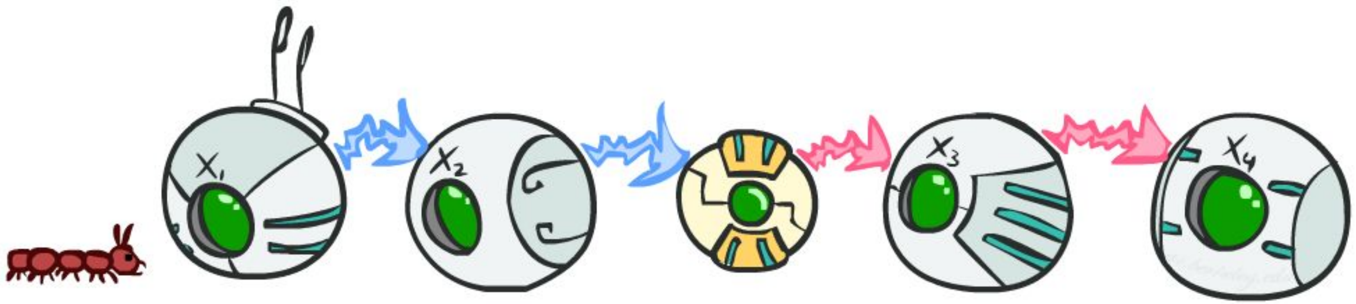
CS 188: Artificial Intelligence

Inference in Markov Models



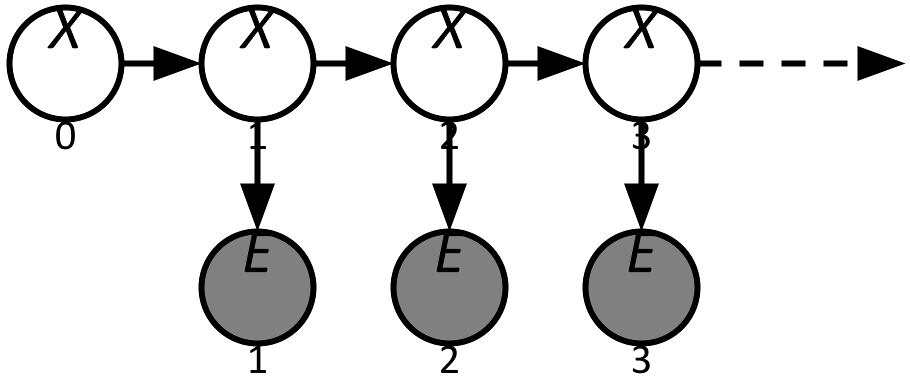
Instructors: Stuart Russell and Dawn Song

University of California, Berkeley

HMM as probability model

- Joint distribution for Markov model: $P(X_0, \ldots, X_T) = P(X_0) \prod_{t=1:T} P(X_t \mid X_{t-1})$
- Joint distribution for hidden Markov model:

  $P(X_0, X_1, \ldots, X_T, E_T) = P(X_0) \prod_{t=1:T} P(X_t \mid X_{t-1}) \, P(E_t \mid X_t)$
- Future states are independent of the past given the present
- Current evidence is independent of everything else given the current state



Useful notation:

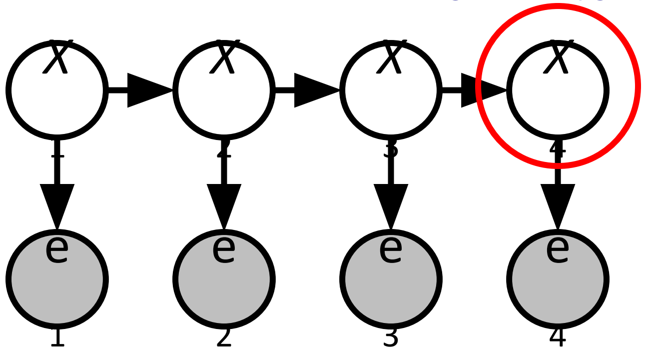
$X_{a:b} = X_a, X_{a+1}, \ldots, X_b$

Inference tasks

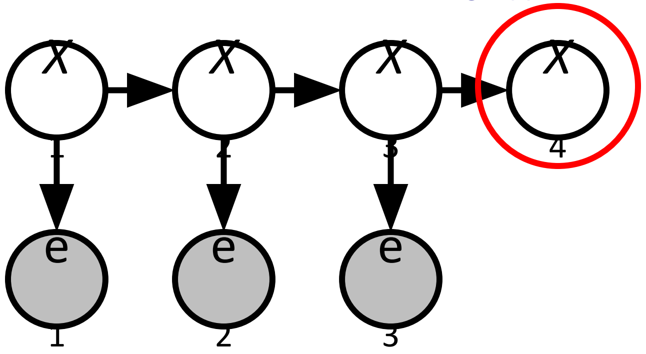
- ***Filtering***: $P(X_t | e_{1:t})$
  - ***belief state***—input to the decision process of a rational agent
- ***Prediction***: $P(X_{t+k} | e_{1:t})$ for $k > 0$
  - evaluation of possible action sequences; like filtering without the evidence
- ***Smoothing***: $P(X_k | e_{1:t})$ for $0 \le k < t$
  - 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

Other HMM Queries

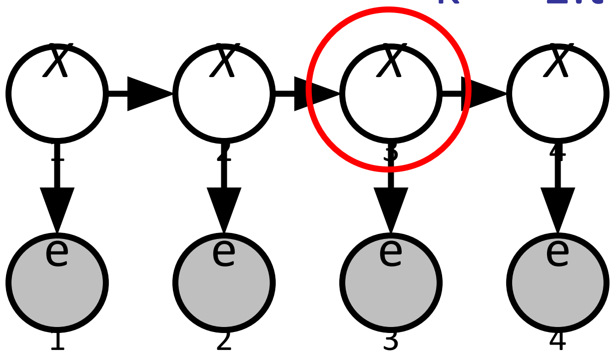
Filtering: $P(X_t | e_{1:t})$



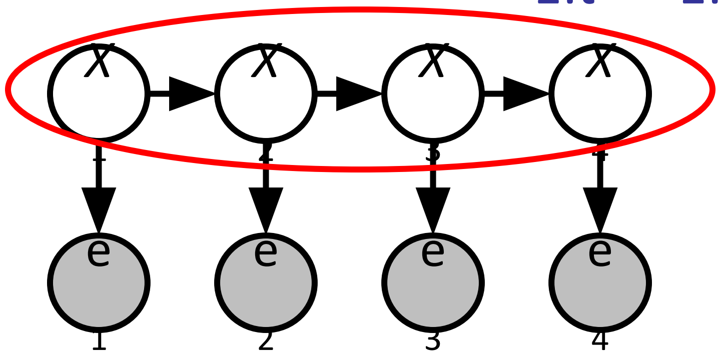
Prediction: $P(X_{t+k} | e_{1:t})$



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 $\boldsymbol{f}_{1:t} = P(X_t | e_{1:t})$ over time

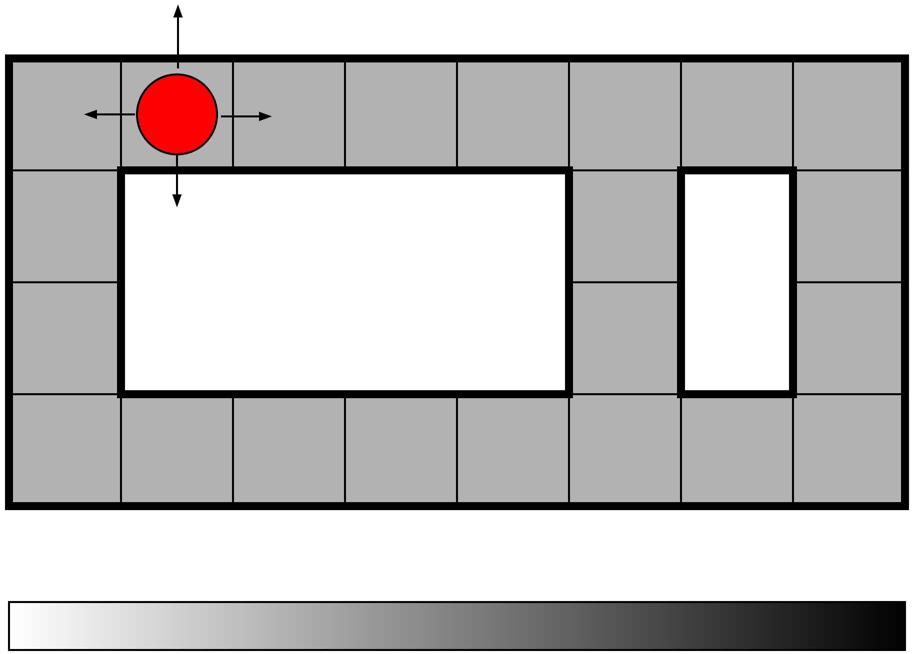
- We start with $\boldsymbol{f}_0$ in an initial setting, usually uniform

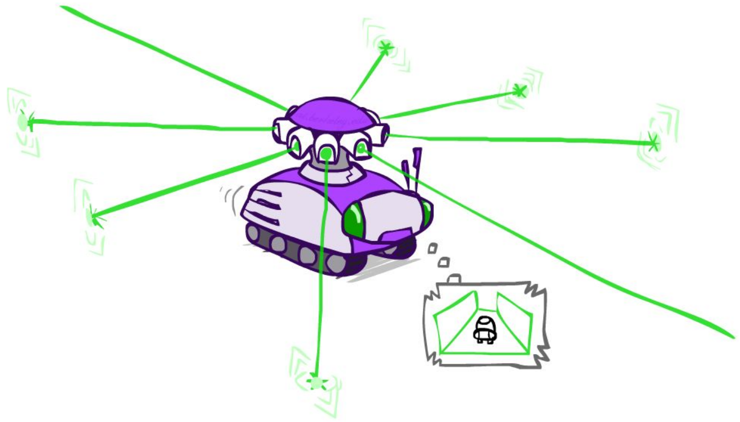
- Filtering is a fundamental task in engineering and science

- 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

Example: Robot Localization

Example from
Michael Pfeiffer





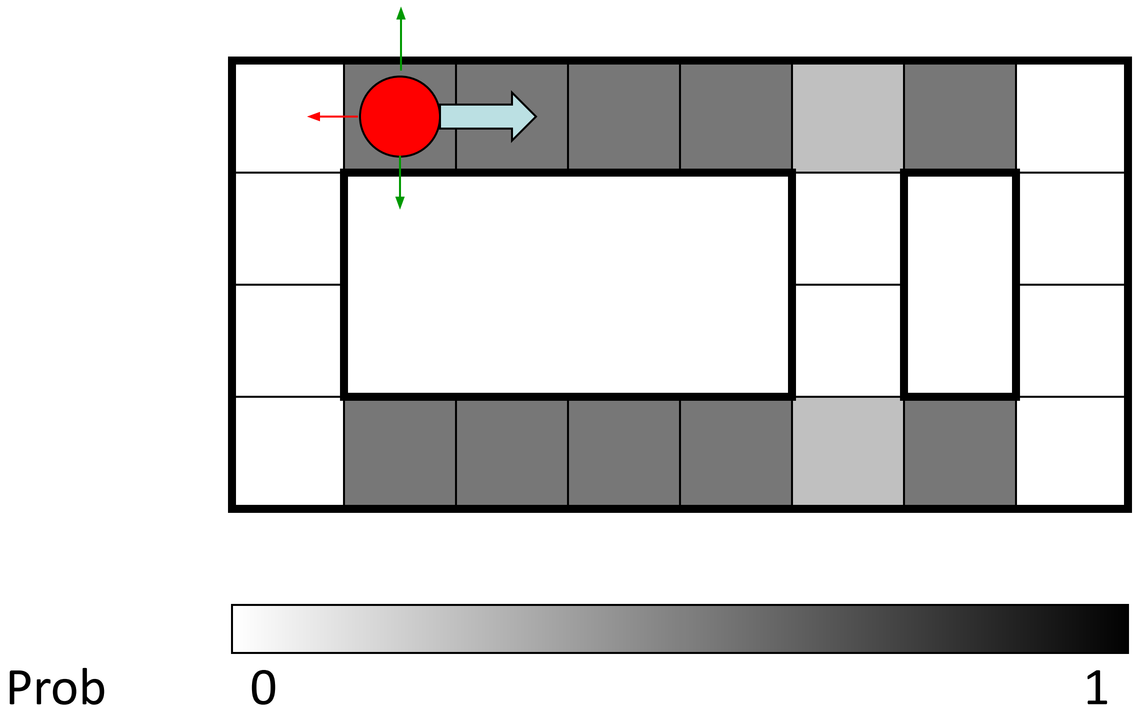
Prob 0 1

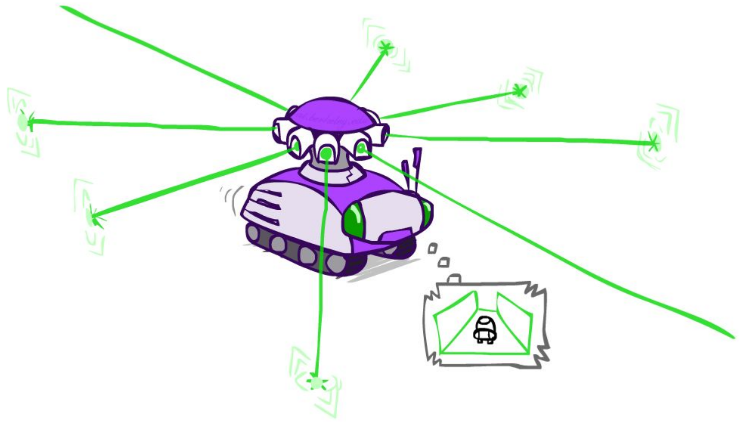
t=0

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: Robot Localization



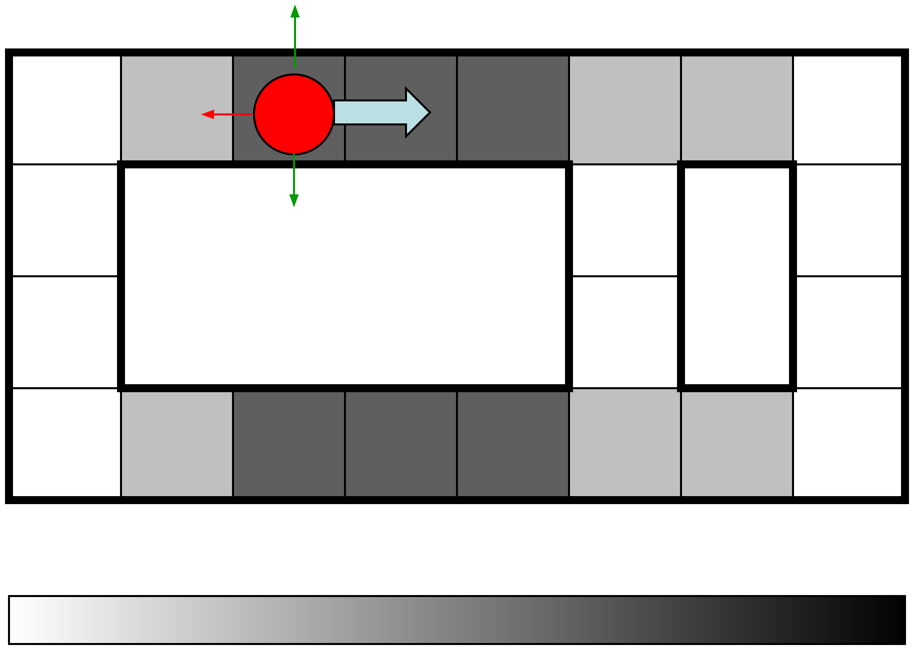


t=1

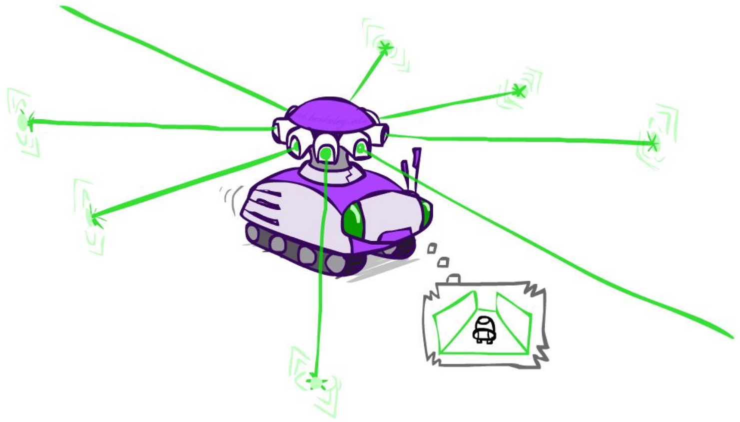
Lighter grey: was ***possible*** to get the reading,

but ***less likely*** (required 1 mistake)

Example: Robot Localization

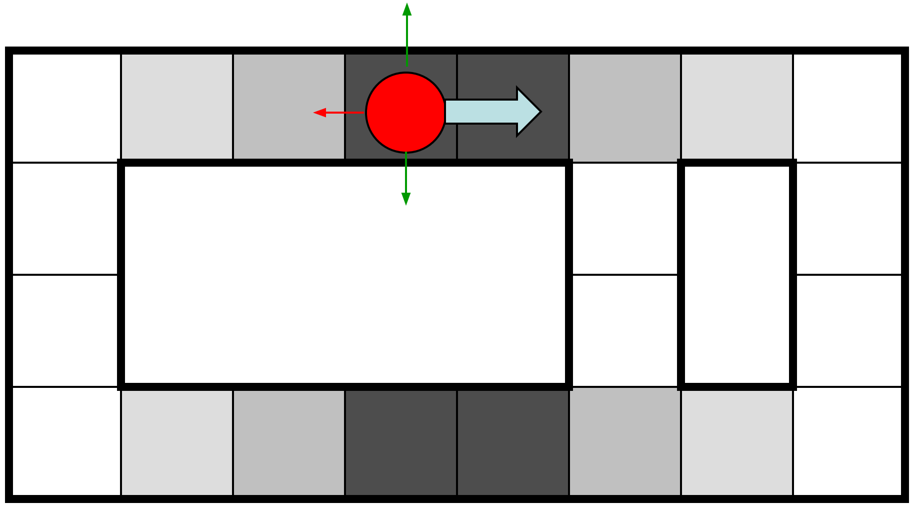


Prob 0 1

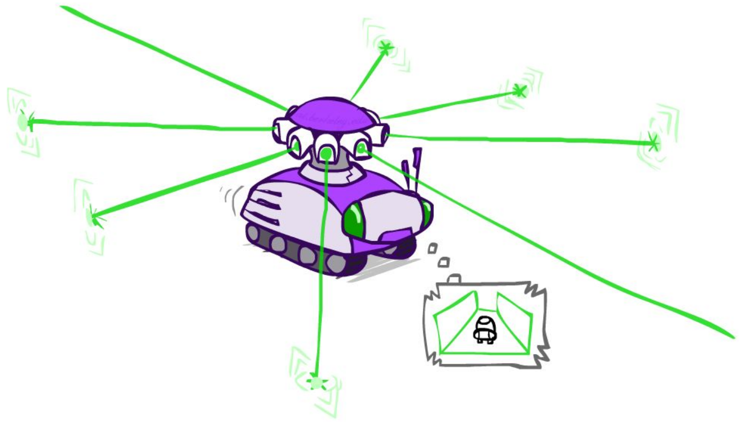


t=2

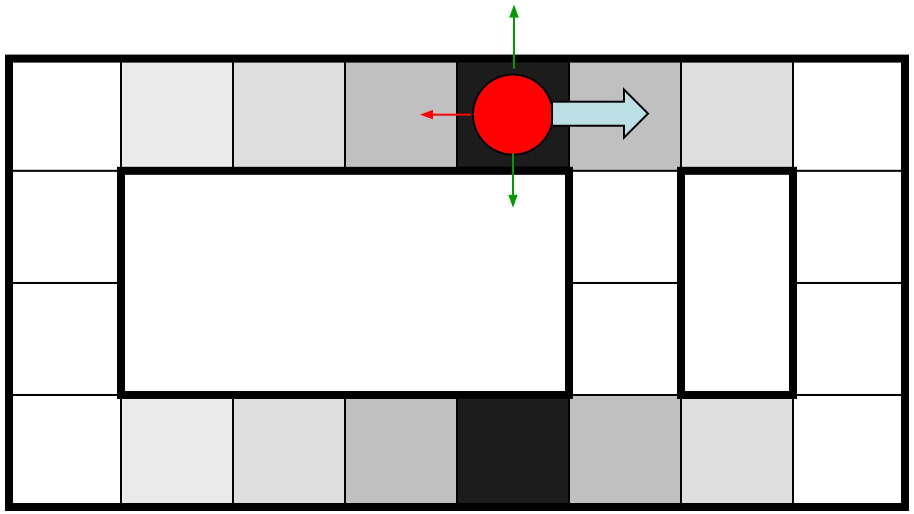
Example: Robot Localization

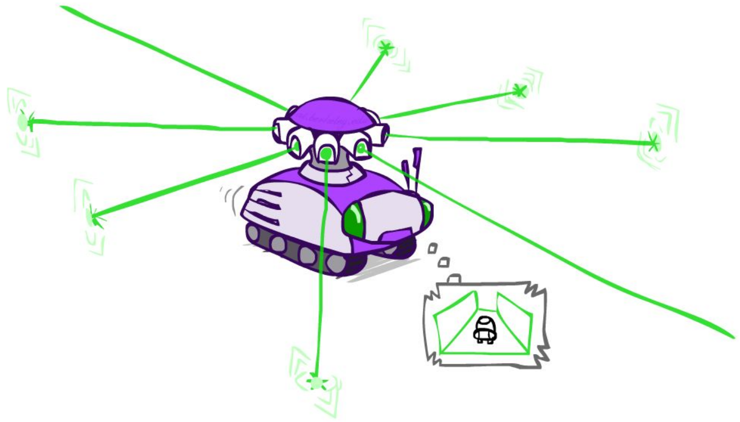


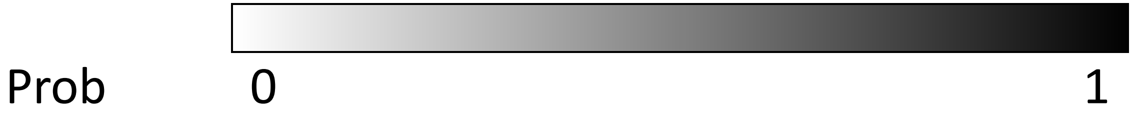
t=3



Example: Robot Localization

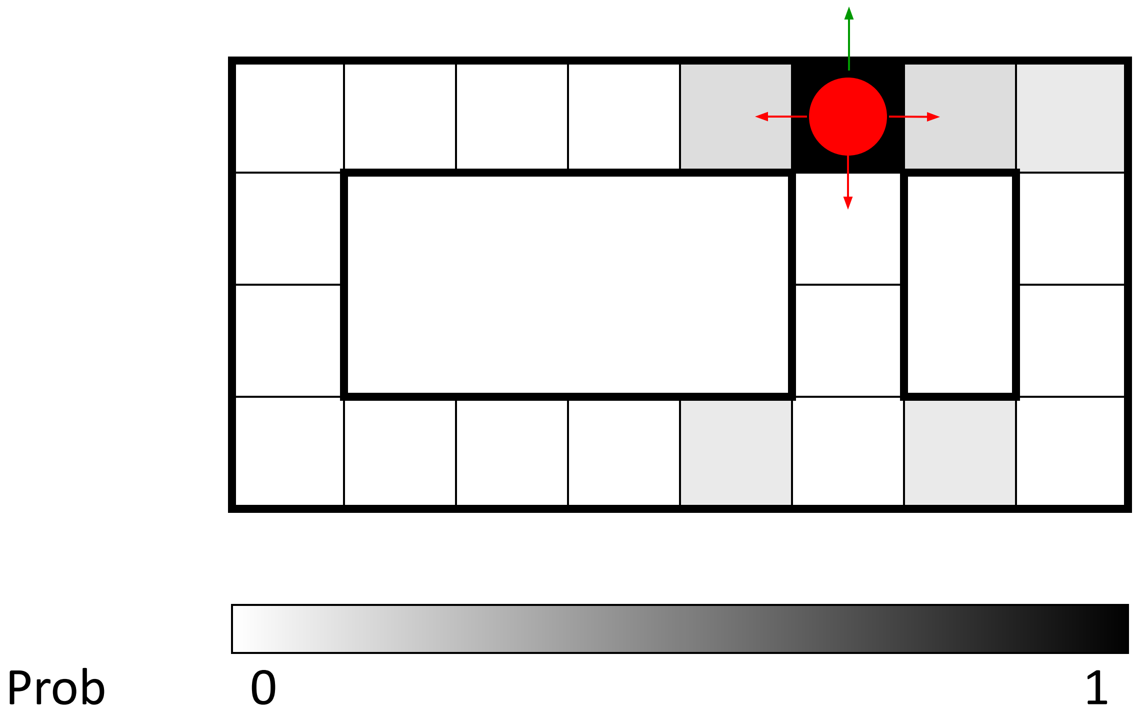


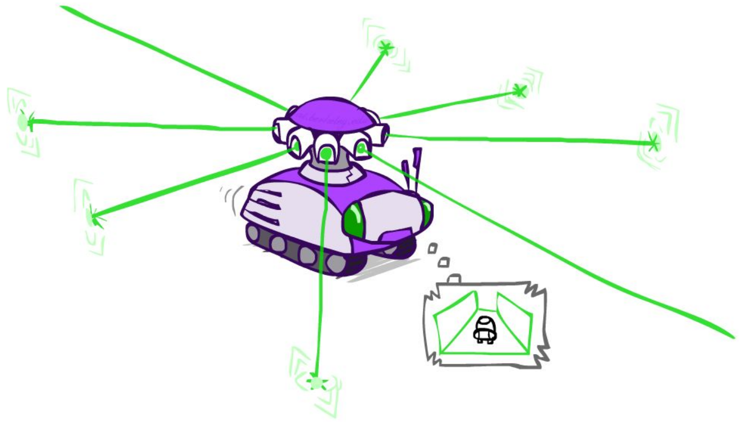




t=4

Example: Robot Localization





t=5

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

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

Apply Bayes' rule

Apply conditional independence

Condition on X_t

Apply conditional independence

Normalize

Update

Predict

Filtering algorithm

- $$P(X_{t+1} | e_{1:t+1}) = \underbrace{\alpha}_{\text{Normalize}} \underbrace{P(e_{t+1} | X_{t+1})}_{\text{Update}} \underbrace{\textstyle\sum_{x_t} P(x_t \mid e_{1:t})\, P(X_{t+1} \mid x_t)}_{\text{Predict}}$$

- $\boldsymbol{f}_{1:t+1} = \textsc{Forward}(\boldsymbol{f}_{1:t}, e_{t+1})$
- 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

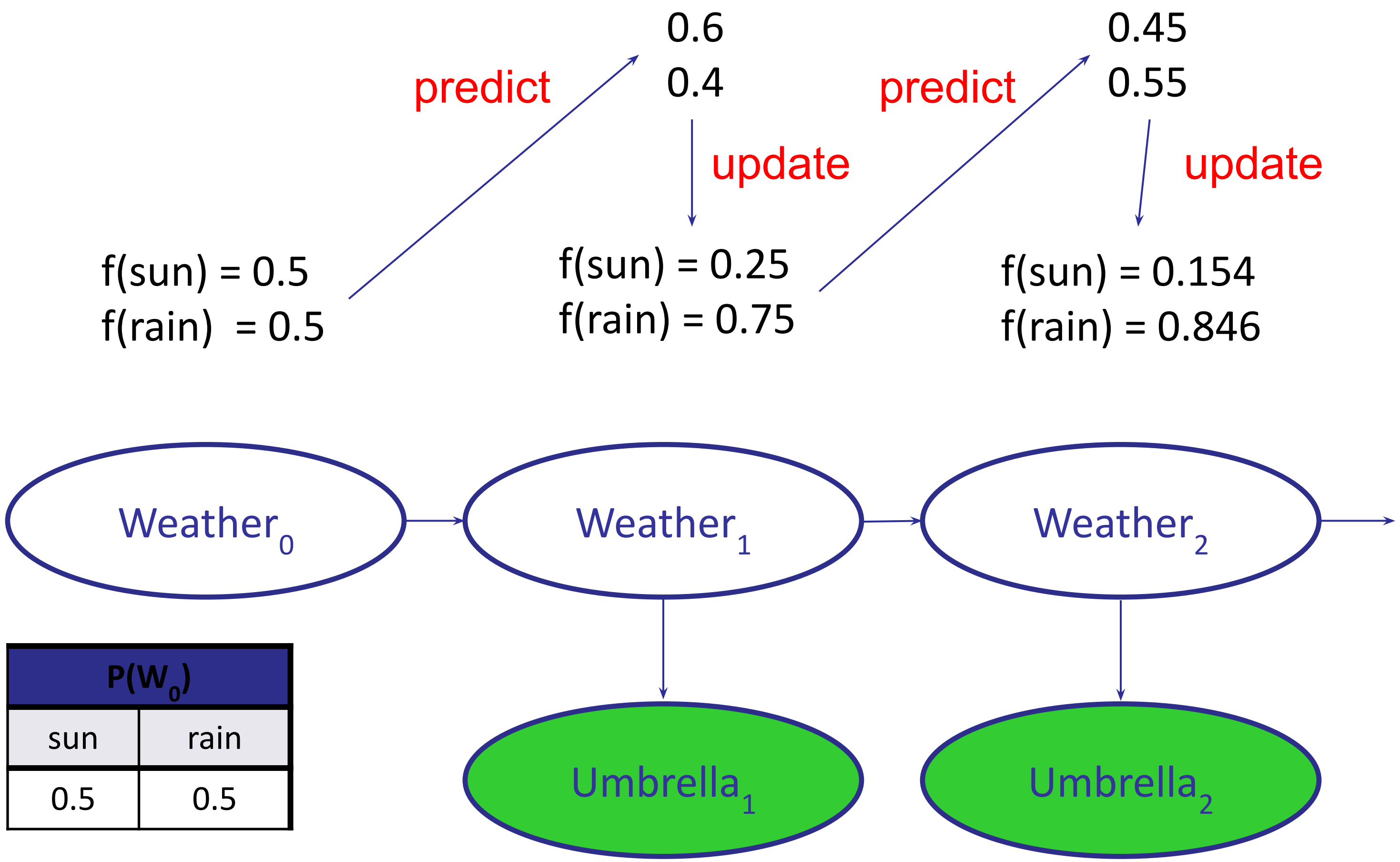
And the same thing in linear algebra

- 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
  - $\boldsymbol{f}_{1:t+1} = \alpha\, O_{t+1} T^{\mathrm{T}} \boldsymbol{f}_{1:t}$

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|W_{t-1})	
	sun	rain
sun	0.9	0.1
rain	0.3	0.7

W_t	P(U_t|W_t)	
	true	false
sun	0.2	0.8
rain	0.9	0.1

Pacman – Hunting Invisible Ghosts with Sonar

[Demo: Pacman – Sonar – No Beliefs(L14D1)]

Video of Demo Pacman – Sonar

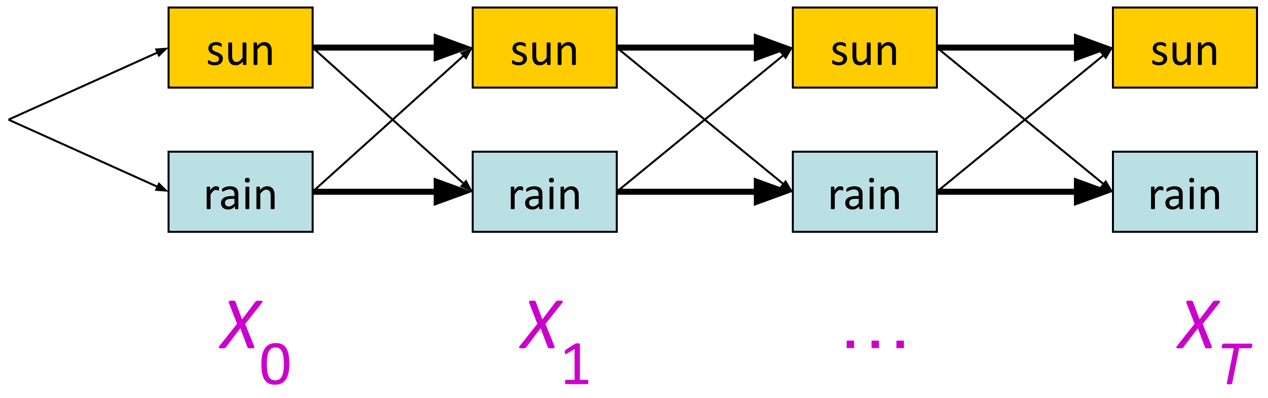
Most Likely Explanation

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 $0 \leq k < t$
  - 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

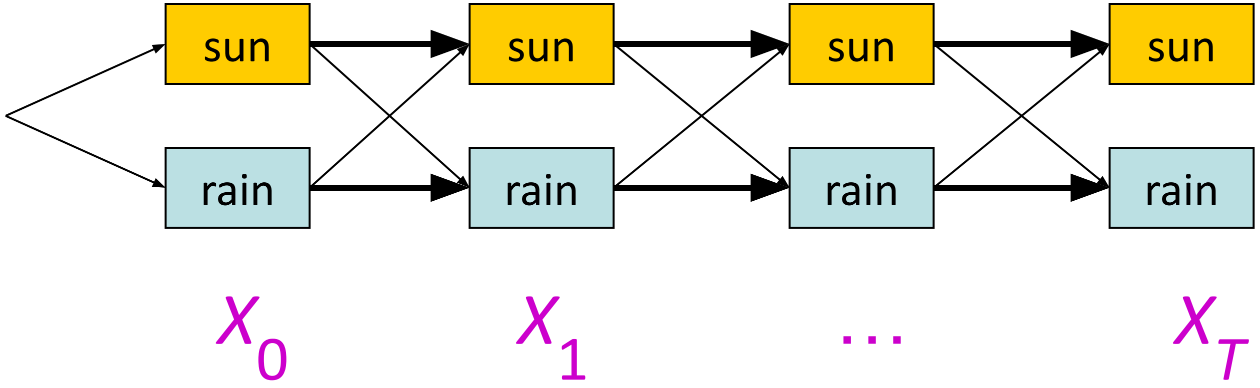
- ***State trellis***: graph of states and transitions over time



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

- 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



Forward Algorithm (sum)

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

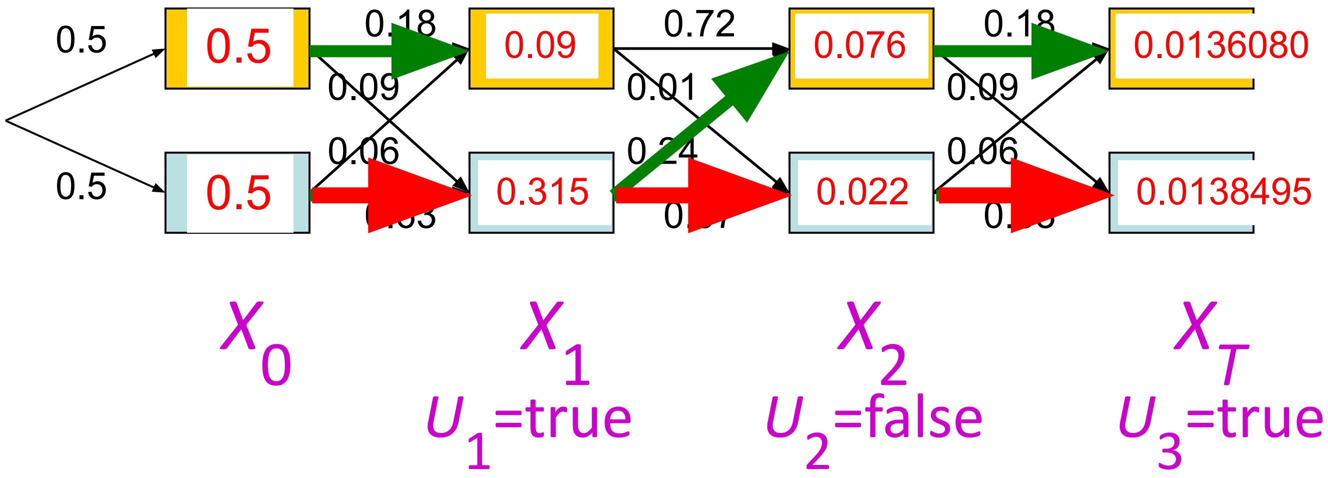
$$\mathbf{f}_{1:t+1} = \text{FORWARD}(\mathbf{f}_{1:t}, e_{t+1})$$
$$= \alpha\, P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1} | x_t)\, \mathbf{f}_{1:t}$$

Viterbi Algorithm (max)

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

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

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

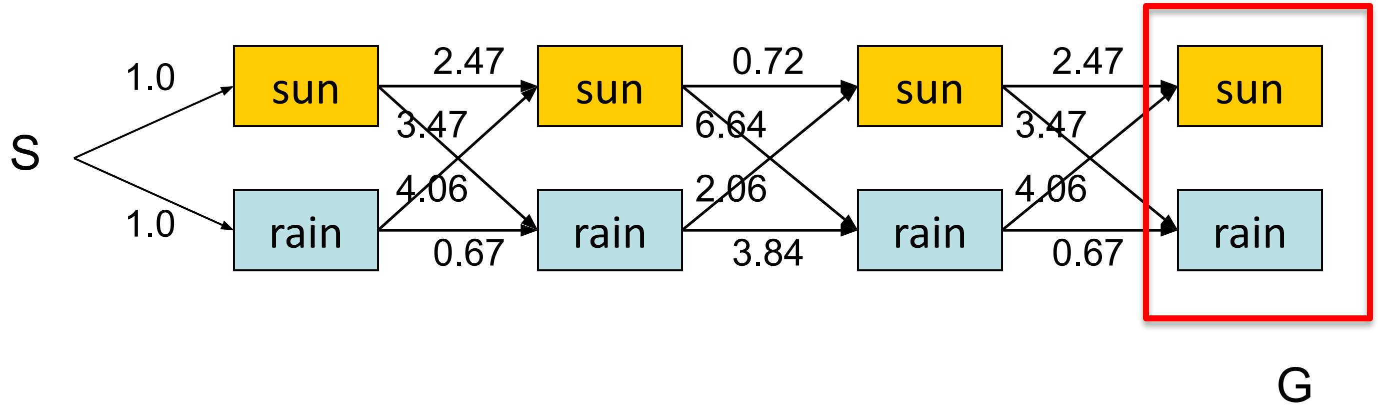
W_t	$P(U_t \mid W_t)$	
	true	false
sun	0.2	0.8
rain	0.9	0.1

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*?