Particle Filters

Ron Parr
CompSci 370
Department of Computer Science
Duke University

Outline

- Problem: Track state over time
  - State = position, orientation of robot (or condition of patient, position of airplane, status of factory, etc.)
- Challenge: State is not observed directly
- Solution: Tracking using a model
  - Exact tracking (previous lecture) not always possible for large or continuous state spaces
  - Approximate tracking using sampling (this lecture)

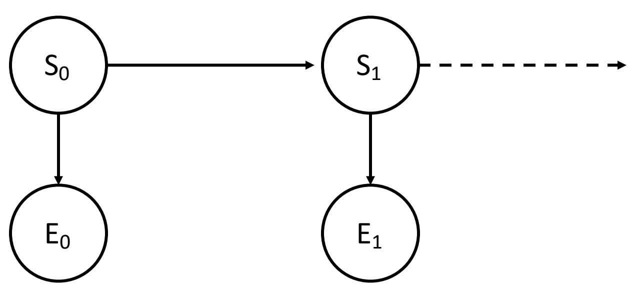
Applications

- Activity recognition by mobile devices
  (hidden state is the activity)

- Robot self localization
  (hidden state is robot position, orientation)

- Tracking objects with limited observations
  (tracking pedestrians with/cars with surveillance cameras)

Recall Bayes Net View of HMMs

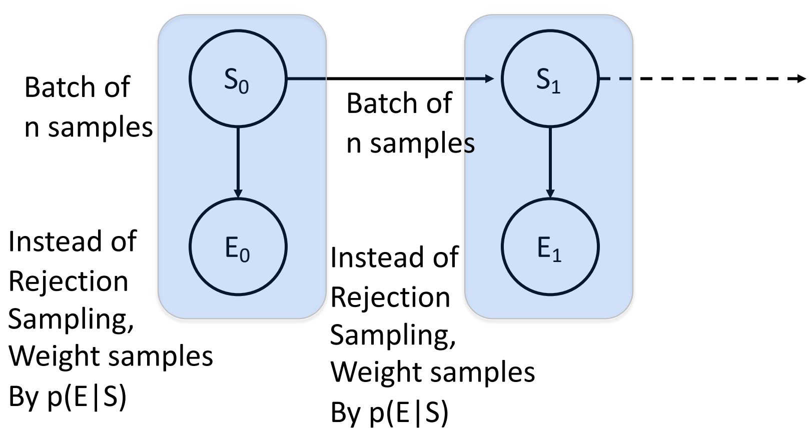


Note: These are random variables, not states!

Recall Sampling Approach to BNs

- Treat Bayes net as a "generative model"
- Sample variables with no unsampled parents
- Marginal probabilities are relative frequencies in population of samples
- Challenges:
  - In tracking, you are never "done" sampling
  - Observations may be continuous – probability 0 that sampled observation will match actual one

Idea: Sample in batches (sequential importance sampling)



Toy Example

- Robot is monitoring door to the AI lab
- D = variable for status of door (True = open)
- Initially we will ignore observations

- Define Markov model for behavior of door:

$$P(d_{t+1} \mid d_t) = 0.8$$
$$P(d_{t+1} \mid \bar{d}_t) = 0.3$$

Example Problem

Suppose we believe the door was open with prob. 0.7 at time t.

What is the prob. that it will be open at time t+1?

$$P(d_{t+1} \mid d_t) = 0.8$$
$$P(d_{t+1} \mid \bar{d}_t) = 0.3$$

Staying open — Switching from closed to open

$$P(d_{t+1}) = P(d_{t+1} \mid d_t)P(d_t) + P(d_{t+1} \mid \bar{d}_t)P(\bar{d}_t)$$
$$= 0.8 * 0.7 + 0.3 * 0.3 = 0.65$$

Example Continued

$$P(d_{t+1} \mid d_t) = 0.8$$
$$P(d_{t+1} \mid \overline{d}_t) = 0.3$$

- Pick n=1000
  - 700 door open samples
  - 300 door closed samples
- For each sample generate a next state
  - For open samples use prob. 0.8 for next state open
  - For closed samples use prob. 0.3 for next state open
- Count no. of open and closed next states

- Can prove that in limit of large n, our count will equal true probability (0.65)

Example Revisited

- D = Door status
- O = Robot's observation of door status
- Observations may not be completely reliable!

$$P(d_{t+1} \mid d_t) = 0.8$$
$$P(d_{t+1} \mid \overline{d}_t) = 0.3$$
$$P(o \mid d) = 0.6$$
$$P(o \mid \overline{d}) = 0.2$$

Modified Sampling

- Problem: How do we adjust sampling to handle evidence?
- Solution: Weight each sample by the probability of the observations
- Called importance sampling (IS), or likelihood weighting (LW)

- *Does the right thing* for large n

Example with evidence

$$P(d_{t+1} \mid d_t) = 0.8$$
$$P(d_{t+1} \mid \bar{d}_t) = 0.3$$
$$P(o \mid d) = 0.6$$
$$P(o \mid \bar{d}) = 0.2$$

- Suppose we observe door **closed** at t+1
- Pick n=1000
  - 700 door open samples
  - 300 door closed samples
- For each sample generate a next state
  - For open samples use prob. 0.8 for next state open
  - For closed samples use prob. 0.3 for next state open
  - If next state is open, weight by 0.4
  - If next state is closed, weight by 0.8

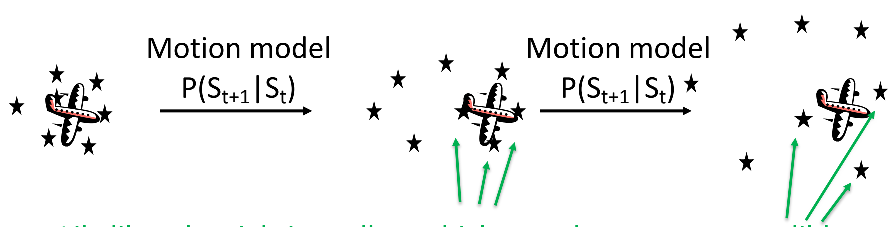
  ← importance weights
- Compute weighted sum of no. of open and closed states to estimate state probabilities at time t+1

Problems with IS (LW)

- Sequential importance sampling (SIS) does the right thing for the limit of large numbers of samples
- Problems for finite numbers of samples:
  - *Effective* sample size (total weight of samples) drops
  - Eventually
    - Something unlikely happens, or
    - A sequence of individually somewhat likely events has the effect of a single unlikely event, and
    - Population of samples drifts away from reality
- Over time: Estimates become unreliable

Example of "Drift"

- Suppose you are tracking an aircraft
- Each sample corresponds to a possible aircraft position
- You have a physics-based simulation model that predicts:
  next_pos = current_pos + velocity*time + noise
- Over time, samples can drift from reality



Likelihood weighting tell us which samples are more credible

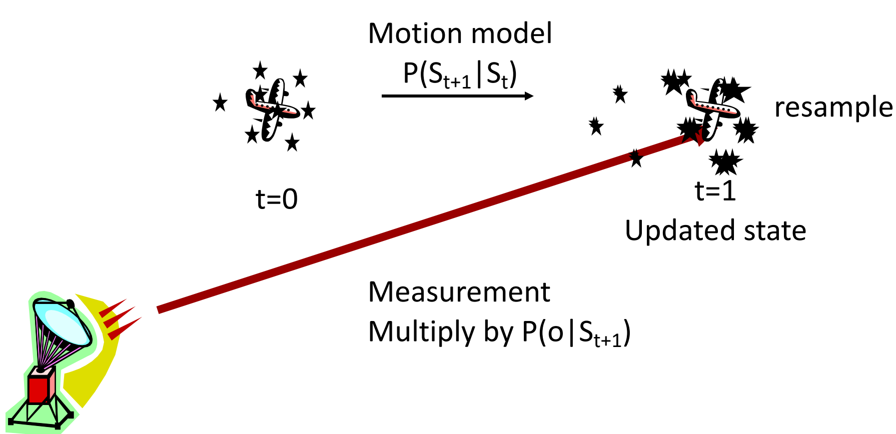
But doesn't fix underlying drift problem, i.e., that most become not very credible over time

Solution: SISR (PF)

Sequential **I**mportance **S**ampling with ***Resampling*** = **P**article **F**ilter

- Maintain n samples for each time step
- Repeat n times:
  - Draw sample from $p(S_t)$ (w/replacement) (according to current weights) ← resampling
  - Simulate transition to S_{t+1}
  - Weight samples by evidence & normalize
- Note: Works for continuous as well as discrete vars!

Particle Filter for Trajectory Tracking



Key Points About Particle Filters

- Given a finite budget of samples:
- PF reallocates resources to samples that better match reality

- Leads to more relevant samples
- Less concern about drift

Example: Robot Localization

- Particle filters combine:
  - A model of state change
  - A model of sensor readings
- To track objects with hidden state over time

- Robot application:
  - Hidden state: Robot position, orientation
  - State change model: Robot motion model
  - Sensor model: Sonar/LiDAR error model

- Note: Robot is tracking itself!

Main Loop

- Sample n robot states
- For each state
  - Simulate next state (action model)
  - Weight states (observation model)
  - Normalize
- Repeat

Robot States

- Robot has X,Y,Z,θ
- Usually ignore z
  - assume floors are flat
  - assume robot stays on one floor
- Form of samples
  - $(X_i, Y_i, \theta_i, p_i)$
  - $\sum_i p_i = 1$

Main Loop

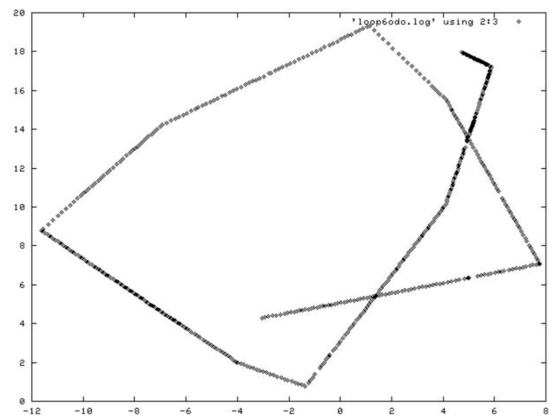
- Sample n robot states
- For each state
  - Simulate next state (action model)
  - Weight states (observation model)
  - Normalize
- Repeat

Main Loop

- Sample n robot states
- For each state
  - Simulate next state (action model)
  - Weight states (observation model)
  - Normalize
- Repeat

Motion Model

- How far has the robot traveled?
- Robots have (noisy) odometers:

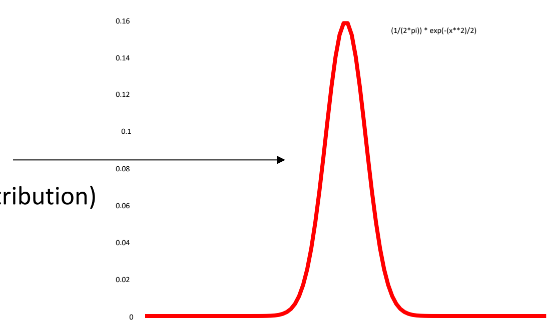


Actual path was a closed loop on the second floor!

Odometer Model

- Odometer is:
  - Relatively accurate model of wheel turn
  - Very inaccurate model of actual movement
- Actual position = odometer X,Y,θ + random noise

Classic,
Bell-shaped
Curve
(normal distribution)



Simulation Implementation

- Start with odometer readings
- Add linear correction factor
  - $X = a_x*X+b_x$
  - $Y = a_y*Y+b_y$
  - $\theta = a_\theta*\theta+b_\theta$

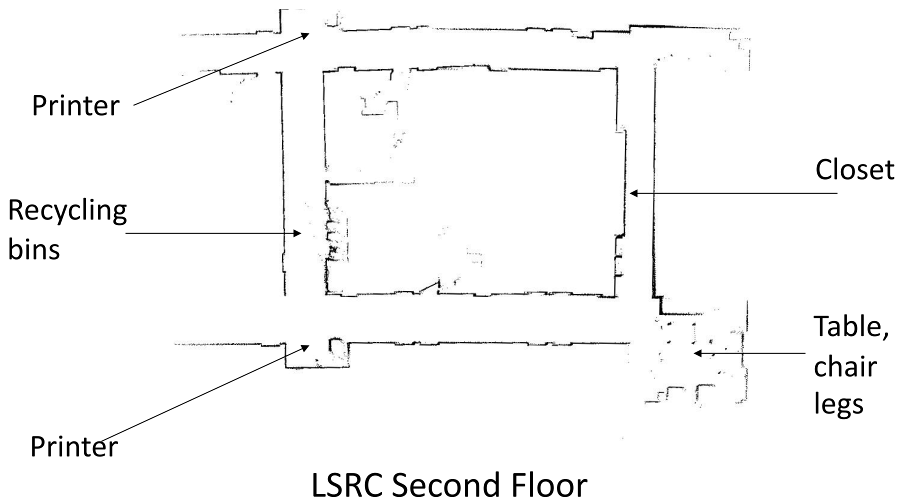
  Linear correction (determined experimentally)
- Add noise from the normal distribution
  - $X = X + N(0,s_x)$
  - $Y = Y + N(0,s_x)$
  - $\theta = \theta + N(0,s_\theta)$

  $N(\mu,s)$ returns random noise from normal distribution with mean μ and standard deviation s (standard deviation determined experimentally)

Main Loop

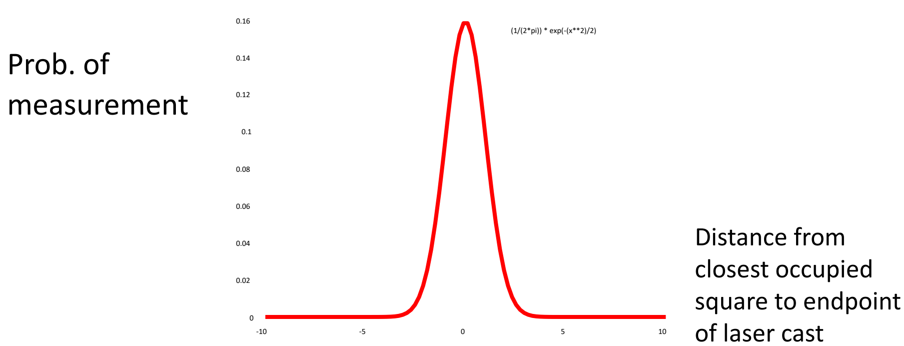
- Sample n robot states
- For each state
  - Simulate next state (action model)
  - Weight states (observation model)
  - Normalize
- Repeat

Internal Map Representation



Laser Error Model

- Laser measures distance at 180 one degree increments in front of the robot (height is fixed)
- Laser rangefinder errors also have a normal distribution



Laser Error Model Contd.

- Probability of error in measurement k for sample i (normal)

$$p_{ik}(x_k) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x_k^2}{2\sigma^2}}$$

- x_k is distance of laser endpoint to closest obstacle
- σ is standard deviation in this measurement (estimated experimentally), usually a few cm.

Laser Error Model Contd.

- Laser measurements are independent
- Weight of sample is product of errors:

$$p_i = \prod_k p_{ik}$$

- Note: Good to bound x to prevent a single bad measurement from making p_i too small

Main Loop

- Sample n robot states
- For each state
  - Simulate next state (action model)
  - Weight states (observation model)
  - Normalize
- Repeat

Summary

- HMMs provide mathematical basis for tracking
- Exact solution intractable for large state spaces
- Particle filters approximate the exact HMM solution using **sampling**, **simulation, weighting**