Week 4: Probabilistic_Reasoning_Over_time

1. Recap and Motivation

The world can change rapidly and the Bayes rules does not take into account the sequential order of the observations.

So, we:

1. need not only consider "sensor model" $P(o|State)$
2. but also a "transition model" that predicts how the state changes over time.

From the belief state and a **transition model**, the agent can predict how the world might evolve in the next time step.
From the percepts observed and a **sensor model**, the agent can update the belief state.

2. Transition Model

2.1. Representing time

$$X_t = \text{set of unobservable state variables at time } t$$
$$E_t = \text{set of observable evidence variables at time } t$$

2.2. Markov Processes (Markov Chains)

2.2.1. Markov Assumption

Markov Assumption: the current state depends on only **a finite fixed number** of previous states.

2.2.2. Transition Model

1. **First—order Markov process**:

$$P(X_t|X_{0:t-1}) = P(X_t|X_{t-1})$$

2. **Second—order Markov process**:

$$P(X_t|X_{0:t-1}) = P(X_t|X_{t-1}, X_{t-2})$$

2.2.3. Sensor Model(Observation Model)

Sensor Markov Assumption:

$$P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_t)$$

2.2.4. Stationary Process

transition model and sensor model fixed for all t

2.3. Hidden Markov Model

Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process – call it X – with **unobservable (“hidden”) states**. HMM assumes that there is another process Y whose behavior “depends” on X. The goal is to learn about X by observing Y.

HMM stipulates that, the system conform transition Markov Assumption and the Sensor Markov Assumption.

2.4. Joint Distribution

Using **chain rule** and Markov Assumption: (Unrolled Over time)

$$P(X_0, E_{1:t}) = P(X_0) \prod_{i=1:t} P(X_i|X_{i-1})P(E_i|X_i)$$

1. Why E_0 ignored?

   Because X_0 is know, and we do not need E_0 to judge X_0

2.5. Fixes of First Order Markov Assumption

First-order Markov Assumption not exactly true in real world!

Possible Fixes:

1. Increase order of Markov Process

2. Augment state

3. Inference on Markov Chains

3.1. Types of Queries

1. **Filtering(state estimation)**: $P(X_t|e_{1\,:\,t})$
2. **Prediction**: $P(X_{t\,+\,k}|e_{1\,:\,t})$
3. **Smoothing**:

   Smoothing provides a better estimate of the state than was available at the time, because it incorporate more evidence.

   > In particular, when tracking a moving object with inaccurate position observations, smoothing gives a smoother estimated trajectory than filtering—hence the name.

4. **Most likely explanation**: $argmax_{x_{1\,:\,t}} P(x_{1\,:\,t}|e_{1\,:\,t})$

   Given a sequence of observations, we might wish to find the sequence of states that is most likely to have generated those observations.

3.2. Filtering and Prediction

3.2.1. Filtering(state estimation)

1. **Recursive estimation**:

$$P(x_{t+1}|e_{1:t+1}) = f(e(t), P(X_t|e_{1:t}))$$

that is, inference the new belief at time t+1 from old belief at time t

$$\begin{aligned}
\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) &= \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}, \mathbf{e}_{t+1}) \quad \text{(dividing up the evidence)} \\
&= \alpha\, \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}, \mathbf{e}_{1:t})\, \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) \quad \text{(using Bayes' rule)} \\
&= \alpha\, \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})\, \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) \quad \text{(by the sensor Markov assumption).} \qquad (15.4)
\end{aligned}$$

HMM_Filtering.png

> The second term is a one-step prediction, and equation 15.4 can be seen as we use new observation update the prediction; the first item is obtainable directly from the sensor model

$$\begin{aligned}
\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) &= \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}) \Sigma_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t, \mathbf{e}_{1:t}) P(\mathbf{x}_t|\mathbf{e}_{1:t}) \\
&= \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}) \Sigma_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{e}_{1:t})
\end{aligned}$$

HMM_Filtering_2.png

1. **Forward Messages**:

$$f_{1:t+1} = Forward(f_{1:t}, e_{t+1})$$

The filtered estimate $P(X_t|e_{1\,:\,t})$ as a “message” f, that is **propagated forward** along the sequence**, modified by each transition and updated by each new observation**.

2. stationary distribution (fixed point)

The predicted distribution for rain converges to a fixed point, after which it remains constant for all time. This is the **stationary distribution** of the Markov process defined by the transition model. In practical terms, this dooms to failure any attempt to predict the actual state for a number of steps that is more than a small fraction of the **mixing time**.

3.2.2. Prediction

predicting beyond a fraction of the mixing time will not work。

3.2.3. Smoothing (Forward-Backward Algorithm)

$$\begin{aligned}
\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\
&= \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{e}_{1:k}) \quad \text{(using Bayes' rule)} \\
&= \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) \quad \text{(using conditional independence)} \\
&= \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t} \qquad (15.8)
\end{aligned}$$

$$b_{k+1:t|X_k} = P(e_{k+1:t}|X_k) : \text{backward message}$$

$$\begin{aligned}
\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) &= \textstyle\sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k) \\
&= \textstyle\sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k) \\
&= \textstyle\sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})P(\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)
\end{aligned}$$

$$b_{k+1:t} = Backward(b_{k+2:t}, e_{k+1})$$
$$b_{t+1:t} = 1$$

- In order to reduce the time complexity to $o(t)$. The key to the linear—time algorithm is to r**ecord the results** of forward filtering over the whole sequence.

3.2.5. Finding the most likely sequence

1. Rough Way: use smoothing to compute states at each time.

   But, local optimal does not mean global optimum

2. Viterbi Algorithm

The easiest way to think about the problem is to view each sequence as a path through a graph whose nodes are the possible states at each time step.

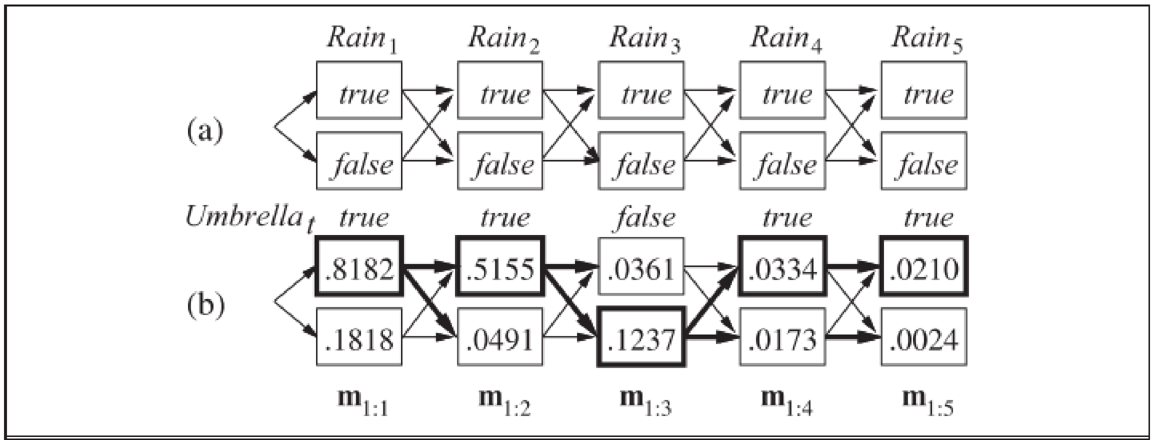


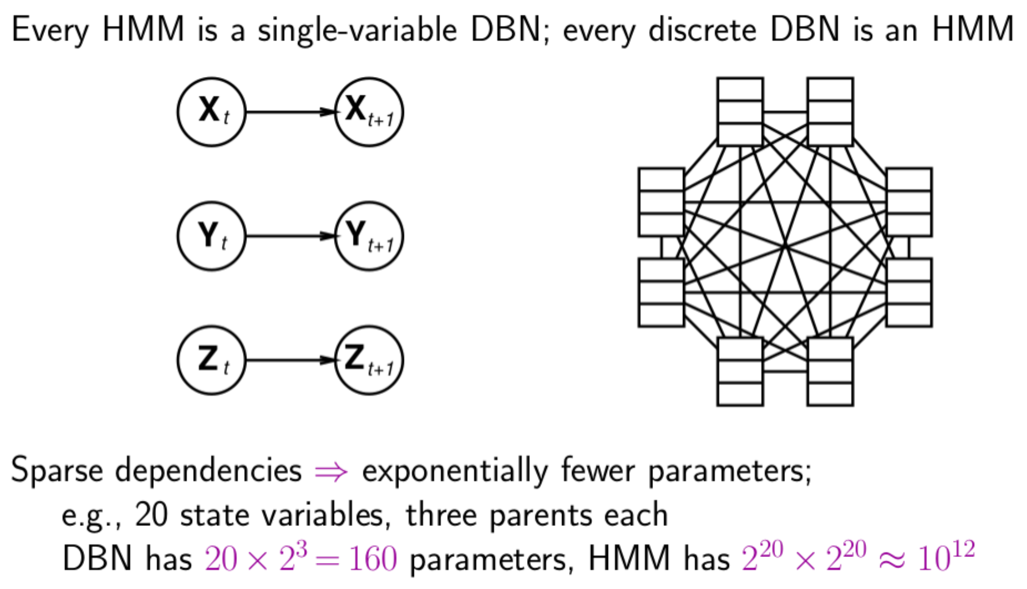
Figure 15.5 (a) Possible state sequences for $Rain_t$ can be viewed as paths through a graph of the possible states at each time step. (States are shown as rectangles to avoid confusion with nodes in a Bayes net.) (b) Operation of the Viterbi algorithm for the umbrella observation sequence $[true, true, false, true, true]$. For each t, we have shown the values of the message $\mathbf{m}_{1:t}$, which gives the probability of the best sequence reaching each state at time t. Also, for each state, the bold arrow leading into it indicates its best predecessor as measured by the product of the preceding sequence probability and the transition probability. Following the bold arrows back from the most likely state in $\mathbf{m}_{1:5}$ gives the most likely sequence.

- Because of Markov Assumption, there is a recursive relationship between most likely paths to each state x_{t+1}, 1 and most likely paths to each state x_t, just like dynamic programming.
- From the most likely statein the previous part, we calculate the next time-step distribution

4. Dynamic Bayesian Networks(DBN)

Every hidden Markov model can be represented as a DBN with a single state variable and a single evidence variable. Every discrete—variable DBN can be represented as an HMM.

> By decomposing the state of a complex system into its constituent variables, we can take advantage of sparseness in the temporal probability model.



Every HMM is a single-variable DBN; every discrete DBN is an HMM

Sparse dependencies $\Rightarrow$ exponentially fewer parameters;

e.g., 20 state variables, three parents each

DBN has $20 \times 2^3 = 160$ parameters, HMM has $2^{20} \times 2^{20} \approx 10^{12}$

Traditional HMM $\longrightarrow$ use a state transition matrix

DBN $\longrightarrow$ use structure like Bayesian network

4.1. Exact Inference in DBNs

1. Naive Method

   unroll the network and run any exact algorithm

2. Variable Elimination

   Variable Elimination is very hard in DBNs, because the factors to construct will be huge, we will include all variables that have parents in previous state.

4.2. Approximate Inference in DBNs

4.2.1. Likelihood methods (not appropriate)

Likelihood methods is not appropriate because:

1. running each sample from step 1 to t: time needed grows over time
2. states are sampled independently of the evidence

4.2.2. Particle Filtering

Basic idea: ensure that the population of samples ("particles") tracks the high-likelihood regions of the state-space

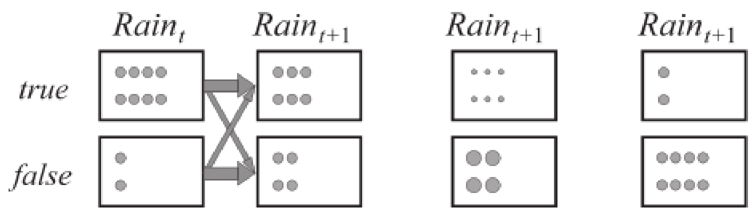
Two main ideas:

1. run all N samples at the same time $\rightarrow$ particles' themselves represent belief
2. resampling $\rightarrow$ focus attention on parts of state space with large probability under the evidence

Particle_Filtering

A family of algorithms called **particle filtering** is designed to do just that. Particle filtering works as follows: First, a population of initial-state samples is created by sampling from the prior distribution $\mathbf{P}(\mathbf{X}_0)$. Then the update cycle is repeated for each time step:

1. Each sample is propagated forward by sampling the next state value $\mathbf{x}_{t+1}$ given the current value $\mathbf{x}_t$ for the sample, based on the transition model $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)$.
2. Each sample is weighted by the likelihood it assigns to the new evidence, $(\mathbf{e}_{t+1} \mid \mathbf{x}_{t+1})$.
3. The population is *resampled* to generate a new population of samples. Each new sample is selected from the current population; the probability that a particular sample is selected is proportional to its weight. The new samples are unweighted.



(a) Propagate (b) Weight (c) Resample

Figure 15.18 The particle filtering update cycle for the umbrella DBN with $= 10$, showing the sample populations of each state. (a) At time t, 8 samples indicate $rain$ and 2 indicate $\neg rain$. Each is propagated forward by sampling the next state through the transition model. At time $t + 1$, 6 samples indicate $rain$ and 4 indicate $\neg rain$. (b) $\neg umbrella$ is observed at $t + 1$. Each sample is weighted by its likelihood for the observation, as indicated by the size of the circles. (c) A new set of 10 samples is generated by weighted random selection from the current set, resulting in 2 samples that indicate $rain$ and 8 that indicate $\neg rain$.

Propagate forward: populations of $\mathbf{x}_{t+1}$ are

$$N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t}) = \Sigma_{\mathbf{x}_t} P(\mathbf{x}_{t+1}|\mathbf{x}_t) N(\mathbf{x}_t|\mathbf{e}_{1:t})$$

assumption:
consistent at stage *t*
$N(\mathbf{x}_t|\mathbf{e}_{1:t})/N = P(\mathbf{x}_t|\mathbf{e}_{1:t})$

Weight samples by their likelihood for $\mathbf{e}_{t+1}$:

$$W(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1}) = P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1}) N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t})$$

Resample to obtain populations proportional to W:

$$\begin{aligned}
N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1})/N &= \alpha W(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1}) = \alpha P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1}) N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t}) \\
&= \alpha P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1}) \Sigma_{\mathbf{x}_t} P(\mathbf{x}_{t+1}|\mathbf{x}_t) N(\mathbf{x}_t|\mathbf{e}_{1:t}) \\
&= \alpha' P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1}) \Sigma_{\mathbf{x}_t} P(\mathbf{x}_{t+1}|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{e}_{1:t}) \\
&= P(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1})
\end{aligned}$$