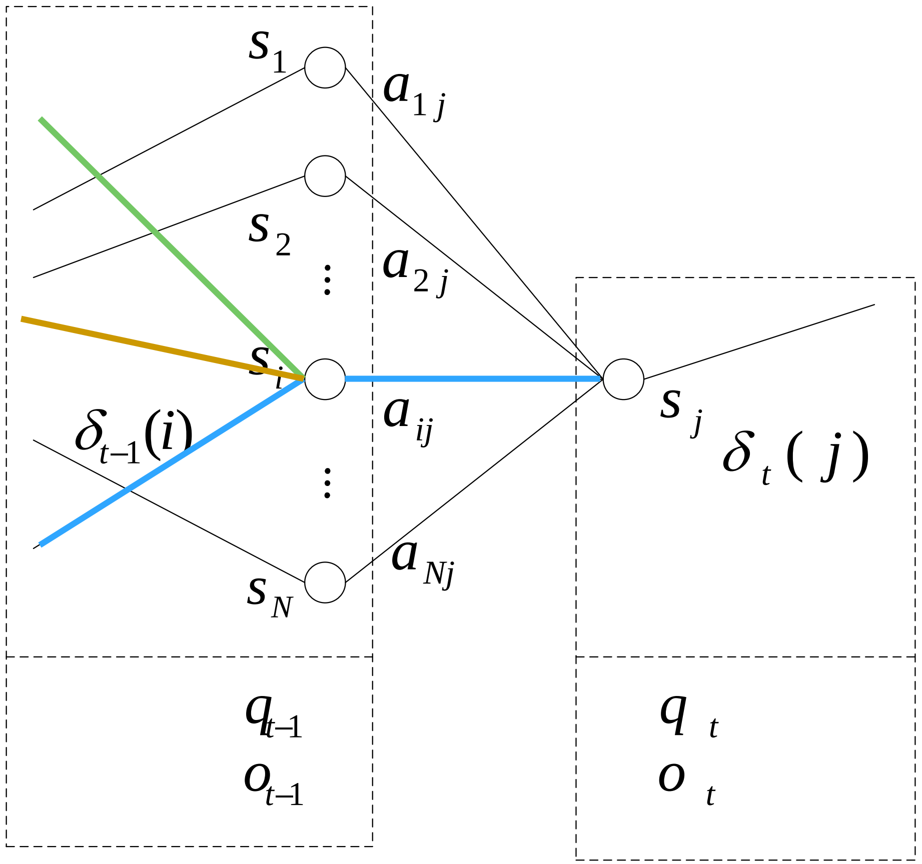
  - The best state sequence is give by the most probable path ending at *the final time step* t=T.
- **Partial Probability**
  - The probability of reaching an intermediate state j at time step t.
- **Maximum Partial Probability**
  - Representing the probability of the most probable path to a state j at time t, and not a total.
  - $\delta_t(j)$ is the maximum probability of all sequences ending at state j at time step t:

$$\delta_t(j) = P(\text{observation} \mid \text{hidden state } s_j) \times P(\text{most probable path to state } s_j \text{ at time } t)$$

- **Partial Best Path**
  - The partial best path is the state sequence which achieves the maximum partial probability.

Q2: A Smart Way

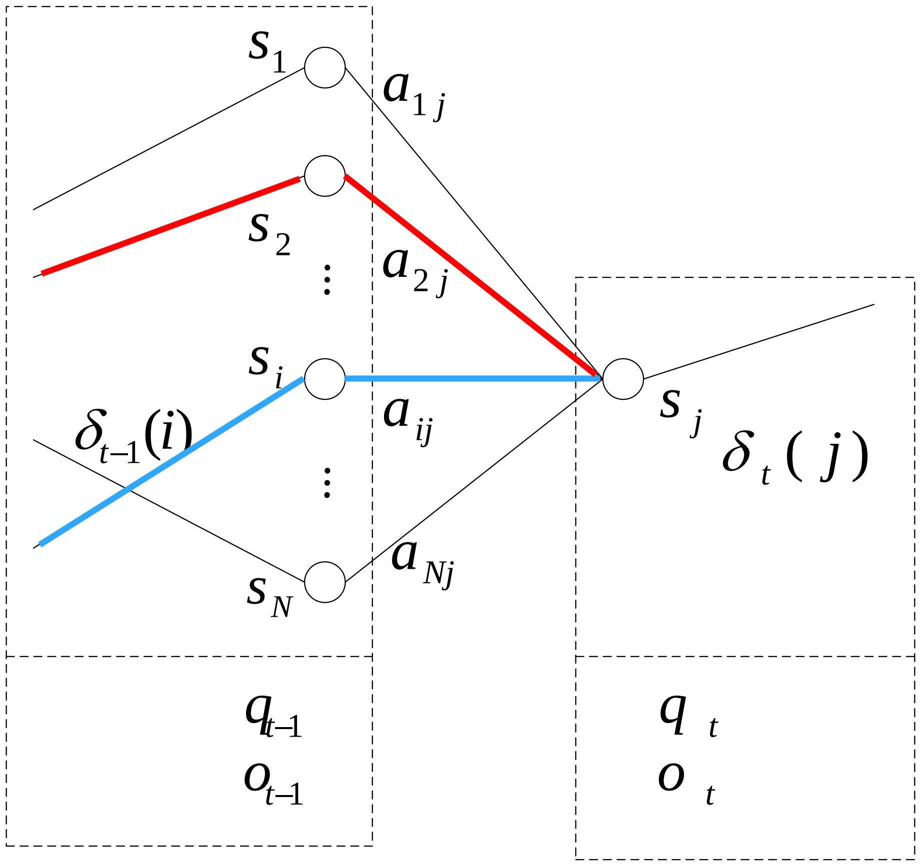
- For each intermediate or terminating state in the trellis, there is *a most probable path* to that state.



如果已知某个时刻t-1的最优状态序列，且t-1时刻的状态为s_i，那么在所有t时刻以s_j为终点且t-1时刻经过s_i的路径中，只有t-1时刻s_i的该最优状态序列是有效的。

Q2: A Smart Way

- For each intermediate or terminating state in the trellis, there is *a most probable path* to that state.



Q2: Viterbi Algorithm

- **Viterbi Probability**
  - The partial probability which gets the best score (highest probability) along a single path of reaching an intermediate state s_j at time step t:

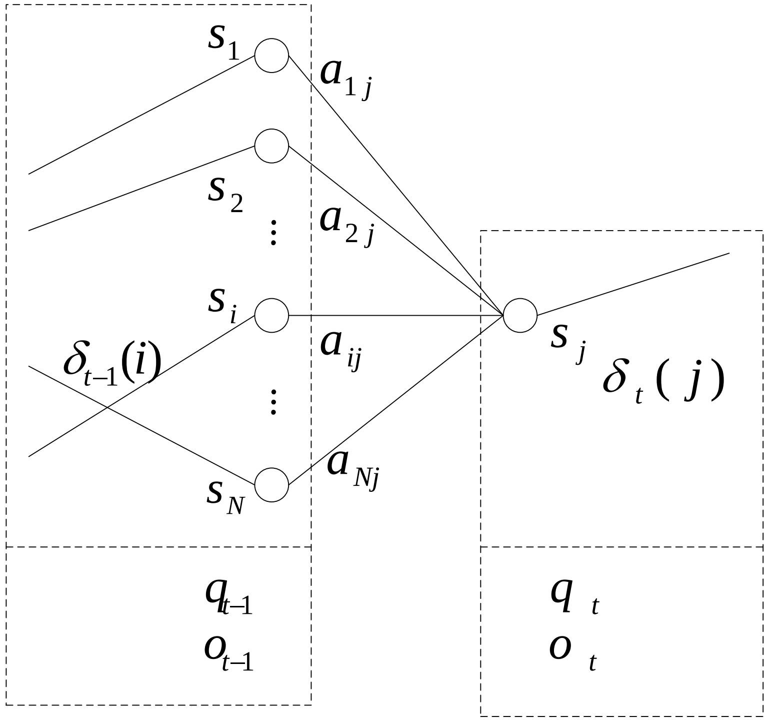
$$\delta_t(j) = \max_{q_1 q_2 \ldots q_{t-1}} P(q_1 q_2 \ldots q_t = s_j, o_1 o_2 \cdots o_t \mid \lambda)$$

  - At time t=1:

$$\delta_1(j) = \pi_j b_j(o_1),\ 1 \le j \le N$$

  - At time step t, $\delta_t(j)$ can be solved inductively from time t-1:

$$\delta_t(j) = \max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij} \right] b_j(o_t),$$
$$1 \le j \le N,\ 2 \le t \le T$$



Q2: Viterbi Algorithm

- **The solution to Q2:**
  - Initialization: $\delta_1(j) = \pi_j b_j(o_1),\ 1 \le j \le N$
  - Induction:
$$\delta_t(j) = \max_i \left[\delta_{t-1}(i) a_{ij} \right] b_j(o_t),\ 1 \le i, j \le N,\ 2 \le t \le T$$
$$\varphi_t(j) = \arg\max_i \left[\delta_{t-1}(i) a_{ij} \right],\ 1 \le i, j \le N,\ 2 \le t \le T$$
  - The last state of the best score path: $l'_T = \arg\max_j \delta_T(j)$
  - Backtracking the best score path:
$$l'_t = \varphi_{t+1}(l'_{t+1}),\ t = T-1, T-2, \ldots, 1$$
- **Computational Complexity**

$$O(T \cdot N^2)$$

Q2: Viterbi Algorithm

```
function viterbi
begin
    // initialization
    for j = 1 to N do begin
        δ_1(j) = π_j b_j(o_t);    φ_1(j) = 0;
    end
    // induction
    for t = 2 to T do begin
        for j = 1 to N do begin
            δ_t(j) = 0;
            for i =1 to N do begin
                δ' = δ_{t-1}(i) a_ij b_j(o_t);
                if δ' > δ_t(j) then begin
                    δ_t(j) = δ';    φ_t(j) = i;
                end
            end
        end
    end
    // get the probability of the best path
    δ' = δ_T(1)
    for j = 2 to N do begin
        if δ' < δ_T(j) then δ' = δ_T(j);
    end
    return δ';
end
```

Q2: Viterbi Algorithm

```
function backtrack
begin
    // get the last state of the best path
    l'_T = 1;
    for j = 2 to N do begin
        if δ_T(l'_T) < δ_T(j) then l'_T = j;
    end
    // backtracking
    for t = T - 1 to 1 do begin
        l'_t = φ_{t+1}(l'_{t+1});
    end
    return l'_t (t = 1,…,T);
end
```

Q2: Viterbi Algorithm

- The algorithm keeps a backward pointer (φ) for each state ($t > 1$), and stores a probability (δ) with each state.

$$\varphi_t(j) = \arg\max_i \left[\delta_{t-1}(i) a_{ij} \right], \; 1 \le i, j \le N, \; 2 \le t \le T$$

- The probability δ is the probability of having reached the state following the path indicated by the back pointers.

- When the algorithm reaches the states at time, $t = T$, the δ 's for the final states are the probabilities of following the optimal (most probable) route to that state.
- Thus selecting the largest, and using the implied route, provides the best answer to the problem.

Q2: Viterbi Algorithm

- **Insensitive to the noise garbles**
  - An important property about the Viterbi algorithm is that it takes a decision based on the whole sequence
    - If there is a particular 'unlikely' event midway through the sequence, the Viterbi algorithm will be insensitive to it.
  - This is particularly valuable in applications such as speech processing
    - Where an intermediate phoneme may be garbled or lost, but the overall sense of the spoken word may be detectable.

Q2: The Weather Example

- Hidden states (weather) = {Sunny, Cloudy, Rainy}

- Observed states (seaweed) = {Dry, Dryish, Damp, Soggy}

- Start probability =

Sunny	Cloudy	Rainy
0.63	0.17	0.20

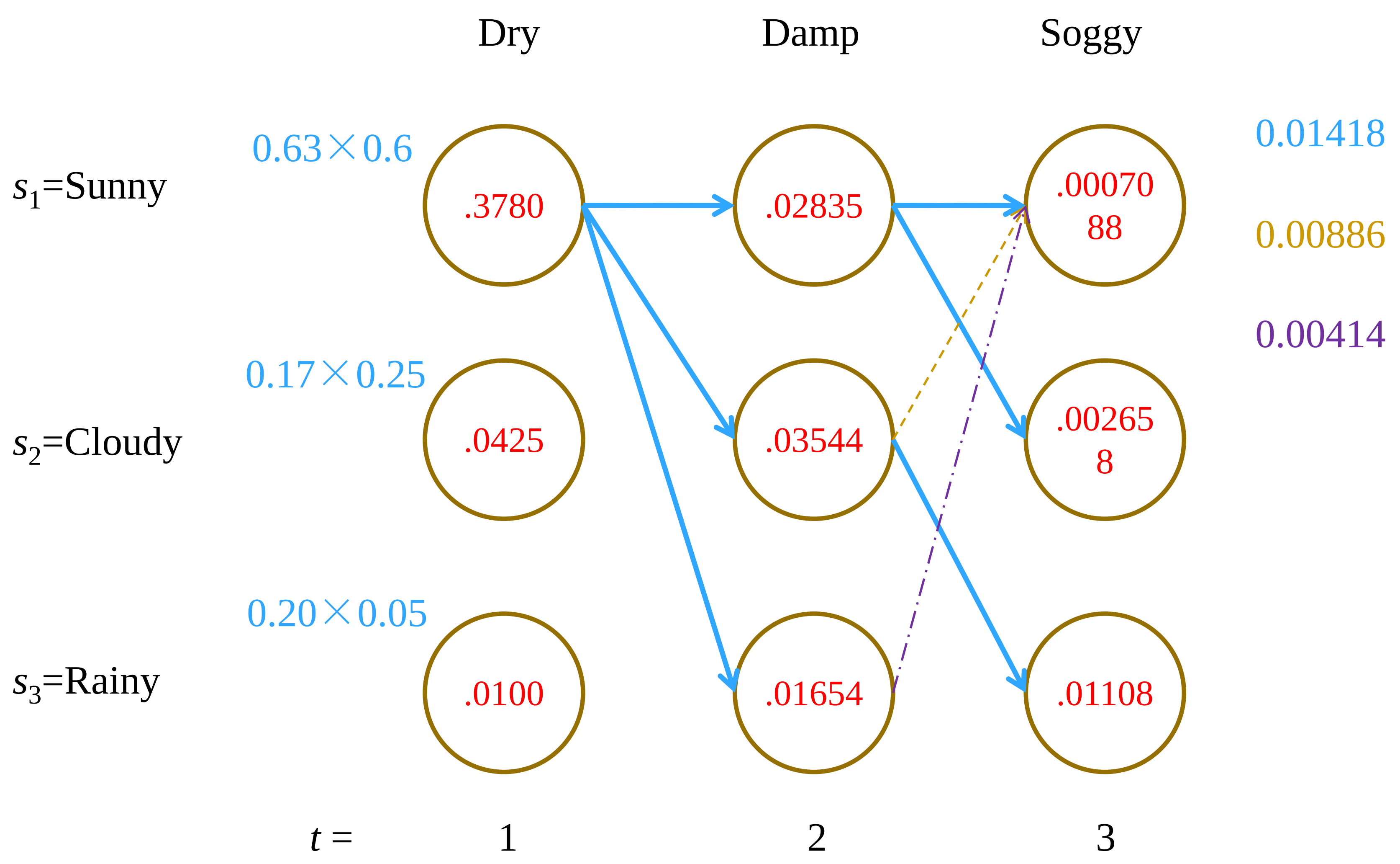
- State transition probability =

	Sunny	Cloudy	Rainy
Sunny	0.500	0.375	0.125
Cloudy	0.250	0.125	0.625
Rainy	0.250	0.375	0.375

- Emission probability =

	Dry	Dryish	Damp	Soggy
Sunny	0.60	0.20	0.15	0.05
Cloudy	0.25	0.25	0.25	0.25
Rainy	0.05	0.10	0.35	0.50

Q2: The Weather Example



Q3: Learning

- The 'useful' problems associated with HMMs are those of evaluation and decoding
  - they permit a measurement of a model's relative applicability, or
  - an estimate of what the underlying model is doing (what 'really happened').

- It can be seen that they both depend upon foreknowledge of the HMM parameters
  - the state transition matrix,
  - the observation matrix, and
  - the start probability

There are, however, many circumstances in practical problems where these are not directly measurable, and have to be estimated - this is the learning problem.

Q3: Baum-Welch Algorithm

$$O = o_1 o_2 \dots o_T \quad \Rightarrow \quad \hat{\lambda} = \arg\max_{\lambda} P(O \mid \lambda)\ ?$$

- No analytical solution
- $P(O|\lambda)$ can be locally maximized through the Baum-Welch Algorithm.
- Given $$P(\theta) = \int p(\xi, \theta)\, d\xi$$
- Consider the following auxiliary function:
$$Z(\theta, \theta') = \frac{1}{P(\theta)} \int p(\xi, \theta) \log p(\xi, \theta')\, d\xi$$
- The Baum-Welch theorem:
$$\text{if } Z(\theta, \theta') > Z(\theta, \theta) \text{ then } P(\theta') > P(\theta)$$

Q3: Baum-Welch Algorithm

- Hence, the following transformation T:

$$T(\theta) = \arg\max_{\theta'} Z(\theta, \theta')$$

- is a growth transformation of the function $P(.)$ on the domain of θ, i.e., it satisfies the condition:

$$P\left(T(\theta)\right) \geq P(\theta)$$

- A sequence $\{\theta_{(n)}, n=1,2,\ldots\}$ that monotonically increases the objective function P can thus be generated as follows, starting with an arbitrary initial guess $\theta_{(0)}$:

$$\theta_{(n)} = T(\theta_{(n-1)}), \qquad n = 1, 2, \ldots$$

- Remarks:
  - P maximization becomes a sequence of optimization problems for Z
  - The method is effectively only if these problems can be easily solved
  - Convergence to a maximum of P is not guaranteed by the existence of the growth function, but the local maximum

Q3: Baum-Welch Algorithm

- For HMM training, given a training sequence O, our goal is to adjust the model λ to maximize $P(O|\lambda)$:

$$O = o_1 o_2 \ldots o_T \quad \Rightarrow \quad \hat{\lambda} = \arg\max_{\lambda} P(O \mid \lambda) \ ?$$

- Then

$$P(\theta) \stackrel{\Delta}{=} P(O|\lambda) = \sum_{\text{all } Q} P(O, Q \mid \lambda) = \sum_{\text{all } Q} P(O \mid Q, \lambda) P(Q \mid \lambda)$$

$$= \sum_{q_1 q_2 \ldots q_T} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \cdots a_{q_{T-1} q_T} b_{q_T}(o_T)$$

- In the Baum-Welch formulation, the function to be maximized is

$$P(\theta) = \int p(\xi, \theta) d\xi$$

- In HMM training:

$$\int \rightarrow \sum$$

$$\theta \rightarrow \lambda$$

$$\xi \rightarrow Q$$

$$P(\xi, \theta) \rightarrow P(O, Q \mid \lambda)$$

Q3: Baum-Welch Algorithm

- The auxiliary function Z takes the form:

$$Z(\lambda,\lambda')=\frac{1}{P(O|\lambda)}\sum_{\text{all } Q}P(O,Q|\lambda)\log P(O,Q|\lambda')$$

$$=\frac{1}{P(O|\lambda)}\sum_{q_1q_2\ldots q_T}P(O,Q|\lambda)\left(\log\pi'_{q_1}+\sum_{t=1}^{T-1}\log a'_{q_tq_{t+1}}+\sum_{t=1}^{T}\log b'_{q_t}(o_t)\right)$$

- and can be decomposed into the sum of three terms which depend only on some components of the parameter set:

$$Z(\lambda,\lambda')=Z_\pi(\lambda,\pi')+Z_a(\lambda,A')+Z_b(\lambda,B')$$

$$Z_\pi(\lambda,\pi')=\frac{1}{P(O|\lambda)}\sum_{q_1q_2\ldots q_T}P(O,Q|\lambda)\log\pi'_{q_1}$$

$$Z_a(\lambda,A')=\frac{1}{P(O|\lambda)}\sum_{q_1q_2\ldots q_T}P(O,Q|\lambda)\sum_{t=1}^{T-1}\log a'_{q_tq_{t+1}}$$

$$Z_b(\lambda,B')=\frac{1}{P(O|\lambda)}\sum_{q_1q_2\ldots q_T}P(O,Q|\lambda)\sum_{t=1}^{T}\log b'_{q_t}(o_t)$$

Q3: Baum-Welch Algorithm

- The maximization of the auxiliary function $Z(\lambda, \lambda')$ over λ' can be solved by separately maximizing Z_π, Z_a, and Z_b.

- These are problems subject to the constraints:

$$\sum_{i=1}^{N} \pi'_i = 1$$

$$a'_{ij} \geq 0, \qquad 1 \leq i, j \leq N$$

$$\sum_{j=1}^{N} a'_{ij} = 1, \qquad 1 \leq i \leq N$$

$$\sum_{k=1}^{M} b'_j(k) = 1 \ \text{ or } \ \int_{-\infty}^{+\infty} b'_j(o)do = 1, \qquad 1 \leq j \leq N$$

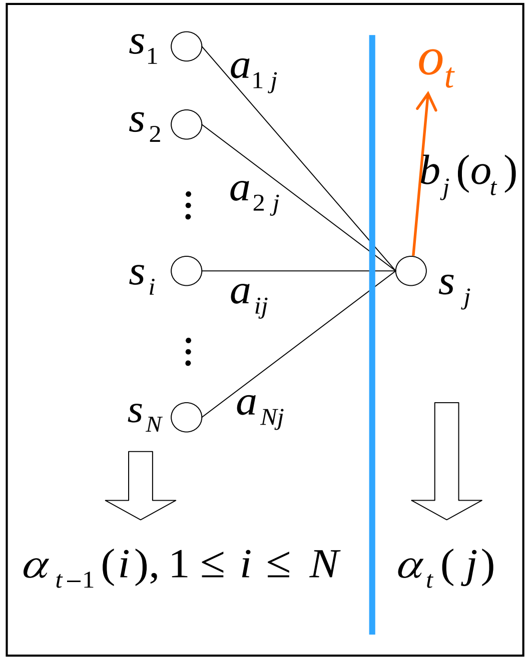
Q3: Forward-Backward Algorithm

- In maximizing the auxiliary function $Z(\lambda, \lambda')$ over λ' in HMM, the λ' value which maximizes $Z(\lambda, \lambda')$ and hence the likelihood P can be expressed in terms of three probabilities:
  - The forward probabilities: α's
  - The backward probabilities: β's, and
  - The forward-backward probabilities: γ's
- In such a way, it is so called the Forward-Backward Algorithm.

Q3: Forward Probability

- The probability of the partial observation sequence $o_1 o_2 \ldots o_{t-1} o_t$ (o_t at time t included)and state s_j at time t, given the model λ:

$$\alpha_t(j) = P(o_1 o_2 \cdots o_{t-1} o_t, q_t = s_j \mid \lambda)$$



- At time t=1:

$$\alpha_1(j) = \pi_j b_j(o_1), \ 1 \le j \le N$$

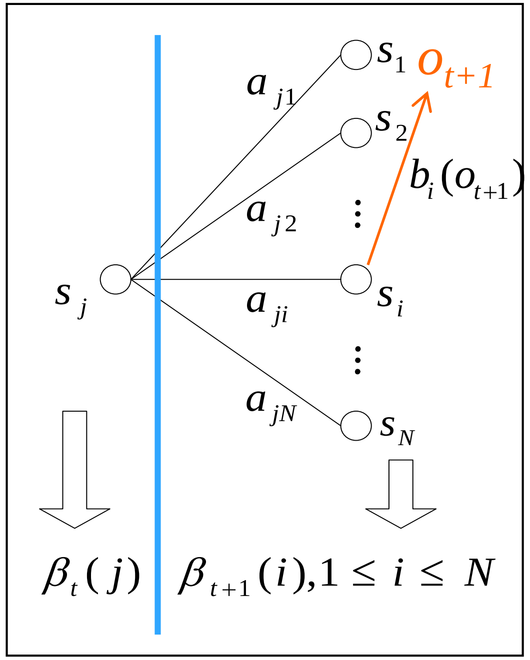
- At time step t, $\alpha_t(j)$ can be solved inductively from time t-1:

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t), \ 1 \le j \le N, \ t = 2,3,\ldots,T$$

Q3: Backward Probability

- The probability of the partial observation sequence from t+1 to the end $o_{t+1}o_{t+2}\ldots o_T$ (time t excluded), given state s_j at time t and the model λ:

$$\beta_t(j) = P(o_{t+1}o_{t+2}\cdots o_T \mid q_t = s_j, \lambda)$$



- At time t=T:

$$\beta_T(j) = 1,\ 1 \le j \le N$$

- At time step t, $\beta_t(j)$ can be solved inductively from time t+1:

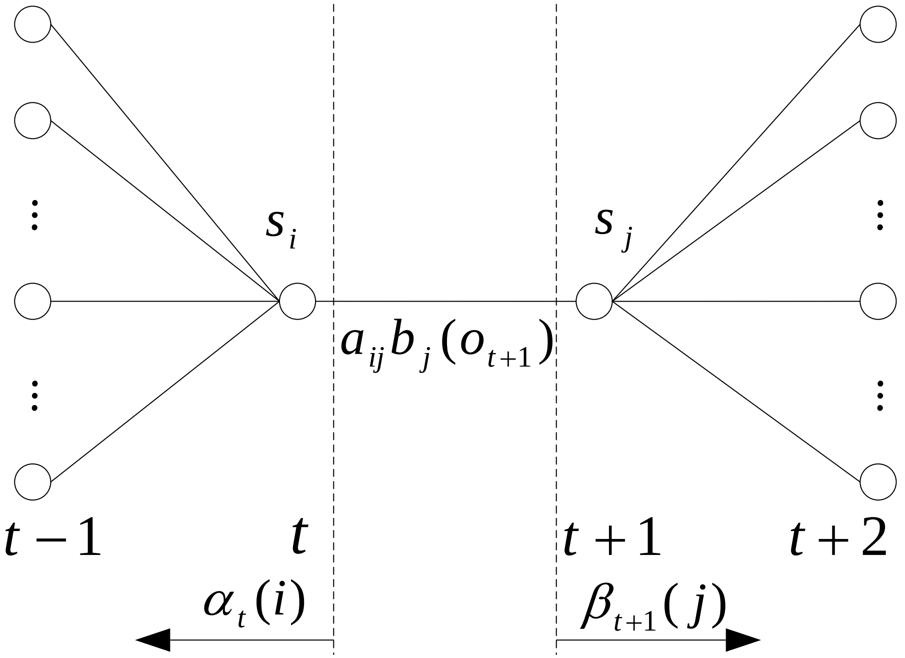
$$\beta_t(j) = \sum_{i=1}^{N} a_{ji} b_i(o_{t+1}) \beta_{t+1}(i),\ 1 \le j \le N,\ t = T-1, T-2, \ldots, 1$$

Q3: Forward-Backward Probability

- Let

$$\gamma_t(i,j) = P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda)$$

- be the probability of being in state s_i at time t, and state s_j at time t+1, given the model and the observation sequence.
- The event sequence leading to the conditions required by the probability $\gamma_t(i, j)$ is illustrated in the figure, which also shows the operations required for the computation of $\gamma_t(i, j)$.



Q3: Forward-Backward Probability

- In fact, from the definition of the forward and backward probabilities:

$$\alpha_t(i) = P(o_1 o_2 \cdots o_{t-1} o_t, q_t = s_i \mid \lambda)$$

$$\beta_t(i) = P(o_{t+1} o_{t+2} \dots o_T \mid q_t = s_i, \lambda)$$

- We have that (by exploiting the independence of observations):

$$\alpha_t(i)\beta_t(i) = P(O, q_t = s_i \mid \lambda)$$

- And then

$$P(O, q_t = s_i, q_{t+1} = s_j \mid \lambda) = \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

- Hence, $\gamma_t(i, j)$ can be written as:

$$\gamma_t(i, j) = P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda)$$

$$= \frac{P(q_t = s_i, q_{t+1} = s_j, O \mid \lambda)}{P(O \mid \lambda)}$$

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

Q3: Forward-Backward Probability

- Let $$\gamma_t(i) = P(q_t = s_i \mid O, \lambda)$$
- be the probability of being in state s_i at time t given the model and the observation sequence.
- From the definition of the forward and backward probabilities, we have that

$$\begin{aligned}
\gamma_t(i) &= P(q_t = s_i \mid O, \lambda) \\
&= \frac{P(q_t = s_i, O \mid \lambda)}{P(O \mid \lambda)} \\
&= \frac{P(q_t = s_i, O \mid \lambda)}{\sum_{i=1}^{N} P(q_t = s_i, O \mid \lambda)} \\
&= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}
\end{aligned}$$

Q3: Forward-Backward Probability

- It is worth noticing that:

$$\gamma_t(i) = \sum_{j=1}^{N} \gamma_t(i, j), \text{ only for } t = 1, 2, \ldots, T-1$$

- is the probability of being in state s_i at time t, given the observation sequence and the model, while
  - the expected number of transitions from s_i:

$$\sum_{t=1}^{T-1} \gamma_t(i) = P(s_i \rightarrow \text{all } s_j, \text{ all } t \mid O, \lambda)$$

  - the expected number of transitions from s_i to s_j.

$$\sum_{t=1}^{T-1} \gamma_t(i, j) = P(s_i \rightarrow s_j, \text{ all } t \mid O, \lambda)$$

Q3: Re-estimation Formulae

- Now we can go back to the maximization of Z_π, Z_a, and Z_b.

- Re-estimation of a'_{ij}
  - a'_{ij} represents the re-estimation of state transition probability from s_i to s_j

$$a'_{ij} = \frac{\text{the exprected number of transitions from } s_i \text{ to } s_j}{\text{the expected number of all transitions from } s_i}$$

$$= \frac{P(s_i \to s_j, \text{ all } t \mid O, \lambda)}{P(s_i \to \text{all } s_j \mid O, \lambda)}$$

$$= \frac{\sum_{t=1}^{T-1} \gamma_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

Q3: Re-estimation Formulae

- Now we can go back to the maximization of Z_π, Z_a, and Z_b.

- Re-estimation of π'_i
  - π'_i represents the re-estimation of start probability of s_i
  - It can be re-estimated from the probability of being in state s_i at time t=1, given the observation sequence

$$\begin{aligned}\pi'_i &= P(q_1 = s_i \mid O, \lambda) \\ &= \gamma_1(i) \\ &= \frac{\alpha_1(i)\beta_1(i)}{\sum_{i=1}^{N}\alpha_1(i)\beta_1(i)}\end{aligned}$$

Q3: Re-estimation Formulae

- Now we can go back to the maximization of Z_π, Z_a, and Z_b.

- Re-estimation of $b'_j(k)$
  - $b_j(k)$ represents the probability of being in state s_j and emitting the observation v_k ($k = 1, 2, \ldots, M$)
  - The re-estimation can be computed as:

$$b'_j(k) = \frac{\text{the probability of being in state } s_j \text{ and emitting the observation } v_k}{\text{the probability of being in state } s_j}$$

$$= \frac{\displaystyle\sum_{t=1;\text{s.t.} o_t = v_k}^{T-1} \gamma_t(j)}{\displaystyle\sum_{t=1}^{T-1} \gamma_t(j)}$$

Discrete emitting / output probability

Discrete Output Probability

- The output probability for the discrete observations
  - The discrete observations: $\mathbf{V}=\{v_1, v_2, \ldots, v_M\}$
  - The discrete output probability:

$$b_j(k) = P(o_t = v_k \mid q_t = s_j),$$
$$1 \le j \le N, \quad 1 \le k \le M, \quad 1 \le t \le T$$

Continuous Output Probability

- The output probability for the continuous observations
  - The continuous output densities:

$$b_j(o_t) = p(o_t \mid q_t = s_j), \quad 1 \le j \le N, \quad 1 \le t \le T$$

  - 1-dimension Gaussian output densities:

$$b_j(o_t) = N(\mu_j, \sigma_j; o_t) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left[\frac{-(o_t - \mu_j)^2}{2\sigma_j^2} \right]$$

  - d-dimension Gaussian densities:

$$b_j(\boldsymbol{o}_t) = N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j; \boldsymbol{o}_t) = \frac{1}{(2\pi)^{d/2} \left|\boldsymbol{\Sigma}_j\right|^{1/2}} \exp\left[-\frac{1}{2} \left(\boldsymbol{o}_t - \boldsymbol{\mu}_j\right)^T \boldsymbol{\Sigma}_j^{-1} \left(\boldsymbol{o}_t - \boldsymbol{\mu}_j\right) \right]$$

  - Mixture of Gaussian densities:

$$b_j(\boldsymbol{o}_t) = \sum_{k=1}^{M} w_{jk} N(\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}; \boldsymbol{o}_t)$$

Q3: Re-estimation Formulae

- For 1-dimension Gaussian output densities:

$$b_j(o_t) = N(\mu_j, \sigma_j; o_t) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left[\frac{-(o_t - \mu_j)^2}{2\sigma_j^2}\right]$$

- The auxiliary function:

$$Z_b(\lambda, B') = \frac{1}{P(O|\lambda)} \sum_{q_1 q_2 \ldots q_T} P(O,Q|\lambda) \sum_{t=1}^{T} \log b'_{q_t}(o_t)$$

$$= \sum_{q_1 q_2 \ldots q_T} \frac{P(O,Q|\lambda)}{P(O|\lambda)} \sum_{t=1}^{T} \log b'_{q_t}(o_t)$$

$$= \sum_{t=1}^{T} \left\{ \sum_{Q: q_t = s_j} \frac{P(q_t = s_j, O|\lambda)}{P(O|\lambda)} \log b'_{q_t = s_j}(o_t) \right\}$$

$$= \sum_{t=1}^{T} \left\{ \sum_{j=1}^{N} \gamma_t(j) \log b'_j(o_t) \right\}$$

$$= \sum_{t=1}^{T} \left\{ \sum_{j=1}^{N} \gamma_t(j) \left[-\frac{1}{2} \log 2\pi - \log \sigma'_j - \frac{(o_t - \mu'_j)^2}{2\sigma'^2_j} \right] \right\}$$

Q3: Re-estimation Formulae

- For 1-dimension Gaussian output densities:

$$Z_b(\lambda, B') = \sum_{t=1}^{T} \left\{ \sum_{j=1}^{N} \gamma_t(j) \left[-\frac{1}{2}\log 2\pi - \log \sigma'_j - \frac{(o_t - \mu'_j)^2}{2\sigma'^2_j} \right] \right\}$$

$$\frac{\partial Z_b(\lambda, B')}{\partial \mu'_j} = \sum_{j=1}^{N} \gamma_t(j) \left[\frac{o_t - \mu'_j}{\sigma'^2_j} \right] = 0$$

$$\frac{\partial Z_b(\lambda, B')}{\partial \sigma'_j} = \sum_{j=1}^{N} \gamma_t(j) \left[-\frac{1}{\sigma'_j} + \frac{(o_t - \mu'_j)^2}{\sigma'^3_j} \right] = 0$$

- Hence:

$$\mu'_j = \frac{\sum_{t=1}^{T} \gamma_t(j) o_t}{\sum_{t=1}^{T} \gamma_t(j)} \qquad \sigma'^2_j = \frac{\sum_{t=1}^{T} \gamma_t(j) o_t^2}{\sum_{t=1}^{T} \gamma_t(j)} - \mu'^2_j$$

Q3: Re-estimation Formulae

- For multi-dimensional Gaussian output densities:

$$b_j(\boldsymbol{o}_t) = N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j; \boldsymbol{o}_t)$$

$$\boldsymbol{\mu}'_j = \frac{\sum_{t=1}^{T} \gamma_t(j)\boldsymbol{o}_t}{\sum_{t=1}^{T} \gamma_t(j)} \qquad \boldsymbol{\Sigma}'_j = \frac{\sum_{t=1}^{T} \gamma_t(j)\boldsymbol{o}_t\boldsymbol{o}_t^{\mathrm{tr}}}{\sum_{t=1}^{T} \gamma_t(j)} - \boldsymbol{\mu}'_j \boldsymbol{\mu}'^{\mathrm{tr}}_j$$

Q3: Re-estimation Formulae

- For mixture of Gaussian densities:

$$b_j(\boldsymbol{o}_t) = \sum_{k=1}^{M} w_{jk} N(\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}; \boldsymbol{o}_t)$$

$$w'_{jk} = \frac{\sum_{t=1}^{T} \gamma_t^k(j)}{\sum_{t=1}^{T}\sum_{k=1}^{M} \gamma_t^k(j)} \qquad \boldsymbol{\mu}'_{jk} = \frac{\sum_{t=1}^{T} \gamma_t^k(j)\boldsymbol{o}_t}{\sum_{t=1}^{T} \gamma_t^k(j)} \qquad \boldsymbol{\Sigma}'_{jk} = \frac{\sum_{t=1}^{T} \gamma_t^k(j)\boldsymbol{o}_t\boldsymbol{o}_t^{\mathrm{tr}}}{\sum_{t=1}^{T} \gamma_t^k(j)} - \boldsymbol{\mu}'_{jk}\,\boldsymbol{\mu}'^{\mathrm{tr}}_{jk}$$

- where the term $\gamma_t^k(j)$ generalizes to $\gamma_t(j)$:

$$\gamma_t^k(j) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)}\right]\left[\frac{w_{jk} N(\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}; \boldsymbol{o}_t)}{\sum_{m=1}^{M} w_{jm} N(\boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}; \boldsymbol{o}_t)}\right]$$

Q3: EM Algorithm

- The EM algorithm can be treated as a particular instance of the Baum-Welch Algorithm
  - Initialization step: choose initial model parameter λ in some way
  - Expectation step: with the current model λ compute:
    - 1) left-to-right stage on trellis: α's
    - 2) right-to-left stage on trellis: β's and γ's
  - Maximization step: compute λ' through the re-estimation formulae
  - Update step: $\lambda \leftarrow \lambda'$
  - Go back to the E step
- The procedure is iterated until the relative likelihood improvement is insignificant.

Revisit: Continuous Output Probability

- The output probability for the continuous observations
  - The continuous output densities:

$$b_j(o_t) = p(o_t \mid q_t = s_j), \quad 1 \le j \le N, \quad 1 \le t \le T$$

  - 1-dimension Gaussian output densities:

$$b_j(o_t) = N(\mu_j, \sigma_j; o_t) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left[\frac{-(o_t - \mu_j)^2}{2\sigma_j^2}\right]$$

  - d-dimension Gaussian densities:

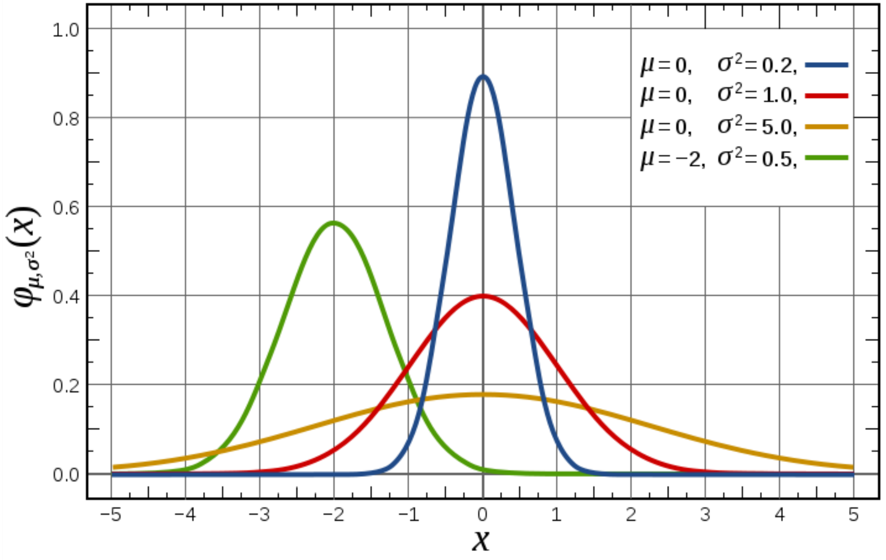
$$b_j(\boldsymbol{o}_t) = N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j; \boldsymbol{o}_t) = \frac{1}{(2\pi)^{d/2} \left|\boldsymbol{\Sigma}_j\right|^{1/2}} \exp\left[-\frac{1}{2}\left(\boldsymbol{o}_t - \boldsymbol{\mu}_j\right)^T \boldsymbol{\Sigma}_j^{-1} \left(\boldsymbol{o}_t - \boldsymbol{\mu}_j\right)\right]$$

  - Mixture of Gaussian densities:

$$b_j(\boldsymbol{o}_t) = \sum_{k=1}^{M} w_{jk} N(\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}; \boldsymbol{o}_t)$$

GMM: Gaussian Mixture Model

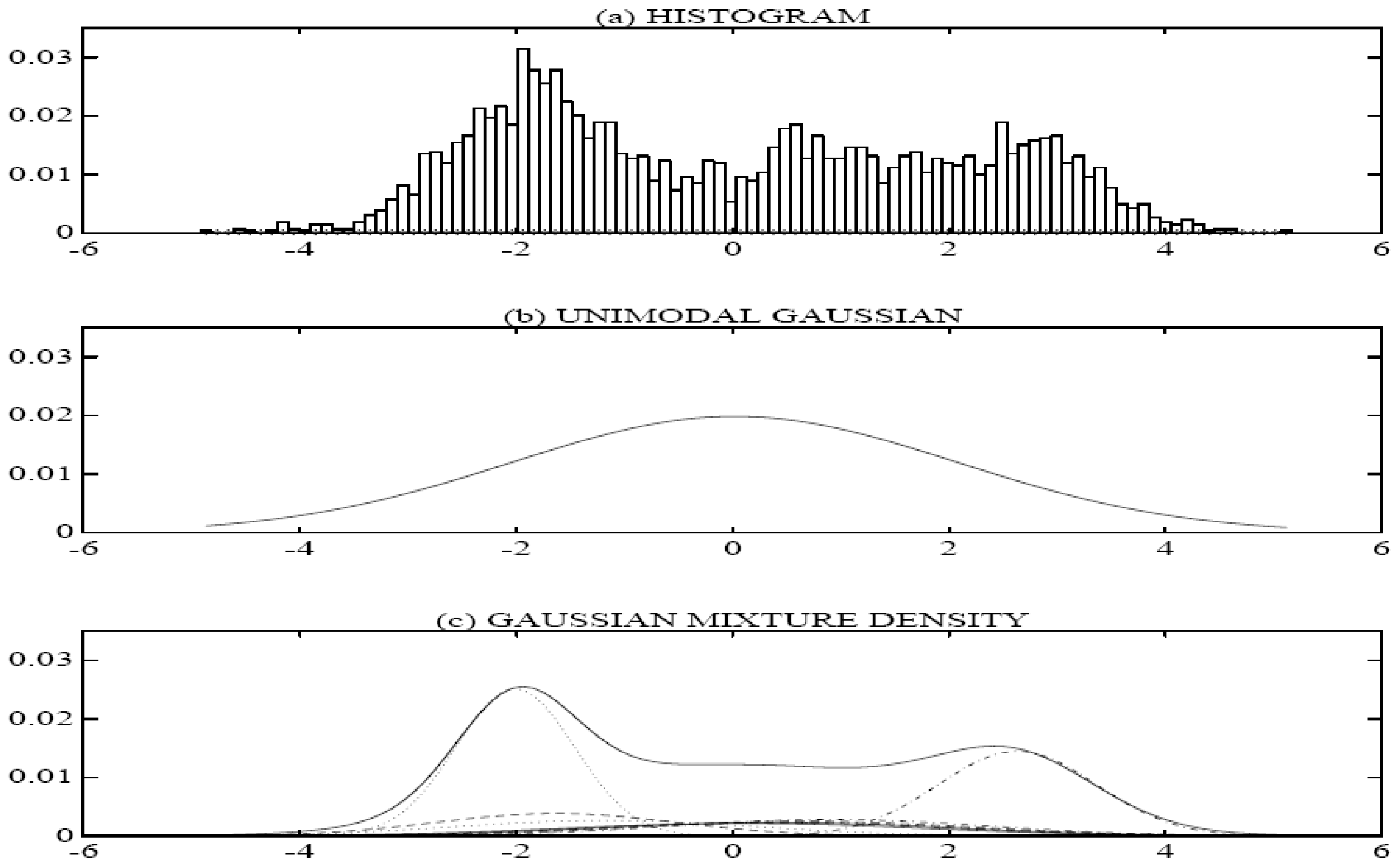
- **GMM**
  - A Gaussian mixture model is a weighted sum of *M* component Gaussian densities
  - One of the powerful attributes of the GMM is its ability to form smooth approximations to arbitrarily shaped densities.



$$b_j(\boldsymbol{o}_t) = \sum_{k=1}^{M} w_{jk} N(\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}; \boldsymbol{o}_t)$$

$$N(\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}; \boldsymbol{o}_t) = \frac{1}{(2\pi)^{d/2} \left|\boldsymbol{\Sigma}_{jk}\right|^{1/2}} \exp\left[-\frac{1}{2}\left(\boldsymbol{o}_t - \boldsymbol{\mu}_{jk}\right)^T \boldsymbol{\Sigma}_{jk}^{-1} \left(\boldsymbol{o}_t - \boldsymbol{\mu}_{jk}\right)\right]$$

GMM: Gaussian Mixture Model



(from: Reynolds, D. Gaussian Mixture Models. MIT Lincoln Laborator)

Review

- What is Deterministic State Machine (DSM)?
- What is Markov process? What are the parameters related to a Markov process?
- What is Hidden Markov Model (HMM)? Why HMM is important? What are the parameters of a HMM?
- What are the three canonical questions of HMM?
- How to compute the probability of a observation sequence given the model? The forward algorithm and the backward algorithm.
- How to compute the optimal hidden state sequence which best explains the observation sequence given the model? The Viterbi algorithm.
- How to estimate the model parameters of a HMM? The Baum-Welch Algorithm or the forward-backward algorithm.
- What are the possible structures of a HMM?
- What are the possible types of a HMM?