

# Fundamentals of Artificial Intelligence – Hidden Markov Models

Matthias Althoff

TU München

Winter semester 2022/23

# 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

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.

# Overview of Probabilistic Methods

This lecture focuses on dynamic environments without actions.

	<b>Static environment</b>	<b>Dynamic environment</b>
<b>Without actions</b>	Bayesian networks (lecture 9)	Hidden Markov models (lecture 10)
<b>With actions</b>	Decision networks (lecture 11)	Markov decision processes (lecture 12)

# 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.
- $X_t$  = set of unobservable state variables at time  $t$   
e.g.,  $BloodSugar_t$ , and  $StomachContents_t$ .
- $E_t$  = 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:**  $X_{a:b} = X_a, X_{a+1}, \dots, X_{b-1}, X_b$ .

# Conversion to Scalar Random Variables

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

## Example

$$X = [FanOfFCB, LivesIn],$$

where

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

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

$$x_1 \hat{=} [true, Munich],$$

$$x_2 \hat{=} [true, somewhereElseInGermany],$$

$$x_3 \hat{=} [false, Munich],$$

$$x_4 \hat{=} [false, somewhereElseInGermany].$$

From now on we will only consider scalar random variables.

# Definitions

## Stochastic Process

A sequence of random variables  $X_1, X_2, X_3$ , etc. is a **stochastic process**.

## Markov Process

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

## Markov Property

A discrete time stochastic process has the Markov property if

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

## 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

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

# 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:

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

**Convenient matrix notation:**

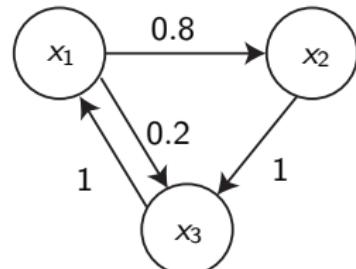
$$\mathbf{p}_n = \mathbf{T} \mathbf{p}_{n-1},$$

where  $(p_i)_n = P(X_n = x_i)$

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

**Example:**

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



# Stationary Markov Chain: Computing Probabilities

The probabilities can be obtained iteratively using

$$p_n = T p_{n-1},$$

or from

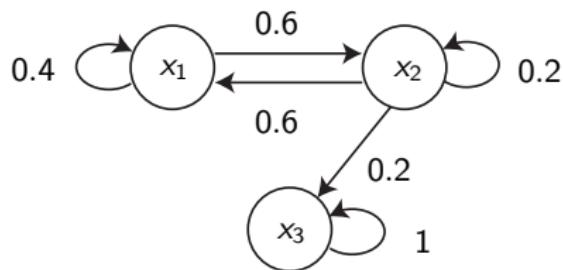
$$p_n = T \dots (T(T p_0)) = T^n p_0. \quad (1)$$

Thus, the probabilities for the previous example become

$$p_1 = T p_0 = \begin{bmatrix} 0 \\ 0.8 \\ 0.2 \end{bmatrix}, \quad p_2 = T p_1 = \begin{bmatrix} 0.2 \\ 0 \\ 0.8 \end{bmatrix}.$$

# Tweedback Questions

Given is the following Markov chain:



- What is the one-step probability for  $p_0 = [1, 0, 0]^T$ ?
  - A  $p_1 = [0.4, 0.6, 0]^T$
  - B  $p_1 = [0.4, 0.4, 0.2]^T$
- What is the probability for  $n \rightarrow \infty$ ?
  - A  $\lim_{n \rightarrow \infty} p_n = [0.4, 0.4, 0.2]^T$
  - B  $\lim_{n \rightarrow \infty} p_n = [0, 0, 1]^T$

# Conversion of Higher-Order Markov Chains

## Higher-order Markov chains

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

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

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:

$$\hat{x}_1 \hat{=} \overbrace{(x_1, \dots, x_1, x_1)}^{m \text{ entries}}$$

$$\hat{x}_2 \hat{=} (x_1, \dots, x_1, x_2)$$

...

$$\hat{x}_d \hat{=} (x_1, \dots, x_1, x_{d_1})$$

$$\hat{x}_{d+1} \hat{=} (x_1, \dots, x_2, x_1)$$

# Conversion of Higher-Order Markov Chains: Example

**Before:**

$$P(X_n = x_1 | X_{n-1} = x_1, X_{n-2} = x_1) = a$$

$$P(X_n = x_1 | X_{n-1} = x_1, X_{n-2} = x_2) = b$$

$$P(X_n = x_1 | X_{n-1} = x_2, X_{n-2} = x_1) = c$$

$$P(X_n = x_1 | X_{n-1} = x_2, X_{n-2} = x_2) = d$$

$$P(X_n = x_2 | X_{n-1} = x_1, X_{n-2} = x_1) = e$$

$$P(X_n = x_2 | X_{n-1} = x_1, X_{n-2} = x_2) = f$$

...

**After:**

$$P(X_n = \hat{x}_1 | X_{n-1} = \hat{x}_1) = a$$

$$P(X_n = \hat{x}_1 | X_{n-1} = \hat{x}_2) = b$$

$$P(X_n = \hat{x}_2 | X_{n-1} = \hat{x}_3) = c$$

$$P(X_n = \hat{x}_2 | X_{n-1} = \hat{x}_4) = d$$

$$P(X_n = \hat{x}_3 | X_{n-1} = \hat{x}_1) = e$$

$$P(X_n = \hat{x}_3 | X_{n-1} = \hat{x}_2) = f$$

...

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

$$\hat{x}_1 \hat{=} (x_1, x_1)$$

$$\hat{x}_2 \hat{=} (x_1, x_2)$$

$$\hat{x}_3 \hat{=} (x_2, x_1)$$

$$\hat{x}_4 \hat{=} (x_2, x_2)$$

# 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:

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

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

$$(p_i)_n = P(X_n = x_i)$$

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

and the matrices

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

$$H_{i,j} = P(E_n = e_i | X_n = x_j).$$

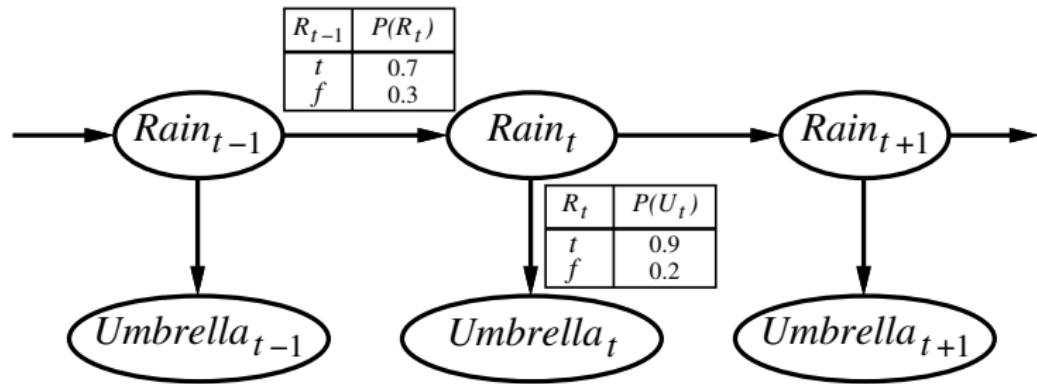
We can compute the probabilities as

$$p_n = T p_{n-1},$$

$$\hat{p}_n = H p_n.$$

# 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  $X_t = \text{Rain}_t$  and the measurement is  $E_t = \text{Umbrella}_t$ .



# Joint Probability Distribution (1)

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

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

**Explanation of formula by example:**

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

# Joint Probability Distribution (2)

**Reminder:**

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

Given are the initial probability distribution  $P(Rain_0 = true) = 0.2$ ,  $P(Rain_0 = false) = 0.8$  and the conditional probabilities from slide 16.

**First iteration ( $t = 1$ ):**

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

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

**Second iteration ( $t = 2$ ):**

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

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

# Inference Tasks in Hidden Markov Models

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

We will see that all algorithms have a time complexity of  $\mathcal{O}(t)$ .

# Filtering (1)

## Aim

Devise a **recursive** state estimation algorithm:

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

Such a recursive algorithm can be obtained as follows:

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

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

## Filtering (2)

Reminder:

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

Prediction by summing out  $X_t$ :

$$\begin{aligned} P(X_{t+1}|e_{1:t+1}) &= \alpha P(e_{t+1}|X_{t+1}) \underbrace{\sum_{x_t} P(X_{t+1}|x_t, e_{1:t})}_{P(X_{t+1}|e_{1:t})} P(x_t|e_{1:t}) \\ &= \alpha P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}|x_t) P(x_t|e_{1:t}). \quad (\text{Markov assumption}) \end{aligned} \tag{2}$$

- $P(e_{t+1}|X_{t+1}) = P(e_t|X_t)$  is directly obtained from the sensor model.
- $P(X_{t+1}|x_t)$  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:

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

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 the filtering algorithm as

$$f_{1:t+1} = \alpha O_{t+1} T f_{1:t},$$

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

# Filtering: Umbrella Example (1)

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

True      0.500

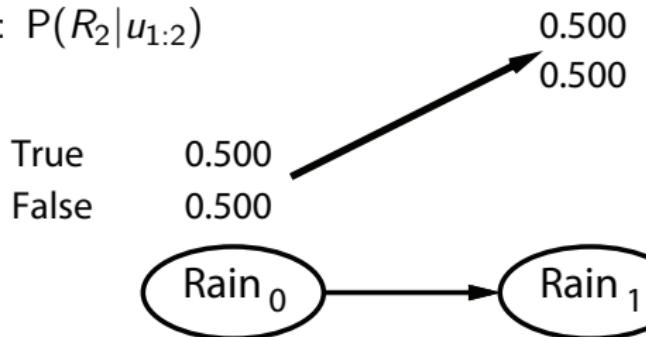
False     0.500



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

## Filtering: Umbrella Example (2)

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



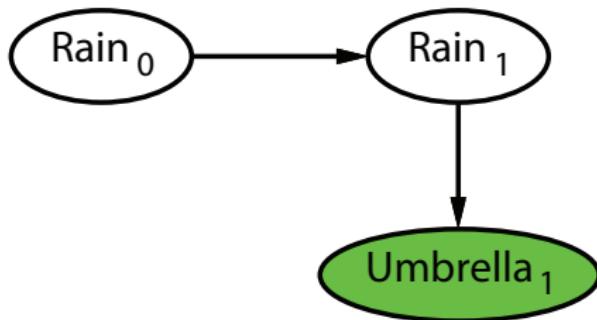
Day 1: The prediction from  $t = 0$  to  $t = 1$  is

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

## Filtering: Umbrella Example (3)

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

		0.500
True	0.500	
False	0.500	

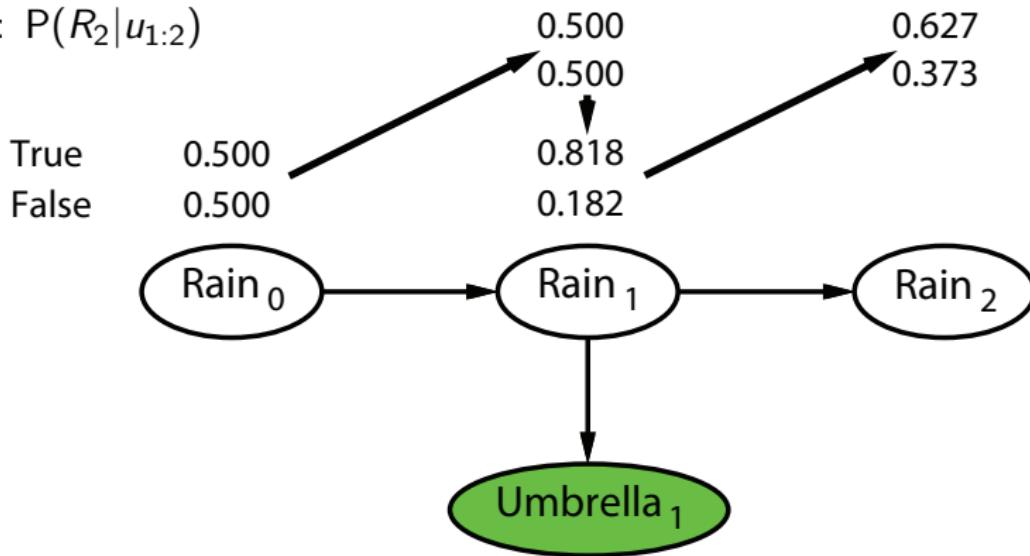


Day 1: The umbrella appears, we incorporate the measurement

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

## Filtering: Umbrella Example (4)

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

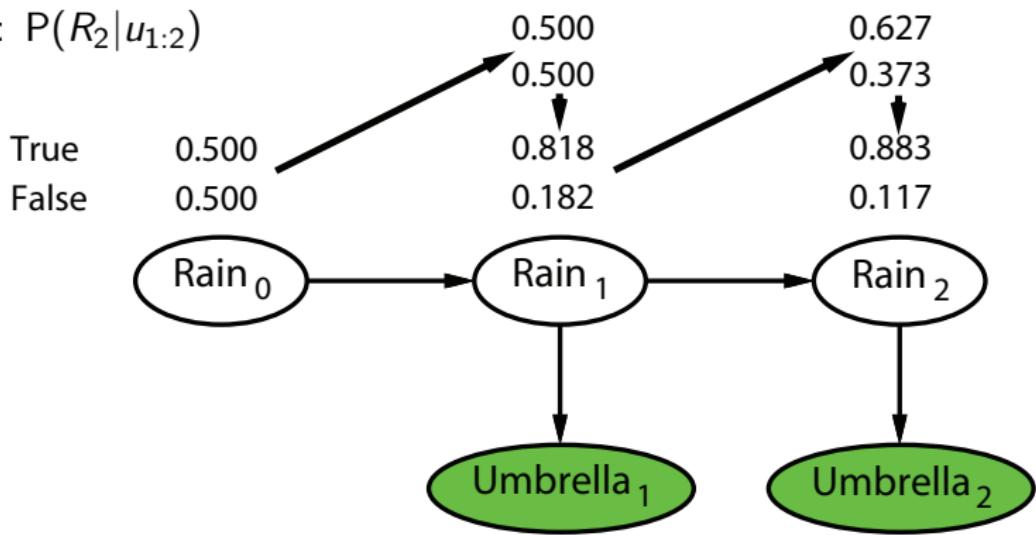


Day 2: The prediction from  $t = 1$  to  $t = 2$  is

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

## Filtering: Umbrella Example (5)

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



Day 2: The umbrella appears, we incorporate the measurement

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

# Filtering: Umbrella Example in Matrix Notation

Observation matrices:

$$O_1 = O_2 = \begin{bmatrix} 0.9 & 0 \\ 0 & 0.2 \end{bmatrix}.$$

Transition matrix:

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

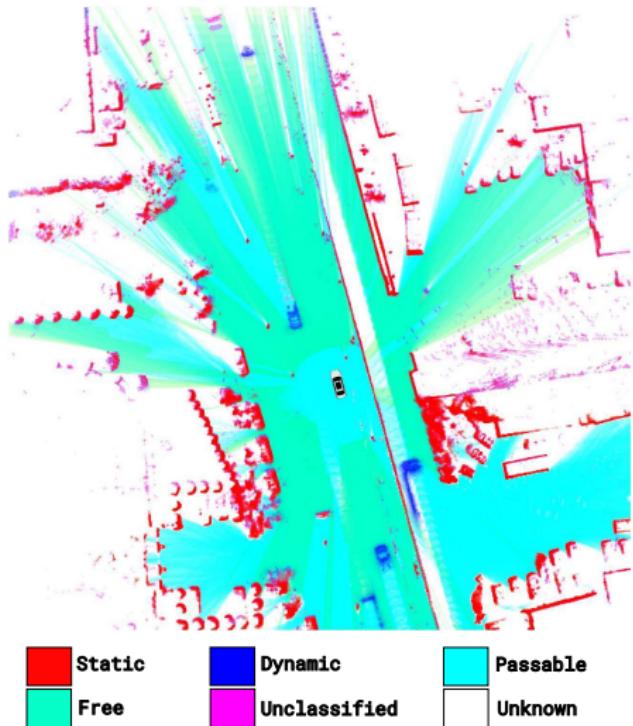
$$f_{1:t+1} = \alpha O_{t+1} T f_{1:t}$$

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

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

# Filtering Example: Dynamic Grid Generation<sup>1</sup>

- Problem: Estimating the occupancy probability of a grid.
- The actual occupancy state is hidden, we only have LiDAR sensor measurements as observations.



<sup>1</sup>Ye, Egon, et al. "Offline Dynamic Grid Generation for Automotive Environment Perception Using Temporal Inference Methods." IEEE Robotics and Automation Letters 6.3 (2021): 5501-5508.

# Prediction

- 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

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

## Comments:

- As  $k \rightarrow \infty$ ,  $P(x_{t+k}|e_{1:t})$  tends to the **stationary distribution** of the included Markov chain, where  $p_{t+k} = T^k p_t$  (see slide 10).
- 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 16, we have:

$$p_0 = \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}, \quad T = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}, \quad H = \begin{bmatrix} 0.9 & 0.2 \\ 0.1 & 0.8 \end{bmatrix}.$$

## Prediction: Umbrella Example (2)

Using (1) on slide 10, one obtains

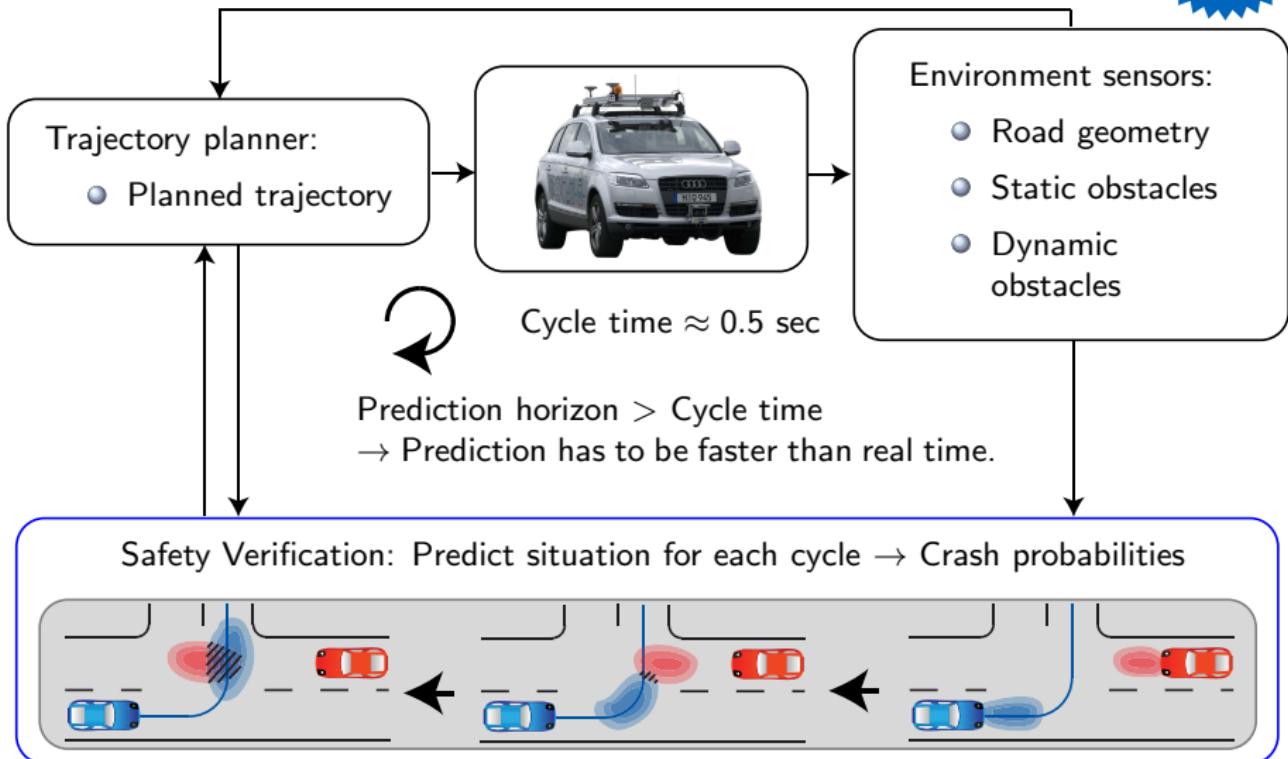
$$\begin{aligned} p_n &= T^n p_0, \\ \hat{p}_n &= H p_n. \end{aligned}$$

### Umbrella example

$$\begin{aligned} p_0 &= \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}, & T^{100} &= \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}, & H &= \begin{bmatrix} 0.9 & 0.2 \\ 0.1 & 0.8 \end{bmatrix}, \\ p_{100} &= \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, & \hat{p}_{100} &= \begin{bmatrix} 0.55 \\ 0.45 \end{bmatrix}. \end{aligned}$$

# Example: Safety Assessment of Autonomous Cars

Not relevant for the exam

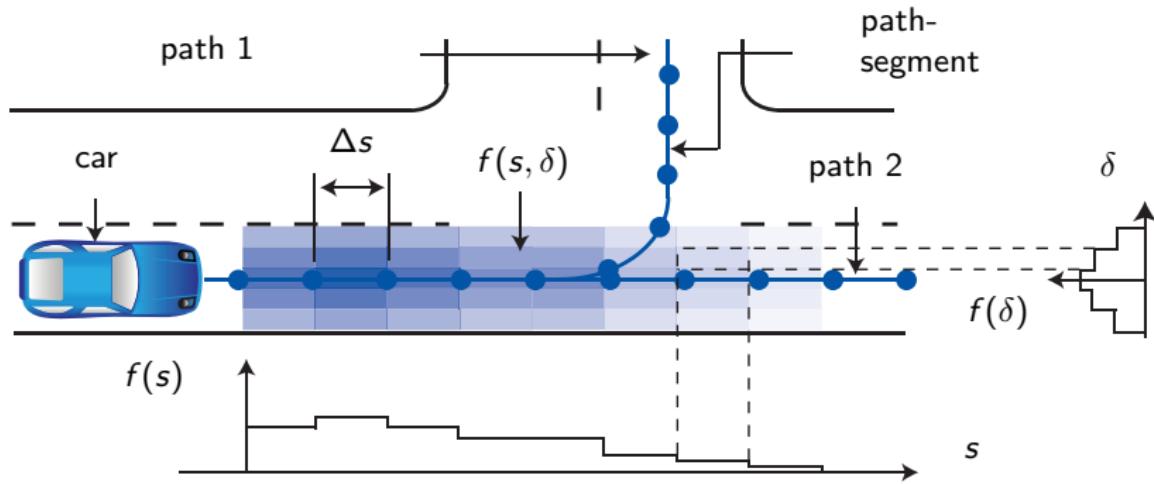


# Modeling of Other Traffic Participants

Not relevant for  
the exam

## Structured environments

- Vehicles follow preferred paths such as traffic lanes.
- Longitudinal distribution  $f(s)$  from hidden Markov model, lateral distribution  $f(\delta)$  is static.

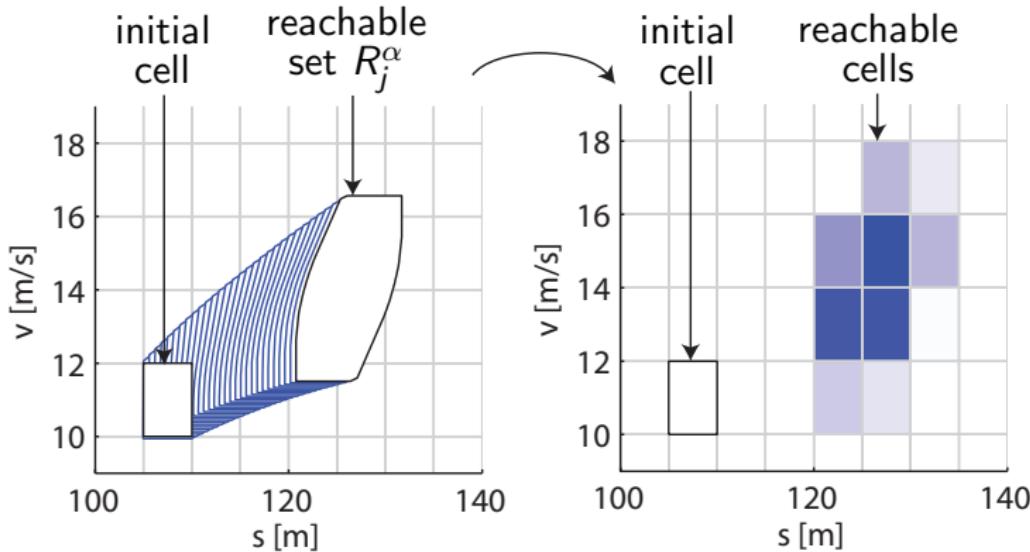


# Abstraction to Hidden Markov Models via Reachability Analysis

Not relevant for the exam

Prediction:  $p_n = T p_{n-1}$ .

Computation of the transition matrix  $T$  via reachability analysis.

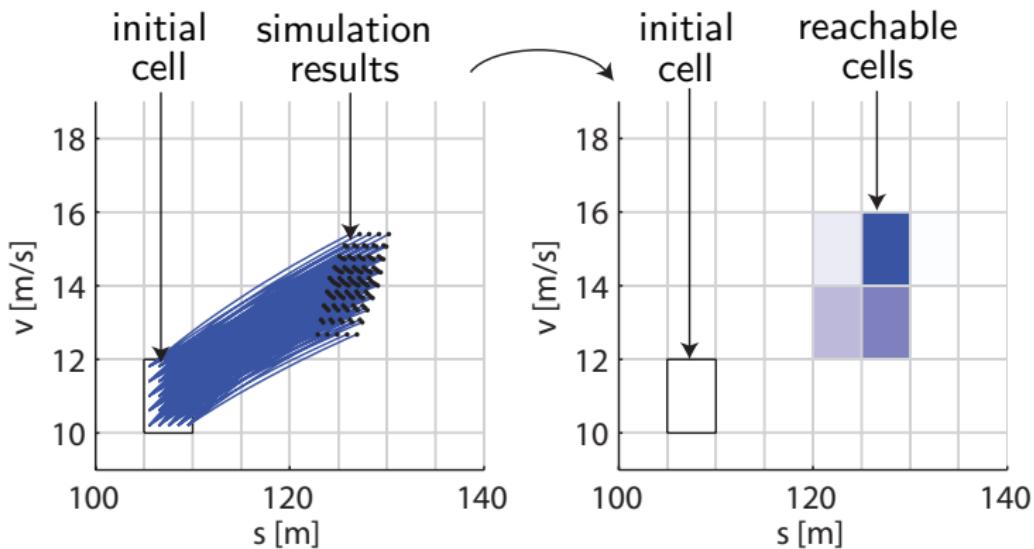


# Abstraction to Hidden Markov Models via Simulation Techniques

Not relevant for the exam

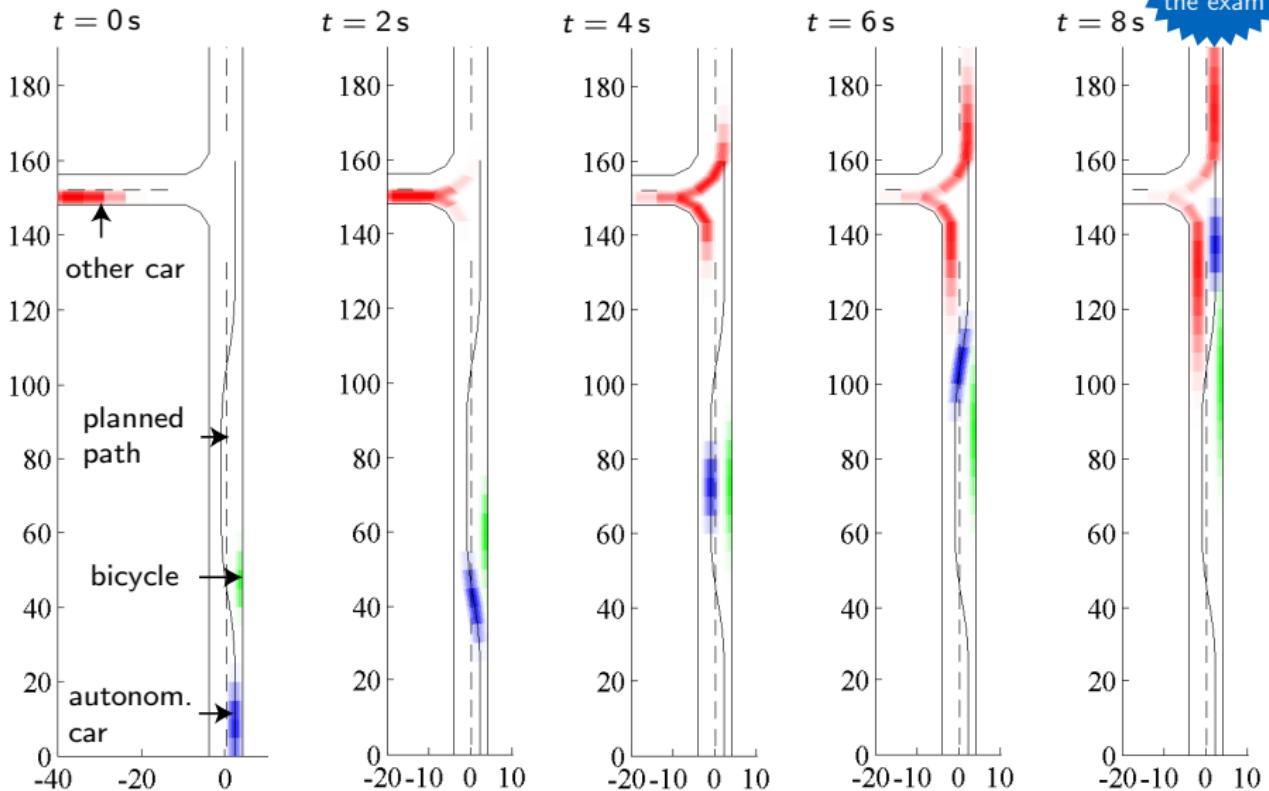
Prediction:  $p_n = T p_{n-1}$ .

Computation of the transition matrix  $T$  via simulation techniques.



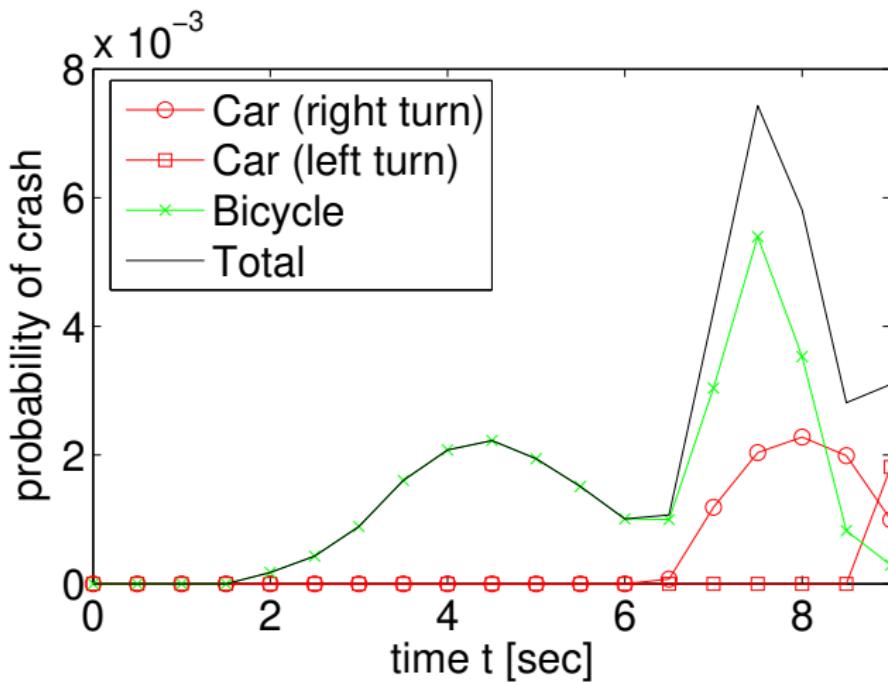
# Overtaking Scenario

Not relevant for  
the exam

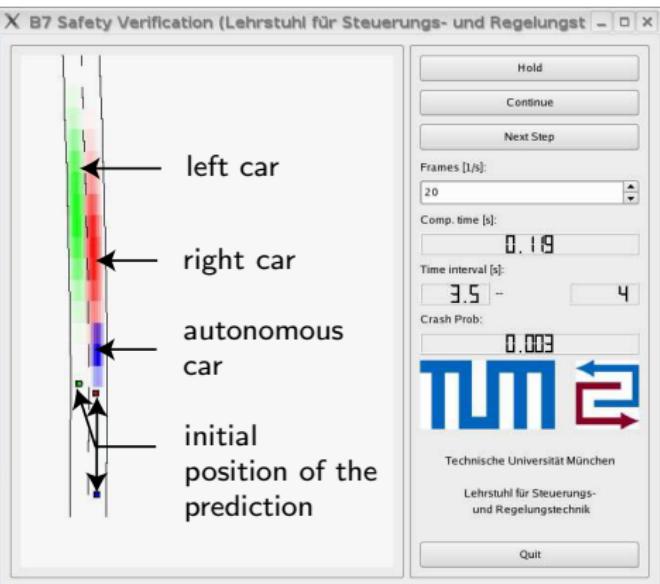


# Crash Probability

Not relevant for the exam

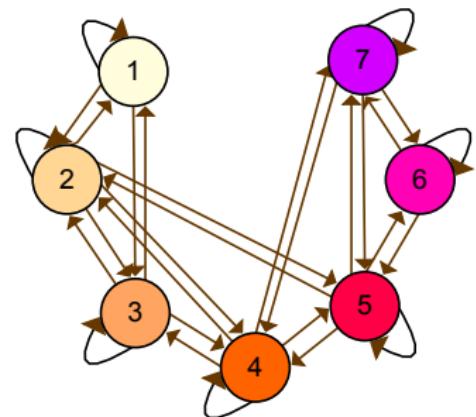


# Test Drive



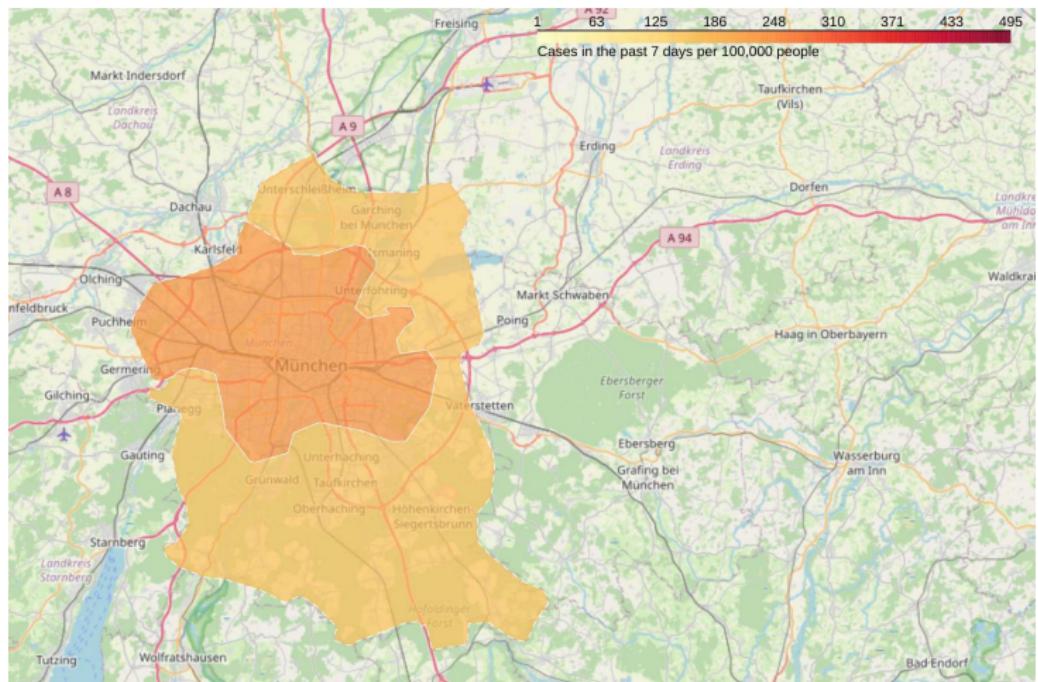
# Prediction Example: Spatio-Temporal Prediction of COVID-19<sup>2</sup>

- In order to control COVID-19 and future pandemics, we need an understanding of the disease's evolution as well as the effectiveness of intervention methods.
- Our observations are the reported cases and deaths in several countries and regions.
- An HMM with seven (hidden) severity states is used to predict the severity level of different regions over a fixed time period.



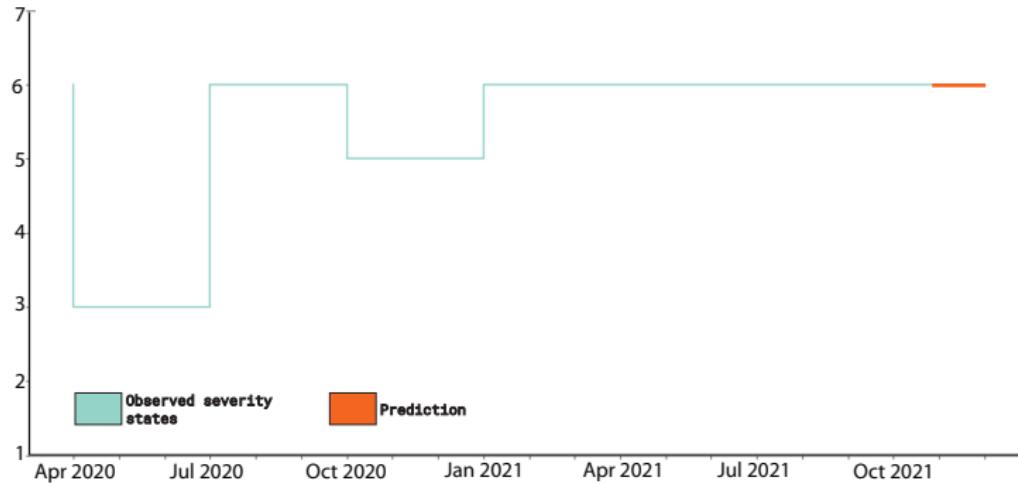
<sup>2</sup>S. Zhou et al., "Application of Hidden Markov Models to Analyze, Group and Visualize Spatio-Temporal COVID-19 Data," in IEEE Access, vol. 9, pp. 134384-134401, 2021

# Prediction Example: COVID-19 Cases per 100.000 in the last 7 Days in the Munich Region, 09.12.2020

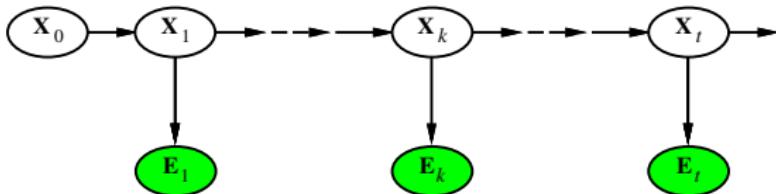


# Prediction Example: COVID-19 Severity Prediction in Munich

In the HMM exercise of 2021/22, we predicted the severity states in November and December 2021. The result, state 6, was actually correct.



# Smoothing (1)



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

$$\begin{aligned}
 P(X_k|e_{1:t}) &= P(X_k|e_{1:k}, e_{k+1:t}) \\
 &= \alpha' P(X_k, e_{1:k}, e_{k+1:t}) \quad (\text{normalization}) \\
 &= \alpha' P(e_{k+1:t}|X_k, e_{1:k}) P(X_k|e_{1:k}) P(e_{1:k}) \quad (\text{using chain rule}) \\
 &= \alpha P(X_k|e_{1:k}) P(e_{k+1:t}|X_k, e_{1:k}) \quad (\text{remove } P(e_{1:k})) \\
 &= \alpha P(X_k|e_{1:k}) P(e_{k+1:t}|X_k) \quad (\text{using conditional independence}) \\
 &= \alpha f_{1:k} \times b_{k+1:t} \quad (f: \text{forward}; b: \text{backward}) \tag{3}
 \end{aligned}$$

## Smoothing (2)

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

- The forward factor  $f_{1:k}$  is computed as for the filtering on slide 21.
- The backward factor  $b_{k+1:t}$  is obtained by the following recursion:

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

## Smoothing (3)

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

- $P(e_{k+1}|x_{k+1})$  is the sensor model, see slide 14.
- $P(x_{k+1}|X_k)$  is the transition probability, see slide 9.
- $P(e_{k+2:t}|x_{k+1})$  is obtained by recursive execution of the above formula backwards in time.
- The backwards recursion is denoted by

$$b_{k+1:t} = \text{Backward}(b_{k+2:t}, e_{k+1}),$$

which implements (4).

- The forward recursion is denoted by

$$f_{1:t+1} = \alpha \text{Forward}(f_{1:t}, e_{t+1}),$$

which implements (2) on slide 21.

- The backward phase is initialized by  $b_{t+1:t} = P(e_{t+1:t}|X_t) = P(\cdot|X_t) = 1$ , where 1 is a vector of ones (probability of observing future evidence is 0).

# Smoothing: Matrix Notation

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

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

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

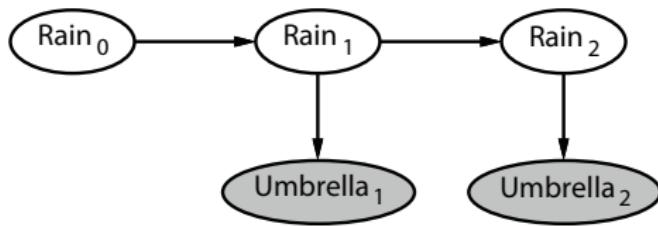
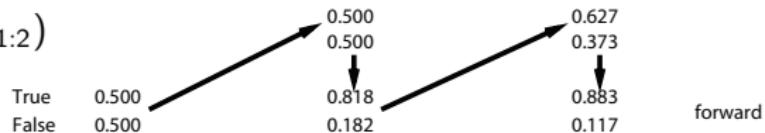
This makes it possible to write (4) as

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

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)

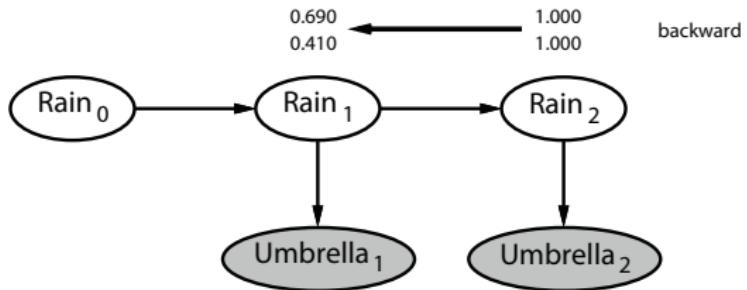
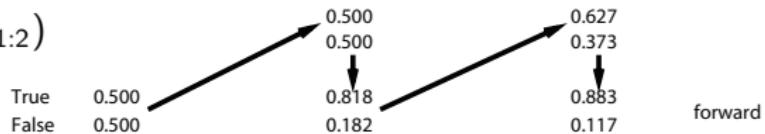
**Query:**  $P(R_1|u_{1:2})$



From (3) on slide 43, we have  $P(R_1|u_{1:2}) = \alpha P(R_1|u_1)P(u_2|R_1)$ .  $P(R_1|u_1)$  we already know from forward filtering to be  $\langle 0.818, 0.182 \rangle$ .

# Smoothing: Umbrella Example (2)

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

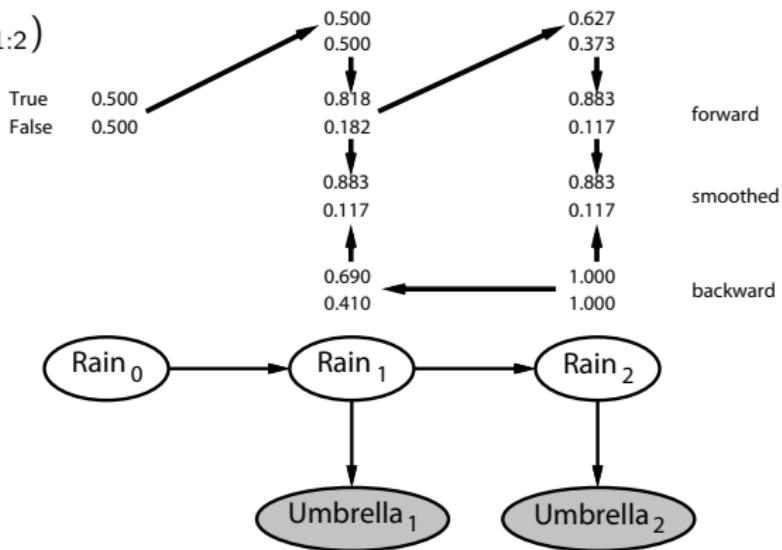


The second term can be computed by the backward recursion in eq. (4):

$$\begin{aligned}
 P(u_2|R_1) &= \sum_{r_2} P(u_2|r_2)P(\quad|r_2)P(r_2|R_1) \\
 &= (0.9 \times 1 \times \langle 0.7, 0.3 \rangle) + (0.2 \times 1 \times \langle 0.3, 0.7 \rangle) = \langle 0.69, 0.41 \rangle.
 \end{aligned}$$

# Smoothing: Umbrella Example (3)

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



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

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

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:

$$O_1 = O_2 = \begin{bmatrix} 0.9 & 0 \\ 0 & 0.2 \end{bmatrix}.$$

Transition matrix:

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

$$b_{k+1:t} = T^T O_{k+1} b_{k+2:t}$$

$$b_{3:2} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$b_{2:2} = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}^T \begin{bmatrix} 0.9 & 0 \\ 0 & 0.2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.69 \\ 0.41 \end{bmatrix}$$

# Forward-Backward Algorithm

(  HMM.ipynb)

**function** Forward-Backward (ev, prior) **returns** a vector of probability distributions

**inputs:** ev, a vector of evidence values for steps  $1, \dots, t$

prior, the prior distribution on the initial state,  $P(X_0)$

**local variables:** fv, a vector of forward messages for steps  $0, \dots, t$   
 b, backward message representation, initially all ones  
 sv, a vector of smoothed estimates for steps  $1, \dots, t$

$f v[0] \leftarrow \text{prior}$

**for**  $i = 1$  **to**  $t$  **do**

$f v[i] \leftarrow \text{Forward}(f v[i - 1], e v[i])$  (see eq. (2) on slide 21)

**for**  $i = t$  **downto** 1 **do**

$s v[i] \leftarrow \text{Normalize}(f v[i] \times b)$

$b \leftarrow \text{Backward}(b, e v[i])$  (see eq. (4) on slide 45)

**return** sv

Unlike filtering, the space complexity is  $\mathcal{O}(t)$ .

# Munich Motion Dataset of Natural Driving (MONA)

- Recordings of five consecutive days.
- Three different perspectives, stationary camera setup.
- 130h of 4k@25fps video material.



Creator: Maximilian Dörrbecker CC BY-SA 2.5

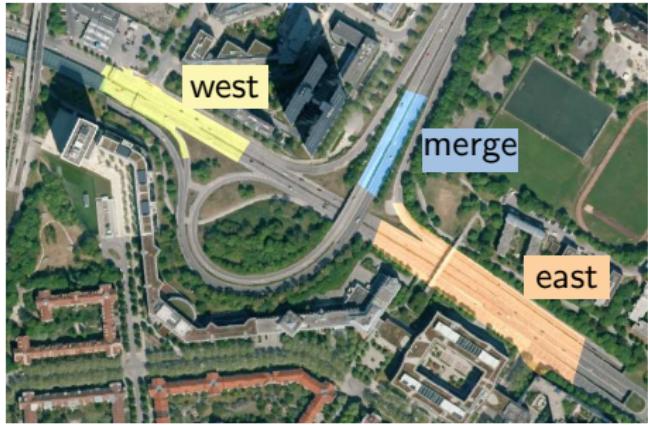
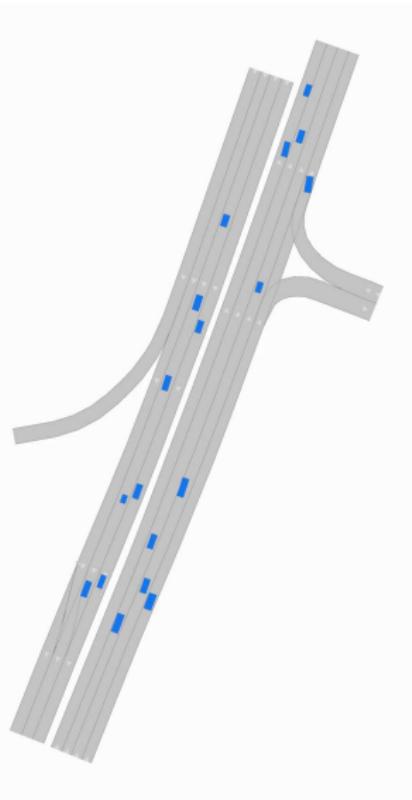


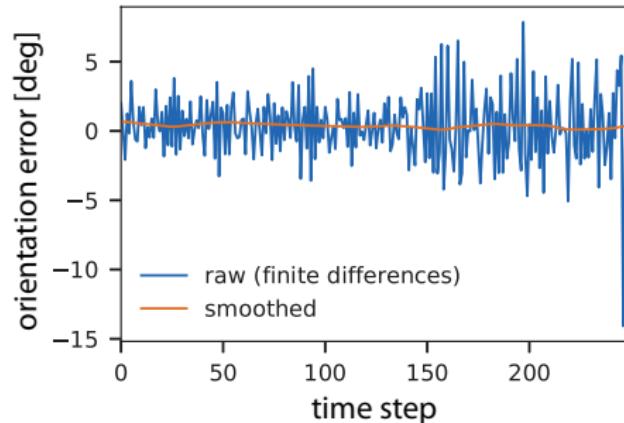
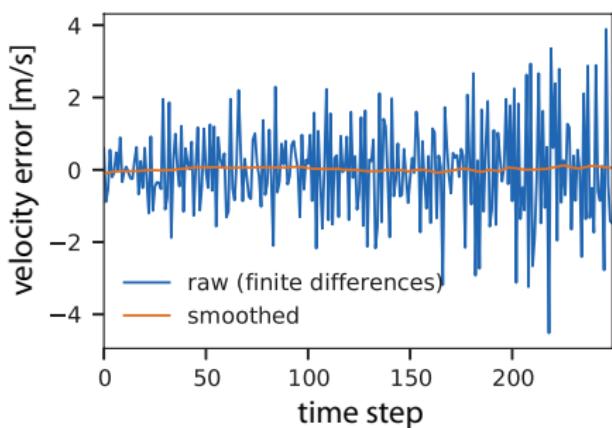
Image source: Bayr. Vermessungsverwaltung (CC BY 3.0 DE).

# MONA Dataset: West

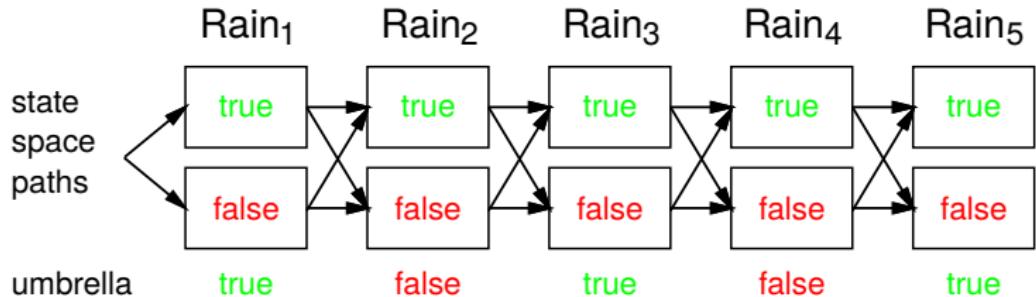


# Detection Error Reduction through Smoothing

- Smoothing effectively reduces the error in orientation and velocity of detected vehicles.
- The ground truth was obtained by an autonomous vehicle.
- Rauch-Tung-Striebel smoothing (continuous version of the presented technique) was used due to the continuous nature of the data.

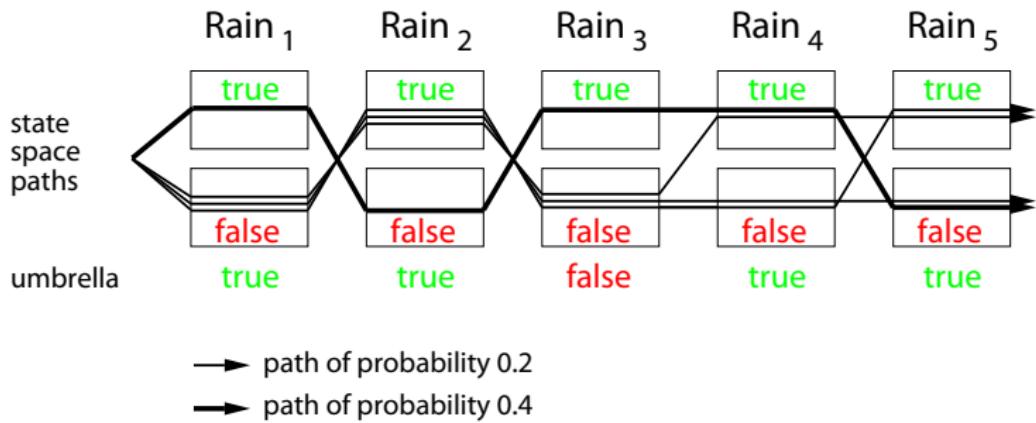


# Most Likely Explanation



- 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!**

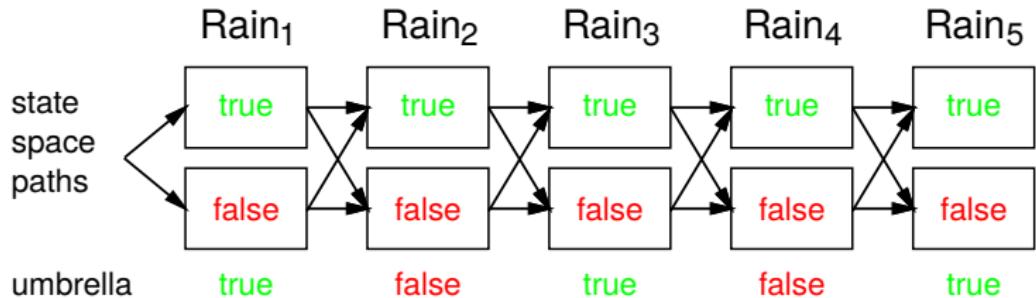
# Example: Most Likely Sequence $\neq$ Sequence of Most Likely States



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  $Rain_5 = \text{true}$ .
- Because of the Markov property, the most likely path to the state  $Rain_5 = \text{true}$  consists of
  - the most likely path to **some** state at time 4
  - followed by a transition to  $Rain_5 = \text{true}$ .
- The state at time 4 becomes a part of the most likely path to  $Rain_5 = \text{true}$ .

# 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:

$$\begin{aligned}
 \max_{x_1 \dots x_t} P(x_1, \dots, x_t, X_{t+1} | e_{1:t+1}) &= \max_{x_1 \dots x_t} P(x_1, \dots, x_t, X_{t+1} | e_{1:t}, e_{t+1}) \\
 &= \max_{x_1 \dots x_t} \alpha P(x_1, \dots, x_t, X_{t+1}, e_{t+1} | e_{1:t}) \quad (\text{normalization}) \\
 &= \max_{x_1 \dots x_t} \alpha P(e_{t+1} | x_1, \dots, x_t, X_{t+1}, e_{1:t}) P(x_1, \dots, x_t, X_{t+1} | e_{1:t}) \quad (P(a, b|c) = \frac{P(a, b, c)}{P(a, c)} P(a, c) = P(b|a, c)P(a|c)) \\
 &= \alpha P(e_{t+1} | X_{t+1}) \max_{x_1 \dots x_t} P(x_1, \dots, x_t, X_{t+1} | e_{1:t}) \quad (\text{conditional independence}) \\
 &= \alpha P(e_{t+1} | X_{t+1}) \max_{x_t} \left( P(X_{t+1} | x_t) \max_{x_1 \dots x_{t-1}} P(x_1, \dots, x_{t-1}, x_t | e_{1:t}) \right)
 \end{aligned}$$

- The algorithm is called **Viterbi algorithm** and is similar in structure compared to the filtering procedure in eq. (2) on slide 21.
- 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

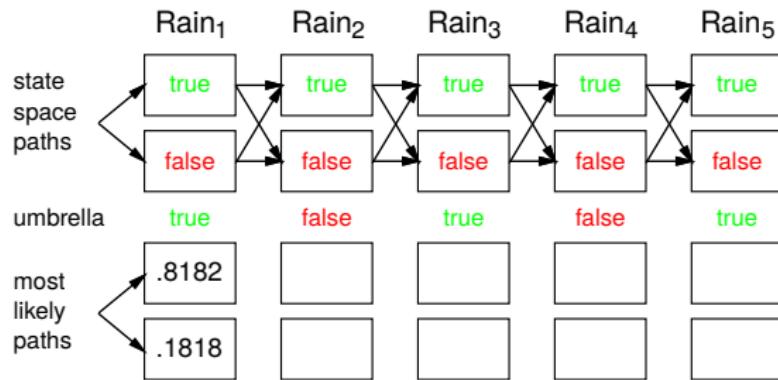
The Viterbi algorithm requires the evaluation of the max-operator:

$$\begin{aligned} & \max_{x_1 \dots x_t} P(x_1, \dots, x_t, X_{t+1} | e_{1:t+1}) \\ &= \alpha P(e_{t+1} | X_{t+1}) \max_{x_t} \left( P(X_{t+1} | x_t) \max_{x_1 \dots x_{t-1}} P(x_1, \dots, x_{t-1}, x_t | e_{1:t}) \right). \end{aligned}$$

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

$$\begin{aligned} & \max_{x_t} P(X_{t+1} | x_t) = \\ & \left\langle \max_{x_t} P(X_{t+1} = \text{true} | x_t), \max_{x_t} P(X_{t+1} = \text{false} | x_t) \right\rangle. \end{aligned}$$

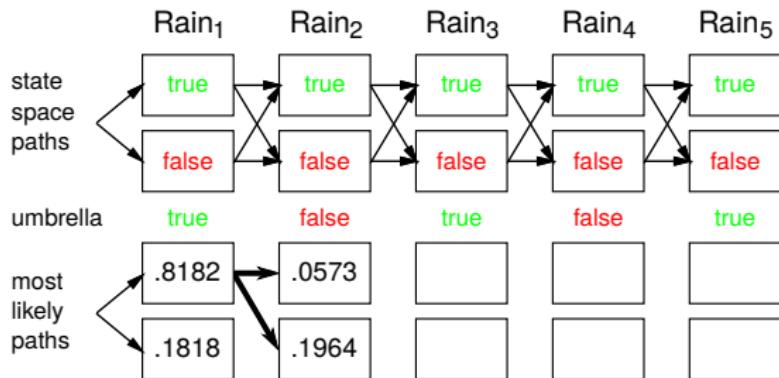
# Most Likely Explanation: Umbrella Example (1)



Initialization with filtering:

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

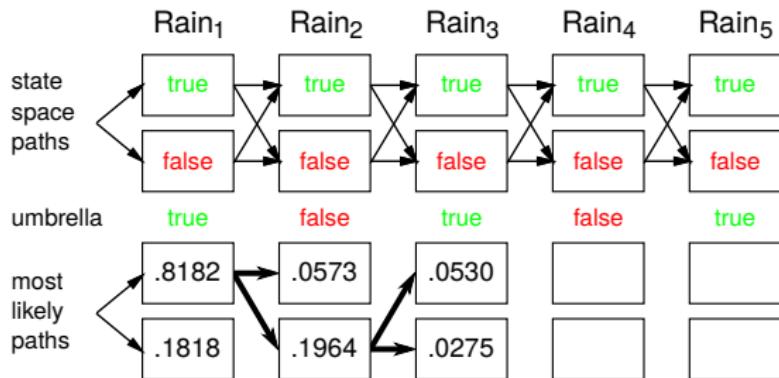
# Most Likely Explanation: Umbrella Example (2)



We omit the optional normalization from now on.

- $r_1 = \text{true} : P(R_2|r_1)P(r_1|u_{1:1}) = \langle 0.7, 0.3 \rangle \times 0.8182 = \langle 0.5727, 0.2455 \rangle$
- $r_1 = \text{false} : P(R_2|r_1)P(r_1|u_{1:1}) = \langle 0.3, 0.7 \rangle \times 0.1818 = \langle 0.0545, 0.1273 \rangle$
- $\max_{r_1} P(r_1, R_2|u_{1:2}) = P(u_2|R_2) \max_{r_1} (P(R_2|r_1)P(r_1|u_{1:1})) = \langle 0.1, 0.8 \rangle \times \langle 0.5727, 0.2455 \rangle = \langle 0.0573, 0.1964 \rangle$

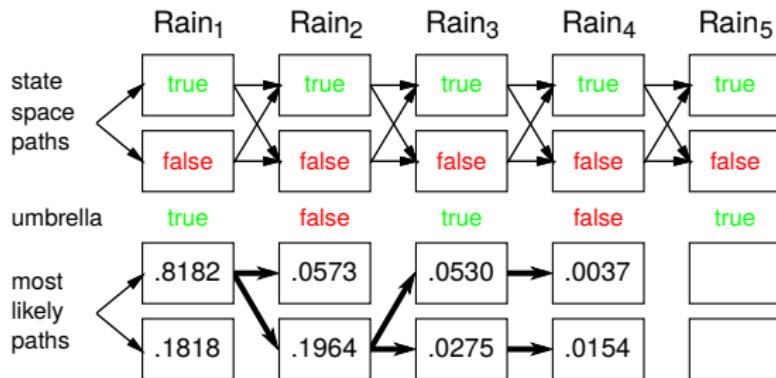
# Most Likely Explanation: Umbrella Example (3)



We omit the optional normalization from now on.

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

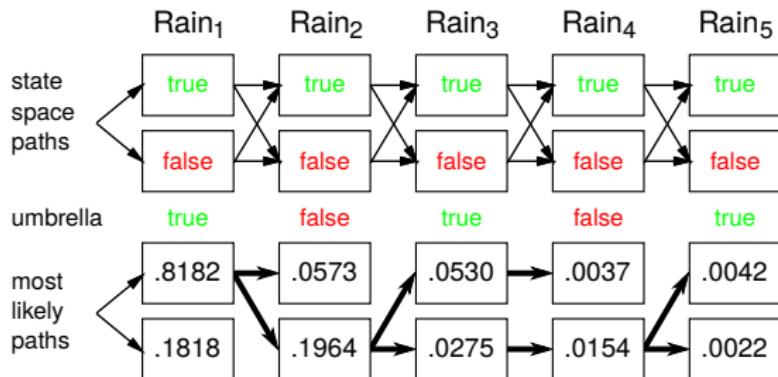
# Most Likely Explanation: Umbrella Example (4)



We omit the optional normalization from now on.

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

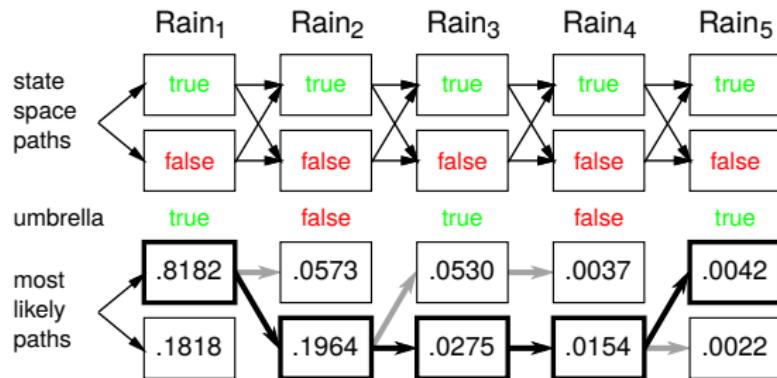
# Most Likely Explanation: Umbrella Example (5)



We omit the optional normalization from now on.

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

# 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

# Viterbi Example: Most Likely Sequence of Animal Movement States

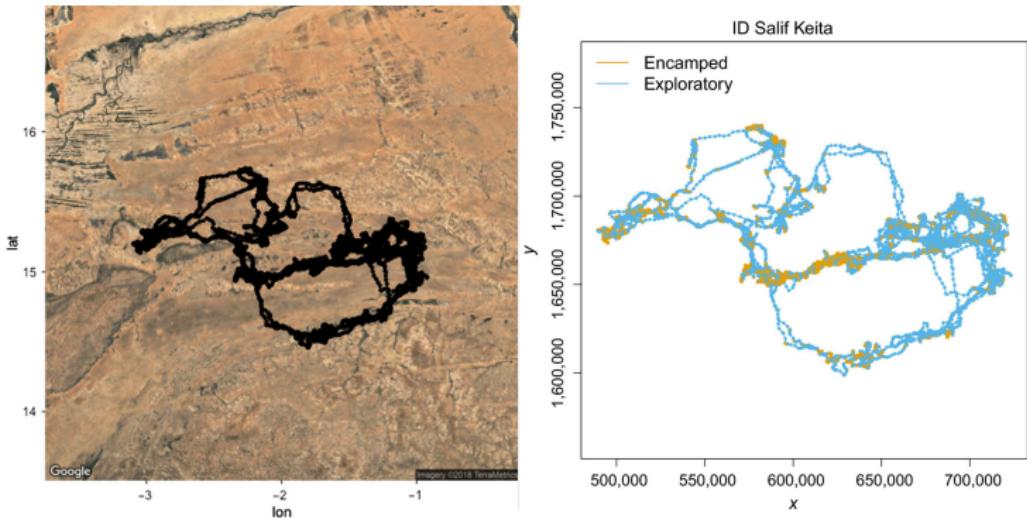
- State sequences are used to detect changes in animal behavior.
- `momentuHMM`<sup>3</sup> is an R package for generalized hidden Markov models of animal movement.
- An African elephant bull's step length and the outside temperature were tracked.



Creator: Rolf Dobberstein

<sup>3</sup> McClintock, B. T., & Michelot, T. (2018). momentuHMM: R package for generalized hidden Markov models of animal movement. *Methods in Ecology and Evolution*, 9(6), 1518-1530.

# Viterbi Example: Movement States of Elephant Bull



- A state of slow undirected movement "**encamped**", and a state of faster and more directed movement "**exploratory**" were identified.
- Step lengths
  - decreased for the "**encamped**" state as temperature increased, and
  - decreased in the late evening and early morning for both states.

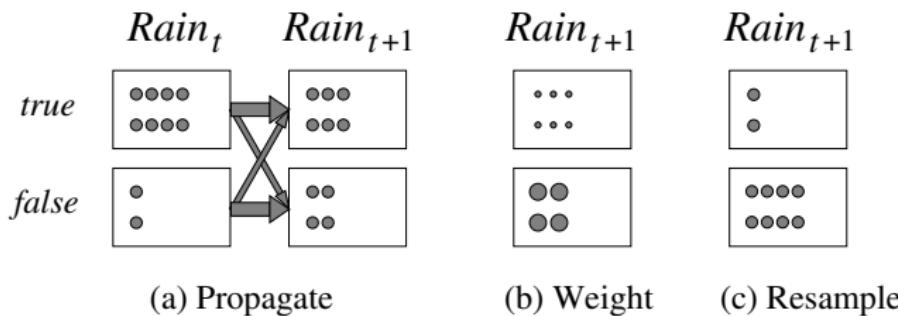
# Estimation of Continuous State Variables



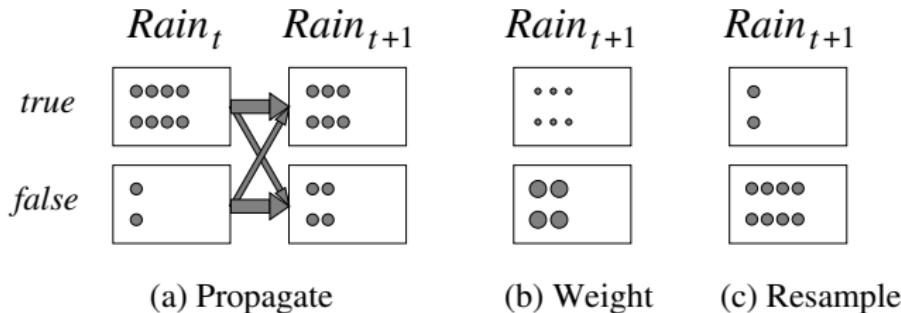
- So far we have only considered estimation of hidden Markov models, which have discrete states.
- Many real-world problems have continuous states, such as position, velocities, forces, temperatures, etc.
- Those problems arise e.g., in
  - robotics,
  - automated driving,
  - smart grids,
  - automated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

# Particle Filtering (1)

- 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.



# Particle Filtering (2)



- 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  $t + 1$ .
- $Umbrella = false$  is observed at  $t + 1$ . Each sample is weighted by its likelihood for the observation, as indicated by the size of the circles.
- 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  $t$ :  $N(x_t|e_{1:t})/N = P(x_t|e_{1:t})$ . (5)
- Propagate forward: populations of  $x_{t+1}$  are

$$N(x_{t+1}|e_{1:t}) = \sum_{x_t} P(x_{t+1}|x_t) N(x_t|e_{1:t}). \quad (6)$$

- Weight samples by their likelihood for  $e_{t+1}$ :

$$W(x_{t+1}|e_{1:t+1}) = P(e_{t+1}|x_{t+1}) N(x_{t+1}|e_{1:t}). \quad (7)$$

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

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

# Example: Robot Localization

A particular instance of filtering is **localization**.

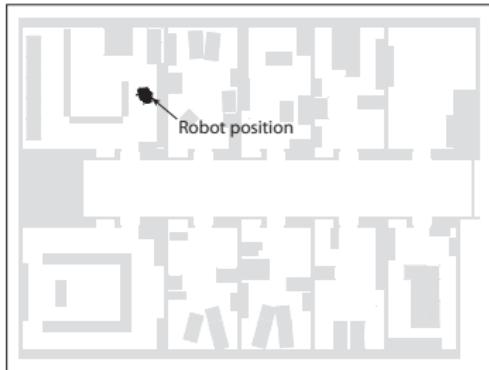
Not relevant for the exam

## Localization

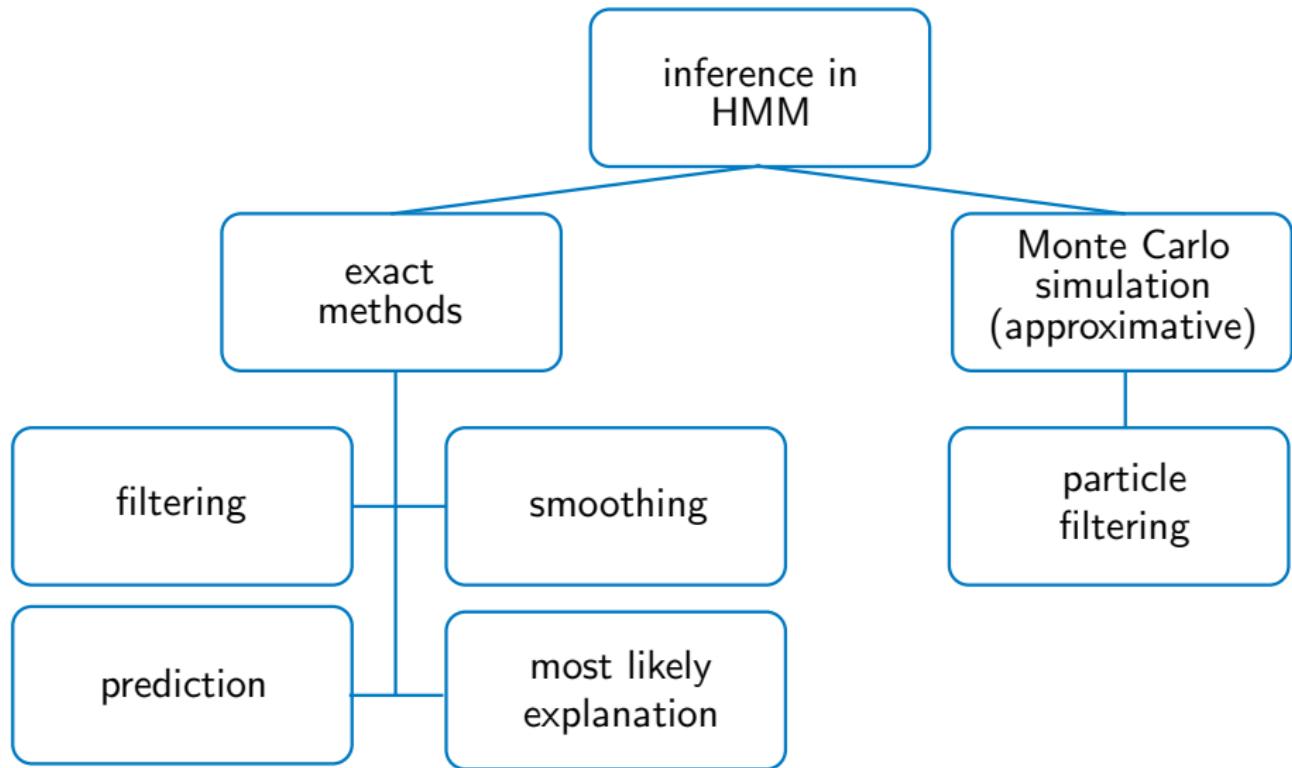
Localization is the problem of finding out where things are – including the robot itself.

To keep things simple, we assume that

- the robot moves slowly on a flat 2D surface,
- the robot knows the exact map of the environment (see figure below).



# Overview of HMM Inference Methods



# Summary

- Temporal models use state and sensor variables replicated over time.
- Markov assumptions and stationarity assumption, so we only need
  - transition model  $P(X_t|X_{t-1})$ ,
  - sensor model  $P(E_t|X_t)$ .
- 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.