Hidden Markov Models

Outline

I. Simplified matrix algorithms

II. HMM application to robot localization

* Figures/images are from the textbook site or drawn by the instructor unless the source is cited specifically.

I. Hidden Markov Models

- The state is described by a single, discrete random variable X_t.
- Its values are the possible states of the world.
- There can be many evidence variables, both discrete and continuous.

I. Hidden Markov Models

- The state is described by a single, discrete random variable X_t.
- Its values are the possible states of the world.
- There can be many evidence variables, both discrete and continuous.

Denote the values of X_t by $c_1, \ldots, c_n$.

I. Hidden Markov Models

- The state is described by a single, discrete random variable X_t.
- Its values are the possible states of the world.
- There can be many evidence variables, both discrete and continuous.

Denote the values of X_t by $c_1, \ldots, c_n$.

Transition model:

$$\boldsymbol{T} = \begin{pmatrix} T_{11} & \cdots & T_{1n} \\ \vdots & \ddots & \vdots \\ T_{n1} & \cdots & T_{nn} \end{pmatrix}$$

I. Hidden Markov Models

- The state is described by a single, discrete random variable X_t.
- Its values are the possible states of the world.
- There can be many evidence variables, both discrete and continuous.

Denote the values of X_t by $c_1, \ldots, c_n$.

Transition model:

$$\boldsymbol{T} = \begin{pmatrix} T_{11} & \cdots & T_{1n} \\ \vdots & \ddots & \vdots \\ T_{n1} & \cdots & T_{nn} \end{pmatrix}$$

where $T_{ij} = P(X_t = c_j \mid X_{t-1} = c_i)$

probability of transition from state i to state j

I. Hidden Markov Models

- The state is described by a single, discrete random variable X_t.
- Its values are the possible states of the world.
- There can be many evidence variables, both discrete and continuous.

Denote the values of X_t by $c_1, \ldots, c_n$.

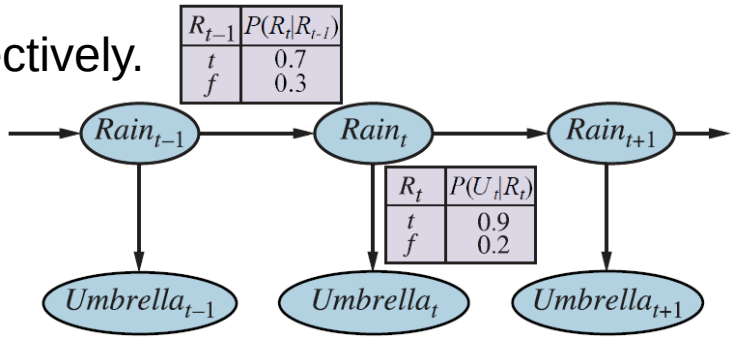
Transition model:

$$\boldsymbol{T} = \begin{pmatrix} T_{11} & \cdots & T_{1n} \\ \vdots & \ddots & \vdots \\ T_{n1} & \cdots & T_{nn} \end{pmatrix}$$

where $T_{ij} = P(X_t = c_j \mid X_{t-1} = c_i)$

probability of transition from state i to state j

For the umbrella world, we denote the states *Rain* = *true* and *Rain* = *false* by c_1 and c_2, respectively.



I. Hidden Markov Models

- The state is described by a single, discrete random variable X_t.
- Its values are the possible states of the world.
- There can be many evidence variables, both discrete and continuous.

Denote the values of X_t by $c_1, \ldots, c_n$.

Transition model:

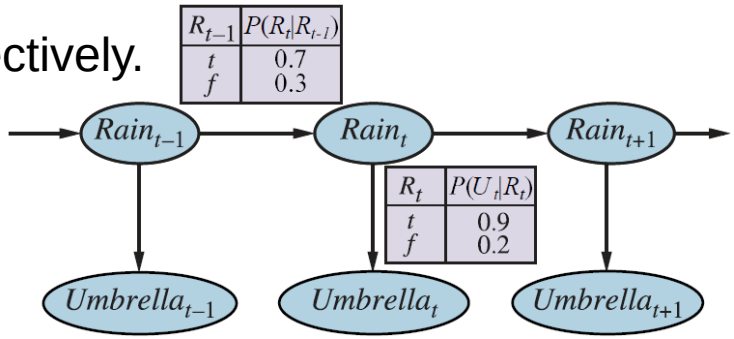
$$\boldsymbol{T} = \begin{pmatrix} T_{11} & \cdots & T_{1n} \\ \vdots & \ddots & \vdots \\ T_{n1} & \cdots & T_{nn} \end{pmatrix}$$

where $T_{ij} = P(X_t = c_j \mid X_{t-1} = c_i)$

probability of transition from state i to state j

For the umbrella world, we denote the states *Rain* = *true* and *Rain* = *false* by c_1 and c_2, respectively.

$$\boldsymbol{T} = P(X_t \mid X_{t-1}) = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$$



Observation Matrix

The value e_t of the evidence variable E_t is known at time t.

For $i = 1, \ldots, n$, the likelihood of the state value c_i causing e_t to appear is

$$P(e_t \mid X_t = c_i)$$

Observation Matrix

The value e_t of the evidence variable E_t is known at time t.

For $i = 1, \ldots, n$, the likelihood of the state value c_i causing e_t to appear is

$$P(e_t \mid X_t = c_i)$$

Observation matrix $\boldsymbol{O}_t = \underbrace{\begin{pmatrix} P(e_t \mid X_t = c_1) & & 0 \\ & \ddots & \\ 0 & & P(e_t \mid X_t = c_n) \end{pmatrix}}_{n \times n}$

Observation Matrix

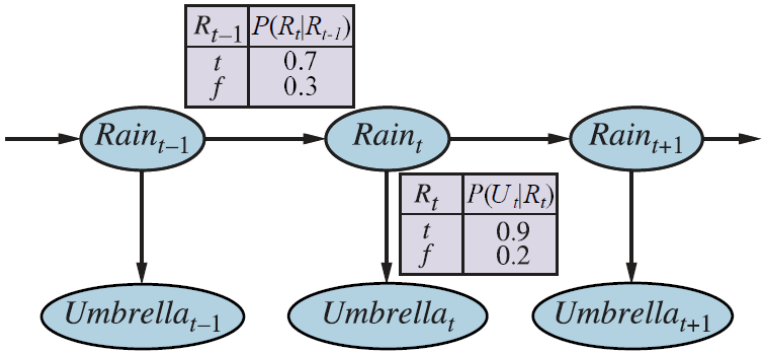
The value e_t of the evidence variable E_t is known at time t.

For $i = 1, \ldots, n$, the likelihood of the state value c_i causing e_t to appear is

$$P(e_t \mid X_t = c_i)$$

Observation matrix $\boldsymbol{O}_t = \underbrace{\begin{pmatrix} P(e_t \mid X_t = c_1) & & 0 \\ & \ddots & \\ 0 & & P(e_t \mid X_t = c_n) \end{pmatrix}}_{n \times n}$

Suppose that $U_1 = \textit{true}$ and $U_3 = \textit{false}$.



Observation Matrix

The value e_t of the evidence variable E_t is known at time t.

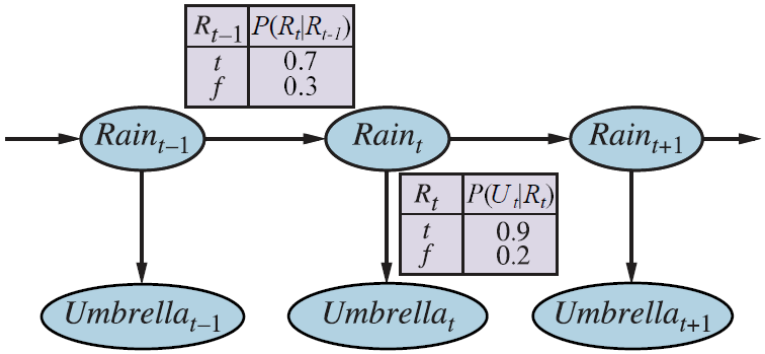
For $i = 1, \ldots, n$, the likelihood of the state value c_i causing e_t to appear is

$$P(e_t \mid X_t = c_i)$$

Observation matrix $\boldsymbol{O}_t = \underbrace{\begin{pmatrix} P(e_t \mid X_t = c_1) & & 0 \\ & \ddots & \\ 0 & & P(e_t \mid X_t = c_n) \end{pmatrix}}_{n \times n}$

Suppose that $U_1 = \textit{true}$ and $U_3 = \textit{false}$.

$$O_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix} \qquad O_3 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.8 \end{pmatrix}$$



Matrix Formulations of Filtering and Smoothing

Forward message used in filtering: $\boldsymbol{f}_{1:t} \equiv \boldsymbol{P}(\boldsymbol{X}_t \mid \boldsymbol{e}_{1:t}) = \begin{pmatrix} P(c_1 \mid \boldsymbol{e}_{1:t}) \\ \vdots \\ P(c_n \mid \boldsymbol{e}_{1:t}) \end{pmatrix}$

Matrix Formulations of Filtering and Smoothing

Forward message used in filtering: $\boldsymbol{f}_{1:t} \equiv \boldsymbol{P}(\boldsymbol{X}_t \mid \boldsymbol{e}_{1:t}) = \begin{pmatrix} P(c_1 \mid \boldsymbol{e}_{1:t}) \\ \vdots \\ P(c_n \mid \boldsymbol{e}_{1:t}) \end{pmatrix}$

$$\boldsymbol{f}_{1:t+1} \equiv \boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{e}_{1:t+1}) = \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1} \mid \boldsymbol{X}_{t+1}) \sum_{\boldsymbol{x}_t} \boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{x}_t) P(\boldsymbol{x}_t \mid \boldsymbol{e}_{1:t})$$

Matrix Formulations of Filtering and Smoothing

Forward message used in filtering: $\boldsymbol{f}_{1:t} \equiv \boldsymbol{P}(\boldsymbol{X}_t \mid \boldsymbol{e}_{1:t}) = \begin{pmatrix} P(c_1 \mid \boldsymbol{e}_{1:t}) \\ \vdots \\ P(c_n \mid \boldsymbol{e}_{1:t}) \end{pmatrix}$

$$\boldsymbol{f}_{1:t+1} \equiv \boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{e}_{1:t+1}) = \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1} \mid \boldsymbol{X}_{t+1}) \sum_{\boldsymbol{x}_t} \boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{x}_t) P(\boldsymbol{x}_t \mid \boldsymbol{e}_{1:t})$$

Rewrite pointwise products as matrix products.

$$// \boldsymbol{T} = \begin{pmatrix} P(X_{t+1} = c_1 \mid X_t = c_1) & \cdots & P(X_{t+1} = c_n \mid X_t = c_1) \\ \vdots & \ddots & \vdots \\ P(X_{t+1} = c_1 \mid X_t = c_n) & \cdots & P(X_{t+1} = c_n \mid X_t = c_n) \end{pmatrix}$$

$$// \boldsymbol{O}_{t+1} = \begin{pmatrix} P(e_{t+1} \mid X_{t+1} = c_1) & & \\ & \ddots & \\ & & P(e_{t+1} \mid X_{t+1} = c_n) \end{pmatrix}$$

Matrix transpose

$$\boldsymbol{f}_{1:t+1} = \alpha \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{f}_{1:t} \quad \text{(filtering)}$$

Matrix Formulations of Filtering and Smoothing

Forward message used in filtering: $\boldsymbol{f}_{1:t} \equiv \boldsymbol{P}(\boldsymbol{X}_t \mid \boldsymbol{e}_{1:t}) = \begin{pmatrix} P(c_1 \mid \boldsymbol{e}_{1:t}) \\ \vdots \\ P(c_n \mid \boldsymbol{e}_{1:t}) \end{pmatrix}$

$$\boldsymbol{f}_{1:t+1} \equiv \boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{e}_{1:t+1}) = \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1} \mid \boldsymbol{X}_{t+1}) \sum_{\boldsymbol{x}_t} \boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{x}_t) P(\boldsymbol{x}_t \mid \boldsymbol{e}_{1:t})$$

Rewrite pointwise products as matrix products.

// $\boldsymbol{T} = \begin{pmatrix} P(X_{t+1} = c_1 \mid X_t = c_1) & \cdots & P(X_{t+1} = c_n \mid X_t = c_1) \\ \vdots & \ddots & \vdots \\ P(X_{t+1} = c_1 \mid X_t = c_n) & \cdots & P(X_{t+1} = c_n \mid X_t = c_n) \end{pmatrix}$

// $\boldsymbol{O}_{t+1} = \begin{pmatrix} P(e_{t+1} \mid X_{t+1} = c_1) & & \\ & \ddots & \\ & & P(e_{t+1} \mid X_{t+1} = c_n) \end{pmatrix}$

Matrix transpose

$$\boldsymbol{f}_{1:t+1} = \alpha \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{f}_{1:t}$$

matrix product

(filtering)

Matrix Formulations of Filtering and Smoothing

Forward message used in filtering: $\boldsymbol{f}_{1:t} \equiv \boldsymbol{P}(\boldsymbol{X}_t \mid \boldsymbol{e}_{1:t}) = \begin{pmatrix} P(c_1 \mid \boldsymbol{e}_{1:t}) \\ \vdots \\ P(c_n \mid \boldsymbol{e}_{1:t}) \end{pmatrix}$

$$\boldsymbol{f}_{1:t+1} \equiv \boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{e}_{1:t+1}) = \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1} \mid \boldsymbol{X}_{t+1}) \sum_{\boldsymbol{x}_t} \boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{x}_t) P(\boldsymbol{x}_t \mid \boldsymbol{e}_{1:t})$$

Rewrite pointwise products as matrix products.

$$// \boldsymbol{T} = \begin{pmatrix} P(X_{t+1} = c_1 \mid X_t = c_1) & \cdots & P(X_{t+1} = c_n \mid X_t = c_1) \\ \vdots & \ddots & \vdots \\ P(X_{t+1} = c_1 \mid X_t = c_n) & \cdots & P(X_{t+1} = c_n \mid X_t = c_n) \end{pmatrix}$$

$$// \boldsymbol{O}_{t+1} = \begin{pmatrix} P(e_{t+1} \mid X_{t+1} = c_1) & & \\ & \ddots & \\ & & P(e_{t+1} \mid X_{t+1} = c_n) \end{pmatrix}$$

Matrix transpose

$$\boldsymbol{f}_{1:t+1} = \alpha \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{f}_{1:t}$$

matrix product

(filtering)

Backward message used in smoothing: $\boldsymbol{b}_{k+1:t} \equiv \boldsymbol{P}(\boldsymbol{e}_{k+1:t} \mid \boldsymbol{X}_k)$

Matrix Formulations of Filtering and Smoothing

Forward message used in filtering: $\boldsymbol{f}_{1:t} \equiv \boldsymbol{P}(\boldsymbol{X}_t \mid \boldsymbol{e}_{1:t}) = \begin{pmatrix} P(c_1 \mid \boldsymbol{e}_{1:t}) \\ \vdots \\ P(c_n \mid \boldsymbol{e}_{1:t}) \end{pmatrix}$

$$\boldsymbol{f}_{1:t+1} \equiv \boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{e}_{1:t+1}) = \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1} \mid \boldsymbol{X}_{t+1}) \sum_{\boldsymbol{x}_t} \boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{x}_t) P(\boldsymbol{x}_t \mid \boldsymbol{e}_{1:t})$$

Rewrite pointwise products as matrix products.

$$// \boldsymbol{T} = \begin{pmatrix} P(X_{t+1} = c_1 \mid X_t = c_1) & \cdots & P(X_{t+1} = c_n \mid X_t = c_1) \\ \vdots & \ddots & \vdots \\ P(X_{t+1} = c_1 \mid X_t = c_n) & \cdots & P(X_{t+1} = c_n \mid X_t = c_n) \end{pmatrix}$$

$$// \boldsymbol{O}_{t+1} = \begin{pmatrix} P(e_{t+1} \mid X_{t+1} = c_1) & & \\ & \ddots & \\ & & P(e_{t+1} \mid X_{t+1} = c_n) \end{pmatrix}$$

Matrix transpose

$$\boldsymbol{f}_{1:t+1} = \alpha \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{f}_{1:t}$$

matrix product

(filtering)

Backward message used in smoothing: $\boldsymbol{b}_{k+1:t} \equiv \boldsymbol{P}(\boldsymbol{e}_{k+1:t} \mid \boldsymbol{X}_k)$

$$\boldsymbol{P}(\boldsymbol{e}_{k+1:t} \mid \boldsymbol{X}_k) = \sum_{\boldsymbol{x}_{k+1}} P(\boldsymbol{e}_{k+1} \mid \boldsymbol{x}_{k+1}) P(\boldsymbol{e}_{k+2:t} \mid \boldsymbol{x}_{k+1}) \boldsymbol{P}(\boldsymbol{x}_{k+1} \mid \boldsymbol{X}_k)$$

Matrix Formulations of Filtering and Smoothing

Forward message used in filtering: $\boldsymbol{f}_{1:t} \equiv \boldsymbol{P}(\boldsymbol{X}_t \mid \boldsymbol{e}_{1:t}) = \begin{pmatrix} P(c_1 \mid \boldsymbol{e}_{1:t}) \\ \vdots \\ P(c_n \mid \boldsymbol{e}_{1:t}) \end{pmatrix}$

$$\boldsymbol{f}_{1:t+1} \equiv \boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{e}_{1:t+1}) = \alpha \boldsymbol{P}(\boldsymbol{e}_{t+1} \mid \boldsymbol{X}_{t+1}) \sum_{\boldsymbol{x}_t} \boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{x}_t) P(\boldsymbol{x}_t \mid \boldsymbol{e}_{1:t})$$

Rewrite pointwise products as matrix products.

$$// \boldsymbol{T} = \begin{pmatrix} P(X_{t+1} = c_1 \mid X_t = c_1) & \cdots & P(X_{t+1} = c_n \mid X_t = c_1) \\ \vdots & \ddots & \vdots \\ P(X_{t+1} = c_1 \mid X_t = c_n) & \cdots & P(X_{t+1} = c_n \mid X_t = c_n) \end{pmatrix}$$

$$// \boldsymbol{O}_{t+1} = \begin{pmatrix} P(e_{t+1} \mid X_{t+1} = c_1) & & \\ & \ddots & \\ & & P(e_{t+1} \mid X_{t+1} = c_n) \end{pmatrix}$$

Matrix transpose

$$\boldsymbol{f}_{1:t+1} = \alpha \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{f}_{1:t}$$

matrix product

(filtering)

Backward message used in smoothing: $\boldsymbol{b}_{k+1:t} \equiv \boldsymbol{P}(\boldsymbol{e}_{k+1:t} \mid \boldsymbol{X}_k)$

$$\boldsymbol{P}(\boldsymbol{e}_{k+1:t} \mid \boldsymbol{X}_k) = \sum_{\boldsymbol{x}_{k+1}} P(\boldsymbol{e}_{k+1} \mid \boldsymbol{x}_{k+1}) P(\boldsymbol{e}_{k+2:t} \mid \boldsymbol{x}_{k+1}) \boldsymbol{P}(\boldsymbol{x}_{k+1} \mid \boldsymbol{X}_k)$$

$$\boldsymbol{b}_{k+1:t} = \boldsymbol{T} \boldsymbol{O}_{k+1} \boldsymbol{b}_{k+2:t}$$

(smoothing)

Complexities and Improvements

$$\boldsymbol{f}_{1:t+1} = \alpha \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{f}_{1:t}$$

$$\boldsymbol{b}_{k+1:t} = \boldsymbol{T}\boldsymbol{O}_{k+1}\ \boldsymbol{b}_{k+2:t}$$

Complexities and Improvements

$$\boldsymbol{f}_{1:t+1} = \alpha \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{f}_{1:t}$$

$$\boldsymbol{b}_{k+1:t} = \boldsymbol{T} \boldsymbol{O}_{k+1} \boldsymbol{b}_{k+2:t}$$

Time complexity: $O(n^2 t)$ // t steps, each requiring two rounds of $O(n^2)$ time
// multiplication of an $n \times n$ matrix by an n-vector.

Complexities and Improvements

$$\boldsymbol{f}_{1:t+1} = \alpha \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{f}_{1:t}$$

$$\boldsymbol{b}_{k+1:t} = \boldsymbol{T} \boldsymbol{O}_{k+1} \boldsymbol{b}_{k+2:t}$$

Time complexity: $O(n^2 t)$ // t steps, each requiring two rounds of $O(n^2)$ time
// multiplication of an $n \times n$ matrix by an n-vector.

Space complexity: $O(nt)$ // $t > n$. The forward pass stores t vectors of size n.

Complexities and Improvements

$$\boldsymbol{f}_{1:t+1} = \alpha \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{f}_{1:t}$$

$$\boldsymbol{b}_{k+1:t} = \boldsymbol{T} \boldsymbol{O}_{k+1} \boldsymbol{b}_{k+2:t}$$

Time complexity: $O(n^2 t)$ // t steps, each requiring two rounds of $O(n^2)$ time
// multiplication of an $n \times n$ matrix by an n-vector.

Space complexity: $O(nt)$ // $t > n$. The forward pass stores t vectors of size n.

Improvements:

- Allows smoothing to be carried out in constant space, independent of the lengths of the sequence.

Complexities and Improvements

$$\boldsymbol{f}_{1:t+1} = \alpha \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{f}_{1:t}$$

$$\boldsymbol{b}_{k+1:t} = \boldsymbol{T} \boldsymbol{O}_{k+1} \boldsymbol{b}_{k+2:t}$$

Time complexity: $O(n^2 t)$ // t steps, each requiring two rounds of $O(n^2)$ time
// multiplication of an $n \times n$ matrix by an n-vector.

Space complexity: $O(nt)$ // $t > n$. The forward pass stores t vectors of size n.

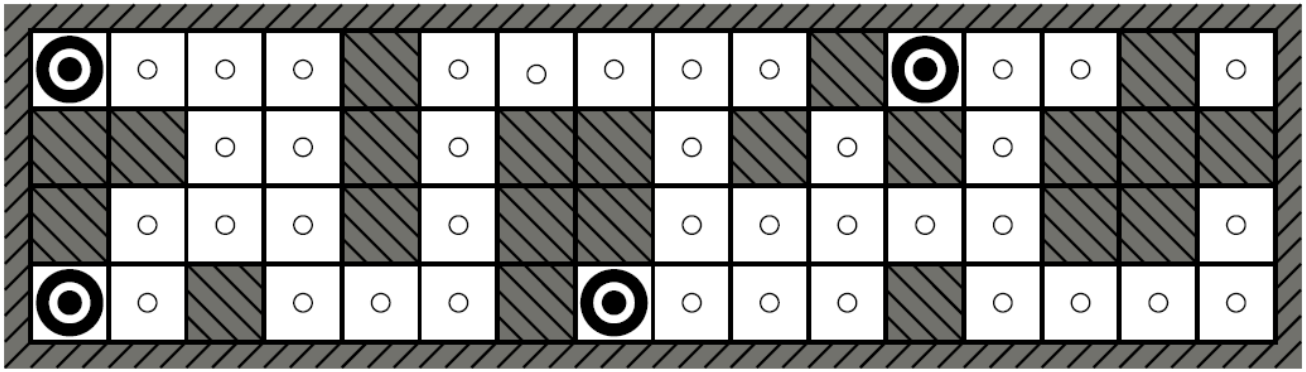
Improvements:

- Allows smoothing to be carried out in constant space, independent of the lengths of the sequence.
- Leads to an algorithm whose time complexity is independent of the length d of the lag.

  Smoothing at time $t - d$, where the current time is t.

II. Revisiting the Localization Task

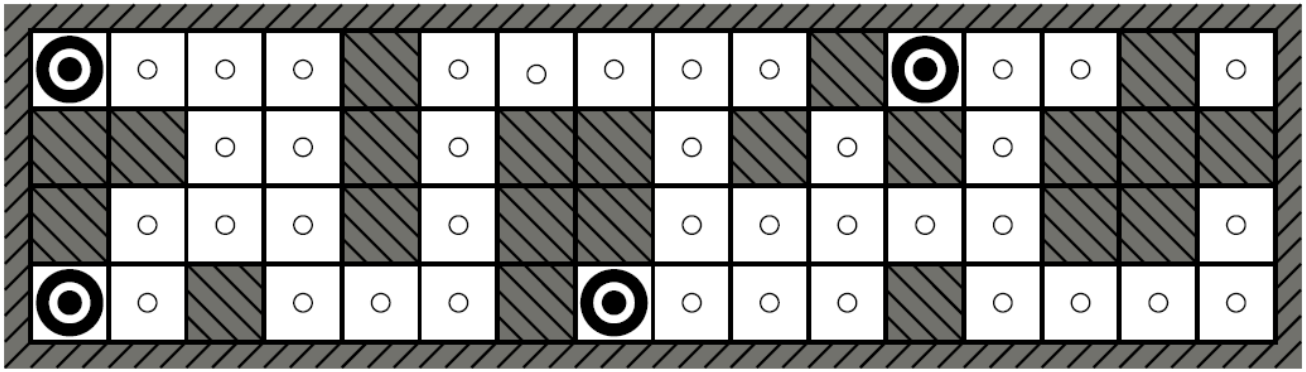
The robot had a single action Move and a perfect sensor to report whether obstacles are immediately to the north, east, south, and west.



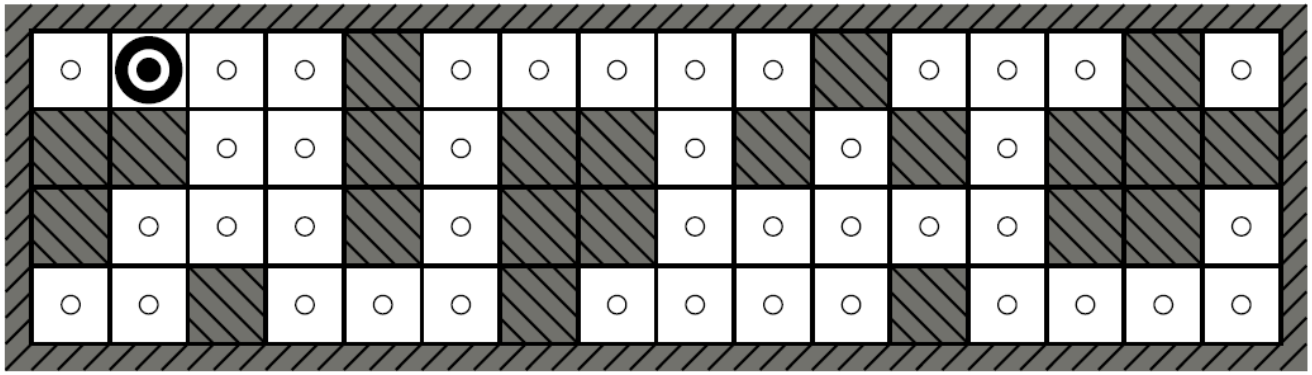
Possible locations of the robot after $E_1 = 1011$ (NESW) (obstacles in the north, south, and west, but not east).

II. Revisiting the Localization Task

The robot had a single action Move and a perfect sensor to report whether obstacles are immediately to the north, east, south, and west.



Possible locations of the robot after $E_1 = 1011$ (obstacles in the north, south, and west, but not east).
NESW



Possible locations after $E_1 = 1011$, $E_2 = 1010$.

HMM Formulation

We now make the problem more realistic:

- Allow noise in sensing whether or not obstacles are immediately to the north, east, south, and west.
- The robot is equally likely to move to any adjacent square.

HMM Formulation

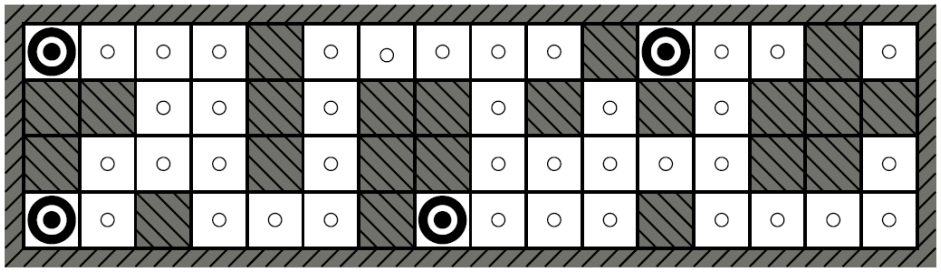
We now make the problem more realistic:

- Allow noise in sensing whether or not obstacles are immediately to the north, east, south, and west.
- The robot is equally likely to move to any adjacent square.

- State X_t: robot location
- $\{1, \ldots, n\}$: set of empty squares (labelled by integers)
- NEIGHBORS(i): set of empty squares that are adjacent to i
- $N(i)$: size of NEIGHBORS(i)

Transition Model

$$P(X_{t+1} = j \mid X_t = i) = \boldsymbol{T}_{ij} = \begin{cases} 1/N(i) & \text{if } j \in \text{NEIGHBORS}(i) \\ 0 & \text{otherwise} \end{cases}$$

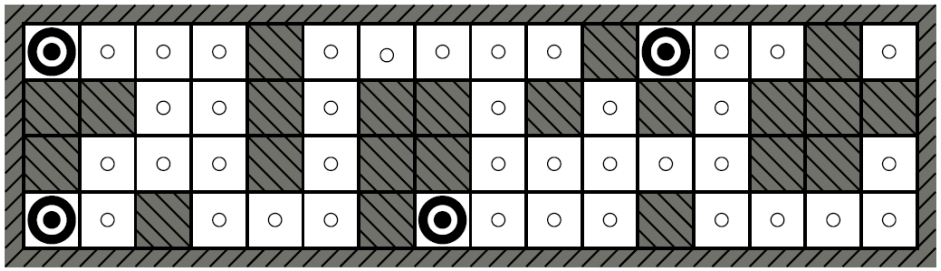


$n = 42$

Matrix $\boldsymbol{T} = (\boldsymbol{T}_{ij})$ has $42 \times 42 = 1764$ entries

Transition Model

$$P(X_{t+1} = j \mid X_t = i) = \boldsymbol{T}_{ij} = \begin{cases} 1/N(i) & \text{if } j \in \text{NEIGHBORS}(i) \\ 0 & \text{otherwise} \end{cases}$$



$n = 42$

Matrix $\boldsymbol{T} = (\boldsymbol{T}_{ij})$ has $42 \times 42 = 1764$ entries

Assume a uniform distribution of the robot's starting location:

$$P(X_0 = i) = 1/n \quad \text{for } 1 \leq i \leq n$$

Sensor Model

Sensor variable $E_t = NESW$ has four bits and $2^4 = 16$ possible values.

Sensor Model

Sensor variable $E_t = NESW$ has four bits and $2^4 = 16$ possible values.

Assumptions:

- Each sensor (N, E, S, or W) has an error rate ε.

Sensor Model

Sensor variable $E_t = NESW$ has four bits and $2^4 = 16$ possible values.

Assumptions:

- Each sensor (N, E, S, or W) has an error rate ε.
- Errors occur independently for the four sensors.

Sensor Model

Sensor variable $E_t = NESW$ has four bits and $2^4 = 16$ possible values.

Assumptions:

- Each sensor (N, E, S, or W) has an error rate ε.
- Errors occur independently for the four sensors.

  Probability of getting all four bits correct: $(1 - \varepsilon)^4$

  Probability of getting all of them wrong: ε^4

Sensor Model

Sensor variable $E_t = NESW$ has four bits and $2^4 = 16$ possible values.

Assumptions:

- Each sensor (N, E, S, or W) has an error rate ε.
- Errors occur independently for the four sensors.

  Probability of getting all four bits correct: $(1-\varepsilon)^4$

  Probability of getting all of them wrong: ε^4

d_{it} = #bits that are different between the true values for square i and actual reading e_t

Sensor Model

Sensor variable $E_t = NESW$ has four bits and $2^4 = 16$ possible values.

Assumptions:

- Each sensor (N, E, S, or W) has an error rate ε.
- Errors occur independently for the four sensors.

  Probability of getting all four bits correct: $(1-\varepsilon)^4$

  Probability of getting all of them wrong: ε^4

d_{it} = #bits that are different between the true values for square i and actual reading e_t

Probability that the robot in square i would receive a sensor reading e_t:

$$P(E_t = e_t \mid X_t = i) = (\boldsymbol{O}_t)_{ii} = (1-\varepsilon)^{4-d_{it}}\varepsilon^{d_{it}}$$

$\boldsymbol{O}_t$: diagonal observation matrix

Sensor Model

Sensor variable $E_t = NESW$ has four bits and $2^4 = 16$ possible values.

Assumptions:

- Each sensor (N, E, S, or W) has an error rate ε.
- Errors occur independently for the four sensors.

  Probability of getting all four bits correct: $(1-\varepsilon)^4$

  Probability of getting all of them wrong: ε^4

d_{it} = #bits that are different between the true values for square i and actual reading e_t

Probability that the robot in square i would receive a sensor reading e_t:

$$P(E_t = e_t \mid X_t = i) = (\boldsymbol{O}_t)_{ii} = (1-\varepsilon)^{4-d_{it}}\varepsilon^{d_{it}}$$

diagonal observation matrix

A square with obstacles to the north and south would produce a sensor reading of 1110 with probability $(1-\varepsilon)^3\varepsilon^1$.

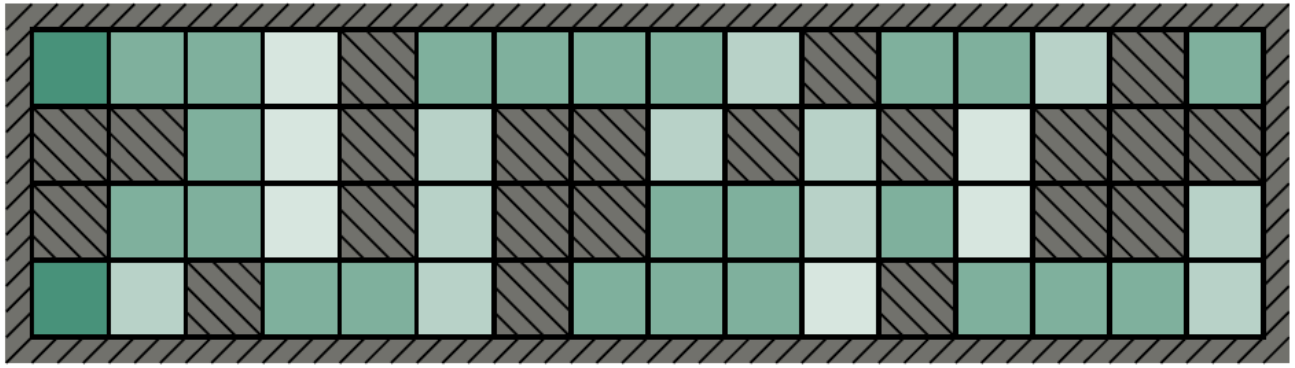
Localization

Posterior distribution over locations using the filtering equation:

$$\boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{e}_{1:t+1}) = \alpha \boldsymbol{O}_{t+1} \boldsymbol{T}^T \boldsymbol{P}(\boldsymbol{X}_t \mid \boldsymbol{e}_{1:t})$$

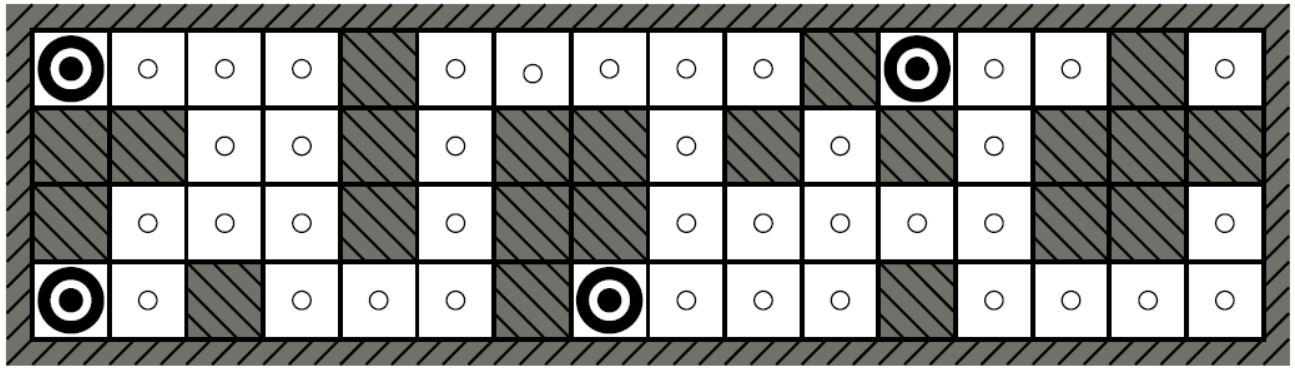
$n \times 1$ column vector $\boldsymbol{f}_{1:t+1}$

$\varepsilon = 0.2$



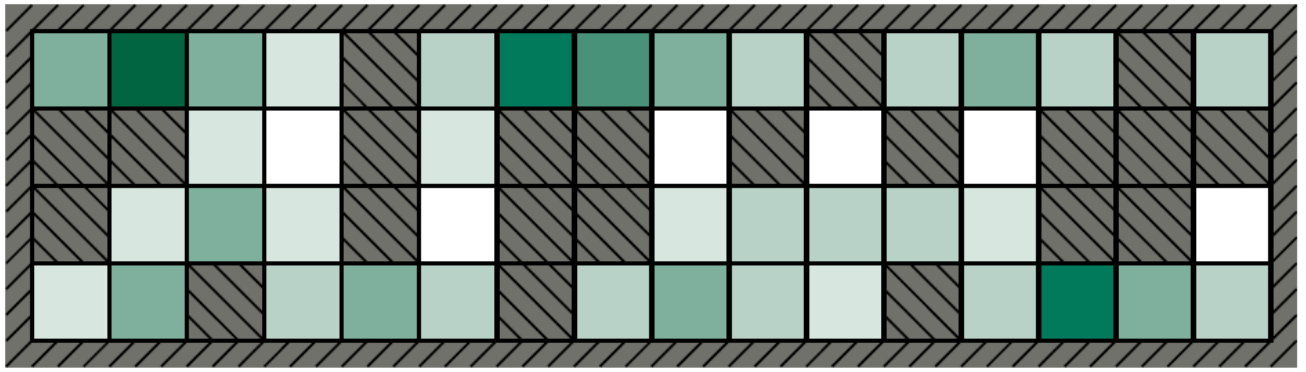
(a) Posterior distribution over robot location after $E_1 = 1011$

Perfect sensing



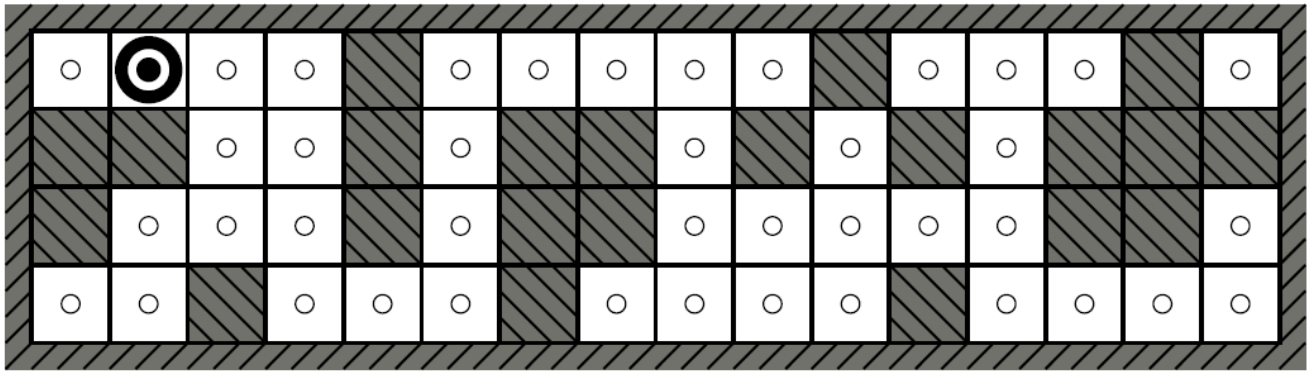
Localization (cont'd)

$\varepsilon = 0.2$



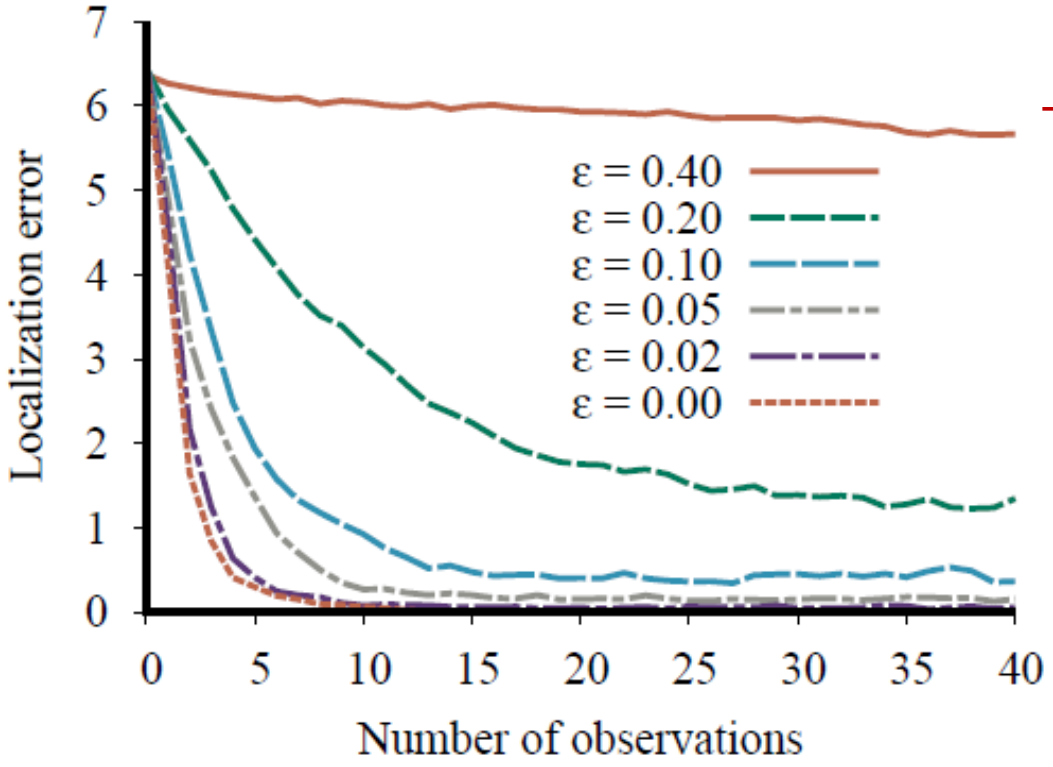
(b) Posterior distribution over robot location after $E_1 = 1011$, $E_2 = 1010$

Perfect sensing



Localization Error

Measured as the Manhattan distance from the true location.

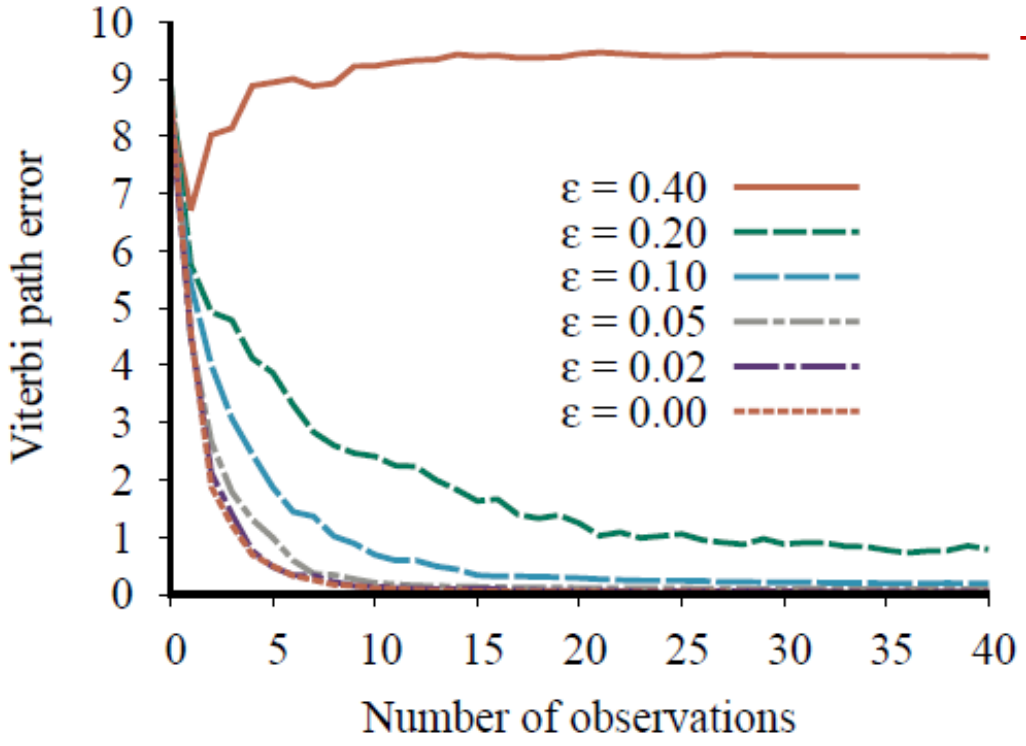


The robot is lost!

The robot can also use smoothing to work out where it was at a given past time.

Viterbi Path Error

Error measured as the average Manhattan distance of states on the Viterbi Path (most likely explanation) from corresponding states on the true path.



The robot is lost!