Inference

- Inference: calculating some statistic from a joint probability distribution
- Examples:
  - Posterior probability:
    \[ P(Q|E_1 = e_1, \ldots, E_k = e_k) \]
  - Most likely explanation:
    \[ \arg\max_q P(Q = q|E_1 = e_1, \ldots) \]

Reminder: Alarm Network

Inference by Enumeration

- Given unlimited time, inference in BNs is easy
- Recipe:
  - State the marginal probabilities you need
  - Figure out ALL the atomic probabilities you need
  - Calculate and combine them
- Example:
  \[ P(b, j, m) = \frac{P(b, j, m) P(j, m)}{P(j, m)} \]
  \[ P(b, j, m) = P(b, e, a, j, m) + \sum_{e \neq a} P(b, e, a, j, m) \]

Example

\[ P(b, j, m) = \frac{P(b, j, m) P(j, m)}{P(j, m)} \]

Where did we use the BN structure?

We didn’t!
Example

- In this simple method, we only need the BN to synthesize the joint entries

\[
P(b, j, m) = \frac{P(b) P(e) P(a | b, e) P(j | a) P(m | a) +}{P(b) P(e) P(\bar{a} | b, e) P(j | \bar{a}) P(m | \bar{a}) +} \frac{P(b) P(e) P(\bar{a} | b, \bar{e}) P(j | a) P(m | a) +}{P(b) P(e) P(a | b, \bar{e}) P(j | \bar{a}) P(m | \bar{a}) +}
\]

Normalization Trick

\[
P(B | j, m) = \frac{P(B, j, m)}{P(j, m)}
\]

\[
P(b, j, m) = \sum_{e, a} P(b, e, a, j, m)
\]

\[
P(\bar{b}, \bar{a}, j, m) = \sum_{e, a} P(\bar{b}, e, a, j, m)
\]

\[
\begin{align*}
P(b, j, m) \\
P(\bar{b}, j, m)
\end{align*}
\]

Normalize \[ P(b[j, m], P(b[j, m]) \]

Inference by Enumeration?

- Atomic inference is extremely slow!
- Slightly clever way to save work:
  - Move the sums as far right as possible
  - Example:

\[
P(b, j, m) = \sum_{e, a} P(b, e, a, j, m)
\]

\[
= \sum_{e, a} P(b) P(e) P(a | b, e) P(j | a) P(m | a)
\]

\[
= P(b) \sum_{e} P(e) \sum_{a} P(a | b, e) P(j | a) P(m | a)
\]

Nesting Sums

- View the nested sums as a computation tree:

- Still repeated work: calculate P(m | a) P(j | a) twice, etc.

Evaluation Tree

Variable Elimination: Idea

- Lots of redundant work in the computation tree
- We can save time if we cache all partial results
- This is the basic idea behind variable elimination
### Basic Objects
- Track objects called \textit{factors}
- Initial factors are local CPTs
  \[
P(B), \quad P(A | B), \quad P(E | B, A)
  \]
- During elimination, create new factors
- Anatomy of a factor:
  \[
  f_{AB}(D, E) \quad \text{variables introduced, summed out}
  \]
  \[
  4 \text{ numbers, one for each value of } D \text{ and } E
  \]

### Basic Operations
- First basic operation: \textit{join factors}
  - Combining two factors:
    - \textit{Just like a database join}
    - \textit{Build a factor over the union of the domains}
  - Example:
    \[
    f_1(A, B, C) \times f_2(B, C) \rightarrow f_3(A, B, C)
    \]
    \[
    f_3(a, b, c) = f_1(a, b) \cdot f_2(b, c)
    \]
    \[
    P(a, b | c) = P(a | b) \cdot P(b | c)
    \]

### Example
\[
P(b, j, m)
\]
\[
= f_B(b) \sum_c f_E(c) f_{AEJM}(b, c)
\]
\[
= f_B(b) \sum_c f_{AEJM}(b, e)
\]
\[
= f_B(b) f_{AEJM}(b)
\]
\[
= f_{ABEJM}(b)
\]

### Variable Elimination
- What you need to know:
  - VE caches intermediate computations
  - Polynomial time for tree-structured graphs!
  - Saves time by marginalizing variables as soon as possible rather than at the end
- We will see special cases of VE later
  - You’ll have to implement the special cases
- Approximations
  - Exact inference is slow, especially when you have a lot of hidden nodes
  - Approximate methods give you a \textit{(close) answer, faster}
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

- This process generates samples with probability
  \[ S_{PS}(x_1 \ldots x_n) = \prod_{i=1}^{k} P(x_i|\text{Parents}(X_i)) = P(x_1 \ldots x_n), \]
  ...i.e. the BN's joint probability

- Let the number of samples of an event be \( N_{PS}(x_1 \ldots x_n) \).

- Then \( \lim_{N \to \infty} P(x_1, \ldots, x_n) = \lim_{N \to \infty} N_{PS}(x_1, \ldots, x_n)/N = S_{PS}(x_1, \ldots, x_n) = P(x_1 \ldots x_n) \)

- I.e., the sampling procedure is consistent

Prior Sampling

- We'll get a bunch of samples from the BN:
  - \( c, \neg s, r, w \)
  - \( c, s, r, w \)
  - \( \neg c, s, r, \neg w \)
  - \( \neg c, s, \neg r, w \)

- If we want to know \( P(W) \):
  - We have counts \( \langle \neg w:4, \neg w:1 \rangle \)
  - Normalize to get \( P(W) = \langle w:0.8, \neg w:0.2 \rangle \)

- This will get closer to the true distribution with more samples

- What about \( P(C|\neg r) \)? \( P(C|\neg r, \neg w) \)?

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 Ss
  - This is rejection sampling

- It is also consistent (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

<table>
<thead>
<tr>
<th>C</th>
<th>P(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>.06</td>
</tr>
<tr>
<td>F</td>
<td>.50</td>
</tr>
</tbody>
</table>

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

\[ w = 1/(0.6 \cdot 0.1 \cdot 0.1) = 10 \]

### Likelihood Weighting

- Sampling distribution if \( z \) sampled and \( e \) fixed evidence:
  \[ S_{W}(z,e) = \prod_{i=1}^{m} P(z_{i}|\text{Parents}(z_{i})) \]
- Now, samples have weights:
  \[ w(z,e) = \prod_{i=1}^{m} P(e_{i}|\text{Parents}(e_{i})) \]
- Together, weighted sampling distribution is consistent:
  \[ S_{W}(z,e)w(z,e) = \prod_{i=1}^{m} P(z_{i}|\text{Parents}(z_{i})) \prod_{i=1}^{m} P(e_{i}|\text{Parents}(e_{i})) \]
  \[ = P(z,e) \]

### 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.