Temporal probability models

Chapter 15, Sections 1–3

Outline

◇ Time and uncertainty

◇ Inference: filtering, prediction, smoothing

◇ Hidden Markov models

Time and uncertainty

The world changes; we need to track and predict it

Our basic idea is to copy state and evidence variables for each time step

$\mathbf{X}_t$ = set of unobservable state variables at time t
e.g., $BloodSugar_t$, $StomachContents_t$, etc.

$\mathbf{E}_t$ = set of observable evidence variables at time t
e.g., $MeasuredBloodSugar_t$, $PulseRate_t$, $FoodEaten_t$

This assumes **discrete time**; the step size depends on the problem

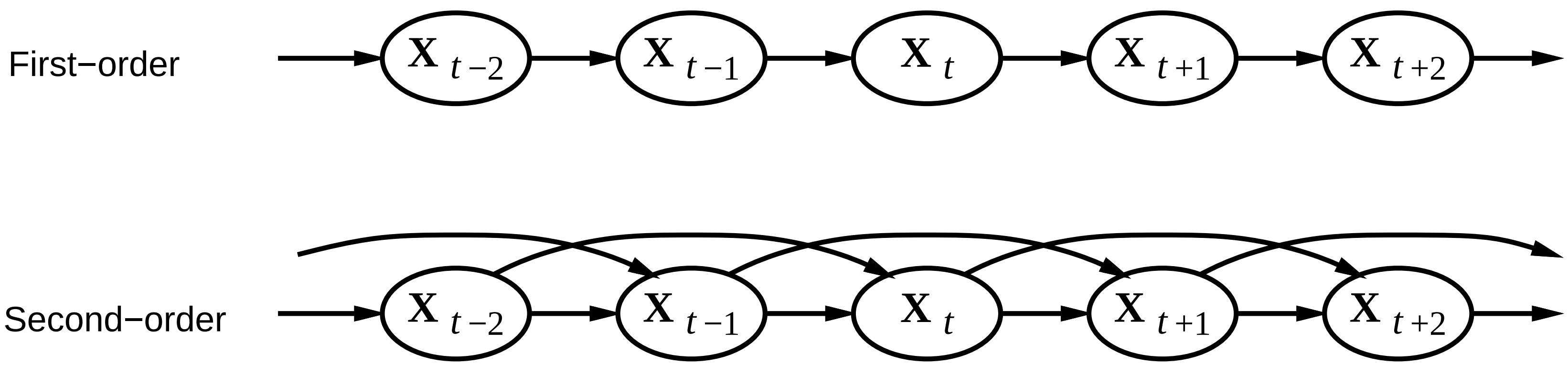
Notation: $\mathbf{X}_{a:b} = \mathbf{X}_a, \mathbf{X}_{a+1}, \ldots, \mathbf{X}_{b-1}, \mathbf{X}_b$

We want to construct a Bayes net from these variables:
– what are the parents of $\mathbf{X}_t$ and $\mathbf{E}_t$?

Markov chains

A Markov chain has a single observable state $\mathbf{X}_t$ that obeys the Markov assumption: $\mathbf{X}_t$ depends on a **bounded** subset of $\mathbf{X}_{0:t-1}$

First-order Markov process: $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$



Second-order Markov process: $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-2}, \mathbf{X}_{t-1})$

(can be reduced to 1st order by using $\langle \mathbf{X}_{t-2}, \mathbf{X}_{t-1} \rangle$ as the state)

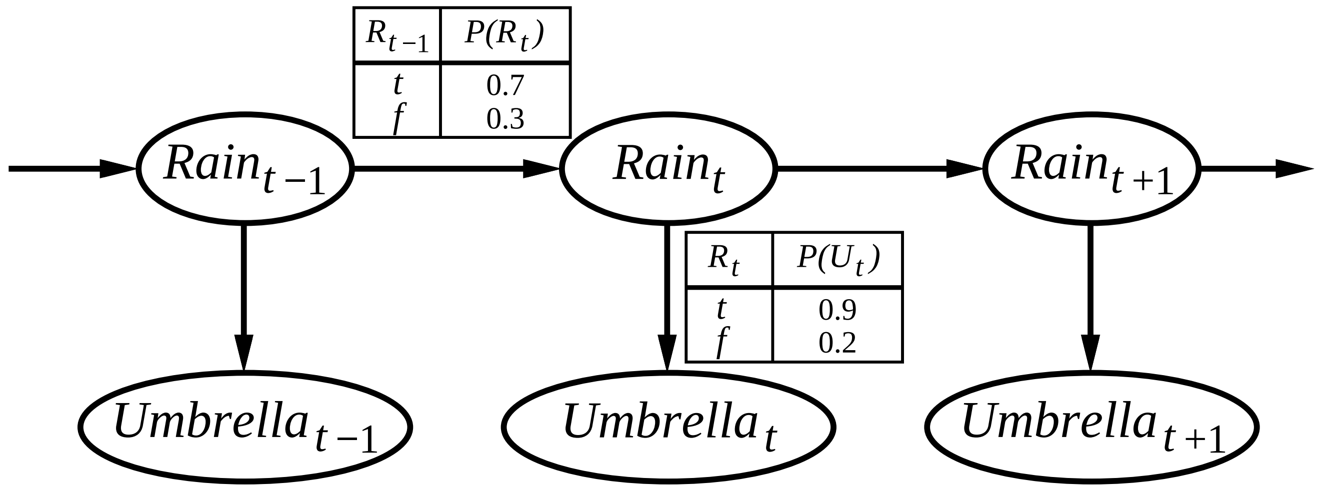
Hidden Markov models (HMM)

A HMM contains a Markov chain $\mathbf{X}_t$, which is **not** observable.

Instead we observe the evidence variables $\mathbf{E}_t$, and assume that they obey the Sensor Markov assumption: $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_{0:t}, \mathbf{E}_{0:t-1}) = \mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$

Both Markov chains and HMMs are stationary processes:
– the transition model $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$ and
– the sensor model $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$ are fixed for all t

Example



Neither the Markov assumption nor the sensor Markov assumtion are exactly true in the real world!

Possible fixes:

1. **Increase the order** of the Markov process
2. **Augment the state**, e.g., add $Temp_t$, $Pressure_t$

Inference tasks

Filtering: $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$

to compute the current belief state given all evidence

better name: state estimation

Prediction: $\mathbf{P}(\mathbf{X}_{t+k}|\mathbf{e}_{1:t})$ for $k > 0$

to compute a **future** belief state, given current evidence

(it's like filtering without all evidence)

Smoothing: $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ for $0 \leq k < t$

to compute a better estimate of past states

Most likely explanation: $\arg\max_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t}|\mathbf{e}_{1:t})$

to compute the state sequence that is most likely, given the evidence

Applications: speech recognition, decoding with a noisy channel, etc.

Filtering / state estimation

A useful filtering algorithm needs to maintain a current state and update it, instead of recalculating everything. I.e., we need a function f such that:

$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) = f(\mathbf{e}_{t+1}, \mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t}))$$

We compose the evidence $\mathbf{e}_{1:t+1}$ into $\mathbf{e}_{1:t}$ and $\mathbf{e}_{t+1}$:

$$\begin{aligned}
\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) &= \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}, \mathbf{e}_{t+1}) && \text{(divide the evidence)} \\
&= \alpha\, \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}, \mathbf{e}_{1:t})\, \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}) && \text{(Bayes' rule)} \\
&= \alpha\, \underbrace{\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})}_{\text{the sensor model}}\, \underbrace{\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})}_{\text{prediction}} && \text{(Sensor Markov assumption)}
\end{aligned}$$

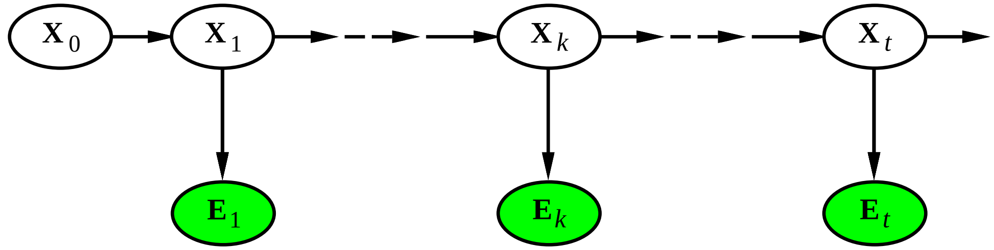
We obtain the one-step prediction by conditioning on the current state $\mathbf{X}_t$:

$$\begin{aligned}
\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{X}_t, \mathbf{e}_{1:t})\, \mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t}) \\
&= \underbrace{\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{X}_t)}_{\text{the Markov model}}\, \underbrace{\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})}_{\text{previous estimate}} && \text{(Markov assumption)}
\end{aligned}$$

Our final equation becomes this:

$$\underbrace{\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1})}_{\text{current estimate} = \mathbf{f}_{1:k+1}} = \alpha\, \underbrace{\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})}_{\text{the sensor model}}\, \underbrace{\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{X}_t)}_{\text{the Markov model}}\, \underbrace{\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})}_{\text{previous estimate} = \mathbf{f}_{1:k}}$$

Smoothing



Divide evidence $\mathbf{e}_{1:t}$ into $\mathbf{e}_{1:k}$, $\mathbf{e}_{k+1:t}$:

$$\begin{aligned}
\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\
&= \alpha\, \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\, \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{e}_{1:k}) && \text{(Bayes' rule)} \\
&= \alpha\, \underbrace{\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})}_{\mathbf{f}_{1:k}}\, \underbrace{\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k)}_{\mathbf{b}_{k+1:t}} && \text{(conditional independence)}
\end{aligned}$$

The backward message $\mathbf{b}_{k+1:t}$ is computed by backwards recursion:

$$\begin{aligned}
\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) &= \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{X}_{k+1})\, \mathbf{P}(\mathbf{X}_{k+1}|\mathbf{X}_k) \\
&= \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_{k+1})\, \mathbf{P}(\mathbf{X}_{k+1}|\mathbf{X}_k) \\
&= \underbrace{\mathbf{P}(\mathbf{e}_{k+1}|\mathbf{X}_{k+1})}_{\text{the sensor model}}\, \underbrace{\mathbf{P}(\mathbf{e}_{k+2:t}|\mathbf{X}_{k+1})}_{\mathbf{b}_{k+2:t}}\, \underbrace{\mathbf{P}(\mathbf{X}_{k+1}|\mathbf{X}_k)}_{\text{the Markov model}}
\end{aligned}$$

Forward and backward

Forward algorithm is used to compute the current belief state

Backward algorithm is used to compute a previous belief state

Forward–backward algorithm: cache forward messages along the way, which can then be used when going backward

Most likely explanation

Most likely sequence $\neq$ sequence of most likely states!

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

Most likely path to each $\mathbf{x}_{t+1}$ = most likely path to **some** $\mathbf{x}_t$, plus one step.
Since we don't care about the exact values, we can forget α.

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

$\mathbf{m}_{1:t}$ is the probability distribution of the most likely path to each $\mathbf{x}_t \in \mathbf{X}_t$, and is calculated by the Viterbi algorithm:

$$
\mathbf{m}_{1:t+1} = \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}) \ \max_{\mathbf{x}_t}(\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t) \ \mathbf{m}_{1:t})
$$

Hidden Markov models

$\mathbf{X}_t$ is a single, discrete variable X_t (and usually $\mathbf{E}_t$ is too)
Assume that the domain of X_t is $\{1, \ldots, S\}$

Transition matrix $\mathbf{T}_{ij} = P(X_t = j | X_{t-1} = i)$,

e.g., the rain matrix $\begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$

Sensor matrix $\mathbf{O}_t$ for each time step t, consists of diagonal elements $P(e_t | X_t = i)$

e.g., with $U_1 = true$, $\mathbf{O}_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$

Forward and backward messages can now be represented as column vectors:

$$\begin{aligned} \mathbf{f}_{1:t+1} &= \alpha\, \mathbf{O}_{t+1}\, \mathbf{T}^{\top}\, \mathbf{f}_{1:t} \\ \mathbf{b}_{k+1:t} &= \mathbf{T}\, \mathbf{O}_{k+1}\, \mathbf{b}_{k+2:t} \end{aligned}$$

The forward-backward algorithm needs time $O(S^2 t)$ and space $O(St)$

Summary for HMMs

Temporal models use state $\mathbf{X}_t$ and sensor $\mathbf{E}_t$ variables replicated over time

To make the models tractable, we introduce simplifying assumptions:

– Markov assumption: $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$

– sensor assumption: $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_{0:t}, \mathbf{E}_{0:t-1}) = \mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$

– stationarity: $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1}) = \mathbf{P}(\mathbf{X}_{t'}|\mathbf{X}_{t'-1})$, $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t) = \mathbf{P}(\mathbf{E}_{t'}|\mathbf{X}_{t'})$

With the assumptions we only need the following models:

– the transition model $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$

– the sensor model $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$

Possible computing tasks:

– filtering/state estimation, prediction, smoothing, most likely sequence

– **all can be done with constant cost per time step**

HMMs and extensions

Hidden Markov models (HMMs) have a single **discrete** state variable

- the rain/umbrella world is an HMM
- used for speech recognition, part-of-speech tagging, etc.
- n discrete state variables can be combined into one "megavariable"

Kalman filters allow n **continuous** state variables

- the state and transition models are linear Gaussian distributions
- update complexity $O(n^3)$
- used for tracking of moving objects, etc.

Dynamic Bayes nets subsume HMMs, Kalman filters

- exact update intractable
- particle filtering is a good approximate filtering algorithm for DBNs