
Artificial Intelligence

V. Machine Learning

4. Markov Models

F. C. Langbein

School of Computer Science
and Informatics
Cardiff University





Overview

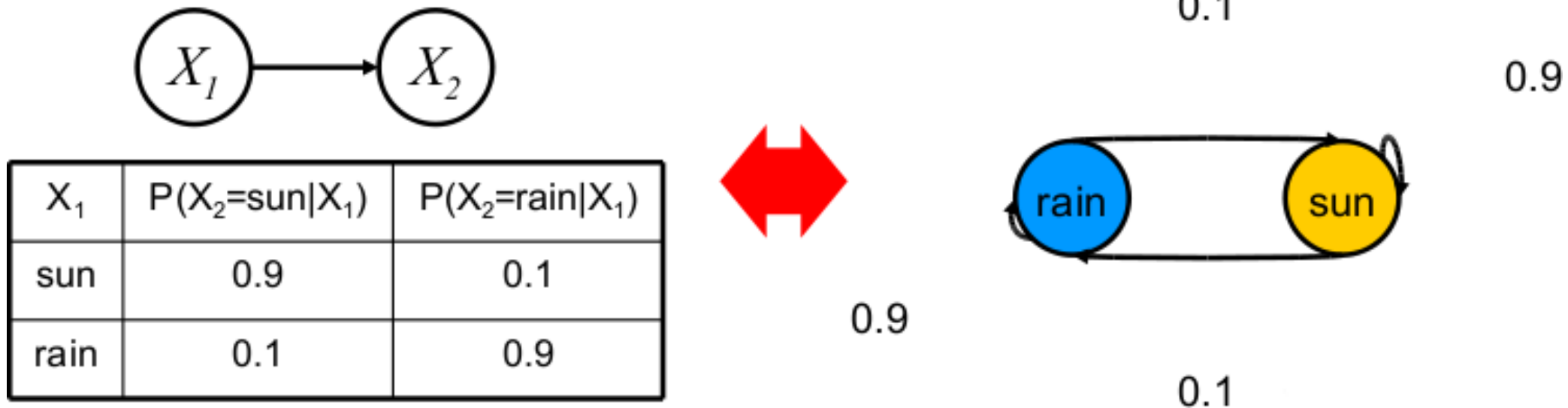
- Markov models
 - Mini forward algorithm
 - Mini viterbi algorithm
- Hidden Markov models
 - Forward algorithm
 - Viterbi algorithm

Markov Models

- Reason about a sequence of observations over time
- **Markov model**: chain-structured Bayesian network
 - Each node is identically distributed
(stationarity or more general parameter-tying)
 - Value of X at a given time is called the *state*
$$X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4 \rightarrow \dots$$
 - With *transition probabilities*
 - specifying how the state evolves over time, $\mathbf{P}(X_t|X_{t-1})$
 - and initial probability, $\mathbf{P}(X_1)$
- Conditional independence: **Markov property**
 - Past and future independent of the presence
 - ➔ Current state contains everything relevant from the past

Weather Example

- States $X_1 = \{\text{rain}, \text{sun}\}$
- Two equivalent ways to represent transition function



- If it is sunny today, what is the weather like tomorrow?

- $P(X_1 = \text{sun}) = 1$
- $Pb(X_2|X_1 = \text{sun}) = \langle 0.9 \cdot 1.0 + 0.1 \cdot 0.0, 0.1 \cdot 1.0 + 0.9 \cdot 0.0 \rangle = \langle 0.9, 0.1 \rangle$

Mini-Forward Algorithm

- What is the probability of being in state x at time t ?
- Inefficient answer
 - Enumerate all sequences of length t that end in x
 - Compute the probability of each (multiply) and add them
- More efficient: cached incremental belief update

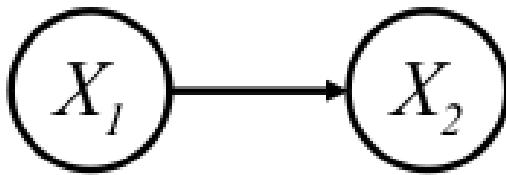
$$P(x_1) = \text{known}$$

$$P(X_t) = \sum_{x_1, \dots, x_{t-1}} P(x_1, \dots, x_{t-1}, X_t)$$

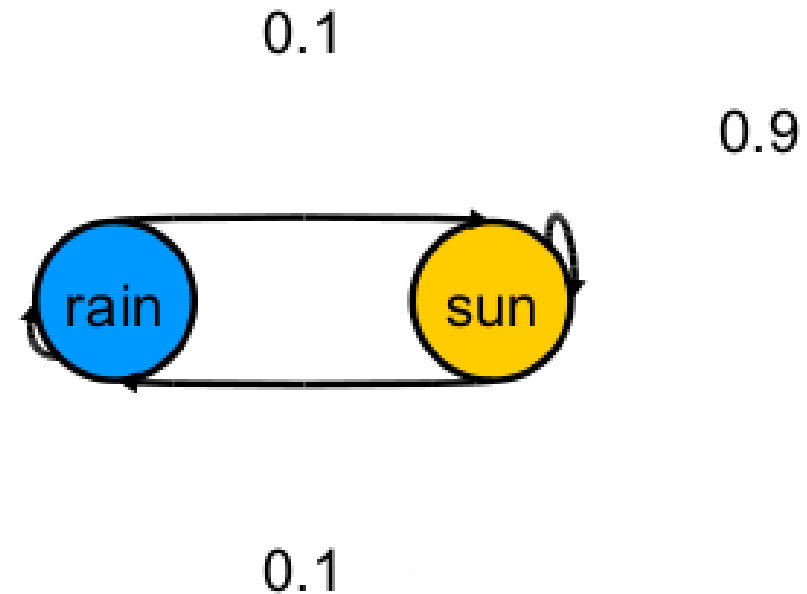
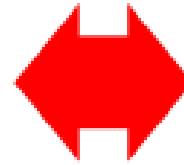
$$= \sum_{x_1, \dots, x_{t-1}} P(X_t | x_{t-1}) P(x_1, \dots, x_{t-1}) = \sum_{x_{t-1}} P(X_t | x_{t-1}) \sum_{x_1, \dots, x_{t-1}} P(x_1, \dots, x_{t-1})$$

$$= \sum_{x_{t-1}} P(X_t | x_{t-1}) P(x_{t-1})$$

Weather Example



X_1	$P(X_2=\text{sun} X_1)$	$P(X_2=\text{rain} X_1)$
sun	0.9	0.1
rain	0.1	0.9



➤ Initial observation of sun:

$\langle 1.0, 0.0 \rangle$

$\rightarrow \langle 0.9 \cdot 1.0 + 0.1 \cdot 0.0, 0.1 \cdot 1.0 + 0.9 \cdot 0.0 \rangle = \langle 0.9, 0.1 \rangle$

$\rightarrow \langle 0.9 \cdot 0.9 + 0.1 \cdot 0.1, 0.1 \cdot 0.9 + 0.9 \cdot 0.1 \rangle = \langle 0.82, 0.18 \rangle$

$\rightarrow \langle 0.9 \cdot 0.82 + 0.1 \cdot 0.18, 0.1 \cdot 0.82 + 0.9 \cdot 0.18 \rangle = \langle 0.76, 0.24 \rangle$

$\rightarrow \dots \rightarrow \langle 0.5, 0.5 \rangle$

Stationary Distributions

- Simulate the chain long enough: uncertainty accumulates
 - Eventually we do not know what the state is
 - Usually we can only predict for a short time ahead
 - ➔ For most chains, resulting distribution is independent of initial distribution: **stationary distribution**
- E.g. Google 1.0 PageRank via web page link graph
 - Web page is a state, initial distribution uniform
 - Transition probabilities:
 - c to uniform jump to random page
 - $1 - c$ to follow a random outlink
 - ➔ Stationary distribution gives highly linked pages

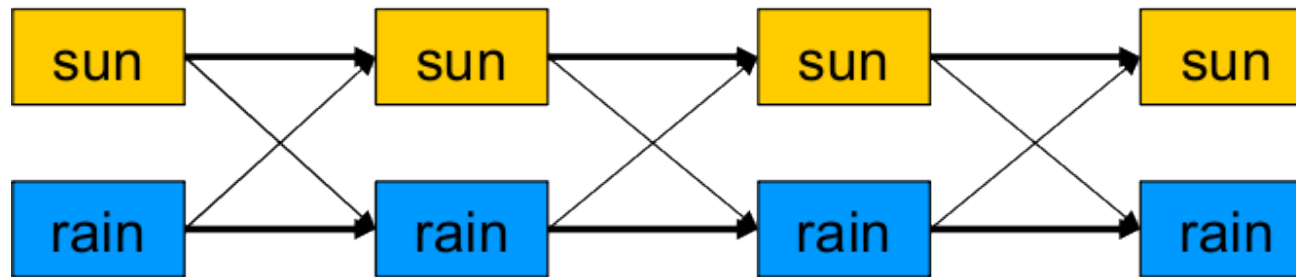
Mini-Viterbi Algorithm

➤ Most likely explanation:

What is the most likely sequence ending in x at t ?

● Find $\max_{x_1, \dots, x_{t-1}} P(x_1, \dots, x_{t-1}, x_t)$

● Evaluating all paths is expensive



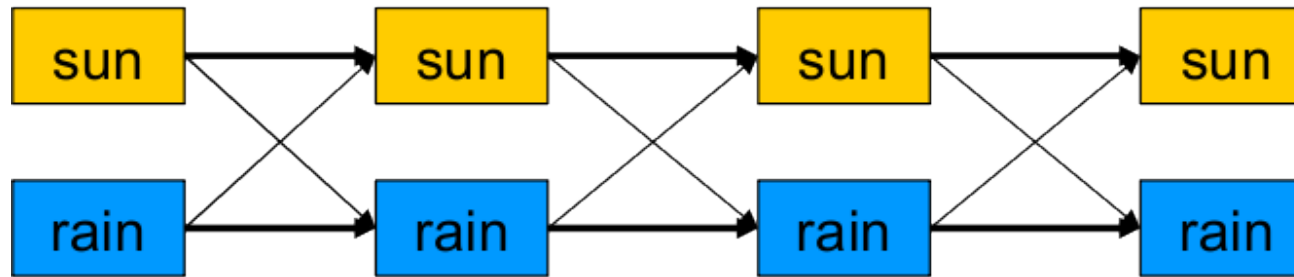
➤ Cache incremental updates:

● $m_t[x] = \max_{x_{1:t-1}} P(x_{1:t-1}, x)$

● $a_t[x] = \arg \max_{x_{1:t-1}} P(x_{1:t-1}, x)$

● Then read best sequence off of m and a vectors

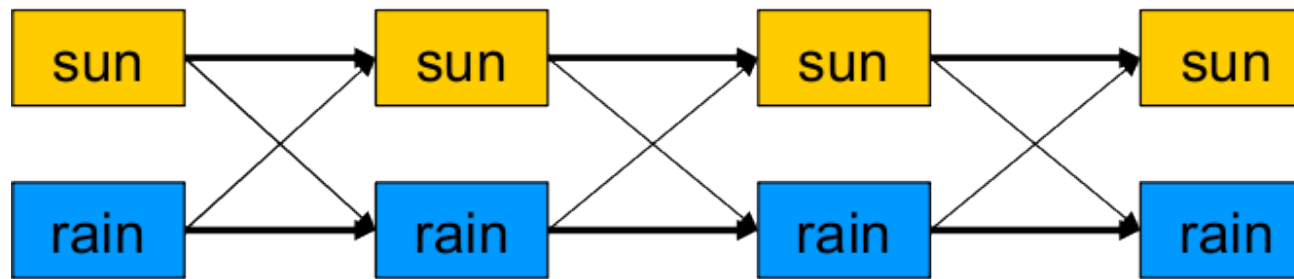
Mini-Viterbi Algorithm



$$\begin{aligned} m_t[x] &= \max_{x_{1:t-1}} P(x_{1:t-1}, x) \\ &= \max_{x_{1:t-1}} P(x_{1:t-1}) P(x|x_{t-1}) \\ &= \max_{x_{t-1}} P(x|x_{t-1}) \max_{x_{1:t-2}} P(x_{1:t-1}) \\ &= \max_{x_{t-1}} P(x|x_{t-1}) m_{t-1}[x] \end{aligned}$$

$$m_1[x] = P(x_1)$$

Mini-Viterbi Algorithm



$$\begin{aligned} & \max_{x_1, \dots, x_{t-1}} P(x_1, \dots, x_{t-1}, x_t) \\ &= \max_{x_{t-1}} \max_{x_1, \dots, x_{t-2}} P(x_1, \dots, x_{t-1}, x_t) \\ &= \max_{x_{t-1}} \max_{x_1, \dots, x_{t-2}} P(x_t | x_{t-1}) P(x_1, \dots, x_{t-1} | t) \\ &= \max_{x_{t-1}} P(x_t | x_{t-1}) \max_{x_1, \dots, x_{t-2}} P(x_1, \dots, x_{t-1} | t) \end{aligned}$$

$$m_t[x_t] = \max_{x_{t-1}} P(x_t | x_{t-1}) m_{t-1}[x]$$

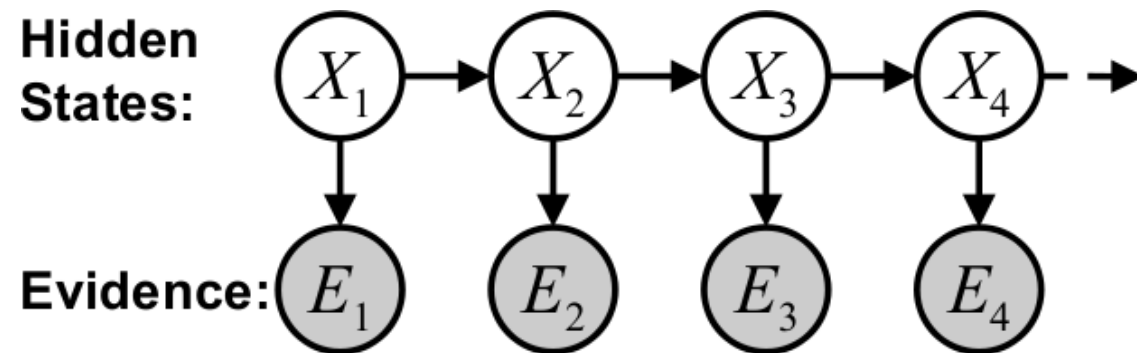
Hidden Markov Models

- Markov models not so useful for most agents
 - Eventually it does not know anything anymore
 - Need observations to update beliefs
- **Hidden Markov Models** (HMMs)
 - Underlying Markov chain over states S
 - Agent observes outputs/effects at each time step

➤ HMM consists of

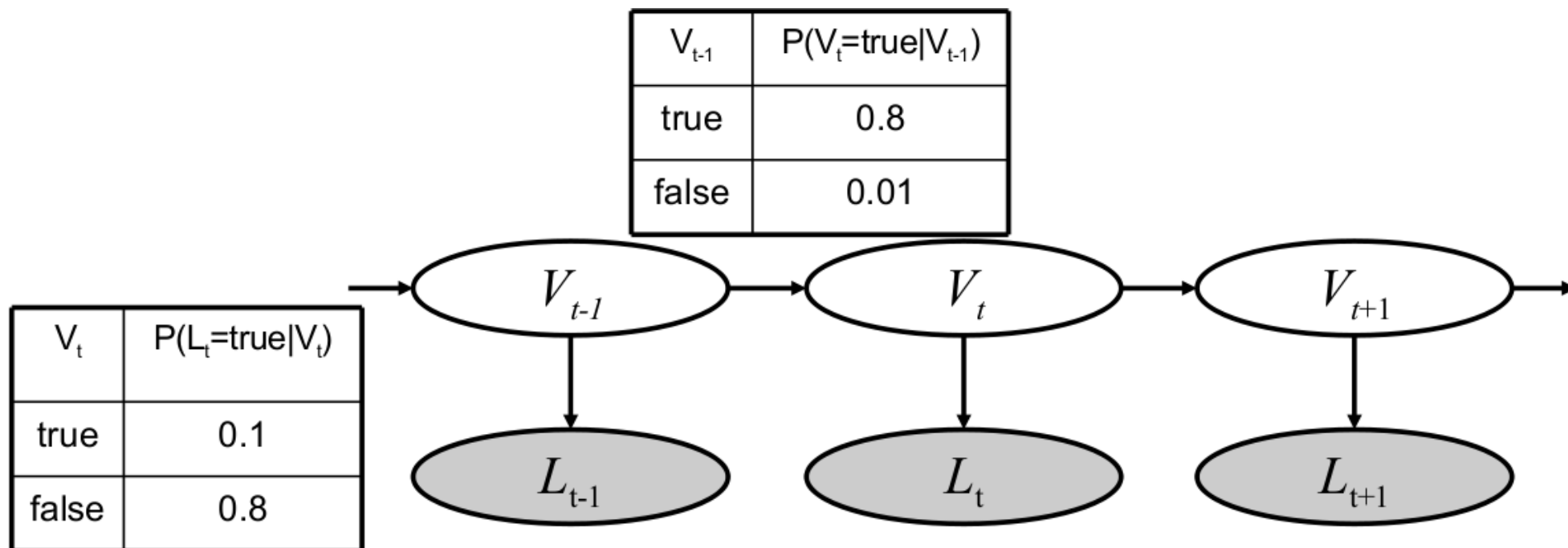
- *Initial distribution* $P(\mathbf{X}_1)$
- *Transitions* $P(\mathbf{X}_t | \mathbf{X}_{t-1})$
- *Emissions* $P(\mathbf{E}_t | \mathbf{X}_t)$

➔ Another Bayesian network



Burglar Agent

- Suppose a burglar agent wishes to determine when the inhabitants of a house are on vacation
- It visits once a night to see if the lights are on (inhabitants tend to have lights on when they are home)
- Inhabitants only go on vacation infrequently, but when they do, they tend to stay for a few days



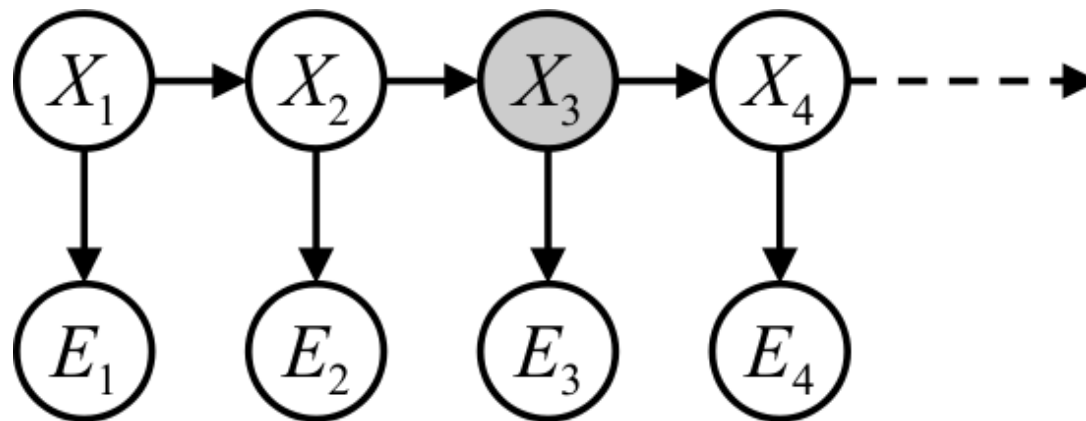
HMM Independence

➤ HMMs have two important independence properties

- **Markov hidden process**

- *Future depends only the present state, not the entire history*

- *Current observations independent of all else given current state*



➤ Note, observations not independent given no evidence due to hidden state

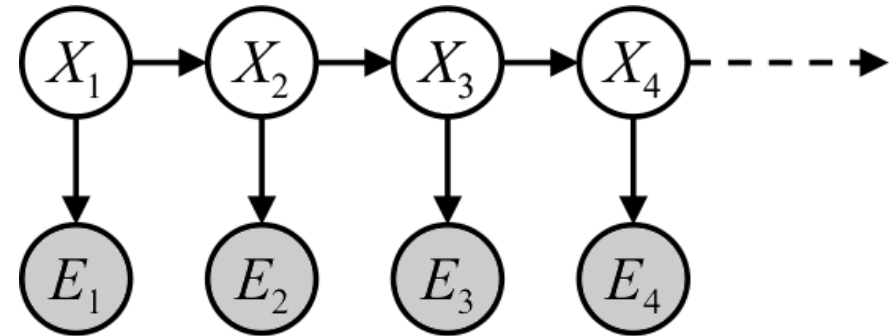
Filtering: Forward Algorithm

- Can ask the same questions for HMMs as for Markov chains
 - Compute $P(X_t = x_t | e_{1:t})$
 - ➔ **Monitoring** or **filtering**

$$\begin{aligned} P(x_t | e_{1:t}) &= \alpha P(x_t, e_{1:t}) \\ &= \alpha \sum_{x_{t-1}} P(x_{t-1}, x_t, e_{1:t}) \\ &= \alpha \sum_{x_{t-1}} P(x_{t-1}, e_{1:t-1}) P(x_t | x_{t-1}) P(e_t | x_t) \\ &= \alpha P(e_t | x_t) \sum_{x_{t-1}} P(x_t | x_{t-1}) P(x_{t-1}, e_{1:t-1}) \end{aligned}$$

Filtering: Forward Algorithm

- Given some evidence, what is the probability of different states at time t ?



$$P(X_t | e_1, \dots, e_t)$$

$$= P(X_t | e_1, \dots, e_{t-1}, e_t)$$

$$= \alpha P(e_t | X_t, e_1, \dots, e_{t-1}) P(X_t | e_1, \dots, e_{t-1})$$

$$= \alpha P(e_t | X_t) P(X_t | e_1, \dots, e_{t-1})$$

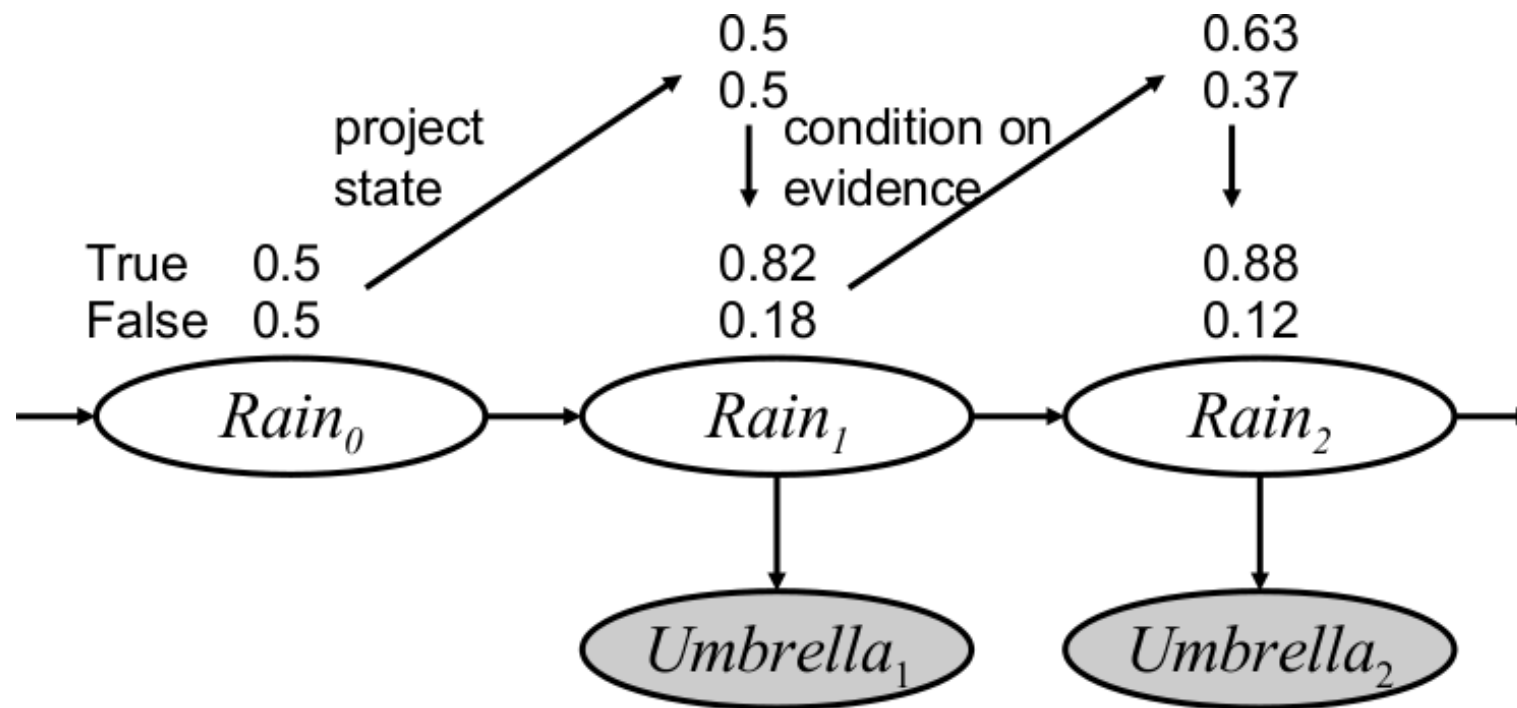
$$= \alpha P(e_t | X_t) \sum_{x_{t-1}} P(X_t | x_{t-1}, e_1, \dots, e_{t-1}) P(x_{t-1} | e_1, \dots, e_{t-1})$$

$$= \alpha P(e_t | X_t) \sum_{x_{t-1}} P(X_t | x_{t-1}) P(x_{t-1} | e_1, \dots, e_{t-1})$$

$$f_t[x_t] = P(e_t | x_t) \sum_{x_{t-1}} P(x_t | x_{t-1}) f_{t-1}[x_{t-1}]$$

Rain Filtering Example

$$P(x_t | e_{1:t}) = \alpha f_t[x_t] = \alpha P(x_t, e_{1:t})$$



$$f_t[x_t] = P(e_t | x_t) \sum_{x_{t-1}} P(x_t | x_{t-1}) f_{t-1}[x_{t-1}]$$

Viterbi Algorithm

- What is the most likely state sequence given the observations?

$$\mathbf{x}_{1:t}^* = \arg \max_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t} | \mathbf{e}_{1:t})$$

$$m_t[\mathbf{x}_t] = \max_{\mathbf{x}_{1:t-1}} P(\mathbf{x}_{1:t-1}, \mathbf{x}_t, \mathbf{e}_{1:t})$$

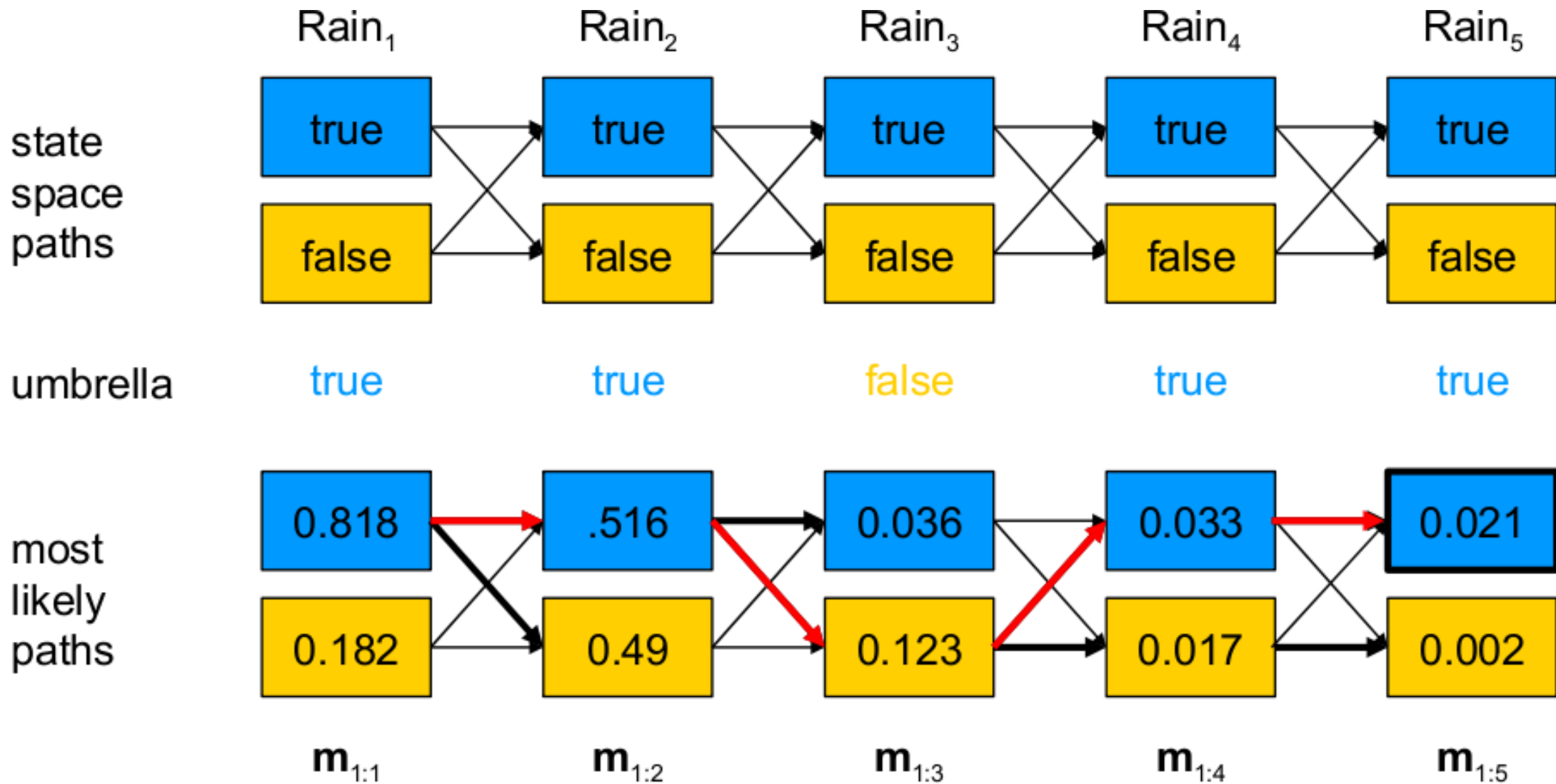
$$= \max_{\mathbf{x}_{1:t-1}} P(\mathbf{x}_{1:t-1}, \mathbf{e}_{1:t-1}) P(\mathbf{x}_t | \mathbf{x}_{t-1}) P(\mathbf{e}_t | \mathbf{x}_t)$$

$$= P(\mathbf{e}_t | \mathbf{x}_t) \max_{\mathbf{x}_{t-1}} P(\mathbf{x}_t | \mathbf{x}_{t-1}) \max_{\mathbf{x}_{1:t-2}} P(\mathbf{x}_{1:t-1}, \mathbf{e}_{1:t-1})$$

$$= P(\mathbf{e}_t | \mathbf{x}_t) \max_{\mathbf{x}_{t-1}} P(\mathbf{x}_t | \mathbf{x}_{t-1}) m_{t-1}[\mathbf{x}_{t-1}]$$

- Same as forward algorithm, but sums are replaced with maxima

Rain Viterbi Example





Viterbi Properties

- Designed for computing most likely state hidden sequence given a sequence of observations in an HMM
- Two passes:
 - Forward to compute the forward probabilities
 - Then backward to reconstruct the maximum sequence
- Time complexity: $O(d^2n)$
 - There are many extensions to Viterbi for other models
 - E.g. syntactic parsing

HMM Applications

➤ Widely applicable algorithm

