AI Assignment 2

Åsmund Brekke

February 2017

1 Part A

The set of unobserved variable(s) for a time slice t, X_t is $Rain_t$.

The set of observable variables for a time slice t, E_t is $Umbrella_t$.

The dynamic model $P(X_t|X_{t-1})$ can be written as

$$T_{ij} = P(Rain_t = j|Rain_{t-1} = i) \tag{1}$$

$$\begin{bmatrix} P(Rain_t = f|Rain_{t-1} = f) & P(Rain_t = t|Rain_{t-1} = f) \\ P(Rain_t = f|Rain_{t-1} = t) & P(Rain_t = t|Rain_{t-1} = t) \end{bmatrix}$$

The observation model can be written as $P(E_t|X_t = i)$ where each i is a state. Since e_t is by definition observable at time t, we can get rid of two of the entries in the matrix. For simplicity, the book chose to place these values on the diagonal.

$$\begin{bmatrix} P(e_t|Rain_t = t) & 0 \\ 0 & P(e_t|Rain_t = f) \end{bmatrix}$$

The main assumption we have encoded in this model is that the current state is only dependent on the last state. This is called a first-order Markov process. We are also assuming that the laws governing the distribution stays consistent for all time slices. That means that its characteristics does not change, no matter what values of t evaluate it at. The assumptions made in this model is "okay". I would argue that the assumption that this is a first-order Markov process is inaccurate. For example, if it has rained for ten days straight, the probability that the director is coming in with an umbrella is higher than if it has only rained the previous day, although the model says these are equivalent.

2 Part B

The forward messages are as follows: $< 0.818181820.18181818 >$
$< 0.883357040.11664296 >$
$< 0.190667940.80933206 >$
$< 0.7307940.269206 >$
$< 0.867338890.13266111 >$

3 Part C

The backward messages are as follows:
$< 1, 1 >$
$< 0.69, 0.41 >$
$< 0.4593, 0.2437 >$
$< 0.090639, 0.150251 >$
$< 0.06611763, 0.04550767 >$
$< 0.04438457, 0.02422283 >$
$< 0.02941565, 0.01537503 >$

The Viterbi algorithm is an improvement over the forward-backward algorithm, and is used by calculating the most possible sequence of states in order to end up at a state X_t. We can look at this as a message $m_{1:t}$ = max(P(x_1 .... x_{t-1}, X_t $\|$e$_{1:t}$))). What this tells us is that we can calculate the most likely sequence of events that has lead us to our current state. In order to to this look at $\alpha * O_{t+1} * max(T_{t+1} m_{1:t})$, which can be calculated for each timeslot t.

```
Pseudo-code:
states = []
for each timeslot t:
    msg, state = calc_msg(msg, t) # Compute message according to (15.11), m_1:t
    states.add(state)
```

The algorithm of calc_msg is described above, and can also be seen in the source code. Note that the first message is just filtering.

Here calc_msg is looping through all possible states at timeslot t, and returns the message $m_{1:t}$, along with the state that achieved this maximum.

If we run the python code for the observations [True, True, False, True, True] (the same as the textbook, we get that the most likely scenario is [True, True, False, True, True].