HHMs for Robot Localization on a Map Grid

Yifan (Brandon) Yang, Tanush Siotia, Timur Anvar [*]
Department of Computer Science
University of Virginia
Charlottesville, VA
{jqm9ba, ebh2cd, tka8edd}@virginia.edu

1 Introduction

In our probabilistic machine learning class, we have extensively studied Markovian Chains, understanding their potential and limitations. HMMs extend this concept by allowing us to deal with situations where the state of the system cannot be directly observed; instead, we must infer the state from visible outputs influenced by these hidden states.

To practically explore the utility of HMMs, we have chosen to implement robot localization on a map grid as our project. Localization is a fundamental challenge in robotics, involving the determination of a robot's position within its environment. Given that many aspects of a robot's environment and its interactions are not directly observable and are subject to various uncertainties—such as sensor noise and unexpected obstacles—an HMM provides an ideal framework for managing these uncertainties and inferring the robot's location based on probabilistic cues.

This project serves as a bridge between theoretical probability models and their real-world applications, highlighting the versatility and robustness of HMMs in solving practical problems in robotics and beyond.

2 Methods

2.1 Map

Suppose we have a map $\mathcal{M}$ defined as:

$$\mathcal{M}_{n\times m} = (\mathcal{R}, \mathcal{O}),$$

where n, m is the width and height of the map, $\mathcal{R}$ is the robot agent, and $\mathcal{O}$ is the set of obstacles. The robot agent $\mathcal{R}$ is initially placed at a random position (i, j) on the map $\mathcal{M}$. The set of obstacles $\mathcal{O}$ is randomly placed on the map $\mathcal{M}$, where each obstacle is placed at a random position (i, j) on the map.

We define the current state of the robot agent $\mathcal{R}$ at timestep t as $\mathcal{R}_{ij}^{(t)} = (i, j)$, where i and j are the row and column indices of the robot agent's position on the map $\mathcal{M}$, respectively. The state space $\mathcal{X}$ is defined as:

$$\mathcal{X} = \{(i, j) \mid i \in \{1, \dots, n\}, j \in \{1, \dots, m\}\},$$

2.2 Motion Model

We also define a motion model with predefined actions and their associated probabilities. Our action space consists of $\mathcal{U} = \{N, E, S, W\}$, where each action corresponds to moving the robot agent in the North, East, South, or West direction, respectively. We add a deterministic noise factor to the motion model to account for the uncertainty in the robot's movement, defined as:

$$\forall u \in \mathcal{U},\ u = \begin{cases} u & \text{with probability } 0.6 \\ \forall i \in \mathcal{U} \setminus u & \text{with probability } 0.1 \\ \emptyset & \text{with probability } 0.1 \end{cases},$$

[*]Work done as part of Probabilistic Machine Learning at UVA.

where $\emptyset$ denotes the robot agent staying in the same position. For example, suppose we have $u = N$, the robot agent moves North with a probability of 0.6, and it moves East, South, or West with a probability of 0.1 each, and stays in the same position with a probability of 0.1. Therefore, we define the robot agent $\mathcal{R}_{ij}^{(t)}$ taking action u at time t as:

$$\mathcal{R}_{ij}^{(t)}(u) = \mathcal{R}_{i'j'}^{(t+1)} = (i', j') = \begin{cases} (i-1, j) & \text{if } u = N, \\ (i, j+1) & \text{if } u = E, \\ (i+1, j) & \text{if } u = S, \\ (i, j-1) & \text{if } u = W, \\ (i, j) & \text{if } u = \emptyset \\ (i, j) & \text{otherwise,} \end{cases}$$

where $\mathcal{R}_{ij}^{(t)} = \mathcal{R}_{ij}^{(t+1)}$ if the robot agent $\mathcal{R}$ hits an obstacle or moves out of the map boundaries.

In the motion model, we define a transition matrix $T \in \mathbb{R}^{nm \times nm \times |\mathcal{U}|}$, where nm is the number of states in the map. For each action $u \in \mathcal{U}$, we define the transition matrix T_u as:

$$T_u = \begin{pmatrix} p\left[\mathcal{R}_{1,1}^{(t+1)} \mid \mathcal{R}_{1,1}^{(t)}, u\right], & p\left[\mathcal{R}_{1,2}^{(t+1)} \mid \mathcal{R}_{1,1}^{(t)}, u\right], & \ldots, & p\left[\mathcal{R}_{n,m}^{(t+1)} \mid \mathcal{R}_{1,1}^{(t)}, u\right] \\ p\left[\mathcal{R}_{1,1}^{(t+1)} \mid \mathcal{R}_{1,2}^{(t)}, u\right], & p\left[\mathcal{R}_{1,2}^{(t+1)} \mid \mathcal{R}_{1,2}^{(t)}, u\right], & \ldots, & p\left[\mathcal{R}_{n,m}^{(t+1)} \mid \mathcal{R}_{1,2}^{(t)}, u\right] \\ \vdots & \vdots & \ddots & \vdots \\ p\left[\mathcal{R}_{1,1}^{(t+1)} \mid \mathcal{R}_{n,m}^{(t)}, u\right], & p\left[\mathcal{R}_{1,2}^{(t+1)} \mid \mathcal{R}_{n,m}^{(t)}, u\right], & \ldots, & p\left[\mathcal{R}_{n,m}^{(t+1)} \mid \mathcal{R}_{n,m}^{(t)}, u\right] \end{pmatrix}$$

and

$$T = (T_N, \quad T_E, \quad T_S, \quad T_W)\,.$$

In the transition matrix T, $p\left[\mathcal{R}_{i',j'}^{(t+1)} \mid \mathcal{R}_{i,j}^{(t)}, u\right]$ is the probability of the robot agent moving to position (i', j') given the current position (i, j) and action u. For each $u \in \mathcal{U}$, each row of T_u should sum to 1 if $\mathcal{R}_{ij}^{(t)}$ is a valid state, otherwise, the row sums to 0.

2.3 Sensor Model

Our robot agent $\mathcal{R}$ has a sensor that can detect obstacles in the North, East, South, and West directions. We initiallize a sensor model that creates a set of *sensor observation probability* matrices based on the map's layout.

We define each possible sensor reading z as a boolean tuple:

$$z = (z_N, z_E, z_S, z_W)\,,$$

where $\forall u \in \mathcal{U}$:

$$z_u = \begin{cases} \text{True} & \text{if there is an obstacle in the direction } u, \\ \text{False} & \text{otherwise.} \end{cases}$$

The sensor has an error rate of ϵ. Therefore, the probability of the sensor detecting an obstacle in the direction u given the true state of the map is:

$$p(z_u | \text{ obstacle in the direction } u) = \begin{cases} 1-\epsilon & \text{if } z_u = \text{True} \\ \epsilon & \text{otherwise.} \end{cases}$$

$$p(z_u | \text{ No obstacle in the direction } u) = \begin{cases} \epsilon & \text{if } z_u = \text{True} \\ 1-\epsilon & \text{otherwise.} \end{cases}$$

Thus, the probability of getting all sensor readings correctly is $(1-\epsilon)^4$. And the probability of getting all sensor readings incorrectly is ϵ^4.

Since each z_u is a binary tuple of four elements, we can define the observation space $\mathcal{Z}$ as follows:

$$\mathcal{Z} = \{a, b, c, d \mid a, b, c, d \in \{\text{True}, \text{False}\}\},$$

where $|\mathcal{Z}| = 2^4 = 16$.

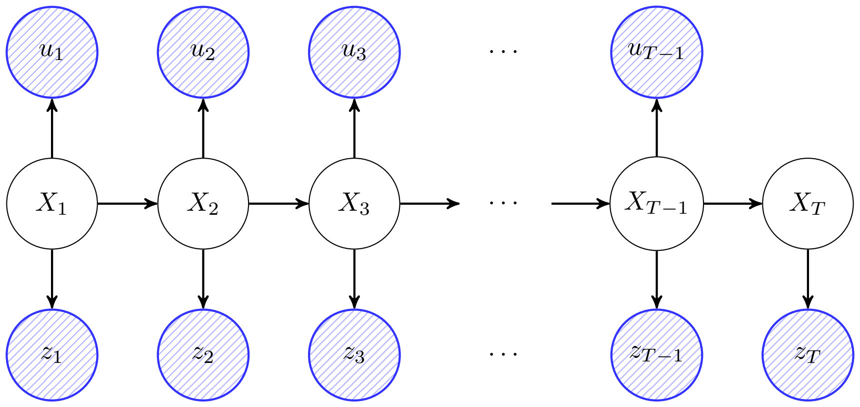
For each observation $z \in \mathcal{Z}$, we define the observation probability matrix M over the state space:

$$M(z^{(t)}) = p(z^{(t)}|\mathcal{R}) = \begin{pmatrix} p(z^{(t)}|\mathcal{R}^{(t)}_{1,1}), & p(z^{(t)}|\mathcal{R}^{(t)}_{1,2}), & \dots, & p(z^{(t)}|\mathcal{R}^{(t)}_{n,m}) \\ p(z^{(t)}|\mathcal{R}^{(t)}_{2,1}), & p(z^{(t)}|\mathcal{R}^{(t)}_{2,2}), & \dots, & p(z^{(t)}|\mathcal{R}^{(t)}_{2,m}) \\ \vdots & \vdots & \ddots & \vdots \\ p(z^{(t)}|\mathcal{R}^{(t)}_{n,1}), & p(z^{(t)}|\mathcal{R}^{(t)}_{n,2}), & \dots, & p(z^{(t)}|\mathcal{R}^{(t)}_{n,m}) \end{pmatrix},$$

2.4 Hidden Markov Model (HMM)

Given the motion model and sensor model, we can define the HMM as a tuple $\mathcal{H} = (\mathcal{X}, \mathcal{U}, T, \mathcal{Z}, M)$, where:

$$\mathcal{H} = \begin{cases} \mathcal{X} & \text{State space} \\ \mathcal{U} & \text{Action space} \\ T & \text{Transition matrices} \\ \mathcal{Z} & \text{Observation space} \\ M & \text{Observation probability matrices} \end{cases}$$



At each timestep t, the robot agent $\mathcal{R}$ is in a particular state $X_t \in \mathcal{X}$, takes an action $u_t \in \mathcal{U}$, and receives an observation $z_t \in \mathcal{Z}$. The robot agent's goal is to infer the probability distribution over the state space $\mathcal{X}$ given the sequence of observations $z_{1:T}$ and actions $u_{1:T-1}$. Note that at timestep T, the robot agent only receives an observation z_T and does not take any action since there is no movement.

2.4.1 Forward Algorithm

Given the HMM $\mathcal{H}$, we can use the forward algorithm to compute the probability of the robot agent being in a particular state X_j at time t given the sequence of observations $z_{1:t}$ up to time t, and sequence of actions $u_{1:t-1}$ up to time $t-1$.

At timestep t, the forward algorithm is defined as:

$$\alpha_t(X_j) = p(X_j, z_{1:t}|\mathcal{H}, u_{t-1}) = \sum_{X_i \in \mathcal{X}} p(X_j|X_i, u_{t-1}) \cdot p(M(z_t)|X_j) \cdot \alpha_{t-1}(X_i),$$

where u_{t-1} is the action taken at time $t-1$. The initial forward probability $\alpha_0(X_j)$ is defined as:

$$\alpha_0(X_j) = \pi_0 \cdot p(M(z_1)|X_j),$$

where π_0 is the initial state distribution over the state space $\mathcal{X}$. We assume a uniform distribution over the initial states.

Thus, the probability of the robot agent being in a particular state X_j at time t given the sequence of observations $z_{1:t}$ and sequence of actions $u_{1:t-1}$ is:

$$p(X_j|z_{1:t}, u_{1:t-1}, \mathcal{H}) = \alpha_t(X_j).$$

The likelihood of the observations $z_{1:t}$ given the HMM $\mathcal{H}$ is:

$$p(z_{1:t}|\mathcal{H}) = \sum_{X_i \in \mathcal{X}} \alpha_t(X_i).$$

2.4.2 Backward Algorithm

Given the HMM $\mathcal{H}$, we can use the backward algorithm to compute the probability of the robot agent being in a particular state X_i at time t given the sequence of observations $z_{t+1:T}$ from time $t+1$ to T and sequence of actions $u_{t:T-1}$ from time t to $T-1$.

At timestep t, the backward algorithm is defined as:

$$\beta_t(X_i) = p(z_{t+1:T}|X_i, \mathcal{H}, u_t) = \sum_{X_j \in \mathcal{X}} p(X_j|X_i, u_t) \cdot p(M(z_{t+1})|X_j) \cdot \beta_{t+1}(X_j),$$

where u_t is the action taken at time t. The initial backward probability $\beta_T(X_i)$ is defined as:

$$\beta_T(X_i) = 1.$$

Therefore, we can also compute the probability of the robot agent being in a particular state X_i at time t given the sequence of observations $z_{t+1:T}$ and sequence of actions $u_{t:T-1}$ as:

$$p(X_i|z_{t+1:T}, u_{t:T-1}, \mathcal{H}) = \beta_t(X_i).$$

Therefore, we can also compute the likelihood of the observations $z_{1:T}$ given $\mathcal{H}$ using $\beta_t(X_i)$:

$$p(z_{1:T}|\mathcal{H}) = \sum_{X_i \in \mathcal{X}} \pi_0 \cdot p(M(z_1)|X_i) \cdot \beta_1(X_i).$$

2.4.3 Smoothing Algorithm

Given the HMM $\mathcal{H}$, we can use the smoothing algorithm to compute the probability of the robot agent being in a particular state X_i at time t given the sequence of observations $z_{1:T}$ and sequence of actions $u_{1:T-1}$.

At timestep t, the smoothing algorithm is defined as:

$$\gamma_t(X_i) = p(X_i|z_{1:T}, u_{1:T-1}, \mathcal{H}) = \frac{\alpha_t(X_i) \cdot \beta_t(X_i)}{p(z_{1:T}|\mathcal{H})}.$$

3 Experimental Results

3.1 Map

We initialize a map $\mathcal{M}_{10\times10}$ shown in fig. 1. We also randomly assign the initial starting position of the robot, as well as the obstacle layout of the map.

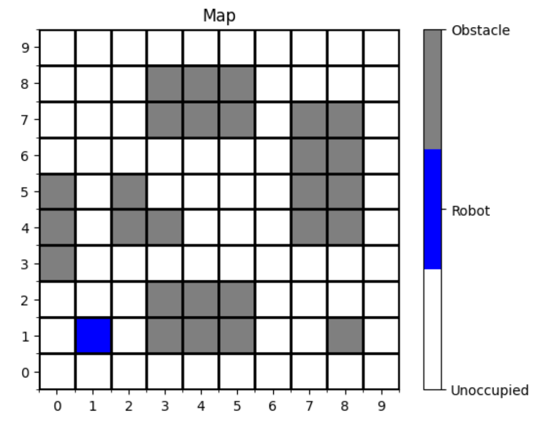


Figure 1: Initial Map $\mathcal{M} = (\mathcal{R}, \mathcal{O})$.

3.2 Motion Model

We then compute the motion model based on the specification in section section 2.2. We demonstrate the that we are able to create the correct transition matrix T_u for each action $u \in \mathcal{U}$ by plotting the sum of rows of the transition matrix in fig. 2. We can see that all four graphs, corresponding to each action in $\mathcal{U}$, exactly emulate the initial setting of the map $\mathcal{M}$.

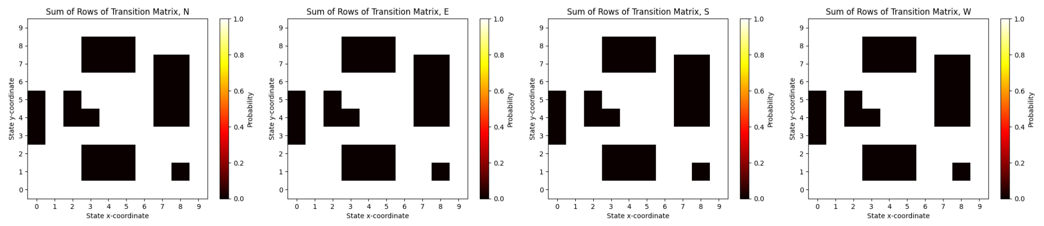


Figure 2: Validation of T using total probability property over the rows.

3.3 Sensor Model

3.4 HMM