COMS W4701: Artificial Intelligence

Lecture 9: Hidden Markov Models

Tony Dear, Ph.D.

Department of Computer Science

School of Engineering and Applied Sciences

Announcements

- HW4 is available now, due on December 10
- Start times for remaining quizzes are now midnight
- **Probability recitation** tomorrow at 8:45pm EST

- Please check details of final exam and let us know ASAP if any conflicts
- Start: Dec 22, 4:00pm EST. End: Dec 23, 4:00pm EST.

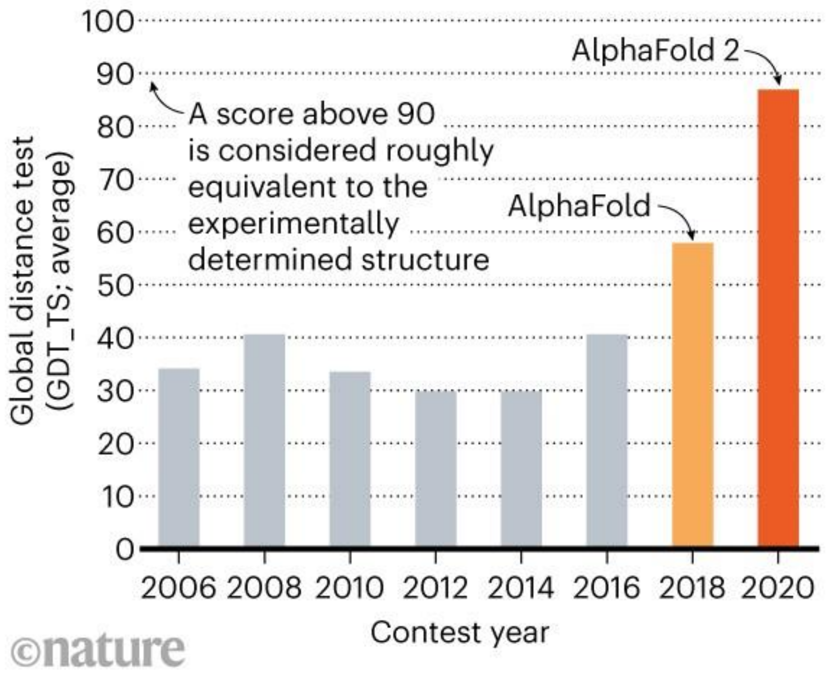
- Courseworks quiz portion: 1.5 hours, similar to post-lecture quizzes
- Jupyter notebook portion: Untimed, similar to homeworks

In the News

- DeepMind's AlphaFold made significant strides in protein structure prediction in biennial contest
- A win for AI and deep learning techniques in biology, medicine, and life sciences
- Accuracy levels close to "gold standard" experimental methods for structure prediction
- Sources: [DeepMind](), [Nature]()

STRUCTURE SOLVER

DeepMind's AlphaFold 2 algorithm significantly outperformed other teams at the CASP14 protein-folding contest — and its previous version's performance at the last CASP.



Today

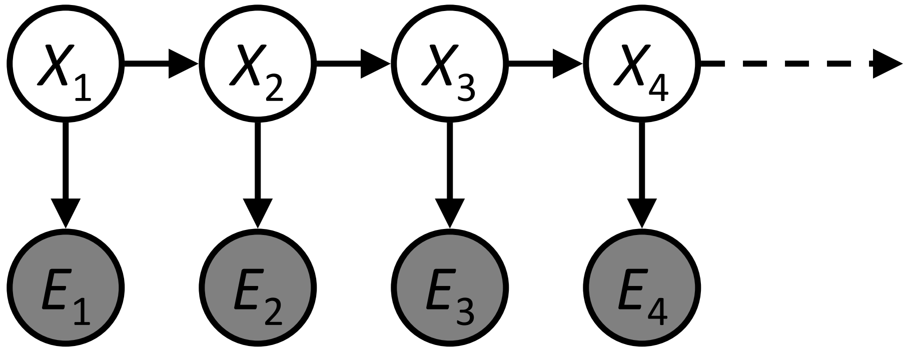
- Hidden Markov models
- State estimation (filtering): Forward algorithm
- Most likely explanation: Viterbi algorithm

Hidden Markov Models

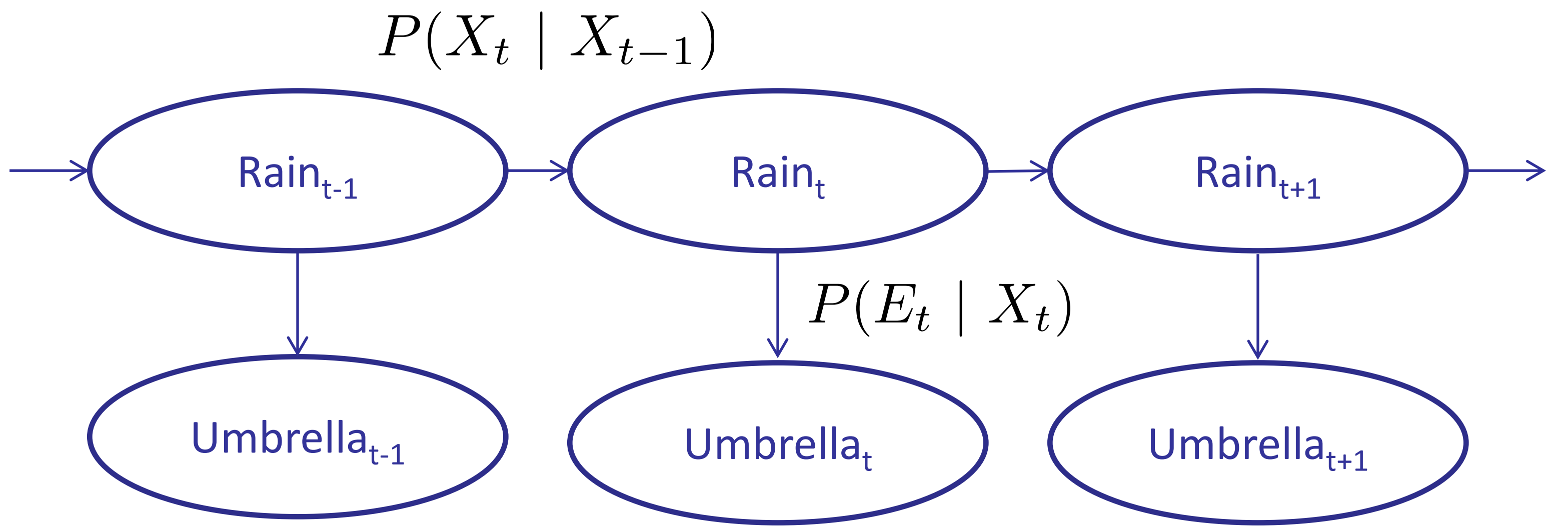
- Last time: Markov chains for dynamic, unobservable environments
- Can’t directly observe state; but can predict how it evolves
- Now let’s suppose we *can* observe indirect *evidence* of states

- **Hidden Markov model**: A Markov process with *hidden* states X_t and *observable* evidence variables E_t

- Initial belief state: $P(X_0)$
- Transition model: $P(X_t|X_{t-1})$
- Observation model: $P(E_t|X_t)$



Example: Weather HMM

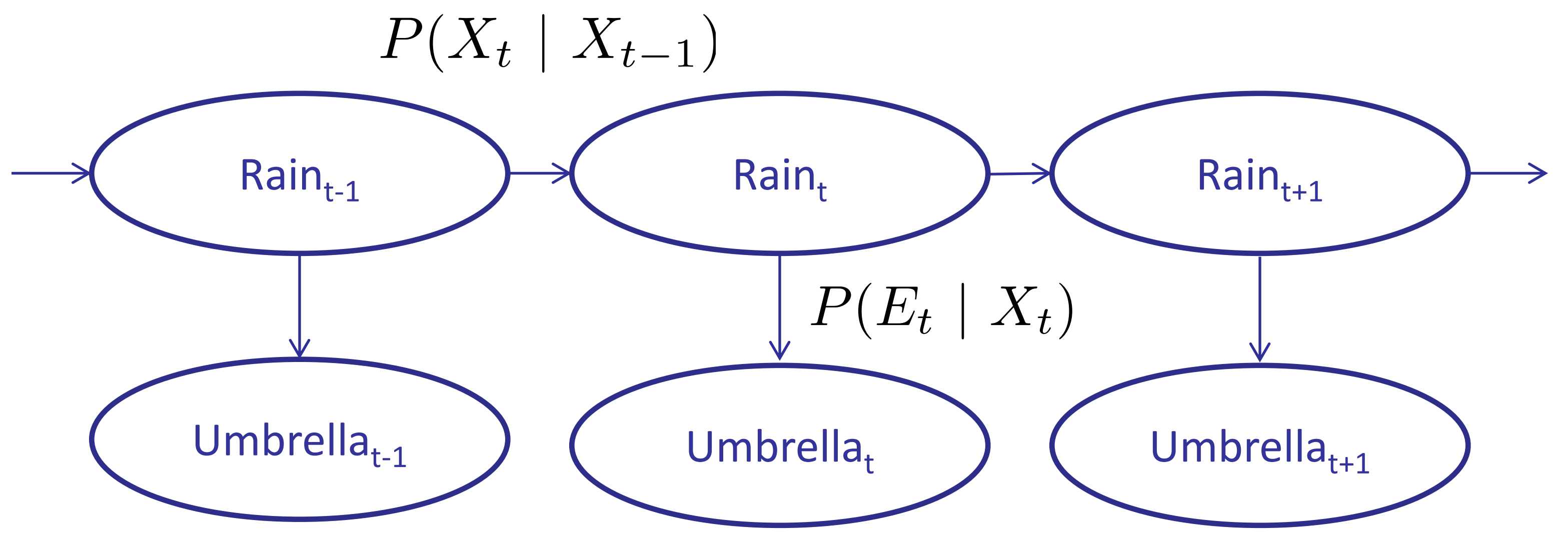


X_{t-1}	X_t	$P(X_t\|X_{t-1})$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

X_t	E_t	$P(E_t\|X_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

- **Stationarity assumption**: Transition and observation models are the same for all *t*
- If we ignore the evidence, this is just a first-order Markov chain

Example: Weather HMM



Suppose we are given

$$P(X_0) = (0.6, 0.4)^T$$

$$E_1 = +u$$

$$T = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$$

X_{t-1}	X_t	$P(X_t \mid X_{t-1})$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

X_t	E_t	$P(E_t \mid X_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

$$P(X_1) = T \cdot P(X_0) = (0.54, 0.46)^T$$

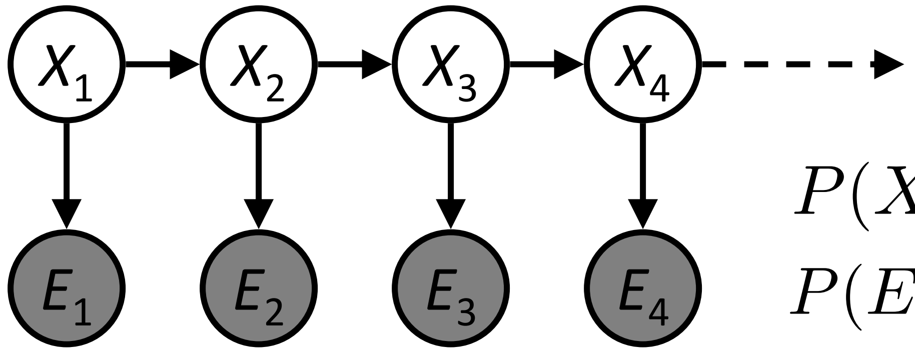
$$P(X_1 \mid e_1) = \frac{P(e_1 \mid X_1) P(X_1)}{P(e_1)}$$

$$= \frac{1}{0.9(0.54) + 0.2(0.46)} \begin{pmatrix} 0.9 \times 0.54 \\ 0.2 \times 0.46 \end{pmatrix} = \begin{pmatrix} 0.841 \\ 0.159 \end{pmatrix}$$

Conditional Independences

- Markov chain independences:

$$X_t \perp\!\!\!\perp X_1, \ldots, X_{t-2} \mid X_{t-1}$$



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

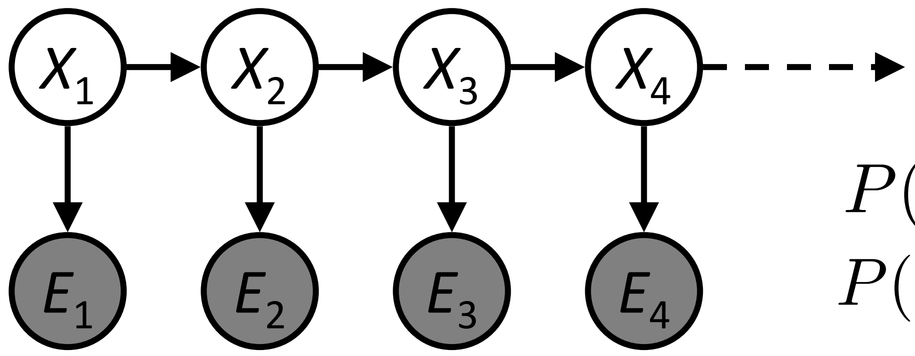
- A state is conditionally independent of past states and evidence given preceding state:

$$X_t \perp\!\!\!\perp X_1, E_1, \ldots, X_{t-2}, E_{t-2}, E_{t-1} \mid X_{t-1}$$

- An observation is conditionally independent of past states and evidence given current state:

$$E_t \perp\!\!\!\perp X_1, E_1, \ldots, X_{t-2}, E_{t-2}, X_{t-1}, E_{t-1} \mid X_t$$

Joint Distribution



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

- General joint distribution:

$$P(X_1, E_1, \ldots, X_T, E_T) = P(X_1)P(E_1|X_1)\prod_{t=2}^{T} P(X_t|X_{t-1})P(E_t|X_t)$$

- Marginal distributions can be found by summing out RVs
- For certain computations we don't even need the entire joint distribution!

HMMs and Inference

- We are generally interested in hidden states X given *observed* evidence e

- **Filtering** (state estimation): Find $P(X_t \mid e_{1:t})$
  - What is the hidden state, given *all evidence to date*?

- **Most likely explanation**: Find $\operatorname{argmax}_{x_{1:t}} P(x_{1:t} \mid e_{1:t})$
  - What is the *sequence* of hidden states best explained by the observed evidence?

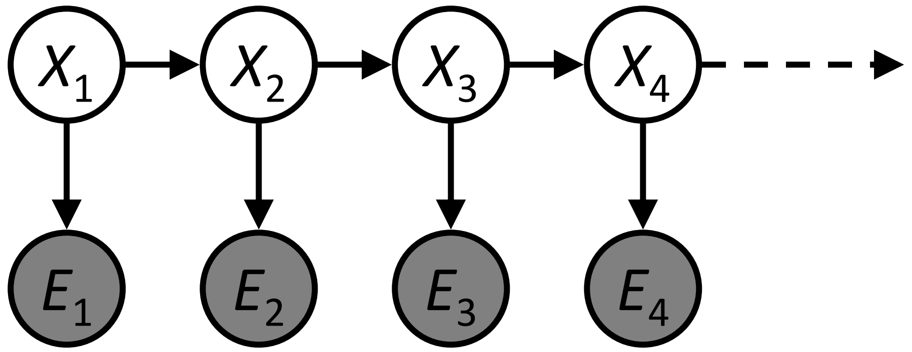
- **Smoothing**: Find $P(X_k \mid e_{1:t})$, for $1 \leq k < t$
  - Use both past and future evidence to *smooth* prediction of a state

State Estimation

- We want to estimate the belief state $P(X_t \mid e_{1:t})$
- We want to compute this recursively

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

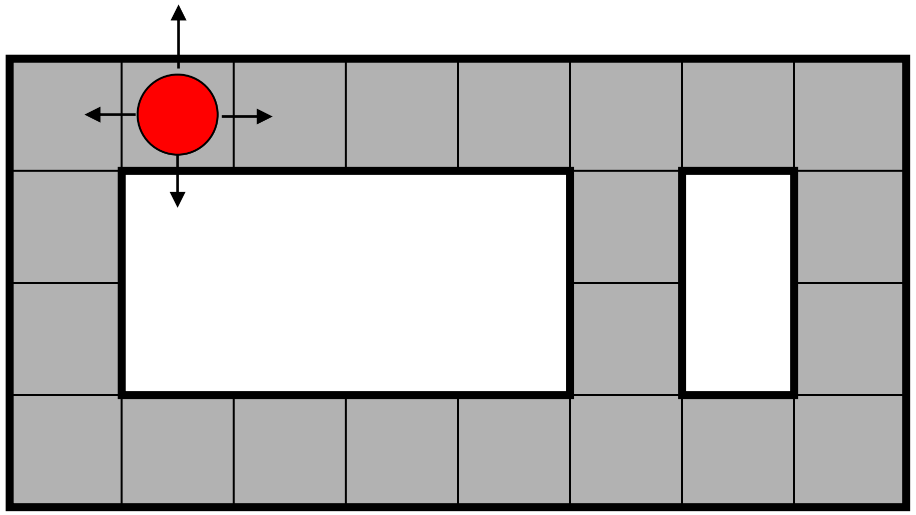
$P(E_t \mid X_t)$

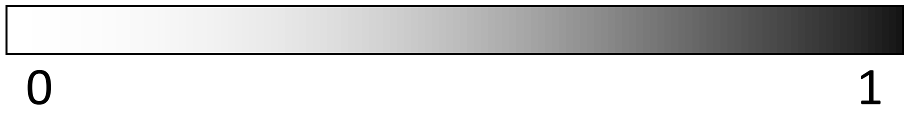


- For each timestep, we update our belief as follows:
- *Elapse* time: Follow the state transition model (same as Markov chains)
- *Observe* evidence: Follow the observation model to account for evidence

Example: Robot Localization

$t = 0$





Example from Michael Pfeiffer

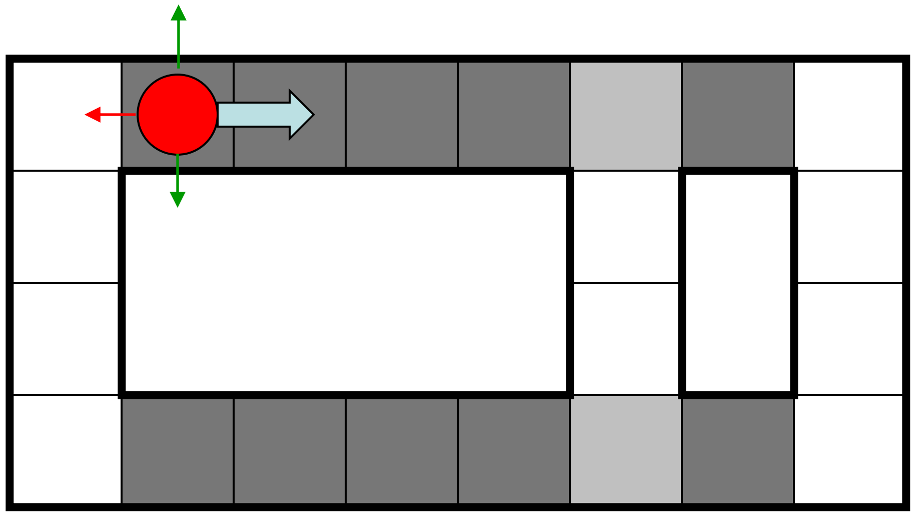
- Hidden state: Robot's true location
- X_t is a RV over 22 possible values

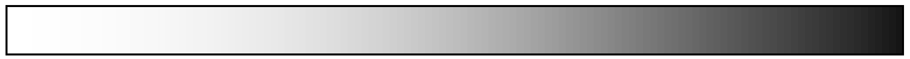
- **Motion** (transition) model is noisy
- Either move in intended direction (more likely) or stay put (less likely)

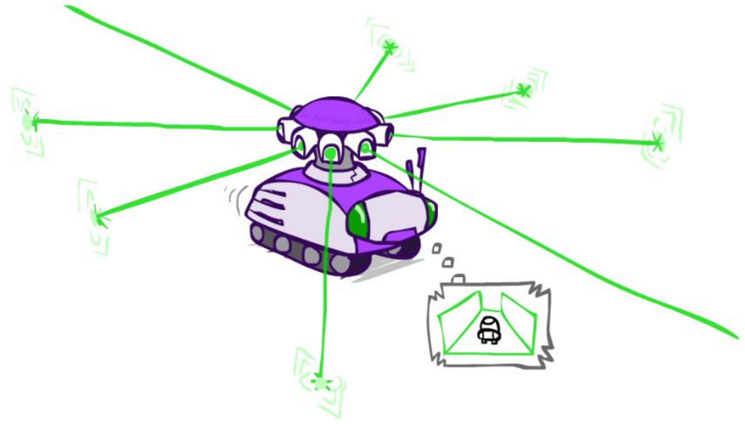
- **Sensor** (observation) model is noisy
- 4-bit binary string indicating presence of wall in each cardinal direction
- At most 1 bit may be an error

Example: Robot Localization

$t = 1$





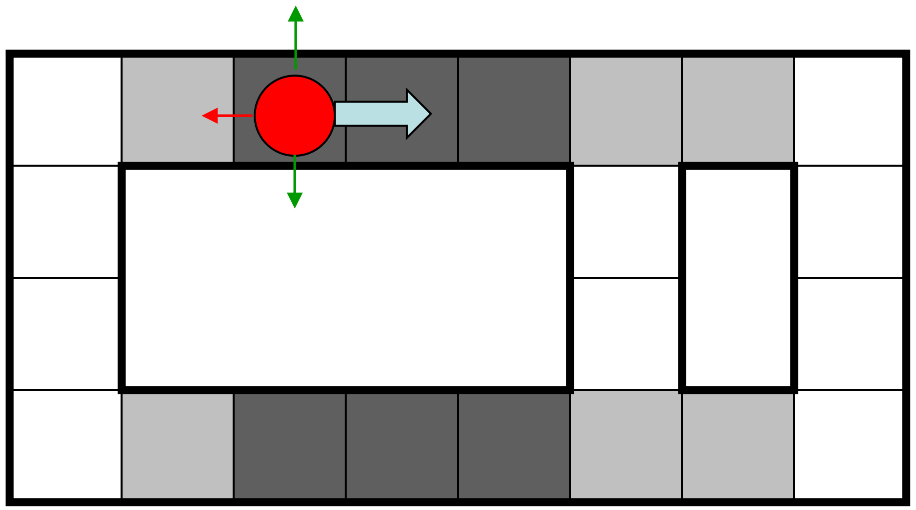


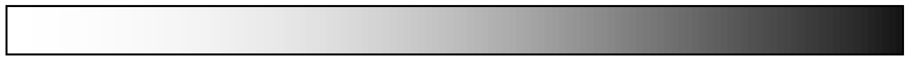
- Robot observes from current location
- Wall above and below, no wall on the left and right

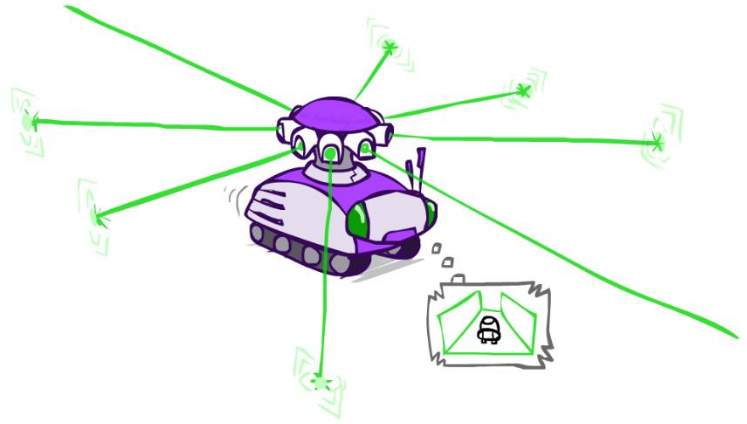
- White locations are ruled out
- Gray locations are all possibilities for robot's true location

Example: Robot Localization

$t = 2$





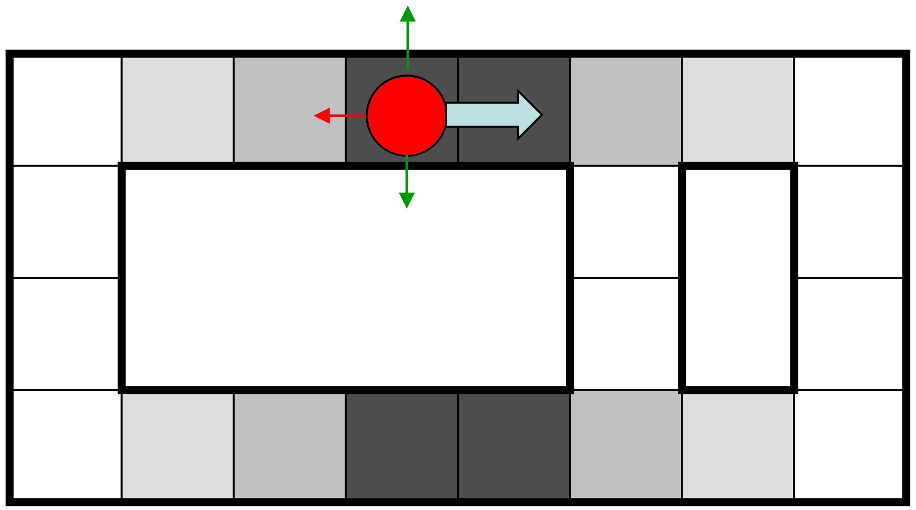


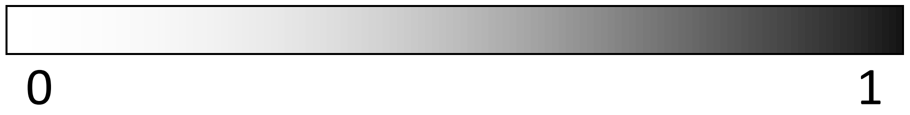
- Robot moves and observes again
- Same observation as before

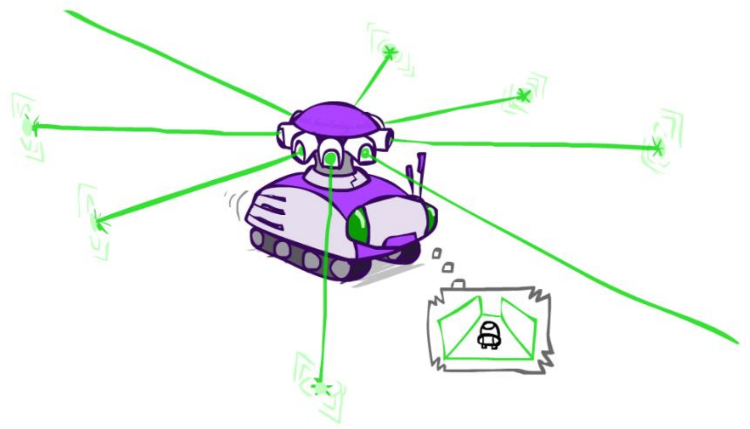
- Light gray cells are less likely to be robot’s location after “moving rightward and observing twice”

Example: Robot Localization

$t = 3$



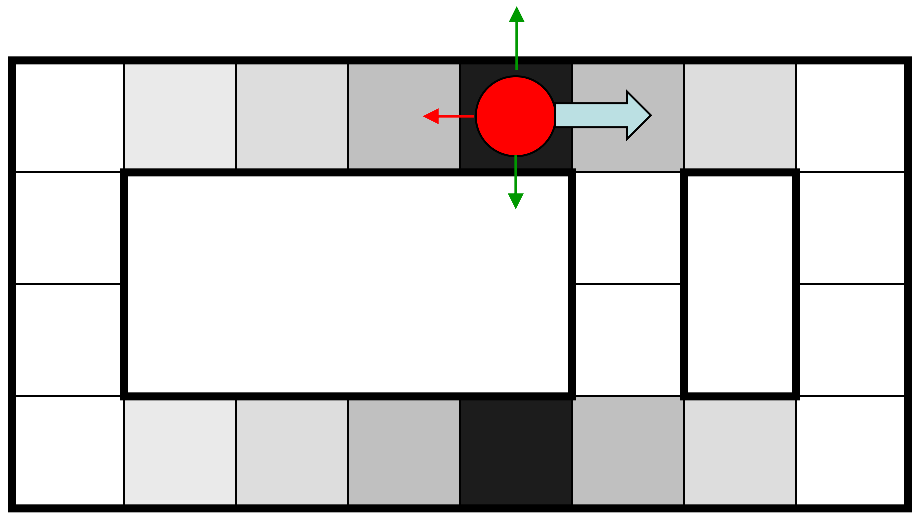


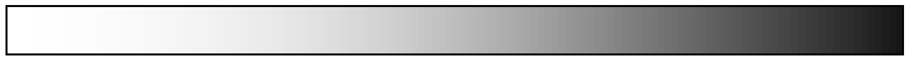


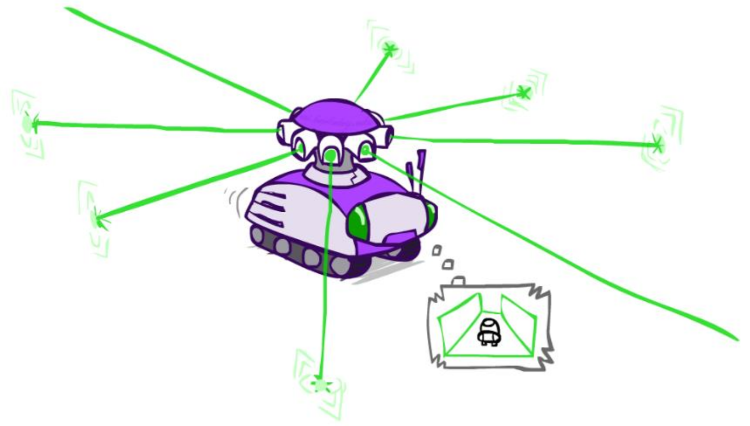
- Robot continues moving, observing, and updating its belief about its location…

Example: Robot Localization

$t = 4$



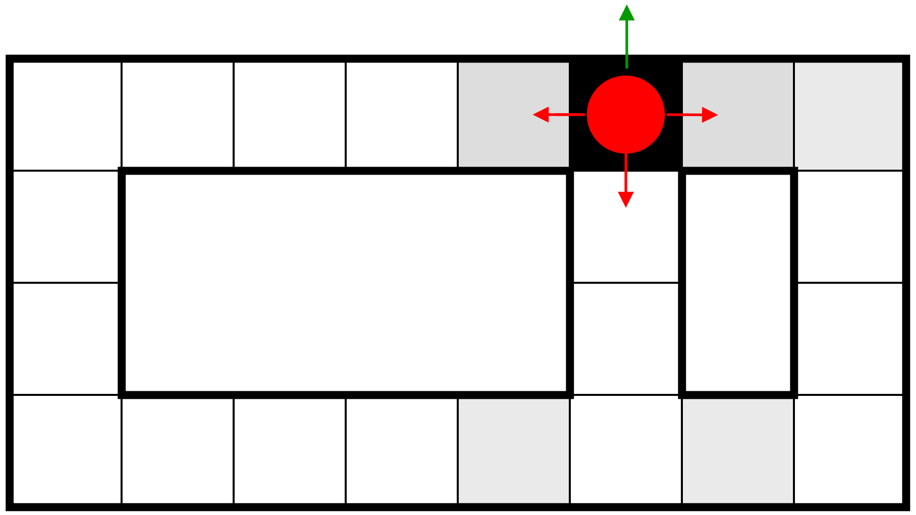


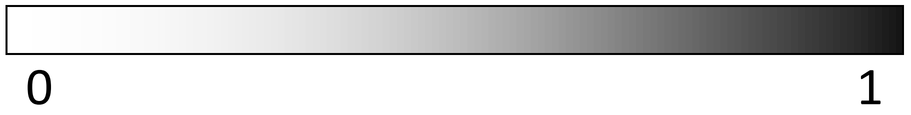


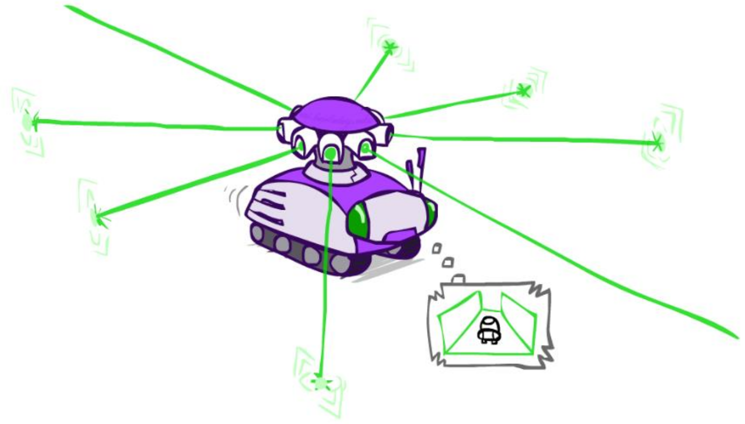
- Robot continues moving, observing, and updating its belief about its location…

Example: Robot Localization

$t = 5$







- We are now very confident about where the robot actually is!

Forward Algorithm

- We have $\boldsymbol{f}_t = P(X_t \mid e_{1:t})$. How to obtain $\boldsymbol{f}_{t+1} = P(X_{t+1} \mid e_{1:t+1})$?

- **Elapse time:**

Transition

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

Conditional independence

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

$$\Rightarrow \quad \boxed{\boldsymbol{f}'_{t+1} = T \boldsymbol{f}_t}$$

- **Observe evidence:**

Observation

Conditional independence

$$P(e_{t+1} \mid X_{t+1}, \cancel{e_{1:t}}) P(X_{t+1} \mid e_{1:t}) = P(X_{t+1}, e_{t+1} \mid e_{1:t})$$

$$\propto_{X_{t+1}} P(X_{t+1} \mid e_{1:t+1})$$

Normalize

$$\Rightarrow \quad \boxed{\boldsymbol{f}_{t+1} \propto_{X_{t+1}} O_{t+1} \boldsymbol{f}'_{t+1}}$$

Normalize

Review: Normalization

- We want to find $P(X_t \mid e_{1:t})$—use def of conditional probability:

$$P(X_t \mid e_{1:t}) = \frac{P(X_t, e_{1:t})}{P(e_{1:t})}$$

- Denominator corresponds to *observed* random variables
- We can compute this, but this is also just a constant (why?)

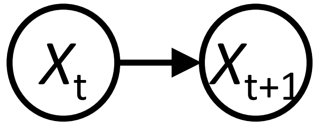
- Since we are computing the entire distribution $P(X_t|e_{1:t})$, we can just normalize $P(X_t, e_{1:t})$:

$$P(X_t \mid e_{1:t}) = \alpha\, P(X_t, e_{1:t}) \propto_{X_t} P(X_t, e_{1:t})$$

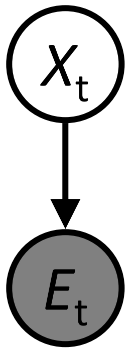
Forward Algorithm

- Forward algorithm takes *constant* space complexity

- Step 1: Elapse time using **transition** model: $\boldsymbol{f}'_{t+1} = T\boldsymbol{f}_t$
- T is a $n \times n$ **transition** matrix, where $T_{ij} = P(X_{t+1} = i | X_t = j)$



- Step 2: Incorporate evidence using **observation** model: $\boldsymbol{f}_{t+1} \propto O_{t+1}\boldsymbol{f}'_{t+1}$
- O_t is a $n \times n$ diagonal **observation** matrix, where $(O_t)_{ii} = P(E_t = e_t | X_t = i)$

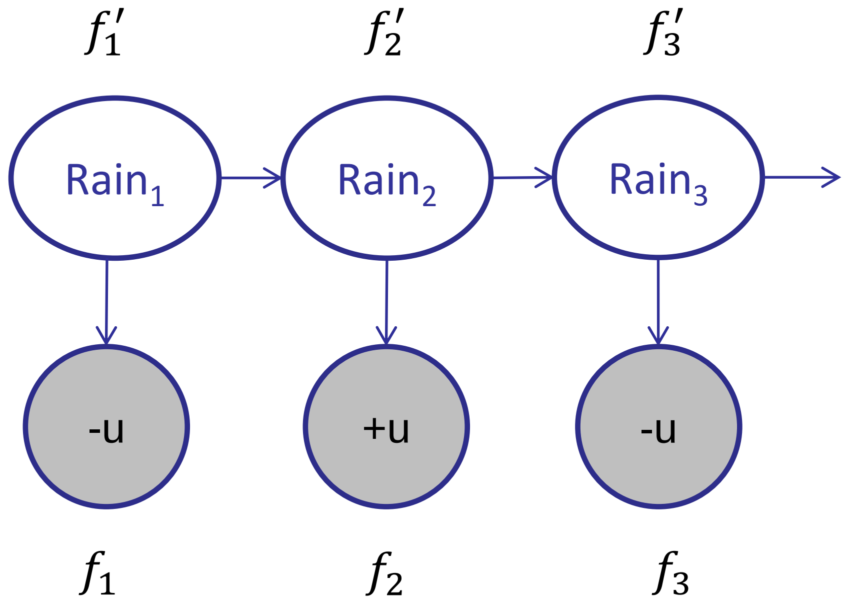


- Observation model gives rise to $|E|$ unique matrices
- We only use one per timestep since e_t is observed

Example: Weather HMM

$$T = \begin{array}{cc|c} 0.7 & 0.3 & +r \\ 0.3 & 0.7 & -r \\ \hline +r & -r & \end{array} \qquad O_1 = O_3 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.8 \end{pmatrix} \qquad O_2 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$$

$$\boxed{\begin{array}{l} \boldsymbol{f}'_{t+1} = T\boldsymbol{f}_t \\ \boldsymbol{f}_{t+1} \propto_{X_{t+1}} O_{t+1}\boldsymbol{f}'_{t+1} \end{array}}$$



Suppose $f_0 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$

$$f'_1 = Tf_0 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \qquad f_1 \propto O_1 f'_1 = \begin{pmatrix} 0.05 \\ 0.4 \end{pmatrix} \propto \begin{pmatrix} 0.11 \\ 0.89 \end{pmatrix}$$

$$f'_2 = Tf_1 = \begin{pmatrix} 0.34 \\ 0.66 \end{pmatrix} \qquad f_2 \propto O_2 f'_2 = \begin{pmatrix} 0.31 \\ 0.13 \end{pmatrix} \propto \begin{pmatrix} 0.7 \\ 0.3 \end{pmatrix}$$

$$f'_3 = Tf_2 = \begin{pmatrix} 0.58 \\ 0.42 \end{pmatrix} \qquad f_3 \propto O_3 f'_3 = \begin{pmatrix} 0.06 \\ 0.34 \end{pmatrix} \propto \begin{pmatrix} 0.15 \\ 0.85 \end{pmatrix}$$

Most Likely Sequence

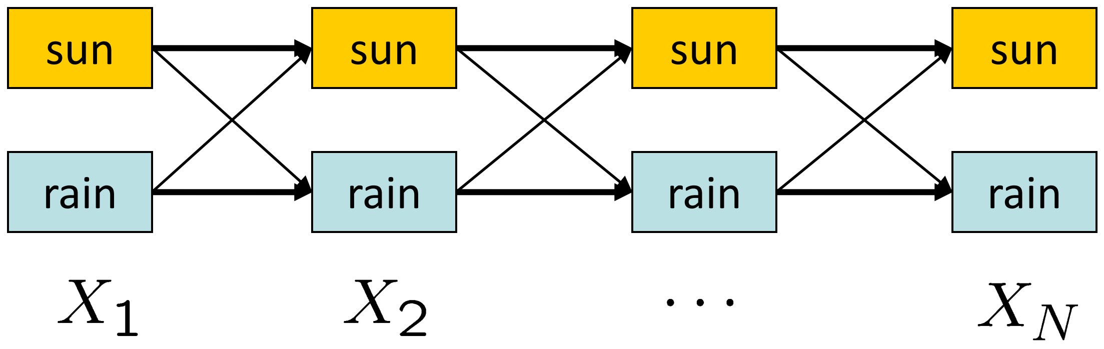
- What is the most likely *sequence* of states given a *sequence* of evidence?
- Argmax of the conditional $P(X_{1:t} \mid e_{1:t})$, or equivalently the joint $P(X_{1:t}, e_{1:t})$
- We **cannot** just run forward algorithm for each state and argmax separately!
- Most likely individual states may differ from that of the most likely sequence

X_1	X_2	$P(X_1, X_2)$
$+x$	$+x$	0.35
$+x$	$-x$	0.25
$-x$	$+x$	0.1
$-x$	$-x$	0.3

argmax $P(X_1) = +x$
argmax $P(X_2) = -x$

BUT argmax $P(X_1, X_2) = (+x, +x)$

State Trellis Diagram

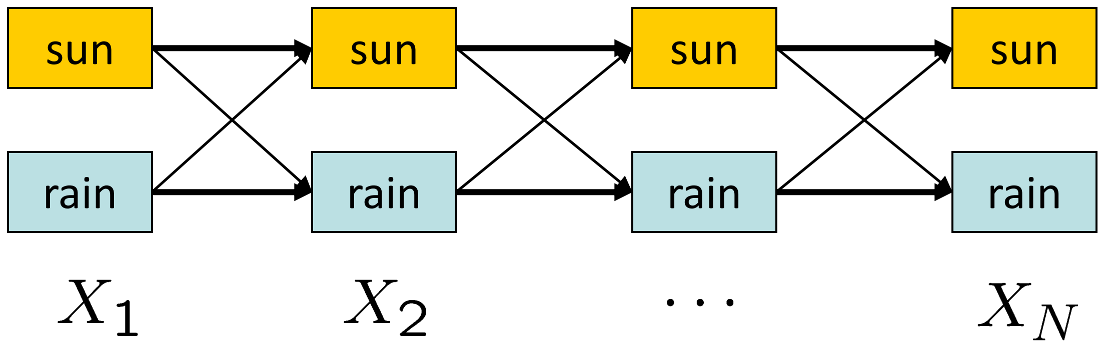


- A state sequence is a *path* through a **state trellis diagram**
- Each arc $x_{t-1} \to x_t$ has weight $P(e_t \mid x_t)P(x_t \mid x_{t-1})$

- Maximizing joint probability of state sequence = maximizing *product* of arc weights
- Problem: Number of possible paths grows exponentially with time

- Idea: Best path to state x_t *includes* best path to state x_{t-1}, followed by a transition

Most Likely Sequence



- Example: Suppose (sun, sun, rain, sun) is most likely sequence leading to $X_4 = \text{sun}$
- "Probability" is given by product of weights $w_{ss}w_{sr}w_{rs}$

- Must be the case that (sun, sun, rain) is most likely sequence leading to X_3 = rain

- If any other sequence produces a larger "probability", that would contradict the original assertion of most likely sequence to $X_4 = \text{sun}$

Most Likely Joint Probabilities

- Define $\boldsymbol{m}_t = \max_{x_1 \ldots x_{t-1}} P(x_{1:t-1}, X_t, e_{1:t})$ as a distribution over X_t
- Each $\boldsymbol{m}_t(x_t)$ is a joint probability of most likely sequence up to x_t

- Ex: Suppose $P(X_1) = (0.5, 0.5)^T$. Then $\boldsymbol{m}_1 = P(X_1, e_1) = (0.05, 0.4)^T$
- $\boldsymbol{m}_2 = \max_{x_1} P(x_1, X_2, e_{1:2})$

X_1	X_2	$P(x_1, x_2, e_{1:2}) = P(x_1)P(e_1 \mid x_1)P(x_2 \mid x_1)P(e_2 \mid x_2)$
+r	+r	$.05 \times .7 \times .9 = .0315$
+r	-r	$.05 \times .3 \times .2 = .003$
-r	+r	$.4 \times .3 \times .9 = .108$
-r	-r	$.4 \times .7 \times .2 = .056$

$\Rightarrow$

X_2	$\boldsymbol{m}_2$
+r	.108
-r	.056

$$T = \begin{array}{cc} \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} & \begin{array}{c} +r \\ -r \end{array} \\ \begin{array}{cc} +r & -r \end{array} & \end{array}$$

$$O_1 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.8 \end{pmatrix}$$

$$O_2 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$$

Viterbi Algorithm

- We can find $\boldsymbol{m}_{t+1}$ from $\boldsymbol{m}_t$ in a manner similar to the forward algorithm:

Conditional independence

Conditional independence

$$\boldsymbol{m}_{t+1} = \max_{x_1 \dots x_t} P(x_{1:t}, X_{t+1}, e_{1:t+1})$$

$$= \max_{x_1 \dots x_t} P(x_t, x_{1:t-1}, e_{1:t}) P(X_{t+1} \mid x_t, \cancel{x_{1:t-1}}, \cancel{e_{1:t}}) P(e_{t+1} \mid X_{t+1}, \cancel{x_t}, \cancel{x_{1:t-1}}, \cancel{e_{1:t}})$$

$$= \max_{x_1 \dots x_t} P(x_{1:t-1}, x_t, e_{1:t}) P(X_{t+1} \mid x_t) P(e_{t+1} \mid X_{t+1})$$

$$= \max_{x_t} P(e_{t+1} \mid X_{t+1}) P(X_{t+1} \mid x_t) \max_{x_1 \dots x_{t-1}} P(x_{1:t-1}, x_t, e_{1:t})$$

$$= \underbrace{P(e_{t+1} \mid X_{t+1})}_{\text{Observation}} \max_{x_t} \underbrace{P(X_{t+1} \mid x_t)}_{\text{Transition}} \boldsymbol{m}_t(x_t)$$

Same as forward algorithm but replace sum with max!

Viterbi Algorithm: Forward Pass

- **Elapse time:** Instead of usual matrix-vector multiplication, replace sum in the row-column dot product with a *max*

$$T = \begin{array}{cc|c} 0.7 & 0.3 & +r \\ 0.3 & 0.7 & -r \\ +r & -r & \end{array}$$

$$\boldsymbol{m}'_{t+1}(x_{t+1}) = \max_{x_t} P(x_{t+1}|x_t)\boldsymbol{m}_t(x_t)$$

Suppose $\boldsymbol{m}_0 = P(X_0) = \begin{pmatrix}0.5\\0.5\end{pmatrix}$

$$m'_1 = \max_{x_0} P(X_1, x_0) = \begin{pmatrix}\max(\mathbf{0.7}(\mathbf{0.5}), 0.3(0.5)) \\ \max(0.3(0.5), \mathbf{0.7}(\mathbf{0.5}))\end{pmatrix} = \begin{pmatrix}0.35\\0.35\end{pmatrix}$$

X_0	$\boldsymbol{m}_0$
+r	0.5
-r	0.5

X_1	$\boldsymbol{m}'_1 = \max_{x_0} P(X_1, x_0)$
+r	$P(X_0 = +r, X_1 = +r) = 0.35$
-r	$P(X_0 = -r, X_1 = -r) = 0.35$

- **Observe evidence:** No need to normalize (why?) $\quad \boldsymbol{m}_{t+1} = P(e_{t+1} \mid X_{t+1})\boldsymbol{m}'_{t+1}$

Example: Weather HMM

$$T = \begin{array}{cc} \begin{pmatrix}0.7 & 0.3\\ 0.3 & 0.7\end{pmatrix} & \begin{matrix}+r\\ -r\end{matrix}\\ \begin{matrix}+r & \quad -r\end{matrix} & \end{array}$$

$$O_1 = O_3 = \begin{pmatrix}0.1 & 0\\ 0 & 0.8\end{pmatrix} \; -u$$

$$O_2 = \begin{pmatrix}0.9 & 0\\ 0 & 0.2\end{pmatrix} \; +u$$

$$\begin{tikzcd} m_1 & m_2 & m_3 \\ \boxed{0.035} \arrow[r] \arrow[dr] & \boxed{.0756} \arrow[r] \arrow[dr] & \boxed{.00529} \\ \boxed{0.28} \arrow[r] \arrow[ur] & \boxed{.0392} \arrow[r] \arrow[ur] & \boxed{.02195} \end{tikzcd}$$

Suppose $m_0 = \begin{pmatrix}0.5\\ 0.5\end{pmatrix}$

$$m_1' = \begin{pmatrix}\max(\mathbf{0.7}(\mathbf{0.5}), 0.3(0.5))\\ \max(0.3(0.5), \mathbf{0.7}(\mathbf{0.5}))\end{pmatrix} = \begin{pmatrix}0.35\\ 0.35\end{pmatrix} \qquad m_1 = O_1 m_1' = \begin{pmatrix}0.035\\ 0.28\end{pmatrix}$$

$$m_2' = \begin{pmatrix}\max(0.7(.035), \mathbf{0.3}(\mathbf{.28}))\\ \max(0.3(.035), \mathbf{0.7}(\mathbf{.28}))\end{pmatrix} = \begin{pmatrix}.084\\ .196\end{pmatrix} \qquad m_2 = O_2 m_2' = \begin{pmatrix}.0756\\ .0392\end{pmatrix}$$

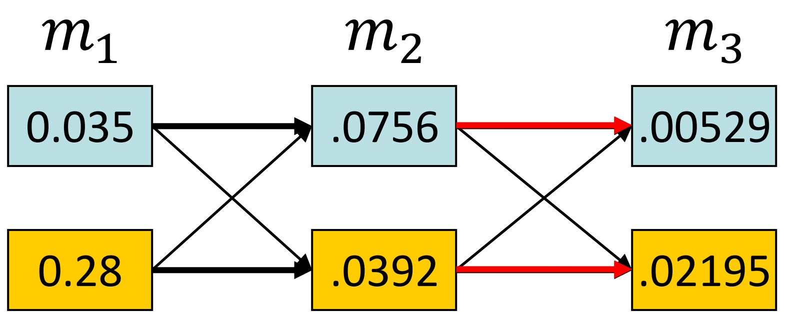
$$m_3' = \begin{pmatrix}\max(\mathbf{0.7}(\mathbf{.0756}), 0.3(.0392))\\ \max(0.3(.0756), \mathbf{0.7}(\mathbf{.0392}))\end{pmatrix} = \begin{pmatrix}.05292\\ .02744\end{pmatrix} \qquad m_3 = O_3 m_3' = \begin{pmatrix}.005292\\ .021952\end{pmatrix}$$

Viterbi Algorithm: Backward Pass

- We still need the likeliest state sequence
- Recall that $\boldsymbol{m}_T = \max_{x_1 \dots x_{T-1}} P(x_{1:T-1}, X_t, e_{1:T})$
- So $X_T = \operatorname{argmax} \boldsymbol{m}_T$, and we need x_T's predecessor x_{T-1}, and its predecessor x_{T-2}, …
- Solution: Record *pointers* to most likely prior state using argmax during forward pass

$$Pointer_{t+1}(x_{t+1}) = \operatorname*{argmax}_{x_t} P(x_{t+1}|x_t)\boldsymbol{m}_t(x_t)$$

- After computing all $\boldsymbol{m}_t$, perform *backward pass* by following pointers back to x_1 and extract most likely states!

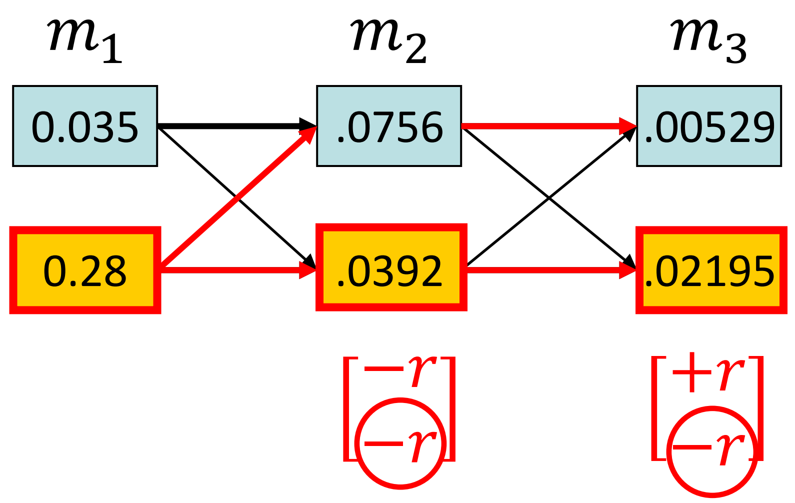


$$m_3' = \begin{pmatrix} \max(\mathbf{0.7}(\mathbf{.0756}), 0.3(.0392)) \\ \max(0.3(.0756), \mathbf{0.7}(\mathbf{.0392})) \end{pmatrix}$$

$$Pointer_3(x_3 = +r) = +r$$
$$Pointer_3(x_3 = -r) = -r$$

Example: Weather HMM

- Viterbi consists of two passes over observations
- *Forward* pass: Compute all m_t and *pointers*
- *Backward* pass: Follow pointers starting from $\operatorname{argmax} m_T$ back to x_1 to extract state sequence



m_1
m_2
m_3
0.035
.0756
.00529
0.28
.0392
.02195
$\begin{bmatrix}-r\\-r\end{bmatrix}$
$\begin{bmatrix}+r\\-r\end{bmatrix}$

$$m'_1 = \begin{pmatrix}\max(\mathbf{0.7}(\mathbf{0.5}), 0.3(0.5))\\ \max(0.3(0.5), \mathbf{0.7}(\mathbf{0.5}))\end{pmatrix} = \begin{pmatrix}0.35\\0.35\end{pmatrix}$$

$$m_1 = O_1 m'_1 = \begin{pmatrix}0.035\\0.28\end{pmatrix}$$

$$m'_2 = \begin{pmatrix}\max(0.7(.035), \mathbf{0.3}(\mathbf{.28}))\\ \max(0.3(.035), \mathbf{0.7}(\mathbf{.28}))\end{pmatrix} = \begin{pmatrix}.084\\.196\end{pmatrix}$$

$$m_2 = O_2 m'_2 = \begin{pmatrix}.0756\\.0392\end{pmatrix}$$

$$m'_3 = \begin{pmatrix}\max(\mathbf{0.7}(\mathbf{.0756}), 0.3(.0392))\\ \max(0.3(.0756), \mathbf{0.7}(\mathbf{.0392}))\end{pmatrix} = \begin{pmatrix}.05292\\.02744\end{pmatrix}$$

$$m_3 = O_3 m'_3 = \begin{pmatrix}.005292\\.021952\end{pmatrix}$$

Backward pointers:
$\operatorname{argmax}_{x_t} \boldsymbol{m}_{t+1}(x_{t+1})$

Most likely sequence:
$(-r, -r, -r)$

Underflow Issues

- The $\boldsymbol{m}_t$ messages computed by Viterbi are not probability distributions!
  - Values do not sum to 1
- In fact, they get smaller in each successive iteration due to reweighting
- Problem: If t is large, values will quickly underflow in a program
- Solution: Renormalize every once in a while (or use log probabilities)
  - We only care about sequence (argmax), so multiplying by constant won't change relative maxes

More Inference

- Forward algorithm has linear time and constant space complexity
- Viterbi algorithm has linear time *and* linear space complexity

- Applications: Digital signals, speech recognition, bioinformatics, finance

- Forward algorithm can be combined with a *backward algorithm* to perform smoothing
- Smoothing can then be used to *learn* unknown HMM model parameters using the **Baum-Welch algorithm**

Summary

- Hidden Markov models incorporate hidden states that evolve according to a transition model and evidence generated by an observation model
- Useful for processes that evolve over time or space

- State estimation: Given a bunch of evidence, what is the current state distribution?

- Viterbi: Given a bunch of evidence, what is the most likely sequence of states?

- Both algorithms involve an "elapse time" step using transition model and a "observe evidence" step using observation model