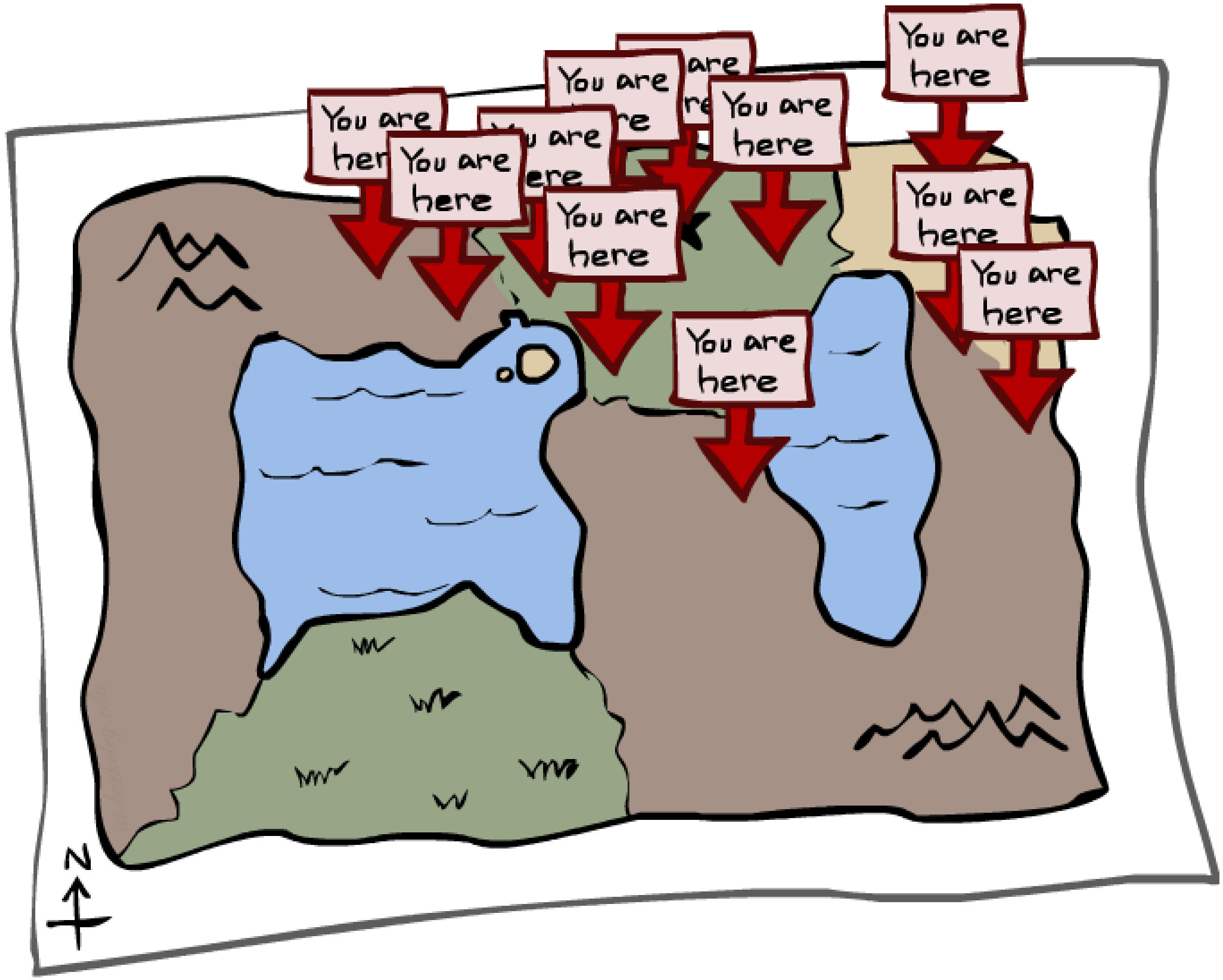
复旦大学大数据学院
School of Data Science, Fudan University
魏忠钰

Particle Filters and Applications of HMMs

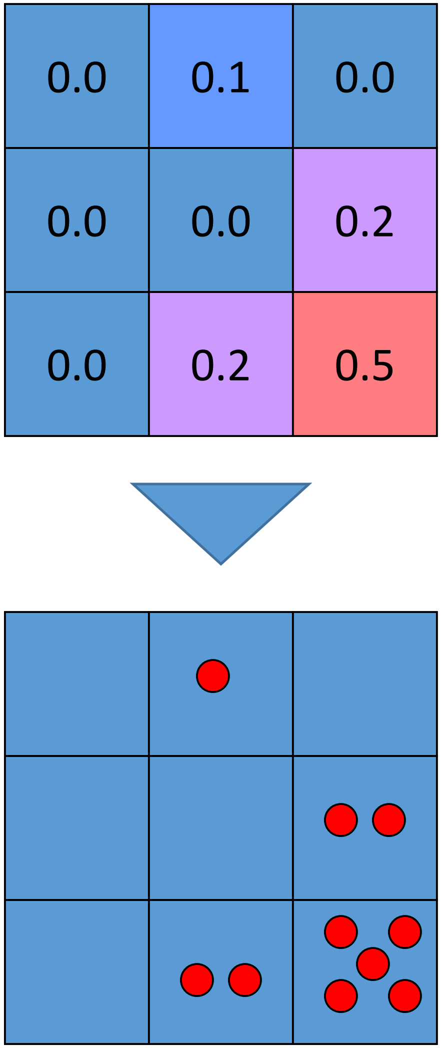
May 16th, 2018

Particle Filtering



Particle Filtering

- Filtering: approximate solution
- Sometimes |X| is too big to use exact inference
  - |X| may be too big to even store B(X)
  - E.g. X is continuous
- Solution: approximate inference
  - Track samples of X, not all values
  - Samples are called particles
  - Time per step is linear in the number of samples
  - But: number needed may be large
  - In memory: list of particles, not states
- This is how robot localization works in practice
- Particle is just new name for sample

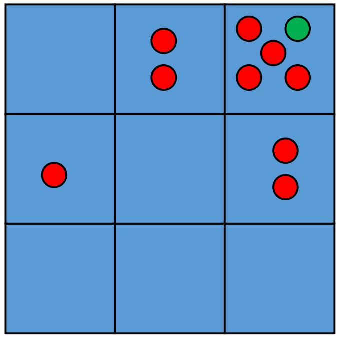


Representation: Particles

- Our representation of P(X) is now a list of N particles (samples)
  - Generally, N << |X|
  - Storing map from X to counts would defeat the point

- P(x) approximated by number of particles with value x
  - So, many x may have P(x) = 0!
  - More particles, more accuracy

- For now, all particles have a weight of 1



Particles:
(3,3)
(2,3)
(3,3)
(3,2)
(3,3)
(3,2)
(1,2)
(3,3)
(3,3)
(2,3)

Particle Filtering: Elapse Time

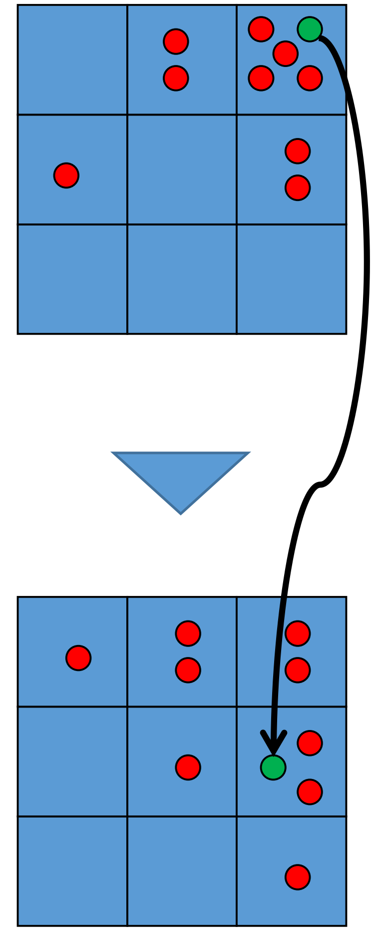
- Each particle is moved by sampling its next position from the transition model

$$x' = \text{sample}(P(X'|x))$$

  - This is like prior sampling – samples' frequencies reflect the transition probabilities
  - Here, most samples move clockwise, but some move in another direction or stay in place

- This captures the passage of time
  - If enough samples, close to exact values before and after (consistent)

Particles:
(3,3)
(2,3)
(3,3)
(3,2)
(3,3)
(3,2)
(1,2)
(3,3)
(3,3)
(2,3)



Particles:
(3,2)
(2,3)
(3,2)
(3,1)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(2,2)

Particle Filtering: Observe

- Similar to likelihood weighting, re-weight samples based on the evidence

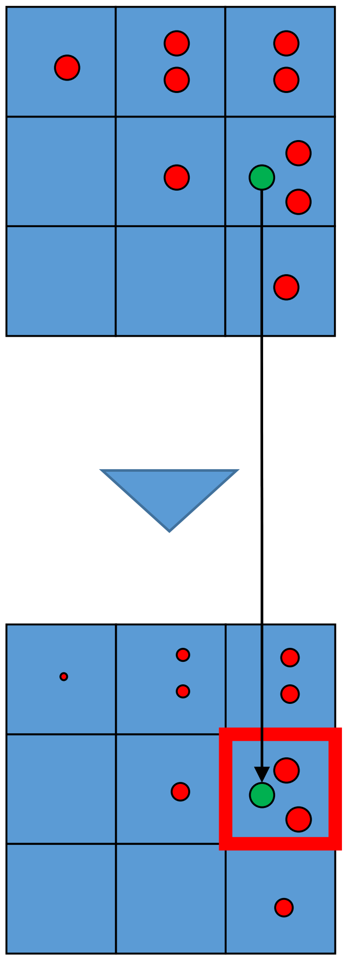
$$w(x) = P(e|x)$$

$$B(X) \propto P(e|X)B'(X)$$

- As before, the probabilities don't sum to one, since all have been re-weighted (in fact they now sum to (N times) an approximation of P(e))

Particles:
(3,2)
(2,3)
(3,2)
(3,1)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(2,2)

Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4



Particle Filtering: Resample

- Rather than tracking weighted samples, we resample

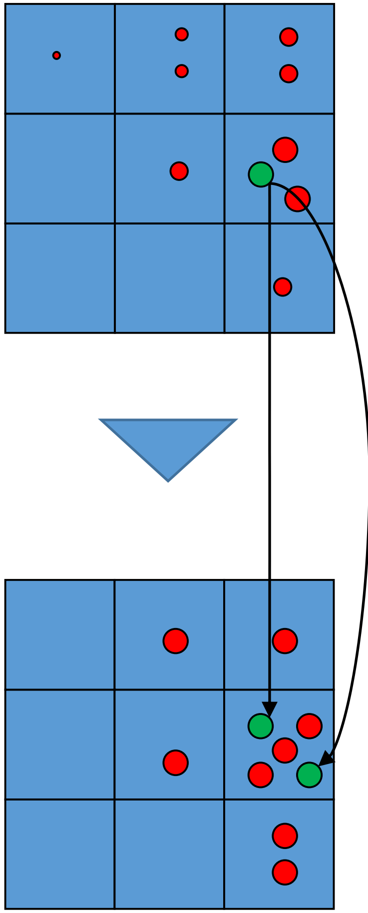
- N times, we choose from our weighted sample distribution (i.e. draw with replacement)

- This is equivalent to renormalizing the distribution

- Now the update is complete for this time step, continue with the next one

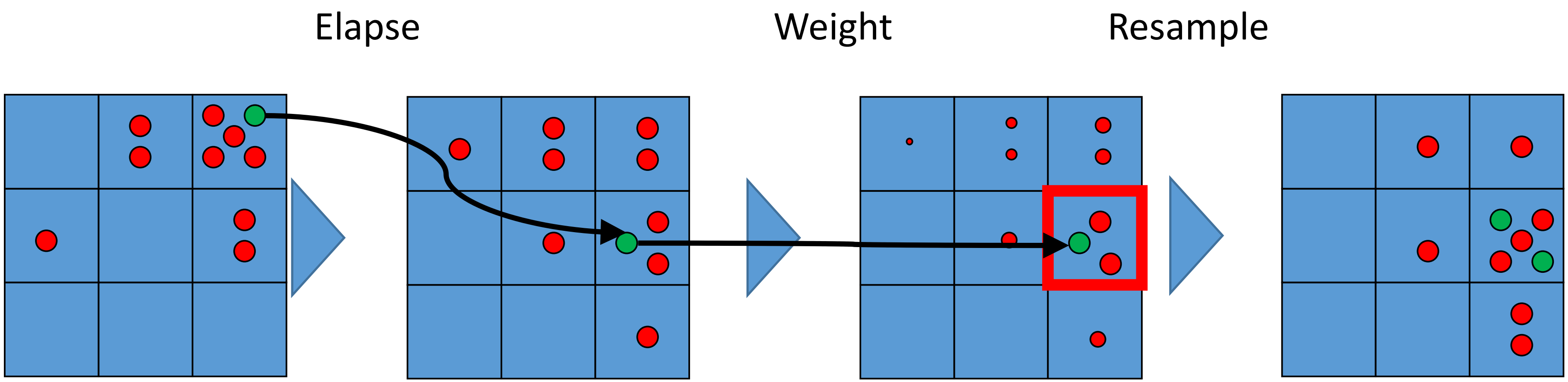
Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

(New) Particles:
(3,2)
(2,2)
(3,2)
(2,3)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(3,2)



Recap: Particle Filtering

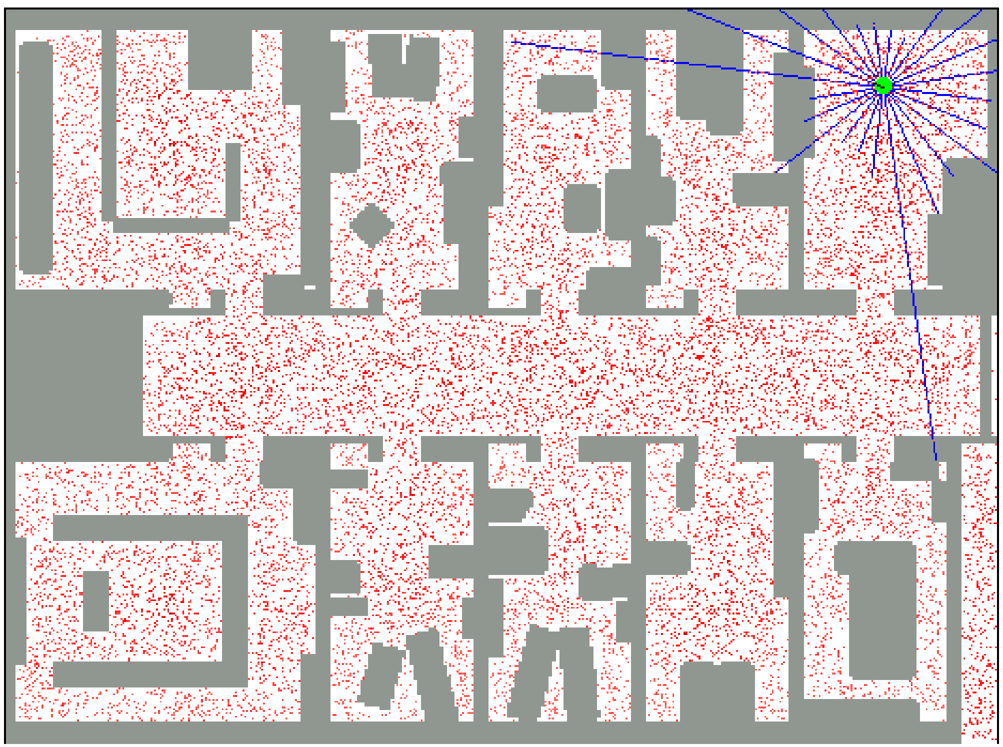
- Particles: track samples of states rather than an explicit distribution

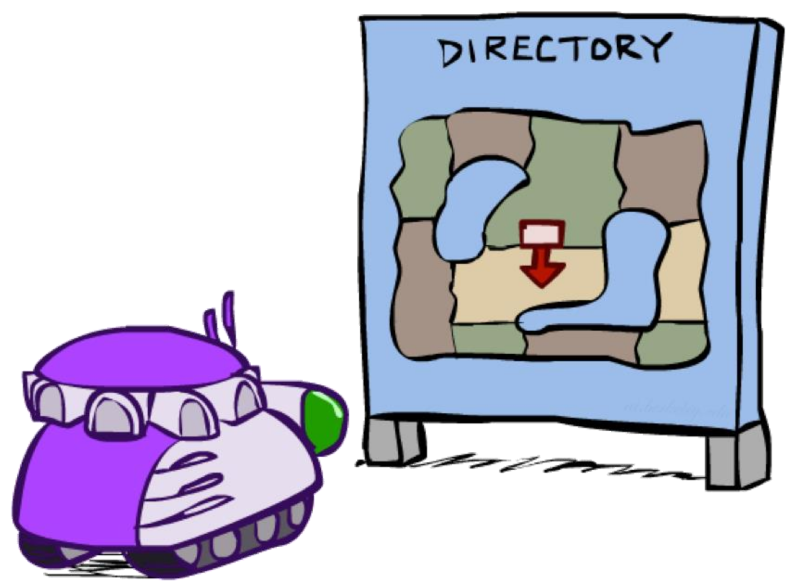


Particles:	Particles:	Particles:	(New) Particles:
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(2,3)	(2,3)	(2,3) w=.2	(2,2)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(3,2)	(3,1)	(3,1) w=.4	(2,3)
(3,3)	(3,3)	(3,3) w=.4	(3,3)
(3,2)	(3,2)	(3,2) w=.9	(3,2)
(1,2)	(1,3)	(1,3) w=.1	(1,3)
(3,3)	(2,3)	(2,3) w=.2	(2,3)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(2,3)	(2,2)	(2,2) w=.4	(3,2)

Robot Localization

- In robot localization:
  - We know the map, but not the robot's position
  - Observations may be vectors of range finder readings
  - State space and readings are typically continuous (works basically like a very fine grid) and so we cannot store B(X)
  - Particle filtering is a main technique

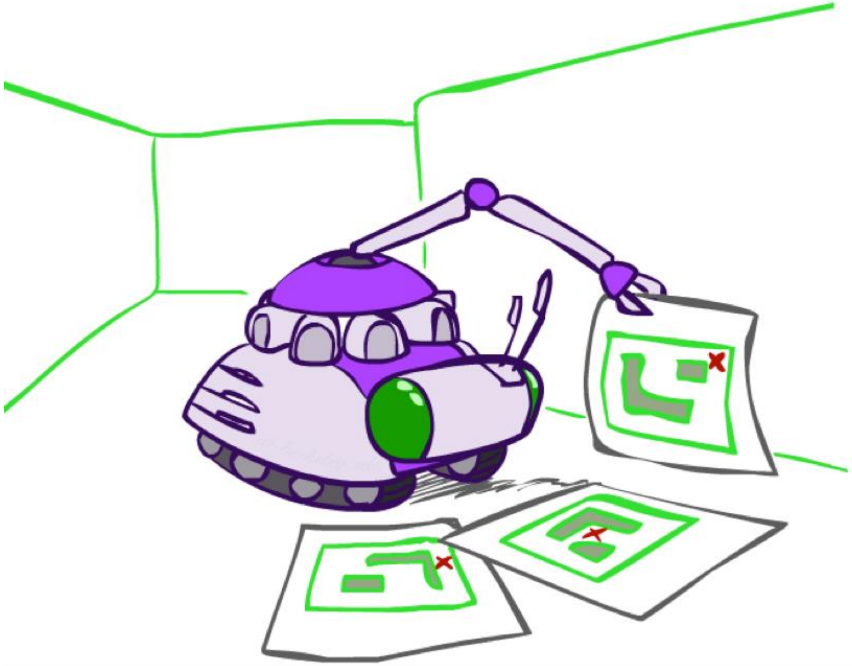




Particle Filter Localization (Sonar)

Robot Mapping

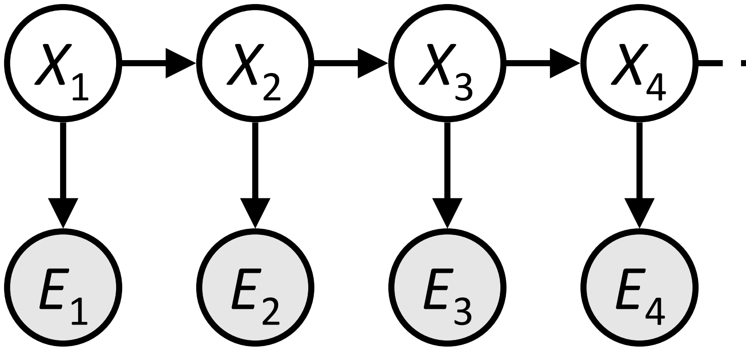
- SLAM: Simultaneous Localization And Mapping
  - We do not know the map or our location
  - State consists of position AND map!
  - Main techniques: Kalman filtering (Gaussian HMMs) and particle methods



Tasks for HMM

- **Smoothing**
  - Computing the posterior distribution over a *past* state, given all evidence up to the present.
  - $\boldsymbol{P}(\boldsymbol{X_k}|\boldsymbol{e_{1:t}})$

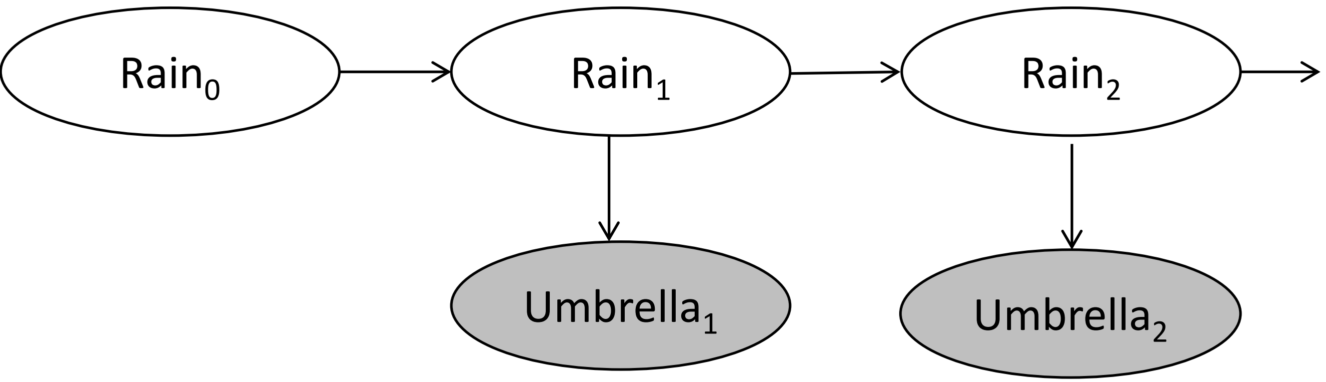
Smoothing



$$
\begin{aligned}
P(X_k|e_{1:t}) &= P(X_k|e_{1:k}, e_{k+1:t}) \\
&\propto P(X_k|e_{1:k})P(e_{k+1:t}|X_k, e_{1:k}) \\
&= P(X_k|e_{1:k})P(e_{k+1:t}|X_k)
\end{aligned}
$$

$$
\begin{aligned}
P(e_{k+1:t}|X_k) &= \sum_{x_{k+1}} P(e_{k+1:t}|X_k, x_{k+1})\, P(x_{k+1}|X_k) \\
&= \sum_{x_{k+1}} P(e_{k+1:t}|x_{k+1})\, P(x_{k+1}|X_k) \\
&= \sum_{x_{k+1}} P(e_{k+1}, e_{k+2:t}|x_{k+1})\, P(x_{k+1}|X_k) \\
&= \sum_{x_{k+1}} P(e_{k+1}|x_{k+1})P(e_{k+2:t}|x_{k+1})\, P(x_{k+1}|X_k)
\end{aligned}
$$

Example: Weather HMM



R_t	R_{t+1}	$P(R_{t+1} \mid R_t)$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

R_t	U_t	$P(U_t \mid R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

B(+r) = 0.5
B(-r) = 0.5

B (+r) =
B (-r) =

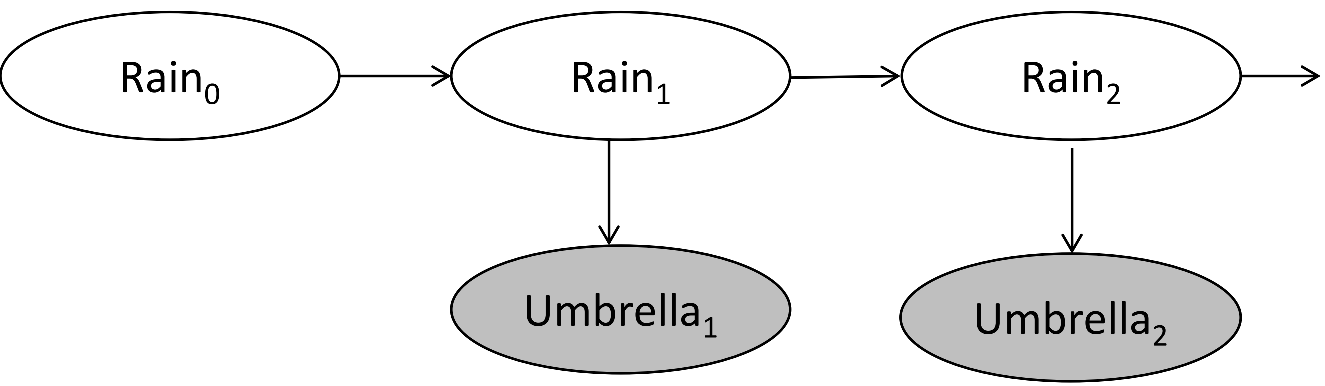
Smoothing

Example: Weather HMM

Filtering

$B(+r) = 0.5$
$B(-r) = 0.5$

$B(+r) = 0.818$
$B(-r) = 0.182$



R_t	R_{t+1}	$P(R_{t+1} \mid R_t)$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

R_t	U_t	$P(U_t \mid R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

Smoothing

$B(+r) = 0.5$
$B(-r) = 0.5$

f: $B(+r) = 0.818$ b: $B(+r) = 0.69$
f: $B(-r) = 0.182$ b: $B(-r) = 0.41$

$\Rightarrow$

$B(+r) = 0.883$
$B(-r) = 0.117$

Tasks for HMM

- **Most Likely Explanation**
  - Given a sequence of observations, find the sequence of states that is most likely to have generated those observations.

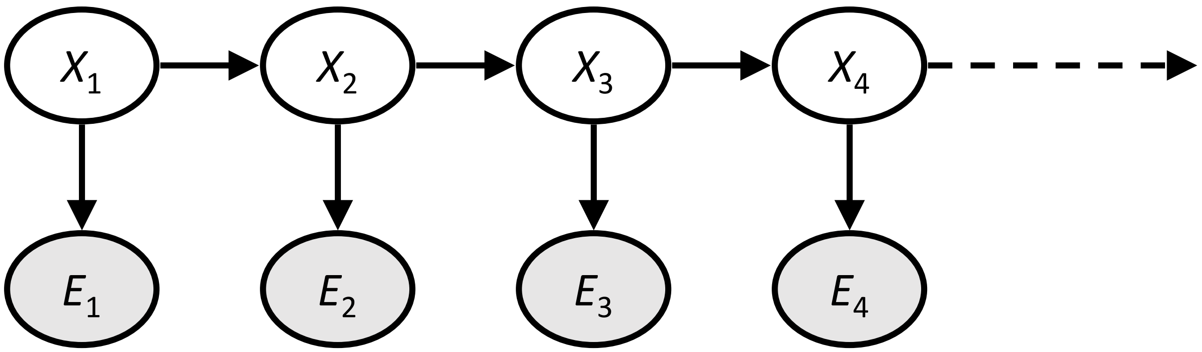
$$\arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$$

Most Likely Explanation

HMMs: MLE Queries

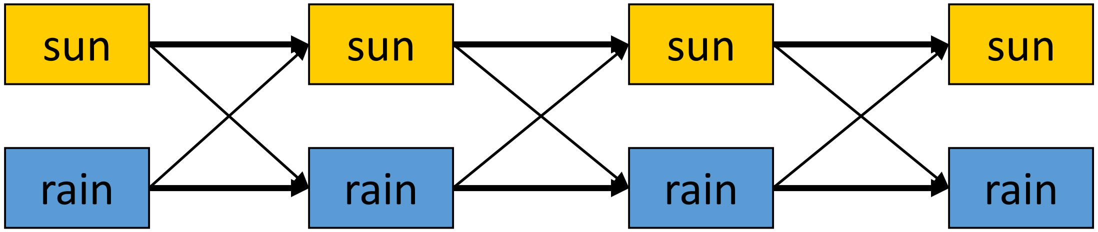
- HMMs defined by
  - States X
  - Observations E
  - Initial distribution: $P(X_1)$
  - Transitions: $P(X|X_{-1})$
  - Emissions: $P(E|X)$

- New query: most likely explanation: $\arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$
- New method: the Viterbi algorithm



Viterbi

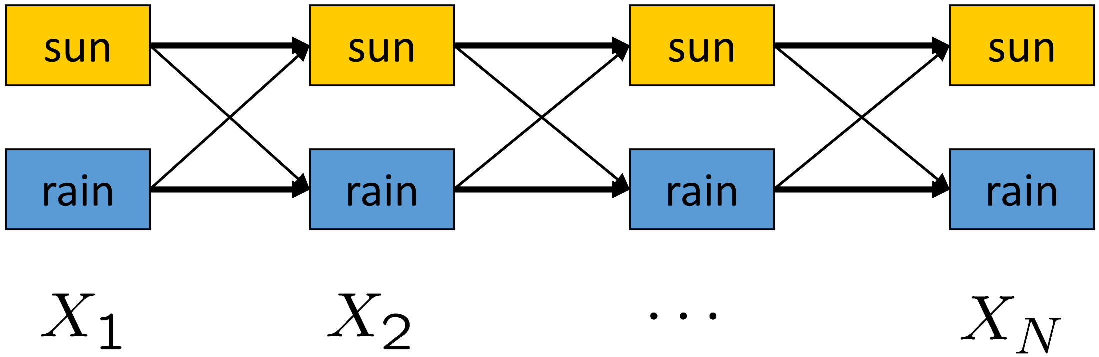
- Graph of states and transitions over time



$$X_1 \qquad X_2 \qquad \cdots \qquad X_N$$

- Each arc represents some transition $x_{t-1} \rightarrow x_t$
- Each arc has weight $P(x_t|x_{t-1})P(e_t|x_t)$
- Each path is a sequence of states
- The product of weights on a path is that sequence's probability along with the evidence
- Forward algorithm computes sums of paths, Viterbi computes best paths

Forward / Viterbi Algorithms



Forward Algorithm (Sum)

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

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

Viterbi Algorithm (Max)

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

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

Tasks for HMM

- **Filtering**
  - Computing the **belief state**—the posterior distribution over the most recent state—given all evidence to date.
  - $\boldsymbol{P(X_t|e_{1:t})}$
- **Prediction**
  - Computing the posterior distribution over the *future* state, given all evidence to date.
  - $\boldsymbol{P(X_{t+k}|e_{1:t})}$
- **Smoothing**
  - Computing the posterior distribution over a *past* state, given all evidence up to the present.
  - $\boldsymbol{P(X_k|e_{1:t})}$
- **Most Likely Explanation**
  - Given a sequence of observations, find the sequence of states that is most likely to have generated those observations.

$$\arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$$