CS 188: Artificial Intelligence
Fall 2008

Lecture 18: Decision Diagrams
10/30/2008

Dan Klein – UC Berkeley
Announcements

- P4 EXTENDED to Tuesday 11/4

- Midterms graded, pick up after lecture

- Midterm course evaluation up on web soon, please fill out!

- Final contest instructions out today!
  - Prizes will be good ☺
Sampling

- **Basic idea:**
  - Draw $N$ samples from a sampling distribution $S$
  - Compute an approximate posterior probability
  - Show this converges to the true probability $P$

- **Outline:**
  - Sampling from an empty network
  - Rejection sampling: reject samples disagreeing with evidence
  - Likelihood weighting: use evidence to weight samples
Prior Sampling

C | P(S|C)
---|---
T | .10
F | .50

| C | P(R|C) |
---|---|
| T | .80 |
| F | .20 |

| S | R | P(W|S,R) |
---|---|---|
| T | T | .99 |
| T | F | .90 |
| F | T | .90 |
| F | F | .01 |
Rejection Sampling

- Let’s say we want $P(C)$
  - No point keeping all samples around
  - Just tally counts of $C$ outcomes
- Let’s say we want $P(C|s)$
  - Same thing: tally $C$ outcomes, but ignore (reject) samples which don’t have $S=s$
  - This is rejection sampling
  - It is also consistent for conditional probabilities (i.e., correct in the limit)
Likelihood Weighting

- Problem with rejection sampling:
  - If evidence is unlikely, you reject a lot of samples
  - You don’t exploit your evidence as you sample
  - Consider $P(B|a)$

- Idea: fix evidence variables and sample the rest

- Problem: sample distribution not consistent!
- Solution: weight by probability of evidence given parents
Likelihood Sampling

\[ w = 1.0 \times 0.1 \times 0.99 \]

\[ \begin{array}{|c|c|} \hline C & P(S|C) \\ \hline T & 0.10 \\ F & 0.50 \\ \hline \end{array} \]

\[ \begin{array}{|c|c|} \hline C & P(R|C) \\ \hline T & 0.80 \\ F & 0.20 \\ \hline \end{array} \]

\[ \begin{array}{|c|c|c|} \hline S & R & P(W|S,R) \\ \hline T & T & 0.99 \\ T & F & 0.90 \\ F & T & 0.90 \\ F & F & 0.01 \\ \hline \end{array} \]
Likelihood Weighting

- Sampling distribution if $z$ sampled and $e$ fixed evidence

$$S_{WS}(z, e) = \prod_{i=1}^{l} P(z_i|\text{Parents}(Z_i))$$

- Now, samples have weights

$$w(z, e) = \prod_{i=1}^{m} P(e_i|\text{Parents}(E_i))$$

- Together, weighted sampling distribution is consistent

$$S_{WS}(z, e)w(z, e) = \prod_{i=1}^{m} P(e_i|\text{Parents}(E_i)) \prod_{i=1}^{m} P(e_i|\text{Parents}(E_i))$$

$$= P(z, e)$$
Likelihood Weighting

- Note that likelihood weighting doesn’t solve all our problems
- Rare evidence is taken into account for downstream variables, but not upstream ones
- A better solution is Markov-chain Monte Carlo (MCMC), more advanced
- We’ll return to sampling for robot localization and tracking in dynamic BNs
Pacman Contest
Recap: Inference Example

- Find $P(W|F=\text{bad})$
  - Restrict all factors
  - No hidden vars to eliminate (this time!)
  - Just join and normalize

$P(W) \quad P(\text{bad}|W)$

$$P(W, \text{bad}) = P(W) \times P(\text{bad}|W)$$

$$P(W|F = \text{bad})$$
Decision Networks

- **MEU**: choose the action which maximizes the expected utility given the evidence

- Can directly operationalize this with decision diagrams
  - Bayes nets with nodes for utility and actions
  - Lets us calculate the expected utility for each action

- **New node types**:
  - Chance nodes (just like BNs)
  - Actions (rectangles, must be parents, act as observed evidence)
  - Utilities (depend on action and chance nodes)
Decision Networks

- **Action selection:**
  - Instantiate all evidence
  - Calculate posterior over parents of utility node
  - Set action node each possible way
  - Calculate expected utility for each action
  - Choose maximizing action

![Diagram of decision network with nodes for Weather, Forecast, Umbrella, and U.](attachment:decision_network_diagram.png)
Example: Decision Networks

\[ EU(\text{leave}) = \sum_w P(w)U(\text{leave}, w) = 0.7 \cdot 100 + 0.3 \cdot 0 = 70 \]

\[ EU(\text{take}) = \sum_w P(w)U(\text{take}, w) = 0.7 \cdot 20 + 0.3 \cdot 70 = 35 \]

Optimal decision = leave

\[ MEU(\emptyset) = \max_a EU(a) = 70 \]
Example: Decision Networks

Umbrella = leave

\[ EU(\text{leave}|\text{bad}) = \sum_{w} P(w|\text{bad})U(\text{bad}, w) \]
\[ = 0.34 \cdot 100 + 0.66 \cdot 0 = 34 \]

Umbrella = take

\[ EU(\text{take}|\text{bad}) = \sum_{w} P(w|\text{bad})U(\text{take}, w) \]
\[ = 0.34 \cdot 20 + 0.66 \cdot 70 = 53 \]

Optimal decision = take

\[ MEU(F = \text{bad}) = \max_{a} EU(a|\text{bad}) = 53 \]
Value of Information

- **Idea:** compute value of acquiring each possible piece of evidence
  - Can be done directly from decision network

- **Example:** buying oil drilling rights
  - Two blocks A and B, exactly one has oil, worth k
  - Prior probabilities 0.5 each, mutually exclusive
  - Current price of each block is k/2
  - MEU = 0 (either action is a maximizer)

- **Solution:** compute value of information
  - = expected gain in MEU from observing new information

- **Probe gives accurate survey of A. Fair price?**
  - Survey may say “oil in a” or “oil in b,” prob 0.5 each
  - If we know O, MEU is k/2 (either way)
  - Gain in MEU?
  - VPI(O) = k/2
  - Fair price: k/2
Value of Information

- Current evidence $E = e$, utility depends on $S = s$
  \[ \text{MEU}(e) = \max_a \sum_s P(s|e) \, U(s, a) \]

- Potential new evidence $E'$: suppose we knew $E' = e'$
  \[ \text{MEU}(e, e') = \max_a \sum_s P(s|e, e') \, U(s, a) \]

- **BUT** $E'$ is a random variable whose value is currently unknown, so:
  - Must compute expected gain over all possible values

\[ \text{VPI}_e(E') = \sum_{e'} P(e'|e) \left( \text{MEU}(e, e') - \text{MEU}(e) \right) \]

- ($\text{VPI} = \text{value of perfect information}$)
VPI Example

MEU with no evidence

\[ \text{MEU}(\emptyset) = \max_a \text{EU}(a) = 70 \]

MEU if forecast is bad

\[ \text{MEU}(F = \text{bad}) = \max_a \text{EU}(a|\text{bad}) = 53 \]

MEU if forecast is good

\[ \text{MEU}(F = \text{good}) = \max_a \text{EU}(a|\text{good}) = 95 \]

Forecast distribution

<table>
<thead>
<tr>
<th>F</th>
<th>P(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>0.59</td>
</tr>
<tr>
<td>bad</td>
<td>0.41</td>
</tr>
</tbody>
</table>

\[
0.59 \cdot (95 - 70) + 0.41 \cdot (53 - 70) = 22
\]

\[
0.59 \cdot (+25) + 0.41 \cdot (-17) = 22
\]
VPI Properties

- Nonnegative in expectation

\[ \forall E', e : \text{VPI}_e(E') \geq 0 \]

- Nonadditive --- consider, e.g., obtaining \( E_j \) twice

\[ \text{VPI}_e(E_j, F_k) \neq \text{VPI}_e(E_j) + \text{VPI}_e(F_k) \]

- Order-independent

\[
\begin{align*}
\text{VPI}_e(E_j, E_k) &= \text{VPI}_e(E_j) + \text{VPI}_{e,E_j}(E_k) \\
&= \text{VPI}_e(E_k) + \text{VPI}_{e,E_k}(E_j)
\end{align*}
\]
VPI Scenarios

- Imagine actions 1 and 2, for which $U_1 > U_2$
- How much will information about $E_j$ be worth?

Little – we’re sure action 1 is better.

A lot – either could be much better

Little – info likely to change our action but not our utility
Reasoning over Time

- Often, we want to reason about a sequence of observations
  - Speech recognition
  - Robot localization
  - User attention
  - Medical monitoring

- Need to introduce time into our models
- Basic approach: hidden Markov models (HMMs)
- More general: dynamic Bayes’ nets
Markov Models

- A Markov model is a chain-structured BN
  - Each node is identically distributed (stationarity)
  - Value of $X$ at a given time is called the state
  - As a BN:

    $X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4 \rightarrow \cdots$

    $P(X_1) \quad P(X|X_{-1})$

- Parameters: called transition probabilities or dynamics, specify how the state evolves over time (also, initial probs)

[DEMO: Battleship]
Conditional Independence

- **Basic conditional independence:**
  - Past and future independent of the present
  - Each time step only depends on the previous
  - This is called the (first order) Markov property

- **Note that the chain is just a (growing) BN**
  - We can always use generic BN reasoning on it (if we truncate the chain)
Example: Markov Chain

- **Weather:**
  - States: \( X = \{\text{rain, sun}\} \)
  - Transitions:

- Initial distribution: 1.0 sun
- What’s the probability distribution after one step?

\[
P(X_2 = \text{sun}) = P(X_2 = \text{sun}|X_1 = \text{sun})P(X_1 = \text{sun}) + P(X_2 = \text{sun}|X_1 = \text{rain})P(X_1 = \text{rain})
\]

\[
0.9 \cdot 1.0 + 0.1 \cdot 0.0 = 0.9
\]
Mini-Forward Algorithm

- Question: probability of being in state x at time t?
- Slow answer:
  - Enumerate all sequences of length t which end in s
  - Add up their probabilities

\[
P(X_t = \text{sun}) = \sum_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, \text{sun})
\]

\[
P(X_1 = \text{sun})P(X_2 = \text{sun}|X_1 = \text{sun})P(X_3 = \text{sun}|X_2 = \text{sun})P(X_4 = \text{sun}|X_3 = \text{sun})
\]

\[
P(X_1 = \text{sun})P(X_2 = \text{rain}|X_1 = \text{sun})P(X_3 = \text{sun}|X_2 = \text{rain})P(X_4 = \text{sun}|X_3 = \text{sun})
\]

\[
\vdots
\]

\[
\vdots
\]
Mini-Forward Algorithm

- Better way: cached incremental belief updates
  - An instance of variable elimination!

\[
P(x_t) = \sum_{x_{t-1}} P(x_t|x_{t-1})P(x_{t-1})
\]

\[
P(x_1) = \text{known}
\]

Forward simulation
Example

- From initial observation of sun

\[
\begin{pmatrix}
1.0 \\
0.0
\end{pmatrix}
\begin{pmatrix}
0.9 \\
0.1
\end{pmatrix}
\begin{pmatrix}
0.82 \\
0.18
\end{pmatrix}
\begin{pmatrix}
0.5 \\
0.5
\end{pmatrix}
\]

\[P(X_1) \quad P(X_2) \quad P(X_3) \quad P(X_\infty)\]

- From initial observation of rain

\[
\begin{pmatrix}
1.0 \\
0.0
\end{pmatrix}
\begin{pmatrix}
0.1 \\
0.9
\end{pmatrix}
\begin{pmatrix}
0.18 \\
0.82
\end{pmatrix}
\begin{pmatrix}
0.5 \\
0.5
\end{pmatrix}
\]

\[P(X_1) \quad P(X_2) \quad P(X_3) \quad P(X_\infty)\]
Stationary Distributions

- If we simulate the chain long enough:
  - What happens?
  - Uncertainty accumulates
  - Eventually, we have no idea what the state is!

- Stationary distributions:
  - For most chains, the distribution we end up in is independent of the initial distribution
  - Called the stationary distribution of the chain
  - Usually, can only predict a short time out

[DEMO: Battleship]
Web Link Analysis

- **PageRank over a web graph**
  - Each web page is a state
  - Initial distribution: uniform over pages
  - Transitions:
    - With prob. c, uniform jump to a random page (dotted lines)
    - With prob. 1-c, follow a random outlink (solid lines)

- **Stationary distribution**
  - Will spend more time on highly reachable pages
  - E.g. many ways to get to the Acrobat Reader download page!
  - Somewhat robust to link spam
  - Google 1.0 returned the set of pages containing all your keywords in decreasing rank, now all search engines use link analysis along with many other factors
Most Likely Explanation

- Question: most likely sequence ending in x at t?
  - E.g. if sun on day 4, what’s the most likely sequence?
  - Intuitively: probably sun all four days
- Slow answer: enumerate and score

\[
P(X_t = \text{sun}) = \max_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, \text{sun})
\]

\[
P(X_1 = \text{sun})P(X_2 = \text{sun}|X_1 = \text{sun})P(X_3 = \text{sun}|X_2 = \text{sun})P(X_4 = \text{sun}|X_3 = \text{sun})
\]

\[
P(X_1 = \text{sun})P(X_2 = \text{rain}|X_1 = \text{sun})P(X_3 = \text{sun}|X_2 = \text{rain})P(X_4 = \text{sun}|X_3 = \text{sun})
\]

\[
\vdots
\]
Mini-Viterbi Algorithm

- Better answer: cached incremental updates

- Define: \[ m_t[x] = \max_{x_{1:t-1}} P(x_{1:t-1}, x) \]
  \[ a_t[x] = \text{arg max}_{x_{1:t-1}} P(x_{1:t-1}, x) \]

- Read best sequence off of \( m \) and \( a \) vectors
Mini-Viterbi

\[ m_t[x] = \max_{x_{1:t-1}} P(x_{1:t-1}, x) \]

\[ = \max_{x_{1:t-1}} P(x_{1:t-1}) P(x|x_{t-1}) \]

\[ = \max_{x_{t-1}} P(x_t|x_{t-1}) \max_{x_{1:t-2}} P(x_{1:t-1}) \]

\[ = \max_{x_{t-1}} P(x_t|x_{t-1}) m_{t-1}[x] \]

\[ m_1[x] = P(x_1) \]