PROBABILISTIC MODELS – PART 6a: HIDDEN MARKOV MODELS (HMMs)

Gerhard Widmer

Institute of Computational Perception
Johannes Kepler University
Linz, Austria

gerhard.widmer@jku.at
www.cp.jku.at/people/widmer

November 17, 2025

Presentation partly based on and inspired by [Koller & Friedman, 2009] and [Russell & Norvig, 2021], including the use of some figures from their books and/or lecture slides.

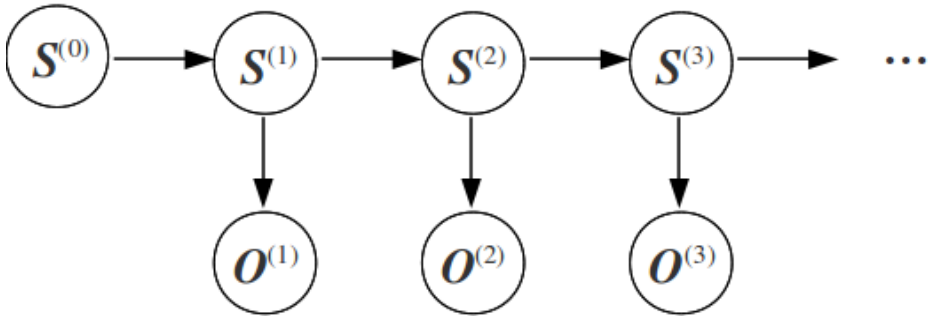
Many thanks to Daphne Koller, Nir Friedman, Stuart Russell, and Peter Norvig for making these available (`pgm.stanford.edu`; `aima.cs.berkeley.edu`).

Goals of this Lecture

- Introduce a very popular class of discrete state-observation models
- Discuss different inference tasks in HMMs
- Show how to do efficient (even real-time!) inference in HMMs
- Show how HMMs can be learned from data
- Discuss some of the advanced issues in HMM modelling (briefly)
- Show how to model continuous observations via Gaussian Mixture Models
- (and how to learn these from data)
- Introduce the general E-M Algorithm
- Present some real-world applications

Outline

Remember: State-Observation Models



1. **Initial Distribution Model** $\mathcal{B}_0 = P(\boldsymbol{S}^{(0)})$
2. **State Transition Model:** $P(\boldsymbol{S}'|\boldsymbol{S}) = P(\boldsymbol{S}^{(t+1)}|\boldsymbol{S}^{(t)})$ (same for all $t > 0$)
3. **Observation Model:** $P(\boldsymbol{O}|\boldsymbol{S}) = P(\boldsymbol{O}^{(t)}|\boldsymbol{S}^{(t)})$ (the same for all $t > 0$)

Assumptions:

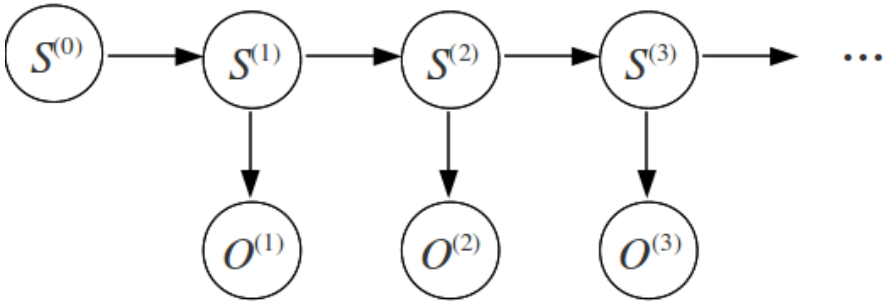
- State variables $\boldsymbol{S}$ are hidden, observation variables $\boldsymbol{O}$ are observable
- In the simplest case (discrete variables), the models are represented by sets of CPD tables.

Hidden Markov Models

Definition

A **HIDDEN MARKOV MODEL (HMM)** is a State-Observation Model with a **single discrete state variable** S and a **single observation variable** O.[a]

[a] Later on – in the speech and music application examples below –, we will have N *continuous* observation variables $\boldsymbol{O}$ and model their joint distribution $P(\boldsymbol{O}|S)$ via *Gaussian Mixture Models*.



Consequences:

- The transition model can be represented as a single CPD table $P(S'|S)$
- The observation model is a single CPD table $P(O|S)$

Graphical Representation of State Transition Model

In HMM applications, the transition model often encodes a lot of structural constraints:

- Many direct state transitions may be impossible
  $\Rightarrow$ corresponding transition probabilities are zero
- Resulting model is **"sparse"**.

☞ **Alternative representation of transition model structure:**

- Represent transition model itself as a directed graph
  (as in our *Grasshopper Markov Chain* – see Ch. 4b)
- One node per different *value* s_i of state variable S
- State transitions with non-zero probability represented as arrows, labelled with transition probability.

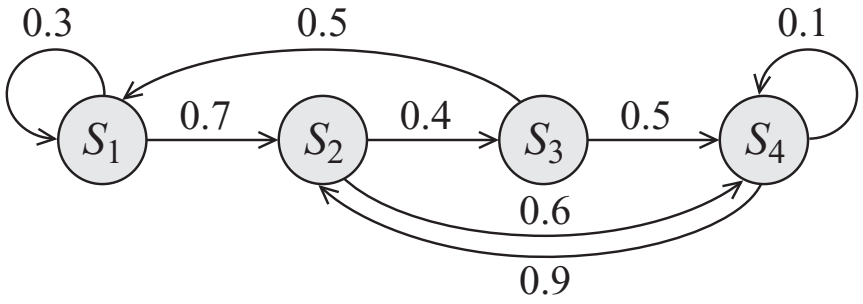
Graphical Representation of State Transition Model

Example:

- HMM with state variable S with $Val(S) = \{s_1, s_2, s_3, s_4\}$
- and transition model $P(S'|S)$:

	s_1	s_2	s_3	s_4
s_1	0.3	0.7	0	0
s_2	0	0	0.4	0.6
s_3	0.5	0	0	0.5
s_4	0	0.9	0	0.1

Graphical representation of transition model:



NOTE:

- The s_i in the HMM transition graph are the different ***values*** of the (single) state variable S !

HMMs: Formal Specification

Definition

A **HIDDEN MARKOV MODEL (HMM)** is fully specified by

1. the hidden **state variable** S with values (states) $\{s_1, s_2, \ldots, s_N\}$
2. the **observation variable** O with values (observations) $\{o_1, o_2, \ldots, o_M\}$
3. a list (vector) $\mathbf{\Pi}$ of **starting probabilities** $\pi_i = P(S^{(0)} = s_i)$
   (the probability of the system starting in any of the possible states
   = the *initial distribution model* $\mathcal{B}_0$)
4. a matrix $\boldsymbol{A}$ of **state transition probabilities** $P(S' = s_j | S = s_i)$
   (the CPD $P(S'|S)$ of the *transition model*)
5. a matrix $\boldsymbol{B}$ of **observation probabilities** $P(O = o_j | S = s_i)$
   (the CPD $P(O|S)$ of the *observation model*)

A (Trivially) Simple Example

Consider a simple model of the weather as a temporal process:

- Only 3 possible (discrete) weather states: sunny, rainy, cloudy
- The weather is observed once per day
- The weather forms a Markov process.

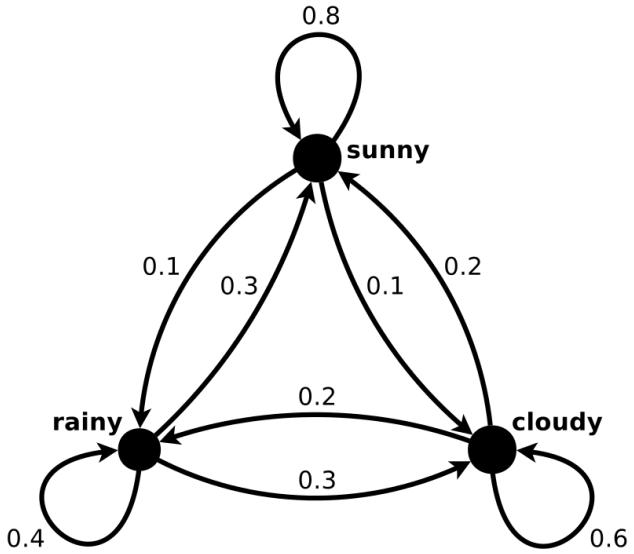
Initial Distribution & State Transition Model:

$$S = \{rainy, cloudy, sunny\}$$

$$\mathbf{\Pi} = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.5 \end{bmatrix}$$

$$\boldsymbol{A} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

where $\boldsymbol{A} = \{a_{ij}\} = \{P(s_j^{(t+1)} | s_i^{(t)})\}$



Example: $a_{13} = P(S' = s_3 | S = s_1) = P(sunny | rainy) = 0.3$

A (Trivially) Simple Example

Observation Model:

- Assume state sequence (weather) is not directly observable (because you are in prison ...)
- Instead, we observe a sequence $o^{(1)}, o^{(2)}, \ldots, o^{(T)}$ of air humidity measurements O:

$$O = \{dry, medium, humid\}$$

- Humidity depends probabilistically on the current weather state:

$$\boldsymbol{B} = \begin{array}{c} \\ \text{dry} \\ \text{medium} \\ \text{humid} \end{array} \begin{array}{c} \begin{array}{ccc} \text{rainy} & \text{cloudy} & \text{sunny} \end{array} \\ \begin{pmatrix} 0.1 & 0.3 & 0.6 \\ 0.5 & 0.4 & 0.2 \\ 0.4 & 0.3 & 0.2 \end{pmatrix} \end{array}$$

where

$$\boldsymbol{B} = \{b_{ij}\} = \{b_j(o_i)\} = \{P(o_i|s_j)\}$$

Example: $b_{13} = b_3(o_1) = P(o_1|s_3) = P(dry|sunny) = 0.6$

A (Trivially) Simple Example

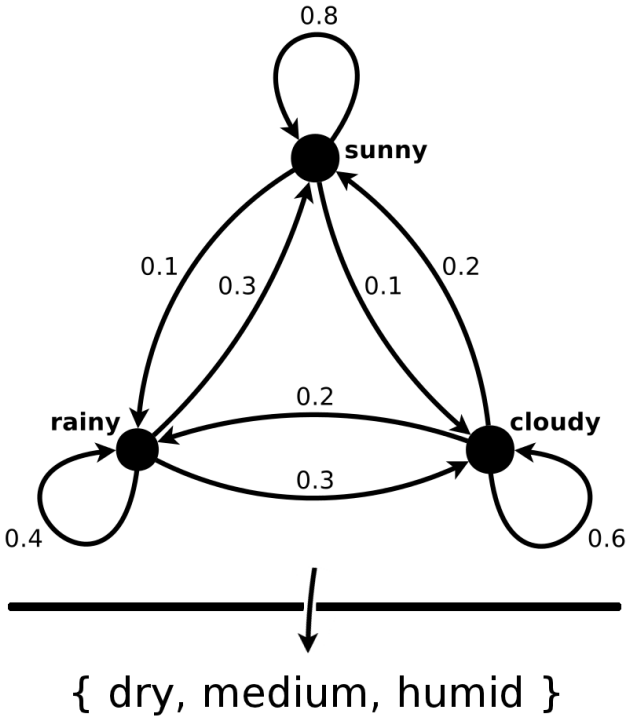
☞ **COMPLETE SPECIFICATION of our weather HMM:**

$$S = \{rainy, cloudy, sunny\}$$
$$O = \{dry, medium, humid\}$$

$$\mathbf{\Pi} = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.5 \end{bmatrix}$$

$$\boldsymbol{A} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

$$\boldsymbol{B} = \begin{array}{c} \\ \text{dry} \\ \text{medium} \\ \text{humid} \end{array} \begin{array}{c} \begin{array}{ccc} \text{rainy} & \text{cloudy} & \text{sunny} \end{array} \\ \begin{pmatrix} 0.1 & 0.3 & 0.6 \\ 0.5 & 0.4 & 0.2 \\ 0.4 & 0.3 & 0.2 \end{pmatrix} \end{array}$$



HMMs: Common Reasoning Tasks

FILTERING / TRACKING ("belief state maintenance")

- Compute

$$\boxed{P(S^{(t)} \mid \boldsymbol{o}^{(1:t)})}$$

- Needed for on-line tracking of the current hidden state

PREDICTION

- Compute

$$\boxed{P(S^{(t+k)} \mid \boldsymbol{o}^{(1:t)})}$$

- Needed to anticipate future situations

SMOOTHING

- Compute

$$\boxed{P(S^{(t)} \mid \boldsymbol{o}^{(1:T)}), t < T}$$

- Needed for better assessment of situation (and for learning)

HMMs: Common Reasoning Tasks

... and two more:

DECODING ("finding the most likely explanation")

- Find

$$\boxed{\arg\max_{s^{(1)},\ldots,s^{(T)}} P(s^{(1)},\ldots,s^{(T)} \mid \boldsymbol{o}^{(1:T)})}$$

  (most likely state sequence the system went through when producing $\boldsymbol{o}^{(1:T)}$)
- Needed for explaining a given observation sequence.

Computing the PROBABILITY OF AN OBSERVATION SEQUENCE (= likelihood of model $\mathcal{M}$)

- Compute

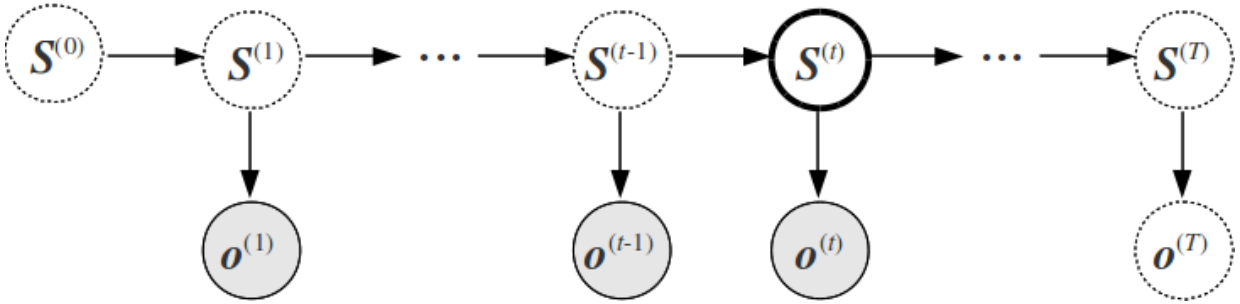
$$\boxed{P(\boldsymbol{o}^{(1:T)} \mid \mathcal{M}) = L(\mathcal{M} : \boldsymbol{o}^{(1:T)})}$$

- Will be used for **classification** (e.g., see speech understanding below)

Task 1: Filtering (Tracking)

Filtering

- Compute $P(S^{(t)} \mid \boldsymbol{o}^{(1:t)})$
- Needed for on-line tracking of the current hidden state



Remember General Filtering Algorithm for S-O Models

Recursive Forward Step

$$\begin{aligned} \mathsf{f}^{(1:t+1)} &= P(\boldsymbol{S}^{(t+1)} \mid \boldsymbol{o}^{(1:t+1)}) \\ &= \frac{1}{Z} \times \underbrace{\underbrace{P(\boldsymbol{o}^{(t+1)}|\boldsymbol{S}^{(t+1)})}_{\text{Conditioning}} \underbrace{\sum_{\boldsymbol{s}^{(t)}} P(\boldsymbol{S}^{(t+1)}|\boldsymbol{s}^{(t)})\mathsf{f}_{\boldsymbol{s}}^{(1:t)}}_{\text{State Forward Propagation}}}_{\text{Function FORWARD}} \end{aligned}$$

Specialised to HMMs:

$$\boxed{\mathsf{f}^{(1:t+1)} = \frac{1}{Z} \times \boldsymbol{B}[o^{(t+1)}]^T \circ \left(\boldsymbol{A}^T \mathsf{f}^{(1:t)}\right)}$$

- $\mathsf{f}^{(1:t)}$ is a column vector of dimension $N = |Val(S)|$
- $\boldsymbol{B}[o^{(t+1)}]$ is the row of $\boldsymbol{B}$ pertaining to observation $o^{(t+1)}$
- $\boldsymbol{A}^T$ is the transpose of $\boldsymbol{A}$; $\boldsymbol{A}^T \mathsf{f}^{(1:t)}$ denotes standard matrix multiplication
- $\circ$ is the *element-wise (Hadamard) product* of two matrices.

Filtering in HMMs: The Forward Algorithm

FORWARD ALGORITHM for FILTERING

Initialisation:

$$\mathsf{f}^{(1:0)} = P(S^{(0)}) = \boldsymbol{\Pi}$$

Recursive forward propagation: Propagate message and re-normalise:

$$\mathsf{f}^{(1:t+1)} = P(S^{(t+1)} \mid \boldsymbol{o}^{(1:t+1)}) = \frac{1}{Z}\textsc{Forward}(\mathsf{f}^{(1:t)}, o^{(t+1)})$$

Encapsulate the two recursive propagation steps (without re-normalisation) into a function FORWARD (will re-use this later, in other inference tasks):

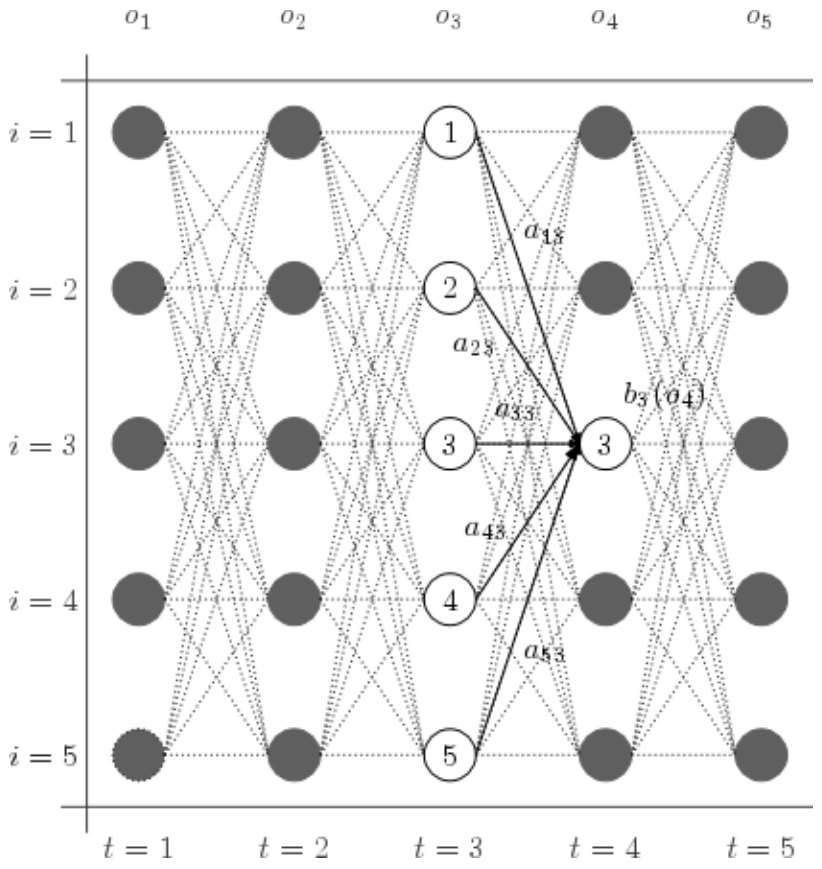
Function FORWARD($\mathsf{f}^{(1:t)}$, $o^{(t+1)}$)

Returns $\boldsymbol{V} = P(S^{(t+1)}, o^{(t+1)}|\boldsymbol{o}^{(1:t)}) = \boldsymbol{B}[o^{(t+1)}] \circ \left(\boldsymbol{A}^T \mathsf{f}^{(1:t)}\right)$:

for $i = 1$ to N : $\boxed{v_i = b_i(o^{(t+1)}) \sum_j a_{ji} \mathsf{f}_j^{(1:t)})}$

The Intuition Behind the Forward Step

$$
\begin{aligned}
&\mathsf{f}_i^{(1:t+1)} \\
&= P(S^{(t+1)} = s_i \mid \boldsymbol{o}^{(1:t+1)}) \\
&= \frac{1}{Z}\left[\sum_j \mathsf{f}_j^{(1:t)} a_{ji}\right] b_i(o^{(t+1)})
\end{aligned}
$$



In Words:

Probability of being in state i at time $t+1$, given observations $\boldsymbol{o}^{(1:t+1)}$, is proportional to the sum, over all predecessor states j, of the probability of having been in state j at t, given the observations up to t, then moving from j to i, and then observing $o^{(t+1)}$ in state i.

Filtering: A Simple Example

Consider our simple weather HMM:

$$S = \{rainy, cloudy, sunny\}$$
$$O = \{dry, medium, humid\}$$

$$\boldsymbol{A} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \qquad \boldsymbol{B} = \begin{array}{c} \\ \text{dry} \\ \text{medium} \\ \text{humid} \end{array} \begin{array}{c} \begin{array}{ccc} \text{rainy} & \text{cloudy} & \text{sunny} \end{array} \\ \begin{pmatrix} 0.1 & 0.3 & 0.6 \\ 0.5 & 0.4 & 0.2 \\ 0.4 & 0.3 & 0.2 \end{pmatrix} \end{array} \qquad \boldsymbol{\Pi} = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.5 \end{bmatrix}$$

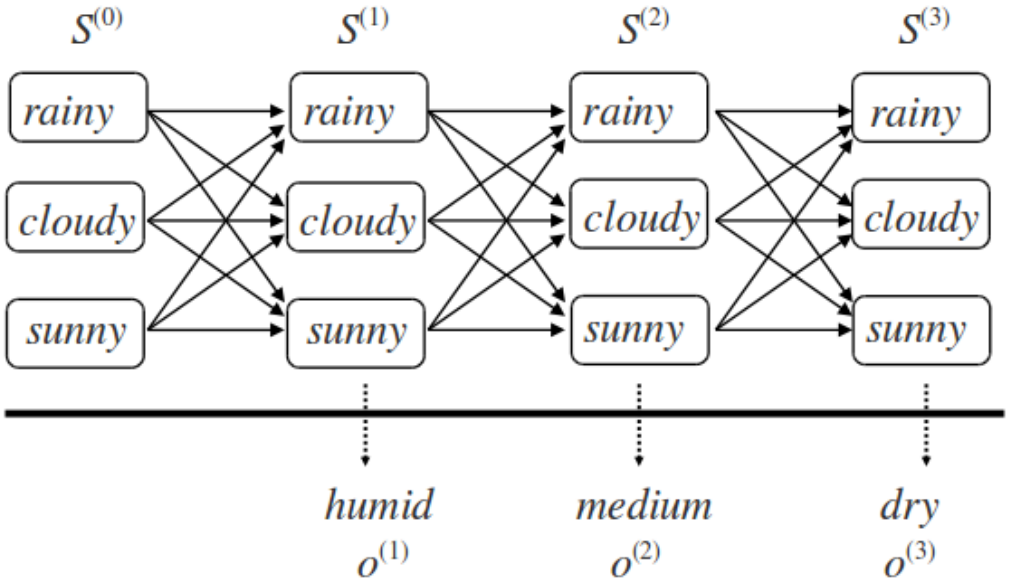
... and the following sequence of observations for three days $1, 2, 3$:

$$\langle humid, medium, dry \rangle$$

$\Rightarrow$ What is the most probable weather state for each of the three days?

Filtering: A Simple Example

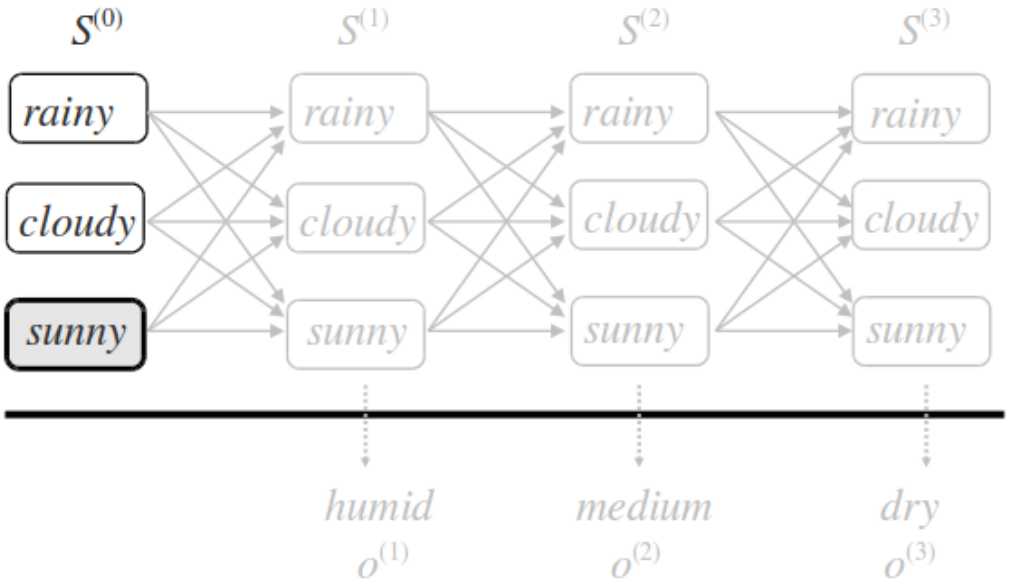
Lattice of all possible paths (state sequences) through the unrolled HMM:



Shorthand Notation: Write

- $P(h|r)$ instead of $P(humid \mid rainy)$
- $P(r^{(2)}|c^{(1)})$ instead of $P(S^{(2)} = rainy \mid S^{(1)} = cloudy)$
- $f_s^{(1:3)}$ to denote the entry relating to state $sunny$ in forward message $f^{(1:3)}$

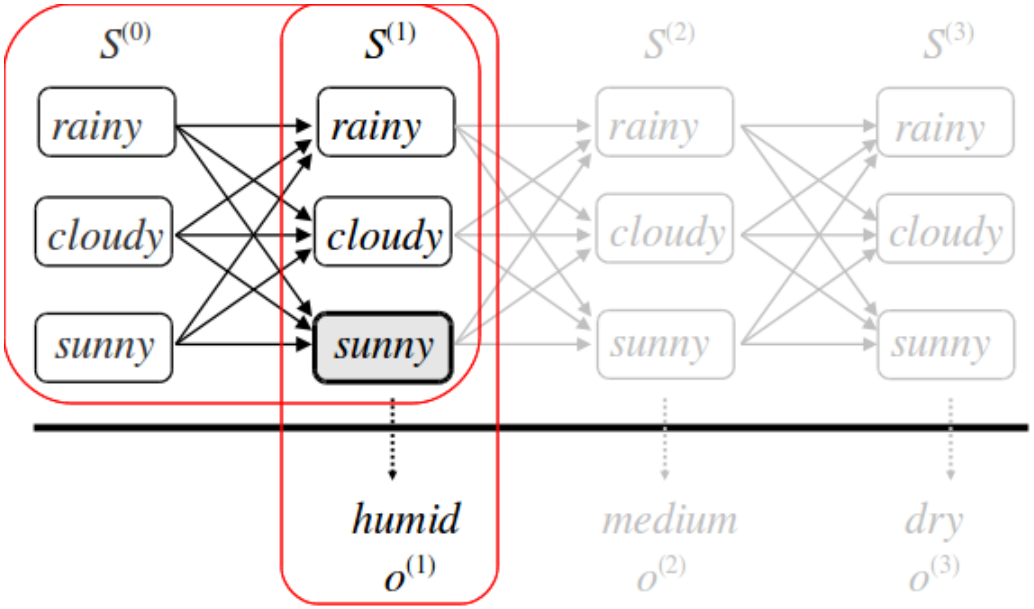
Filtering: A Simple Example



$t = 0$:

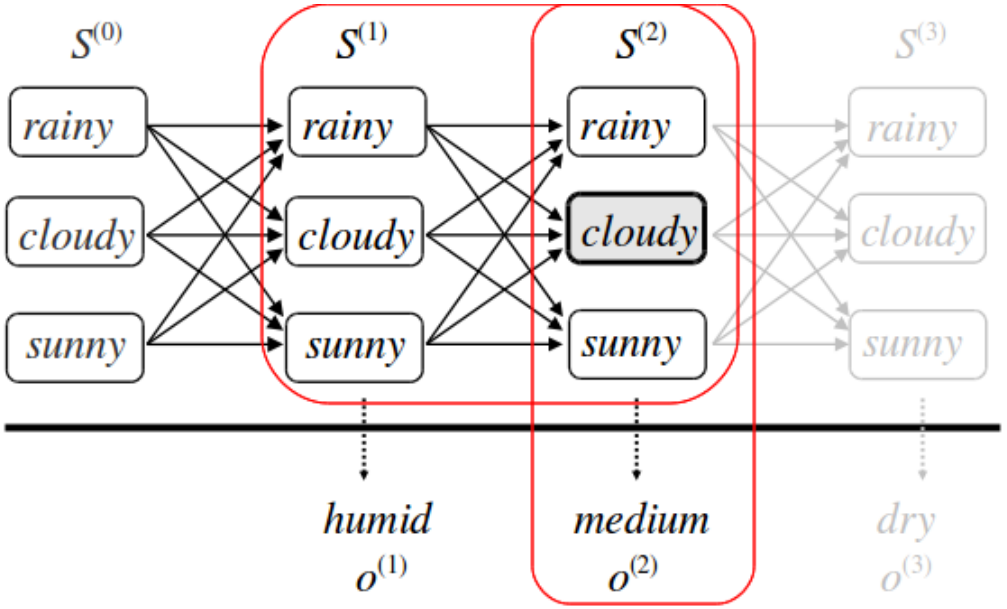
$$\mathsf{f}^{(1:0)} = P(S^{(0)}) = \mathbf{\Pi} = \begin{bmatrix} 0.2 \\ 0.3 \\ \mathbf{0.5} \end{bmatrix} \leftarrow \max : sunny$$

Filtering: A Simple Example



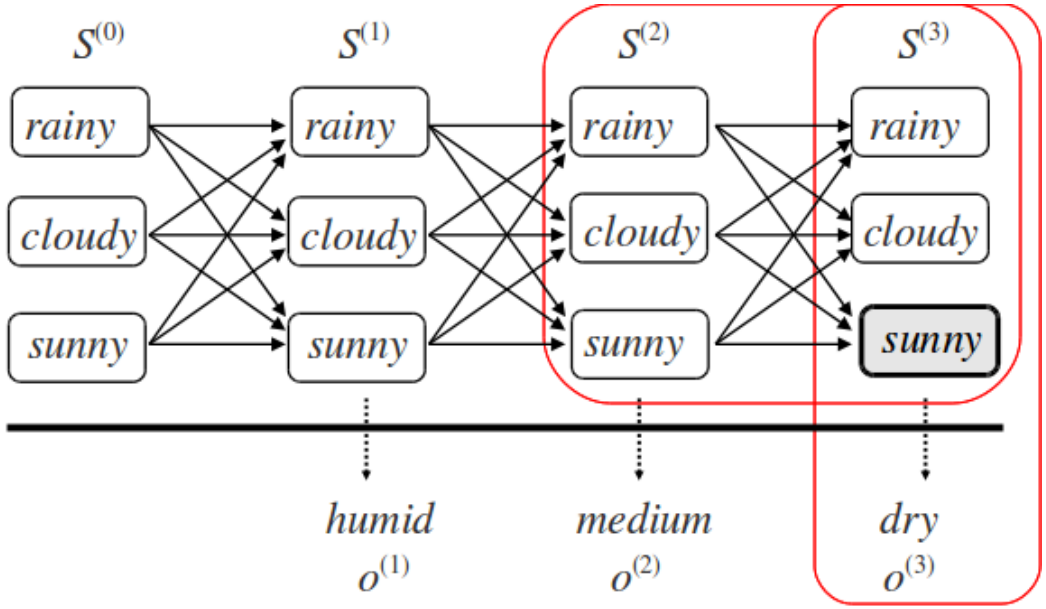
$$
\begin{aligned}
\mathsf{f}^{(1:1)} &= P(S^{(1)}|\langle h^{(1)}\rangle) \\
&= \frac{1}{Z}\begin{bmatrix} P(h|r)\times(P(r^{(1)}|r^{(0)})\mathsf{f}_r^{(1:0)} + P(r^{(1)}|c^{(0)})\mathsf{f}_c^{(1:0)} + P(r^{(1)}|s^{(0)})\mathsf{f}_s^{(1:0)}) \\ P(h|c)\times(P(c^{(1)}|r^{(0)})\mathsf{f}_r^{(1:0)} + P(c^{(1)}|c^{(0)})\mathsf{f}_c^{(1:0)} + P(c^{(1)}|s^{(0)})\mathsf{f}_s^{(1:0)}) \\ P(h|s)\times(P(s^{(1)}|r^{(0)})\mathsf{f}_r^{(1:0)} + P(s^{(1)}|c^{(0)})\mathsf{f}_c^{(1:0)} + P(s^{(1)}|s^{(0)})\mathsf{f}_s^{(1:0)}) \end{bmatrix} \\
&= \frac{1}{Z}\begin{bmatrix} 0.4\times(0.4\cdot 0.2 + 0.2\cdot 0.3 + 0.1\cdot 0.5) \\ 0.3\times(0.3\cdot 0.2 + 0.6\cdot 0.3 + 0.1\cdot 0.5) \\ 0.2\times(0.3\cdot 0.2 + 0.2\cdot 0.3 + 0.8\cdot 0.5) \end{bmatrix} = \frac{1}{Z}\begin{bmatrix} 0.076 \\ 0.087 \\ 0.104 \end{bmatrix} = \begin{bmatrix} 0.2846 \\ 0.3258 \\ \mathbf{0.3895} \end{bmatrix}
\end{aligned}
$$

Filtering: A Simple Example



$$
\begin{aligned}
\mathsf{f}^{(1:2)} &= P(S^{(2)}|\langle h^{(1)}, m^{(2)}\rangle) \\
&= \frac{1}{Z}\begin{bmatrix} P(m|r)\times(P(r^{(2)}|r^{(1)})\mathsf{f}_r^{(1:1)} + P(r^{(2)}|c^{(1)})\mathsf{f}_c^{(1:1)} + P(r^{(2)}|s^{(1)})\mathsf{f}_s^{(1:1)}) \\ P(m|c)\times(P(c^{(2)}|r^{(1)})\mathsf{f}_r^{(1:1)} + P(c^{(2)}|c^{(1)})\mathsf{f}_c^{(1:1)} + P(c^{(2)}|s^{(1)})\mathsf{f}_s^{(1:1)}) \\ P(m|s)\times(P(s^{(2)}|r^{(1)})\mathsf{f}_r^{(1:1)} + P(s^{(2)}|c^{(1)})\mathsf{f}_c^{(1:1)} + P(s^{(2)}|s^{(1)})\mathsf{f}_s^{(1:1)}) \end{bmatrix} \\
&= \frac{1}{Z}\begin{bmatrix} 0.1090 \\ 0.1279 \\ 0.0924 \end{bmatrix} = \begin{bmatrix} 0.3309 \\ \mathbf{0.3884} \\ 0.2806 \end{bmatrix}
\end{aligned}
$$

Filtering: A Simple Example

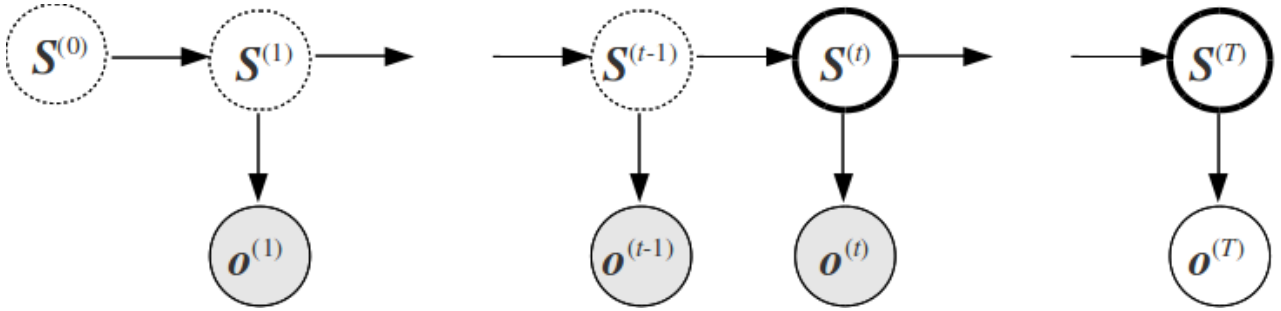


$$
\begin{aligned}
\mathsf{f}^{(1:3)} &= P(S^{(3)}|\langle h^{(1)}, m^{(2)}, d^{(3)}\rangle) \\
&= \frac{1}{Z}\begin{bmatrix} P(d|r) \times (P(r^{(3)}|r^{(2)})\mathsf{f}_r^{(1:2)} + P(r^{(3)}|c^{(2)})\mathsf{f}_c^{(1:2)} + P(r^{(3)}|s^{(2)})\mathsf{f}_s^{(1:2)}) \\ P(d|c) \times (P(c^{(3)}|r^{(2)})\mathsf{f}_r^{(1:2)} + P(c^{(3)}|c^{(2)})\mathsf{f}_c^{(1:2)} + P(c^{(3)}|s^{(2)})\mathsf{f}_s^{(1:2)}) \\ P(d|s) \times (P(s^{(3)}|r^{(2)})\mathsf{f}_r^{(1:2)} + P(s^{(3)}|c^{(2)})\mathsf{f}_c^{(1:2)} + P(s^{(3)}|s^{(2)})\mathsf{f}_s^{(1:2)}) \end{bmatrix} \\
&= \frac{1}{Z}\begin{bmatrix} 0.0238 \\ 0.1081 \\ 0.2409 \end{bmatrix} = \begin{bmatrix} 0.0639 \\ 0.2900 \\ \mathbf{0.6461} \end{bmatrix}
\end{aligned}
$$

Task 2: Prediction

Prediction

- Compute $P(S^{(t+k)} \mid \boldsymbol{o}^{(1:t)})$
- Needed to anticipate future situations



Remember General Prediction Algorithm for S-O Models

Given $\mathrm{p}^{(t+k)} = P(\boldsymbol{S}^{(t+k)} \mid \boldsymbol{o}^{(1:t)})$, predict next distribution as

$$\mathrm{p}^{(t+k+1)} = P(\boldsymbol{S}^{(t+k+1)} \mid \boldsymbol{o}^{(1:t)}) = \sum_{\boldsymbol{s}^{(t+k)}} \underbrace{P(\boldsymbol{S}^{(t+k+1)} | \boldsymbol{s}^{(t+k)})}_{\text{State Forward Propagation}} \mathrm{p}_{\boldsymbol{s}}^{(t+k)}$$

In Words:

The probability of a specific state $\boldsymbol{s}^{(t+k+1)}$ at time $t+k+1$ is the sum, over all possible predecessor states $\boldsymbol{s}^{(t+k)}$, of the probability of having been in $\boldsymbol{s}^{(t+k)}$ at $t+k$ and then moving from $\boldsymbol{s}^{(t+k)}$ to $\boldsymbol{s}^{(t+k+1)}$ in one step.

Specialised to HMM:

$$\boxed{\mathrm{p}^{(t+k+1)} = \boldsymbol{A}^T \mathrm{p}^{(t+k)}}$$

Prediction in HMMs: The Forward Algorithm

FORWARD ALGORITHM for PREDICTION

Initialisation:

$$\mathsf{p}^{(t+0)} = P(S^{(t)} \mid \boldsymbol{o}^{(1:t)})$$

(computed via Filtering Algorithm)

Recursive Forward Propagation:

$$\boxed{\mathsf{p}^{(t+k+1)}} = P(S^{(t+k+1)} \mid \boldsymbol{o}^{(1:t)}) = \boxed{\boldsymbol{A}^T \mathsf{p}^{(t+k)}}$$

... or, if you are not so good at reading matrix equations:

$$\text{for } i = 1 \text{ to } N: \quad \boxed{\mathsf{p}_i^{(t+k+1)}} = P(S^{(t+k+1)} = s_i \mid \boldsymbol{o}^{(1:t)})$$

$$= \boxed{\sum_j a_{ji} \mathsf{p}_j^{(t+k)}}$$

Prediction: A Simple Example

Consider our simple weather HMM:

$$S = \{rainy, cloudy, sunny\}$$
$$O = \{dry, medium, humid\}$$

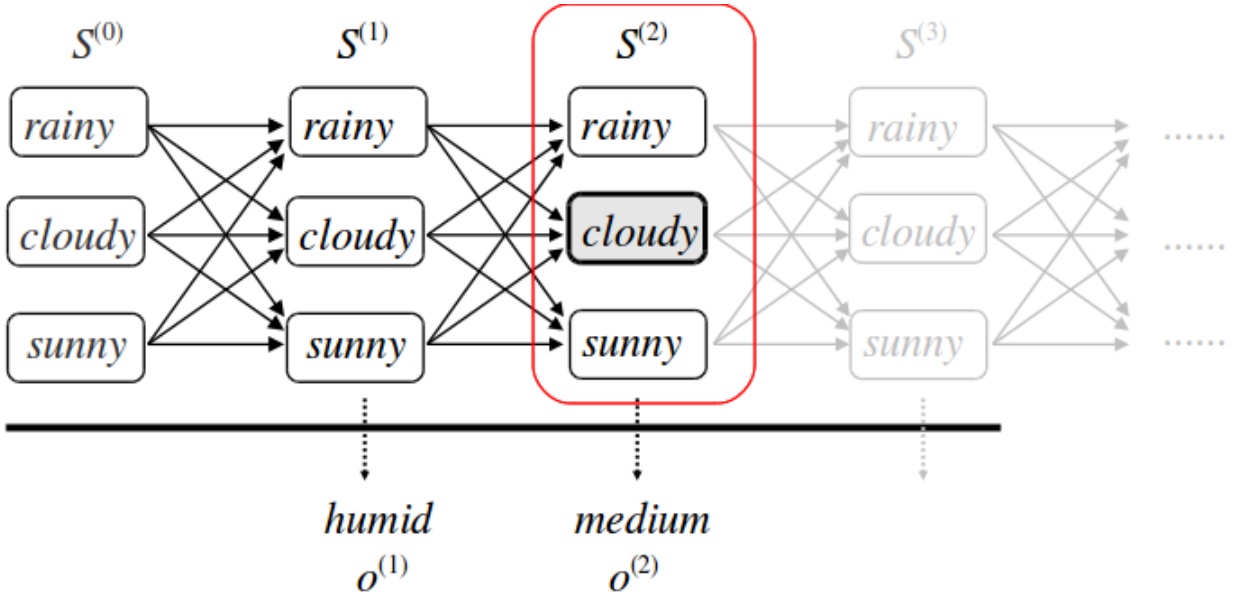
$$\boldsymbol{A} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \qquad \boldsymbol{B} = \begin{array}{c} \\ \text{dry} \\ \text{medium} \\ \text{humid} \end{array} \begin{array}{c} \begin{array}{ccc} \text{rainy} & \text{cloudy} & \text{sunny} \end{array} \\ \begin{pmatrix} 0.1 & 0.3 & 0.6 \\ 0.5 & 0.4 & 0.2 \\ 0.4 & 0.3 & 0.2 \end{pmatrix} \end{array} \qquad \boldsymbol{\Pi} = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.5 \end{bmatrix}$$

... and the following sequence of observations for two days $1, 2$:

$$\langle humid, medium \rangle$$

$\Rightarrow$ **What is the most probable continuation of the weather for the next days?**

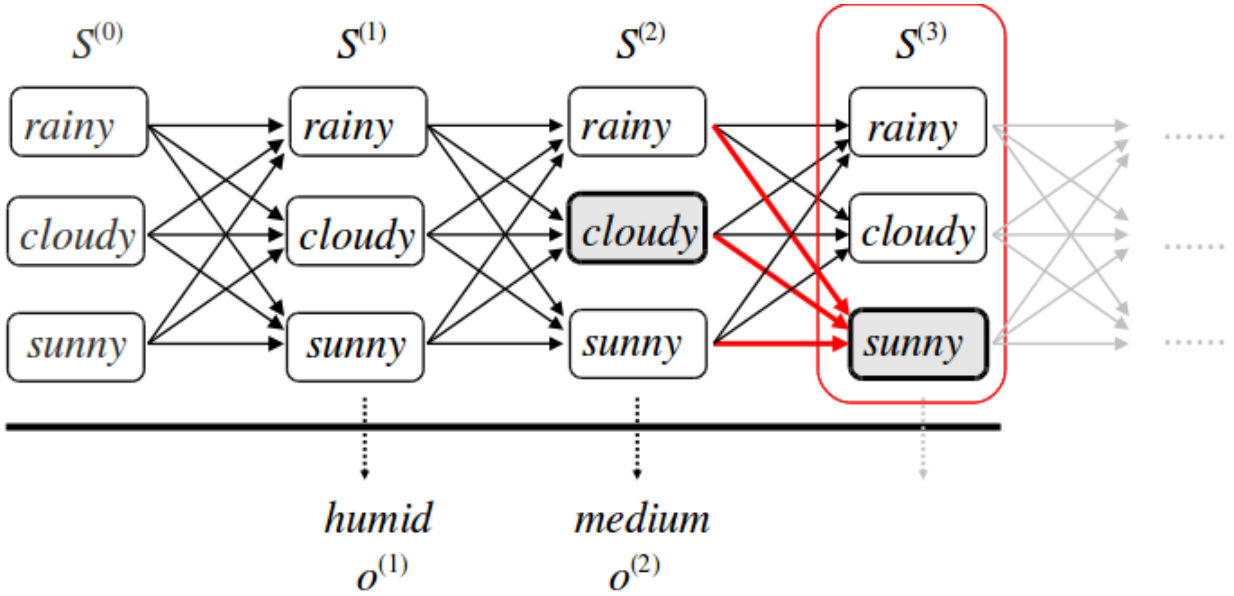
Prediction: A Simple Example



Starting Point:

$$\mathsf{p}^{(2)} = \mathsf{f}^{(1:2)} = P(S^{(2)} \mid \langle h^{(1)}, m^{(2)} \rangle) = \begin{bmatrix} 0.3309 \\ \mathbf{0.3884} \\ 0.2806 \end{bmatrix}$$

Prediction: A Simple Example



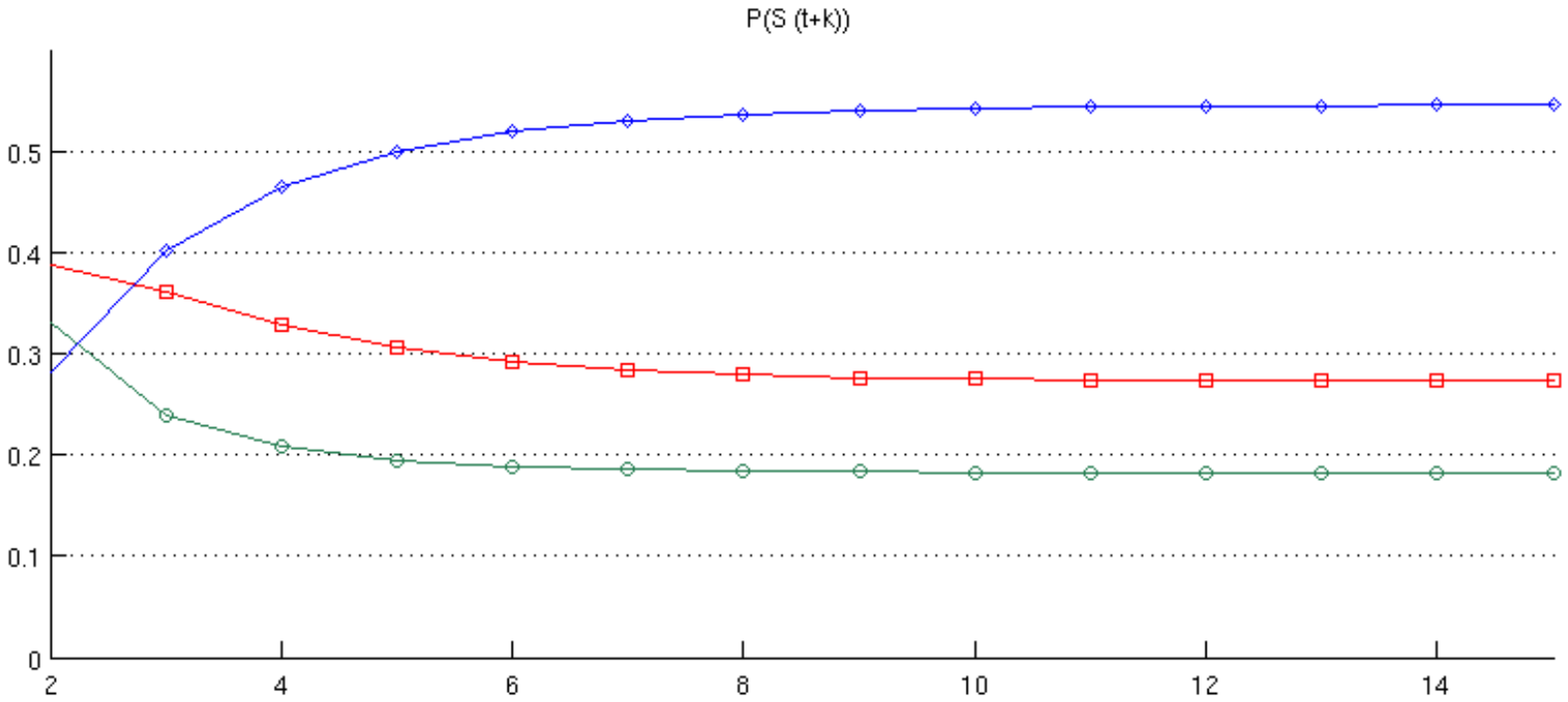
$$\mathsf{p}^{(3)} = P(S^{(3)} \mid \langle h^{(1)}, m^{(2)} \rangle) = \begin{bmatrix} 0.2381 \\ 0.3604 \\ \mathbf{0.4015} \end{bmatrix}$$

$$\mathsf{p}^{(4)} = P(S^{(4)} \mid \langle h^{(1)}, m^{(2)} \rangle) = \begin{bmatrix} 0.2075 \\ 0.3278 \\ \mathbf{0.4647} \end{bmatrix}$$

etc. etc.

Prediction: A Simple Example

Prediction of $P(S^{(t+k)})$ for the next 13 time steps:



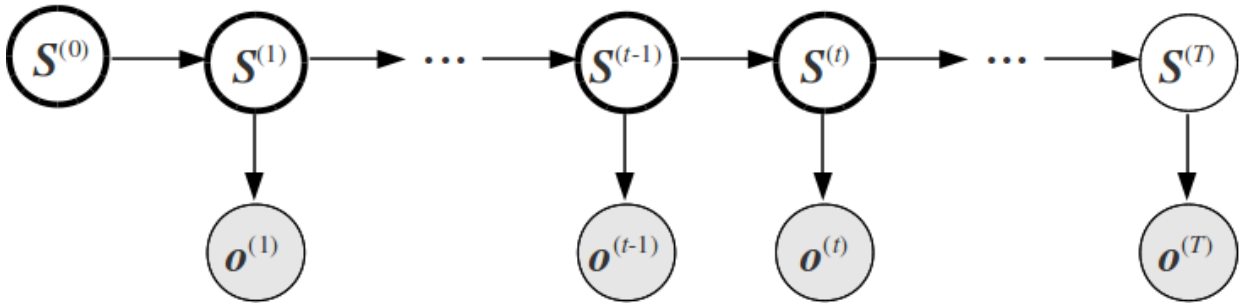
☞ **With growing k, $P(S^{(t+k)}|o^{(1:t)})$ converges to the stationary distribution of the Markov chain (transition model) $P(S'|S)$:**

$$P^*(S \mid \langle h^{(1)}, m^{(2)} \rangle) = P^*(S) \approx \begin{bmatrix} 0.1818 \\ 0.2727 \\ \mathbf{0.5455} \end{bmatrix}$$

Task 3: Smoothing

Smoothing

- Compute $P(S^{(k)} \mid \boldsymbol{o}^{(1:T)}), k < T$
- Needed for better assessment of situation (and for learning)



Remember General Smoothing Algorithm for S-O Models

$$\underline{P(\boldsymbol{S}^{(k)} \mid \boldsymbol{o}^{(1:T)})} = 1/Z \times \underbrace{P(\boldsymbol{S}^{(k)} \mid \boldsymbol{o}^{(1:k)})}\, \underbrace{P(\boldsymbol{o}^{(k+1:T)} \mid \boldsymbol{S}^{(k)})}$$

$$= \underline{1/Z \times \quad \mathsf{f}^{(1:k)} \quad \times \quad \mathsf{b}^{(k+1:T)}}$$

where

$$\mathsf{f}^{(1:k)} = \text{as computed by forward algorithm (filtering)}$$

$$\mathsf{b}^{(k+1:T)} = \sum_{\boldsymbol{s}^{(k+1)}} \underbrace{P(\boldsymbol{s}^{(k+1)}|\boldsymbol{S}^{(k)})}_{\text{Transition Model}} \underbrace{P(\boldsymbol{o}^{(k+1)}|\boldsymbol{s}^{(k+1)})}_{\text{Observation Model}} \underbrace{P(\boldsymbol{o}^{(k+2:T)}|\boldsymbol{s}^{(k+1)})}_{\mathsf{b}^{(k+2:T)}}$$

Specialised to HMMs:

$$\boxed{\mathsf{b}^{(k+1:T)} = \boldsymbol{A}\left(\boldsymbol{B}[o^{(k+1)}]^T \circ \mathsf{b}^{(k+2:T)}\right)}$$

where

- $\boldsymbol{B}[o^{(k+1)}]$ is the row of $\boldsymbol{B}$ pertaining to observation $o^{(k+1)}$
- $\circ$ denotes the *element-wise (Hadamard) product* of two matrices.

Computing the Backward Message

Function BACKWARD($o^{(k+1)}$, $\mathsf{b}^{(k+2:T)}$)

Input:

- Observation $o^{(k+1)}$ at time $k+1$
- Backward message $\mathsf{b}^{(k+2:T)} = P(\boldsymbol{o}^{(k+2:T)} \mid S^{(k+1)})$
  (a column vector of dimension $N = |Val(S)|$)

Computes

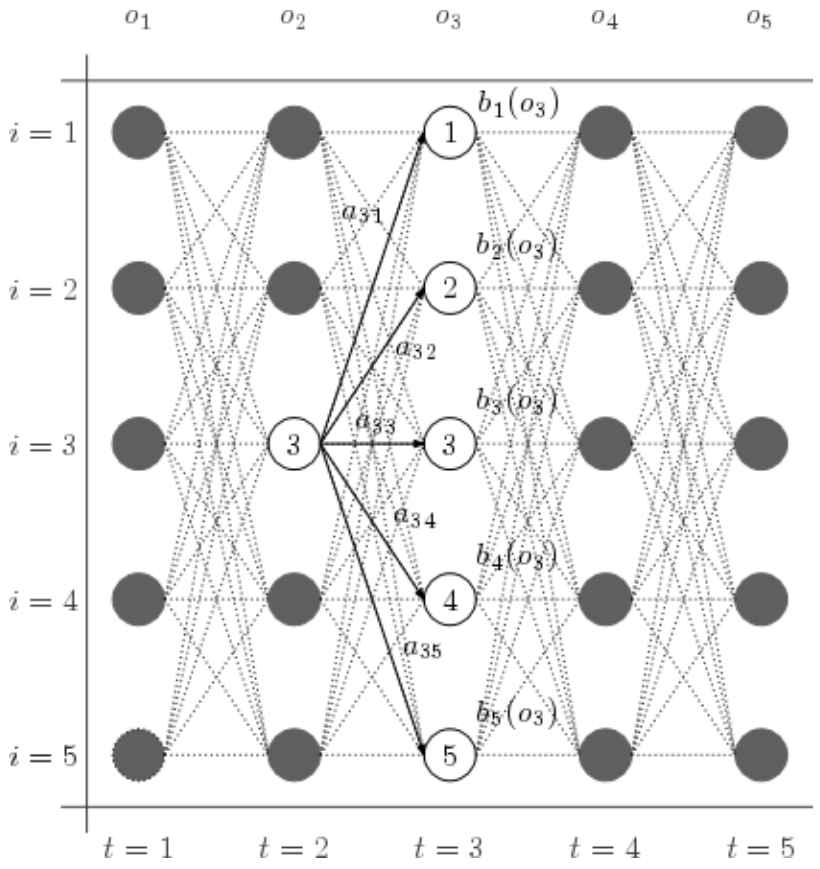
$$\mathsf{b}^{(k+1:T)} = P(\boldsymbol{o}^{(k+1:T)} \mid S^{(k)}) = \boldsymbol{A}\left(\boldsymbol{B}[o^{(k+1)}]^T \circ \mathsf{b}^{(k+2:T)}\right)$$

as follows:

for $i = 1$ to N : $\boxed{\mathsf{b}_i^{(k+1:T)} = \sum_j a_{ij} b_j(o^{(k+1)}) \mathsf{b}_j^{(k+2:T)}}$

The Intuition Behind the Backward Step

$$\mathsf{b}_i^{(k+1:T)} = \sum_j a_{ij} b_j(o^{(k+1)}) \mathsf{b}_j^{(k+2:T)}$$



In Words:

The probability of observing $\boldsymbol{o}^{(k+1:T)}$, given that at k we are in state s_i, is the sum, over all successor states s_j, of the probability of moving to s_j from s_i, then observing $o^{(k+1)}$ in s_j, and then observing the rest of the sequence $\boldsymbol{o}^{(k+2)}$ from s_j onwards.

Smoothing in HMMs: The Forward-Backward Algorithm

The FORWARD-BACKWARD ALGORITHM for HMMs

GOAL: Compute $P(S^{(k)} \mid \boldsymbol{o}^{(1:T)})$, for all $k = 0, \ldots, T$

- **Initialise f:** $\underline{\mathsf{f}^{(1:0)} = \boldsymbol{\Pi}}$
- **Forward Pass:** For $k = 1$ to T compute (and then ***store***):

$$\underline{\mathsf{f}^{(1:k)} = 1/Z \times \textsc{Forward}(\mathsf{f}^{(1:k-1)}, o^{(k)})}$$

- **Initialise b:** $\underline{\mathsf{b}^{(T+1:T)} = \mathbf{1}}$
- **Backward Pass:** For $k = T$ downto 1 do
  1. Compute smoothed posterior distribution at k:

$$\boxed{P(S^{(k)} \mid \boldsymbol{o}^{(1:T)}) = 1/Z \times \mathsf{f}^{(1:k)} \circ \mathsf{b}^{(k+1:T)}}$$

  2. Pass backward message backwards:

$$\underline{\mathsf{b}^{(k:T)} = \textsc{Backward}(o^{(k)}, \mathsf{b}^{(k+1:T)})}$$

Smoothing: A Simple Example

Consider our simple weather HMM:

$$S = \{rainy, cloudy, sunny\}$$
$$O = \{dry, medium, humid\}$$

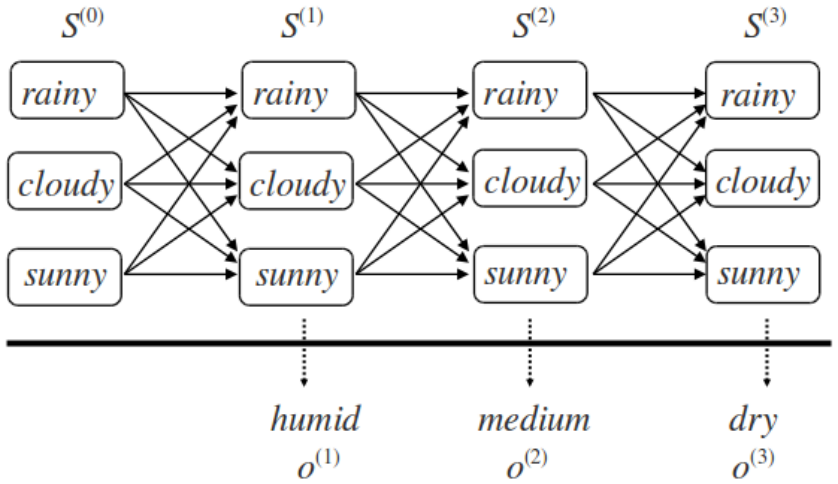
$$\boldsymbol{A} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \qquad \boldsymbol{B} = \begin{array}{c} \\ \text{dry} \\ \text{medium} \\ \text{humid} \end{array} \begin{array}{c} \begin{array}{ccc} \text{rainy} & \text{cloudy} & \text{sunny} \end{array} \\ \begin{pmatrix} 0.1 & 0.3 & 0.6 \\ 0.5 & 0.4 & 0.2 \\ 0.4 & 0.3 & 0.2 \end{pmatrix} \end{array} \qquad \boldsymbol{\Pi} = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.5 \end{bmatrix}$$

... and the following sequence of observations:

$$\langle humid, medium, dry \rangle$$

$\Rightarrow$ **Considering this whole sequence, what was the most probable weather on any of the 3 days?**

Smoothing: A Simple Example



$$\mathsf{f}^{(1:0)}\begin{bmatrix}0.2000\\0.3000\\\mathbf{0.5000}\end{bmatrix} \rightarrow \mathsf{f}^{(1:1)}\begin{bmatrix}0.2846\\0.3258\\\mathbf{0.3895}\end{bmatrix} \rightarrow \mathsf{f}^{(1:2)}\begin{bmatrix}0.3309\\\mathbf{0.3884}\\0.2806\end{bmatrix} \rightarrow \mathsf{f}^{(1:3)}\begin{bmatrix}0.0639\\0.2900\\\mathbf{0.6461}\end{bmatrix}$$

$$\mathsf{b}^{(1:3)}\begin{bmatrix}0.0393\\0.0381\\0.0270\end{bmatrix} \leftarrow \mathsf{b}^{(2:3)}\begin{bmatrix}0.1316\\0.1286\\0.1115\end{bmatrix} \leftarrow \mathsf{b}^{(3:3)}\begin{bmatrix}0.3100\\0.3200\\0.5200\end{bmatrix} \leftarrow \mathsf{b}^{(4:3)}\begin{bmatrix}1.0000\\1.0000\\1.0000\end{bmatrix}$$

$$\underline{P(S^{(t)}|\boldsymbol{o}^{(1:3)})}\begin{bmatrix}0.2399\\0.3490\\\mathbf{0.4112}\end{bmatrix} \quad \begin{bmatrix}0.3051\\0.3413\\\mathbf{0.3537}\end{bmatrix} \quad \begin{bmatrix}0.2751\\0.3334\\\mathbf{0.3914}\end{bmatrix} \quad \begin{bmatrix}0.0639\\0.2900\\\mathbf{0.6461}\end{bmatrix}$$

Observations

Effect of Smoothing:

- $\arg\max_i P(s_i^{(2)}|\boldsymbol{o}^{(1:3)}) \neq \arg\max_i P(s_i^{(2)}|\boldsymbol{o}^{(1:2)})$
- In words: hindsight changes our belief as to what the most probable weather state was on day 2
- Do you understand why? (Hint: Day 3 was most likely sunny, and probability of a cloudy day being followed by a sunny one is rather low ...)

☞ ***"Smoothing"*** of state sequence ...

Mathematical Observations:

- Forward message $\mathrm{f}^{(1:k)} = P(S^{(k)}|\boldsymbol{o}^{(1:k)})$ is a proper probability distribution over S, but backward message $\mathrm{b}^{(k+1:T)} = P(\boldsymbol{o}^{(k+1:T)}|S^{(k)})$ is **not**: it is a **likelihood function** – the probability of seeing a fixed future observation sequence $\boldsymbol{o}^{(k+1:T)}$, given a model and a current state $s_i^{(k)}$
- $\mathrm{b}^{(k:T)}$ is getting exponentially smaller as k goes to 1, and thus $\boldsymbol{o}^{(k:T)}$ becomes longer and more specific. Will lead to serious numerical problems.
  ☞ Need for **re-scaling** (or work with **logarithms**)!

Task 4: Decoding ("Finding the Most Likely Explanation")

The Decoding Problem

Compute the most likely sequence of states that might have produced a given observation sequence $\boldsymbol{o}^{(1:T)}$:

$$\begin{aligned}\hat{\boldsymbol{s}}^{(0:T)} &= \underset{s^{(0)},\ldots,s^{(T)}}{\arg\max}\, P(s^{(0)},\ldots,s^{(T)} \mid \boldsymbol{o}^{(1:T)}) \\ &= \underset{s^{(0)},\ldots,s^{(T)}}{\arg\max}\, P(s^{(0)},\ldots,s^{(T)}, \boldsymbol{o}^{(1:T)})\end{aligned}$$

Example

- Speech Recognition: given an audio signal, what was the most likely sequence of syllables that were spoken (= 'produced' the observed signal)?
- Speech Denoising: given possibly corrupted encoded bit stream, identify the most probable true bit sequence [1]

[1] https://viterbischool.usc.edu/news/2017/03/viterbi-algorithm-50

Task 4: Decoding ("Finding the Most Likely Explanation")

NOTE:

- The most likely state sequence is **not** the same as the sequence of most likely states as computed in smoothing!
- Smoothing computes distributions $P(S^{(t)} \mid \boldsymbol{o}^{(1:T)})$ over *isolated states (time points)* $S^{(t)}$
- In contrast, decoding must select from distribution $P(\boldsymbol{S}^{(0:T)} \mid \boldsymbol{o}^{(1:T)})$ over **sequences** of states $\boldsymbol{S}^{(0:T)}$.

A little exercise for you (non-trivial):

- *Construct a temporal model and an observation sequence where the sequence of most likely states as computed by filtering/smoothing would correspond to an **impossible** state sequence – i.e., a sequence with*

$$P(\boldsymbol{s}^{(0:T)} \mid \boldsymbol{o}^{(1:T)}) = 0.$$

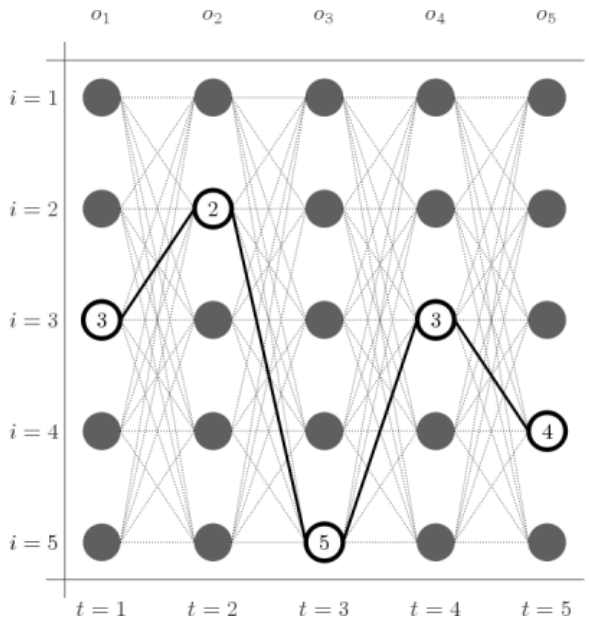
Decoding: The Naive Approach

Naive Algorithm:

- Compute full distribution $\boldsymbol{P}(\boldsymbol{S}^{(0:T)}, \boldsymbol{o}^{(1:T)})$ over all state sequences:
- enumerate all state sequences $\boldsymbol{s}^{(0:T)} = \langle s^{(0)}, \ldots, s^{(T)} \rangle$ and compute

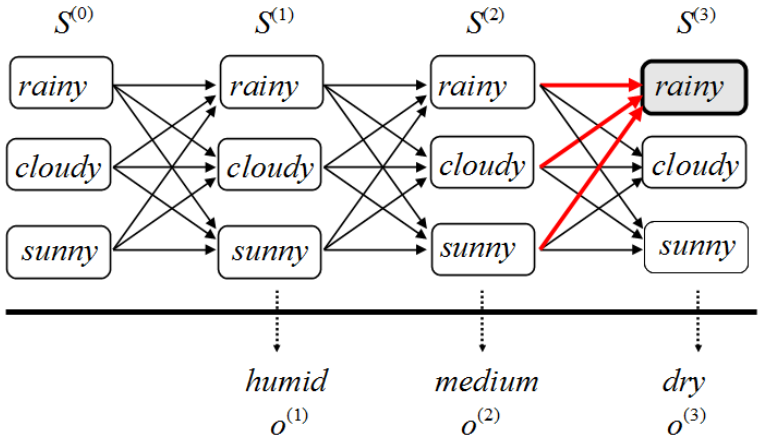
$$P(\boldsymbol{s}^{(0:T)}, \boldsymbol{o}^{(1:T)}) = \pi_{s^{(0)}} \prod_{t=1}^{T} a_{s^{(t-1)}, s^{(t)}} b_{s^{(t)}}(o^{(t)})$$

(an atomic event ...)



Problem: Combinatorial number of possible state sequences!

Efficient Decoding: The Intuition



Consider paths that reach state $rainy^{(3)}$:

- Most likely path to $rainy^{(3)}$ consists of most likely path to *some* state $s_i^{(2)}$ at $t = 2$, followed by transition from $s_i^{(2)}$ to $rainy^{(3)}$ and observing $o^{(3)}$
- The **probability** of that path is $P(most\ likely\ path\ to\ s_i^{(2)}) \times P(s_i^{(2)} \rightarrow rainy) \times P(o^3|rainy)$
- Most likely path to $rainy^{(3)}$ will go through that **previous state** $s_i^{(2)}$ at $t = 2$ that maximises the product $P(most\ likely\ path\ to\ s_i^{(2)}) \times a_{i,rainy}$
- Suggests recursive algorithm very similar to Forward Algorithm

Basic Idea of the Viterbi Algorithm

- Propagate forward **"Viterbi message"**

$$\mathsf{v}^{(1:t)} = \max_{s^{(0)} \ldots s^{(t-1)}} P(s^{(0)}, \ldots, s^{(t-1)}, S^{(t)}, \boldsymbol{o}^{(1:t)})$$

In words: i^{th} component of vector $\mathsf{v}^{(1:t)}$ is probability of the most likely way of producing observations $\boldsymbol{o}^{(1:t)}$ **and ending in state** $s_i^{(t)}$ at time t

- Recursively compute $\mathsf{v}^{(1:t+1)}$ from $\mathsf{v}^{(1:t)}$
- For each state $s_i^{(t)}$, keep **back pointer** to best state that leads to it:[2]

$$\mathsf{bp}_i^{(t)} = \arg\max_j \ \mathsf{v}_j^{(1:t-1)} a_{ji}$$

In words: State $s_i^{(t)}$ points to state $s_j^{(t-1)}$ at $t-1$ that maximises $P(\textit{most likely path to } s_j^{(t-1)}) \times P(s_j \to s_i)$

- At end, trace best path by following back pointers from best final state.

[2] see comment 3 on previous slide

Decoding in HMMs: The Viterbi Algorithm

The Viterbi Algorithm

Initialisation:

$$\mathsf{v}^{(1:0)} = \mathbf{\Pi}$$

Recursive forward propagation:

$$\text{for } i = 1 \text{ to } N: \quad \mathsf{v}_i^{(1:t+1)} = \max_j \mathsf{v}_j^{(1:t)} a_{ji} b_i(o^{(t+1)})$$

$$\mathsf{bp}_i^{(t+1)} = \arg\max_j \mathsf{v}_j^{(1:t)} a_{ji}$$

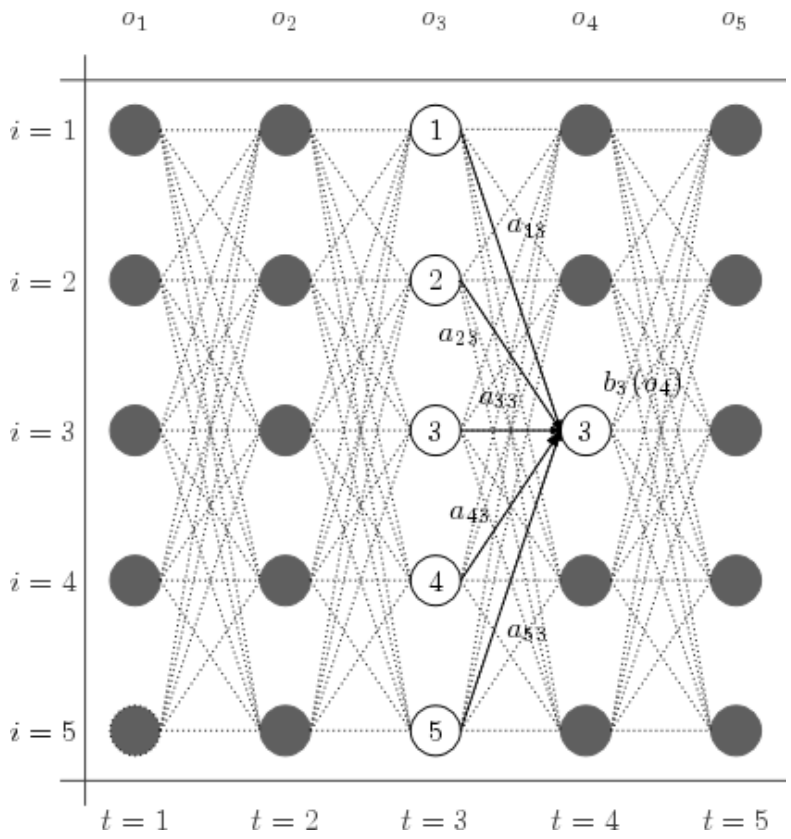
Termination: Trace back path from best final state

$$s^{(T)*} = \arg\max_i \mathsf{v}_i^{(1:T)}$$

using series of $\mathsf{bp}^{(t)}$ vectors.

Once More: The Intuition Behind the Viterbi Decoding Step

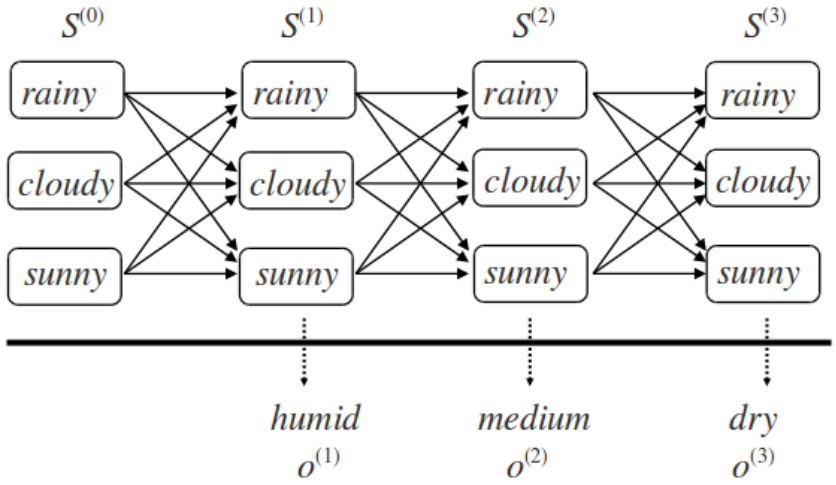
$$\mathsf{v}_i^{(1:t+1)} = \max_j \mathsf{v}_j^{(1:t)} a_{ji} b_i(o^{(t+1)})$$



In Words:

The probability of the most likely path ending in state s_i at time $t + 1$ along with the observation sequence $\boldsymbol{o}^{(1:t+1)}$ is the maximum, over all predecessor states s_j, of the maximum probability path to s_j at time t, times the probability of the one-step transition $s_j \rightarrow s_i$, times the probability of then observing $o^{(t+1)}$ in state $s_i^{(t+1)}$.

Decoding: A Simple Example



$$\mathsf{v}^{(1:0)} \begin{bmatrix} 0.2000 \\ 0.3000 \\ 0.5000 \end{bmatrix} \rightarrow \mathsf{v}^{(1:1)} \begin{bmatrix} 0.0320 \\ 0.0540 \\ 0.0800 \end{bmatrix} \rightarrow \mathsf{v}^{(1:2)} \begin{bmatrix} 0.0064 \\ 0.0130 \\ 0.0128 \end{bmatrix} \rightarrow \mathsf{v}^{(1:3)} \begin{bmatrix} 0.0003 \\ 0.0023 \\ \mathbf{0.0061} \end{bmatrix}$$

$$\mathsf{bp}^{(1)} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \leftarrow \qquad \mathsf{bp}^{(2)} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \leftarrow \qquad \mathsf{bp}^{(3)} \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix} \leftarrow \mathbf{3}$$

---

sunny *sunny* *sunny* *sunny*

Task 5: Computing the Likelihood of a HMM

Computing the LIKELIHOOD of a HMM

Goal:
Compute the **likelihood** of the HMM $\mathcal{M}$, given observation sequence $\boldsymbol{o}^{(1:T)}$:

$$L(\mathcal{M} : \boldsymbol{o}^{(1:T)}) = P_{\mathcal{M}}(\boldsymbol{o}^{(1:T)})$$

In other words:
Compute the probability with which $\mathcal{M}$ would produce the sequence $\boldsymbol{o}^{(1:T)}$

Motivation:

- Useful for rating different temporal models that might have produced the same evidence sequence
- Example: speech recognition; a given sequence of sounds might be the result of different words that were spoken
- Will be used for **classification** (e.g., see speech understanding below).

Computing the Likelihood of a HMM

Observation:

$$P(\boldsymbol{o}^{(1:T)}) = \sum_i P(\boldsymbol{o}^{(1:T)}, S^{(T)} = s_i)$$

In Words:

The probability of the model producing a sequence $\boldsymbol{o}^{(1:T)}$ is equal to the probability, over all possible final states $s_i^{(T)}$, of the model producing sequence $\boldsymbol{o}^{(1:T)}$ and ending in state $s_i^{(T)}$ *(Law of total probability)*.

Motivation for a simple forward algorithm:

- Compute $P(\boldsymbol{o}^{(1:T)}, S^{(T)} = s_i)$ for all possible end states s_i, then sum over all s_i to obtain $P(\boldsymbol{o}^{(1:T)})$
- $P(\boldsymbol{o}^{(1:t)}, S^{(t)} = s_i)$ can be computed recursively, similar to filtering
- Propagate forward a "likelihood forward message" $\text{lf}^{(1:t)}$, using the same FORWARD function as in filtering (without renormalisation)

Computing the Likelihood of a HMM

Forward Algorithm for Likelihood Computation

Initialisation:

$$\mathsf{lf}^{(1:0)} = P(\boldsymbol{o}^{(1:0)}, S^{(0)}) = P(S^{(0)}) = \boldsymbol{\Pi}$$

Recursive forward propagation:

$$\mathsf{lf}^{(1:t+1)} = \textsc{Forward}(\mathsf{lf}^{(1:t)}, o^{(t+1)})$$

Termination:

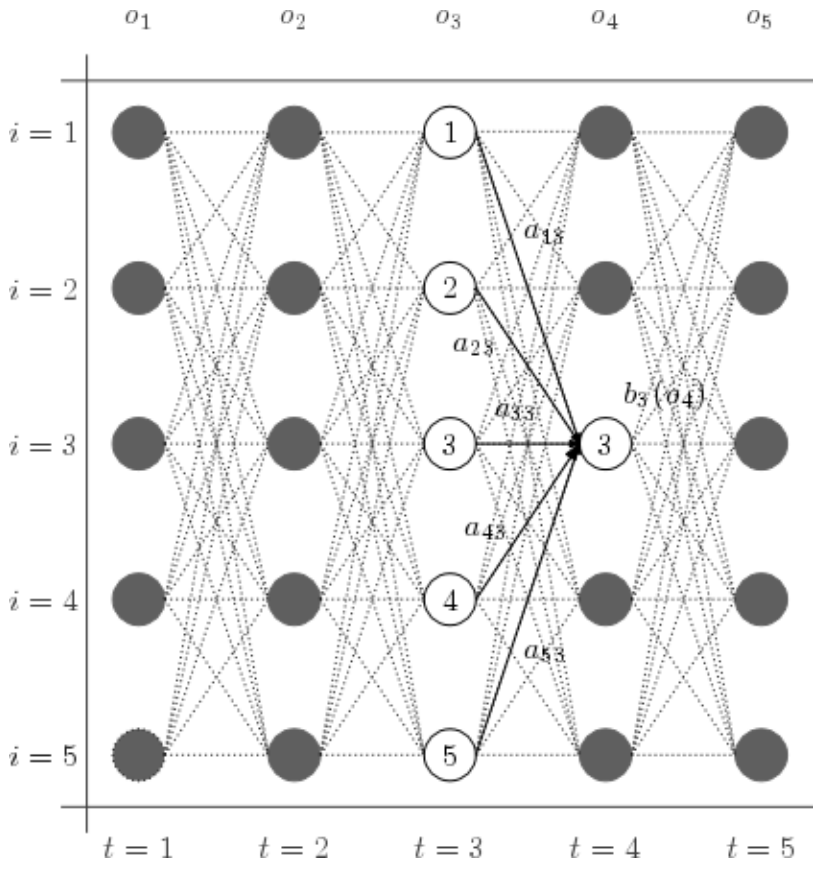
$$\boxed{P(\boldsymbol{o}^{(1:T)})} = \sum_i P(\boldsymbol{o}^{(1:T)}, S^{(T)} = s_i) = \boxed{\sum_i \mathsf{lf}_i^{(1:T)}}$$

Notes:

- Identical to Forward Filtering Algorithm, except for re-normalisation
- No need for re-normalisation here: instead of *conditional distribution* $P(S_i^{(t)}|\boldsymbol{o}^{(1:t)})$, we compute $P(S_i^{(t)} \wedge \boldsymbol{o}^{(1:t)})$

The Intuition Behind the Forward Step

$$
\begin{aligned}
&\mathsf{lf}_i^{(1:t+1)} \\
&= P(\boldsymbol{o}^{(1:t+1)}, S^{(t+1)} = s_i) \\
&= \left[\sum_j \mathsf{lf}_j^{(1:t)} a_{ji} \right] b_i(o^{(t+1)})
\end{aligned}
$$



In Words:

The probability of observing $\boldsymbol{o}^{(1:t+1)}$ and ending up in state s_i at time $t+1$ is the sum, over all predecessor states s_j, of having observed $\boldsymbol{o}^{(1:t)}$ and ended up in s_j at time t, then moving from s_j to s_i, and then observing $o^{(t+1)}$ in state s_i.

HMM Likelihood: A Simple Example

Consider our simple weather HMM:

$$S = \{rainy, cloudy, sunny\}$$
$$O = \{dry, medium, humid\}$$

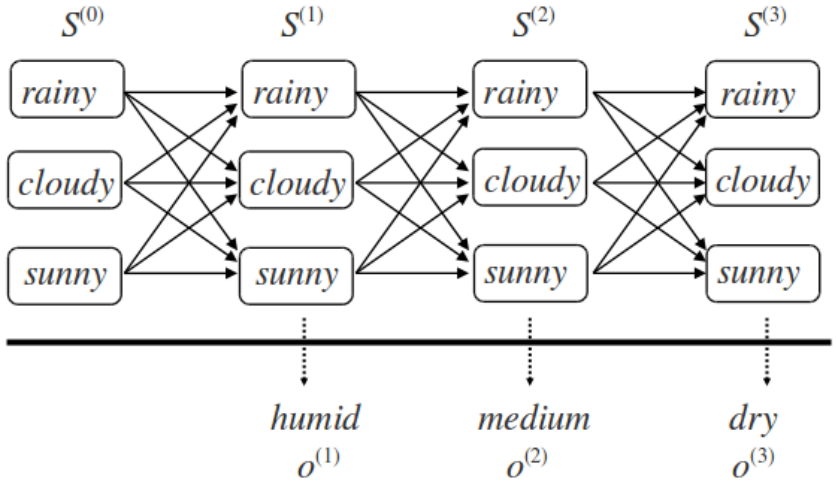
$$\boldsymbol{A} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \qquad \boldsymbol{B} = \begin{array}{c} \\ \text{dry} \\ \text{medium} \\ \text{humid} \end{array} \begin{array}{c} \begin{array}{ccc} \text{rainy} & \text{cloudy} & \text{sunny} \end{array} \\ \begin{pmatrix} 0.1 & 0.3 & 0.6 \\ 0.5 & 0.4 & 0.2 \\ 0.4 & 0.3 & 0.2 \end{pmatrix} \end{array} \qquad \boldsymbol{\Pi} = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.5 \end{bmatrix}$$

... and the following sequence of observations for three days:

$$\langle humid, medium, dry \rangle$$

⇒ **What is the probability that our weather HMM would produce exactly this sequence?**

HMM Likelihood: A Simple Example



$$\mathsf{lf}^{(1:0)} \begin{bmatrix} 0.2000 \\ 0.3000 \\ 0.5000 \end{bmatrix} \rightarrow \mathsf{lf}^{(1:1)} \begin{bmatrix} 0.0760 \\ 0.0870 \\ 0.1040 \end{bmatrix} \rightarrow \mathsf{lf}^{(1:2)} \begin{bmatrix} 0.0291 \\ 0.0342 \\ 0.0247 \end{bmatrix} \rightarrow \mathsf{lf}^{(1:3)} \begin{bmatrix} 0.0021 \\ 0.0095 \\ 0.0212 \end{bmatrix}$$

$$\Sigma = \mathbf{0.0328}$$

Learning HMMs from Observation Sequences

Where do HMMs come from?

- Structure is automatically defined, once state and observation variables are given
- Problem: find good parameters $\boldsymbol{\Pi}$, $\boldsymbol{A}$, $\boldsymbol{B}$
- In some applications, appropriate parameters may be determined 'manually' (at least in part – see music beat tracking example below)
- In most cases, however, we do not know the dynamics of a complex hidden system, or how observations depend on hidden states

☞ Model parameters must be **learned**.

Additional Complication:

- True states of the system are unobservable, perhaps impossible to determine

☞ Must learn model parameters from **example observation sequences only!**

☞ Will formulate HMM parameter estimation as an **optimisation problem**.

Learning HMMs from Observation Sequences

THE LEARNING TASK

Given:

- a fixed set of possible **hidden states** $S = \{s_1, s_2, \ldots, s_N\}$
- a fixed vocabulary of **observation symbols** $O = \{o_1, o_2, \ldots, o_M\}$
- a **training set** $\mathcal{O}$ of observation sequences $\mathcal{O} = \{\boldsymbol{O}_1, \boldsymbol{O}_2, \ldots, \boldsymbol{O}_K\}$

Find (estimate):

- **Parameters** $\hat{\boldsymbol{\theta}} = \{\boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{B}\}$ that give the highest **likelihood** $L(\hat{\boldsymbol{\theta}} : \mathcal{O})$:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} P(\mathcal{O} \mid \boldsymbol{\theta})$$

In Words:

Find the parametrisation (transition model $\boldsymbol{A}$, observation model $\boldsymbol{B}$, starting probabilities $\boldsymbol{\Pi}$) for a HMM (out of the infinite set of possible parametrisations) that would have the highest probability of producing the given observation sequences $\mathcal{O}$.

The Central Problem: Hidden Variables

Central problem in learning HMMs:

- HMM contains **unobservable** variables (the states, and the transitions between them)
- Cannot be observed in training sequences
  $\Rightarrow$ how can they be estimated via counting?

  (Consider learning a weather HMM only from sequences of air humidity measurements ...)
- Problem concerns all three sets of parameters:
  $\boldsymbol{\Pi} = P(S^{(0)})$
  $\boldsymbol{A} = P(S' \mid S)$
  $\boldsymbol{B} = P(O \mid S)$
- Will develop a “magic” algorithm that finds an approximate solution

Some Thoughts: On the One Hand ...

If we knew (could observe) ...

- the numbers $N(s_i), N(s_i^{(t)})$ of occurrences of each hidden state s_i in $\mathcal{O}$
- the numbers $N(s_i \rightarrow s_j)$ of state transitions in $\mathcal{O}$
- the numbers $N(s_j \cap o_i)$ of co-occurrences between hidden states and observations

we could estimate the parameters $\boldsymbol{\theta} = \{\boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{B}\}$ via relative frequencies:

- use $\frac{N(s_i^{(0)})}{|\mathcal{O}|}$ to estimate $\boldsymbol{\Pi} = \{P(s_i^{(0)})\}$
- use $\frac{N(s_i \rightarrow s_j)}{N(s_i)}$ to estimate $\boldsymbol{A} = \{P(s_j|s_i)\}$
- use $\frac{N(s_j \cap o_i)}{N(s_j)}$ to estimate $\boldsymbol{B} = \{P(o_i|s_j) = b_j(o_i)\}$

Problem:

- Do not have this information, because true states are not observable
- Only have access to sequences of observation symbols.

Some Thoughts: On the Other Hand ...

If we knew ...

- the parameters $\boldsymbol{\theta} = \{\boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{B}\}$ (i.e., the true HMM)

we could estimate the expected values of these counts $N(s_i^{(t)})$ etc:

- Align training observation sequences to model via smoothing algorithm
- Estimate number of state occurrences, state transitions etc. from the obtained sequences of state distributions (details see below).

Problem:

- Do not know the true parameters $\boldsymbol{\theta}$ (i.e., the true HMM)
  – that's what we want to learn!

Basic Idea for an Iterative Procedure

Summary of the Dilemma:

- If we knew the hidden counts $N(\cdot)$, we could estimate the model $\boldsymbol{\theta}$
- If we knew the model $\boldsymbol{\theta}$, we could estimate the hidden counts $N(\cdot)$

Approach to Solution:

- Break the vicious cycle:
- Assume we **do** have an initial model $\boldsymbol{\theta}^0 = \{\boldsymbol{\Pi}^0, \boldsymbol{A}^0, \boldsymbol{B}^0\}$ (possibly a very poor one)

  (in fact, we can start with a *random model*)
- Design an iterative procedure to improve the model step by step by **fitting the data to the model** ($\rightarrow$ estimating the corresponding hidden counts $N(\cdot)$) and **re-estimating the parameters** $\boldsymbol{\theta}$ ($\rightarrow$ learning an improved model).

The General Re-Estimation Procedure

Generic Re-Estimation Algorithm

Initialisation:

- Simplest approach: Initialise parameters $\boldsymbol{\theta}^0$ with *random distributions*

Iterative Re-Estimation: Repeat until convergence:

1. **"Expectation Step":** Compute the expected values

$$\boldsymbol{E} = \{E(N(s_i)), E(N(s_i \rightarrow s_j)), E(N(s_j \cap o_i))\}$$

   based on training observations $\mathcal{O}$ and current model $\boldsymbol{\theta}^i$

2. **"Maximisation Step:"** Given these expected values, re-estimate parameters $\boldsymbol{\theta}^{i+1}$ such that they maximise the likelihood:

$$\boldsymbol{\theta}^{i+1} = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta} : \boldsymbol{E})$$

☞ Baum-Welch Algorithm

Computing the Expected Values of the Hidden Counts

Define

$$\tau^{(t)}(i,j) \quad = \quad P(s_i^{(t)}, s_j^{(t+1)} \mid \boldsymbol{O}, \boldsymbol{\theta})$$

$$\tau^{(t)}(i) \quad = \quad P(s_i^{(t)} \mid \boldsymbol{O}, \boldsymbol{\theta}) = \sum_{j=1}^{N} \tau^{(t)}(i,j)$$

In Words:

- $\tau^{(t)}(i,j)$ is the probability that, in producing observation sequence $\boldsymbol{O}$, the model went through state s_i at time t, followed by state s_j at $t+1$
- $\tau^{(t)}(i)$ is the probability that, in producing $\boldsymbol{O}$, the model was in state s_i at time t
- **Note:** $\tau^{(t)}(i)$ is what we compute in forward-backward smoothing.

Computing the Expected Values of the Hidden Counts

How to compute $\tau^{(t)}(i,j)$ (see next slide for a formal proof):

$$\boxed{\tau^{(t)}(i,j)} = P(s_i^{(t)}, s_j^{(t+1)} \mid \boldsymbol{O}, \boldsymbol{\theta}) = \frac{\mathsf{lf}_i^{(1:t)} a_{ij} b_j(o^{(t+1)}) \mathsf{b}_j^{(t+2:T)}}{P(\boldsymbol{O}|\boldsymbol{\theta})}$$

$$= \boxed{\frac{1}{Z} \mathsf{lf}_i^{(1:t)} a_{ij} b_j(o^{(t+1)}) \mathsf{b}_j^{(t+2:T)}}$$

where

- $\mathsf{lf}^{(1:t)}$ is the **'likelihood forward message'** as computed in Task 5
- $\mathsf{b}^{(t+2:T)}$ is the **backward message** as computed in the smoothing algorithm.

Note: Expression $\mathsf{lf}_i^{(1:t)} a_{ij} b_j(o^{(t+1)}) \mathsf{b}_j^{(t+2:T)}$ (without normalisation) is the probability of the model producing the observation sequence $\boldsymbol{O}$ *and* passing through states $s_i^{(t)}, s_j^{(t+1)}$.

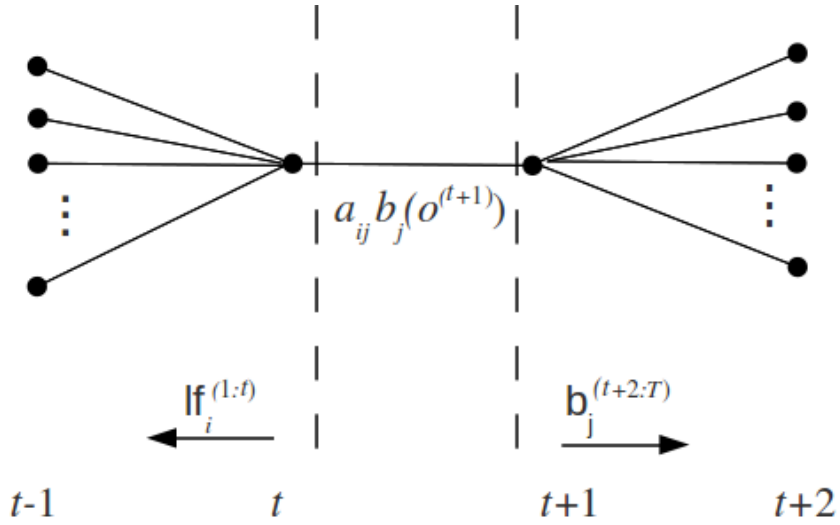
The Proof

$$
\begin{aligned}
\boxed{P(s_i^{(t)}, s_j^{(t+1)}|\boldsymbol{o}^{(1:T)})} &= \frac{1}{Z} P(s_i^{(t)}, s_j^{(t+1)}, \boldsymbol{o}^{(1:t+1)}, \boldsymbol{o}^{(t+2:T)}) \\
&= \frac{1}{Z} P(s_i^{(t)}, s_j^{(t+1)}, \boldsymbol{o}^{(1:t+1)}) \underline{P(\boldsymbol{o}^{(t+2:T)}|s_i^{(t)}, s_j^{(t+1)}, \boldsymbol{o}^{(1:t+1)})} \\
&= \frac{1}{Z} P(s_i^{(t)}, \boldsymbol{o}^{(1:t)}, s_j^{(t+1)}, \boldsymbol{o}^{(t+1)}) P(\boldsymbol{o}^{(t+2:T)}|s_j^{(t+1)}) \\
&= \frac{1}{Z} P(s_i^{(t)}, \boldsymbol{o}^{(1:t)}) \underline{P(s_j^{(t+1)}, \boldsymbol{o}^{(t+1)}|s_i^{(t)}, \boldsymbol{o}^{(1:t)})} \times \mathsf{b}_j^{(t+2:T)} \\
&= \frac{1}{Z} \mathsf{lf}_i^{(1:t)} \times P(s_j^{(t+1)}, \boldsymbol{o}^{(t+1)}|s_i^{(t)}) \times \mathsf{b}_j^{(t+2:T)} \\
&= \frac{1}{Z} \mathsf{lf}_i^{(1:t)} \times P(s_j^{(t+1)}|s_i^{(t)}) \underline{P(\boldsymbol{o}^{(t+1)}|s_j^{(t+1)}, s_i^{(t)})} \times \mathsf{b}_j^{(t+2:T)} \\
&= \frac{1}{Z} \mathsf{lf}_i^{(1:t)} \times a_{ij} \times P(\boldsymbol{o}^{(t+1)}|s_j^{(t+1)}) \times \mathsf{b}_j^{(t+2:T)} \\
&= \boxed{\frac{1}{Z} \mathsf{lf}_i^{(1:t)} \times a_{ij} \times b_j(o^{(t+1)}) \times \mathsf{b}_j^{(t+2:T)}}
\end{aligned}
$$

where ..... indicates reducibility due to conditional independence.

The Intuition

$$\tau^{(t)}(i,j) = P(s_i^{(t)}, s_j^{(t+1)} \mid \boldsymbol{O}, \boldsymbol{\theta}) \quad = \quad \boxed{\frac{1}{Z} \mathsf{lf}_i^{(1:t)} a_{ij} b_j(o^{(t+1)}) \mathsf{b}_j^{(t+2:T)}}$$



- $\mathsf{lf}_i^{(1:t)}$ is the probability of seeing $o^{(1:t)}$ and being in state s_i at time t
- $\mathsf{b}_j^{(t+2:T)}$ is the probability of seeing $o^{(t+2:T)}$, given that state is s_j at $t+1$
- Probability that state is s_i at t and s_j at $t+1$ is proportional to $\mathsf{lf}_i^{(1:t)}$ times probability of making transition $s_i \rightarrow s_j$, observing $o^{(t+1)}$ in s_j, then seeing the rest ($\mathsf{b}_j^{(t+2:T)}$).

Computing the Expected Values of the Hidden Counts

$$\tau^{(t)}(i,j) = P(s_i^{(t)}, s_j^{(t+1)} \mid \boldsymbol{O}); \qquad \tau^{(t)}(i) = P(s_i^{(t)} \mid \boldsymbol{O})$$

Summing over time t and all $O \in \mathcal{O}$ gives expected number of times a state s_i is visited, or a transition $s_i \to s_j$ is made:[a]

- Expected number of transitions from s_i to s_j:

$$E(N(s_i \to s_j)) = \sum_{t=0}^{T-1} \tau^{(t)}(i,j)$$

- Expected number of occurrences of s_i:

$$E(N(s_i)) = \sum_{t=0}^{T} \tau^{(t)}(i)$$

- In a similar way (*exercise* :-), can compute $E(N(s_j \cap o_i))$, the expected number of co-occurrences of state s_j and observation o_i.

[a] $\tau^{(t)}(i)$ can be interpreted as the probability of an *indicator variable* $\mathbb{1}_i^{(t)}$ being true (1); $\sum_t P(\mathbb{1}_i^{(t)})$ then gives the expected number of $\mathbb{1}_i$'s that are 1 for state s_i.

The Baum-Welch Algorithm

Baum-Welch Algorithm for Learning HMMs

Initialisation:

- Initialise model $\boldsymbol{\theta}^0 = \{\boldsymbol{\Pi}^0, \boldsymbol{A}^0, \boldsymbol{B}^0\}$ with *random distributions*

Iterative Re-Estimation: **Repeat until convergence:**

1. "Expectation":

- Run a modified forward-backward algorithm to compute the $\tau^{(t)}(i,j)$
- Compute $E(N(s_i)), E(N(s_i \rightarrow s_j)), E(N(s_j \cap o_i))$ as indicated above

2. "Maximisation": Re-estimate parameters $\boldsymbol{\theta}' = \{\boldsymbol{\Pi}', \boldsymbol{A}', \boldsymbol{B}'\}$ as:

$$\pi'_i = \text{probability of state } s_i \text{ at time } 0 = \tau^{(0)}(i)$$

$$a'_{i,j} = \frac{\text{expected number of transitions from } s_i \text{ to } s_j}{\text{expected number of times in state } s_i} = \frac{E(N(s_i \rightarrow s_j))}{E(N(s_i))}$$

$$b'_j(o_i) = \frac{\text{expected number of times in state } s_j \text{ and observing } o_i}{\text{expected number of times in state } s_j} = \frac{E(N(s_j \cap o_i))}{E(N(s_j))}$$

The Baum-Welch Algorithm

Baum-Welch is an instance of a very general algorithm schema for estimating hidden model parameters by iterating between

- Aligning the training data to the current model, and in this way estimating the expected values of some hidden variables
- Given this alignment and the expected values, re-fitting the parameters of the model to give maximum likelihood, relative to these expected values

☞ **E-M ("Expectation-Maximisation") Algorithm**

Can be *proven* to improve the likelihood in every single step!
☞ **Ends up in a local maximum of the likelihood function**

Why E-M Works: The Intuition

Abstract View of the Baum-Welch Algorithm

Initialisation:

- Start with a very poor model $\boldsymbol{\theta}^0$ (very far from the observations)

Iterative Re-Estimation: Repeat until convergence:

1. **"Expectation Step":** Compute the expected values

$$\boldsymbol{E} = \{E(N(s_i)), E(N(s_i \rightarrow s_j)), E(N(s_j \cap o_i))\}$$

   $\boldsymbol{E}$ = *'Summary' of* ***how current model*** $\boldsymbol{\theta}^i$ ***would most likely have generated*** $\mathcal{O}$

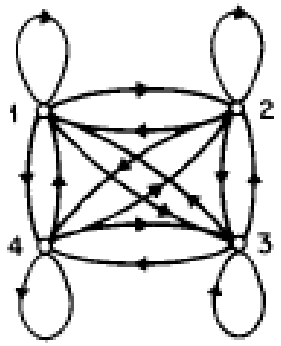
2. **"Maximisation Step:"** Given these expected values, re-estimate parameters $\boldsymbol{\theta}^{i+1}$ such that they maximise the likelihood $P(\mathcal{O} \mid \boldsymbol{E}, \boldsymbol{\theta})$

   $\boldsymbol{\theta}^{i+1}$ = ***most likely model*** *that would have generated* $\mathcal{O}$ ***in this way***

☞ **New model $\boldsymbol{\theta}^{i+1}$ is at least as likely as previous model $\boldsymbol{\theta}^i$ ...**

Advanced Issues (1): Constrained Transition Model Structures

The process being modelled may imply a specific kind of structure for the transition model:

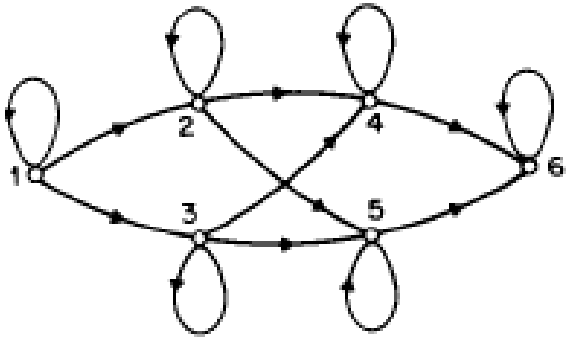


(a)

fully connected ('ergodic') model

(b)

left-to-right model permitting skips
(e.g., for tracking live music)

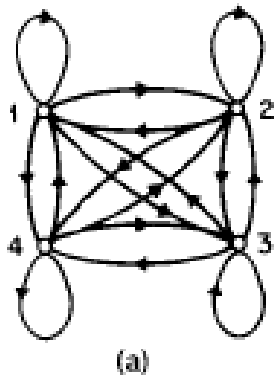


(c)

parallel-path left-to-right model
(e.g., for speech recognition)

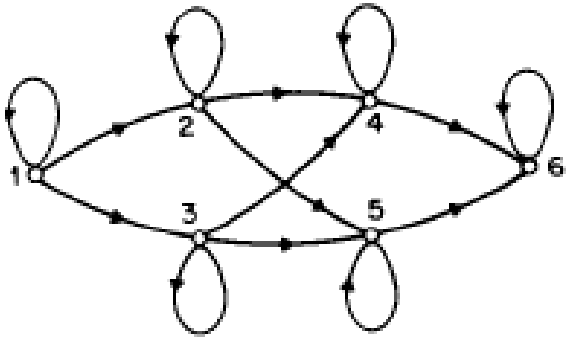
Advanced Issues (1): Constrained Transition Model Structures

Realisation: structured, sparse State Transition Matrices



(a)

(b)



(c)

$$A = \begin{bmatrix} x & x & x & x & x & x & x & \ldots\ldots \\ x & x & x & x & x & x & x & \ldots\ldots \\ x & x & x & x & x & x & x & \ldots\ldots \\ x & x & x & x & x & x & x & \ldots\ldots \\ .. & .. & .. & .. & .. & .. & .. & \ldots\ldots \\ .. & .. & .. & .. & .. & .. & .. & \ldots\ldots \end{bmatrix}$$

$$A = \begin{bmatrix} x & x & x & 0 & 0 & 0 & 0 & \ldots\ldots \\ 0 & x & x & x & 0 & 0 & 0 & \ldots\ldots \\ 0 & 0 & x & x & x & 0 & 0 & \ldots\ldots \\ 0 & 0 & 0 & x & x & x & 0 & \ldots\ldots \\ .. & .. & .. & .. & .. & .. & .. & \ldots\ldots \\ .. & .. & .. & .. & .. & .. & .. & \ldots\ldots \end{bmatrix}$$

Figure out for yourself what (c) looks like ...

Advanced Issues (2): Continuous Observations

Problem: What if observations are not discrete, but continuous?

- How to represent observation model $P(O \mid S)$?
- And: what if we have *more than one* (continuous) observation variable? (e.g., when observations are represented as vectors of feature values)

Solution 1: Discretisation / "Vector Quantisation"

- Discretise continuous observations (e.g., split $Val(O)$ into discrete intervals and use these as values; or: cluster the data and replace each observation with one of the k cluster centers)
- In the case of *several* (discretised) observation variables $O_1, ..., O_N$: Create new *'super-variable'* $O = O_1 \times \cdots \times O_N$
- Set of observations is discrete and finite again
- Can use standard table CPD

Problems:

- Information loss due to discretisation
- Possibly huge number of values $Val(O)$ (especially if O is multi-dimensional)
- Impractical to represent $P(O|S)$ as a table CPD.

Advanced Issues (2): Continuous Observations

Problem: What if observations are not discrete, but continuous?

- How to represent observation model $P(O \mid S)$?
- And: what if we have *more than one* (continuous) observation variable? (e.g., when observations are represented as vectors of feature values)

Solution 2: Continuous Probability Density Models

- Model $P(O|S)$ directly with a continuous probability density function p
- Place some restriction on the form of p so that the parameters of p can be effectively estimated in the learning step
- Common choice: model p as a normal (Gaussian) distribution or a mixture of Gaussians (GMM).

Problem:

- Observation distributions $p(O|S)$ may not be well describable by chosen family of distributions (e.g., true data distribution may be very different from a Gaussian distribution)
- Large error introduced by approximation of $p(O|S)$ by a Gaussian.

Gaussian Observation Models: The Univariate Case

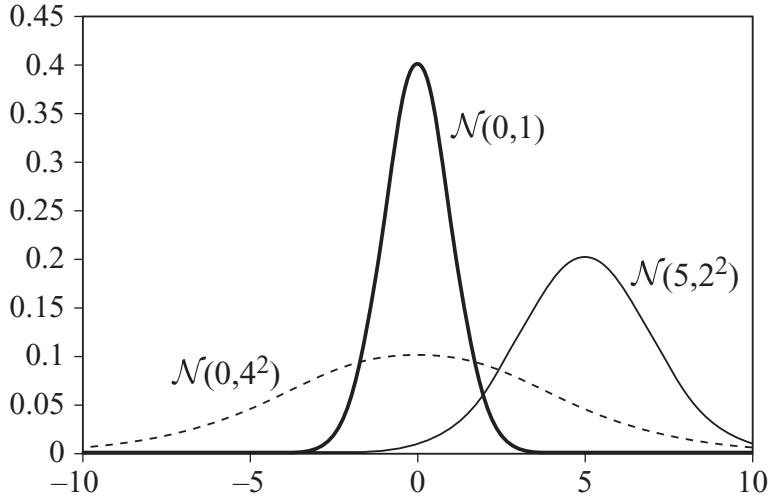
Given:

- Discrete state variable S, single continuous observation variable O.

Single-Gaussian Observation Model:

- Model $P(O \mid S)$ by a **separate normal distribution for each possible state:** $p(O \mid S = s_i) \sim \mathcal{N}(\mu_i, \sigma_i^2)$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



- Instead of a CPD table, $P(O \mid S)$ is represented by n pairs of numbers (μ_i, σ_i^2) (where n = number of possible states s_i)

Gaussian Observation Models: The Multivariate Case

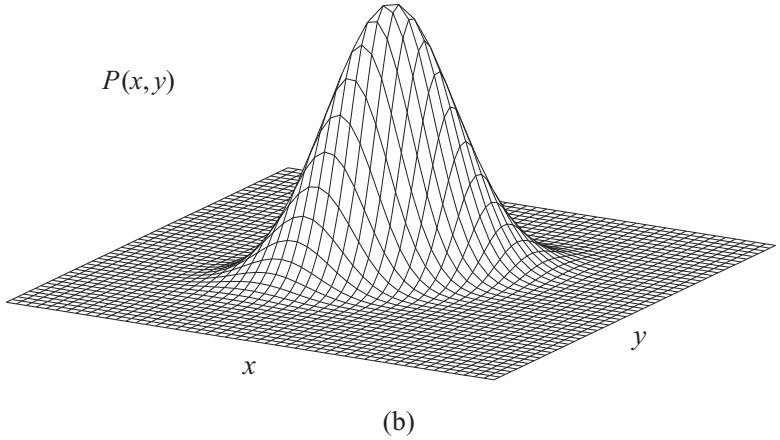
Given:

- Discrete state variable S, N continuous observation variables $O_1, ..., O_N$

Solution 2.a – Single-Gaussian Multivariate Observation Model:

- Model joint conditional distribution $P(\boldsymbol{O} \mid S)$ by a **multivariate Gaussian distribution** $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, one for each value $s_i \in S$

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{N/2}|\boldsymbol{\Sigma}|^{1/2}} e^{\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]}$$



(b)

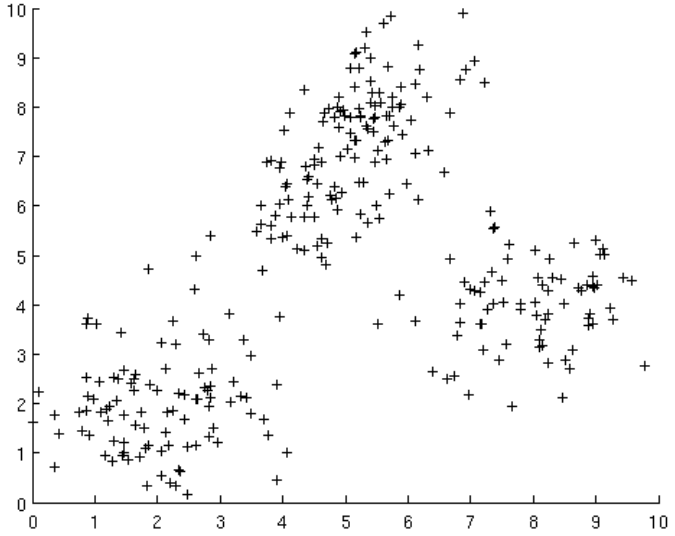
where

- $\boldsymbol{\mu}_i$ is an N-dimensional **mean vector** (= vector of means of $O_1, ...O_N$)
- $\boldsymbol{\Sigma}_i$ is an $N \times N$ **covariance matrix**, where σ_{kl} = correlation between variables O_k and O_l

Gaussian Observation Models: The Multivariate Case

Problem:

- Highly unlikely (especially in high-dimensional spaces) that joint distribution of N variables will be well described by a single Gaussian



☞ **Solution 2.b – Gaussian Mixture Model:**

- Model joint conditional distribution $P(\boldsymbol{O}|S)$ by a **weighted combination of single Gaussians**

Digression: Gaussian Mixture Models (GMMs)

Definition

A **GAUSSIAN MIXTURE MODEL (GMM)** is a joint probability density function over a set of N continuous random variables $\boldsymbol{X} = \{X_1, \ldots, X_N\}$, defined by a weighted sum of k Gaussian distributions ("components"):

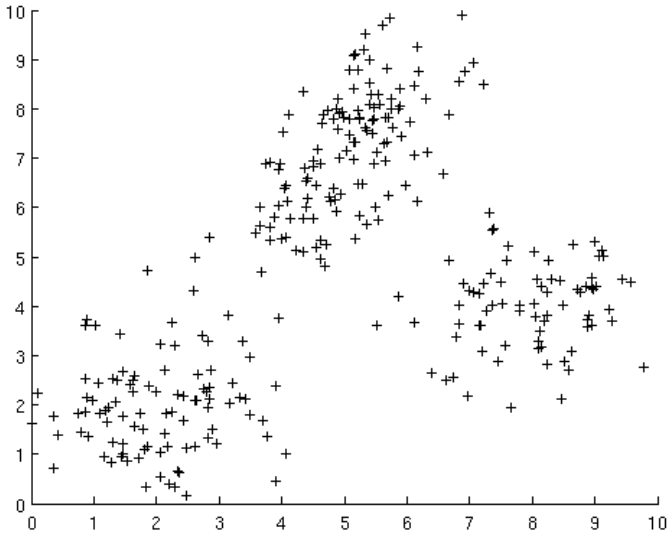
$$p(\boldsymbol{x}) = \sum_{i=1}^{k} w_i \times p_i(\boldsymbol{x})$$

where

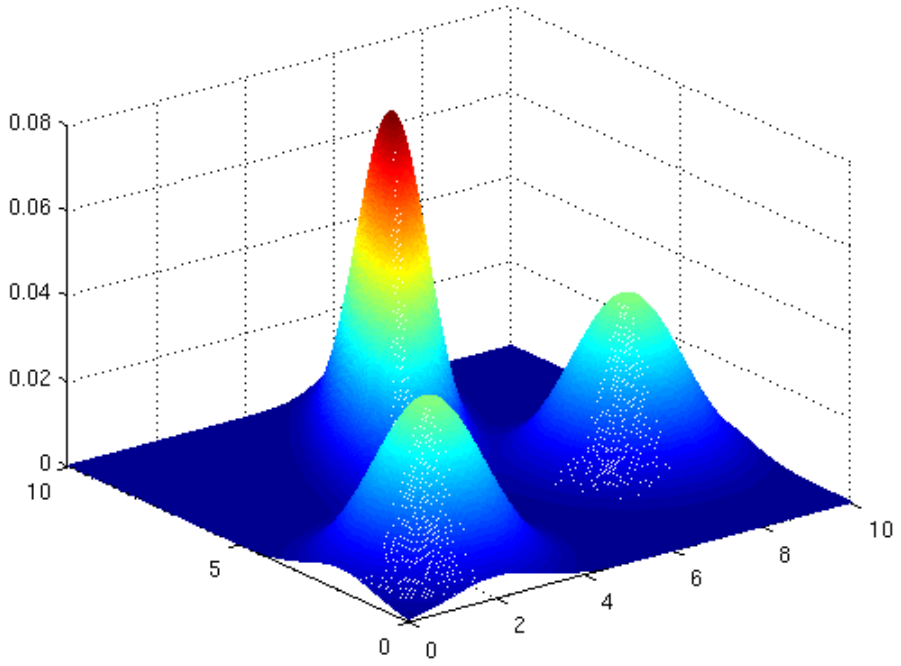
- w_i is the **weight ('mixture coefficient')** of the i^{th} component
- $p_i(\boldsymbol{x})$ is the density at $\boldsymbol{x}$ under the i^{th} component, where p_i is a multivariate Gaussian density function defined by its own mean vector $\boldsymbol{\mu}_i$ and covariance matrix $\boldsymbol{\Sigma}_i$.

An Example GMM in 2 Dimensions

300 sample points in a 2-D variable space:



A GMM describing the distribution of the points:



A Generative View of GMMs

$$p(\boldsymbol{x}) = \sum_{i=1}^{k} w_i \times p_i(\boldsymbol{x})$$

Consider a GMM as a probabilistic generator of observations:

- GMM is a weighted combination of simple Gaussian data generators $p_i(\cdot)$
- Each p_i would generate a random point according to $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$
- Probability that component p_i gets to generate next point $\boldsymbol{x}$ is proportional to its weight w_i
- View weight w_i as *probability* $P(C = i)$ that $\boldsymbol{x}$ is generated by component i
- Process of generating an observation:
  1. Probabilistically choose generating component $C = p_i$ according to $w_i = P(C = i)$
  2. Probabilistically generate a point $\boldsymbol{x}$ according to $p_i(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$

$$p(\boldsymbol{x}) = \sum_{i=1}^{k} P(C = i) p(\boldsymbol{x} \mid C = i)$$

Fitting a GMM onto Data

The Learning / Optimisation Problem

Given:

- Training set: a set $\mathcal{D}$ of example points $x_i \in Val(X_1) \times \ldots \times Val(X_N)$
- Model complexity parameter: a pre-defined number k of components

Find:

- GMM $\mathcal{M}$ that best 'explains' or 'fits' the observed data $\mathcal{D}$:
- Values for the parameters $w_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$ (for $i = 1 \ldots k$) that maximise the **likelihood** of the resulting GMM $\mathcal{M}$:

$$\begin{aligned} L(\mathcal{M} : \mathcal{D}) &= p(\mathcal{D} \mid \mathcal{M}) \\ &=^{a} \prod_i p(\boldsymbol{x}_i \mid \mathcal{M}) \end{aligned}$$

[a] Assumption: the example points are *i.i.d.* (*independent and identically distributed*)

Fitting a GMM onto Data: The E-M Algorithm

The E-M Algorithm for GMMs

Initialisation:

'Guess' initial model parameters $w_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$ (for $i = 1 \ldots k$).
Simplest approach: initialise all parameters with *random values*.

Iterate until convergence:

1. **E ("Expectation") Step:**
   - Compute probabilistic assignment of samples $\boldsymbol{x}_j$ to components i = probability that $\boldsymbol{x}_j$ was generated by component i

$$q_{ij} = P(C = i|\boldsymbol{x}_j) = \frac{p(\boldsymbol{x}_j|C = i)P(C = i)}{\sum_k p(\boldsymbol{x}_j|C = k)P(C = k)}$$

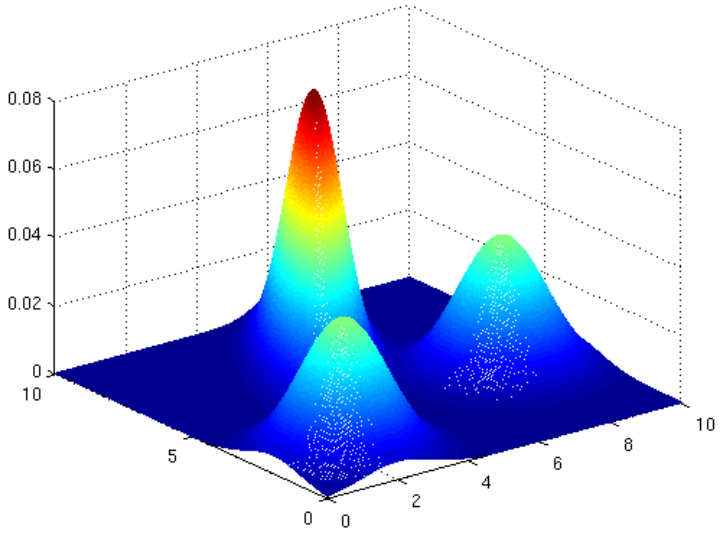
   - Compute total 'weight'of component i: $q_i = \sum_j q_{ij}$
2. **M ("Maximisation") Step:**
   - Compute new weighted means, covariances, and component weights:

$$\begin{aligned}\boldsymbol{\mu}_i &= \textstyle\sum_j q_{ij}\boldsymbol{x}_j/q_i \\ \boldsymbol{\Sigma}_i &= \textstyle\sum_j q_{ij}(\boldsymbol{x}_j - \boldsymbol{\mu}_i)(\boldsymbol{x}_j - \boldsymbol{\mu}_i)^T/q_i \\ w_i &= q_i\end{aligned}$$

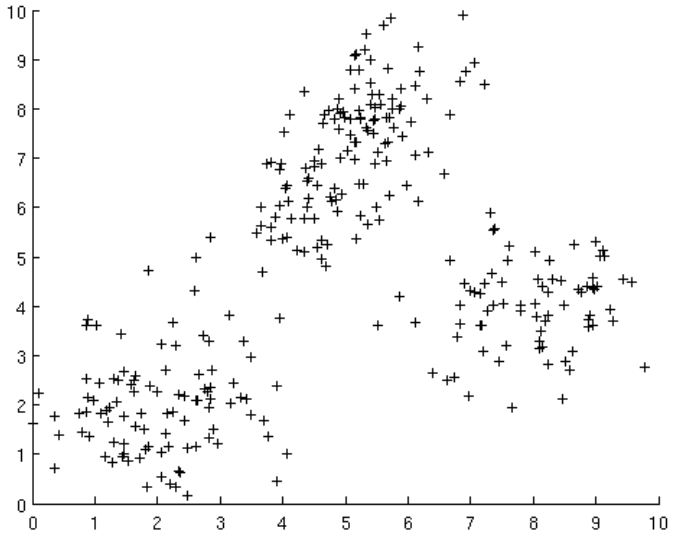
Learning GMMs: A Little Experiment

Setup:

- Define a hidden 'true' distribution (in the form of a 3-GMM)
- Randomly sample 300 training points from this distribution
- Run E-M to fit various GMMs on these points
- (initial covariances: circular, with fixed variance along each dimension)



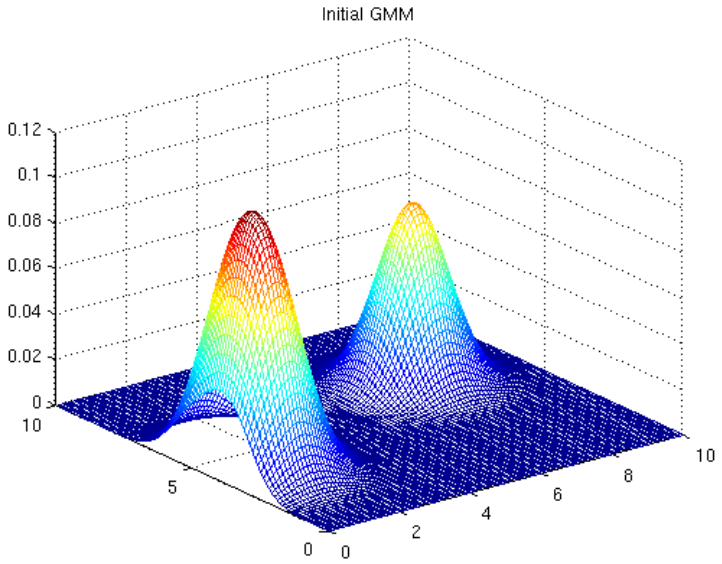
Hidden generating distribution (GMM)



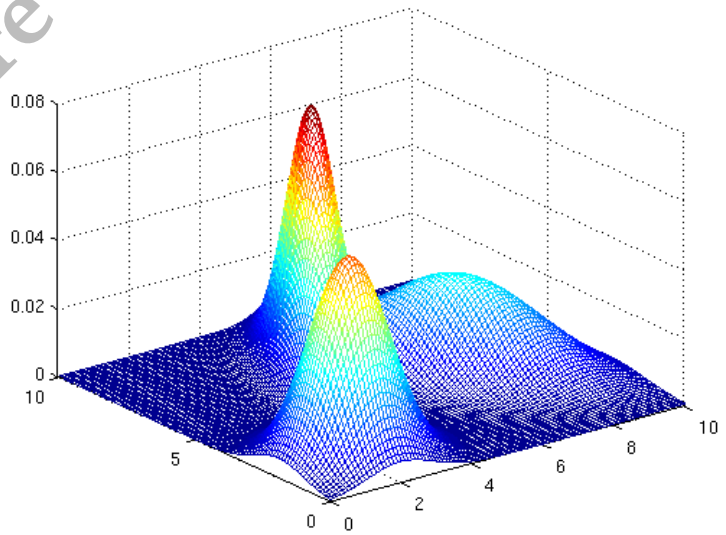
300 points randomly sampled

Learning GMMs: A Little Experiment

Fitting a GMM with $k = 3$:



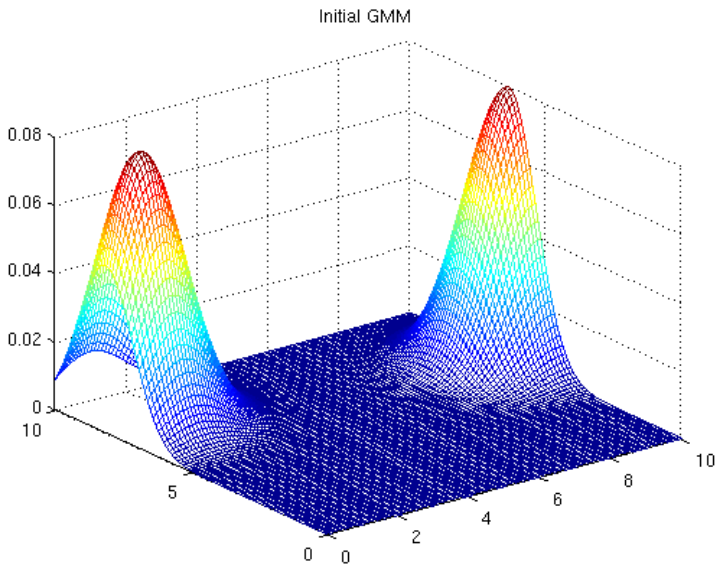
Initial GMM (with random $\boldsymbol{\mu}_i$)



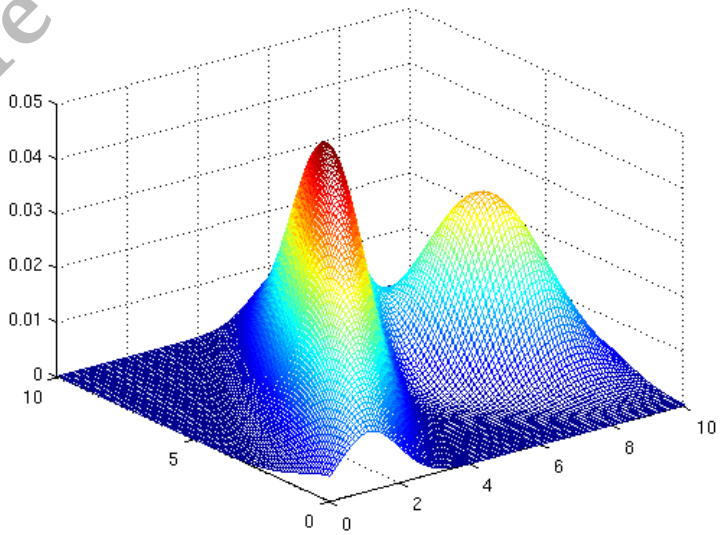
GMM after 10 iterations

Learning GMMs: A Little Experiment

Fitting a GMM with $k = 2$:



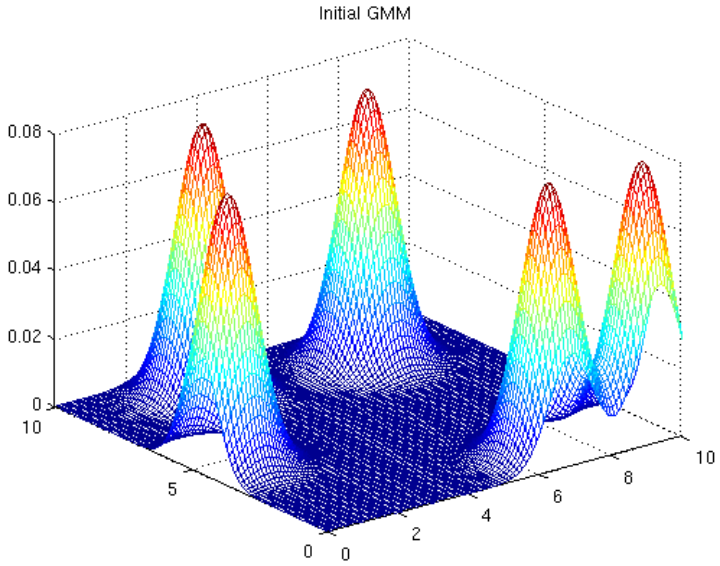
Initial GMM (with random $\boldsymbol{\mu}_i$)



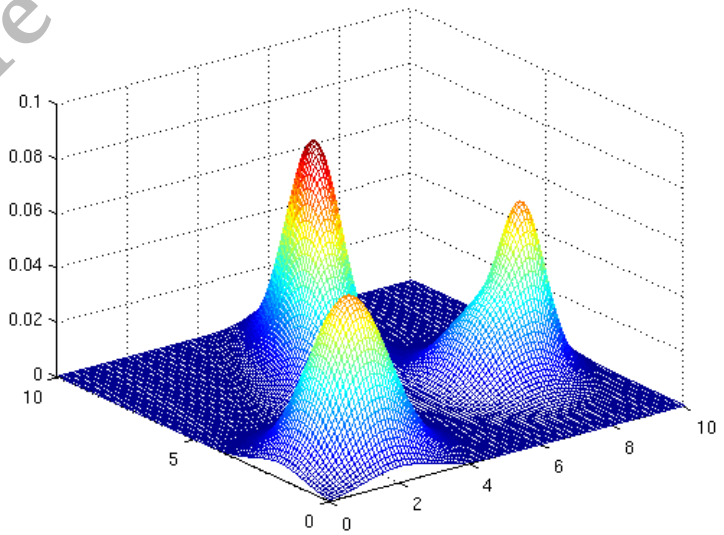
GMM after 10 iterations

Learning GMMs: A Little Experiment

Fitting a GMM with $k = 5$:



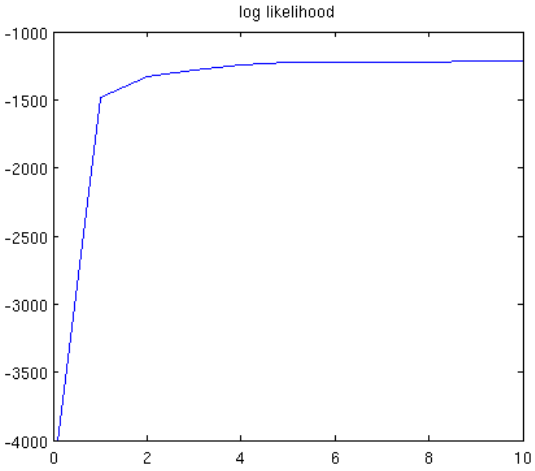
Initial GMM (with random $\boldsymbol{\mu}_i$)



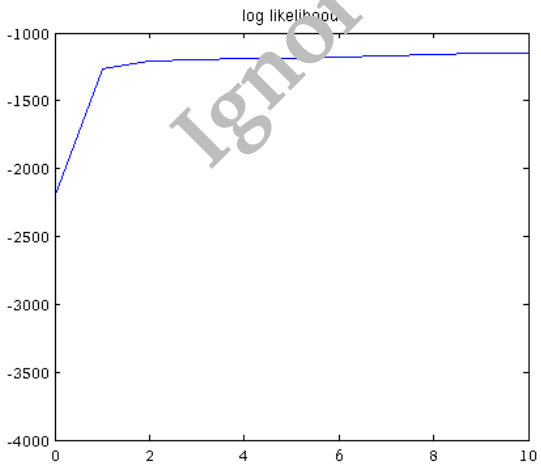
GMM after 10 iterations

Learning GMMs: A Little Experiment

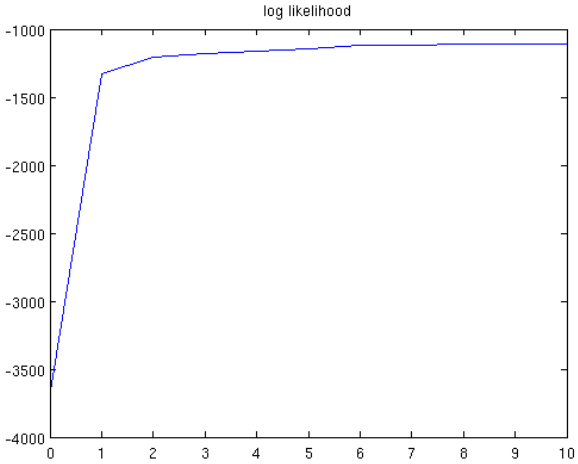
Development of the log-likelihood:



$k = 3$



$k = 2$



$k = 5$

Properties of the E-M Algorithm

Interpretation of the E-M Steps:

- **E (Expectation) Step:** Can be interpreted as computing the expected value of the hidden indicator variables Z_{ij}, where

$$Z_{ij} = \begin{cases} 1 & \text{if } \boldsymbol{x}_i \text{ was generated by component } i, \\ 0 & \text{otherwise.} \end{cases}$$

- **M (Maximisation) Step:** Finds new parameter values that maximise the likelihood of the model, given the expected values of the Z_{ij}.

Central Property of EM:

- EM (provably) increases the likelihood of the model at every iteration; converges to a local maximum.

The General Form of the E-M Algorithm

Here is the E-M algorithm in its most general form[3]
(covers E-M for clustering, for fitting GMMs, for learning HMMs, learning the parameters of general Bayes networks with hidden variables, etc. etc.)

The General E-M Algorithm

$$\boldsymbol{\theta}^{(i+1)} = \arg\max_{\boldsymbol{\theta}} \sum_{\boldsymbol{z}} L(\boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z} \mid \boldsymbol{\theta}^{(i)}) P(\boldsymbol{Z} = \boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(i)})$$

where

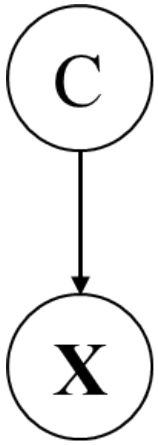
- $\boldsymbol{x}$ is all the observed values in all the training examples
- $\boldsymbol{Z}$ is all the hidden variables in all the examples
- $\boldsymbol{\theta}$ is the parameters of the model we wish to learn.

In one sentence: E-M searches for the parameters $\boldsymbol{\theta}^{(i+1)}$ that maximise the expected probability of the observed data $\boldsymbol{x}$ (the *likelihood* L), given the current model parameters $\boldsymbol{\theta}^{(i)}$, where the expectation is computed with respect to the distribution over the hidden variables $P(\boldsymbol{Z} = \boldsymbol{z} | \boldsymbol{x}, \boldsymbol{\theta}^{(i)})$ ...

[3] Don't worry. You will not need to know or understand this for the exam.

The GMM as a Graphical Model

Did you notice that a GMM corresponds to a simple Graphical Model?



... represents the joint distribution

$$p(C, \boldsymbol{X}) = P(C)p(\boldsymbol{X}|C)$$

Thus, $p(\boldsymbol{x})$ is (by standard inference by enumeration):

$$p(\boldsymbol{x}) = \sum_i P(C = c^i, \boldsymbol{x})$$
$$= \sum_i P(C = c^i)p(\boldsymbol{x} \mid C = c^i)$$

☞ exactly the distribution represented by the GMM!

Consequences:

- The E-M algorithm for GMMs is really just a special case of the E-M algorithm for learning HMMs (Baum-Welch)
- No temporal structure $\Rightarrow$ no alignment via smoothing needed.

Real-World Applications of HMMs

1. **HMMs for Classification: SPEECH RECOGNITION**
2. **HMMs for Real-time Tracking: MUSICAL BEAT TRACKING**

Application Example 1: Speech Recognition

The following slides relate to Chapter 15, Section 6 of [Russell & Norvig, 2003] and have been adapted from materials provided by Stuart Russell (including figures). Many thanks to Stuart for making slides sources and figures available.

Speech Understanding as Probabilistic Inference

"It's not easy to wreck a nice beach."

$\Rightarrow$ **Speech signals are noisy, variable, ambiguous.**

Speech Understanding as a Probabilistic Inference Task:

- What is the most likely word sequence, given a speech signal?
- Choose $words$ to maximize $P(Words \mid signal)$

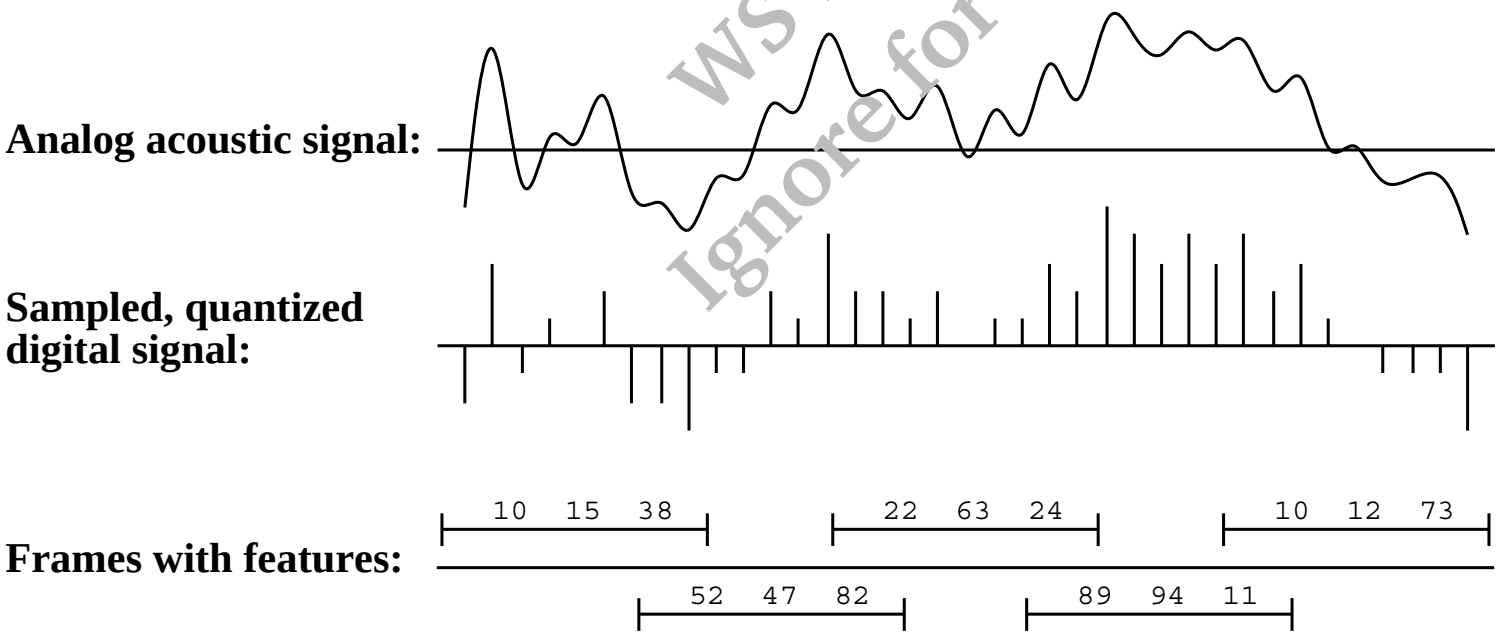
$$P(Words \mid signal) = \frac{1}{Z} P(signal \mid Words) P(Words)$$

Think of this as a state-observation model:

- $Words$ are the hidden state sequence
- $signal$ is the observation sequence
- Observation Model = **Acoustic Model** (model of pronunciation)
- Transition Model = **Language Model** (model of likely phoneme and word sequences)

Acoustic Model: Speech Sound Features

Raw signal is the microphone displacement as a function of time.
Processed into short overlapping **frames** (e.g., 30 ms), each described by **features**



Typical features: Formants (peaks in the power spectrum) or MFCCs (compact encoding of the spectral envelope)

Phones

All human speech is composed from 40-50 **phones**, determined by the configuration of **articulators** (lips, teeth, tongue, vocal cords, air flow).

Example: ARPAbet designed for American English

[iy]	b**ea**t	[b]	**b**et	[p]	**p**et
[ih]	b**i**t	[ch]	**Ch**et	[r]	**r**at
[ey]	b**e**t	[d]	**d**ebt	[s]	**s**et
[ao]	b**ough**t	[hh]	**h**at	[th]	**th**ick
[ow]	b**oa**t	[hv]	**h**igh	[dh]	**th**at
[er]	B**er**t	[l]	**l**et	[w]	**w**et
[ix]	ros**e**s	[ng]	si**ng**	[en]	butt**on**
⋮	⋮	⋮	⋮	⋮	⋮

E.g., “ceiling” is [s iy l ih ng] / [s iy l ix ng] / [s iy l en]

A 'Simple' Problem: Recognition of Single Words

Task:

- Given: Audio signal of a single spoken word (e.g., voice command)
- Decide: Which of a set of possible words was spoken

Modelling and Learning:

- Build a HMM $\mathcal{M}_{w_i}$ for each word w_i to be recognised
- Hidden states: Phones ($\sim$ "sub-syllables") making up the word
- Transition Model: Ways of speaking the word (as a sequence of phones)
- Observation Model: relates phones to audio features
- Train models on large number of example recordings (different speakers)

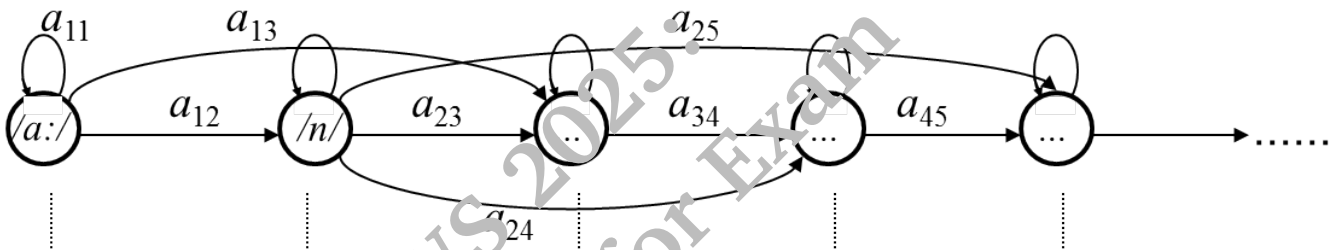
Classification via Maximum Likelihood:

- Compute likelihood of signal under each word model $\mathcal{M}_{w_i}$ (using the Likelihood Forward Algorithm)
- Output word with maximum likelihood:

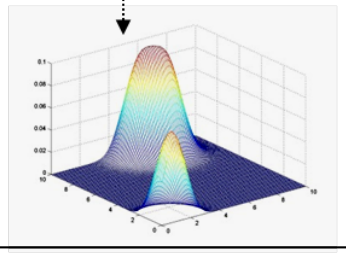
$$\hat{w} = \arg\max_{w_i} L(\mathcal{M}_{w_i} : signal) = \arg\max_{w_i} P(signal \mid \mathcal{M}_{w_i})$$

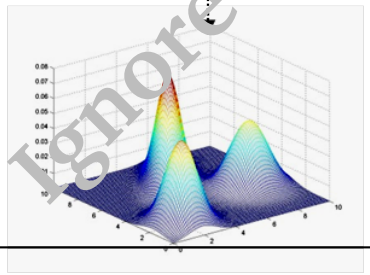
A 'Simple' Problem: Recognition of Single Words

Hidden layer:
Word model:
(states ≈ phones)



Observation model:
GMM

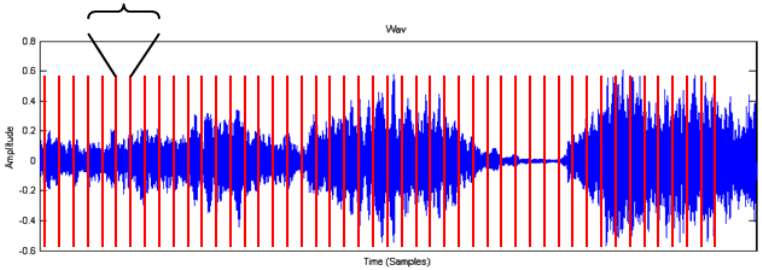




.....

Feature extraction:
FFT, MFCCs, etc.

Observations:
Spoken words
(cut into audio frames
of,e.g., 20 ms)



A Multi-level Approach to Speech Modelling

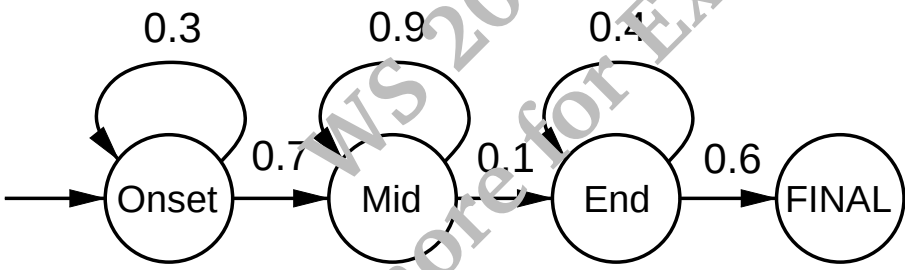
Real-world speech recognition systems are more complex.

Several levels of hidden state variables:

- **Language Model:** Distribution over sequences of words
- **Word Pronunciation Model:** Distribution over phone sequences within a word
- **Phone Model:** Sequences of sound production states within a phone
- **Acoustic Model:** Relates phone states to audio features from speech signal

Phone Model

Phone HMM for [m]:



Output probabilities for the phone HMM:

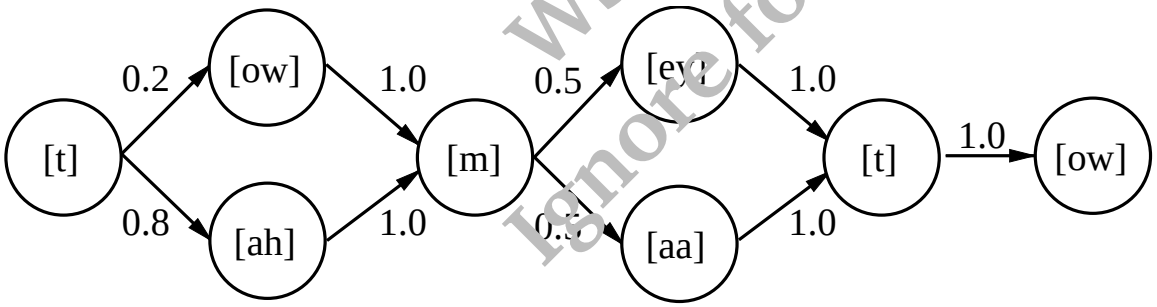
Onset:	Mid:	End:
C1: 0.5	C3: 0.2	C4: 0.1
C2: 0.2	C4: 0.7	C6: 0.5
C3: 0.3	C5: 0.1	C7: 0.4

☞ **Relates phone sequences (hidden) to audio features (observable)**

Word Pronunciation Model

Markov Model for Each Word:

- Each word described as a distribution over phone sequences
- Distribution represented as an HMM transition model



$$P([towmeytow] \mid \text{“tomato”}) = P([towmaatow] \mid \text{“tomato”}) = 0.1$$
$$P([tahmeytow] \mid \text{“tomato”}) = P([tahmaatow] \mid \text{“tomato”}) = 0.4$$

☞ **Relates phone sequences (hidden) to audio features (observable)**

Structure is created manually, transition probabilities learned from data

Language Model

Not just a sequence of isolated-word recognition problems!

- Adjacent words highly correlated
- Sequence of most likely words $\neq$ most likely sequence of words
- Segmentation: there are few gaps in speech
- Cross-word coarticulation—e.g., "next thing"

Need to model sequences of words.

Prior probability of a word sequence is given by chain rule:

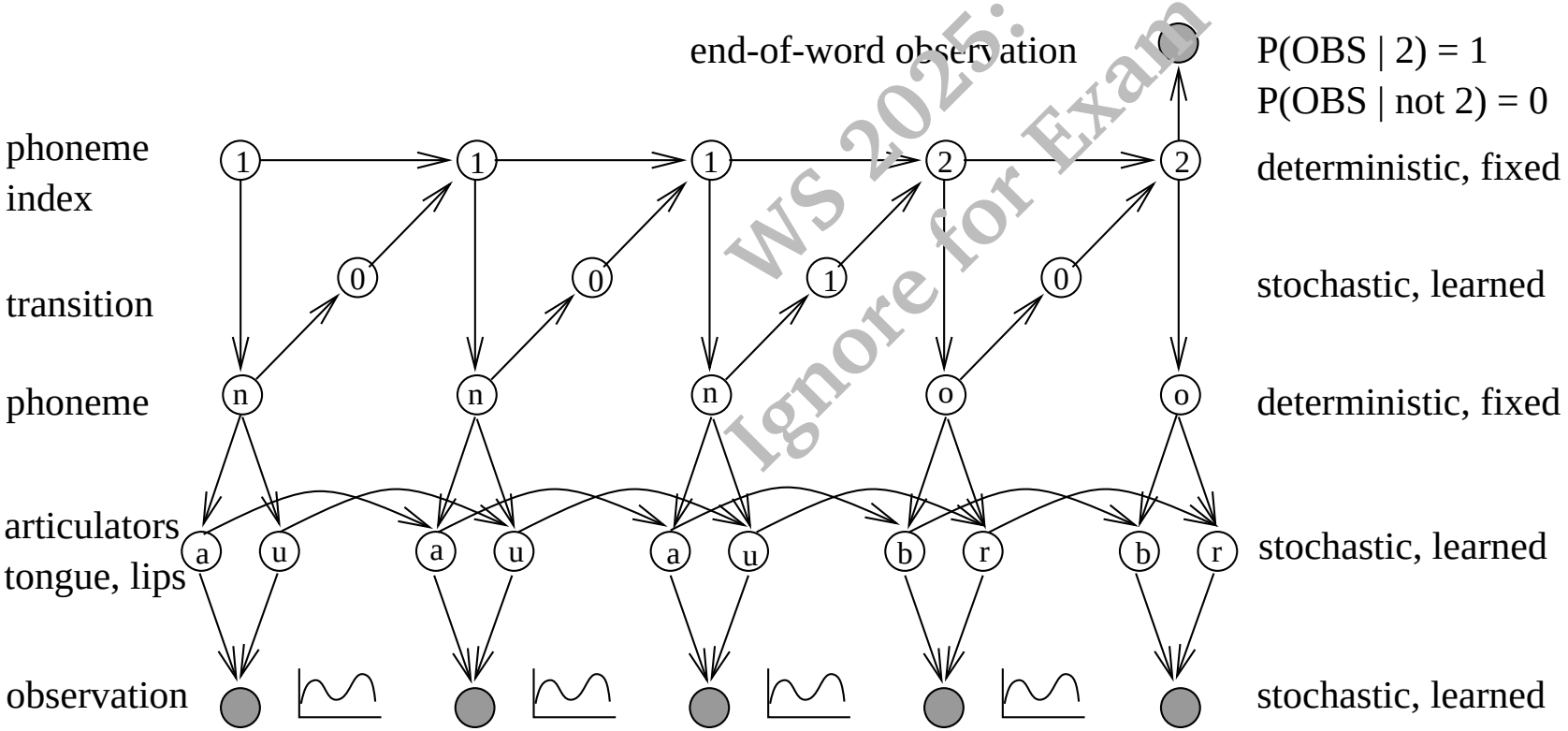
$$P(w_1 \cdots w_n) = \prod_{i=1}^{n} P(w_i | w_1 \cdots w_{i-1})$$

Often used in practice: Bigram Model:

$$P(w_i | w_1 \cdots w_{i-1}) \approx P(w_i | w_{i-1})$$

Train by counting all word pairs in a large text corpus
More sophisticated models (trigrams, grammars, etc.) help a little bit

A General Dynamic Bayes Net for Speech Recognition[4]



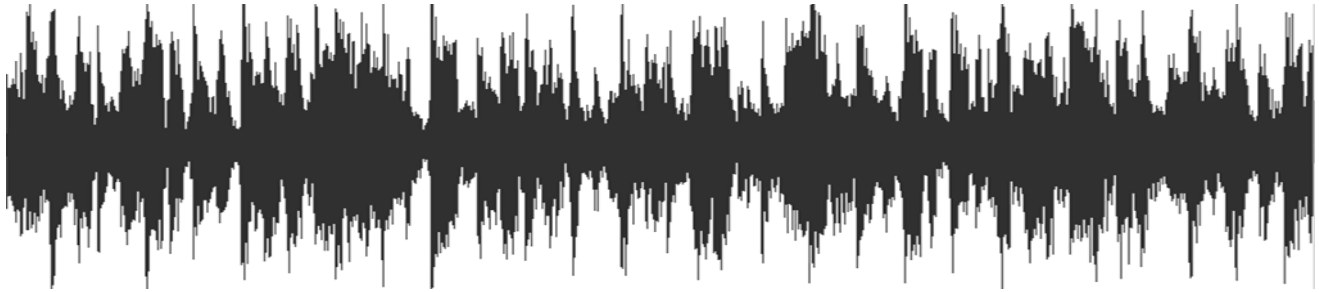
Easy to add additional variables for, e.g., gender, accent, speed ...

[4] See (Zweig & Russell, 1998).

Application Example 2: HMMs for On-line Tracking: Musical Beat and Meter Tracking[5]

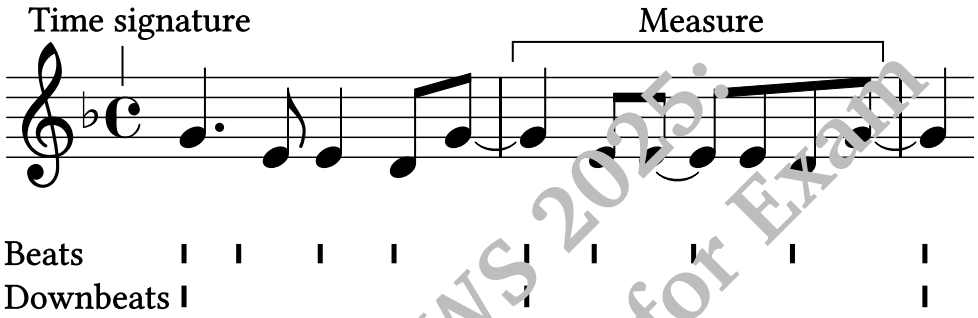
Task:

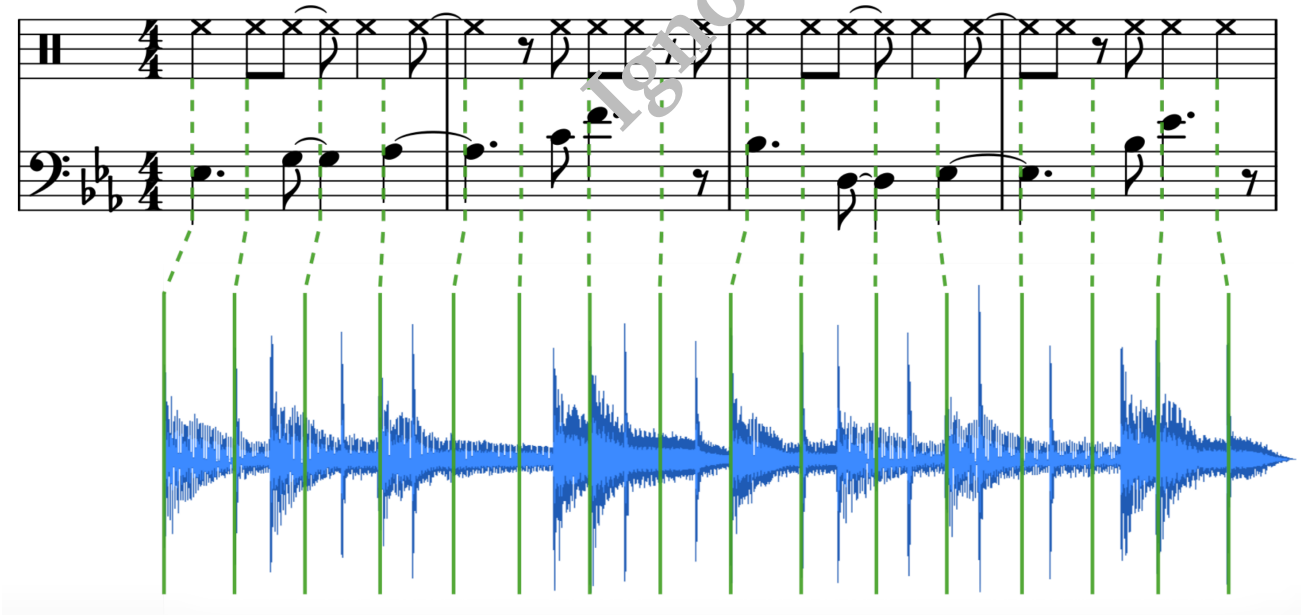
- Listen to a piece of music (audio) in real time
- Find out where the beat is, and track it over time (even if there are changes in tempo)
- In addition, infer what the most likely current rhythm and metrical structure is
- Input is *only* the raw audio signal, no information about the notes etc.



[5] Work by Florian Krebs, Inst. of Computational Perception (Krebs, 2016).
Of course, you need not learn this for the exam.

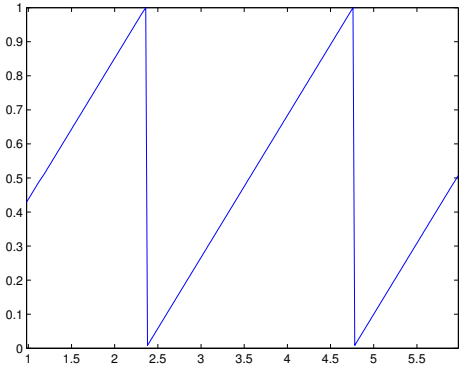
Basic Concepts of Musical Meter and Rhythm





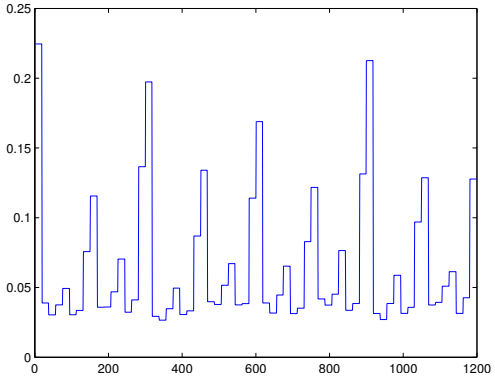
Definition of Hidden State Variables

1. **Bar position** $p_t \in [0, 1[$ (*continuous*)

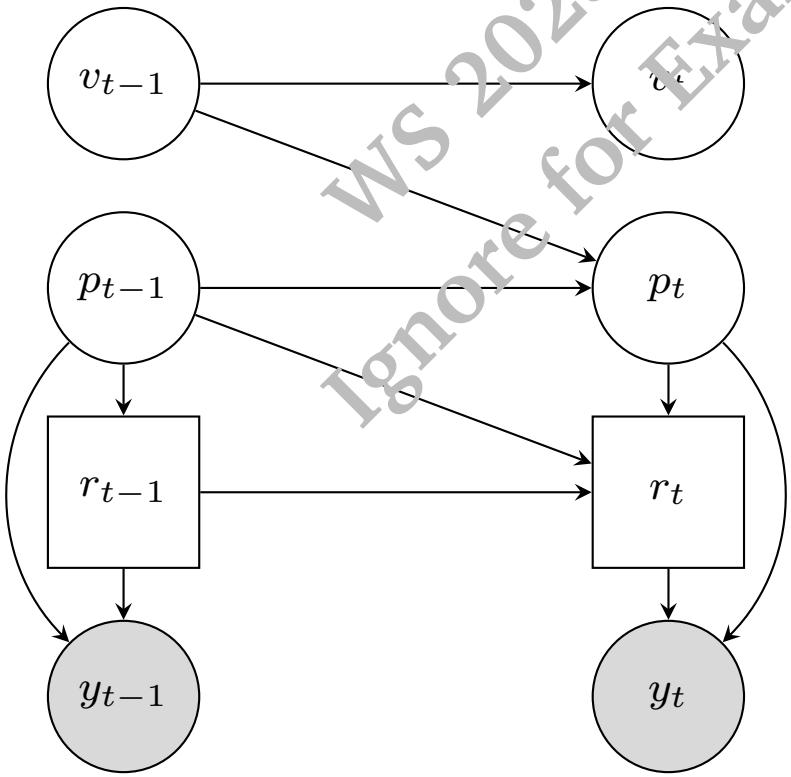


2. **Velocity** v_t of a hypothetical bar pointer in $[\frac{\text{bar positions}}{\text{audio frame}}]$ (*continuous*)

3. **Rhythmic pattern** $r_t \in \{R_1, R_2, ..., R_8\}$ (*discrete ID*)



Dynamic Bayes Net (DBN) Representation



Discretisation

Problem: The DBN model is too complex for exact inference:

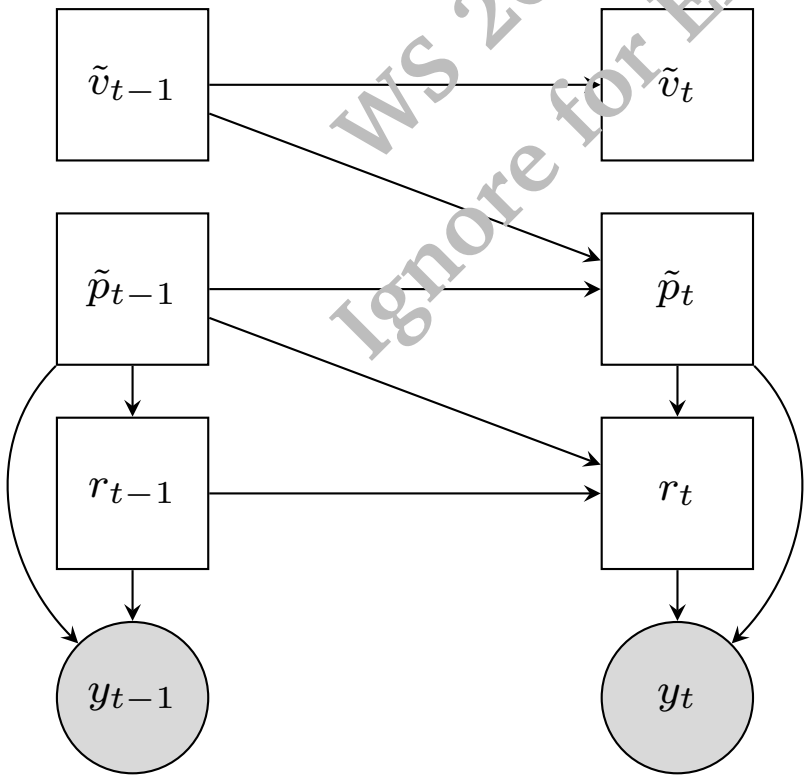
- Mixture of continuous and discrete state variables
- Continuous variables cannot be modelled using Gaussian or linear Gaussian models (see next chapter on the Kalman Filter)

Solution:

- Discretise the continuous variables p_t and v_t into discrete variables $\tilde{p}_t$ (200 values) and $\tilde{v}_t$ (30 values)

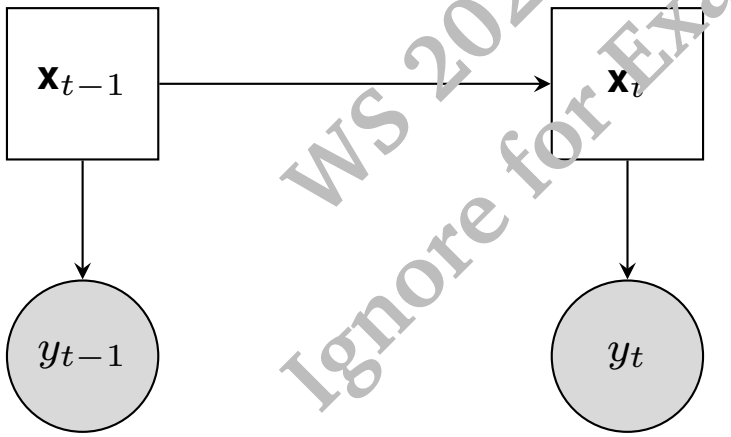
Discretisation

The model approximates to ...



HMM Representation

which can be seen as a HMM with one multi-dimensional discrete state variable:



where $\boldsymbol{x}_t = [\tilde{p}_t, \tilde{v}_t, r_t]^T$

Note:

- Large number of possible states: $|Val(X)| = 200 \times 30 \times 8 = 48,000$
- Transition model $\boldsymbol{A}$ has $48,000 \times 48,000$ entries!
- But: Transition model is very sparse – almost all 0s (see next slide)

Transition Model

The transition model was constructed manually (without learning):

- **Position** $\tilde{p}_k$:
  The position at the next time frame is calculated deterministically as

$$\tilde{p}_k = (\tilde{p}_{k-1} + \tilde{v}_{k-1} - 1) \mod (1) + 1$$

- **Velocity** $\tilde{v}_k$:
  In each time frame, there are three possible velocity transitions:

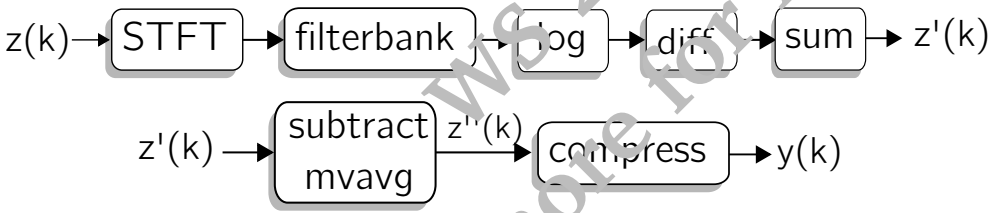
$$p(\tilde{v}_k|\tilde{v}_{k-1}) = \begin{cases} 1 - p_{\tilde{v}}, & \tilde{v}_k = \tilde{v}_{k-1} \\ \frac{p_{\tilde{v}}}{2}, & \tilde{v}_k = \tilde{v}_{k-1} + 0.001 \\ \frac{p_{\tilde{v}}}{2}, & \tilde{v}_k = \tilde{v}_{k-1} - 0.001 \end{cases}$$

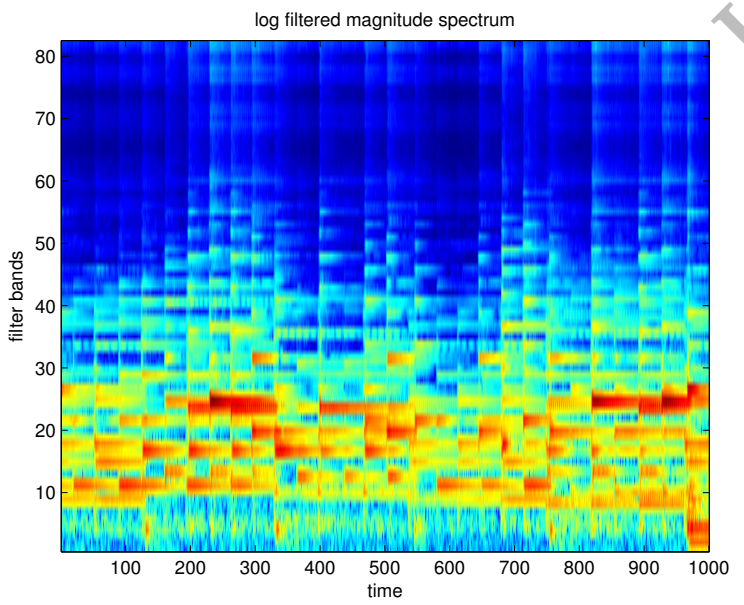
- **Rhythmic patterns** r_k:
  Switching between rhythmic patterns is assumed to be possible only at bar boundaries ($\tilde{p}_k < \tilde{p}_{k-1}$):

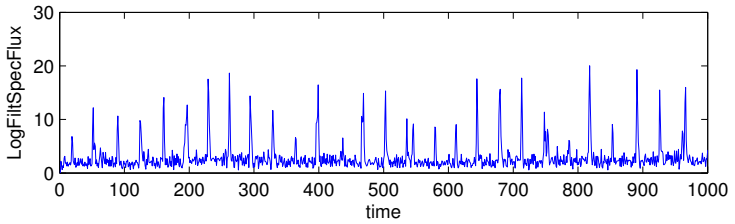
$$p(r_k|r_{k-1}, \tilde{p}_{k-1}, \tilde{p}_k) = \begin{cases} p_r, & \tilde{p}_k < \tilde{p}_{k-1}, r_k \neq r_{k-1} \\ 1 - p_r, & \tilde{p}_k < \tilde{p}_{k-1}, r_k = r_{k-1} \\ 1, & \tilde{p}_k > \tilde{p}_{k-1}, r_k = r_{k-1} \\ 0, & \tilde{p}_k > \tilde{p}_{k-1}, r_k \neq r_{k-1} \end{cases}$$

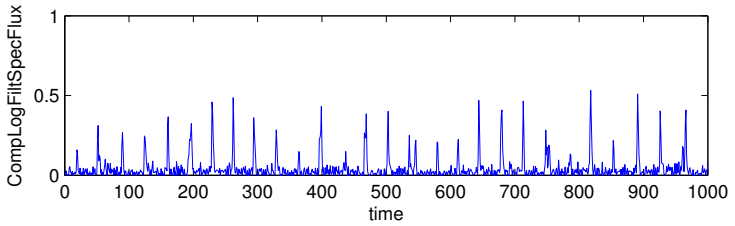
Observation Variable

Observation variable (multi-dimensional continuous feature) y_t computed from the audio input: "Compressed LogFiltSpecFlux"[6]
(computed separately for high and low part of the spectrum)









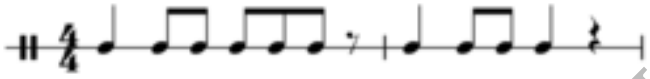
[6] See (Böck & Widmer, 2013)

Observation Model

Probability $p(y_t \mid \tilde{p}_t, r_t)$ of observation y_t depends on bar position and rhythmic pattern.

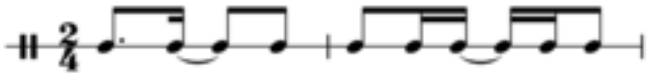
Examples:

Cha-cha-cha

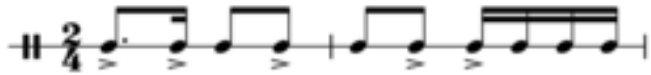


Bolero

Rumba



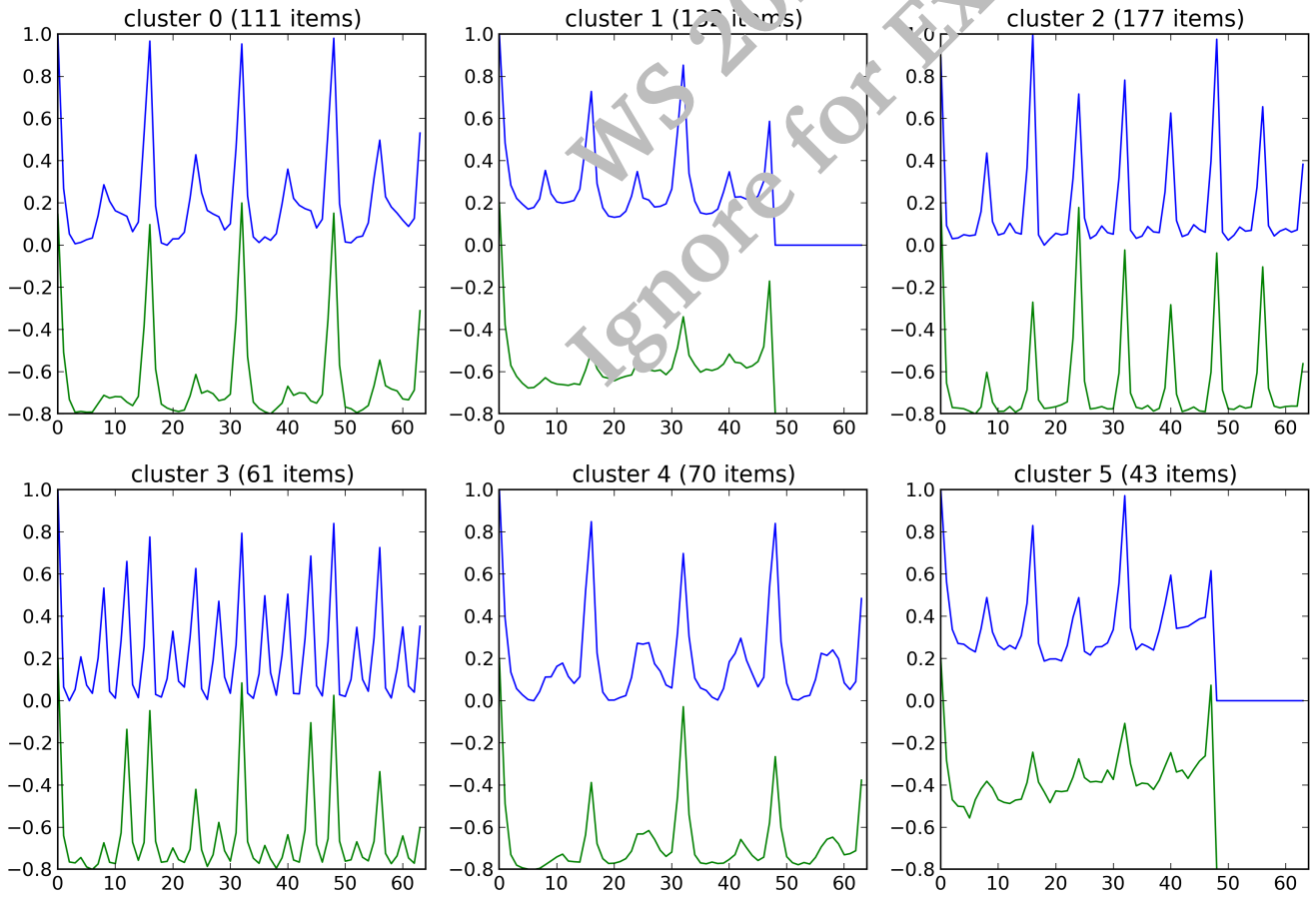
Merengue



Learning Task: For each possible combination of $\tilde{p}_t$ and r_t values (i.e., for each rhythmic pattern, and each possible position within it), learn a Gaussian Mixture Model (GMM) that describes the distribution of the (lo_flux, hi_flux) feature value pairs expected at this point.

Learning of Rhythmic Patterns via k-Means Clustering[7]

Typical rhythmic patterns and the observation model (what observations to expect for different rhythm patterns) are learned from audio recordings



[7] See (Krebs *et al.*, 2014)

Demo: Drumotron 3000

Department of
Computational
Perception

RoboD in the IEEE 2017 Challenge

IEEE Signal Processing Cup 2017
ICASSP 2017, New Orleans

Sebastian Böck
Amaury Durand
Florian Krebs

https://www.youtube.com/watch?v=21X9SWqOgmw&list=UUBUSjbI49Cns5hf1FoCtvUg&index=20

What you should remember of this section

- Definition and formal specification of a HMM
- Inference tasks in HMMs, and basic idea of the corresponding algorithms: Filtering, Prediction, Smoothing, Decoding, Likelihood Computation
- Why these algorithms work, and why they are efficient
- Learning HMMs: Basic idea of Baum-Welch algorithm, and why it is an instance of E-M
- How to express constraints on HMM structure
- How to model continuous observations: Discretisation and GMMs
- Definition of Gaussian Mixture Model (GMM)
- E-M Algorithm for learning GMMs

Literature

Koller, Daphne and Friedman, Nir (2009).
Probabilistic Graphical Models: Principles and Techniques. Cambridge, MA: MIT Press.

Rabiner, Lawrence E. (1989).
A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE* 77(2), 257 - 286.

Russell, Stuart J. and Norvig, Peter (2003).
Artificial Intelligence: A Modern Approach. Englewood Cliffs, NJ: Prentice Hall.

Böck, S. and Widmer, G. (2013).
Maximum Filter Vibrato Suppression for Onset Detection. In *Proceedings of the 16th International Conference on Digital Audio Effects (DAFx-13)*, Maynooth, Ireland.

Krebs, F., Korzeniowski, F., Grachten, M. and Widmer, G. (2014).
Unsupervised Learning and Refinement of Rhythmic Patterns for Beat and Downbeat Tracking. In *Proceedings of the 22nd European Signal Processing Conference (EUSIPCO 2014)*, Lisbon, Portugal.

Krebs, F., Bck, S., and Widmer, G. (2015).
An Efficient State-Space Model for Joint Tempo and Meter Tracking. In *Proceedings of the 16th Conference of the International Society for Music Information Retrieval (ISMIR 2015)*, Malaga, Spain.

Krebs, F. (2016).
Metrical Analysis of Musical Audio Using Probabilistic Models. Ph.D. Dissertation, Inst. of Computational Perception, Johannes Kepler University Linz.

Zweig, G. and Russell, S.J. (1998).
Speech Recognition with Dynamic Bayesian Networks. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI 1998)*, Madison, WI.