Grundlagen der künstlichen Intelligenz – Hidden Markov Models

Matthias Althoff

TU München

December 12, 2019

12/12
12/13
12/19

Organization

1. Time and Uncertainty
2. Hidden Markov Models (HMMs)
3. Inference in Hidden Markov Models
   - Filtering
   - Prediction
   - Smoothing
   - Most Likely Explanation
4. Approximate Inference in Hidden Markov Models
5. Application: Speech Recognition

The content is covered in the AI book by the section "Probabilistic Reasoning Over Time" and Sec. 5 of "Natural Language for Communication".

Learning Outcomes

- You know and understand the definition of *stochastic process*, *Markov process*, *Markov property*, and *stationary process*.
- You can compute the probability distribution of a *stationary Markov chain*.
- You can convert *higher-order Markov chains* to a standard Markov chain.
- You can create a *hidden Markov model*.
- You can compute the joint probability distribution of a hidden Markov model.
- You can perform *filtering*, *prediction*, *smoothing*, and find *the most likely explanation* of a hidden Markov model.
- You can perform *particle filtering* for hidden Markov models.

Motivation

The world changes: We need to track and predict it.

Diabetes management vs vehicle diagnosis

- Vehicle diagnosis: We assume that whatever is broken remains broken during the diagnosis.
- Diabetes management: Blood sugar levels change over time, affecting the diagnosis.

Other examples where the dynamics of the system is essential:

- Locating robots,
- tracking the economic activity of a nation,
- language processing,
- smart grid control,
- etc.

Time-Varying Random Variables

- **Basic idea**: Copy state and evidence variables for each time step.
- **Assumption**: The set of variables does not change over time.
- $\mathbf{X}_t$ = set of unobservable state variables at time t
e.g., $BloodSugar_t$, and $StomachContents_t$.
- $\mathbf{E}_t$ = set of observable evidence variables at time t
e.g., $MeasuredBloodSugar_t$, $PulseRate_t$, and $FoodEaten_t$.
- This assumes **discrete time**; step size depends on problem.
- **Notation**: $\mathbf{X}_{a:b} = \mathbf{X}_a, \mathbf{X}_{a+1}, \ldots, \mathbf{X}_{b-1}, \mathbf{X}_b$.

Conversion to Scalar Random Variables

For finite discrete state spaces, we can assume scalar random variables without loss of generality.

Example

$$\mathbf{X} = [FanOfFCB, LivesIn],$$

where

$$FanOfFCB \in \{true, false\}, \qquad LivesIn \in \{Munich, somewhereElseInGermany\}.$$

We introduce the new scalar random variable $\hat{X} \in \{x_1, x_2, x_3, x_4\}$, where

$$\begin{aligned}
x_1 &\hat{=} [true, Munich],\\
x_2 &\hat{=} [true, somewhereElseInGermany],\\
x_3 &\hat{=} [false, Munich],\\
x_4 &\hat{=} [false, somewhereElseInGermany].
\end{aligned}$$

From now on we will only consider scalar random variables and evidence variables.

Definitions

Stochastic Process

The sequence of random variables X_1, X_2, X_3, etc. is referred to as a **stochastic process**.

Markov Process

A Markov process is a stochastic process that has the Markov property.

Markov Property

Cond prob dist of future states depends only upon the present state, not seq of events that preceded it

A discrete time stochastic process has the Markov property if

$$P(X_n = x_i | X_{n-1} = x_j, X_{n-2} = x_k, \ldots, X_0 = x_l) = \boxed{P(X_n = x_i | X_{n-1} = x_j)}.$$

Stationary Process

A stationary process is a stochastic process whose joint probability distribution does not change when shifted in time. A Markov process is stationary if

$$\forall t: \quad P(X_n = x_i | X_{n-1} = x_j) = P(X_{n+t} = x_i | X_{n-1+t} = x_j).$$

Stationary Markov Chain

A stationary Markov chain is a discrete stationary process with the Markov property. The probability distribution is obtained by the law of total probability:

$$P(X_n = x_i) = \sum_{j=1}^{N} P(X_n = x_i | X_{n-1} = x_j) P(X_{n-1} = x_j)$$

For convenience, we write the above formula in matrix notation:

$$\mathbf{p}_n = \mathbf{T}\mathbf{p}_{n-1},$$

where

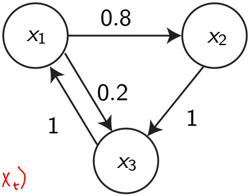
$$(p_i)_n = P(X_n = x_i)$$
$$T_{i,j} = P(X_n = x_i | X_{n-1} = x_j).$$

Example:

$$\mathbf{p}_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{T} = \begin{bmatrix} 0 & 0 & 1 \\ 0.8 & 0 & 0 \\ 0.2 & 1 & 0 \end{bmatrix}.$$

Transition Matrix

$T: p(X_{t+1} | X_t)$



Stationary Markov Chain: Computing Probabilities

The probabilities can be obtained iteratively using

$$\mathbf{p}_n = \mathbf{T}\mathbf{p}_{n-1},$$

or from

$$\mathbf{p}_n = \mathbf{T}\dots(\mathbf{T}(\mathbf{T}\mathbf{p}_0)) = \mathbf{T}^n\mathbf{p}_0. \quad * \tag{1}$$

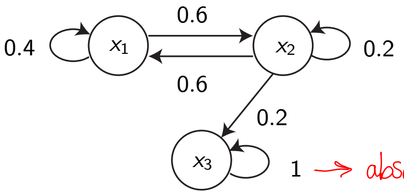
Thus, the probabilities for the previous example become

$$\mathbf{p}_1 = \mathbf{T}\mathbf{p}_0 = \begin{bmatrix} 0 \\ 0.8 \\ 0.2 \end{bmatrix}, \quad \mathbf{p}_2 = \mathbf{T}\mathbf{p}_1 = \begin{bmatrix} 0.2 \\ 0 \\ 0.8 \end{bmatrix}.$$

Tweedback Questions

Given is the following Markov chain:

$$T = \begin{bmatrix} 0.4 & 0.6 & 0 \\ 0.6 & 0.7 & 0 \\ 0 & 0.2 & 1 \end{bmatrix}$$



$\rightarrow$ absorbing state

- What is the one-step probability for $\mathbf{p}_0 = [1,\ 0,\ 0]^T$?
  - A $\mathbf{p}_1 = [0.4,\ 0.6,\ 0]^T$
  - B $\mathbf{p}_1 = [0.4,\ 0.4,\ 0.2]^T$
- What is the probability for $n \to \infty$?
  - A $\lim_{n\to\infty} \mathbf{p}_n = [0.4,\ 0.4,\ 0.2]^T$
  - B $\lim_{n\to\infty} \mathbf{p}_n = [0,\ 0,\ 1]^T$

Conversion of Higher-Order Markov Chains

Higher-order Markov chains

A Markov chain of m^{th} order is defined as

$$\begin{aligned} &P(X_n = x_i | X_{n-1} = x_j, X_{n-2} = x_k, \ldots, X_1 = x_o) \\ =&P(X_n = x_i | X_{n-1} = x_j, X_{n-2} = x_k, \ldots, X_{n-m} = x_l) \text{ for } n > m \end{aligned}$$

We can always rewrite a higher-order Markov chain to a normal one by introducing new states corresponding to the sequence of the m previous states:

$$\begin{aligned} \hat{x}_1 &\hat{=} \overbrace{(x_1, \ldots, x_1, x_1)}^{m \text{ entries}} \\ \hat{x}_2 &\hat{=} (x_1, \ldots, x_1, x_2) \\ &\ldots \\ \hat{x}_d &\hat{=} (x_1, \ldots, x_1, x_{d_1}) \\ \hat{x}_{d+1} &\hat{=} (x_1, \ldots, x_2, x_1) \\ &\ldots \\ \hat{x}_\eta &\hat{=} (x_{d_m}, \ldots, x_{d_2}, x_{d_1}) \qquad (\eta = \prod_{i=1}^{m} d_i) \end{aligned}$$

Conversion of Higher-Order Markov Chains: Example

Before:

$$P(X_n = x_1 | X_{n-1} = x_1, X_{n-2} = x_1) = a$$
$$P(X_n = x_1 | X_{n-1} = x_1, X_{n-2} = x_2) = b$$
$$P(X_n = x_1 | X_{n-1} = x_2, X_{n-2} = x_1) = c$$
$$P(X_n = x_1 | X_{n-1} = x_2, X_{n-2} = x_2) = d$$
$$P(X_n = x_2 | X_{n-1} = x_1, X_{n-2} = x_1) = e$$
$$P(X_n = x_2 | X_{n-1} = x_1, X_{n-2} = x_2) = f$$
$$P(X_n = x_2 | X_{n-1} = x_2, X_{n-2} = x_1) = g$$
$$P(X_n = x_2 | X_{n-1} = x_2, X_{n-2} = x_2) = h$$

After:

$$P(X_n = \hat{x}_1 | X_{n-1} = \hat{x}_1) = a$$
$$P(X_n = \hat{x}_1 | X_{n-1} = \hat{x}_2) = b$$
$$P(X_n = \hat{x}_2 | X_{n-1} = \hat{x}_3) = c$$
$$P(X_n = \hat{x}_2 | X_{n-1} = \hat{x}_4) = d$$
$$P(X_n = \hat{x}_3 | X_{n-1} = \hat{x}_1) = e$$
$$P(X_n = \hat{x}_3 | X_{n-1} = \hat{x}_2) = f$$
$$P(X_n = \hat{x}_4 | X_{n-1} = \hat{x}_3) = g$$
$$P(X_n = \hat{x}_4 | X_{n-1} = \hat{x}_4) = h$$

New states for $\mathcal{D}_x = \{x_1, x_2\}$:

$$\hat{x}_1 \widehat{=} (x_1, x_1)$$
$$\hat{x}_2 \widehat{=} (x_1, x_2)$$
$$\hat{x}_3 \widehat{=} (x_2, x_1)$$
$$\hat{x}_4 \widehat{=} (x_2, x_2)$$

Sensor Model

In many examples, the state of a system cannot be directly measured (see lecture Cyber-Physical Systems) and has to be inferred from sensor values.

Examples:

- Identify persons from images
- Identify drowsiness of human drivers
- Speech recognition
- Stock market analysis

We assume that the random sensor values E_t only depend on the current state:

$$\mathbf{P}(E_t | X_{0:t}, E_{0:t-1}) = \mathbf{P}(E_t | X_t).$$

If this is not the case, one simply has to add more states to the system.

Hidden Markov Model

Combining a Markov chain with the previous sensor model results in a **hidden Markov model** (HMM).

After introducing the probabilities

$$(p_i)_n = P(X_n = x_i) \quad \text{States}$$
$$(\hat{p}_i)_n = P(E_n = e_i) \quad \text{Evidence}$$

and the matrices

$$T_{i,j} = P(X_n = x_i | X_{n-1} = x_j), \quad \text{Transition}$$
$$H_{i,j} = P(E_n = e_i | X_n = x_j). \quad \text{Measurement}$$

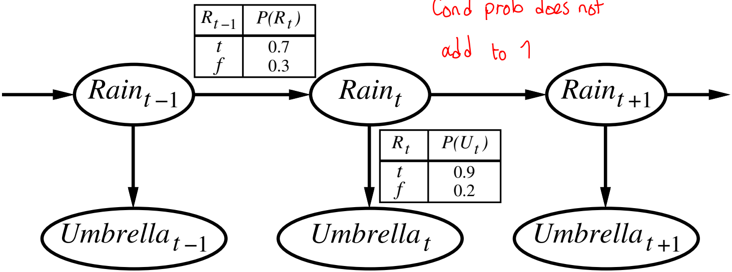
We can compute the probabilities as

$$\mathbf{p}_n = \mathbf{T}\mathbf{p}_{n-1},$$
$$\hat{\mathbf{p}}_n = \mathbf{H}\mathbf{p}_n.$$

Umbrella Example

- You are the security guard stationed at a secret underground installation.
- You want to know if it is rainy today.
- Your only measurement is to check whether the director coming in has an umbrella or not.
- The state is $X_t = Rain_t$ and the measurement is $E_t = Umbrella_t$.

Cond prob does not add to 1



R_{t-1}	$P(R_t)$
t	0.7
f	0.3

R_t	$P(U_t)$
t	0.9
f	0.2

Joint Probability Distribution (1)

Given the initial probability distribution at time 0, $\mathbf{P}(X_0)$, we can compute the joint probability distribution of state and measurement using the chain rule and the Markov property:

$$* \quad \mathbf{P}(X_{0:t}, E_{1:t}) = \left(\prod_{i=1}^{t} \mathbf{P}(E_i|X_i)\mathbf{P}(X_i|X_{i-1}) \right) \mathbf{P}(X_0).$$

Explanation of formula by example:

* Evidence depend only on the current state & not on the previous state

$$\begin{aligned} & P(E_1|X_1)P(X_1|X_0)P(X_0) \\ =& P(E_1|X_1, X_0)P(X_1|X_0)P(X_0) \\ =& P(E_1|X_1, X_0)P(X_1, X_0) \\ =& P(E_1, X_1, X_0). \end{aligned}$$

Joint Probability Distribution (2)

Reminder:

$$\mathbf{P}(X_{0:t}, E_{1:t}) = \left(\prod_{i=1}^{t} \mathbf{P}(E_i|X_i)\mathbf{P}(X_i|X_{i-1})\right)\mathbf{P}(X_0).$$

Given are the initial probability distribution $P(Rain_0 = true) = 0.2$, $P(Rain_0 = false) = 0.8$ and the conditional probabilities from slide 15.

First iteration ($t = 1$):

$$P(r_0, r_1, u_1) = P(u_1|r_1)P(r_1|r_0)P(r_0) = 0.9 \cdot 0.7 \cdot 0.2 = 0.126,$$
$$P(r_0, r_1, \neg u_1) = P(\neg u_1|r_1)P(r_1|r_0)P(r_0) = 0.1 \cdot 0.7 \cdot 0.2 = 0.014, \ldots$$

Second iteration ($t = 2$):

$$\begin{aligned} P(r_0, r_1, r_2, u_1, u_2) &= P(u_2|r_2)P(r_2|r_1)P(u_1|r_1)P(r_1|r_0)P(r_0) \\ &= 0.9 \cdot 0.7 \cdot 0.9 \cdot 0.7 \cdot 0.2 = 0.07938, \\ P(r_0, r_1, r_2, \neg u_1, u_2) &= P(u_2|r_2)P(r_2|r_1)P(\neg u_1|r_1)P(r_1|r_0)P(r_0) \\ &= 0.9 \cdot 0.7 \cdot 0.1 \cdot 0.7 \cdot 0.2 = 0.0088, \ldots \end{aligned}$$

Inference Tasks in Hidden Markov Models

Important

given $e_{1:t}$

1. **Filtering**: $\mathbf{P}(X_t|e_{1:t})$ "State Estimation"
   belief state – input to the decision process of a rational agent
2. **Prediction**: $\mathbf{P}(X_{t+k}|e_{1:t})$ for $k > 0$
   evaluation of possible action sequences;
   like filtering without the evidence
3. **Smoothing**: $\mathbf{P}(X_k|e_{1:t})$ for $0 \leq k < t$ "within"
   better estimate of past states, essential for learning
4. **Most likely explanation**: $\arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$
   speech recognition, decoding with a noisy channel

Filtering (1)

Aim

Devise a **recursive** state estimation algorithm:

$$\mathbf{P}(X_{t+1}|e_{1:t+1}) = f(e_{t+1}, \mathbf{P}(X_t|e_{1:t}))$$

Such a recursive algorithm can be obtained as follows:

$$\begin{aligned}
&\mathbf{P}(X_{t+1}|e_{1:t+1}) \\
&= \mathbf{P}(X_{t+1}|e_{1:t}, e_{t+1}) \qquad \text{(dividing the evidence)} \\
&= \alpha \mathbf{P}(e_{t+1}|X_{t+1}, e_{1:t})\mathbf{P}(X_{t+1}|e_{1:t}) \qquad \text{(using Bayes' rule)} \\
&= \alpha \mathbf{P}(e_{t+1}|X_{t+1})\mathbf{P}(X_{t+1}|e_{1:t}) \qquad \text{(Markov assumption on sensors)}
\end{aligned}$$

The probability $\mathbf{P}(X_{t+1}|e_{1:t})$ represents a one-step prediction of the next state as discussed on the next slide.

Filtering (2)

Reminder:

$$\mathbf{P}(X_{t+1}|e_{1:t+1}) = \alpha \mathbf{P}(e_{t+1}|X_{t+1})\mathbf{P}(X_{t+1}|e_{1:t})$$

Prediction by summing out X_t:

$$\mathbf{P}(X_{t+1}|e_{1:t+1}) = \alpha \mathbf{P}(e_{t+1}|X_{t+1}) \underbrace{\sum_{x_t} \mathbf{P}(X_{t+1}|\underline{x_t}, e_{1:t}) P(\underline{x_t}|e_{1:t})}_{\mathbf{P}(X_{t+1}|e_{1:t})}$$

$$= \alpha \underbrace{\mathbf{P}(e_{t+1}|X_{t+1})}_{\text{sensor model}} \sum_{x_t} \underbrace{\mathbf{P}(X_{t+1}|x_t)}_{\text{transition model}} \underbrace{P(x_t|e_{1:t})}_{\text{previous result or current state dist}}. \qquad \text{(Markov assumption)} \qquad (2)$$

- $\mathbf{P}(e_{t+1}|X_{t+1}) = \mathbf{P}(e_t|X_t)$ is directly obtained from the sensor model.
- $\mathbf{P}(X_{t+1}|x_t)$ comes from the transition model.
- $P(x_t|e_{1:t})$ comes from the current state distribution.
- Algorithm is time and space **constant** (independent of t).

Filtering: Matrix Notation

Reminder:

$$\mathbf{P}(X_{t+1}|e_{1:t+1}) = \alpha \mathbf{P}(e_{t+1}|X_{t+1}) \sum_{x_t} \mathbf{P}(X_{t+1}|x_t) P(x_t|e_{1:t}).$$

To bring the filtering algorithm in matrix notation, we introduce

$$(f_i)_{1:t} = P(X_t = x_i|e_{1:t})$$
$$(O_{ij})_t = \begin{cases} P(e_t|X_t = x_i), \text{ if } j = i \\ 0, \text{ otherwise.} \end{cases}$$

This makes it possible to write (2) as

$$\mathbf{f}_{1:t+1} = \alpha \mathbf{O}_{t+1} \mathbf{T} \mathbf{f}_{1:t}$$

where $\mathbf{T}$ was the transition matrix. Note that the transition matrix is defined as its transpose in the AI book.

Filtering: Umbrella Example (1)

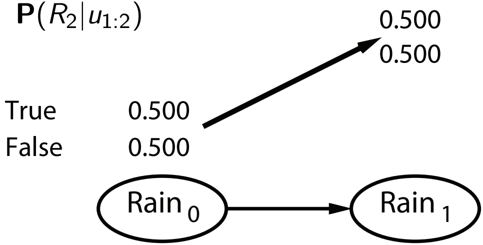
Query: $\mathbf{P}(R_2 | u_{1:2})$

True	0.500
False	0.500

Day 0: No observations; Only the security guard's belief, which is $\mathbf{P}(R_0) = \langle 0.5, 0.5 \rangle$.

Filtering: Umbrella Example (2)

Query: $\mathbf{P}(R_2|u_{1:2})$



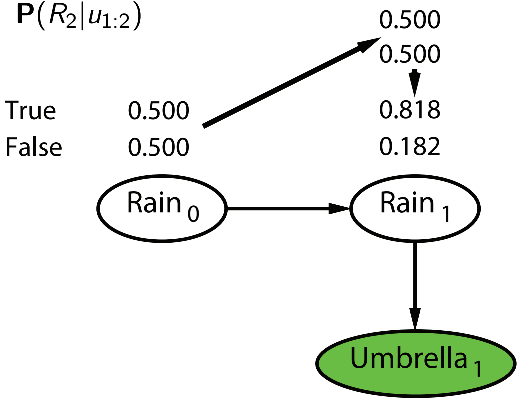
Day 1: The prediction from $t = 0$ to $t = 1$ is

$$\mathbf{P}(R_1) = \sum_{r_0} \mathbf{P}(R_1|r_0)P(r_0) = \langle 0.7, 0.3 \rangle \times 0.5 + \langle 0.3, 0.7 \rangle \times 0.5 = \langle 0.5, 0.5 \rangle.$$

t
f

Filtering: Umbrella Example (3)

Query: $\mathbf{P}(R_2|u_{1:2})$

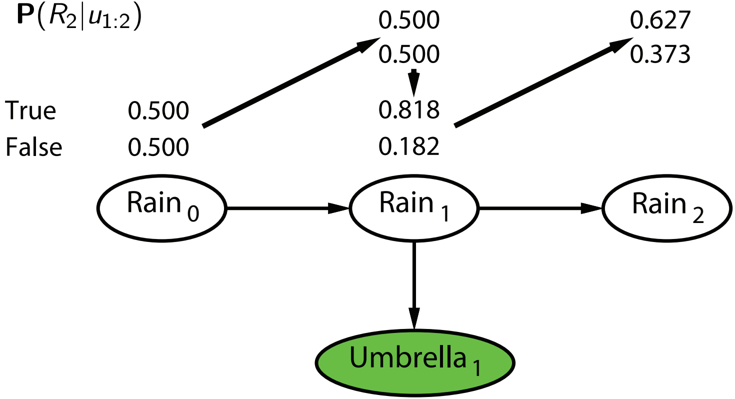


Day 1: The umbrella appears, we incorporate the measurement

$$\boxed{\mathbf{P}(R_1|u_1)} = \alpha\mathbf{P}(u_1|R_1)\mathbf{P}(R_1) = \alpha(\langle 0.9, 0.2\rangle \times \langle 0.5, 0.5\rangle) = \alpha\langle 0.45, 0.1\rangle$$
$$\approx \langle 0.818, 0.182\rangle.$$

Filtering: Umbrella Example (4)

Query: $\mathbf{P}(R_2|u_{1:2})$

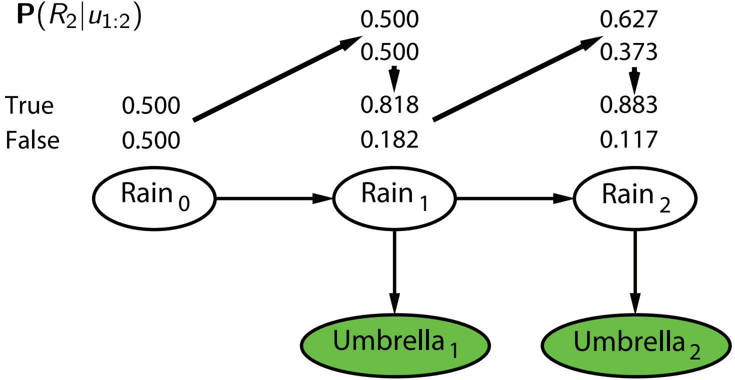


Day 2: The prediction from $t = 1$ to $t = 2$ is

$$\boxed{\mathbf{P}(R_2|u_1)} = \sum_{r_1} \mathbf{P}(R_2|r_1)P(r_1|u_1) = \langle 0.7, 0.3\rangle \times 0.818 + \langle 0.3, 0.7\rangle \times 0.182$$
$$\approx \langle 0.627, 0.373\rangle.$$

Filtering: Umbrella Example (5)

Query: $\mathbf{P}(R_2|u_{1:2})$



Day 2: The umbrella appears, we incorporate the measurement

$$\mathbf{P}(R_2|u_1, u_2) = \alpha \mathbf{P}(u_2|R_2)\mathbf{P}(R_2|u_1) = \alpha(\langle 0.9, 0.2\rangle \times \langle 0.627, 0.373\rangle)$$
$$= \alpha\langle 0.565, 0.075\rangle \approx \langle 0.883, 0.117\rangle.$$

Filtering: Umbrella Example in Matrix Notation

Observation matrices:

$$\mathbf{O}_1 = \mathbf{O}_2 = \begin{bmatrix} 0.9 & 0 \\ 0 & 0.2 \end{bmatrix}.$$

Transition matrix:

$$\mathbf{T} = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}.$$

$$\mathbf{f}_{1:t+1} = \alpha \mathbf{O}_{t+1} \mathbf{T} \mathbf{f}_{1:t}$$

$$\mathbf{f}_{1:1} = \alpha \begin{bmatrix} 0.9 & 0 \\ 0 & 0.2 \end{bmatrix} \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \alpha \begin{bmatrix} 0.45 \\ 0.1 \end{bmatrix} \approx \begin{bmatrix} 0.818 \\ 0.182 \end{bmatrix}$$

$$\mathbf{f}_{1:2} = \alpha \begin{bmatrix} 0.9 & 0 \\ 0 & 0.2 \end{bmatrix} \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix} \begin{bmatrix} 0.818 \\ 0.182 \end{bmatrix} = \alpha \begin{bmatrix} 0.5645 \\ 0.0746 \end{bmatrix} \approx \begin{bmatrix} 0.883 \\ 0.117 \end{bmatrix}$$

Prediction

2

- The task of **prediction** can be seen simply as filtering without the addition of new evidence.
- The filtering process already incorporates a one-step prediction.

It is trivial to see that

$$\mathbf{P}(X_{t+k+1}|e_{1:t}) = \sum_{x_{t+k}} \mathbf{P}(X_{t+k+1}|x_{t+k})P(x_{t+k}|e_{1:t}).$$

Comments:

- As $k \to \infty$, $P(x_{t+k}|e_{1:t})$ tends to the **stationary distribution** of the included Markov chain, where $\mathbf{p}_{t+k} = \mathbf{T}^k \mathbf{p}_t$ (see slide 9).
- It is obvious that we cannot accurately predict the state when the time horizon is relatively long.

Prediction: Umbrella Example (1)

Reminder: We use the probabilities

$$(p_i)_n = P(X_n = x_i)$$
$$(\hat{p}_i)_n = P(E_n = e_i)$$

and the matrices

$$T_{i,j} = P(X_n = x_i | X_{n-1} = x_j),$$
$$H_{i,j} = P(E_n = e_i | X_n = x_j).$$

Umbrella example

Given $x_1 \hat{=} r, x_2 \hat{=} \neg r, e_1 = u, e_2 = \neg u$, the initial probability distribution, and the conditional probabilities from slide 15, we have:

Complements are col-wise

$$\mathbf{p}_0 = \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 0.9 & 0.2 \\ 0.1 & 0.8 \end{bmatrix}.$$

slide 17

Prediction: Umbrella Example (2)

Using (1) on slide 9, one obtains

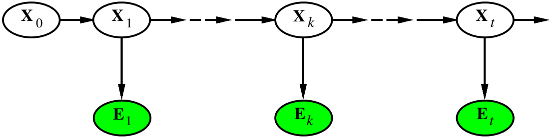
$$\boxed{\begin{aligned}\mathbf{p}_n &= \mathbf{T}^n \mathbf{p}_0,\\ \hat{\mathbf{p}}_n &= \mathbf{H}\mathbf{p}_n.\end{aligned}}$$

Umbrella example

$$\mathbf{p}_0 = \begin{bmatrix}0.2\\0.8\end{bmatrix}, \quad \mathbf{T}^{100} = \begin{bmatrix}0.5 & 0.5\\0.5 & 0.5\end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix}0.9 & 0.2\\0.1 & 0.8\end{bmatrix},$$

$$\mathbf{p}_{100} = \begin{bmatrix}0.5\\0.5\end{bmatrix}, \quad \hat{\mathbf{p}}_{100} = \begin{bmatrix}0.55\\0.45\end{bmatrix}.$$

Smoothing (1)
3



- **Smoothing** is the process of computing the distribution over past states given evidence up to the present: $\mathbf{P}(X_k|e_{1:t})$ for $0 \le k < t$.
- In anticipation of creating another recursive algorithm (as for filtering), we divide the evidence $e_{1:t}$ into $e_{1:k}$ and $e_{k+1:t}$:

$$
\begin{aligned}
\mathbf{P}(X_k|e_{1:t}) &= \mathbf{P}(X_k|e_{1:k}, e_{k+1:t}) \\
&= \alpha'\mathbf{P}(X_k, e_{1:k}, e_{k+1:t}) \qquad \text{(normalization)} \\
&= \alpha'\mathbf{P}(e_{k+1:t}|X_k, e_{1:k})\mathbf{P}(X_k|e_{1:k})P(e_{1:k}) \qquad \text{(using chain rule)} \\
&= \alpha\mathbf{P}(X_k|e_{1:k})\mathbf{P}(e_{k+1:t}|X_k, e_{1:k}) \qquad \text{(remove } P(e_{1:k})\text{)} \quad \alpha = \alpha' p(e_{1:k}) \\
&= \alpha\mathbf{P}(X_k|e_{1:k})\mathbf{P}(e_{k+1:t}|X_k) \qquad \text{(using conditional independence)} \\
&= \alpha\mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t} \qquad \text{(f: forward; b: backward)} \quad \hookrightarrow \text{State subsume all prev states} \qquad (3)
\end{aligned}
$$

Smoothing (2)

Reminder: $\mathbf{P}(X_k|e_{1:t}) = \alpha \underbrace{\mathbf{P}(X_k|e_{1:k})}_{\mathbf{f}_{1:k}} \underbrace{\mathbf{P}(e_{k+1:t}|X_k)}_{\mathbf{b}_{k+1:t}}$

- The forward factor $\mathbf{f}_{1:k}$ is computed as for the filtering on slide 20.
- The backward factor $\mathbf{b}_{k+1:t}$ is obtained by the following recursion:

$$
\begin{aligned}
&\mathbf{P}(e_{k+1:t}|X_k) \\
&= \sum_{x_{k+1}} \mathbf{P}(e_{k+1:t}, x_{k+1}|X_k) \qquad \text{(rule for total probability)} \\
&= \sum_{x_{k+1}} \mathbf{P}(e_{k+1:t}|X_k, x_{k+1})\mathbf{P}(x_{k+1}|X_k) \qquad \left(P(a|b,c)P(b|c) = \frac{P(a,b,c)}{P(b,c)}\frac{P(b,c)}{P(c)} = \frac{P(a,b,c)}{P(c)} = P(a,b|c)\right) \\
&= \sum_{x_{k+1}} P(e_{k+1:t}|x_{k+1})\mathbf{P}(x_{k+1}|X_k) \qquad \text{(by conditional independence)} \\
&= \sum_{x_{k+1}} P(e_{k+1}, e_{k+2:t}|x_{k+1})\mathbf{P}(x_{k+1}|X_k) \qquad \text{(by evidence splitting)} \\
&= \sum_{x_{k+1}} P(e_{k+1}|x_{k+1})P(e_{k+2:t}|x_{k+1})\mathbf{P}(x_{k+1}|X_k) \qquad \text{(by cond. ind.)}
\end{aligned}
$$

Smoothing (3)

Sensor model   transition

Reminder:
$$\mathbf{P}(e_{k+1:t}|X_k) = \sum_{x_{k+1}} P(e_{k+1}|x_{k+1}) P(e_{k+2:t}|x_{k+1}) \mathbf{P}(x_{k+1}|X_k) \quad (4)$$

- $P(e_{k+1}|x_{k+1})$ is the sensor model, see slide 13.
- $\mathbf{P}(x_{k+1}|X_k)$ is the transition probability, see slide 8.
- $P(e_{k+2:t}|x_{k+1})$ is obtained by recursive execution of the above formula backwards in time.
- The backwards recursion is denoted by

$$\mathbf{b}_{k+1:t} = \texttt{Backward}(\mathbf{b}_{k+2:t}, e_{k+1}),$$

  which implements (4).
- The forward recursion is denoted by

$$\mathbf{f}_{1:t+1} = \alpha\,\texttt{Forward}(\mathbf{f}_{1:t}, e_{t+1}),$$

  which implements (2) on slide 20.
- The backward phase is initialized by $\mathbf{b}_{t+1:t} = \mathbf{P}(e_{t+1:t}|X_t) = P(\quad|X_t) = \mathbf{1}$, where $\mathbf{1}$ is a vector of ones (probability of observing future evidence is 0).

Smoothing: Matrix Notation

Reminder: $$\mathbf{P}(e_{k+1:t}|X_k) = \sum_{x_{k+1}} P(e_{k+1}|x_{k+1})P(e_{k+2:t}|x_{k+1})\mathbf{P}(x_{k+1}|X_k)$$

To bring the filtering algorithm in matrix notation, we introduce

$$(b_i)_{k+1:t} = P(e_{k+1:t}|X_k = x_i)$$

and use again

$$(O_{ij})_t = \begin{cases} P(e_t|X_t = x_i), \text{ if } j = i \\ 0, \text{ otherwise.} \end{cases}$$

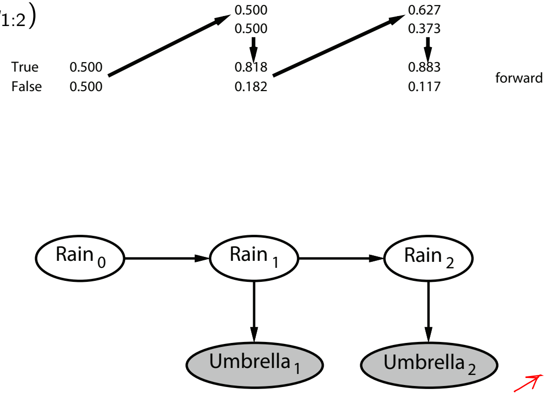
This makes it possible to write (4) as

$$\mathbf{b}_{k+1:t} = \mathbf{T}^T \mathbf{O}_{k+1} \mathbf{b}_{k+2:t},$$

where $\mathbf{T}$ was the transition matrix. Note that the transition matrix is defined as its transpose in the AI book.

Smoothing: Umbrella Example (1)

Query: $\mathbf{P}(R_1|u_{1:2})$

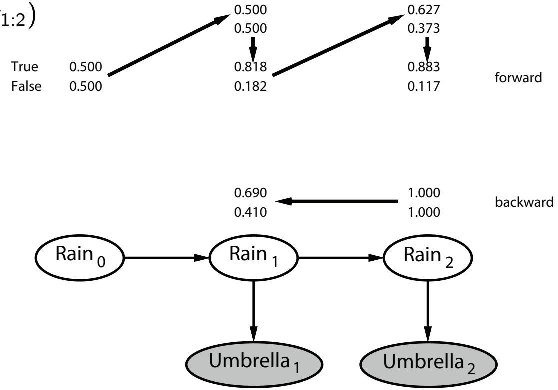


$\alpha f_{1:K} \times b_{K+1:t}$

From (3) on slide 31, we have $\mathbf{P}(R_1|u_{1:2}) = \alpha \mathbf{P}(R_1|u_1)\mathbf{P}(u_2|R_1)$. $\mathbf{P}(R_1|u_1)$ we already know from forward filtering to be $\langle 0.818, 0.182 \rangle$.

Smoothing: Umbrella Example (2)

Query: $\mathbf{P}(R_1|u_{1:2})$



The second term can be computed by the backward recursion in eq. (4):

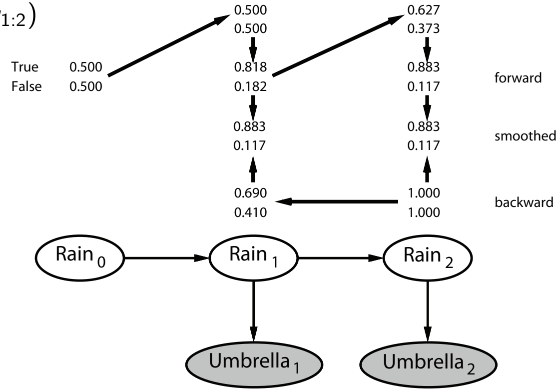
$$\mathbf{P}(u_2|R_1) = \sum_{r_2} P(u_2|r_2)P(\quad|r_2)\mathbf{P}(r_2|R_1)$$

Sensor transition

$$= (0.9 \times 1 \times \langle 0.7, 0.3\rangle) + (0.2 \times 1 \times \langle 0.3, 0.7\rangle) = \langle 0.69, 0.41\rangle.$$

Smoothing: Umbrella Example (3)

Query: $\mathbf{P}(R_1|u_{1:2})$



Inserting the previous results into $\mathbf{P}(R_1|u_{1:2}) = \alpha\mathbf{P}(R_1|u_1)\mathbf{P}(u_2|R_1)$ yields:

$$\mathbf{P}(R_1|u_{1:2}) = \alpha\langle 0.818, 0.182\rangle \times \langle 0.69, 0.41\rangle \approx \langle 0.883, 0.117\rangle.$$

The smoothed estimate for rain is higher than for the filtered one, since the umbrella on day 2 makes it more likely that day 1 was rainy.

Smoothing: Umbrella Example in Matrix Notation

Observation matrices:

$$\mathbf{O}_1 = \mathbf{O}_2 = \begin{bmatrix} 0.9 & 0 \\ 0 & 0.2 \end{bmatrix}.$$

Transition matrix:

$$\mathbf{T} = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}.$$

$$\begin{aligned} \mathbf{b}_{k+1:t} &= \mathbf{T}^T \mathbf{O}_{k+1} \mathbf{b}_{k+2:t} \\ \mathbf{b}_{3:2} &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ \mathbf{b}_{2:2} &= \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}^T \begin{bmatrix} 0.9 & 0 \\ 0 & 0.2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.69 \\ 0.41 \end{bmatrix} \end{aligned}$$

Smoothing: Performance

- Forward and backward recursion take a constant amount of time per step.
- Hence, the time complexity of smoothing with respect to the evidence $e_{1:t}$ for a particular time step k is $\mathcal{O}(t)$.
- For the whole sequence, we need to perform smoothing for all steps, resulting in a time complexity of $\mathcal{O}(t^2)$.
- A better approach uses a simple application of dynamic programming to reduce the complexity to $\mathcal{O}(t)$ (see next slide).
- The idea is to record the results of forward filtering. When running backwards, the stored information is used and not re-computed, resulting in the so-called **forward-backward algorithm**.

Forward-Backward Algorithm (HMM.ipynb)

function Forward-Backward ($\mathbf{ev}$, *prior*) **returns** a vector of probability distributions

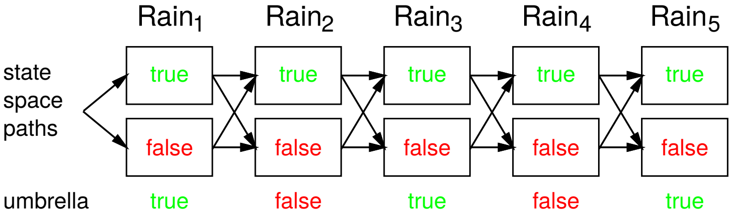
inputs: $\mathbf{ev}$, a vector of evidence values for steps $1, \ldots, t$
prior, the prior distribution on the initial state, $\mathbf{P}(\mathbf{X}_0)$

local variables: $\mathbf{fv}$, a vector of forward messages for steps $0, \ldots, t$
$\mathbf{b}$, a representation of the backward message, initially all ones
$\mathbf{sv}$, a vector of smoothed estimates for steps $1, \ldots, t$

$\mathbf{fv}[0] \leftarrow prior$
for $i = 1$ **to** t **do**
  $\mathbf{fv}[i] \leftarrow \texttt{Forward}(\mathbf{fv}[i-1], \mathbf{ev}[i])$ (see eq. (2) on slide 20)
for $i = t$ **downto** 1 **do**
  $\mathbf{sv}[i] \leftarrow \texttt{Normalize}(\mathbf{fv}[i] \times \mathbf{b})$
  $\mathbf{b} \leftarrow \texttt{Backward}(\mathbf{b}, \mathbf{ev}[i])$ (see eq. (4) on slide 33)
return $\mathbf{sv}$

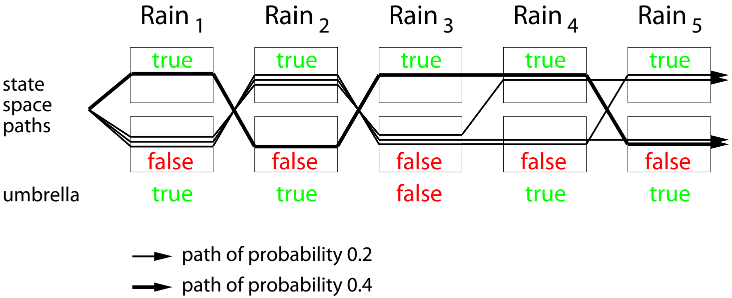
4

Most Likely Explanation



- Given is the above sequence of umbrellas. What is the weather sequence most likely to explain this?
- In all, there are 2^5 possible weather sequences.
- Is there a way to find the most likely one without enumerating all sequences?
- Try smoothening (linear-time procedure): find the distribution for the weather at each time step; then construct the sequence using at each step the most likely one.
- But: **Most likely sequence $\neq$ sequence of most likely states!!!**

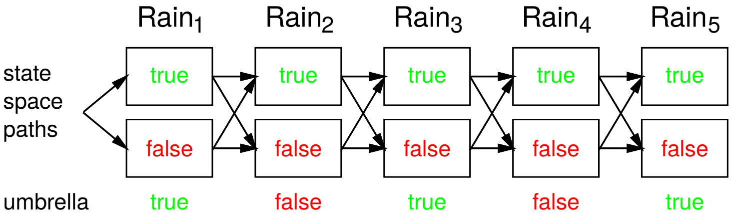
Example: Most Likely Sequence $\neq$ Sequence of Most Likely States



Most likely sequence: true, false, true, true, false

Sequence of most likely states: false, true, false, true , false

Recursive Procedure for the Most Likely Explanation



- Each sequence is a path through a graph whose nodes are the possible states at each time step (see figure above).
- Let us focus on the path that reaches $Rain_5 = true$.
- Because of the Markov property, the most likely path to the state $Rain_5 = true$ consists of
  - the most likely path to **some** state at time 4
  - followed by a transition to $Rain_5 = true$.
- The state at time 4 becomes a part of the most likely path to $Rain_5 = true$.

Viterbi Algorithm (HMM.ipynb)

Previous slide: a recursive relationship between the most likely path to each state x_{t+1} and the most likely path to each state x_t, which we formalize:

$$\max_{x_1 \dots x_t} \mathbf{P}(x_1, \dots, x_t, X_{t+1} | e_{1:t+1}) = \max_{x_1 \dots x_t} \mathbf{P}(x_1, \dots, x_t, X_{t+1} | e_{1:t}, e_{t+1})$$

$$= \max_{x_1 \dots x_t} \alpha \mathbf{P}(x_1, \dots, x_t, X_{t+1}, e_{t+1} | e_{1:t}) \qquad \text{(normalization)}$$

$$= \max_{x_1 \dots x_t} \alpha \mathbf{P}(e_{t+1} | x_1, \dots, x_t, X_{t+1}, e_{1:t}) \mathbf{P}(x_1, \dots, x_t, X_{t+1} | e_{1:t}) \qquad \left(P(a,b|c) = \frac{P(a,b,c)}{P(a,c)} \frac{P(a,c)}{P(c)} = P(b|a,c)P(a|c)\right)$$

$$= \alpha \mathbf{P}(e_{t+1} | X_{t+1}) \max_{x_1 \dots x_t} \mathbf{P}(x_1, \dots, x_t, X_{t+1} | e_{1:t}) \qquad \text{(conditional independence)}$$

$$= \alpha \underbrace{\mathbf{P}(e_{t+1} | X_{t+1})}_{\text{sensor}} \max_{x_t} \Big(\underbrace{\mathbf{P}(X_{t+1} | x_t)}_{\text{transition}} \max_{x_1 \dots x_{t-1}} \underbrace{P(x_1, \dots, x_{t-1}, x_t | e_{1:t})}_{\text{previous result}} \Big)$$

- The algorithm is called **Viterbi algorithm** and is similar in structure compared to the filtering procedure in eq. (2) on slide 20.
- Like the filtering algorithm, the Viterbi algorithm has time complexity $\mathcal{O}(t)$.
- Unlike filtering, which uses constant space, the space requirement is $\mathcal{O}(t)$ since one has to keep the pointers for the best sequence to each state.

Viterbi Algorithm: Interpretation of the max-Operator

The Viterbi algorithm requires the evaluation of the max-operator:

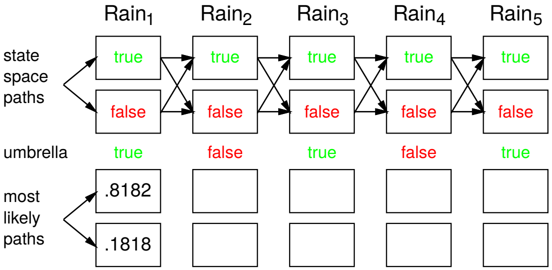
$$\max_{x_1 \ldots x_t} \mathbf{P}(x_1, \ldots, x_t, X_{t+1} | e_{1:t+1})$$
$$= \alpha \mathbf{P}(e_{t+1} | X_{t+1}) \max_{x_t} \left(\mathbf{P}(X_{t+1} | x_t) \max_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, x_t | e_{1:t}) \right).$$

The semantics of the max-operator in combination with the **P** operator is demonstrated by example for $X_t \in \{true, false\}$:

2 different X_t

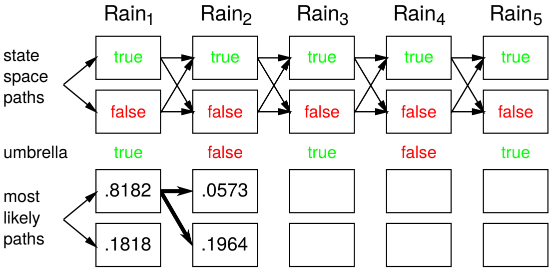
$$\max_{x_t} \mathbf{P}(X_{t+1} | x_t) =$$
$$\left\langle \max_{x_t} P(X_{t+1} = true | x_t), \max_{x_t} P(X_{t+1} = false | x_t) \right\rangle.$$

Most Likely Explanation: Umbrella Example (1)



Initialization with filtering:
$\mathbf{P}(R_1|u_{1:1}) = \langle 0.8182, 0.1818 \rangle$.

Most Likely Explanation: Umbrella Example (2)



Max in bold

We omit the optional normalization from now on.

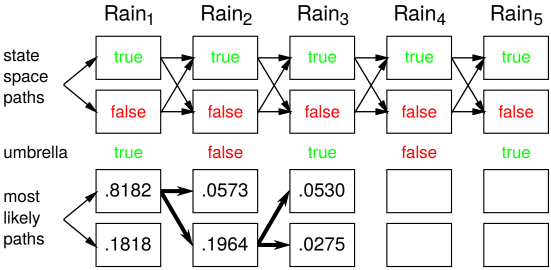
- $r_1 = true: \quad \mathbf{P}(R_2|r_1)P(r_1|u_{1:1}) = \langle 0.7, 0.3\rangle \times 0.8182 = \langle \mathbf{0.5727}, \mathbf{0.2455}\rangle$
- $r_1 = false: \quad \mathbf{P}(R_2|r_1)P(r_1|u_{1:1}) = \langle 0.3, 0.7\rangle \times 0.1818 = \langle 0.0545, 0.1273\rangle$

max bet

- $\max_{r_1} \mathbf{P}(r_1, R_2|u_{1:2}) = \mathbf{P}(u_2|R_2) \max_{r_1} (\mathbf{P}(R_2|r_1)P(r_1|u_{1:1})) =$
  $\langle 0.1, 0.8\rangle \times \langle 0.5727, 0.2455\rangle = \langle 0.0573, 0.1964\rangle$

Most Likely Explanation: Umbrella Example (3)

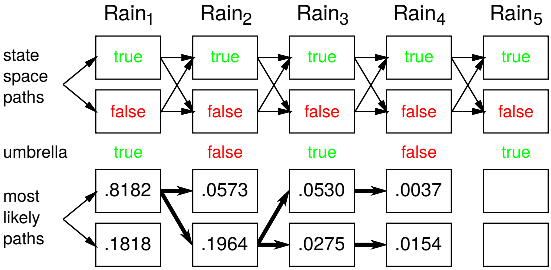
we are only interested in seq path, not state prob



We omit the optional normalization from now on.

- $r_2 = true:\quad \mathbf{P}(R_3|r_2)\max_{r_1} P(r_1, r_2|u_{1:2})$
  $= \langle 0.7, 0.3\rangle \times 0.0573 = \langle 0.0401, 0.0172\rangle$
- $r_2 = false:\quad \mathbf{P}(R_3|r_2)\max_{r_1} P(r_1, r_2|u_{1:2})$
  $= \langle 0.3, 0.7\rangle \times 0.1964 = \langle \mathbf{0.0589}, \mathbf{0.1375}\rangle$
- $\max_{r_1,r_2} \mathbf{P}(r_1, r_2, R_3|u_{1:3}) = \mathbf{P}(u_3|R_3)\max_{r_2}\left(\mathbf{P}(R_3|r_2)\max_{r_1} P(r_1, r_2|u_{1:2})\right) =$
  $\langle 0.9, 0.2\rangle \times \langle 0.0589, 0.1375\rangle = \langle 0.0530, 0.0275\rangle$

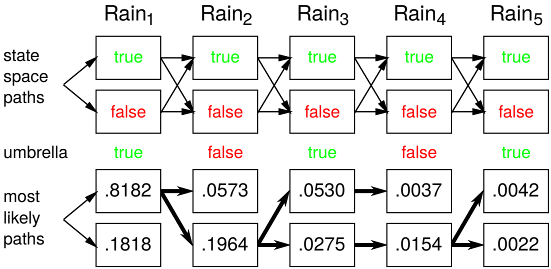
Most Likely Explanation: Umbrella Example (4)



We omit the optional normalization from now on.

- $r_3 = \mathit{true}: \quad \mathbf{P}(R_4|r_3)\max_{r_1,r_2} P(r_1,\ldots,r_3|u_{1:3})$
  $= \langle 0.7, 0.3\rangle \times 0.0530 = \langle \mathbf{0.0371}, 0.0159\rangle$
- $r_3 = \mathit{false}: \quad \mathbf{P}(R_4|r_3)\max_{r_1,r_2} P(r_1,\ldots,r_3|u_{1:3})$
  $= \langle 0.3, 0.7\rangle \times 0.0275 = \langle 0.0082, \mathbf{0.0192}\rangle$
- $\max_{r_1,\ldots,r_3} \mathbf{P}(r_1,\ldots,r_3,R_4|u_{1:4}) =$
  $\mathbf{P}(u_4|R_4)\max_{r_3}\left(\mathbf{P}(R_4|r_3)\max_{r_1,r_2} P(r_1,\ldots,r_3|u_{1:3})\right) =$
  $\langle 0.1, 0.8\rangle \times \langle 0.0371, 0.0192\rangle = \langle 0.0037, 0.0154\rangle$ (here: different r_3 values!)

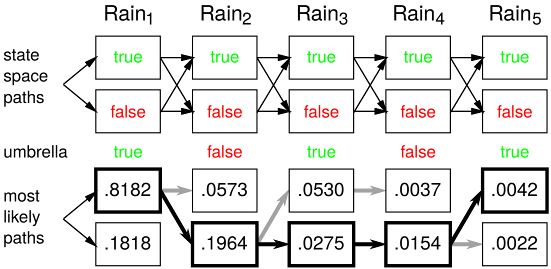
Most Likely Explanation: Umbrella Example (5)



We omit the optional normalization from now on.

- $r_4 = \mathit{true}$: $\quad \mathbf{P}(R_5|r_4)\max_{r_1,\ldots,r_3} P(r_1,\ldots,r_4|u_{1:4})$
  $= \langle 0.7, 0.3\rangle \times 0.0037 = \langle 0.0026, 0.0011\rangle$
- $r_4 = \mathit{false}$: $\quad \mathbf{P}(R_5|r_4)\max_{r_1,\ldots,r_3} P(r_1,\ldots,r_4|u_{1:4})$
  $= \langle 0.3, 0.7\rangle \times 0.0154 = \langle \mathbf{0.0046}, \mathbf{0.0108}\rangle$
- $\max_{r_1,\ldots,r_4} \mathbf{P}(r_1,\ldots,r_4,R_5|u_{1:5}) =$
  $\mathbf{P}(u_5|R_5)\max_{r_4}\left(\mathbf{P}(R_5|r_4)\max_{r_1,\ldots,r_3} P(r_1,\ldots,r_4|u_{1:4})\right) =$
  $\langle 0.9, 0.2\rangle \times \langle 0.0046, 0.0108\rangle = \langle 0.0042, 0.0022\rangle$

Most Likely Explanation: Umbrella Example (6)



Obtaining the most likely sequence:

- start with the most likely final state
- backtracking along the path which maximized the probability of the final state

Estimation of Continuous State Variables

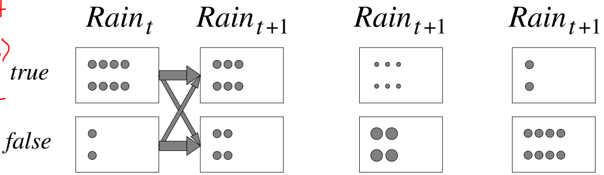
- So far we have only considered estimation of hidden Markov models, which have discrete states.
- Many real-world problems have continuous states, such as position, velocities, forces, temperatures, etc.
- Those problems arise e.g., in
  - robotics,
  - automated driving,
  - smart grids,
  - automated processes, such as in chemical plants,
  - surveillance of automated processes,
  - etc.
- Those aspects are covered in the lecture *Cyber-Physical Systems*.

Particle Filtering (1)

(jupyter HMM.ipynb)

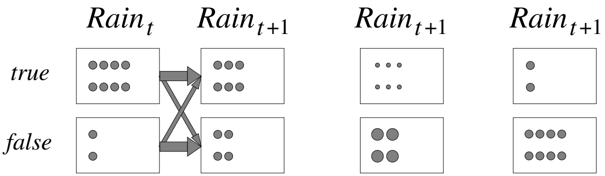
- Particle filtering can be interpreted as a Monte Carlo method for hidden Markov models.
- The approach can also be applied to continuous systems.
- Basic idea: ensure that the population of samples ("particles") tracks the high-likelihood regions of the state-space.
- Widely used for tracking nonlinear systems, especially in computer vision.

PF uses a set of particles (Samples) to rep posterior dist of some stochastic process given noisy n/v partial observations



(a) Propagate (b) Weight (c) Resample

Particle Filtering (2)



(a) Propagate (b) Weight (c) Resample

a) At time t, 8 samples indicate $Rain = true$ and 2 indicate $Rain = false$. Propagation through the transition model yields 6 samples for $Rain = true$ and 4 for $Rain = false$ at time $t + 1$.
b) $Umbrella = false$ is observed at $t + 1$. Each sample is weighted by its likelihood for the observation, as indicated by the size of the circles.
c) A new set of 10 samples is generated by weighted random selection from the current set, resulting in 2 samples for $Rain = true$ and 8 for $Rain = false$.

Consistency of Particle Filtering

- Assume consistency at time t: $$N(x_t|e_{1:t})/N = P(x_t|e_{1:t}). \tag{5}$$

1 - Propagate forward: populations of x_{t+1} are
$$N(x_{t+1}|e_{1:t}) = \sum_{x_t} P(x_{t+1}|x_t)N(x_t|e_{1:t}). \tag{6}$$

2 - Weight samples by their likelihood for e_{t+1}:
$$W(x_{t+1}|e_{1:t+1}) = P(e_{t+1}|x_{t+1})N(x_{t+1}|e_{1:t}). \tag{7}$$

3 - Re-sample to obtain populations proportional to W:

$$\begin{aligned}
N(x_{t+1}|e_{1:t+1})/N &= \alpha W(x_{t+1}|e_{1:t+1}) \\
&= \alpha P(e_{t+1}|x_{t+1})N(x_{t+1}|e_{1:t}) && \text{(using (7))} \\
&= \alpha P(e_{t+1}|x_{t+1}) \sum_{x_t} P(x_{t+1}|x_t)N(x_t|e_{1:t}) && \text{(using (6))} \\
&= \alpha' P(e_{t+1}|x_{t+1}) \sum_{x_t} P(x_{t+1}|x_t)P(x_t|e_{1:t}) && \text{(using (5))} \\
&= P(x_{t+1}|e_{1:t+1}) && \text{(using (2))}
\end{aligned}$$

Filtering slide 20

Speech as Probabilistic Inference

1^{st} ML / NN
2^{nd} best HMM

Motivating example

It's not easy to wreck a nice beach.
It's not easy to recognize speech.
It's not easy to wreck an ice beach.

- Speech signals are noisy, variable, ambiguous.
- What is the **most likely** word sequence, given the speech signal? I.e., choose *Words* to maximize $P(Words|signal)$.
- Use Bayes' rule:

$$P(Words|signal) = \alpha P(signal|Words)P(Words)$$

I.e., decomposes into **acoustic model** + **language model**.

- *Words* are the hidden state sequence, *signal* is the observation sequence.

Phones

Acoustic Model

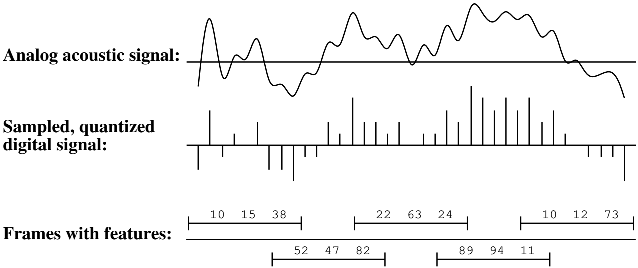
- All human speech is composed from around 100 **phones** (speech sounds), determined by the configuration of **articulators** (lips, teeth, tongue, vocal cords, air flow)
- Form an intermediate level of hidden states between words and signal $\Rightarrow$ acoustic model = pronunciation model + phone model
- 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]

Speech Sounds

Raw signal is the microphone displacement as a function of time; processed into overlapping 30ms **frames**, each described by **features**.



Frame features are often **formants** – peaks in the power spectrum. After discretization, one obtains numbers as shown in the figure.

Phone Models

- Frame features in $P(features|phone)$ summarized by
  - an integer in $[0 \ldots 255]$ (using **vector quantization**); or
  - the parameters of a mixture of Gaussians

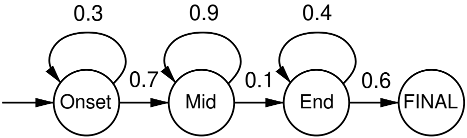
- **Three-state phones**: each phone has three phases (Onset, Mid, End)
  E.g., [t] has silent Onset, explosive Mid, hissing End
  $\rightarrow P(features|phone, phase)$

- **Triphone context**: each phone becomes n^2 distinct phones, depending on the phones to its left and right
  E.g., [t] in "star" is written [t(s,aa)] (different from "tar"!)

- Triphones useful for handling **coarticulation** effects: the articulators have inertia and cannot switch instantaneously between positions
  E.g., [t] in "eighth" has tongue against front teeth

Phone Model Example

Phone HMM for [m]:



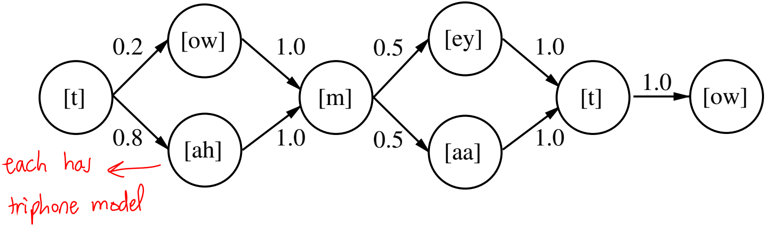
Output probabilities for the phone HMM:

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

- High probabilities for self-transitions indicate that this part has a longer duration.
- The outputs $C_1, \ldots, C_7$ represent combinations of feature values.

Word Pronunciation Models

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



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

- Structure is created manually, transition probabilities learned from data.

Continuous Speech

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".

Continuous speech systems manage 60–80% accuracy on a good day.

2

Language Model

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

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

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.

Combined Hidden Markov Model

- States of the combined language+word+phone model are labeled by the word we're in + the phone in that word + the phone state in that phone.
- Viterbi algorithm finds the most likely **phone state** sequence.
- Does segmentation by considering all possible word sequences and boundaries.
- Does not always give the most likely word sequence because each word sequence is the sum over many state sequences.

Summary

- Temporal models use state and sensor variables replicated over time.
- Markov assumptions and stationarity assumption, so we need
  - transition model $\mathbf{P}(X_t|X_{t-1})$,
  - sensor model $\mathbf{P}(E_t|X_t)$.
- Tasks are filtering, prediction, smoothing, most likely sequence; **all done recursively with constant cost per time step**.
- Hidden Markov models have a single discrete state variable; this is no loss of generality.
- Hidden Markov models are used in numerous applications, such as speech recognition.
- Particle filtering is a good approximative filtering algorithm.