Temporal probability models
Chapter 15, Sections 1–5

Outline
♦ Time and uncertainty
♦ Inference: filtering, prediction, smoothing
♦ Hidden Markov models
♦ Kalman filters (a brief mention)
♦ Dynamic Bayesian networks
♦ Particle filtering

Outline

Evaluation

Example

First-order Markov assumption not exactly true in real world!
Possible fixes:
1. Increase order of Markov process
2. Augment state: e.g., add Temp, Pressure

Example: robot state, e.g., add Umbrella, Battery

Time and uncertainty

The world changes; we need to track and predict it
Diabetes management vs vehicle diagnosis
Basic idea: copy state and evidence variables for each time step
  - \( X_t \) = set of unobservable state variables at time \( t \)
  - e.g., BloodSugar, StomachContents, etc.
  - \( E_t \) = set of observable evidence variables at time \( t \)
  - e.g., MeasuredBloodSugar, PulseRate, FoodEaten
This assumes discrete time; step size depends on problem
Notation: \( X_{t:t} = X_t, X_{t+1}, \ldots, X_{t-1}, X_t \)

Inference tasks

Filtering: \( P(X_t|\theta_{1:t}) \)
  - belief state—input to the decision process of a rational agent
Prediction: \( P(X_{t+k}|\theta_t) \) for \( k > 0 \)
  - evaluation of possible action sequences;
  - like filtering without the evidence
Smoothing: \( P(X_t|\theta_{0:t}) \) for \( 0 \leq k < t \)
  - better estimate of past states, essential for learning
Most likely explanation: arg \( \max_{\phi_{0:t}} P(X_{t:t}|\theta_t) \)
  - speech recognition, decoding with a noisy channel

Sensor Markov assumption: \( P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_t) \)
Stationary process: transition model \( P(X_t|X_{t-1}) \) and
sensor model \( P(E_t|X_t) \) fixed for all \( t \)
Filtering example

```
True  0.500  0.500  0.627
False 0.500  0.818  0.983

Rain₁  Rain₂  Rain₃

Umbrella₁  Umbrella₂
```

Most likely explanation

Most likely sequence ≠ sequence of most likely states!!!!

Most likely path to each $x_{1:t+1}$ = most likely path to some $x_i$ plus one more step

$$P(x_1, \ldots, x_{t+1}|e_{1:t+1})$$

$P(\theta_{t+1}|x_{t+1})\max \left(P(x_{t+1}|x) \max_{x_{t-1}} P(x_1, \ldots, x_{t-1}, x|e_t)\right)$

Identical to filtering, except $f_{1:k}$ replaced by

$$m_{i,t+1} = \max_{x_{t+1}} P(x_1, \ldots, x_{t-1}, x_{t+1}|e_t)$$

i.e., $m_{i,t}(i)$ gives the probability of the most likely path to state $i$. Update has sum replaced by max, giving the Viterbi algorithm:

$$m_{i,t+1} = P(\theta_{t+1}|x_{t+1})\max(P(x_{t+1}|x)m_{i,t})$$

Viterbi example

```
<table>
<thead>
<tr>
<th>State</th>
<th>Rain₁</th>
<th>Rain₂</th>
<th>Rain₃</th>
<th>Rain₄</th>
<th>Rain₅</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>Umbrella</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>0.818</td>
<td>5155</td>
<td>0361</td>
<td>0334</td>
<td>0210</td>
</tr>
<tr>
<td></td>
<td>18.18</td>
<td>0491</td>
<td>1233</td>
<td>0178</td>
<td>0024</td>
</tr>
</tbody>
</table>
```

Divide evidence $e_{1:t}$ into $e_{1:k}, e_{k+1:t}$:

$$P(X_x|e_{1:k}) = P(X_x|e_{1:k}, e_{k+1:t})$$

$$= \alpha P(X_x|e_{1:k}) P(e_{k+1:t}|x_{k+1}, e_{1:k})$$

$$= \alpha P(X_x|e_{1:k}) P(e_{k+1:t}|x_k)$$

$$= \alpha f_{1:k} b_{k+1:t}$$

Backward message computed by a backwards recursion:

$$P(\theta_{k+1}|X_k) = \sum_{X_{k+1}} P(\theta_{k+1}|X_k, X_{k+1}) P(X_{k+1}|X_k)$$

$$= \sum_{X_{k+1}} P(\theta_{k+1}|X_{k+1}) P(x_{k+1}|X_k)$$

$$= \sum_{X_{k+1}} P(\theta_{k+1}|X_{k+1}) P(e_{k+2:t}|x_{k+1}) P(x_{k+1}|X_k)$$
Country dance algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

\[ f_{t+1} = \alpha O_{t+1}^T f_t \]
\[ O_{t+1}^T f_{t+1} = \alpha f_t \]
\[ \alpha'(T^T)^{-1}O_{t+1}^T f_{t+1} = f_t \]

Algorithm: forward pass computes \( f_t \), backward pass does \( f_{t+1} \), \( b_t \).

Country dance algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

\[ f_{t+1} = \alpha O_{t+1}^T f_t \]
\[ O_{t+1}^T f_{t+1} = \alpha f_t \]
\[ \alpha'(T^T)^{-1}O_{t+1}^T f_{t+1} = f_t \]

Algorithm: forward pass computes \( f_t \), backward pass does \( f_{t+1} \), \( b_t \).
Country dance algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

\[ f_{t+1} = \alpha O_{t+1} T^T f_t \]

\[ O_{t+1} f_{t+1} = \alpha T f_t \]

\[ \alpha (T^T)^{-1} O_{t+1} f_{t+1} = f_t \]

Algorithm: forward pass computes \( f_t \), backward pass does \( f_{t+1}, b_t \)

Kalman filters

Modelling systems described by a set of continuous variables, e.g., tracking a bird flying—\( X_t = X, Y, Z, \dot{X}, \dot{Y}, \dot{Z} \).

Airplanes, robots, ecosystems, economies, chemical plants, planets, …

\[ \dot{X}_t = X_{t+1} \]

\[ Z_t \]

Gaussian prior, linear Gaussian transition model and sensor model
Gaussian distributions
Prediction step: if $P(X_{t|e_1:t})$ is Gaussian, then prediction
$$P(X_{t+1|e_1:t}) = \int P(x_{t|e_1:t}) \, dx_t$$
- Computes transition as locally linear around $x_t = \mu_t$
- Fails if systems is locally unsmooth

Chapter 15, Sections 1–5

Non-Gaussian)

Transition and sensor models:
$$P(x_{t+1|x_t}) = N(Fx_t, \Sigma_{x_t|x_t})$$
$$P(z_t|x_t) = N(Hx_t, \Sigma_{z_t|x_t})$$

$F$ is the matrix for the transition; $\Sigma_x$ the transition noise covariance
$H$ is the matrix for the sensors; $\Sigma_z$ the sensor noise covariance

Filter computes the following update:
$$\mu_{t+1} = F\mu_t + K_{t+1}(z_{t+1} - HF\mu_t)$$
$$\Sigma_{t+1} = (I - K_{t+1}H)(F\Sigma_x F^T + \Sigma_z)$$

where $K_{t+1} = (F\Sigma_x F^T + \Sigma_z H^T(H(F\Sigma_x F^T + \Sigma_z) H^T + \Sigma_z)^{-1}$

is the Kalman gain matrix
$\Sigma_z$ and $K_t$ are independent of observation sequence, so compute offline
DBNs vs. HMMs

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

\[ X_0 \rightarrow X_1 \rightarrow X_2 \]
\[ Y_0 \rightarrow Y_1 \rightarrow Y_2 \]
\[ Z_0 \rightarrow Z_1 \rightarrow Z_2 \]

Sparse dependencies ⇒ exponentially fewer parameters;
- e.g., 20 state variables, three parents each
- DBN has \( 20 \times 2^3 = 160 \) parameters, HMM has \( 2^{20} \times 2^3 \approx 10^{12} \)

Likelihood weighting for DBNs

Set of weighted samples approximates the belief state

\[ \text{LW samples pay no attention to the evidence!} \]
\[ \Rightarrow \text{fraction “agreeing” falls exponentially with } f \]
\[ \Rightarrow \text{number of samples required grows exponentially with } f \]

DBNs vs Kalman filters

Every Kalman filter model is a DBN, but few DBNs are KFs;
- real world requires non-Gaussian posteriors

E.g., where are bin Laden and my keys? What’s the battery charge?

\[ \text{BMBroken} \rightarrow \text{BMBroken} \rightarrow \text{Battery} \]
\[ \text{Battery} \rightarrow X_0 \rightarrow X_1 \rightarrow X_2 \]
\[ \text{Battery} \rightarrow Z_0 \rightarrow Z_1 \rightarrow Z_2 \]

Particle filtering

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

Replicate particles proportional to likelihood for \( \theta_1 \)

\[ \begin{array}{c|cc|cc}
\text{true} & \text{Rain}_1 & \text{Rain}_3 & \text{Rain}_5 & \text{Rain}_n \\
\text{false} & \ast & \ast & \ast & \ast \\
\end{array} \]

(a) Propagate \hspace{1cm} (b) Weight \hspace{1cm} (c) Resample

Widely used for tracking nonlinear systems, esp. in vision
- Also used for simultaneous localization and mapping in mobile robots
- \( 10^7 \)-dimensional state space
Particle filtering performance

Approximation error of particle filtering remains bounded over time, at least empirically—theoretical analysis is difficult

Summary

Temporal models use state and sensor variables replicated over time

Markov assumptions and stationarity assumption, so we need
- transition model \( P(X_t|X_{t-1}) \)
- sensor model \( P(E_t|X_t) \)

Tasks are filtering, prediction, smoothing, most likely sequence;
all done recursively with constant cost per time step

Hidden Markov models have a single discrete state variable; used for speech recognition

Kalman filters allow \( n \) state variables, linear Gaussian, \( O(n^3) \) update

Dynamic Bayes nets subsume HMMs; Kalman filters; exact update intractable

Particle filtering is a good approximate filtering algorithm for DBNs