Bayes Filter Example

Joey Wilson, Maani Ghaffari

January 26, 2024

UNIVERSITY OF MICHIGAN

CURLY
Explore the Unknown

Overview

1. Review Bayes Filter
2. Example notebook

Bayes Filter

Bayes' Rule

- Derive from conditional probability
- Equation to update prior distribution given data

- $p(x,y) = p(x|y)p(y) = P(y|x)p(x)$
- $$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\sum_{x \in \mathcal{X}} p(y|x)p(x)}$$
  $$p(\text{hypothesis}|\text{data}) = \frac{p(\text{data}|\text{hypothesis})p(\text{hypothesis})}{p(\text{data})}$$
- $$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence (Marginal Likelihood)}}$$

Framework

- Given:
  - Stream of observations $z_{1:t}$ and action data $u_{1:t}$
  - Sensor/measurement model $p(z_t|x_t)$
  - Action/motion/transition model $p(x_t|x_{t-1},u_t)$
- Wanted:
  - The state X_t of dynamical system
  - The posterior of state is called belief $bel(x_t) = p(x_t|z_{1:t}, u_{1:t})$

Markov Assumption

*The Markov property states that "***the future is independent of the past if the present is known.***" A stochastic process that has this property is called a* **Markov process***.*

Assumption (Markov Assumption)

z_n *is* **independent** *of* $z_1, \ldots, z_{n-1}$ **if** *we* **know** x.

$$p(x|z_1,\ldots,z_n) = \frac{p(z_n|x)p(x|z_1,\ldots,z_{n-1})}{p(z_n|z_1,\ldots,z_{n-1})}$$

$$= \eta_n \; p(z_n|x)p(x|z_1,\ldots,z_{n-1}) = \eta_{1:n} \prod_{i=1}^{n} p(z_i|x)p(x)$$

where $\eta_{1:n} := \eta_1 \eta_2 \cdots \eta_n$.

Algorithm

Algorithm `Bayes-filter`

Require: Belief $bel(x_{t-1}) = p(x_{t-1}|z_{1:t-1}, u_{1:t-1})$, action u_t, measurement z_t;

1: **for** all state variables **do**

2: $\overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_t) bel(x_{t-1}) \mathrm{d}x_{t-1}$ // Predict using action/control input u_t

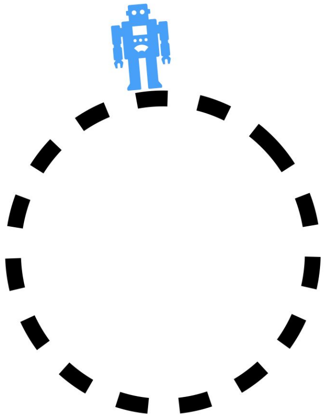
3: $bel(x_t) = \eta p(z_t|x_t) \overline{bel}(x_t)$ // Update using perceptual data z_t

4: **return** $bel(x_t)$

Example Problem

The Problem

- Consider a robot navigating a 1-D world
- The world is discretized, with twenty cells
    - In a circle, so the robot doesn't fall off the Earth!
- The robot can move one cell forward, or stay at the current location



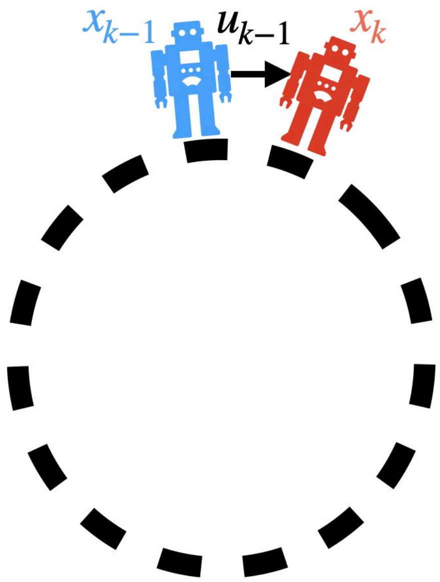
Motion Model

- Let k represent the time step, u the motion, and x the state
- If the robot moves forward ($u_{k-1} = 1$) then:

$$p(x_k | x_{k-1}, u_{k-1}) = \begin{cases} 1, \text{if, } x_k = x_{k-1} + 1, \\ 0, \text{otherwise} \end{cases}$$

- If the robot stays in place, then:

$$p(x_k | x_{k-1}, u_{k-1}) = \begin{cases} 1, \text{if, } x_k = x_{k-1}, \\ 0, \text{otherwise} \end{cases}$$



Motion Model Code

```
[2]  # Motion model
     # Motion model changes the belief based on action u
     def motion_model(xi, xj, u):
         if u == 1:  # move forward
             dx = xi - xj
             if dx == 1 or dx == -19:
                 p = 1
             else:
                 p = 0
         elif u == 0:  # stay
             dx = xi - xj
             if dx == 0:
                 p = 1
             else:
                 p = 0
         # print(p)
         # else:
         #     assert (u == 1 or u == 0), 'The action is not defined'
         return p
```

Measurement Model

- The robot has a sensor which can help with localization
- However, the sensor only has a strong reading (z = 1) at positions 4, 9, and 13 and is noisy

$$p(z_k = 1|x_k) = \begin{cases} 0.8, \text{if}, x_k = 4, 9, or 13, \\ 0.05, \text{otherwise.} \end{cases}$$

- The probability of no reading can be easily calculated

$$p(z_k = 0|x_k) = \begin{cases} 0.2, \text{if}, x_k = 4, 9, or 13, \\ 0.95, \text{otherwise.} \end{cases}$$

Update

- Use measurement model to update the beliefs of each state

$$bel(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t) \quad // \text{ Update using perceptual data } z_t$$

- Calculate the marginal likelihood for normalization

$$\eta = \frac{1}{\sum_{x_t} p(z_t \mid x_t)\overline{bel}(x_t)}$$

Implementation

- Loop over each possible state (since it is discrete)
- Calculate the likelihood using measurement model for each state
- Calculate un-normalized belief
- Normalize beliefs

```
eta = 0  # normalization constant
for i in range(len(X)):
    likelihood = measurement_model(X[i], likelihood_map, z=1)  # get measurement likelihood
    bel[:, i] = likelihood * bel_predicted[:, i]  # unnormalized Bayes update
    eta = eta + bel[:, i]
bel = bel / eta  # normalize belief
```

How It Works

Algorithm Review

Algorithm `Bayes-filter`

Require: Belief $bel(x_{t-1}) = p(x_{t-1}|z_{1:t-1}, u_{1:t-1})$, action u_t, measurement z_t;

1: **for** all state variables **do**

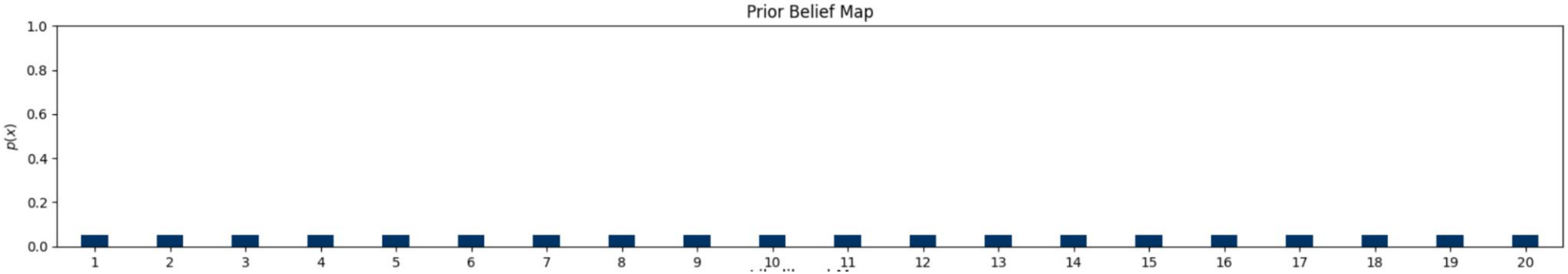
2: $\quad \overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_t) bel(x_{t-1}) \mathrm{d}x_{t-1}$ // Predict using action/control input u_t

3: $\quad bel(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t)$ // Update using perceptual data z_t

4: **return** $bel(x_t)$

First Step

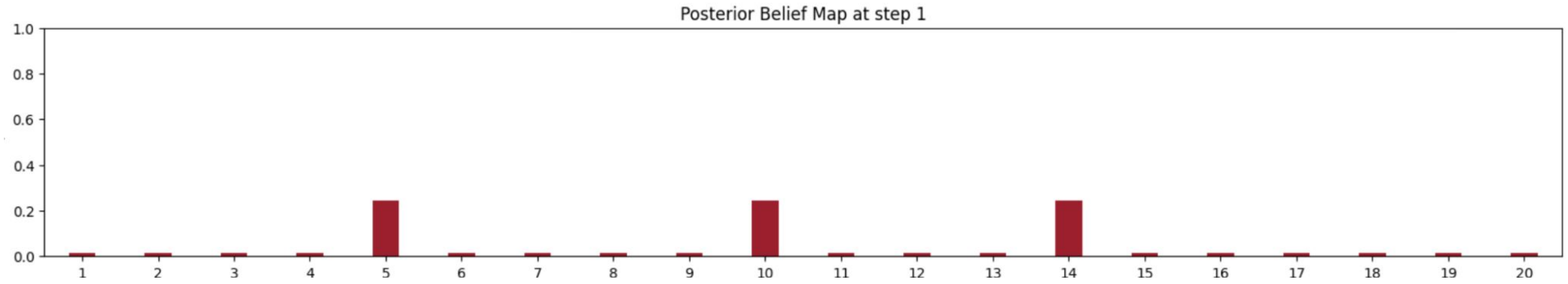
- Initialize state to uniform prior



- Suppose we detect a positive measurement and move forward. How will our posterior change?

Second Step

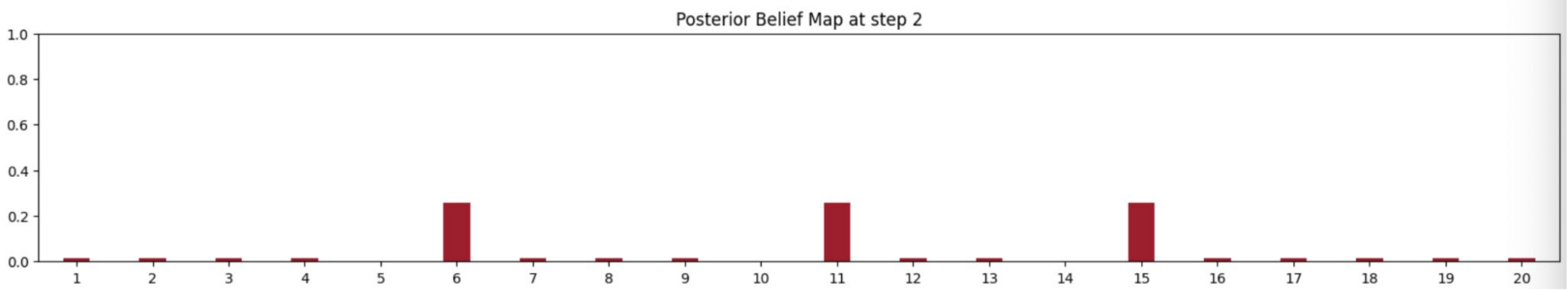
- More likely to be at locations 4, 9, or 13 before motion
- After motion, more likely locations 5, 10, 14



- Next, suppose we detect no measurement and move forward.

Third Step

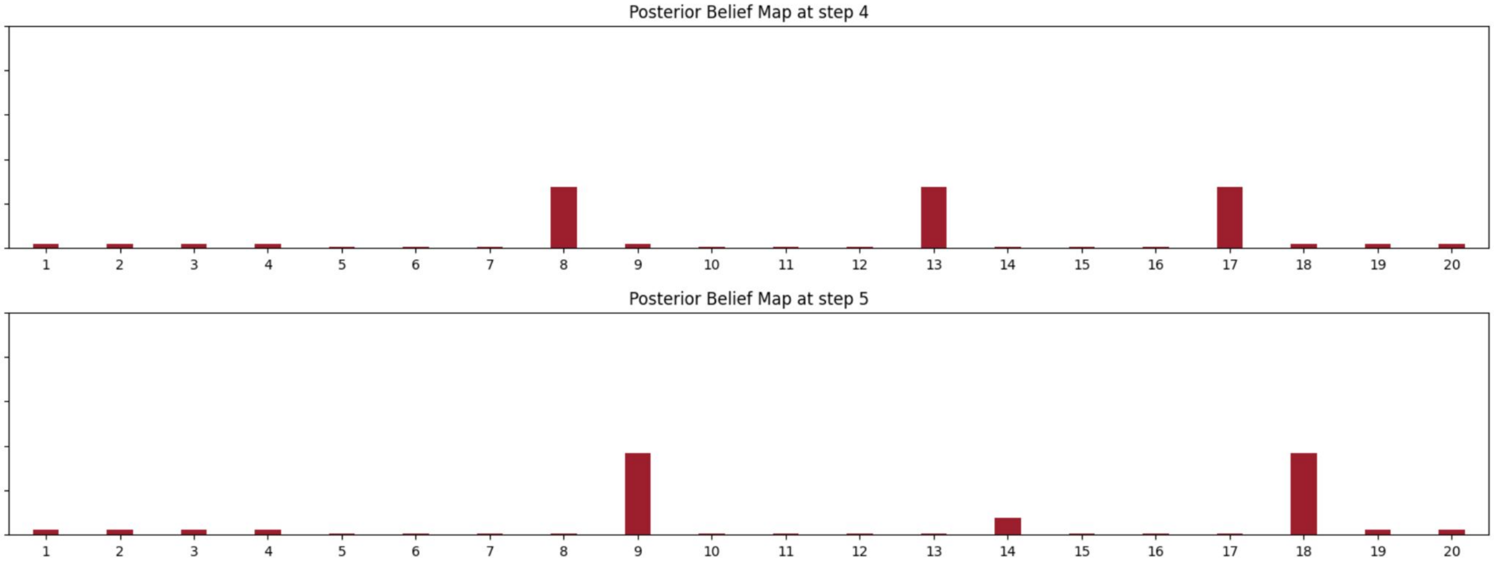
- Why do locations 5, 10, 14 have low probability?



- Next, we move a few more steps.

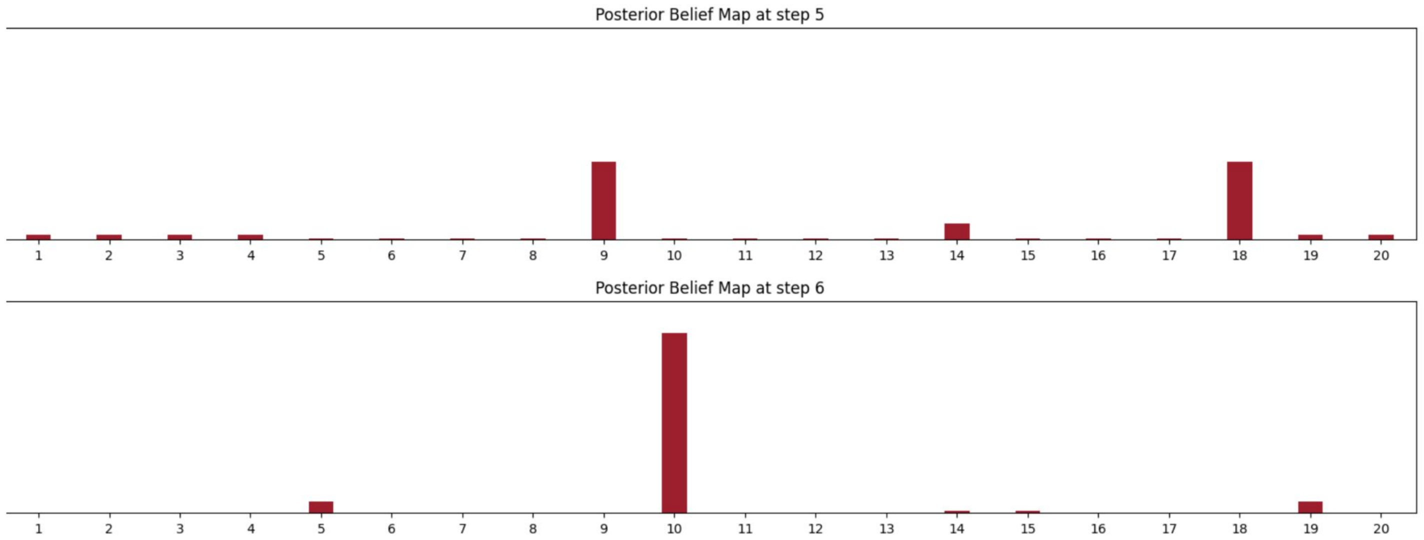
Fifth Step

- Negative sensor reading, which is unlikely at positions 4, 9, and 13



Sixth Step

- Positive sensor reading, which is likely at positions 4, 9, and 13



Demo