Particle Filters

Ron Parr
CPS 1/296

Outline

- Problem: Track state over time
  - State = position, orientation of robot (condition of patient, position of airplane, status of factory, etc.)
- Challenge: State is not observed directly
- Solution: Tracking using a model
  - Exact
  - Approximate (Particle filter)

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:
  
  \[
  P(D_{t+1} | D_t) = 0.8 \\
  P(D_{t+1} | \overline{D}_t) = 0.3
  \]

Problem

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

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

\[
P(D_{t+1} | D_t) = 0.8 \\
P(D_{t+1} | \overline{D}_t) = 0.3
\]

Staying open \quad Switching from closed to open

\[
P(D_{t+1} | D_t) = P(D_{t+1} | D_t)P(D_t) + P(D_{t+1} | \overline{D}_t)P(\overline{D}_t) \\
= 0.8 \times 0.7 + 0.3 \times 0.3 = 0.65
\]

Generalizing

- Suppose states are not binary:
  \[
  p(S_{t+1}) = \sum_{S_t} p(S_{t+1} | S_t) p(S_t)
  \]
- Suppose states are continuous
  \[
  p(S_{t+1}) = \int p(S_{t+1} | S_t) p(S_t) dS_t
  \]
- Issue: For large or continuous state spaces this may be hard to deal with exactly

Monte Carlo Approximation

(Sampling)

- We can approximate a nasty integral by sampling and counting:
  \[
  p(S_{t+1}) = \int p(S_{t+1} | S_t) p(S_t) dS_t
  \]
- Repeat n times:
  - Draw sample from $p(S_t)$
  - Simulate transition to $S_{t+1}$
- Count proportion of states for each value of $S_{t+1}$
Example

- 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!

\[ P(D_{n+1} | D_n) = 0.8 \]
\[ P(D_{n+1} | \overline{D}_n) = 0.3 \]
\[ P(O | D) = 0.6 \]
\[ P(O | \overline{D}) = 0.2 \]

Modified Sampling

- Problem: How do we adjust sampling to handle evidence?
- Solution: Weight each sample by the probability of the observations
- Called importance sampling, or likelihood weighting
- Does the right thing for large n

Example with evidence

- 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
- Compute weighted sum of no. of open and closed states

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 drops over time
  - Unlikely events are only small fraction of sample population
  - Eventually
    - Something unlikely happens
    - A sequence of individually likely events has the effect of a single unlikely event
    - Estimates become unreliable bit based on a small no. of samples

Solution: SISR (PF)

- Maintain n samples for each time step
- Repeat n times:
  - Draw sample from p(S_n)
    (according to current weights)
  - Simulate transition to S_{n+1}
  - Weight samples by evidence
- Count proportion of states for each value of S_{n+1}
Monte Carlo Approximation (Particle Filter)

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: Laser range finder 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, \( \theta \)
- Usually ignore z
  - assume floors are flat
  - assume robot stays on one floor
- Form of samples
  - \( (X_i, Y_i, \theta_i, p_i) \)
  - \( \sum p_i = 1 \)

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

Sampling Robot States
- Need to generate n new samples from our previous set of n samples
- Draw n new robot states with replacement
  - for \( i = 1 \) to \( n \)
    - \( r = \text{rand}[0...1] \)
    - \( \text{temp} = 0 \)
    - while(\text{temp} < r)
      - \( \text{temp} = \text{temp + samples[i].p} \)
      - \( k = \text{int}\) \( \text{temp} \)
    - \( \text{newsamples}[i] = \text{samples}[k-1] \) (off this should copy)
- \( \text{samples} = \text{newsamples} \)
Main Loop

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

Action Model

- How far has the robot traveled?
- What does the odometer tell us?

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

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
  - \( X = X + N(0, \sigma_x) \)
  - \( Y = Y + N(0, \sigma_y) \)
  - \( \theta = \theta + N(0, \sigma_\theta) \)
  - \( N(\mu, \sigma) \) returns random noise from normal distribution with mean \( \mu \) and standard deviation \( \sigma \) (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

Table, chair legs

Closet

Printers

Recycling bins

LSRC Second Floor
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

![Distance from closest occupied square to endpoint of laser cast](image)

Laser Error Model Contd.

- Probability of error in measurement $k$ for sample $i$ (normal)
  \[ p_{ik} (x_i) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{x_i^2}{2\sigma^2}} \]
- $x_i$ is distance of laser endpoint to closest obstacle
- $\sigma$ is standard deviation in this measurement (estimated experimentally), usually a few cm.

Main Loop

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

Best Guess of Position

- Recover best guess of true position from weighted average of particle positions:
  \[ \bar{x} = \sum_{i} sample[i].x \times sample[i].p \]

How do we use this?