# CS 188: Artificial Intelligence

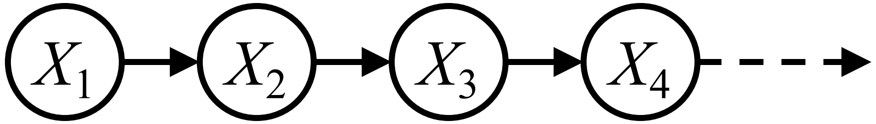
# Hidden Markov Models II

Summer 2024: Eve Fleisig & Evgeny Pobachienko

[Slides adapted from Saagar Sanghavi, Dan Klein, Pieter Abbeel, Anca Dragan, Stuart Russell]

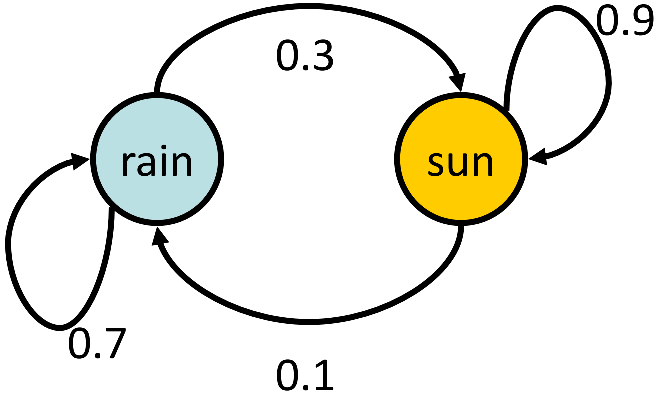
# Markov Chains

- Value of X at a given time is called the **state**



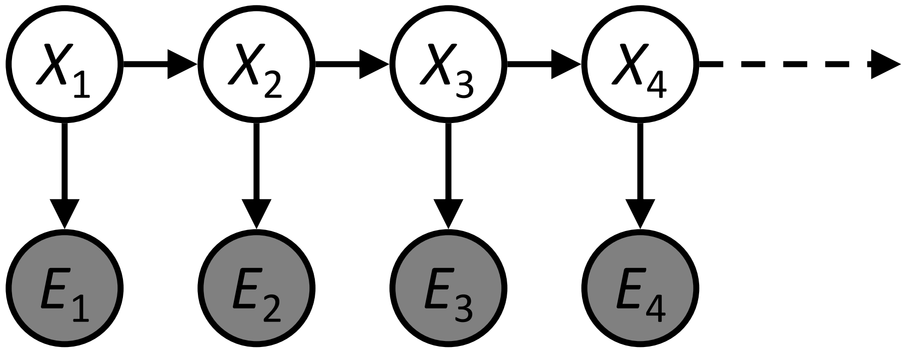

$$P(X_1) \qquad P(X_t|X_{t-1}) \qquad P(X_t) = ?$$

- Transition probabilities (**dynamics**): $P(X_t \mid X_{t-1})$ specify how the state evolves over time

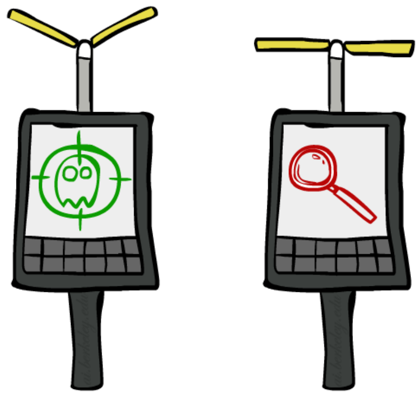


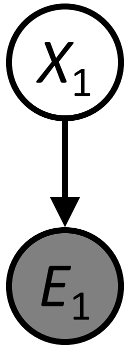
# Hidden Markov Models

- Hidden Markov models (HMMs)
  - Underlying Markov chain over states $X_i$
  - You observe outputs (effects) at each time step
- An HMM is defined by:
  - Initial distribution: $P(X_1)$
  - Transitions: $P(X_t \mid X_{t-1})$
  - Emissions: $P(E_t \mid X_t)$



# Inference: Base Cases

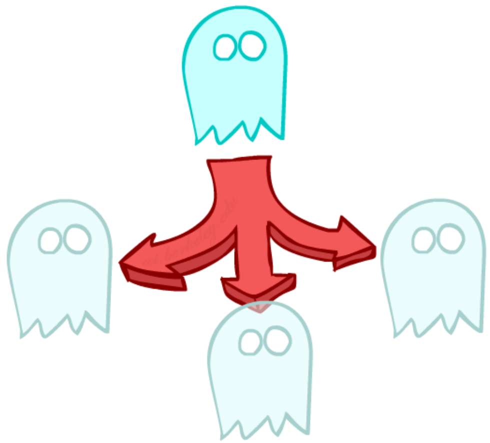


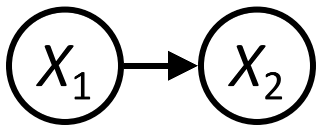



$$P(X_1|e_1)$$

$$P(X_1|e_1) = \frac{P(X_1, e_1)}{\sum_{x_1} P(x_1, e_1)}$$

$$P(X_1|e_1) = \frac{P(e_1|X_1)P(X_1)}{\sum_{x_1} P(e_1|x_1)P(x_1)}$$






$$P(X_2)$$

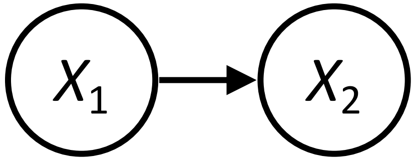
$$P(X_2) = \sum_{x_1} P(x_1, X_2)$$

$$P(X_2) = \sum_{x_1} P(X_2|x_1)P(x_1)$$

# Passage of Time

- Assume we have current belief P(X | evidence to date)

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




- Then, after one time step passes:

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

- Basic idea: beliefs get "pushed" through the transitions

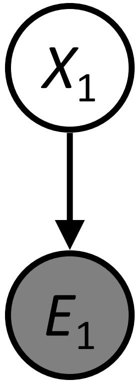
# Observation

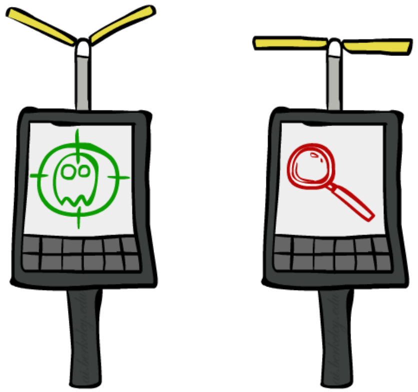
- Assume we have current belief P(X | previous evidence):

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

- Then, after evidence comes in:

$$\begin{aligned}
P(X_{t+1}|e_{1:t+1}) &= P(X_{t+1}, e_{t+1}|e_{1:t})/P(e_{t+1}|e_{1:t}) \\
&\propto_{X_{t+1}} P(X_{t+1}, e_{t+1}|e_{1:t}) \\
&= P(e_{t+1}|e_{1:t}, X_{t+1})P(X_{t+1}|e_{1:t}) \\
&= P(e_{t+1}|X_{t+1})P(X_{t+1}|e_{1:t})
\end{aligned}$$



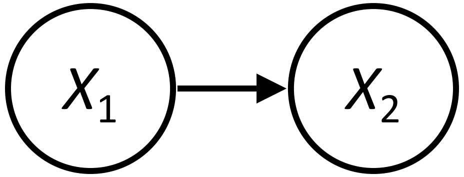



  - Basic idea: beliefs "reweighted" by likelihood of evidence
  - Unlike passage of time, we have to renormalize

# Online Belief Updates

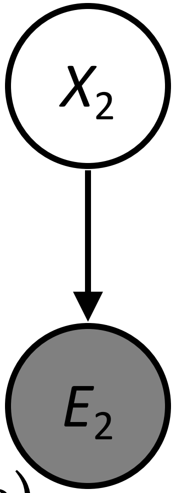
- Every time step, we start with current P(X | evidence)
- We update for time:

$$P(x_t|e_{1:t-1}) = \sum_{x_{t-1}} P(x_{t-1}|e_{1:t-1}) \cdot P(x_t|x_{t-1})$$




- We update for evidence:

$$P(x_t|e_{1:t}) \propto_X P(x_t|e_{1:t-1}) \cdot P(e_t|x_t)$$




- The forward algorithm does both at once (and doesn't normalize)

# The Forward Algorithm

- We are given evidence at each time and want to know

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

- We can derive the following updates

$$P(x_t|e_{1:t}) \propto_{X_t} P(x_t, e_{1:t})$$

$$= \sum_{x_{t-1}} P(x_{t-1}, x_t, e_{1:t})$$

$$= \sum_{x_{t-1}} P(x_{t-1}, e_{1:t-1}) P(x_t|x_{t-1}) P(e_t|x_t)$$

$$= P(e_t|x_t) \sum_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}, e_{1:t-1})$$

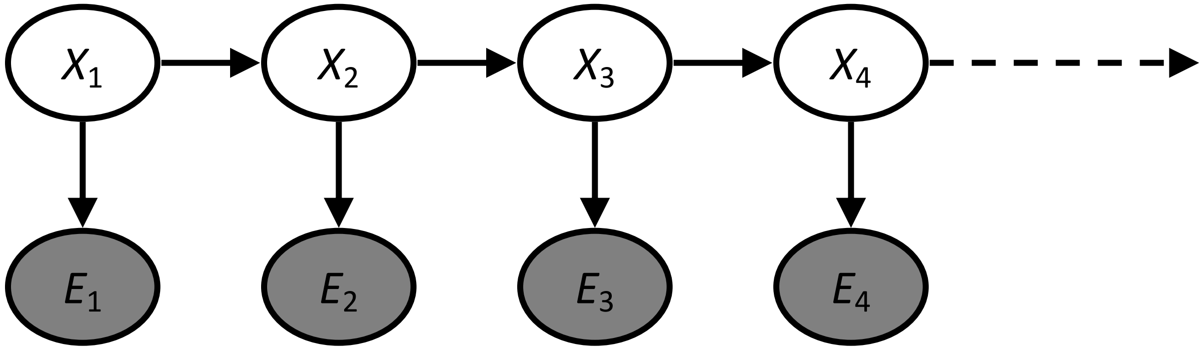
We can normalize as we go if we want to have P(x|e) at each time step, or just once at the end…

# Video of Demo Pacman – Sonar (with beliefs)

# Most Likely Explanation

# HMMs: MLSE Queries

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

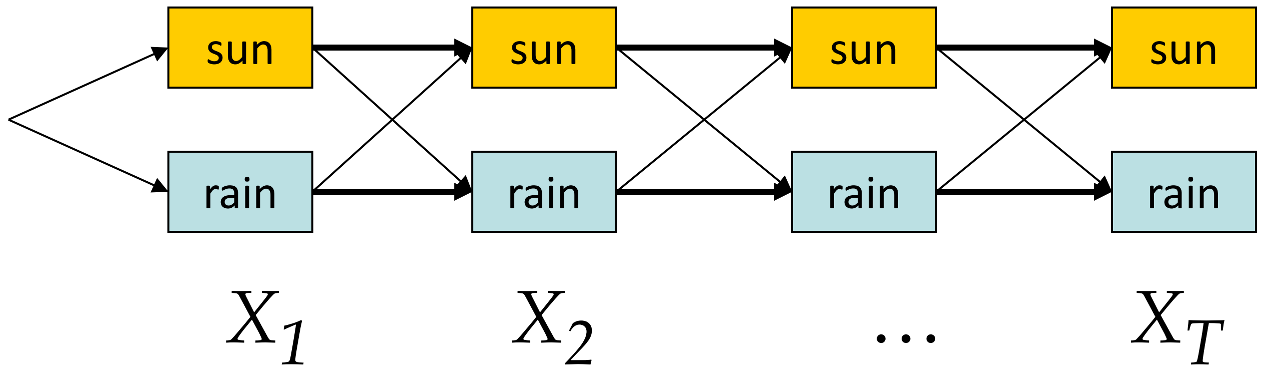



- New query: most likely explanation: $\arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$

- New method: the Viterbi algorithm

# Most likely explanation = most probable path

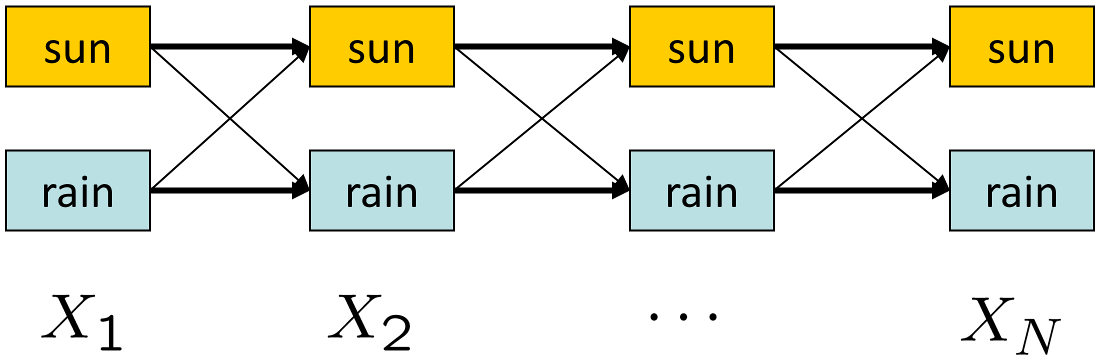
- **State trellis**: graph of states and transitions over time




$$\text{argmax}_{x_{1:t}} P(x_{1:t} \mid e_{1:t})$$
$$= \text{argmax}_{x_{1:t}} P(x_{1:t}, e_{1:t})$$
$$= \text{argmax}_{x_{1:t}} P(x_0) \prod_t P(x_t \mid x_{t\text{-}1})\, P(e_t \mid x_t)$$

- 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 seq's probability
- Forward algorithm: sums of paths
- **Viterbi algorithm:** best paths
  - Dynamic Programming: solve subproblems, combine them as you go along

# Forward / Viterbi Algorithms




## Forward Algorithm (Sum)

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

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

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

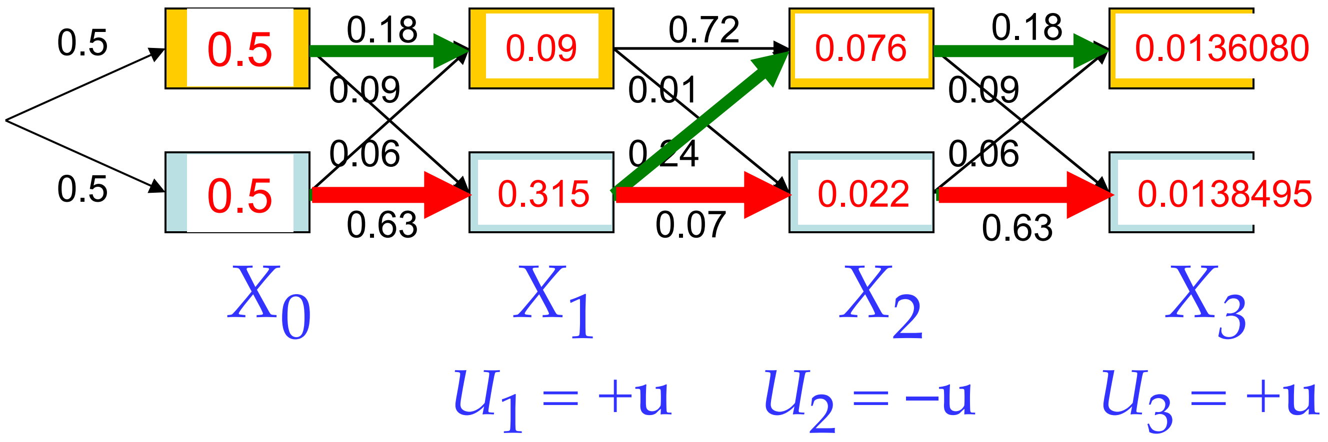
## Viterbi Algorithm (Max)

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

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

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

# Viterbi algorithm




| $R_t$ | $R_{t+1}$ | $P(R_{t+1}\|R_t)$ |
|---|---|---|
| +r | +r | 0.7 |
| +r | -r | 0.3 |
| -r | +r | 0.1 |
| -r | -r | 0.9 |

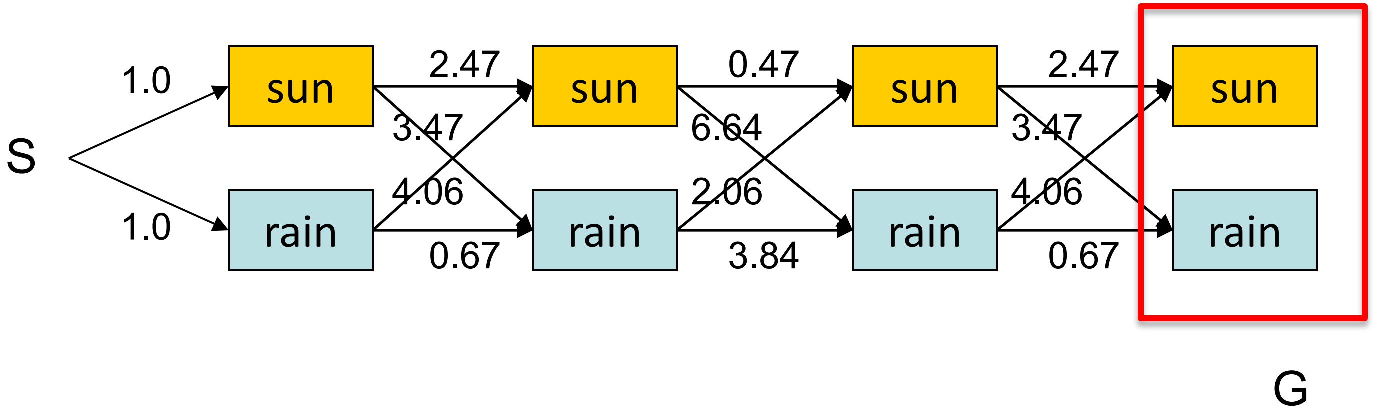
| $R_t$ | $U_t$ | $P(U_t\|R_t)$ |
|---|---|---|
| +r | +u | 0.9 |
| +r | -u | 0.1 |
| -r | +u | 0.2 |
| -r | -u | 0.8 |

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 uniform cost graph search

# Viterbi Algorithm Pseudocode

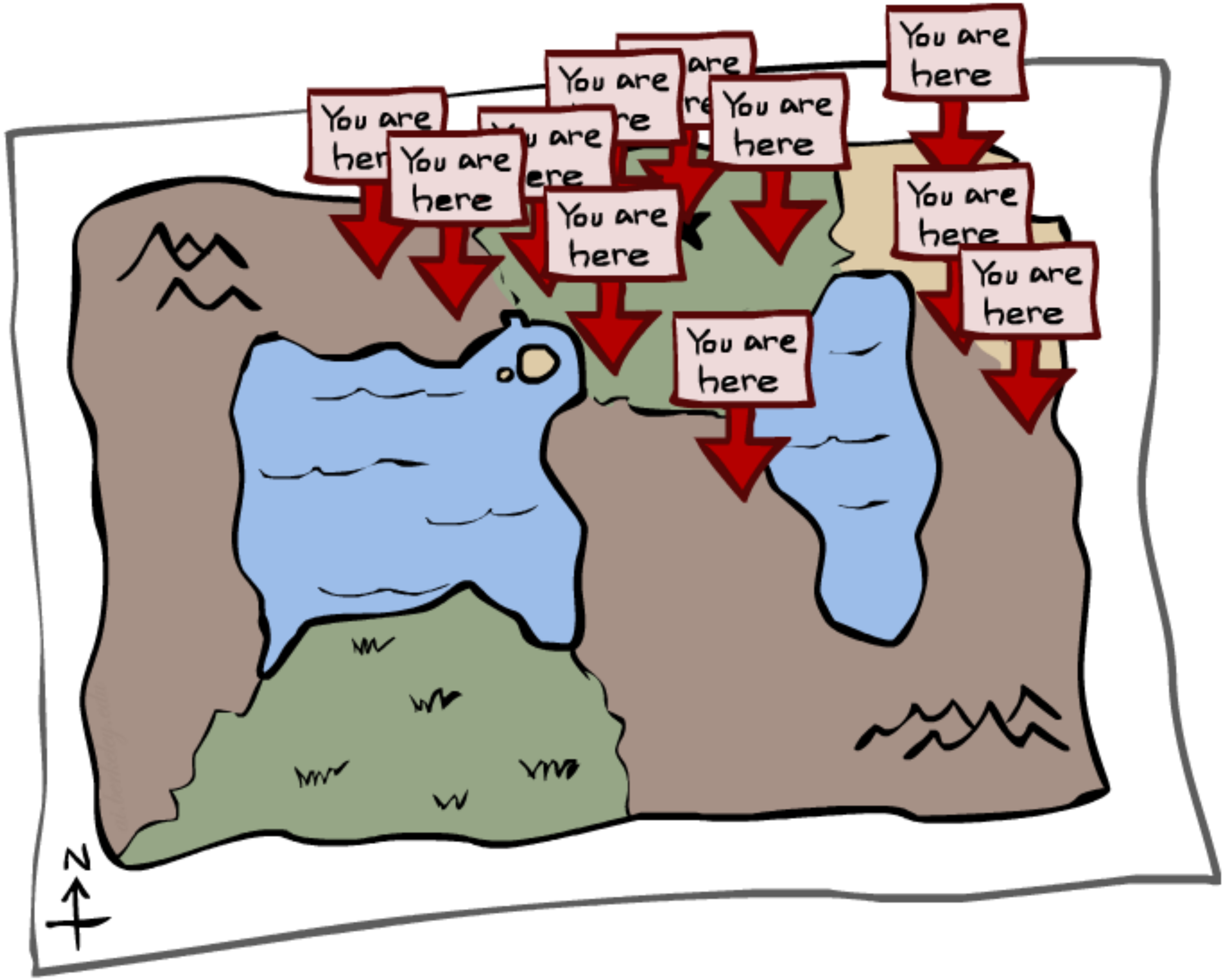
**function** *VITERBI*$(O, S, \Pi, Y, A, B) : X$
  **for** each state $i = 1, 2, \ldots, K$ **do**
    $T_1[i, 1] \leftarrow \pi_i \cdot B_{iy_1}$
    $T_2[i, 1] \leftarrow 0$
  **end for**
  **for** each observation $j = 2, 3, \ldots, T$ **do**
    **for** each state $i = 1, 2, \ldots, K$ **do**
      $T_1[i, j] \leftarrow \max_k (T_1[k, j-1] \cdot A_{ki} \cdot B_{iy_j})$
      $T_2[i, j] \leftarrow \arg\max_k (T_1[k, j-1] \cdot A_{ki} \cdot B_{iy_j})$
    **end for**
  **end for**
  $z_T \leftarrow \arg\max_k (T_1[k, T])$
  $x_T \leftarrow s_{z_T}$
  **for** $j = T, T-1, \ldots, 2$ **do**
    $z_{j-1} \leftarrow T_2[z_j, j]$
    $x_{j-1} \leftarrow s_{z_{j-1}}$
  **end for**
  **return** $X$
**end function**

| | |
|---|---|
| Observation Space | $O = \{o_1, o_2, \ldots, o_N\}$ |
| State Space | $S = \{s_1, s_2, \ldots, s_K\}$ |
| Initial probabilities | $\Pi = (\pi_1, \pi_2, \ldots, \pi_K)$ |
| Observations | $Y = (y_1, y_2, \ldots, y_T)$ |
| Transition Matrix | $A \in \mathbb{R}^{K \times K}$ |
| Emission Matrix | $B \in \mathbb{R}^{K \times N}$ |

Matrix $T_1[i, j]$ stores probabilities of most likely path so far with $x_j = s_i$

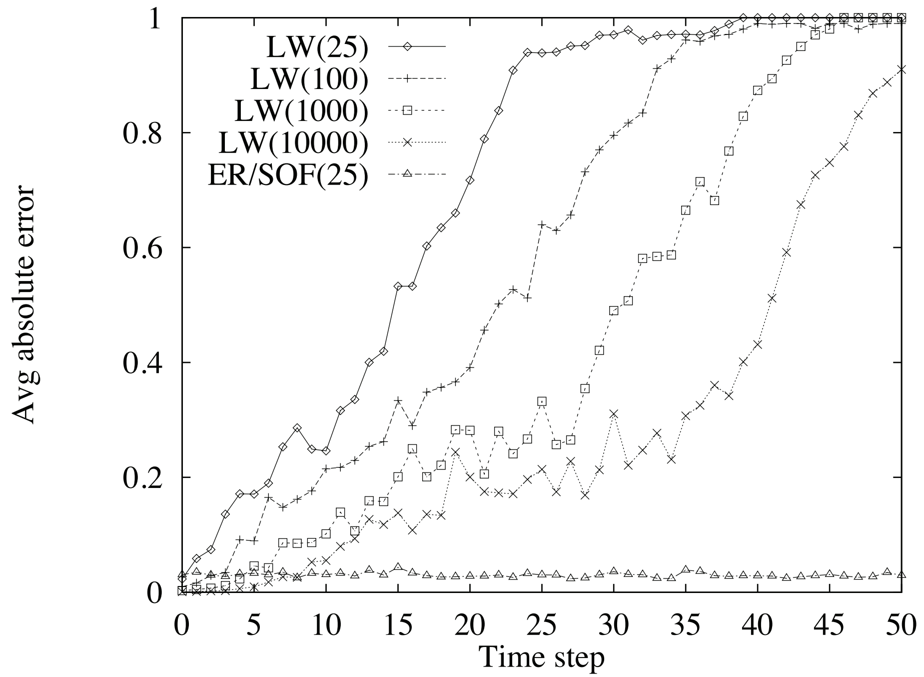
Matrix $T_2[i, j]$ stores $x_{j-1}$ of most likely path so far with $x_j = s_i$

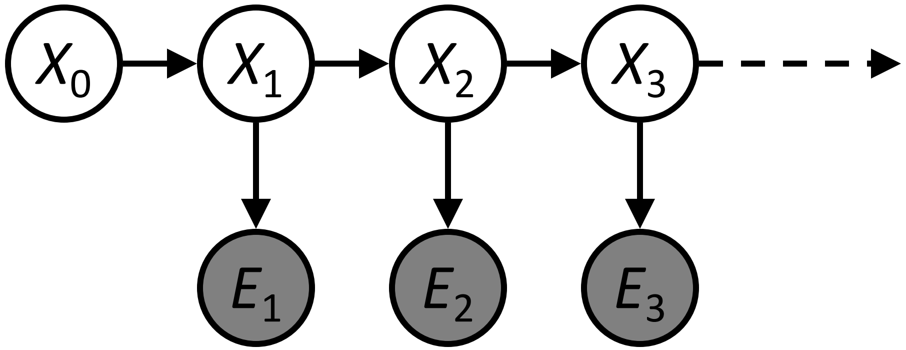
# Particle Filtering



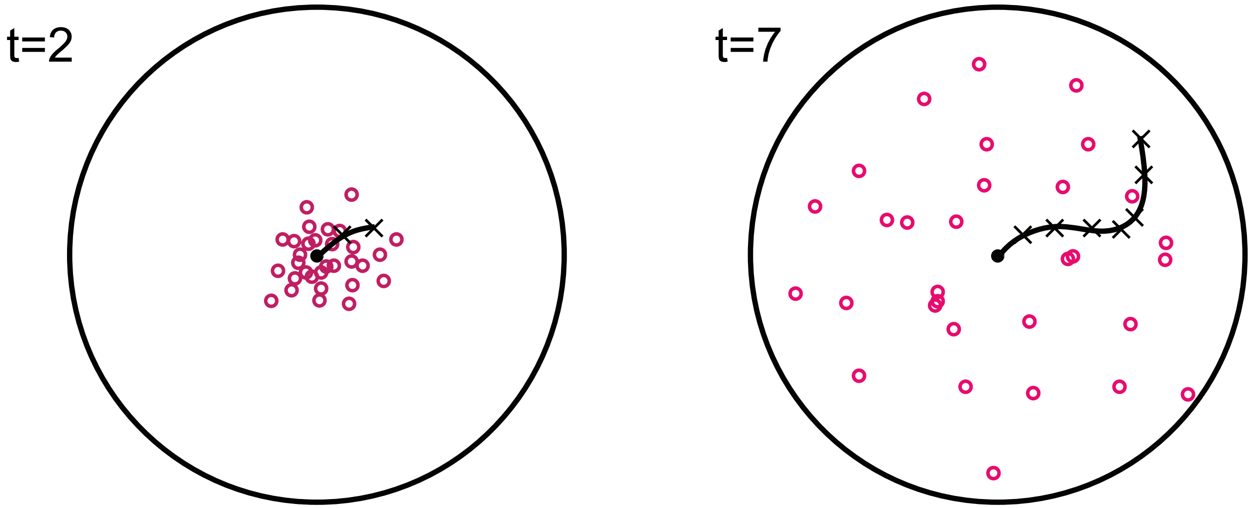
# Approximate Inference on HMMs

- When $|X|$ is more than $10^6$ or so (e.g., 3 ghosts in a 10x20 world), exact inference becomes infeasible
- Likelihood weighting fails completely – number of samples needed grows ***exponentially*** with $T$



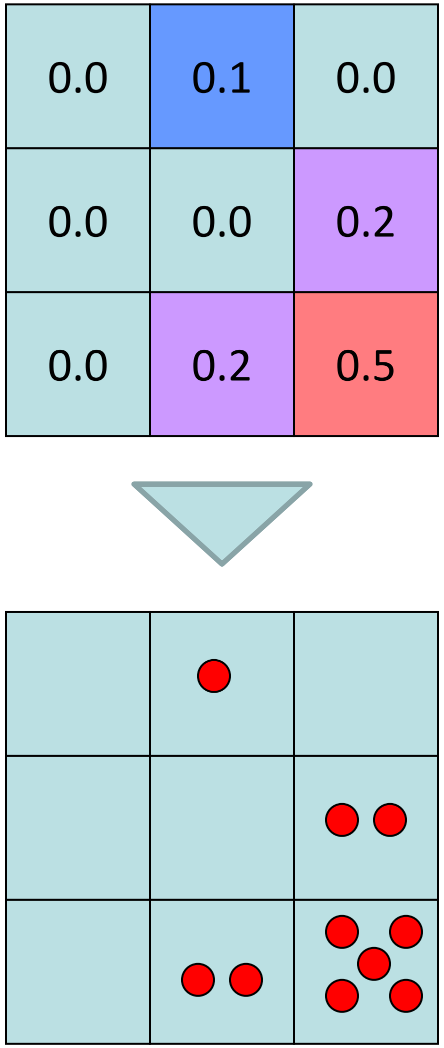
# We need a new idea!




- The problem: sample state trajectories go off into low-probability regions, ignoring the evidence; too few “reasonable” samples
- Solution: kill the bad ones, make more of the good ones
- This way the population of samples stays in the high-probability region
- This is called ***resampling*** or survival of the fittest

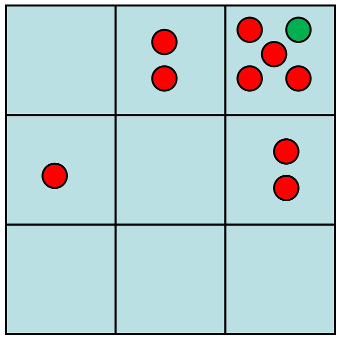
# Particle Filtering

- Filtering: approximate solution
- Sometimes |X| is too big to use exact inference
  - |X| may be too big to even store $P(X \mid e_{1:T})$
- 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



# Representation: Particles

- Our representation of P(X) is now a list of N particles (samples)
  - Generally, N << |X|
- 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)

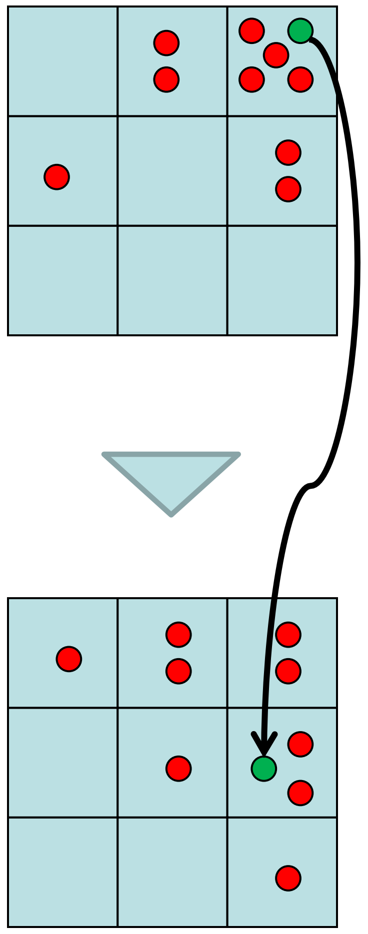
# Particle Filtering: Elapse Time

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

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

  - This is like prior sampling – sample's 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: Incorporate Observation

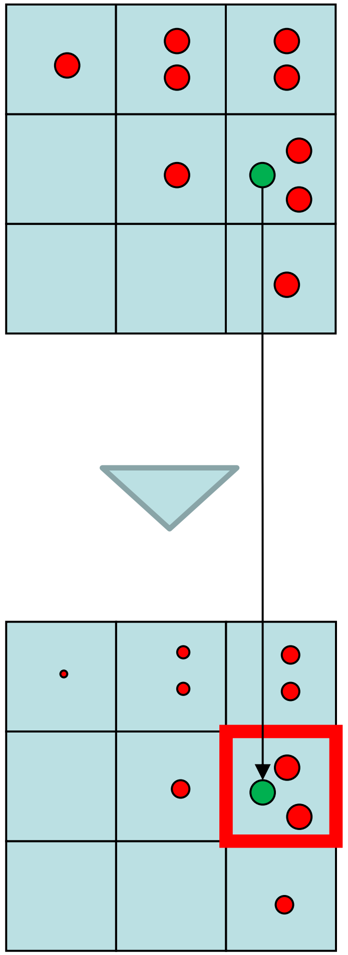
- After observing Evidence $e_{t+1}$:
  - Don't sample observation, fix it
  - Similar to likelihood weighting, downweight samples based on the evidence

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

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

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

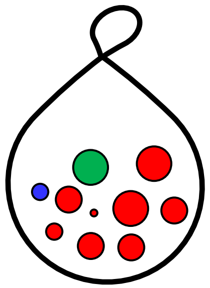
Particles:
(3,2)
(2,3)
(3,2)
(3,1)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(2,2)



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

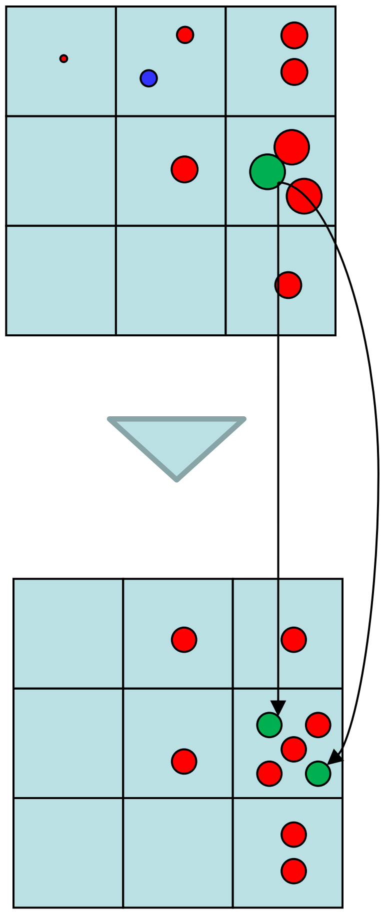
# Particle Filtering: Resample

- 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



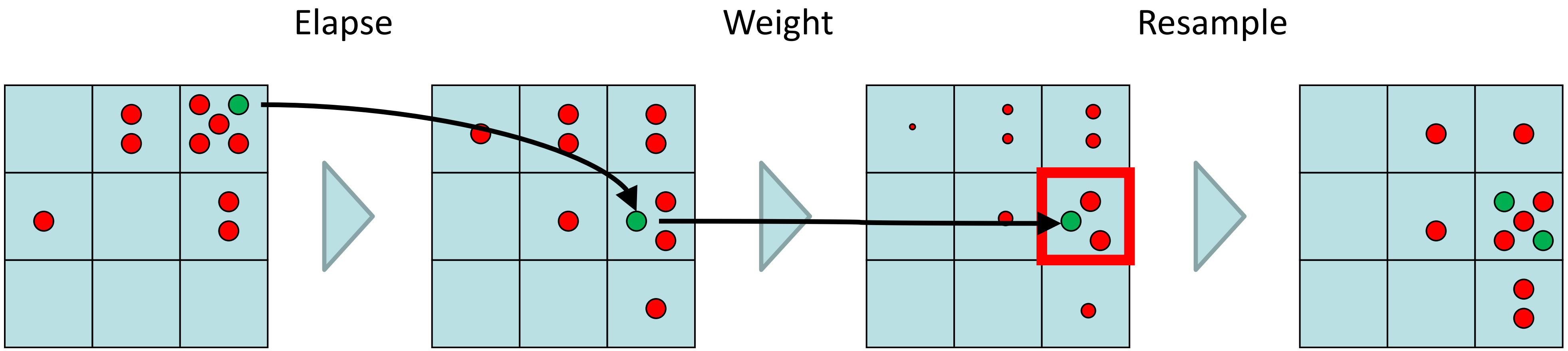
Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

(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) |

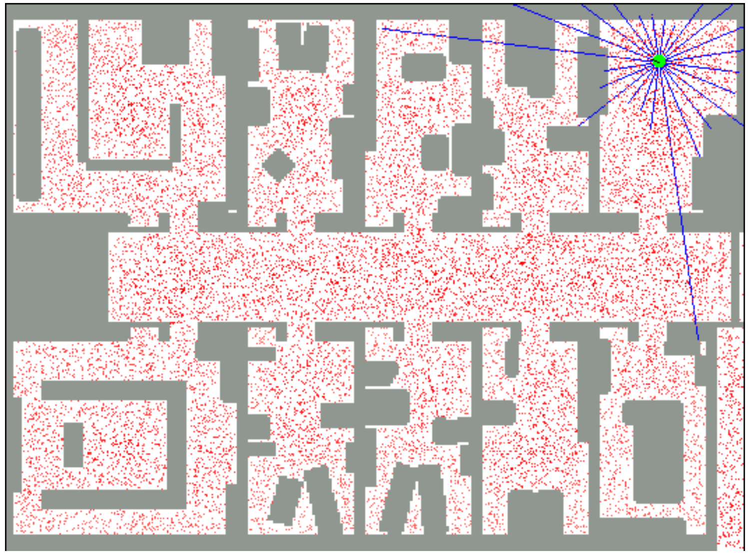
# Video of Demo – Moderate Number of Particles

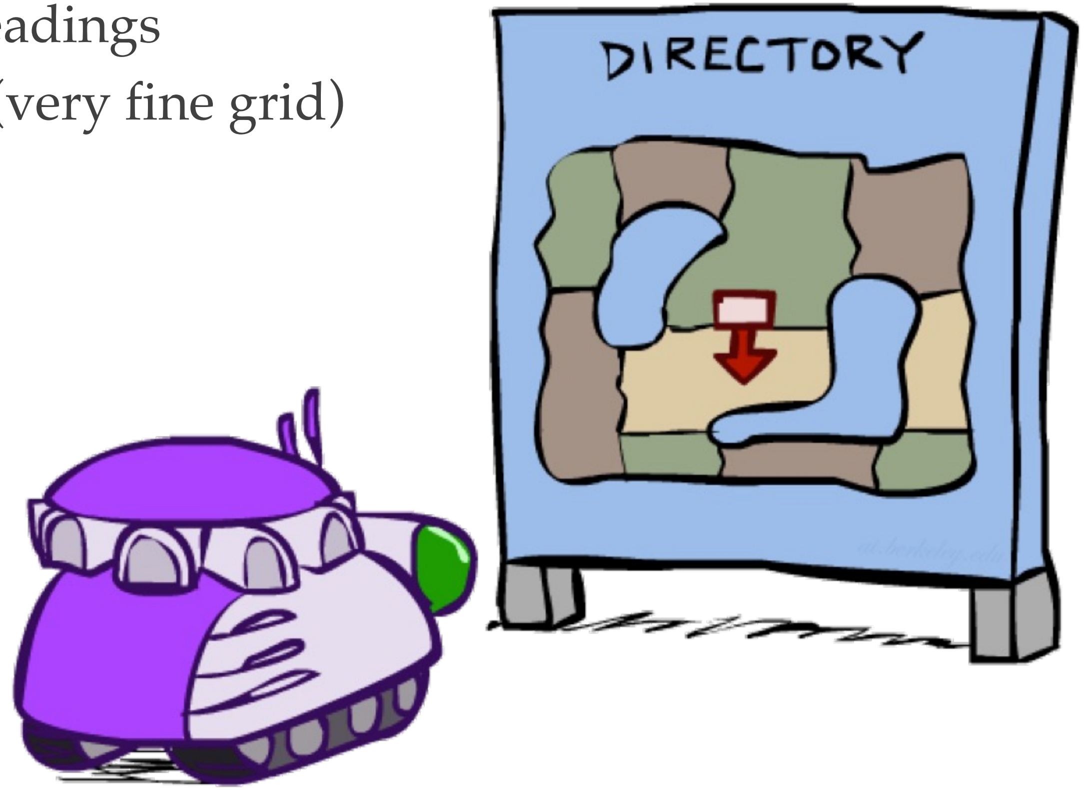
# Video of Demo – One Particle

# Video of Demo – Huge Number of Particles

# Robot Localization

- In robot localization:
  - Know the map, but not the robot's position
  - Observations may be vectors of range finder readings
  - State space and readings typically continuous (very fine grid) and so we cannot store $P(X_t \mid e_{1:t})$
  - Particle filtering is a main technique



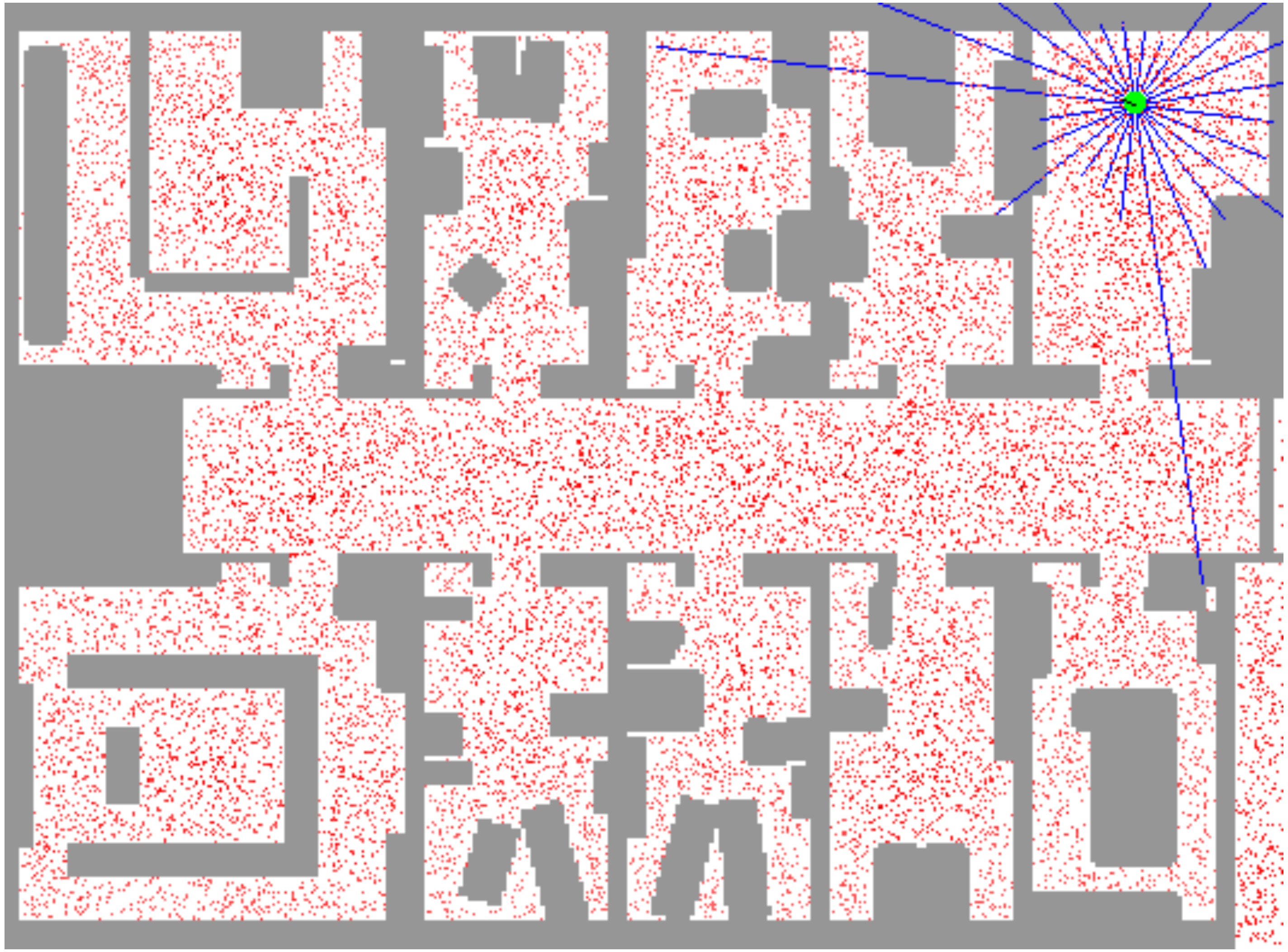


# Particle Filter Localization (Sonar)


[Dieter Fox, et al.]

[Video: global-sonar-uw-annotated.avi]

# Particle Filter Localization (Laser)

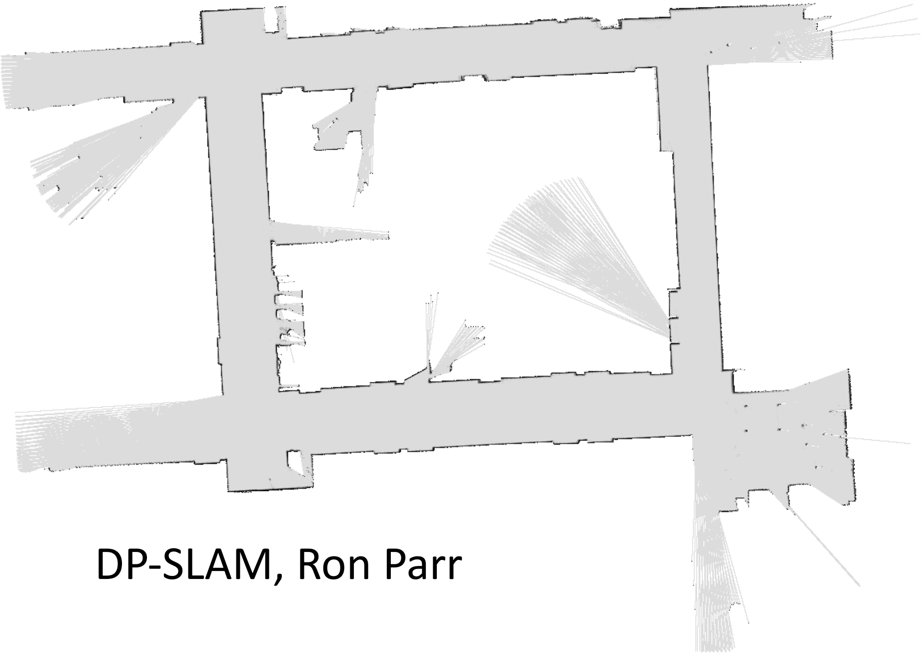


[Dieter Fox, et al.]

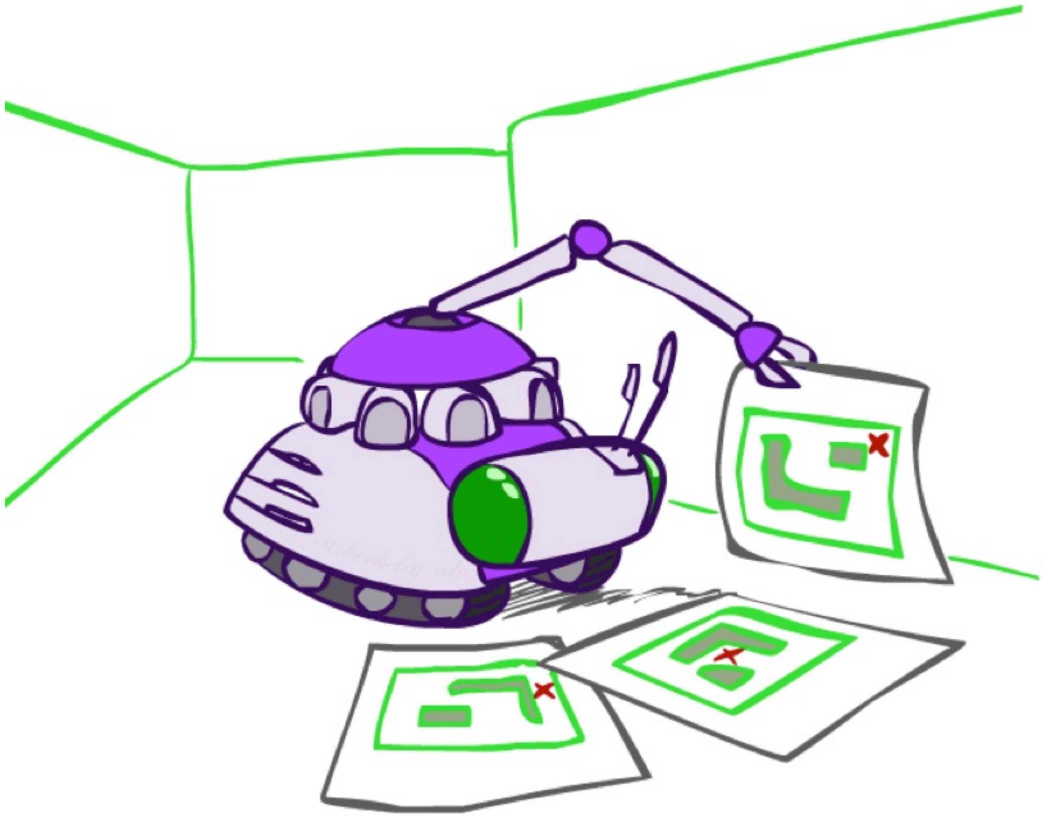
[Video: global-floor.gif]

# 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

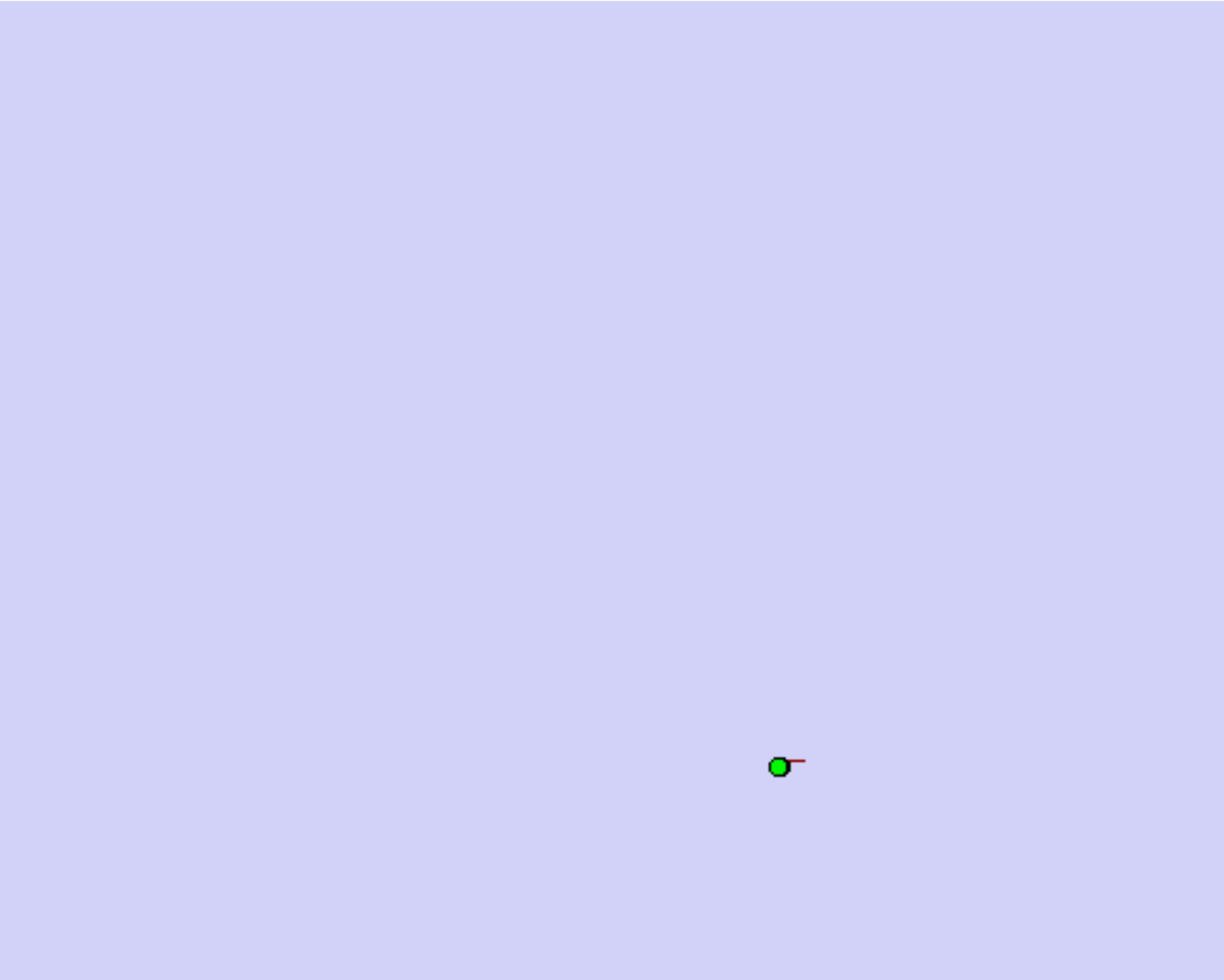


DP-SLAM, Ron Parr



[Demo: PARTICLES-SLAM-mapping1-new.avi]

# Particle Filter SLAM – Video



[Demo: PARTICLES-SLAM-fastslam.avi]