Lecture 8 - Reasoning under Uncertainty (Part II)

Jesse Hoey
School of Computer Science
University of Waterloo

June 2, 2022

Readings: Poole & Mackworth (2nd ed.) Chapt. 8.5 - 8.9

Probability and Time

- A node repeats over time
- explicit encoding of time
- chain has length = amount of time you want to model
- event-driven times or clock-driven times
- e.g. Markov chain

![Markov chain diagram]

Markov assumption

\[ P(S_{t+1}|S_1, \ldots, S_t) = P(S_{t+1}|S_t) \]

This distribution gives the dynamics of the Markov chain

Hidden Markov Models (HMMs)

Add: observations \( O_t \) (always observed, so the node is square) and observation function \( P(O_t|S_t) \)

Given a sequence of observations \( O_1, \ldots, O_t \), can estimate filtering:

\[ P(S_t|O_1, \ldots, O_t) \]

or smoothing, for \( k < t \)

\[ P(S_k|O_1, \ldots, O_t) \]
Speech Recognition

- Most well-known application of HMMs
- Observations: audio features
- States: phonemes
- Dynamics: models e.g. co-articulation
- HMMs: words
- Can build hierarchical models (e.g. sentences)

Belief Monitoring in HMMs

Filtering:

\[ \alpha_i = P(S_i | o_0, \ldots, o_i) \]
\[ \propto P(S_i, o_0, \ldots, o_i) \]
\[ = P(o_i | S_i) \sum_{S_{i-1}} P(S_i, S_{i-1}, o_0, \ldots, o_{i-1}) \]
\[ = P(o_i | S_i) \sum_{S_{i-1}} P(S_i | S_{i-1}) P(S_{i-1}, o_0, \ldots, o_{i-1}) \]
\[ \propto P(o_i | S_i) \sum_{S_{i-1}} P(S_i | S_{i-1}) \alpha_{i-1} \]

Smoothing:

\[ \beta_{i+1} = P(o_{i+1}, \ldots, o_T | S_i) \]
\[ = \sum_{S_{i+1}} P(S_{i+1}, o_{i+1}, \ldots, o_T | S_i) \]
\[ = \sum_{S_{i+1}} P(o_{i+1} | S_{i+1}, o_{i+2}, \ldots, o_T, S_i) P(S_{i+1}, o_{i+2}, \ldots, o_T | S_i) \]
\[ = \sum_{S_{i+1}} P(o_{i+1} | S_{i+1}) P(o_{i+2}, \ldots, o_T | S_{i+1}, S_i) P(S_{i+1} | S_i) \]
\[ = \sum_{S_{i+1}} P(o_{i+1} | S_{i+1}) P(S_{i+1} | S_i) \beta_{i+2} \]
Dynamic Bayesian Networks (DBNs)

- In general, any Bayesian network can repeat over time: DBN
- Many examples can be solved with variable elimination, may become too complex with enough variables
- Event-driven times or clock-driven times

Example: localization

Suppose a robot wants to determine its location based on its actions and its sensor readings: Localization

- This can be represented by the augmented HMM:

Example localization domain

- Circular corridor, with 16 locations:
- Doors at positions: 2, 4, 7, 11.
- Noisy Sensors
- Stochastic Dynamics
- Robot starts at an unknown location and must determine where it is, known as the kidnapped robot problem.
- See handout robotloc.pdf

Example Sensor Model

- \[ P(\text{Observe Door} \mid \text{At Door}) = 0.8 \]
- \[ P(\text{Observe Door} \mid \text{Not At Door}) = 0.1 \]
Example Dynamics Model

- \( P(\text{Loc}_{t+1} = l | \text{Action}_t = \text{goRight} \land \text{Loc}_t = l) = 0.1 \)
- \( P(\text{Loc}_{t+1} = l + 1 | \text{Action}_t = \text{goRight} \land \text{Loc}_t = l) = 0.8 \)
- \( P(\text{Loc}_{t+1} = l + 2 | \text{Action}_t = \text{goRight} \land \text{Loc}_t = l) = 0.074 \)
- \( P(\text{Loc}_{t+1} = l' | \text{Action}_t = \text{goRight} \land \text{Loc}_t = l) = 0.002 \) for any other location \( l' \).
  
  ▶ All location arithmetic is modulo 16.
  
  ▶ The action \text{goLeft} works the same but to the left.

Example sequence

- observe door, go right, observe no door, go right, observe door
- where is the robot?

\[
P(\text{Loc}_2 = 4 | O_0 = d, A_0 = r, O_1 = \neg d, A_1 = r, O_2 = d) = 0.42
\]

Combining sensor information

- **Example:** we can combine information from a light sensor and the door sensor **Sensor Fusion**
- **Key Point:** Bayesian probability ensures that evidence is integrated **proportionally to its precision**.
- **Sensors are** **precisely weighted**

Example: user modeling


Figure 2: A portion of a Bayesian user model for inferring the likelihood that a user needs assistance, considering profile information as well as observations of recent activity.

### Stochastic Simulation

- **Idea:** probabilities ↔ samples
- Get probabilities from samples:

<table>
<thead>
<tr>
<th>$X$</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$n_1$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x_k$</td>
<td>$n_k$</td>
</tr>
<tr>
<td>total</td>
<td>$m$</td>
</tr>
</tbody>
</table>

$$\leftrightarrow$$

<table>
<thead>
<tr>
<th>$X$</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$n_1/m$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x_k$</td>
<td>$n_k/m$</td>
</tr>
</tbody>
</table>

- If we could sample from a variable’s (posterior) probability, we could estimate its (posterior) probability.

### Generating samples from a distribution

For a variable $X$ with a discrete domain or a (one-dimensional) real domain:

- **Totally order** the values of the domain of $X$.
- Generate the **cumulative probability distribution**:
  $$f(x) = P(X \leq x).$$
- Select a value $y$ uniformly in the range $[0, 1]$.
- Select the $x$ such that $f(x) = y$.

### Hoeffding Bound

$p$ is true probability, $s$ is sample average, $n$ is number of samples

$$P(|s - p| > \epsilon) \leq 2e^{-2n\epsilon^2}$$

- If we want an error greater than $\epsilon$ in less than a fraction $\delta$ of the cases, solve for $n$:

$$2e^{-2n\epsilon^2} < \delta$$

$$n > -\frac{\ln\delta}{2\epsilon^2}$$

- We have

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>cases with error $&gt; \epsilon$</th>
<th>samples needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1/20</td>
<td>184</td>
</tr>
<tr>
<td>0.01</td>
<td>1/20</td>
<td>18,445</td>
</tr>
<tr>
<td>0.1</td>
<td>1/100</td>
<td>265</td>
</tr>
</tbody>
</table>
Forward sampling in a belief network

- Sample the variables **one at a time**;
- Sample **parents** of $X$ before you sample $X$.
- Given values for the parents of $X$, sample from the probability of $X$ given its parents.
- For samples $s_i, i = 1 \ldots N$:

$$P(X) \propto \sum_{s_i} \delta(x_i) = N_{x=x_i}$$

where

$$\delta(x_i) = \begin{cases} 1 & \text{if } X = x_i \text{ in } s_i \\ 0 & \text{otherwise} \end{cases}$$

Sampling for a belief network: inference

<table>
<thead>
<tr>
<th>Sample</th>
<th>Malfunction</th>
<th>Cancer</th>
<th>TestB</th>
<th>TestA</th>
<th>Report</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>$s_2$</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>$s_3$</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>$s_4$</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>$s_5$</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>$s_6$</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>$s_7$</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>$s_{1000}$</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>

To get $P(H = h_i | E = e_i)$ simply

- Count the number of samples that have $H = h_i$ and $E = e_i$, $N(h_i, e_i)$
- Divide by the number of samples that have $E = e_i$, $N(e_i)$
- $P(H = h_i | E = e_i) = \frac{P(H = h_i \land E = e_i)}{P(E = e_i)} = \frac{N(h_i, e_i)}{N(e_i)}$
- $P(C = True | Database = True)$ based on first 7 samples?

Forward Sampling

![Inference via sampling graph]

**Inference via sampling**

| $P(cancer|database)$ | 0.9 | 0.85 | 0.8 | 0.75 | 0.7 | 0.65 | 0.6 | 0.55 |
|----------------------|-----|------|-----|------|-----|------|-----|------|
| number of samples    | 0   | 500  | 1000| 1500 | 2000|      |     |      |

Rejection Sampling

- To estimate a posterior probability given evidence
  $Y_1 = v_1 \land \ldots \land Y_j = v_j$:
- If, for any $i$, a sample assigns $Y_i$ to any value other than $v_i$, reject that sample.
- The **non-rejected** samples are distributed according to the posterior probability.
- In the Hoeffding bound, $n$ is the number of **non-rejected samples**
Example Network

\[
P(A = \text{true}) = 0.4\quad A \rightarrow E \quad P(E = \text{true}) = 0.1
\]

If we draw \( N \) samples \( s_i=1...N \) by
- sampling \( A \): \( a_i=1...N \)
- sampling from \( E \) given \( A \): \( e_i=1...N \)

then
\[
N_t = 0.4N \text{ of them will have } A = \text{true}, \text{ and of these 10% will have } E = \text{true}
\]
\[
N_f = 0.6N \text{ of them will have } A = \text{false}, \text{ and of these 30% will have } E = \text{true}
\]

so we have
\[
A \quad E \quad N_{AE}
\]

\[
\begin{align*}
\text{true} & \quad \text{false} & \quad N_{tt} = 0.4 \times 0.9 \times N \\
\text{true} & \quad \text{true} & \quad N_{tt} = 0.4 \times 0.1 \times N \\
\text{false} & \quad \text{false} & \quad N_{ff} = 0.6 \times 0.7 \times N \\
\text{false} & \quad \text{true} & \quad N_{ft} = 0.6 \times 0.3 \times N
\end{align*}
\]

We want to compute
\[
P(a|e) = P(A = \text{true}|E = \text{true}) \propto \sum_i \delta(a_i = \text{true})\delta(e_i = \text{true})
\]

\[
P(a|e) = \frac{P(a \land e)}{P(e)} = \frac{N_{tt}}{N_{tt} + N_{ft}}
\]

\[
= \frac{0.1 \times 0.4 \times N}{0.1 \times 0.4 \times N + 0.3 \times 0.6 \times N} = 0.182
\]

Importance weights

- we can do better since we can weight the samples
- weights = prob. that the evidence is observed
- \( N_t \) samples with \( A = \text{true} \) have weight of \( w_t = 0.1 \)
  this is \( P(E = \text{true}|A = \text{true}) \)
- \( N_f \) samples with \( A = \text{false} \) have weight of \( w_f = 0.3 \)
  this is \( P(E = \text{true}|A = \text{false}) \)
- can do better because we don’t need to generate the 90%
  of samples (when \( A = \text{true} \)) that don’t agree with the
  evidence - we simply assign all samples a weight of 0.1
- thus, we are mixing exact inference (the 0.1) with sampling.
Importance weights

<table>
<thead>
<tr>
<th>$P(A = \text{true})$</th>
<th>0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \rightarrow E$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$P(E = \text{true})$</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$ = true</td>
<td></td>
</tr>
<tr>
<td>$false$</td>
<td>0.3</td>
</tr>
</tbody>
</table>

- In fact, the As don’t need to even be sampled from $P(A)$
- Can be sampled from some $q(A)$, say $q(A = \text{true}) = 0.5$
- and each sample will have a new weight $P(a)/q(a)$
- $q(A)$ is a proposal distribution.

- $N'_t = q(a)N$ samples with $A = \text{true}$ have weight of $0.1 \times \frac{P^*(a)}{q(a)} = 0.1 \times \frac{0.4}{0.5}$
- $N'_f = q(\overline{a})N$ samples with $A = \text{false}$ have weight of $0.3 \times \frac{P^*(\overline{a})}{q(\overline{a})} = 0.3 \times \frac{0.6}{0.5}$

- total weight of all samples with $A = \text{true}$

$$W'_t = \sum_i w_i \delta(a_i = \text{true}) = N'_t \times 0.1 \times \frac{0.4}{0.5} = 0.5N \times 0.1 \times \frac{0.4}{0.5} = 0.1 \times \alpha \times 0.4 \times N$$

- total weight of all samples with $A = \text{false}$

$$W'_f = \sum_i w_i \delta(a_i = \text{true}) = N'_f \times 0.3 \times \frac{0.6}{0.5} = 0.5N \times 0.3 \times \frac{0.6}{0.5} = 0.3 \times \alpha \times 0.6 \times N$$
Importance weights

\[
\begin{array}{c|c|c}
A & P(E = true) \\
\hline
\text{true} & 0.1 \\
\text{false} & 0.3 \\
\end{array}
\]

finally, compute

\[
P(a|e) = \frac{W_i}{W_t + W_i} = \frac{0.1 \times \alpha \times 0.4 \times N}{0.1 \times \alpha \times 0.4 \times N + 0.3 \times \alpha \times 0.6 \times N}
\]

Sometimes we may want to choose a proposal distribution that is **different** than the actual probability distribution.

We may want to **skew** the proposal - because we may have some **additional knowledge** about the data, for example.

or, we can generate proposals **from the data itself** using some procedural knowledge that is not directly encoded in the BN.

Can be important in multiple/many dimensions,

---

**Example Proposal Distributions**

Recall variable elimination: To compute

\[
P(Z, Y_1 = v_1, \ldots, Y_j = v_j),
\]

we sum out the other variables, \(Z_1, \ldots, Z_k = \{X_1, \ldots, X_n\} - \{Z\} - \{Y_1, \ldots, Y_j\}\).

\[
P(Z, Y_1 = v_1, \ldots, Y_j = v_j)
= \sum_{Z_1} \cdots \sum_{Z_k} \prod_{i=1}^n P(X_i|\text{parents}(X_i))Y_i = v_i, \ldots, Y_j = v_j
\]

Now, we sample \(Z_{i+1}, \ldots, Z_k\) and sum \(Z_1, \ldots, Z_i\),

\[
= \sum_{s_i = \{z_{i+1}, \ldots, z_k\}} \left[ \sum_{Z_1 \cdots Z_i} \prod_{i=1}^j P(Z|\text{parents}(Z_i))Y_i = v_i, \ldots, Y_j = v_j \right] \frac{P(Z_{i+1}, \ldots, Z_k, i)}{q(Z_{i+1}, \ldots, Z_k, i)}
\]

---

**Stochastic Sampling**

**Importance Sampling example**

Compute \(P(B|D = true, A = false)\) by sampling \(C\) and \(M\).

- use \(q(C = true) = P(C = true) = 0.32\)
  and \(q(M = true) = P(M = true) = 0.08\)
- use \(q(C = true) = 0.5\)
  and \(q(M = true) = P(M = true) = 0.08\)
- use \(q(C = true) = q(M = true) = 0.5\)

\[
P(B|D = true, A = false) \propto \sum_{s_i = \{c_i, m_i\}} P(B, D = true, A = false|c_i, m_i)
\]
Stochastic Sampling for HMMs (and other DBNS)

Sequential Monte Carlo or Particle Filter
- sequential stochastic sampling
- keep track of $P(S_t)$ at the current time $t$
- represent $P(S_t)$ with a set of samples
- update as new observations $o_{t+1}$ arrive
  1. predict $P(S_{t+1}) \propto P(S_{t+1} | S_t)$
  2. compute weights as $P(o_{t+1} | S_{t+1})$
  3. resample according to weights

Particle Filtering

step 0: prior belief
- sample $i$: $\{x_i, w_i\}$

step 1: hypothesis (prediction)
- sample $i$: $\{x_i, w_i\}$
Particle Filtering

**step 2: stochastic spreading**

sample i: \( \{x_i, w_i\} \)

**step 3: evidence**

sample i: \( \{x_i, w_i\} \)

Particle Filtering

**step 4: resample**

sample i: \( \{x_i, w_i\} \)

Bayesian Sequential Updates

predict

spread

evidence
Resampling

- avoids degeneracies in the samples
- all importance weights → 0 except one
- performance of the algorithm depends on the resampling method

Next:

- Supervised Learning under Uncertainty (Poole & Mackworth (2nd ed.)) 10.1, 10.4