Belief networks

Motivation

- Joint distribution $P(X_1=x_1, X_2=x_2, \ldots, X_n=x_n)$ involves $O(2^n)$ numbers for $n$ binary random variables
- More compact representations?
- More efficient algorithms for inference?

Alarm example

- Binary random variables $\{0, 1\}$
  - $B =$ burglary
  - $E =$ earthquake
  - $J =$ John calls
  - $M =$ Mary calls
  - $A =$ alarm
- Joint distribution
  $$P(B, E, A, J, M) = P(B)P(E|B)P(A|B, E)P(J|B, E, A)P(M|B, E, A, J)$$
  \[ \rightarrow \text{this is true in general, but can we simplify this with conditional independence assumptions?} \]
- Conditional independence
  $$P(B, E, A, J, M) = P(B)P(E)P(A|B, E)P(J|A)P(M|A)$$
  \[ \rightarrow \text{we can represent this as a DAG!} \]
**Directed Acyclic Graph (DAG)**

![DAG Diagram]

<table>
<thead>
<tr>
<th>B</th>
<th>E</th>
<th>P(E=1) = 0.002</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.001</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.29</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.94</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.95</td>
</tr>
</tbody>
</table>

- **Conditional Probability Table (CPT)**

| B | E | P(A=1 | B,E) |
|---|---|---------|
| 0 | 0 | 0.001 |
| 0 | 1 | 0.29 |
| 1 | 0 | 0.94 |
| 1 | 1 | 0.95 |

- **Joint probabilities**

\[
P(B=1, E=0, A=1, J=1, M=1) = P(B=1) P(E=0) P(A=1 | B=1, E=0) P(J=1 | A=1) P(M=1 | A=1) = 10.001 (1 - 0.002)...
\]

- Any "query" can be answered from joint distribution

  **Ex:** query \(P(B=1, E=0 | M=1)\)
  - product rule: \(P(B=1, E=0 | M=1) = \frac{P(B=1, E=0, M=1)}{P(M=1)}\)
  - From marginalization:
    \[
P(B=1, E=0, M=1) = \sum_{a \in \{0,1\}} \sum_{j \in \{0,1\}} P(B=1, E=0, M=1, A=a, J=j)
    \]
    \[
P(M=1) = \sum_{b, e, a, j} P(B=b, E=e, M=1, A=a, J=j)
    \]

- Q: "why may this not be efficient?"
  - A: "you're not exploiting structure in graph"

More efficient algorithms? Yes.

Exploit structure of DAGs.
Belief networks (BN)

A BN is a DAG in which
(i) nodes represent random variables
(ii) edges represent conditional dependences
(iii) CPTs describe how each node depends on parents

- Conditional independence
  - generally true that
    \[ P(X_1, ..., X_n) = P(X_1) P(X_2|X_1) P(X_3|X_1, X_2) ... P(X_n|X_1, ..., X_{n-1}) \]
    \[ = \prod_{i=1}^{n} P(X_i|X_1, X_2, ..., X_{i-1}) \]

- in a given domain, suppose that:
  \[ P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|\text{parents}(X_i)) \]
  where \( \text{parents}(X_i) \) is a subset of \( \{X_1, ..., X_{i-1}\} \)
  Big idea: represent independence relations by a DAG

- Constructing a BN
  (1) choose random variables
  (2) choose ordering
  (3) while there are variables left:
    (a) add node \( X_i \)
    (b) set \( \text{parents}(X_i) \) to minimal set satisfying (A)
    (c) define CPT \( P(X_i|\text{pa}(X_i)) \) parent configuration

- advantages
  - complete, compact, non-redundant, consistent
    representation of joint distribution
    Ex: for binary variables, \( O(n 2^k) \) with \( n \) nodes, if \( k \) is max # parents (in-degree) of graph
  - clean separation of qualitative vs quantitative knowledge
  - DAG encodes conditional independences
  - CPT encode numerical influence
- Node ordering
  - Best order is to add "root causes", then the variables they influence, and so on.
  - Wrong order: \{M, J, A, B, E\}

- From "misordered" graph, conditional independences in world not obvious
- More numbers needed to specify the same joint distribution
- Less natural, more difficult to assess CPTs or to learn CPTs from world experience/observation

- Representing CPTs
  - For simplicity, consider binary variables
  \[ X_i \in \{0,1\} \]
  \[ Y \in \{0,1\} \]

  How to represent CPT \( P(Y=1 \mid X_1, X_2, \ldots, X_k) \)?
  1. Lookup table: \( O(2^k) \) can store arbitrary CPT
  2. Deterministic node
     "AND" \( P(Y=1 \mid X_1, X_2, \ldots, X_k) = \prod_{i=1}^{k} X_i \)
     "OR" \( P(Y=0 \mid X_1, X_2, \ldots, X_k) = \prod_{i=1}^{k} (1-X_i) \)
  3. Noisy-OR node
     Use \( k \) numbers \( p_i \in [0,1] \) to parameterize \( O(2^k) \) in entire CPT.
     \( P(Y=0 \mid X_1, X_2, \ldots, X_k) = \prod_{i=1}^{k} (1-p_i) X_i \)
     \( P(Y=1 \mid X_1, X_2, \ldots, X_k) = 1 - \prod_{i=1}^{k} (1-p_i) X_i \)
Why called "noisy-OR"?

Look at $P(Y=1 \mid X_1=0, X_2=\ldots=X_k=0) = 1 - \prod_{i=1}^{k} (1-p_i) \neq 0$

Look at $P(Y=1 \mid X_i, X_1, \ldots, X_k)$ when one and only one parent is equal to one

$P(Y=1 \mid X_i=0, X_1=0, \ldots, X_{i-1}=0, X_{i+1}=1, X_{i+2}=0, \ldots, X_k=0)$

$= P(Y=1 \mid X_i=1, X_{j \neq i}=0)$

$= 1 - (1-p_i) \left( \prod_{j \neq i} (1-p_j) \right) = p_i$

Intuitively, $p_i$ is the probability that $X_i=1$ by itself triggers $Y=1$. 
