Artificial Intelligence Foundation – JC3001

Lecture 23: Uncertainty over Time II

Prof. Aladdin Ayesh (aladdin.ayesh@abdn.ac.uk)

Dr. Binod Bhattarai (binod.bhattarai@abdn.ac.uk)

Dr. Gideon Ogunniye, (g.ogunniye@abdn.ac.uk)

September 2025

Material adapted from:

Russell and Norvig (AIMA Book): Chapter 14 (14.1–14.3)

Course Progression

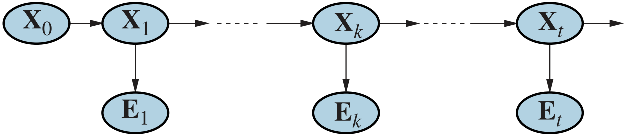
- Part 1: Introduction
  1. Introduction to AI ✓
  2. Agents ✓
- Part 2: Problem-solving
  1. Search 1: Uninformed Search ✓
  2. Search 2: Heuristic Search ✓
  3. Search 3: Local Search ✓
  4. Search 4: Adversarial Search ✓
- Part 3: Reasoning and Uncertainty
  1. Reasoning 1: Constraint Satisfaction ✓
  2. Reasoning 2: Logic and Inference ✓
  3. Probabilistic Reasoning 1: BNs ✓
  4. **Probabilistic Reasoning 2: HMMs**
- Part 4: Planning
  1. Planning 1: Intro and Formalism
  2. Planning 2: Algos and Heuristics
  3. Planning 3: Hierarchical Planning
  4. Planning 4: Stochastic Planning
- Part 5: Learning
  1. Learning 1: Intro to ML
  2. Learning 2: Regression
  3. Learning 3: Neural Networks
  4. Learning 4: Reinforcement Learning
- Part 6: Conclusion
  1. Ethical Issues in AI
  2. Conclusions and Discussion

Objectives

- Time and Uncertainty ✓
- Inference in Temporal Models
- Hidden Markov Models

Outline

- ▶ Inference in Temporal Models
- ▶ Hidden Markov Models



Smoothing computes $P(X_k \mid \mathbf{e}_{1:t})$, the posterior distribution of the state at some past time k given a complete sequence of observations from 1 to t.

$$P(X_k \mid \mathbf{e}_{1:t}) = P(X_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t})$$

$$
\begin{aligned}
P(X_k \mid \mathbf{e}_{1:t}) &= P(X_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\
&= \alpha P(X_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid X_k, \mathbf{e}_{1:k}) \qquad \text{(using Bayes' rule, given } \mathbf{e}_{1:k}\text{)}
\end{aligned}
$$

$$
\begin{aligned}
P(X_k \mid \mathbf{e}_{1:t}) &= P(X_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\
&= \alpha P(X_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid X_k, \mathbf{e}_{1:k}) && \text{(using Bayes' rule, given } \mathbf{e}_{1:k}\text{)} \\
&= \alpha P(X_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid X_k) && \text{(using conditional independence)}
\end{aligned}
$$

$$
\begin{aligned}
P(X_k \mid \mathbf{e}_{1:t}) &= P(X_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\
&= \alpha P(X_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid X_k, \mathbf{e}_{1:k}) && \text{(using Bayes' rule, given } \mathbf{e}_{1:k}\text{)} \\
&= \alpha P(X_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid X_k) && \text{(using conditional independence)} \\
&= \alpha \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}
\end{aligned}
$$

- where “$\times$” represents pointwise multiplication of vectors.
- backward message $\mathbf{b}_{k+1:t}$ can be computed by recursive process that runs backward from t

$$
\begin{aligned}
P(X_k \mid \mathbf{e}_{1:t}) &= P(X_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\
&= \alpha P(X_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid X_k, \mathbf{e}_{1:k}) && \text{(using Bayes' rule, given } \mathbf{e}_{1:k}\text{)} \\
&= \alpha P(X_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid X_k) && \text{(using conditional independence)} \\
&= \alpha \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}
\end{aligned}
$$

- where “$\times$” represents pointwise multiplication of vectors.
- backward message $\mathbf{b}_{k+1:t}$ can be computed by recursive process that runs backward from t

$$
P(\mathbf{e}_{k+1:t} \mid X_k) = \sum_{x_{k+1}} P(\mathbf{e}_{k+1:t} \mid X_k, x_{k+1}) P(x_{k+1} \mid X_k) \qquad \text{(conditioning on} X_{k+1}\text{)}
$$

$$
\begin{aligned}
P(X_k \mid \mathbf{e}_{1:t}) &= P(X_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\
&= \alpha P(X_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid X_k, \mathbf{e}_{1:k}) && \text{(using Bayes' rule, given } \mathbf{e}_{1:k}\text{)} \\
&= \alpha P(X_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid X_k) && \text{(using conditional independence)} \\
&= \alpha \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}
\end{aligned}
$$

- where "$\times$" represents pointwise multiplication of vectors.
- backward message $\mathbf{b}_{k+1:t}$ can be computed by recursive process that runs backward from t

$$
\begin{aligned}
P(\mathbf{e}_{k+1:t} \mid X_k) &= \sum_{x_{k+1}} P(\mathbf{e}_{k+1:t} \mid X_k, x_{k+1}) P(x_{k+1} \mid X_k) && \text{(conditioning on} X_{k+1}\text{)} \\
&= \sum_{x_{k+1}} P(\mathbf{e}_{k+1:t} \mid x_{k+1}) P(x_{k+1} \mid X_k) && \text{(by conditional independence)}
\end{aligned}
$$

$$
\begin{aligned}
P(X_k \mid \mathbf{e}_{1:t}) &= P(X_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\
&= \alpha P(X_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid X_k, \mathbf{e}_{1:k}) && \text{(using Bayes' rule, given } \mathbf{e}_{1:k}\text{)} \\
&= \alpha P(X_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid X_k) && \text{(using conditional independence)} \\
&= \alpha \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}
\end{aligned}
$$

- where "$\times$" represents pointwise multiplication of vectors.
- backward message $\mathbf{b}_{k+1:t}$ can be computed by recursive process that runs backward from t

$$
\begin{aligned}
P(\mathbf{e}_{k+1:t} \mid X_k) &= \sum_{x_{k+1}} P(\mathbf{e}_{k+1:t} \mid X_k, x_{k+1}) P(x_{k+1} \mid X_k) && \text{(conditioning on} X_{k+1}\text{)} \\
&= \sum_{x_{k+1}} P(\mathbf{e}_{k+1:t} \mid x_{k+1}) P(x_{k+1} \mid X_k) && \text{(by conditional independence)} \\
&= \sum_{x_{k+1}} P(\mathbf{e}_{k+1}, \mathbf{e}_{k+2:t} \mid x_{k+1}) P(x_{k+1} \mid X_k)
\end{aligned}
$$

$$
\begin{aligned}
P(X_k \mid \mathbf{e}_{1:t}) &= P(X_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\
&= \alpha P(X_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid X_k, \mathbf{e}_{1:k}) && \text{(using Bayes' rule, given } \mathbf{e}_{1:k}\text{)} \\
&= \alpha P(X_k \mid \mathbf{e}_{1:k}) P(\mathbf{e}_{k+1:t} \mid X_k) && \text{(using conditional independence)} \\
&= \alpha \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}
\end{aligned}
$$

- where "$\times$" represents pointwise multiplication of vectors.
- backward message $\mathbf{b}_{k+1:t}$ can be computed by recursive process that runs backward from t

$$
\begin{aligned}
P(\mathbf{e}_{k+1:t} \mid X_k) &= \sum_{x_{k+1}} P(\mathbf{e}_{k+1:t} \mid X_k, x_{k+1}) P(x_{k+1} \mid X_k) && \text{(conditioning on} X_{k+1}\text{)} \\
&= \sum_{x_{k+1}} P(\mathbf{e}_{k+1:t} \mid x_{k+1}) P(x_{k+1} \mid X_k) && \text{(by conditional independence)} \\
&= \sum_{x_{k+1}} P(\mathbf{e}_{k+1}, \mathbf{e}_{k+2:t} \mid x_{k+1}) P(x_{k+1} \mid X_k) \\
&= \sum_{x_{k+1}} \underbrace{P(\mathbf{e}_{k+1} \mid x_{k+1})}_{\text{sensor model}} \underbrace{P(\mathbf{e}_{k+2:t} \mid x_{k+1})}_{\text{recursion}} \underbrace{P(x_{k+1} \mid X_k)}_{\text{transition model}}
\end{aligned}
$$

We now have a recursive formulation for the backward message:

$$P(\mathbf{e}_{k+1:t} \mid X_k) = \sum_{x_{k+1}} P(\mathbf{e}_{k+1} \mid x_{k+1}) P(\mathbf{e}_{k+2:t} \mid x_{k+1}) P(x_{k+1} \mid X_k)$$

Which we can now use as a function in our next algorithm:

$$\mathbf{b}_{k+1:t} = \text{Backward}(\mathbf{b}_{k+2:t}, \mathbf{e}_{k+1})$$

function Forward-Backward($\boldsymbol{ev}$,$prior$) **returns** a vector of probability distributions
inputs: $\boldsymbol{ev}$, a vector of evidence values for steps $1, \ldots, t$
$prior$, the prior distribution on the initial state $P(X_0)$
local variables: $\boldsymbol{fv}$, a vector of forward messages for steps $0, \ldots, t$
b, a representation of the backward message, initially all 1s
$\boldsymbol{sv}$, a vector of smoothed estimates for steps $1, \ldots, t$
$\boldsymbol{fv}[0] \leftarrow prior$
for $i = 1$ **to** t **do**
$\boldsymbol{fv}[i] \leftarrow$ Forward$(\boldsymbol{fv}[i-1], \boldsymbol{ev}[i])$
for $i = t$ **down to** 1 **do**
$\boldsymbol{sv}[i] \leftarrow$ Normalize$(\boldsymbol{fv}[i] \times b)$
b $\leftarrow$ Backward(**b**$, ev[i])$
return $\boldsymbol{sv}$

The forward-backward algorithm for smoothing: computing posterior probabilities of a sequence of states given a sequence of observations

Back to the umbrella example, let us compute the smoothed estimate for rain on day one, given we saw an umbrella on days one and two, given by:

$$P(R_1 \mid u_1, u_2) = \alpha P(R_1 \mid u_1) P(u_2 \mid R_1)$$

Back to the umbrella example, let us compute the smoothed estimate for rain on day one, given we saw an umbrella on days one and two, given by:

$$P(R_1 \mid u_1, u_2) = \alpha P(R_1 \mid u_1) P(u_2 \mid R_1)$$

We know (from our filtering example), that $P(R_1 \mid u_1) = \langle 0.818, 0.182 \rangle$, so we need to compute the backward recursion

$$P(u_2 \mid R_1) = \sum_{r_2} P(u_2 \mid r_2) P(\mid r_2) P(r_2 \mid R_1)$$

Back to the umbrella example, let us compute the smoothed estimate for rain on day one, given we saw an umbrella on days one and two, given by:

$$P(R_1 \mid u_1, u_2) = \alpha P(R_1 \mid u_1) P(u_2 \mid R_1)$$

We know (from our filtering example), that $P(R_1 \mid u_1) = \langle 0.818, 0.182 \rangle$, so we need to compute the backward recursion

$$\begin{aligned} P(u_2 \mid R_1) &= \sum_{r_2} P(u_2 \mid r_2) P(\mid r_2) P(r_2 \mid R_1) \\ &= (0.9 \times 1 \times \langle 0.7, 0.3 \rangle) + (0.2 \times 1 \times \langle 0.3, 0.7 \rangle) \end{aligned}$$

Back to the umbrella example, let us compute the smoothed estimate for rain on day one, given we saw an umbrella on days one and two, given by:

$$P(R_1 \mid u_1, u_2) = \alpha P(R_1 \mid u_1) P(u_2 \mid R_1)$$

We know (from our filtering example), that $P(R_1 \mid u_1) = \langle 0.818, 0.182 \rangle$, so we need to compute the backward recursion

$$\begin{aligned} P(u_2 \mid R_1) &= \sum_{r_2} P(u_2 \mid r_2) P(\mid r_2) P(r_2 \mid R_1) \\ &= (0.9 \times 1 \times \langle 0.7, 0.3 \rangle) + (0.2 \times 1 \times \langle 0.3, 0.7 \rangle) = \langle 0.69, 0.41 \rangle \end{aligned}$$

Back to the umbrella example, let us compute the smoothed estimate for rain on day one, given we saw an umbrella on days one and two, given by:

$$P(R_1 \mid u_1, u_2) = \alpha P(R_1 \mid u_1) P(u_2 \mid R_1)$$

We know (from our filtering example), that $P(R_1 \mid u_1) = \langle 0.818, 0.182 \rangle$, so we need to compute the backward recursion

$$\begin{aligned} P(u_2 \mid R_1) &= \sum_{r_2} P(u_2 \mid r_2) P(\mid r_2) P(r_2 \mid R_1) \\ &= (0.9 \times 1 \times \langle 0.7, 0.3 \rangle) + (0.2 \times 1 \times \langle 0.3, 0.7 \rangle) = \langle 0.69, 0.41 \rangle \end{aligned}$$

Plugging this value in the equation above we have:

$$P(R_1 \mid u_1, u_2) = \alpha \langle 0.818, 0.182 \rangle \times \langle 0.69, 0.41 \rangle \approx \langle 0.883, 0.117 \rangle$$

Finding the most likely sequence

- There is a linear-time algorithm for finding the most likely sequence
- It relies on the same Markov property that yielded efficient algorithms for filtering and smoothing
- View each sequence as a path through a graph whose nodes are the possible states at each time step.
- likelihood of any path is the product of the transition probabilities along the path and the probabilities of the given observations at each state
- There is a recursive relationship between most likely paths to each state x_{t+1} and most likely paths to each state x_t

Finding the most likely sequence

- Recursively computed message $m_{1:t}$

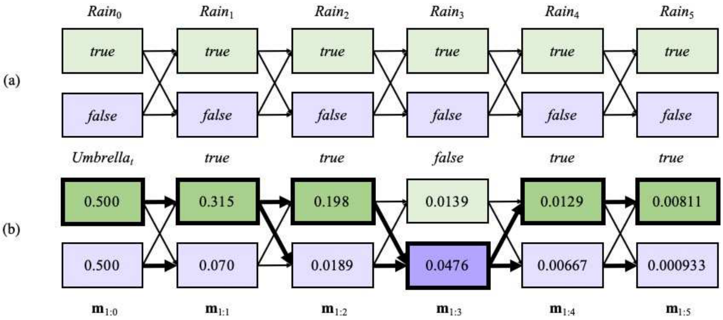
$$m_{1:t} = \max_{x_{1:t-1}}$$

$$\begin{aligned} m_{1:t+1} &= \max_{x_{1:t}} P(x_{1:t}, X_{t+1}, \mathbf{e}_{1:t+1}) = \max_{x_{1:t}} P(x_{1:t}, X_{t+1}, \mathbf{e}_{1:t}, \mathbf{e}_{t+1}) \\ &= \max_{x_{1:t}} P(\mathbf{e}_{t+1} \mid x_{t:1}, X_{t+1}, \mathbf{e}_{1:t}) P(x_{1:t}, X_{t+1}, \mathbf{e}_{1:t}) \\ &= P(\mathbf{e}_{t+1} \mid X_{t+1}) \max_{x_{1:t}} P(X_{t+1}, \mid x_t) P(x_{1:t}, \mathbf{e}_{1:t}) \\ &= P(\mathbf{e}_{t+1} \mid X_{t+1}) \max_{x_t} P(X_{t+1}, \mid x_t) \max_{x_{1:t-1}} P(x_{1:t-1}, x_t, \mathbf{e}_{1:t}) \end{aligned}$$

- $m_{1:t}$ will contain the probability for the most likely sequence reaching *each* of the final states.

Viterbi algorithm

- Select the final state of the most likely sequence overall.
- In order to identify the actual sequence, as opposed to just computing its probability, the algorithm will also need to record, for each state, the best state that leads to it.



(a) Possible state sequences for Rain can be viewed as paths through a graph of the possible states at each time step.

(b) Operation of the Viterbi algorithm for the umbrella observation sequence $[true; true; false; true; true]$, where the evidence starts at time 1.

- Inference in Temporal Models
- Hidden Markov Models

- Hidden Markov model, or HMM:
  - temporal probabilistic model
  - the state of the process is described by a single, discrete random variable
  - No restriction on the evidence variables.
    There can be many evidence variables, both discrete and continuous.

Simplified matrix algorithms

The transition model $P(X_t \mid X_{t-1})$ becomes an $S \times S$ matrix T where:

$$T_{i,j} = P(X_t = j \mid X_{t-1} = i)$$

$T_{i,j}$ is the probability of a transition from state i to state j

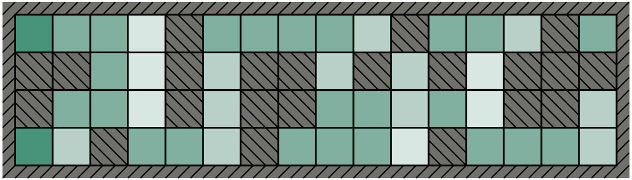
- Matrix formulation allows improved algorithms
  - simple variation on the forward-backward algorithm that allows smoothing to be carried out in constant space, independently of the length of the sequence
  - Online smoothing with a fixed lag.

function Fixed-Lag-Smoothing($\mathbf{e}$, hmm, d)
inputs: $\mathbf{e}_t$, the current evidence for time step t
hmm, a hidden Markov model with $S \times S$ transition matrix T
d, the length of the lag for smoothing
persistent: t, the current time, initially 1
$\mathbf{f}$, the forward message $P(X_t|\mathbf{e}_{1:t})$, initially hmm.Prior
B, the d-step backward transformation matrix, initially the identity matrix
$\mathbf{e}_{t-d:t}$, double-ended list of evidence from $t-d$ to t, initially empty
local variables: O_{t-d}, O_t , diagonal matrices containing the sensor model information
add $\mathbf{e}_t$ to the end of $\mathbf{e}_{t-d:t}$
$O_t \leftarrow$ diagonal matrix containing $P(\mathbf{e}_t \mid X_t)$
if t > d **then**
  $\mathbf{f} \leftarrow$ Forward$(\mathbf{f}, \mathbf{e}_{t-d})$
  remove $\mathbf{e}_{t-d-1}$ from the beginning of $\mathbf{e}_{t-d:t}$
  $O_{t-d} \leftarrow$ diagonal matrix containing $P(\mathbf{e}_{t-d} \mid X_{t-d})$
  $B \leftarrow O_{t-d}^{-1} T^{-1} B T O_t$
else $B \leftarrow BTO_t$
$t \leftarrow t+1$
if $t > d+1$ **then return** Normalize$(\mathbf{f} \times B1)$ **else return** $null$

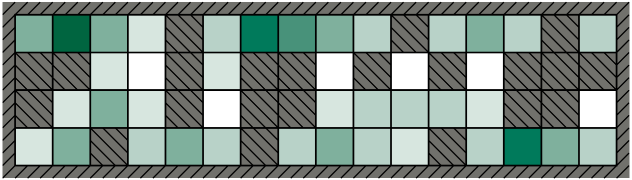
An algorithm for smoothing with a fixed time lag of d steps, implemented as an online algorithm that outputs the new smoothed estimate given the observation for a new time step.

Robot Localization:

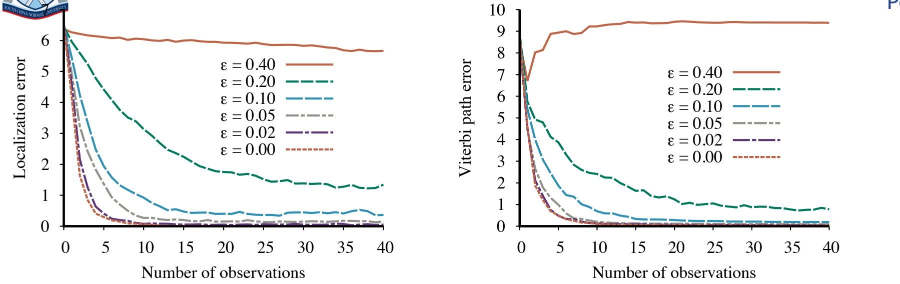
- Robot can start at any valid location of the 4×16 grid
- Moves randomly over the map following
- 42 Valid positions
- Sensor E_t has 16 values: 4-bit sequences of whether there is an obstacle: NESW



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



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



(a) (b)

HMM localization performance over the length of the observation sequence for various different sensor error probabilities ϵ; data averaged over 400 runs.

(a) The localization error, defined as the Manhattan distance from the true location.

(b) The Viterbi path error, defined as the average Manhattan distance of states on the Viterbi path from corresponding states on the true path

Uncertainty over Time Summary

- Temporal Models
  - Filtering
  - Prediction
  - Smoothing
  - Most likely explanation
- Viterbi's Algorithm
- Hidden Markov Models

Any Questions.