Probabilistic Reasoning Over Time

Joe Shymanski

November 13, 2023

1 HMMs

This section focuses on Hidden Markov Models where each timestep includes only a single state with any number of evidence variables per state.

1.1 State Estimation

Equation 1 defines the computation for the forward messages in the filtering or state estimation process.

$$\mathbf{f}_{1:t} = \alpha \mathbf{O}_t \mathbf{T}^T \mathbf{f}_{1:t-1} \tag{1}$$

Using this equation, we can compute state estimates for the HMM described in Exercises 15.15 and 15.17. Table 1 shows these estimates.

t	$P(\neg EnoughSleep_t \mid e_{1:t})$	$P(EnoughSleep_t \mid e_{1:t})$
1	0.13573407	0.86426593
2	0.49899428	0.50100572
3	0.89554467	0.10445533

Table 1: Results of state estimation.

1.2 Forward-Backward Smoothing

Equation 2 demonstrates how to compute the backward messages in the forward-backward algorithm.

t	$P(\neg EnoughSleep_t \mid e_{1:3})$	$P(EnoughSleep_t \mid e_{1:3})$
1	0.27225184	0.72774816
2	0.72431591	0.27568409
3	0.89554467	0.10445533

Table 2: Results of forward-backward smoothing.

t	$P_S(\neg E_t) - P_F(\neg E_t)$	$P_S(E_t) - P_F(E_t)$
1	0.13651777	-0.13651777
2	0.22532163	-0.22532163

Table 3: Differences between forward-backward smoothing and filtering.

$$\mathbf{b}_{k:t} = \mathbf{TO}_k \mathbf{b}_{k+1:t} \tag{2}$$

Using this equation, along with Equation 1 for the forward messages, we can smooth our probabilities using both past and future information at each state. The results of this forward-backward smoothing process can be seen in Table 2.

The differences between the original and the smoothed estimates can be seen in Table 3. As you can see, the probability that the student did not get enough sleep on days 1 and 2 increased substantially when we were able to consider the evidence after each day.

1.3 Fixed-Lag Smoothing

To expand the evidence given in Exercise 15.17, they are repeated in order to obtain 25 observations. This means that $e_1 = e_4 = ... = e_{25}$, $e_2 = e_5 = ... = e_{23}$, and $e_3 = e_6 = ... = e_{24}$. Now we can run the new fixed-lag algorithm to smooth these values even further. The results for lag values in $[2, 3, 4, 5]$ are given by Table 4.

1.4 *Viterbi* Algorithm

The most likely sequence, using the filtering results from Table 1, would be $[EnoughSleep, EnoughSleep, \neg EnoughSleep]$.

$t-d$	$d=2$	$d=3$	$d=4$	$d=5$
1	0.35282675	0.31498118	0.3084257	0.30067389
2	0.30584972	0.29350534	0.28386226	0.28506345
3	0.51481566	0.53617951	0.5410708	0.54473572
4	0.3077509	0.3099013	0.3040945	0.29898933
5	0.29361837	0.28810288	0.28312931	0.28458223
6	0.51134882	0.53592837	0.540475	0.54461527
7	0.30665714	0.30962599	0.30390947	0.29894732
8	0.29304866	0.28788361	0.28310324	0.28455994
9	0.51121519	0.53591887	0.54045158	0.54461065
10	0.30661531	0.30961514	0.30390226	0.29894571
11	0.29302638	0.28787509	0.28310223	0.28455907
12	0.51121001	0.5359185	0.54045067	0.54461047
13	0.30661368	0.30961472	0.30390198	0.29894565
14	0.29302551	0.28787476	0.28310219	0.28455904
15	0.51120981	0.53591848	0.54045064	0.54461046
16	0.30661362	0.3096147	0.30390197	0.29894564
17	0.29302547	0.28787474	0.28310218	0.28455903
18	0.51120978	0.53591846	0.54045061	0.54461043
19	0.30661359	0.30961466	0.30390192	0.29894557
20	0.2930254	0.2878746	0.28310202	0.28455882
21	0.51120939	0.53591781	0.54044986	0.
22	0.30661295	0.30961352	0.	0.
23	0.29302343	0.	0.	0.

Table 4: Comparing $P(EnoughSleep_{t-d})$ for different lag (d) values in fixed-lag smoothing.

2 DBNs

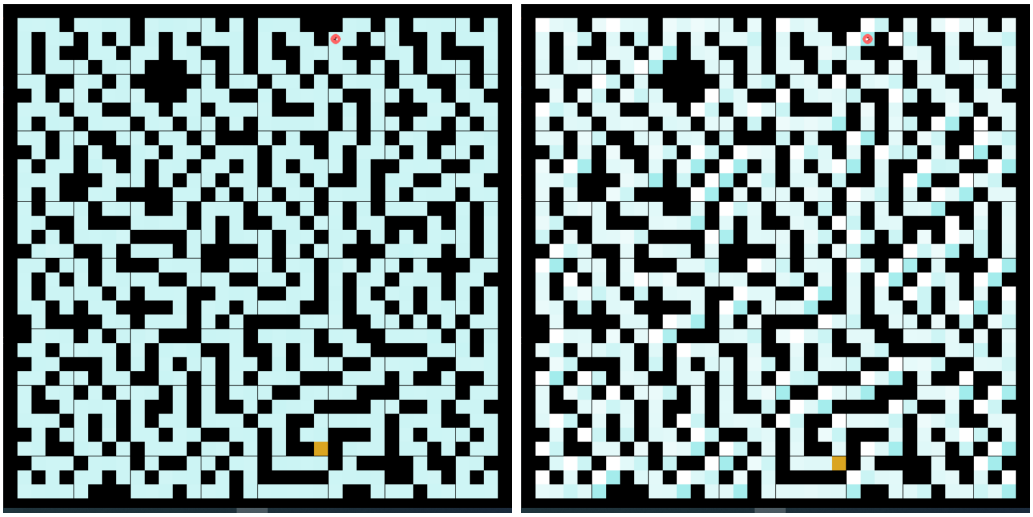
This portion of the project utilizes a Dynamic Bayesian Network (DBN) instead of the simple HMM structure from before, as there are multiple state variables at each timestep.

2.1 Particle Filtering for Robot Localization

Figure 1 shows screenshots of the robot localization environment at various timesteps $t \in [1, 100]$. The actual robot location is denoted by the red dot, and its true heading is given by the white line on the dot. The magnitude of each cell's location probability correlates to the blue color of the cell. The cell with the largest probability is highlighted in gold. The heading probabilities are displayed as black lines on the robot whose lengths represent the magnitude. In order to obtain more highlighted squares in the visualization, the fourth root of the location probability was used to calculate the color intensity of each cell.

The following assignments were used for the parameters of both the environment and the particle filtering algorithm: `seed` = 12345, `action_bias` = .1, `observation_noise` = .1, `action_noise` = .2, `dimensions` = $(36, 36)$, and `num_samples` = 50000. This means that the robot slightly prefers to move to the south and to the east, where every intended move has an 80% chance of occurring in the direction that the robot is now heading and every observation has a 90% chance of being entirely correct. At every timestep, the particle filtering algorithm samples 50000 states given the observation of the walls around the robot and the various probabilities of the environment (the transition and observation tables). A single state is represented as a vector $\langle location, heading \rangle$ where $location = (x, y)$ and $heading \in (S, E, N, W)$.

As you can see from Figure 1a, the grid starts out with roughly equal probabilities across the grid. Figure 1b shows how one additional step narrows down the location probabilities significantly. After roughly 30 steps, Figure 1c shows how particle filtering has specified two similarly shaped areas of the maze that the robot could be, though it correctly prefers the northern area to the southern one. Then the robot begins to drift into new territories in the maze. According to Figure 1d, particle filtering correctly eliminates the possibility of the robot being at the bottom of the maze and now has a highly accurate estimate of the robot's location.



(a) $t = 1$ (b) $t = 2$

(c) $t = 34$ (d) $t = 67$

Figure 1: Robot localization in a 36×36 maze.