CS 188: Artificial Intelligence
Fall 2007

Lecture 21: Particle Filtering
11/08/2007

Dan Klein – UC Berkeley

Announcements

- Project 5 is up, due 11/19 (an extension of 4)
- Probability review and BN/HMM recap sessions

Laws of Probability

- Marginalization

$$P(a) = \sum_b P(a, b)$$

- Definition of conditional probability

$$P(a|b) = P(a, b)/P(b)$$

- Chain rule

$$P(a, b, c) = P(a)P(b|a)P(c|a, b)$$

- Combinations, e.g. conditional chain rule

$$P(b, c|a) = P(b|a)P(c|a, b)$$

Some More Laws

- Chain rule (always true)

$$P(a, b, c) = P(a)P(b|a)P(c|a, b)$$

- With A and C independent given B

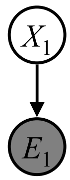
$$P(a, b, c) = P(a)P(b|a)P(c|a)$$

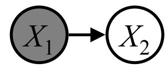
- If we want a conditional distribution over A, can just normalize the corresponding joint wrt A

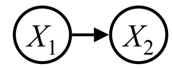
$$P(a|b) = P(a, b)/P(b)$$
$$\propto_A P(a, b)$$

Recap: Some Simple Cases

Models







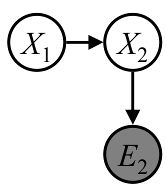
Queries

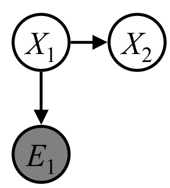
$$P(X_1|e_1) \qquad P(X_2|x_1) \qquad P(X_2)$$

$$\begin{aligned} P(x_1|e_1) &= P(x_1, e_1)/P(e_1) \\ &\propto_{X_1} P(x_1, e_1) \\ &= P(x_1)P(e_1|x_1) \end{aligned}$$

$$\begin{aligned} P(x_2) &= \sum_{x_1} P(x_1, x_2) \\ &= \sum_{x_1} P(x_1)P(x_2|x_1) \end{aligned}$$

Recap: Some Simple Cases



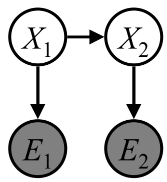


$$P(X_2|e_2) \qquad P(X_2|e_1)$$

$$\begin{aligned} P(x_2|e_2) &= P(x_2, e_2)/P(e_2) \\ &\propto_{X_2} P(x_2, e_2) \\ &= P(x_2)P(e_2|x_2) \\ &= \sum_{x_1} P(x_1, x_2)P(e_2|x_2) \\ &= P(e_2|x_2) \sum_{x_1} P(x_1)P(x_2|x_1) \end{aligned}$$

$$\begin{aligned} P(x_2|e_1) &= P(x_2, e_1)/P(e_1) \\ &\propto_{X_2} P(x_2, e_1) \\ &= \sum_{x_1} P(x_1, x_2, e_1) \\ &= \sum_{x_1} P(x_2|x_1)P(e_1|x_1)P(x_1) \end{aligned}$$

Recap: Some Simple Cases



$$P(X_2|e_2, e_1)$$

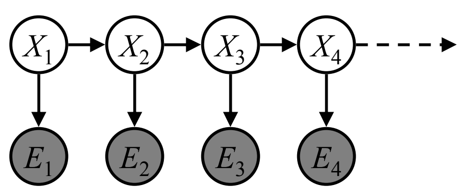
$$\propto_{X_2} P(x_2, e_1, e_2)$$

$$= \sum_{x_1} P(x_1, x_2, e_1, e_2)$$

$$= \sum_{x_1} P(x_1)P(x_2|x_1)P(e_1|x_1)P(e_2|x_2)$$

$$= P(e_2|x_2) \sum_{x_1} P(x_2|x_1)P(e_1|x_1)P(x_1)$$

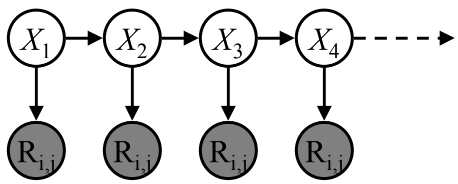
Hidden Markov Models



- An HMM is
  - Initial distribution: $P(X_1)$
  - Transitions: $P(X|X_{-1})$
  - Emissions: $P(E|X)$

Battleship HMM

- $P(X_1)$ = uniform
- $P(X|X')$ = usually move according to fixed, known patrol policy (e.g. clockwise), sometimes move in a random direction or stay in place
- $P(R_{ij}|X)$ = as before: depends on distance from ships in x to (i,j) (really this is just one of many independent evidence variables that might be sensed)



1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$P(X_1)$

1/6	1/6	1/2
0	1/6	0
0	0	0

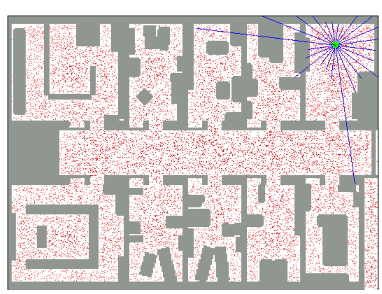
P(X|X'=<1,2>)

Filtering / Monitoring

- Filtering, or monitoring, is the task of tracking the belief state:

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

- We start with B(X) in an initial setting, usually uniform
- As time passes, or we get observations, we update B(X)



The Forward Algorithm

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

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

- We can derive the following updates

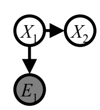
We can normalize as we go if we want to have P(x|e) at each time step, or just once a the end…

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

Belief Updates

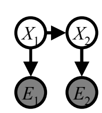
- Every time step, we start with current P(X | evidence)
- We update for time:

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



- We update for evidence:

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

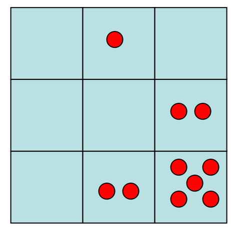


- The forward algorithm does both at once (and doesn't normalize)
- Problem: space is |X| and time is $|X|^2$ per time step

Particle Filtering

- Sometimes |X| is too big to use exact inference
  - |X| may be too big to even store B(X)
  - E.g. X is continuous
  - $|X|^2$ may be too big to do updates
- Solution: approximate inference
  - Track samples of X, not all values
  - Time per step is linear in the number of samples
  - But: number needed may be large
- This is how robot localization works in practice

0.0	0.1	0.0
0.0	0.0	0.2
0.0	0.2	0.5

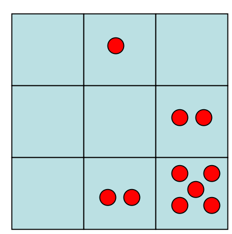


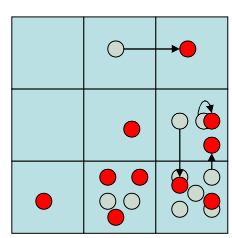
Particle Filtering: Time

- 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 probs
  - Here, most samples move clockwise, but some move in another direction or stay in place
- This captures the passage of time
  - If we have enough samples, close to the exact values before and after (consistent)





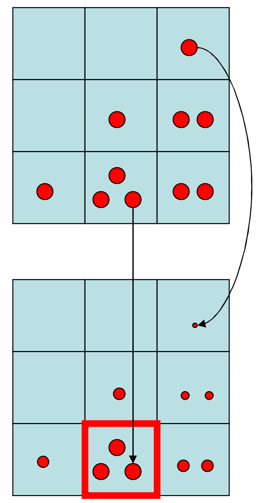
Particle Filtering: Observation

- Slightly trickier:
  - We don't sample the observation, we fix it
  - This is similar to likelihood weighting, so we downweight our samples based on the evidence

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

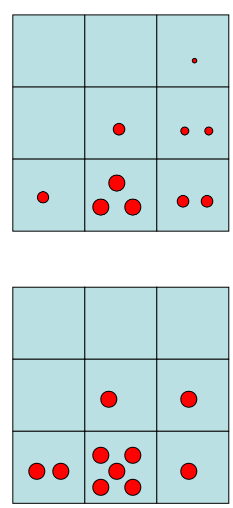
$$B(X) \propto P(e|X)B'(X)$$

  - Note that, as before, the probabilities don't sum to one, since most have been downweighted (in fact they sum to an approximation of P(e))



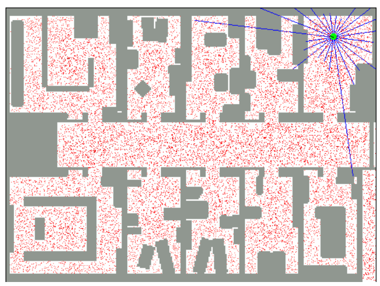
Particle Filtering: Resampling

- 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



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
- [DEMOS]



SLAM

- SLAM = Simultaneous Localization And Mapping
  - We do not know the map or our location
  - Our belief state is over maps and positions!
  - Main techniques: Kalman filtering (Gaussian HMMs) and particle methods
- [DEMOS]

DP-SLAM, Ron Parr

Most Likely Explanation

- Question: most likely sequence ending in x at t?
  - E.g. if sun on day 4, what's the most likely sequence?
  - Intuitively: probably sun all four days
- Slow answer: enumerate and score

$$P(X_t = sun) = \max_{x_1 \ldots x_{t-1}} P(x_1, \ldots x_{t-1}, sun)$$

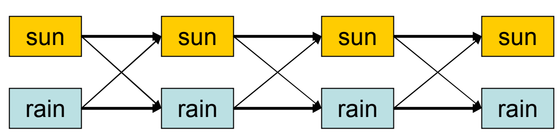
$$P(X_1 = sun)P(X_2 = sun|X_1 = sun)P(X_3 = sun|X_2 = sun)P(X_4 = sun|X_3 = sun)$$

$$P(X_1 = sun)P(X_2 = rain|X_1 = sun)P(X_3 = sun|X_2 = rain)P(X_4 = sun|X_3 = sun)$$

⋮

Mini-Viterbi Algorithm

- Better answer: cached incremental updates

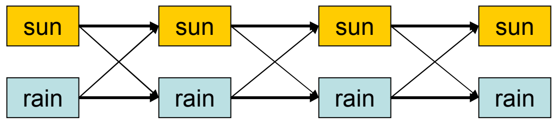


- Define: $m_t[x] = \max_{x_{1:t-1}} P(x_{1:t-1}, x)$

$$a_t[x] = \arg\max_{x_{1:t-1}} P(x_{1:t-1}, x)$$

- Read best sequence off of m and a vectors

Mini-Viterbi



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

$$= \max_{x_{1:t-1}} P(x_{1:t-1}) P(x|x_{t-1})$$

$$= \max_{x_{t-1}} P(x_t|x_{t-1}) \max_{x_{1:t-2}} P(x_{1:t-1})$$

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

$$m_1[x] = P(x_1)$$

Viterbi Algorithm

- Question: what is the most likely state sequence given the observations?
  - Slow answer: enumerate all possibilities
  - Better answer: cached incremental version

$$x^*_{1:T} = \arg\max_{x_{1:T}} P(x_{1:T}|e_{1:T})$$

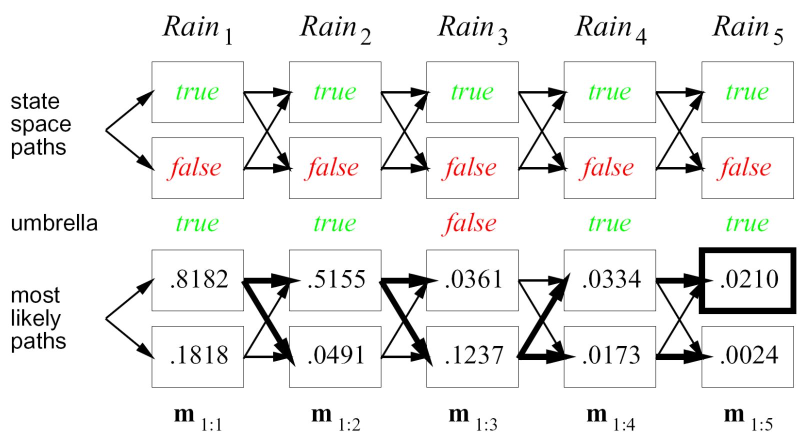
$$m_t[x_t] = \max_{x_{1:t-1}} P(x_{1:t-1}, x_t, e_{1:t})$$

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

$$= P(e_t|x_t) \max_{x_{t-1}} P(x_t|x_{t-1}) \max_{x_{1:t-2}} P(x_{1:t-1}, e_{1:t-1})$$

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

Example


Rain 1
Rain 2
Rain 3
Rain 4
Rain 5
state space paths
true
false
umbrella
true
true
false
true
true
most likely paths
.8182
.1818
.5155
.0491
.0361
.1237
.0334
.0173
.0210
.0024
m 1:1
m 1:2
m 1:3
m 1:4
m 1:5