9/30

Motivation

* Joint distribution $P(X_1 = x_1, X_2 = x_2, X_3 = x_3, \cdots, X_n = x_n)$ involves $O(2^n)$ numbers for binary random variables.
* More compact representations?
  More efficient algorithms?

Example

* Binary random variables
  - B = burglary
  - E = earthquake
  - A = alarm
  - J = John calls
  - M = Mary calls
* Joint distribution

$$P(B, E, A, J, M) = P(B)\,P(E|B)\,P(A|E,B)\,P(J|A,E,B)\,P(M|J,A,E,B) \quad \text{product rule}$$

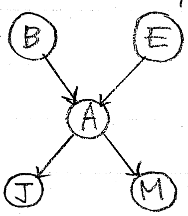
* Conditional independence

$$P(B, E, A, J, M) = P(B)\,P(E)\,P(A|E,B)\,P(J|A)\,P(M|A)$$

* Directed acyclic graph (DAG)

$P(B=1) = 0.001$

$P(E=1) = 0.002$



B	E	$P(A=1 \mid B, E)$
0	0	0.01
1	0	0.94
0	1	0.29
1	1	0.95

$P(J=1 \mid A=0) = 0.05$
$P(J=1 \mid A=1) = 0.9$

$P(M=1 \mid A=0) = 0.01$
$P(M=1 \mid A=1) = 0.7$

"conditional probability tables" (CPTs)

* Joint probability

$$P(B=1, E=0, A=1, J=1, M=1) = P(B=1)\,P(E=0)\,P(A=1|B=1, E=0)\,P(J=1|A=1)\,P(M=1|A=1)$$
$$= (0.001)\,(1-0.002)\,(0.94)\,(0.9)\,(0.7)$$

* Any "query" can be answered from joint distribution:

$$\text{e.g., } P(B=1, E=0 \mid M=1) \underset{\substack{\uparrow \\ \text{product rule}}}{=} \frac{P(B=1, E=0, M=1)}{P(M=1)} \underset{\text{marginalization}}{=} \frac{\sum_{a,j} P(B=1, E=0, M=1, A=a, J=j)}{\sum_{b,e,a',j'} P(M=1, B=b, E=e, A=a', J=j')}$$

* More efficient algorithms? Yes.
  Exploit structure of DAG (conditional independence)

$\boxed{\text{Belief networks}}$ (BNs)

A BN is a DAG in which:

(i) nodes represent random variables.

(ii) edges represent conditional dependencies.

(iii) CPTs describe how each node depends on its parents.

$\boxed{\text{BN = DAG + CPTs}}$

* Conditional independence

- Generally true that $P(X_1, X_2, \cdots, X_n) = P(X_1)\, P(X_2 | X_1) \cdots P(X_n | X_1, X_2, \cdots, X_{n-1})$

$$= \prod_{i=1}^{n} P(X_i \mid X_1, X_2, \cdots, X_{i-1}).$$

In any given domain, suppose that:

$P(X_1, X_2, \cdots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{parents}(X_i))$ (*) where parents (X_i) is a subset of $\{X_1, X_2, \cdots, X_{i-1}\}$. $\text{parents}(X_i) = pa(X_i) \subseteq \{X_1, X_2, \cdots, X_{i-1}\}$.

- Big idea: represent conditional dependences by DAG

$\boxed{\text{Constructing a BN}}$

(i) choose random variables

(ii) choose ordering

(iii) while there are variables left:

(a) add node X_i

(b) set parents (X_i) to minimal set satisfying (*)

(c) define CPT $P(X_i \mid pa(X_i))$.

* Advantages

- complete, consistent, compact, non-redundant representation of joint distribution.

ex: for binary variables, if k = max # parents of node in BN,

then $O(n \cdot 2^k)$ to represent joint distribution over $O(2^n)$ configuration.

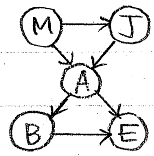
- clean separation of qualitative vs. quantitative knowledge

DAGs encode conditional independence
CPTs encode numerical influences

* Node ordering

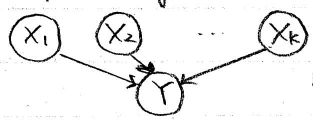
- Best ordering is to add "root causes" then variables they influence, and so on.
- Ex: wrong order $\{M, J, A, B, E\}$

$$P(M,J,A,B,E) = P(M)\, P(J|M)\, P(A|J,M)\, \underbrace{P(B|A,J,M)}_{P(B|A)}\, \underbrace{P(E|M,J,A,B)}_{P(E|A,B)}$$



- 2 extra edges in graph
- more numbers to fill in CPTs.
- more difficult CPTs to assess.

* Representing CPTs



for simplicity, binary variables. $X \in \{0,1\}$, $Y = \{0,1\}$

How to represent CPT $P(Y=1 \mid X_1, X_2, \cdots, X_k)$?

(i) lookup table $O(2^k)$ numbers store arbitrary CPT

(ii) deterministic node

"AND" $P(Y=1 \mid X_1, X_2, \cdots, X_k) = \prod_{i=1}^{k} X_i$

"OR" $P(Y=0 \mid X_1, X_2, \cdots, X_k) = \prod_{i=1}^{k} (1-X_i)$

(iii) noisy-OR CPT $Y \in \{0,1\}$, $X \in \{0,1\}$

Use k numbers $p_i \in [0,1]$ to parameterize 2^k entries in CPT.

$P(Y=0 \mid X_1, X_2, \cdots, X_k) = \prod_{i=1}^{k} (1-p_i)^{X_i}$

$P(Y=1 \mid X_1, X_2, \cdots, X_k) = 1 - \prod_{i=1}^{k} (1-p_i)^{X_i}$

Look at probability that $Y=1$ when exactly one parent is on:

$P(Y=1 \mid X_1 = X_2 = \cdots = X_{i-1} = 0,\ X_i = 1,\ X_{i+1} = \cdots = X_k = 0)$

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

$= 1 - (1-p_i)^1 \prod_{j \neq i} (1-p_j)^0$

$= p_i$

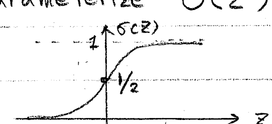
Intuitively, p_i is trigger probability that just $X_i = 1$ turns on $Y=1$.

Setting $p_i = 1$ for all i recovers OR-gate.

(iv) sigmoid CPT

Use k real numbers θ_i to parameterize $O(2^k)$ elements of CPT.

* let $\sigma(z) = 1 / (1+e^{-z})$: converts argument into $[0,1]$



* sigmoid CPT : $P(Y=1 \mid X_1, X_2, \cdots, X_k) = \sigma\left(\sum_{i=1}^{k} \theta_i X_i\right)$

* also known as logistic regression

" " " activation function (neural networks)

* if θ_i strongly negative, then $X_i = 1$ suppresses $Y=1$.

" " " positive, then $X_i = 1$ triggers $Y=1$.

* different than noisy-OR: sigmoid can mix inhibition and excitation.