# Artificial Intelligence

## 14. Markov Models

Shashi Prabh

School of Engineering and Applied Science
Ahmedabad University

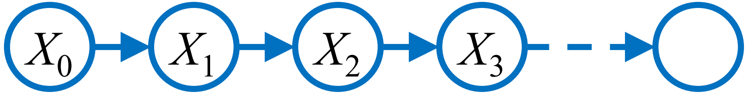
# Uncertainty and Time

- Often, we want to reason about a sequence of observations where the state of the underlying system is changing
  - Speech recognition
  - Robot localization
  - User attention
  - Medical monitoring
  - Global climate

- Need to introduce time into our models

# Markov Models

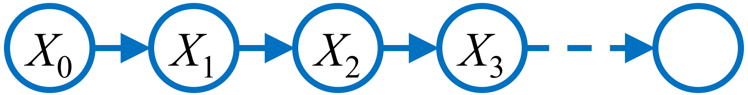
- Value of X at a given time is called the state (usually discrete, finite)




- Discrete-time model: view the problem as snapshots in time, called time slices
  - Each time slice contains a set of random variables, some observable and some not
    - We will assume the same subset of variables are observable in every time slice
- Transition model: $P(X_t \mid X_{t-1})$ how the state evolves over time
- Stationarity assumption: transition probabilities are the same at all times

# Markov Models

- Markov assumption: "future is independent of the past given the present"



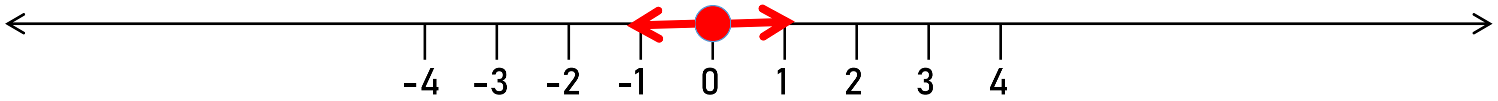

$P(X_0)$ $\quad$ $P(X_t \mid X_{0:t-1}) = P(X_t \mid X_{t-1})$

  - $X_{t+1}$ is independent of $X_0, \ldots, X_{t-1}$ given $X_t$
  - This is a first-order Markov model (a kth-order model allows dependencies on k earlier steps)
  - Higher order Markov chains can be transformed to the first order chain
  - Also called Markov chain or Markov process
- Joint distribution $P(X_0, \ldots, X_T) = P(X_0) \prod_t P(X_t \mid X_{t-1})$

# Are Markov models a special case of Bayes nets?

- Yes and no!
- Yes:
  - Directed acyclic graph, joint = product of conditionals
- No:
  - Infinitely many variables (unless we truncate)
  - Repetition of transition model not part of standard Bayes net syntax
- They are “growable” Bayes nets

# Example: Random walk in one dimension




- State: location on the unbounded integer line
- Initial probability: starts at 0
- Transition model: $P(X_t = k | X_{t-1} = k \pm 1) = 0.5$
- Applications: particle motion in crystals, stock prices, gambling, genetics, etc.
- Questions:
  - How far does it get as a function of $t$?
    - Expected distance is $O(\sqrt{t})$
  - Does it get back to 0 or can it go off for ever and not come back?
    - In 1D and 2D, returns w.p. 1; in 3D, returns w.p. 0.34053733

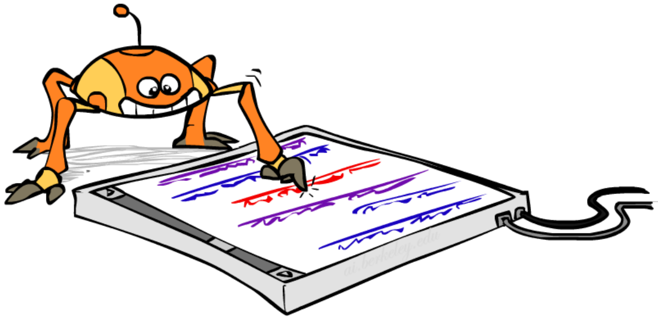
# Example: n-gram models

- State: word at position $t$ in text (can also build letter n-grams)
- Transition model (probabilities come from empirical frequencies):
  - Unigram (zero-order): $P(Word_t = i)$
    - "logical are as are confusion a may right tries agent goal the was . . ."
  - Bigram (first-order): $P(Word_t = i \mid Word_{t-1} = j)$
    - "systems are very similar computational approach would be represented . . ."
  - Trigram (second-order): $P(Word_t = i \mid Word_{t-1} = j, Word_{t-2} = k)$
    - "planning and scheduling are integrated the success of naive bayes model is . . ."
- Applications: text classification, spam detection, author identification, language classification, speech recognition

We call ourselves *Homo sapiens*—man the wise—because our **intelligence** is so important to us. For thousands of years, we have tried to understand *how we think*; that is, how a mere handful of matter can perceive, understand, predict, and manipulate a world far larger and more complicated than itself. ....

# Example: Web browsing

- State: URL visited at step t
- Transition model:
  - With probability p, choose an outgoing link at random
  - With probability (1-p), choose an arbitrary new page
- Question: What is the stationary distribution over pages?
  - I.e., if the process runs forever, what fraction of time does it spend in any given page?
- Application: Google page rank



# Example: Weather

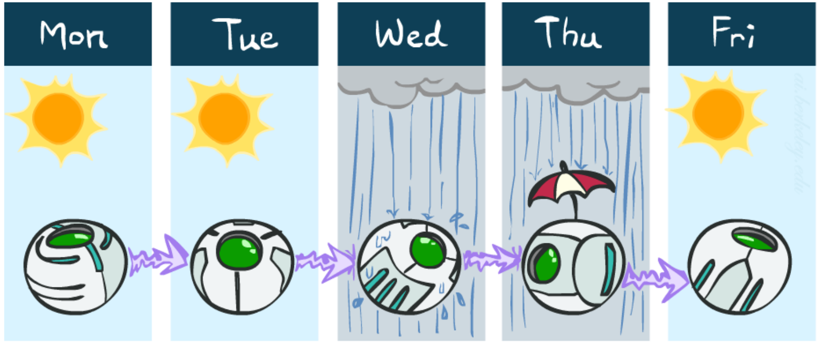
- States {rain, sun}

- Initial distribution $P(X_0)$

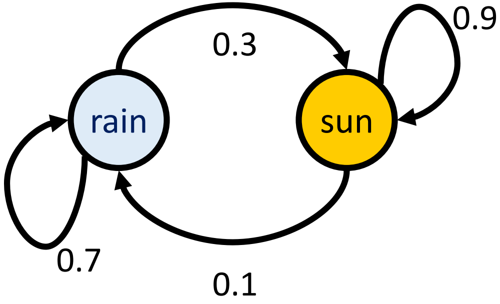
| $P(X_0)$ | |
|---|---|
| sun | rain |
| 0.5 | 0.5 |

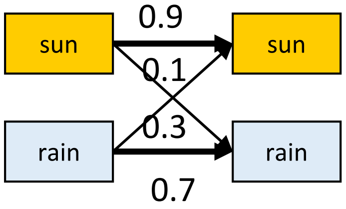
- Transition model $P(X_t \mid X_{t-1})$

| $X_{t-1}$ | $P(X_t \mid X_{t-1})$ | |
|---|---|---|
| | sun | rain |
| sun | 0.9 | 0.1 |
| rain | 0.3 | 0.7 |




More ways of representing the same CPT

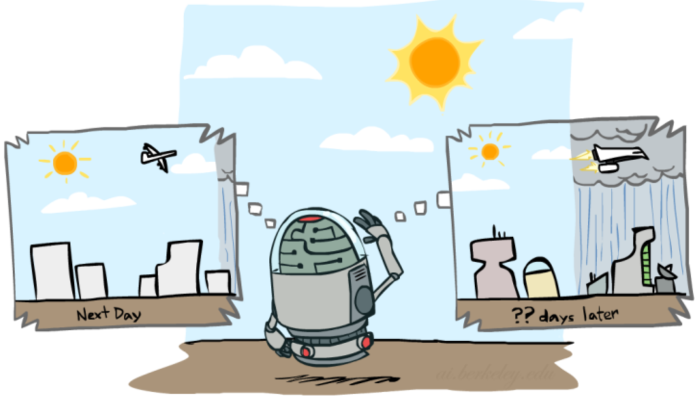


# Weather prediction

- Time 0: $\langle 0.5, 0.5 \rangle$

| $X_{t-1}$ | $P(X_t \mid X_{t-1})$ | |
|---|---|---|
| | sun | rain |
| sun | 0.9 | 0.1 |
| rain | 0.3 | 0.7 |



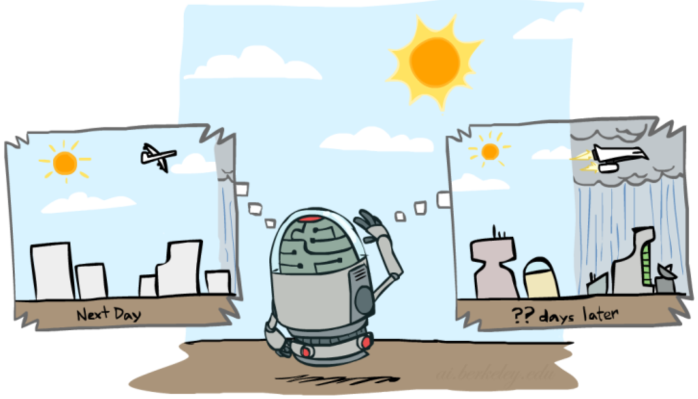

- What is the weather like at time 1?

$$P(X_1) = \sum_{x_0} P(X_1, X_0=x_0)$$
$$= \sum_{x_0} P(X_0=x_0)\, P(X_1 \mid X_0=x_0)$$
$$= 0.5\langle 0.9, 0.1\rangle + 0.5\langle 0.3, 0.7\rangle = \langle 0.6, 0.4\rangle$$

# Weather prediction, contd.

- Time 1: $\langle 0.6, 0.4 \rangle$

| $X_{t-1}$ | $P(X_t \mid X_{t-1})$ | |
|---|---|---|
| | sun | rain |
| sun | 0.9 | 0.1 |
| rain | 0.3 | 0.7 |




- What is the weather like at time 2?

$$
\begin{aligned}
P(X_2) &= \textstyle\sum_{x_1} P(X_2, X_1 = x_1) \\
&= \textstyle\sum_{x_1} P(X_1 = x_1)\, P(X_2 \mid X_1 = x_1) \\
&= 0.6\langle 0.9, 0.1 \rangle + 0.4\langle 0.3, 0.7 \rangle = \langle 0.66, 0.34 \rangle
\end{aligned}
$$

# Weather prediction, contd.

- Time 2: <0.66, 0.34>

| $X_{t-1}$ | $P(X_t \mid X_{t-1})$ | |
|---|---|---|
| | sun | rain |
| sun | 0.9 | 0.1 |
| rain | 0.3 | 0.7 |

- What is the weather like at time 3?

$$P(X_3) = \sum_{x_2} P(X_3, X_2 = x_2)$$
$$= \sum_{x_2} P(X_2 = x_2)\, P(X_3 \mid X_2 = x_2)$$
$$= 0.66\langle 0.9, 0.1\rangle + 0.34\langle 0.3, 0.7\rangle = \langle 0.696, 0.304\rangle$$

Homework

| $P(X_0)$ | |
|---|---|
| sun | rain |
| 0 | 1 |

- The influence of initial distribution gets less and less over time. The distribution much later becomes independent of the initial distribution

# Forward algorithm (simple form)

Transition model

- What is the state at time t?
  - $P(X_t) = \sum_{x_{t-1}} P(X_t, X_{t-1}=x_{t-1})$
  - $= \sum_{x_{t-1}} P(X_{t-1}=x_{t-1})\, P(X_t \mid X_{t-1}=x_{t-1})$
- Iterate this update starting at t=0
  - This is called a recursive update: $P_t = g(P_{t-1}) = g(g(g(g( \ldots P_0))))$

# And the same thing in linear algebra

- What is the weather like at time 2?
  - $P(X_2) = 0.6\langle 0.9,0.1\rangle + 0.4\langle 0.3,0.7\rangle = \langle 0.66,0.34\rangle$
- In matrix-vector form:
  - $P(X_2) = \begin{pmatrix} 0.9 & 0.3 \\ 0.1 & 0.7 \end{pmatrix} \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} = \begin{pmatrix} 0.66 \\ 0.34 \end{pmatrix}$

| $X_{t-1}$ | $P(X_t \mid X_{t-1})$ | |
|---|---|---|
| | sun | rain |
| sun | 0.9 | 0.1 |
| rain | 0.3 | 0.7 |

i.e., multiply by $T^T$, transpose of transition matrix

# Stationary Distributions

- The limiting distribution is called the stationary distribution $P_\infty$ of the chain
- It satisfies $P_\infty = P_{\infty+1} = T^T P_\infty$
- Solving for $P_\infty$ in the example:

$$\begin{pmatrix} 0.9 & 0.3 \\ 0.1 & 0.7 \end{pmatrix} \begin{pmatrix} p \\ 1-p \end{pmatrix} = \begin{pmatrix} p \\ 1-p \end{pmatrix}$$

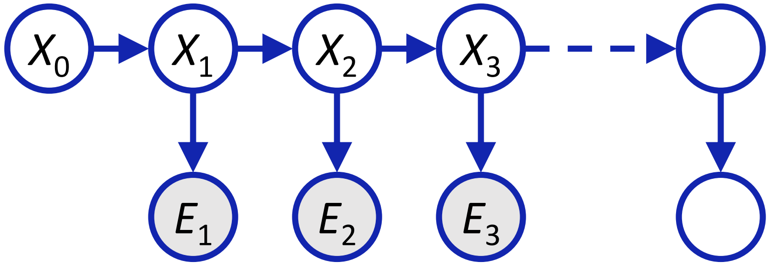
$0.9p + 0.3(1-p) = p$

$p = 0.75$

Stationary distribution is <0.75,0.25> regardless of the starting distribution

# Hidden Markov Models

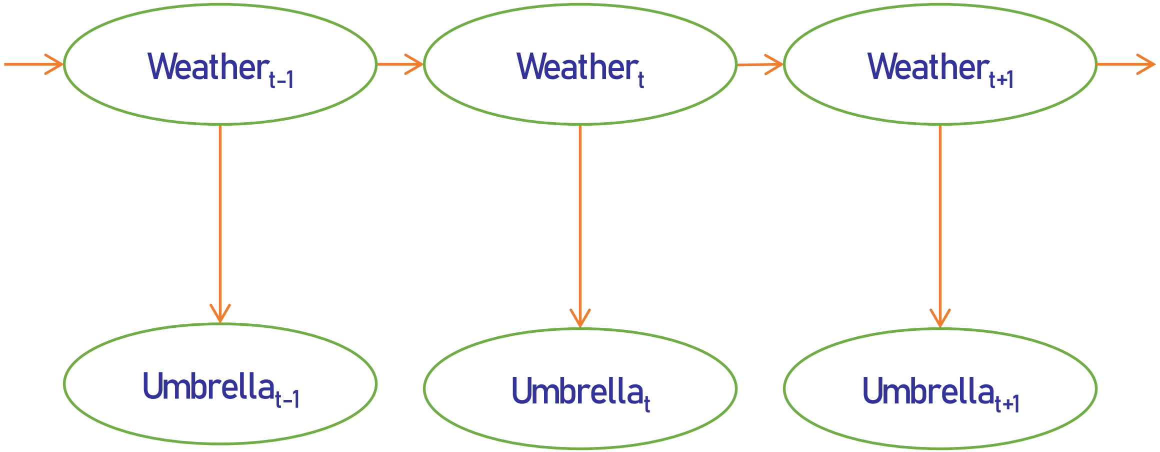
- Usually the true state is not observed directly
  - An agent maintains a belief state
- Hidden Markov models (HMMs)
  - Underlying Markov chain over belief states X
  - You observe evidence E at each time step
  - $X_t$ is a single discrete variable; $E_t$ may be continuous and may consist of several variables



# Example: Weather HMM

- An HMM is defined by:
  - Initial distribution: $P(X_0)$
  - Transition model: $P(X_t \mid X_{t-1})$
  - Sensor model: $P(E_t \mid X_t)$

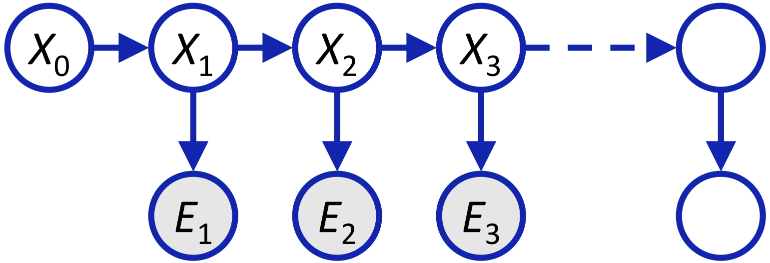
| $W_{t-1}$ | $P(W_t \mid W_{t-1})$ | |
|---|---|---|
| | sun | rain |
| sun | 0.9 | 0.1 |
| rain | 0.3 | 0.7 |




| $W_t$ | $P(U_t \mid W_t)$ | |
|---|---|---|
| | true | false |
| sun | 0.2 | 0.8 |
| rain | 0.9 | 0.1 |

# HMM as probability model

- Joint distribution for Markov model: $P(X_{0:t}) = P(X_0) \prod_{i=1:t} P(X_i \mid X_{i-1})$
- Joint distribution for hidden Markov model:
  $P(X_{0:t}, E_{1:t}) = P(X_0) \prod_{i=1:t} P(X_i \mid X_{i-1}) P(E_i \mid X_i)$
  - Future states are independent of the past given the present
  - Current evidence is independent of everything else given the current state
- Are evidence variables independent of each other?




Useful notation:

$X_{a:b} = X_a, X_{a+1}, \ldots, X_b$

# Real HMM Examples

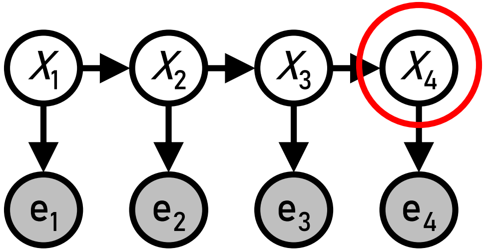
- Speech recognition HMMs:
  - Observations are acoustic signals (continuous valued)
  - States are specific positions in specific words (so, tens of thousands)
- Machine translation HMMs:
  - Observations are words (tens of thousands)
  - States are translation options
- Robot tracking:
  - Observations are range readings (continuous)
  - States are positions on a map (continuous)

# Inference tasks

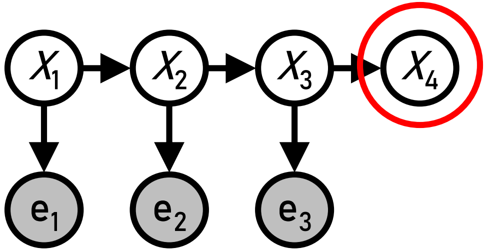
- **Filtering**: $P(X_t|e_{1:t})$
  - belief state—input to the decision process of a rational agent
- **Prediction**: $P(X_{t+k}|e_{1:t})$ for $k > 0$
  - evaluation of possible action sequences; like filtering without the evidence
- **Smoothing**: $P(X_k|e_{1:t})$ for $0 \le k < t$
  - better estimate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

# Inference tasks

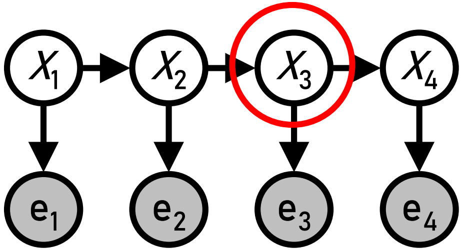
## Filtering: $P(X_t|e_{1:t})$



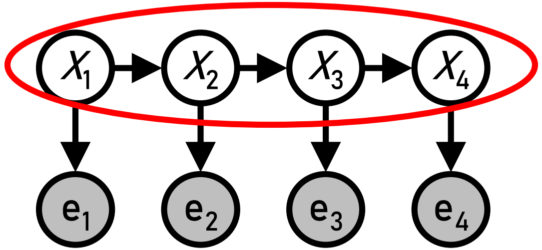

## Prediction: $P(X_{t+k}|e_{1:t})$




## Smoothing: $P(X_k|e_{1:t})$, $k<t$




## Explanation: $P(X_{1:t}|e_{1:t})$



# Filtering / Monitoring

- Filtering, or monitoring, or state estimation, is the task of maintaining the distribution $f_{1:t} = P(X_t|e_{1:t})$ over time
- We start with $f_0$ in an initial setting, usually uniform
- Filtering is a fundamental task in engineering and science
- The Kalman filter (continuous variables, linear dynamics, Gaussian noise) was invented in 1960 and used for trajectory estimation in the Apollo program
  - Core ideas used by Gauss for planetary observations
  - 788,000 papers on Google Scholar

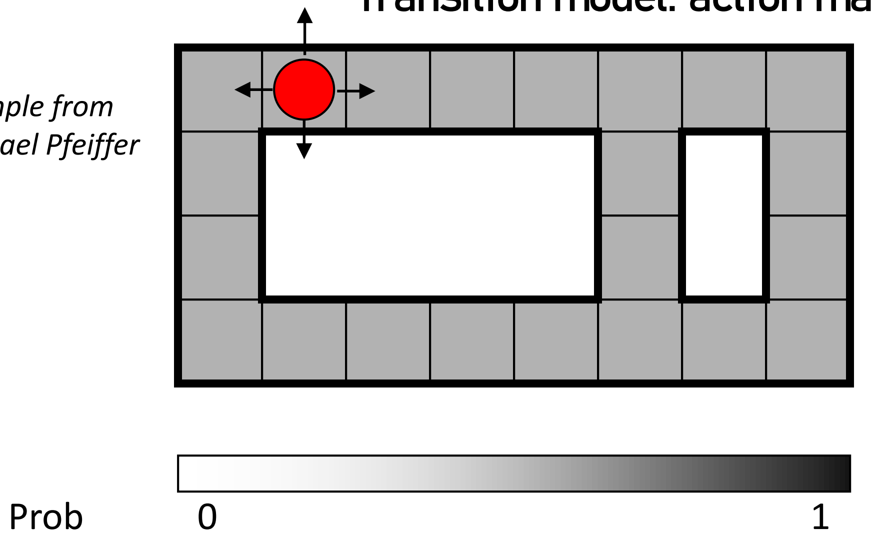
# Example: Robot Localization

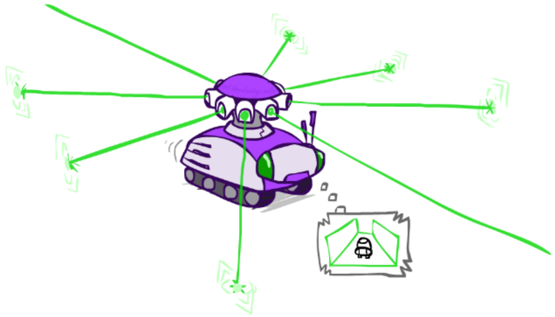
t=0

Sensor model: four bits for wall/no-wall in each direction, never more than 1 mistake

Transition model: action may fail with small prob.

*Example from Michael Pfeiffer*

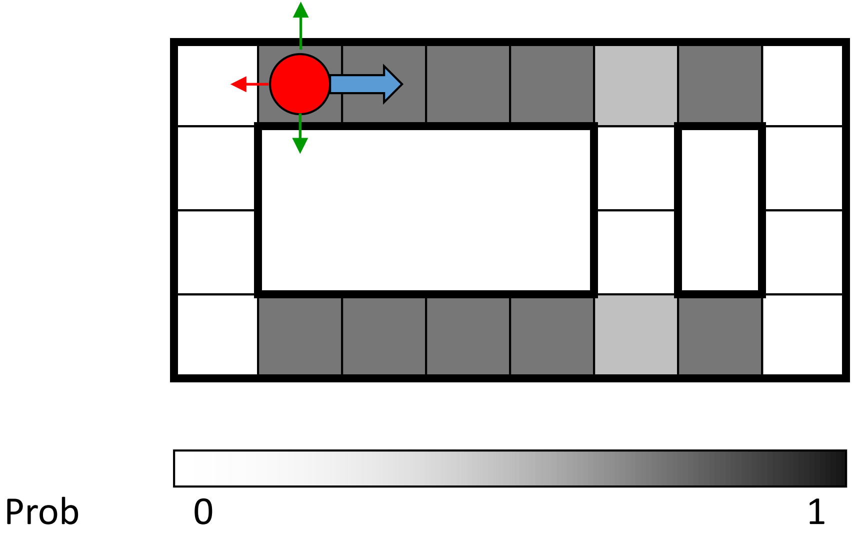


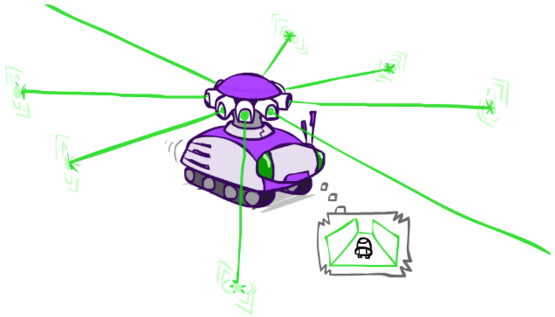



# Example: Robot Localization

t=1

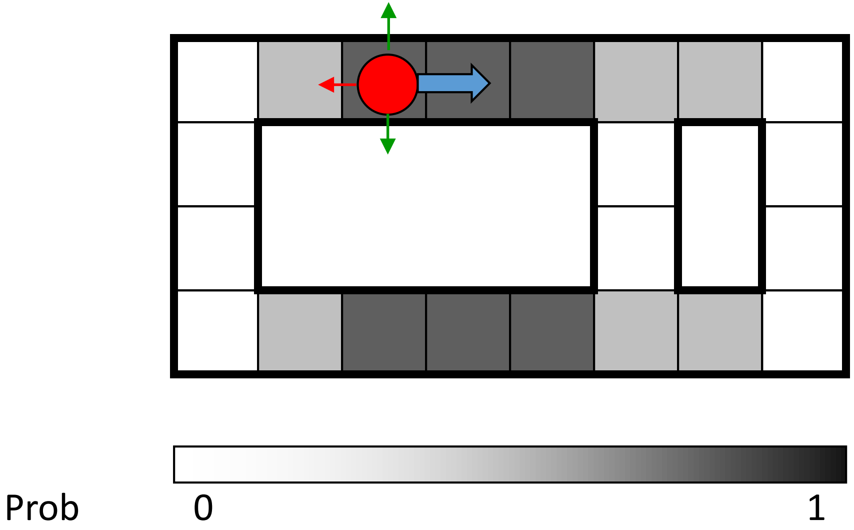
Lighter grey: was *possible* to get the reading, but *less likely* (required 1 mistake)

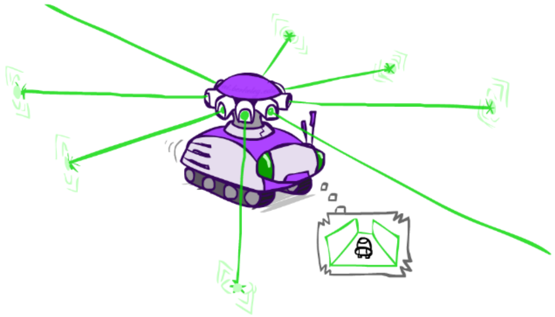


# Example: Robot Localization

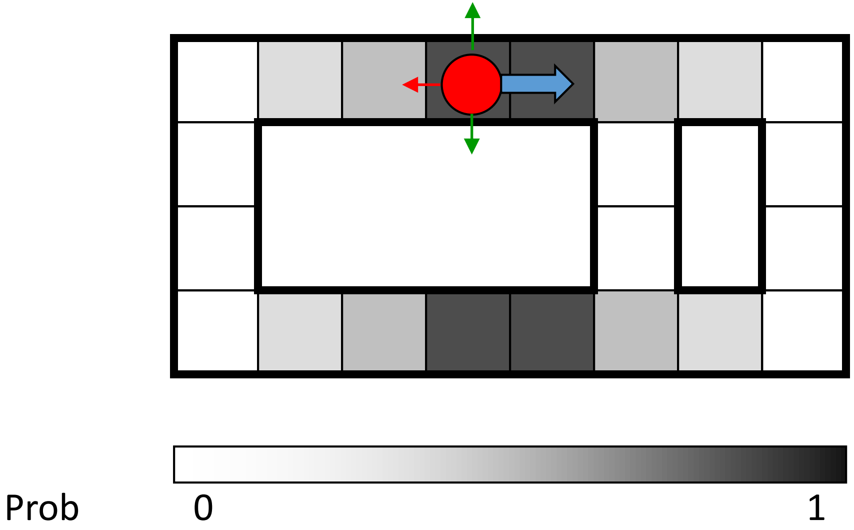
t=2

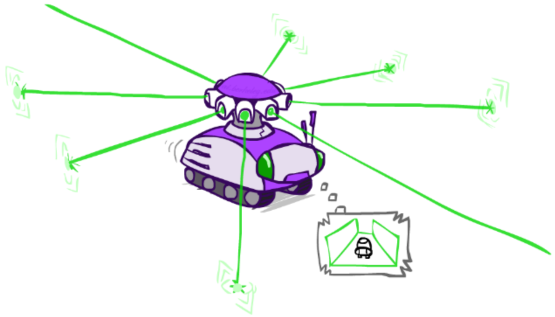


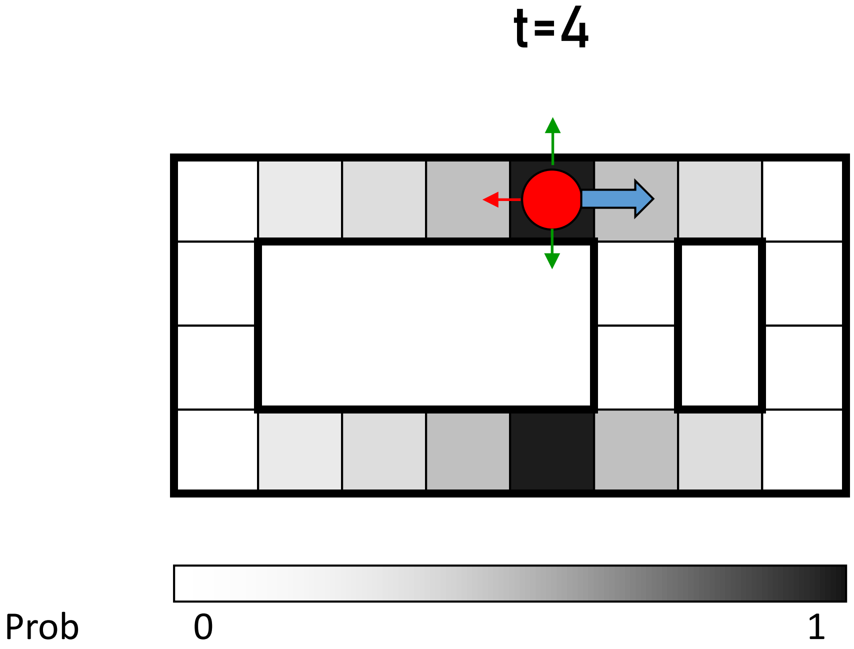
# Example: Robot Localization

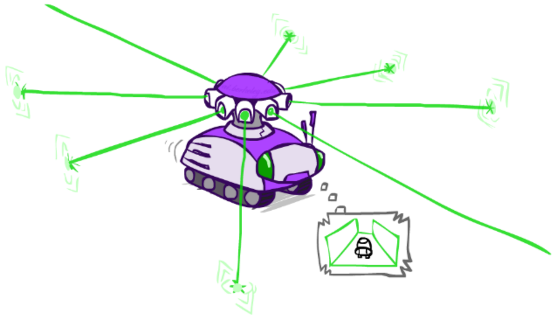
t=3



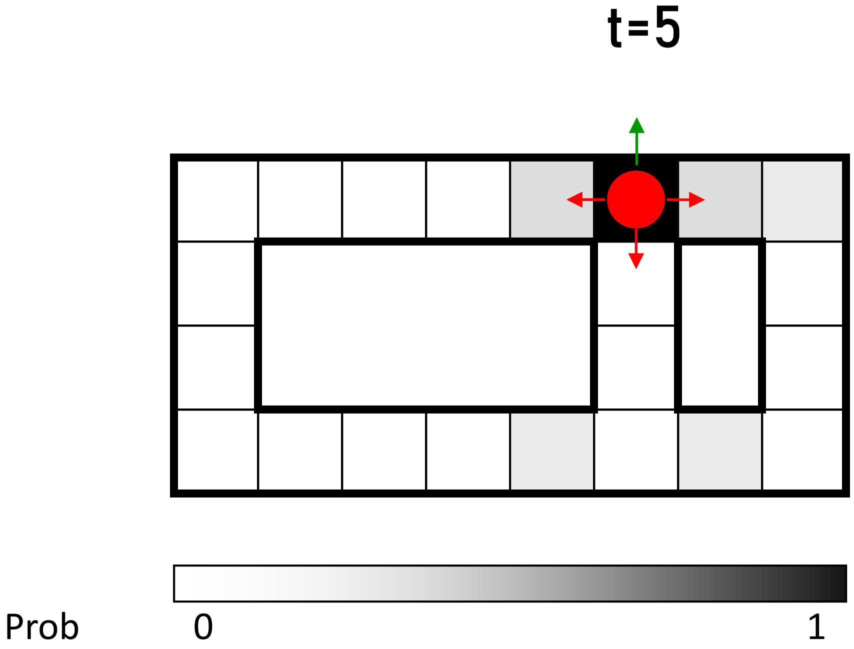



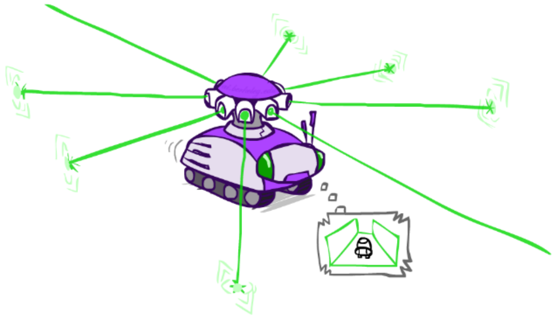
# Example: Robot Localization



# Example: Robot Localization



# Filtering algorithm

- Aim: devise a recursive filtering algorithm of the form
  - $P(X_{t+1}|e_{1:t+1}) = g(e_{t+1}, P(X_t|e_{1:t}))$

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

# Filtering algorithm

- Aim: devise a recursive filtering algorithm of the form
  - $P(X_{t+1}|e_{1:t+1}) = f(e_{t+1}, P(X_t|e_{1:t}))$

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

# Filtering algorithm

- Aim: devise a recursive filtering algorithm of the form
  - $P(X_{t+1}|e_{1:t+1}) = f(e_{t+1}, P(X_t|e_{1:t}))$

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

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

Apply Bayes' rule

# Filtering algorithm

- Aim: devise a recursive filtering algorithm of the form
  - $P(X_{t+1}|e_{1:t+1}) = f(e_{t+1}, P(X_t|e_{1:t}))$

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

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

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

Apply sensor Markov conditional independence

# Filtering algorithm

- Aim: devise a recursive filtering algorithm of the form
  - $P(X_{t+1}|e_{1:t+1}) = f(e_{t+1}, P(X_t|e_{1:t}))$

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

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

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

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

Condition on $X_t$

# Filtering algorithm

- Aim: devise a recursive filtering algorithm of the form
  - $P(X_{t+1}|e_{1:t+1}) = f(e_{t+1}, P(X_t|e_{1:t}))$

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

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

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

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

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

Apply conditional independence

Normalize

Sensor model

Transition model

Recursion

# Filtering algorithm

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

Normalize — Update — Predict

- $f_{1:t+1}$ = FORWARD($f_{1:t}$ , $e_{t+1}$)
- Cost per time step: $O(|X|^2)$ where $|X|$ is the number of states
- Time and space costs are constant, independent of $t$
- $O(|X|^2)$ is infeasible for models with many state variables
- We get to invent really cool approximate filtering algorithms

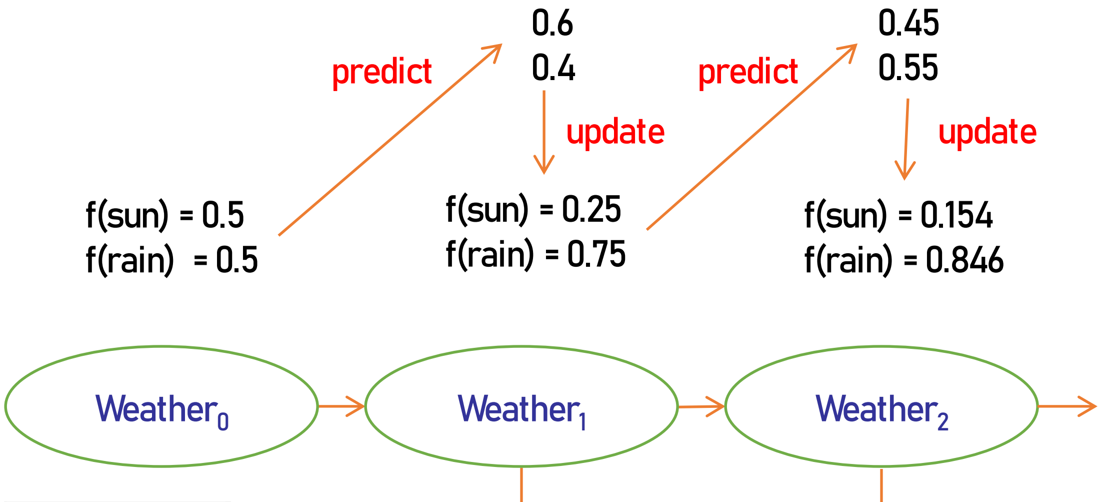

# And the same thing in linear algebra

- Transition matrix $T$, observation matrix $O_t$
  - Observation matrix has state likelihoods for $E_t$ along diagonal
  - E.g., for $U_1 = true$, $O_1 = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.9 \end{pmatrix}$
- Filtering algorithm becomes
  - $f_{1:t+1} = \alpha \, O_{t+1} T^T f_{1:t}$

| $X_{t-1}$ | $P(X_t \mid X_{t-1})$ | |
|---|---|---|
| | sun | rain |
| sun | 0.9 | 0.1 |
| rain | 0.3 | 0.7 |

| $W_t$ | $P(U_t \mid W_t)$ | |
|---|---|---|
| | true | false |
| sun | 0.2 | 0.8 |
| rain | 0.9 | 0.1 |

# Example: Weather HMM




| $P(W_0)$ | |
|---|---|
| sun | rain |
| 0.5 | 0.5 |

| $W_{t-1}$ | $P(W_t \mid W_{t-1})$ | |
|---|---|---|
| | sun | rain |
| sun | 0.9 | 0.1 |
| rain | 0.3 | 0.7 |

| $W_t$ | $P(U_t \mid W_t)$ | |
|---|---|---|
| | true | false |
| sun | 0.2 | 0.8 |
| rain | 0.9 | 0.1 |

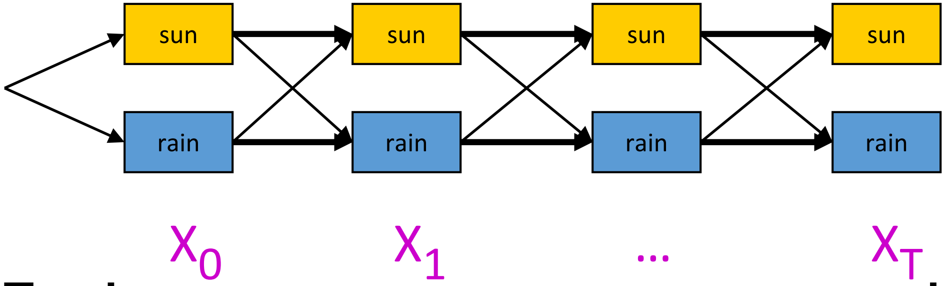
# Most Likely Explanation

# Inference tasks

- 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 $0 \le k < t$
  - better estimate of past states, essential for learning
- Most likely explanation: $\arg\max_{x_{1:t}} P(x_{1:t} \mid e_{1:t})$
  - speech recognition, decoding with a noisy channel

# Most likely explanation = most probable path

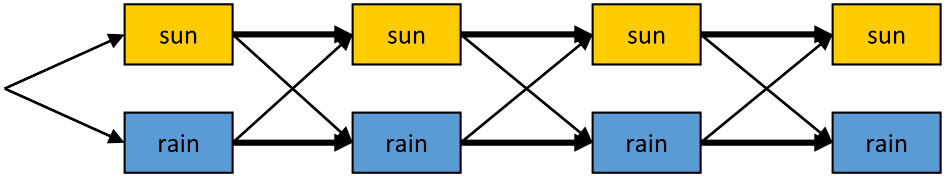
- State trellis: graph of states and transitions over time




$$\arg\max_{x_{1:t}} P(x_{1:t} \mid e_{1:t})$$
$$= \arg\max_{x_{1:t}} \alpha P(x_{1:t}, e_{1:t})$$
$$= \arg\max_{x_{1:t}} P(x_{1:t}, e_{1:t})$$
$$= \arg\max_{x_{1:t}} P(x_0) \prod_t P(x_t \mid x_{t-1}) P(e_t \mid x_t)$$

- Each arc represents some transition $x_{t-1} \rightarrow x_t$
- Each arc has weight $P(x_t \mid x_{t-1}) P(e_t \mid x_t)$
  - Arcs to initial states have weight $P(x_0)$ )
- The **product** of weights on a path is proportional to that state sequence's probability
- Forward algorithm computes sums of paths, Viterbi algorithm computes best paths

# Forward / Viterbi algorithms




$X_0$ $X_1$ … $X_T$

## Forward Algorithm (sum)

For each state at time t, keep track of the total probability of all paths to it

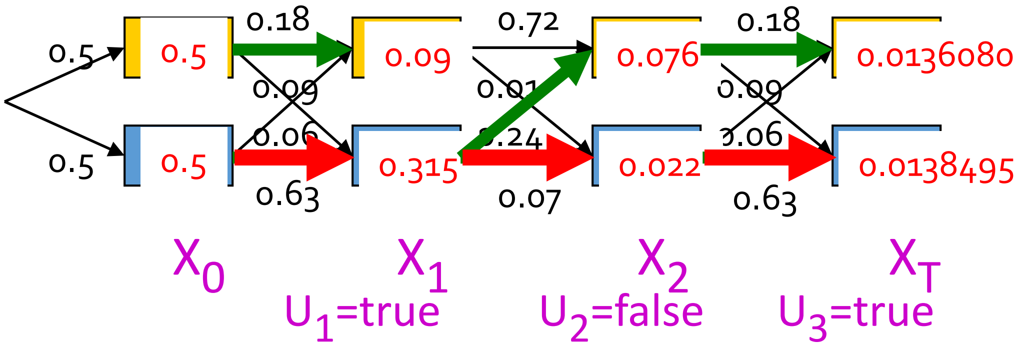
$f_{1:t+1} = \text{FORWARD}(f_{1:t}, e_{t+1})$
$= \alpha\, P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}| x_t)\, f_{1:t}$

## Viterbi Algorithm (max)

For each state at time t, keep track of the maximum probability of any path to it

$m_{1:t+1} = \text{VITERBI}(m_{1:t}, e_{t+1})$
$= P(e_{t+1}|X_{t+1}) \max_{x_t} P(X_{t+1}| x_t)\, m_{1:t}$

# Viterbi algorithm contd.




| $W_{t-1}$ | $P(W_t \mid W_{t-1})$ | |
|---|---|---|
| | sun | rain |
| sun | 0.9 | 0.1 |
| rain | 0.3 | 0.7 |

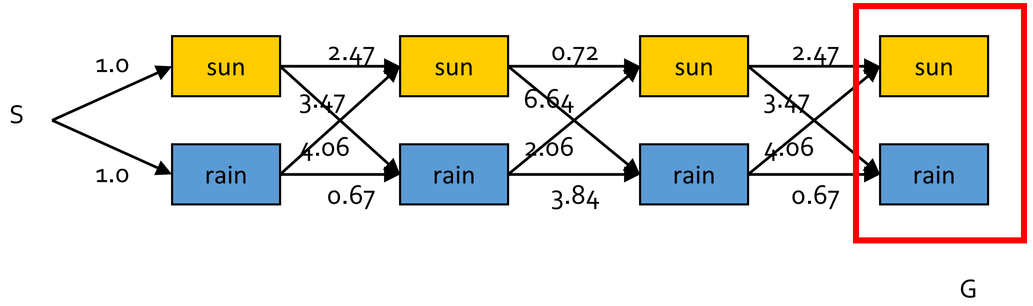
| $W_t$ | $P(U_t \mid W_t)$ | |
|---|---|---|
| | true | false |
| sun | 0.2 | 0.8 |
| rain | 0.9 | 0.1 |

**Time complexity?**
$O(|X|^2 T)$

**Space complexity?**
$O(|X| T)$

**Number of paths?**
$O(|X|^T)$

# Viterbi in negative log space




| $W_{t-1}$ | $P(W_t \mid W_{t-1})$ | |
|---|---|---|
| | sun | rain |
| sun | 0.9 | 0.1 |
| rain | 0.3 | 0.7 |

| $W_t$ | $P(U_t \mid W_t)$ | |
|---|---|---|
| | true | false |
| sun | 0.2 | 0.8 |
| rain | 0.9 | 0.1 |

argmax of product of probabilities
= argmin of sum of negative log probabilities
= minimum-cost path

Viterbi is essentially breadth-first graph search
What about A*?

# Next time

- Chapter 16. Utility theory