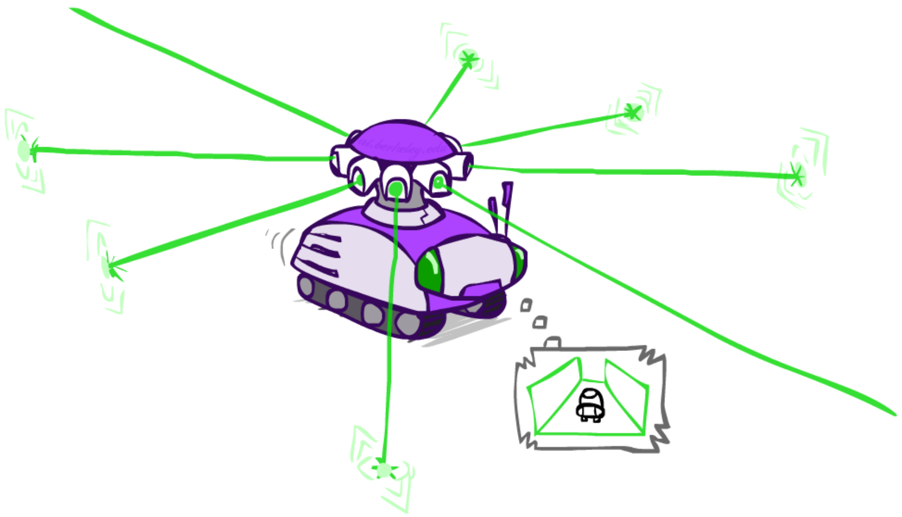
COMS W4701: Artificial Intelligence

Lecture 17: Inference in Hidden Markov Models



Instructor: Tony Dear

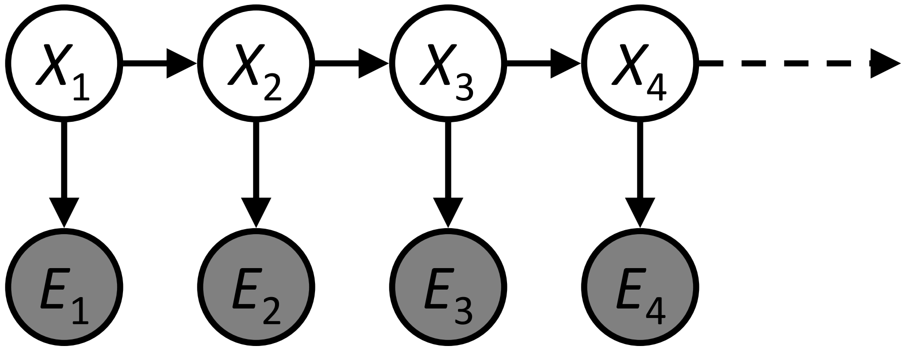
*Lecture materials derived from UC Berkeley's AI course at ai.berkeley.edu

HMM Conditional Independences

- Markov chain independences:

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

$$P(E_t \mid X_t)$$



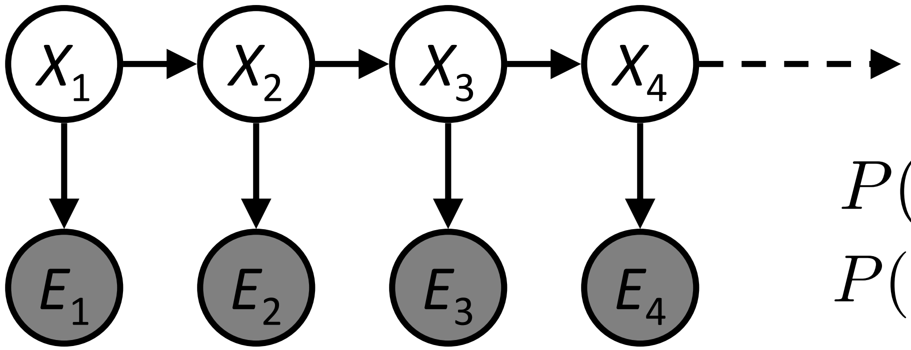
- A state is conditionally independent of past states and evidence given preceding state:

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

- An emission is conditionally independence of past states and evidence given current state:

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

HMM 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!

Applications of HMMs

- Speech recognition
  - Observations: Acoustic signals / waveforms
  - States: Positions in words
- Machine translation
  - Observations: Words to be translated
  - States: Translation options
- Robot tracking
  - Observations: Range readings
  - States: Positions on a map

Today

- Inference tasks in hidden Markov models
- State estimation (filtering): Forward algorithm
- Most likely explanation: Viterbi algorithm

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 that best explains the observed evidence?

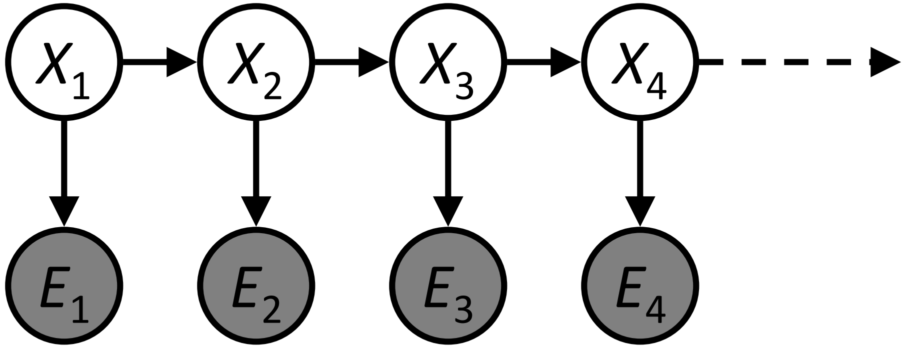
- **Smoothing**: Find $P(X_k \mid e_{1:t})$, for $1 \leq k < t$
  - Use both past and future evidence to *smooth* prediction of a state

Inference: State Estimation

- Idea: Track a *belief state* over time: $P(X_t \mid e_{1:t})$
- We want to compute this recursively (constant time)

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

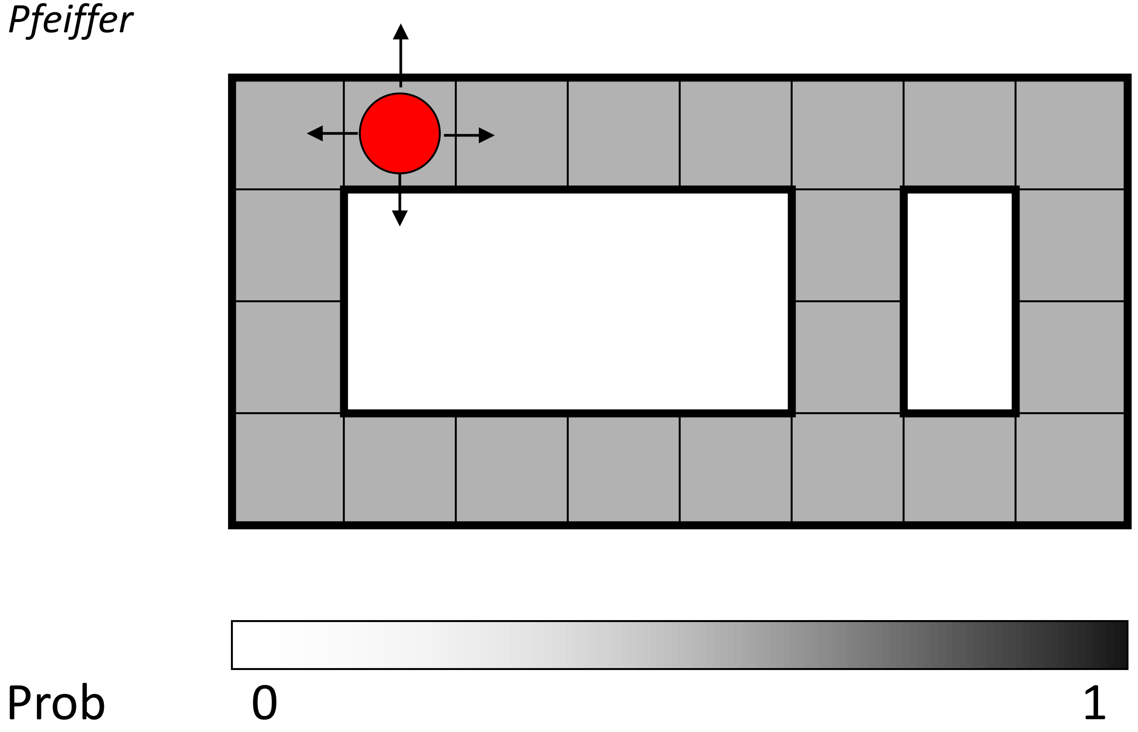
$P(E_t \mid X_t)$

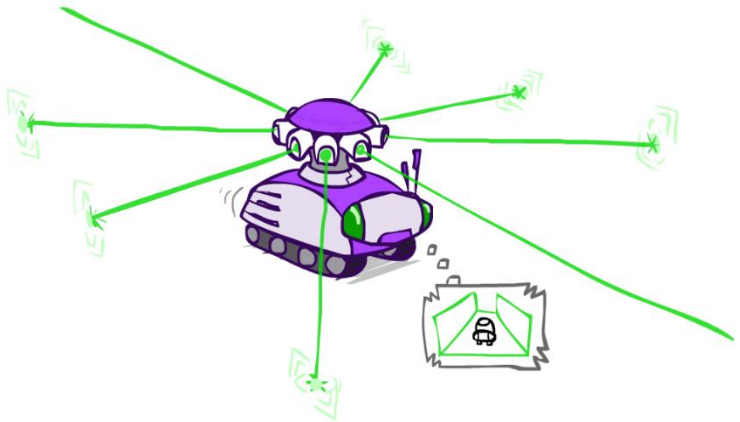


- For each timestep, we update our belief as follows:
- *Elapse* time: Follow the state transition model (same as Markov chains)
- *Observe* evidence: Follow the emissions model to update belief

Example: Robot Localization

Example from Michael Pfeiffer

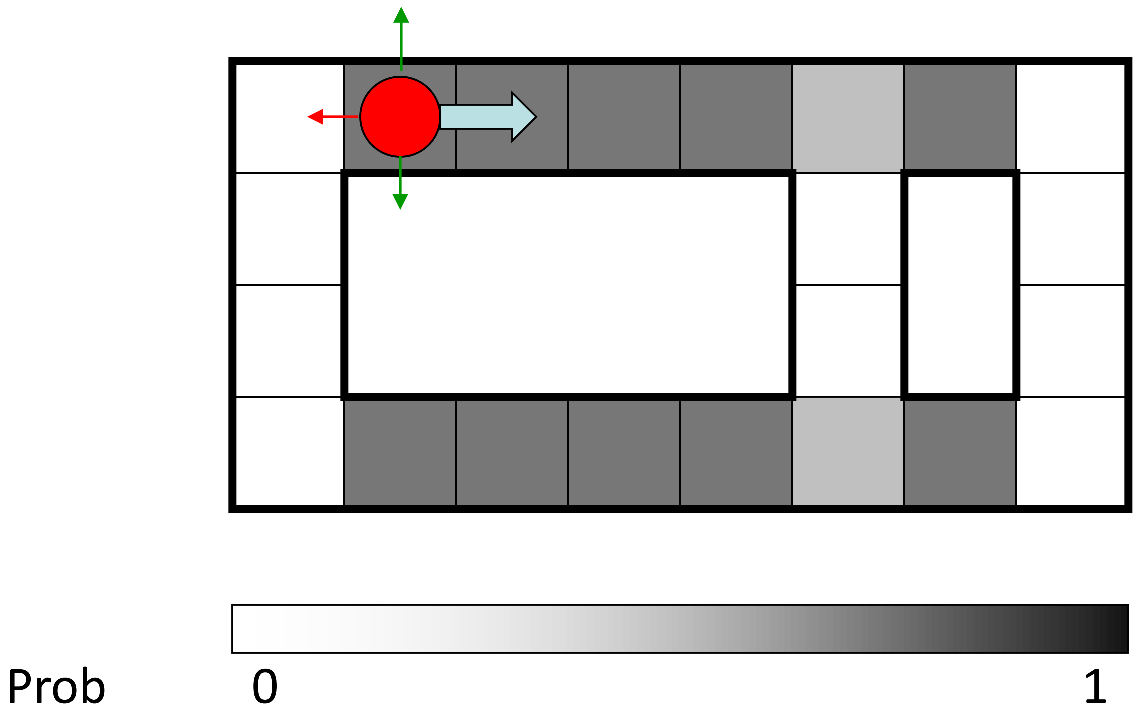




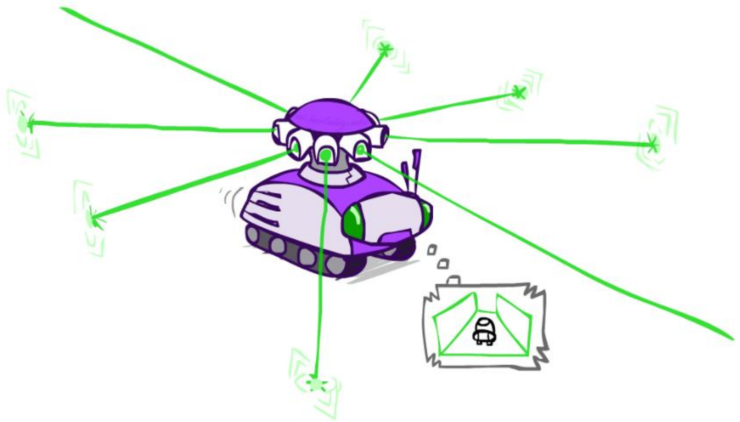
- Hidden state: Robot's true location
- **Motion** (transition) model
  - Move in *intended* direction with larger probability)
- **Sensor** (emissions) model
  - Wall or no wall in each cardinal direction, noisy readings

t=0

Example: Robot Localization



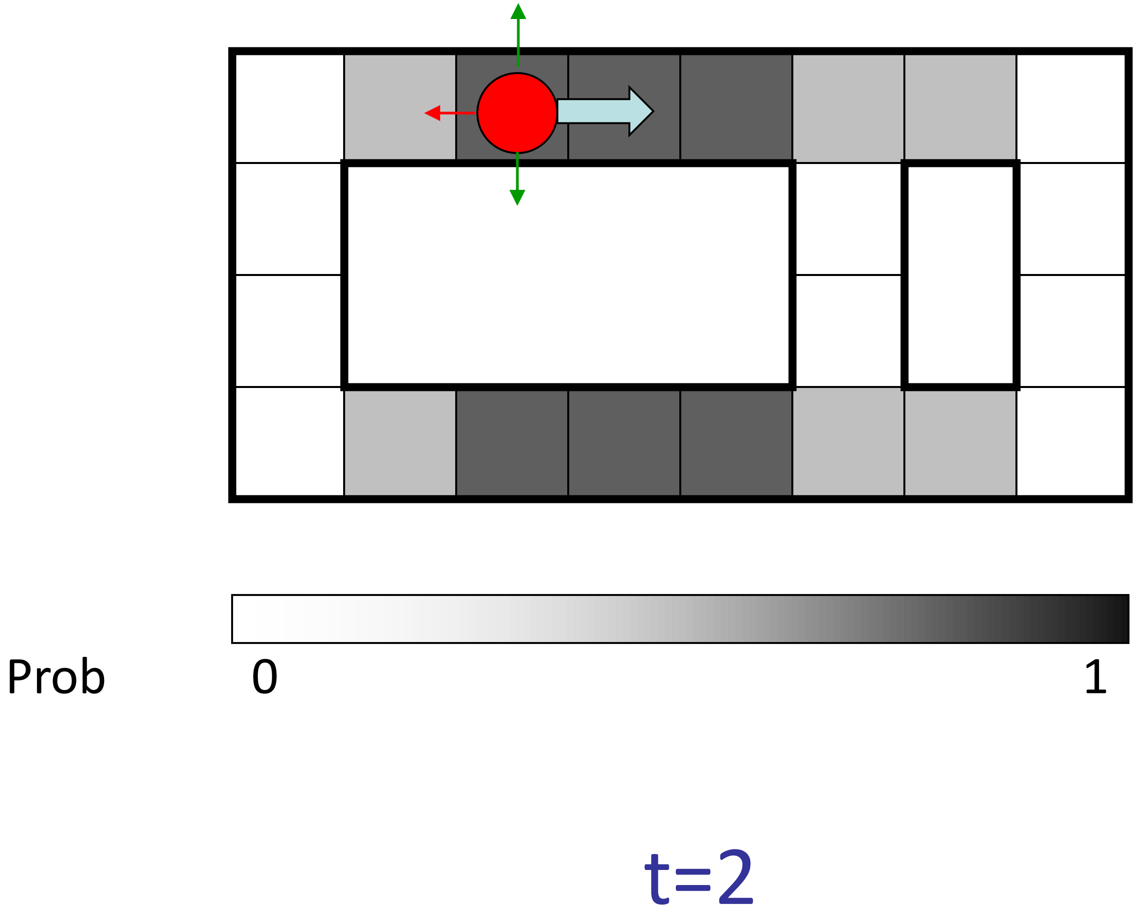
t=1

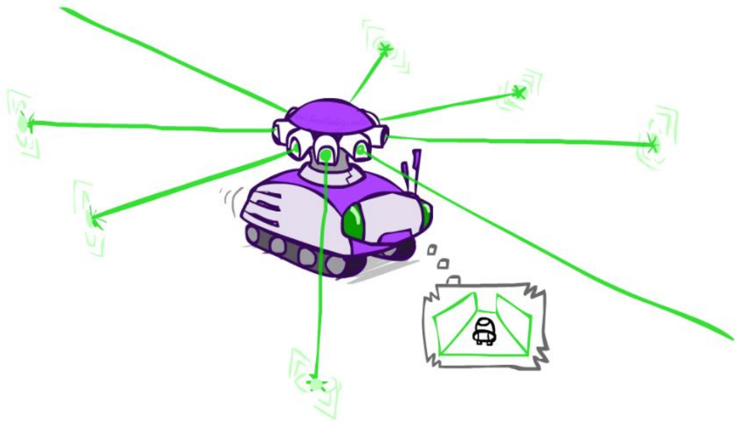


- One observation, before moving:
- Can narrow down to non-corner states at top and bottom

- Note the less likely (but prob non-zero) states in the middle!

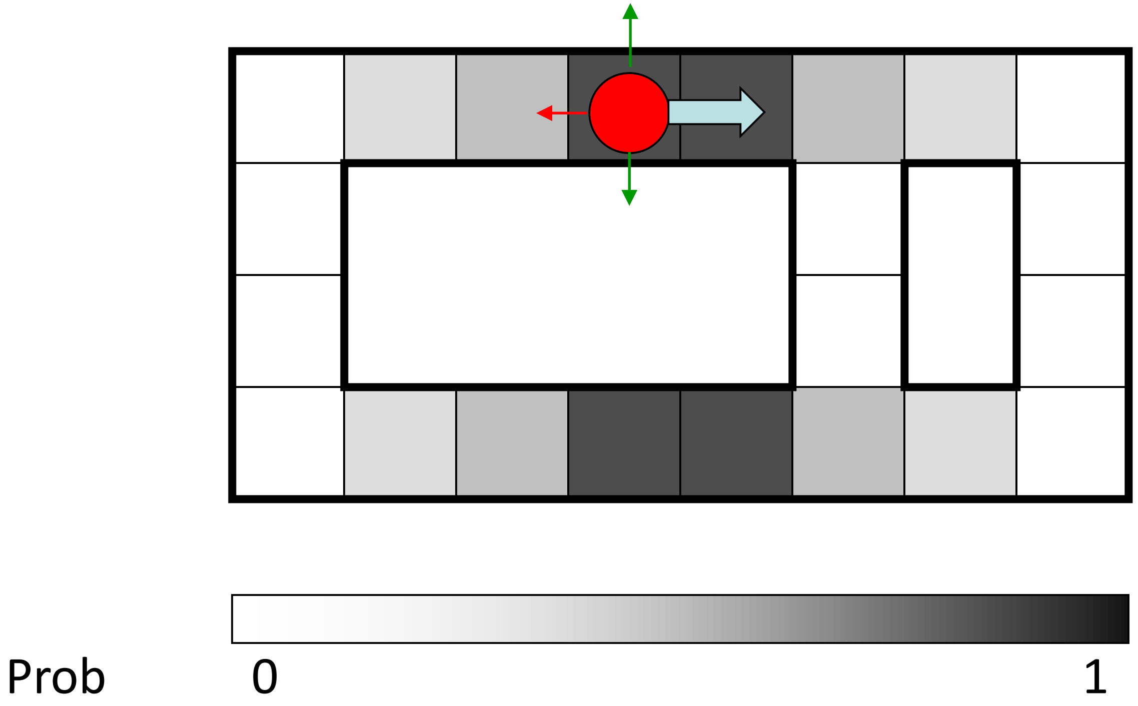
Example: Robot Localization

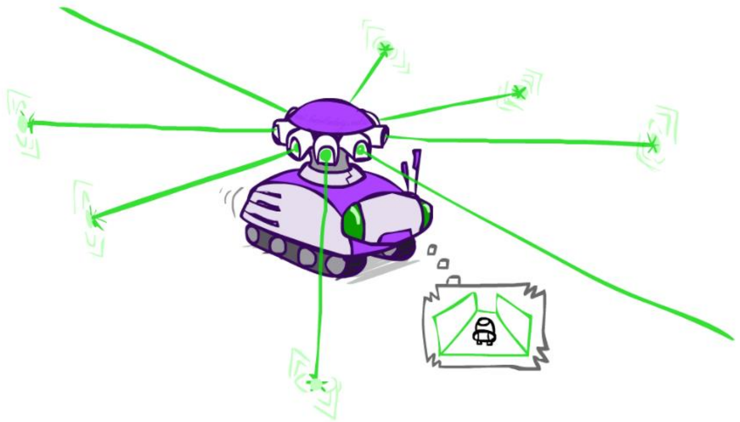




- Move and sense two walls again
- More likely for robot to be in the middle darker locations
- Light grey on left: More likely that we moved than not

Example: Robot Localization

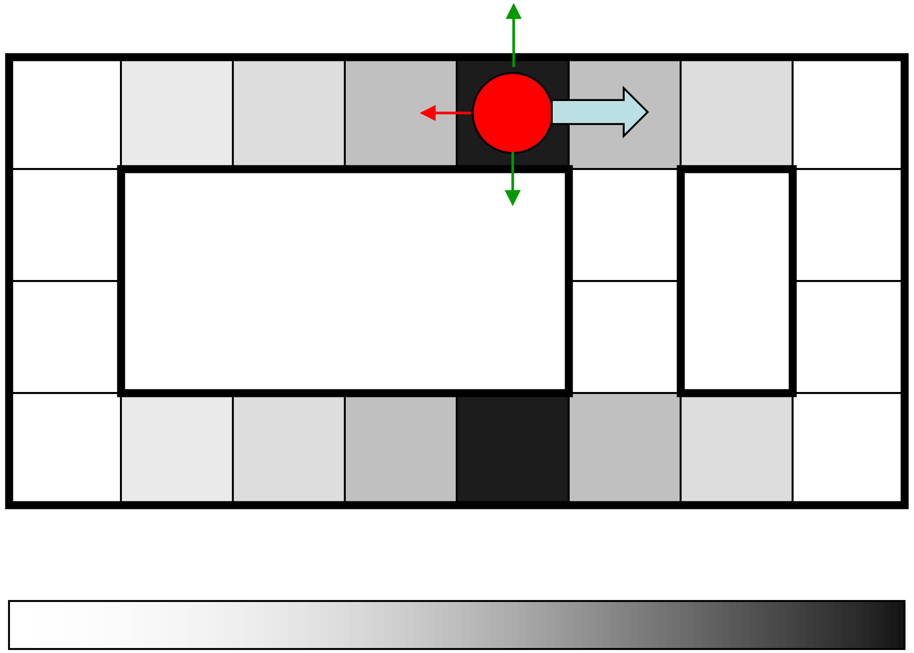




- Robot continues updating its belief state of where it is…

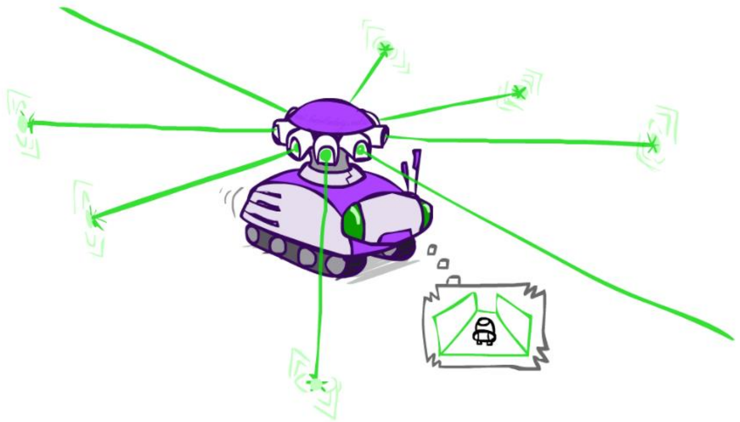
t=3

Example: Robot Localization



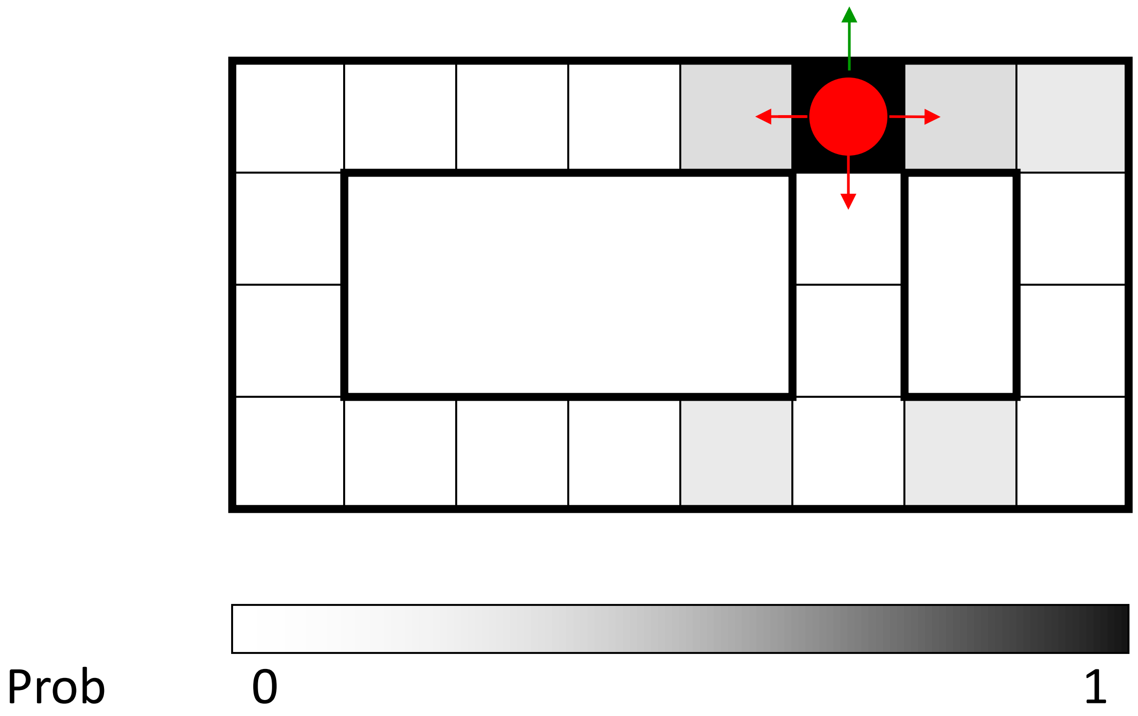
Prob 0 1

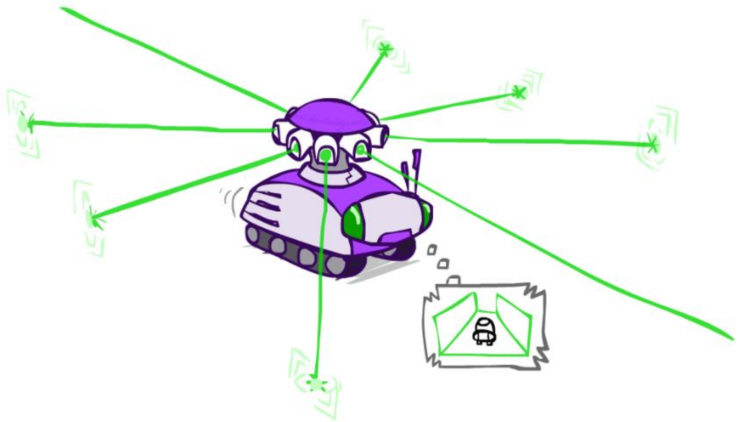
t=4



- Robot continues updating its belief state of where it is…

Example: Robot Localization





- Robot is now very confident in its belief about its current location!

t=5

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?)

- Instead of trying to compute $P(e_{1:t})$, we can simply *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

- Let's suppose we have $\boldsymbol{f}_t = P(X_t \mid e_{1:t})$
- **Elapse time:**

Transition

Conditional independence

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

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

$$\Rightarrow \boxed{\boldsymbol{f}_t \cdot P(X_{t+1} \mid X_t) = \boldsymbol{f}'_{t+1}}$$

- **Observe evidence:**

Emission

Conditional independence

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

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

$$\Rightarrow \boxed{\boldsymbol{f}'_{t+1} * P(e_{t+1} \mid X_{t+1}) \propto_{X_{t+1}} \boldsymbol{f}_{t+1}}$$

Pointwise multiply

Normalize

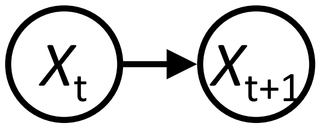
Normalize

Forward Algorithm

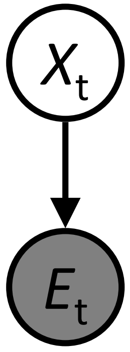
- Updates with *constant* time and space complexity despite unbounded sequence of observations
- Base case: Observe evidence for initial distribution: $\boldsymbol{f}_1 \propto_{X_1} \boldsymbol{f}_0 * P(e_1 \mid X_1)$
- Elapse time *increases uncertainty*

$$\begin{pmatrix} p_1 & \cdots & p_n \end{pmatrix} \begin{pmatrix} p_{1|1} & \cdots & p_{n|1} \\ \vdots & \ddots & \vdots \\ p_{1|n} & \cdots & p_{n|n} \end{pmatrix}$$

$$\boldsymbol{f}'_{t+1} = \boldsymbol{f}_t \cdot P(X_{t+1} \mid X_t)$$



- Observation reweights beliefs, *decreases uncertainty*



$$\boldsymbol{f}_{t+1} \propto_{X_{t+1}} \boldsymbol{f}'_{t+1} * P(e_{t+1} \mid X_{t+1})$$

$$\begin{pmatrix} p_1' & \cdots & p_n' \end{pmatrix} * \begin{pmatrix} \ddots & \boxed{\begin{matrix} p_{e|1} \\ \vdots \\ p_{e|n} \end{matrix}} & \ddots \\ \ddots & & \ddots \\ \ddots & & \ddots \end{pmatrix}$$

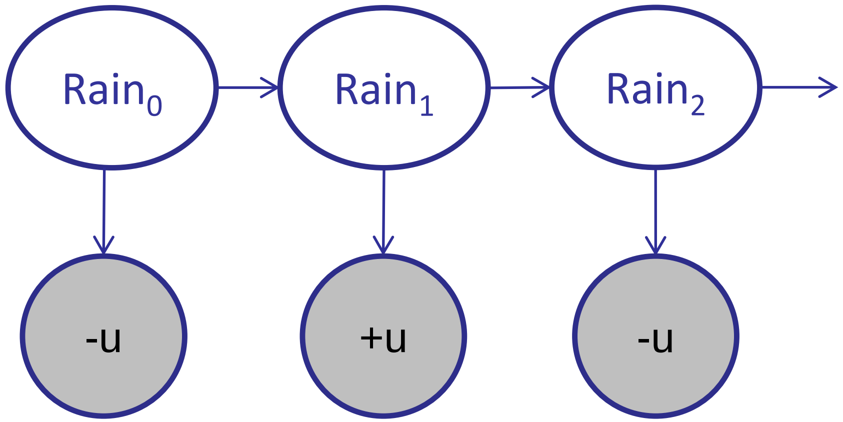
Example: Weather HMM

$$P(R_{t+1} \mid R_t) = \begin{array}{cc} \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} & \begin{array}{c} +r \\ -r \end{array} \\ \begin{array}{cc} +r & \quad -r \end{array} & \end{array} \qquad P(U_t \mid R_t) = \begin{array}{c} \begin{pmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{pmatrix} \\ \begin{array}{cc} +u & \quad -u \end{array} \end{array}$$

$$\boldsymbol{f}'_{t+1} = \boldsymbol{f}_t \cdot P(X_{t+1} \mid X_t)$$

$$\boldsymbol{f}_{t+1} \propto_{X_{t+1}} \boldsymbol{f}'_{t+1} * P(e_{t+1} \mid X_{t+1})$$

$$\begin{array}{ccccc} f_0 = (0.5, 0.5) & & f'_2 = (.34, .66) & & f'_3 = (.58, .42) \\ \downarrow & \nearrow & \downarrow & \nearrow & \downarrow \\ f_1 = (.11, .89) & & f_2 = (0.7, 0.3) & & f_3 = (.15, .85) \end{array}$$



$$f_0 * (0.1, 0.8) = (0.05, 0.4) \propto (.11, .89) = f_1$$

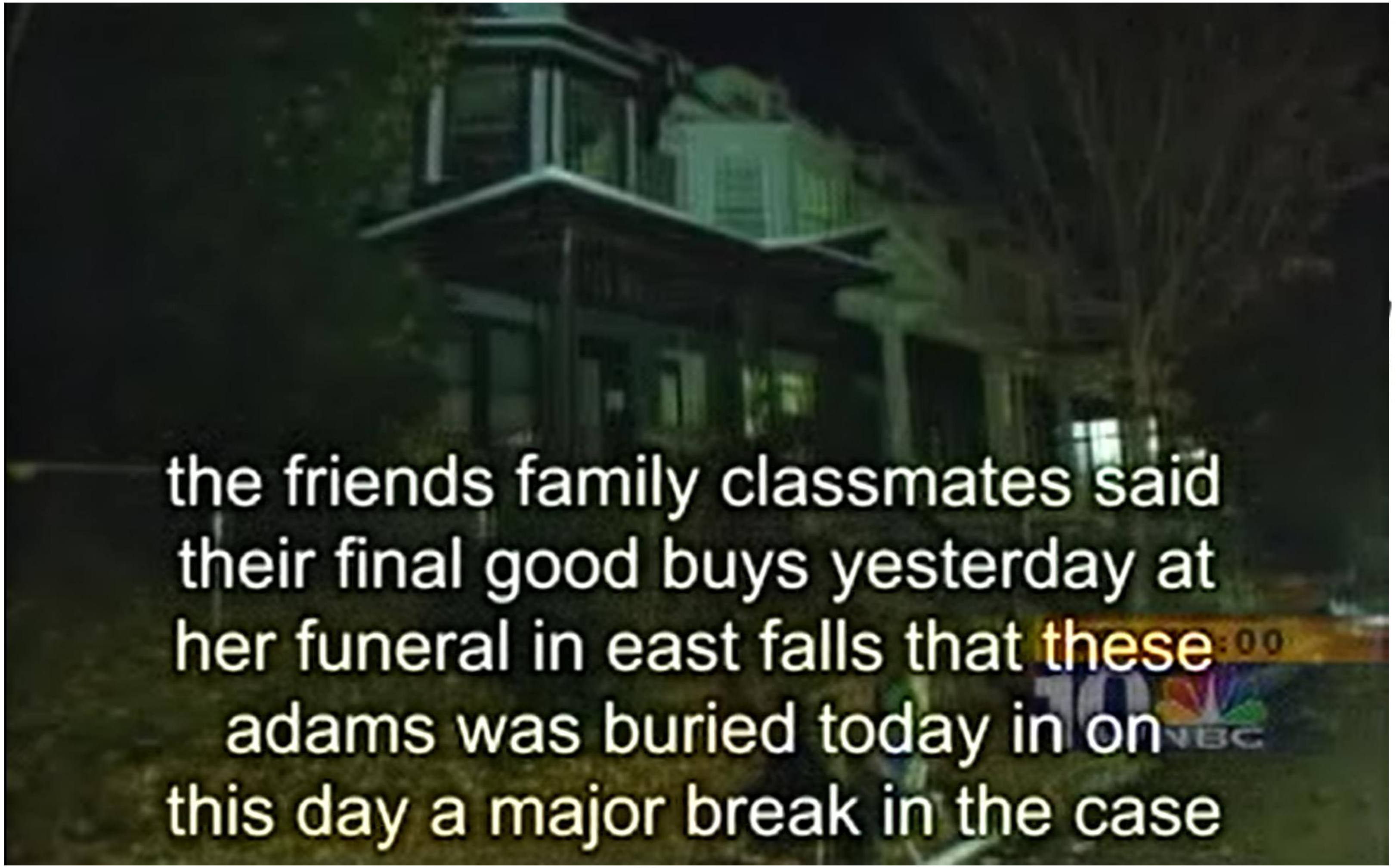
$$f'_2 = f_1 \cdot \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} = (.34, .66)$$

$$f'_2 * (0.9, 0.2) = (.31, .13) \propto (0.7, 0.3) = f_2$$

$$f'_3 = f_2 \cdot \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} = (.58, .42)$$

$$f'_3 * (0.1, 0.8) = (.06, .34) \propto (.15, .85) = f_3$$

Inference: Most Likely Sequence



Most Likely Sequence

- What is the most likely *sequence* of states given a *sequence* of evidence?

$$\text{argmax}_{x_{1:t}} P(X_{1:t} \mid e_{1:t})$$

- Equivalently, we can argmax the **joint probability** $P(X_{1:t} \mid e_{1:t}) \propto P(X_{1:t}, e_{1:t})$

- We **cannot** just run forward algorithm for each state and find argmax!

This is why you can't just look at the individual values of the states

- Most likely individual states may differ from that of the most likely sequence

X_1	X_2	X_3	$P(X_1, X_2)$
$+x$	$+x$	$+x$	0.05
$+x$	$+x$	$-x$	0.1
$+x$	$-x$	$+x$	0.3
$+x$	$-x$	$-x$	0.15
$-x$	$+x$	$+x$	0
$-x$	$+x$	$-x$	0.2
$-x$	$-x$	$+x$	0.05
$-x$	$-x$	$-x$	0.15

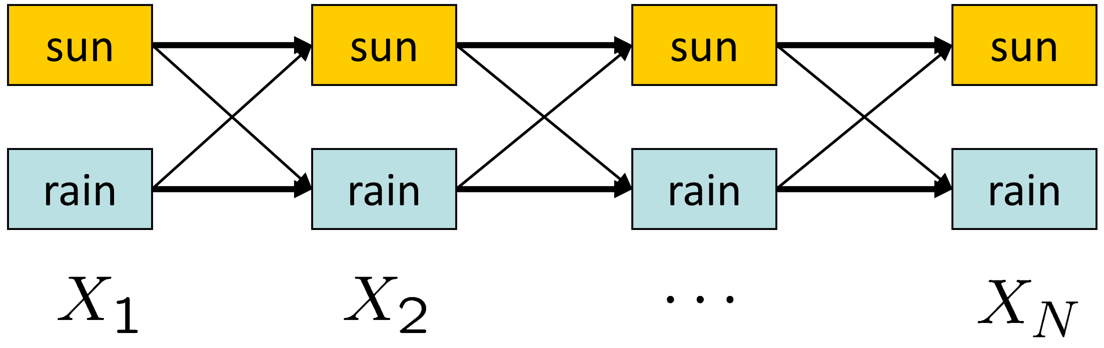
Sum of the positive values of variable vs the negative values

$$\left\{\begin{array}{l} \text{argmax } P(X_1) = +x \\ \text{argmax } P(X_2) = -x \\ \text{argmax } P(X_3) = -x \end{array}\right.$$

$\neq$

$$\text{argmax } P(X_1, X_2, X_3) = (+x, -x, +x)$$

State Trellis Diagram



- A state sequence is a *path* through a **state trellis diagram**
- Each arc is a transition $x_{t-1} \rightarrow x_t$ with weight $P(e_t \mid x_t)P(x_t \mid x_{t-1})$
- A state sequence is also a specific *event* of joint state values
- Maximizing the joint probability = maximizing the *product* of arc weights along a path

- Idea: Best path to state x_t *includes* best path to state x_{t-1}, followed by a transition
- *Recursively* compute best paths by recording max joint probabilities so far

Viterbi Algorithm

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

- Then $\boldsymbol{m}_{t+1}(x_{t+1})$ "concatenates" $\boldsymbol{m}_t$ with a state value x_t:

Conditional independence

Conditional independence

$$
\begin{aligned}
\boldsymbol{m}_{t+1}(x_{t+1}) &= \max_{x_1 \dots x_t} P(x_{1:t}, x_{t+1}, e_{1:t+1}) \\
&= \max_{x_1 \dots x_t} P(x_{1:t-1}, x_t, e_{1:t}) P(x_{t+1} \mid x_t, \cancel{x_{1:t-1}}, \cancel{e_{1:t}}) P(e_{t+1} \mid x_{t+1}, \cancel{x_t}, \cancel{x_{1:t-1}}, \cancel{e_{1:t}}) \\
&= \max_{x_1 \dots x_t} P(x_{1:t-1}, x_t, e_{1:t}) P(x_{t+1} \mid x_t) P(e_{t+1} \mid x_{t+1}) \\
&= \max_{x_t} P(e_{t+1} \mid x_{t+1}) P(x_{t+1} \mid x_t) \max_{x_1 \dots x_{t-1}} P(x_{1:t-1}, x_t, e_{1:t}) \\
&= P(e_{t+1} \mid x_{t+1}) \max_{x_t} P(x_{t+1} \mid x_t)\, \boldsymbol{m}_t(x_t)
\end{aligned}
$$

Emission

Transition

Same as forward algorithm but replace sum with max!

Viterbi Algorithm

- **Elapse time:** Each value of $\boldsymbol{m}'_{t+1}$ maxes over *pointwise product* between $\boldsymbol{m}_t$ and corresponding *column* of transition matrix

$$\boldsymbol{m}'_{t+1}(x_{t+1}) = \max(\boldsymbol{m}_t * P(x_{t+1} \mid X_t))$$

$$\begin{array}{cc} & * \, x_{t+1} \\ (p_1 \quad \cdots \quad p_n) & \begin{pmatrix} p_{1|1} & \cdots & p_{n|1} \\ \vdots & \vdots & \vdots \\ p_{1|n} & \cdots & p_{n|n} \end{pmatrix} \\ & \max \end{array}$$

- Since we want a *sequence of states*, we also need a pointer to best *parent* of each x_{t+1}: $Pointer_{t+1}(x_{t+1}) = \text{argmax}_{x_t}(\boldsymbol{m}_t * P(x_{t+1} \mid X_t))$

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

- *Backward pass*: Starting with $Pointer_T(\max \boldsymbol{m}_T)$, follow pointers backwards to x_1 to extract most likely sequence of states

Example: Weather HMM

$$P(R_{t+1} \mid R_t) = \begin{array}{cc} & \begin{array}{cc} +r & -r \end{array} \\ \begin{array}{c} +r \\ -r \end{array} & \begin{pmatrix} 0.6 & 0.4 \\ 0.3 & 0.7 \end{pmatrix} \end{array}$$

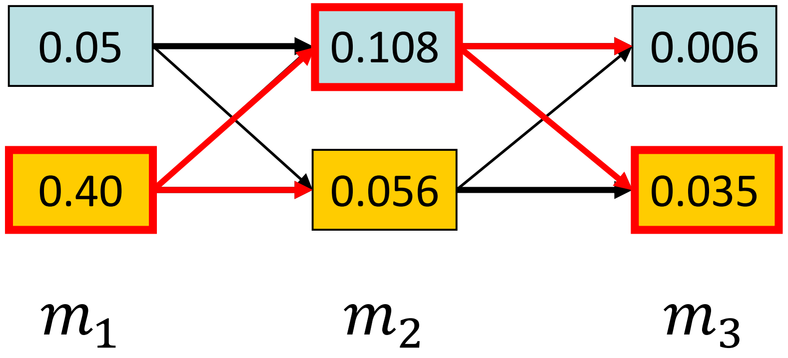
$$P(U_t \mid R_t) = \begin{array}{cc} \begin{pmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{pmatrix} & \begin{array}{c} +r \\ -r \end{array} \\ \begin{array}{cc} +u & -u \end{array} & \end{array}$$

Observed evidence:

$e_1 = -u$

$e_2 = +u$

$e_3 = -u$



Backward pointers:
$\mathrm{argmax}_{x_t}\, \boldsymbol{m}_{t+1}(x_{t+1})$ $\begin{bmatrix} -r \\ -r \end{bmatrix}$ $\begin{bmatrix} +r \\ +r \end{bmatrix}$

$m_0 = (0.5, 0.5)$

$m_1 = m_0 * (0.1, 0.8) = (0.05, 0.4)$

$m_2'(+r) = \max\big((0.05, 0.4) * (0.6, 0.3)\big) = .12$

$m_2'(-r) = \max\big((0.05, 0.4) * (0.4, 0.7)\big) = .28$

$m_2 = (.12, .28) * (0.9, 0.2) = (.108, .056)$

$m_3'(+r) = \max\big((.11, .06) * (0.6, 0.3)\big) = .065$

$m_3'(-r) = \max\big((.11, .06) * (0.4, 0.7)\big) = .043$

$m_3 = (.065, .043) * (0.1, 0.8) = (.006, .035)$

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

Inference Applications

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

- Both are heavily used in digital signals (cellular, satellite, LAN, etc.), speech recognition (audio to text), bioinformatics (gene decoding), finance (stock, asset trends)

- 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**