AIFA: Reasoning Under Uncertainty - Inference

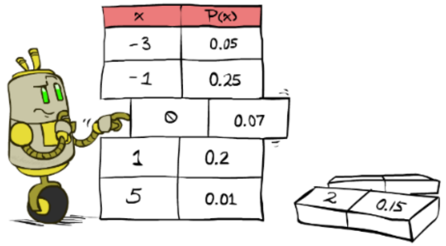
17/03/2025

Koustav Rudra

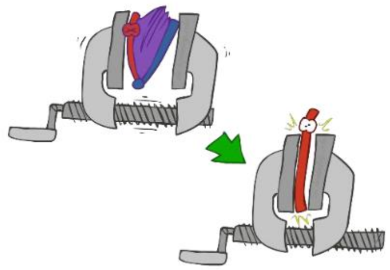
Inference by Enumeration

- General Case:
  - Evidence variables: $E_1, E_2, \dots E_k = e_1, e_2, \dots, e_k$
  - Query Variable: Q
  - Hidden variables: $H_1, H_2, \dots H_r$
  - $X_1, X_2, \dots X_n$ all variables
- We want: $P(Q | e_1, e_2, \dots, e_k)$

Step 1: Select the entries consistent with the evidence



Step 2: Sum out H to get joint of Query and evidence



$$P(Q, e_1, e_2, \dots, e_k) = \sum_{h_1, h_2, \dots, h_r} P(Q, h_1, h_2, \dots, h_r, e_1, e_2, \dots, e_k)$$

Step 3: Normalize

$$\times \frac{1}{Z}$$

$$Z = \sum_q P(Q, e_1, e_2, \dots, e_k)$$

$$P(Q | e_1, e_2, \dots, e_k) = \frac{1}{Z} P(Q, e_1, e_2, \dots, e_k)$$

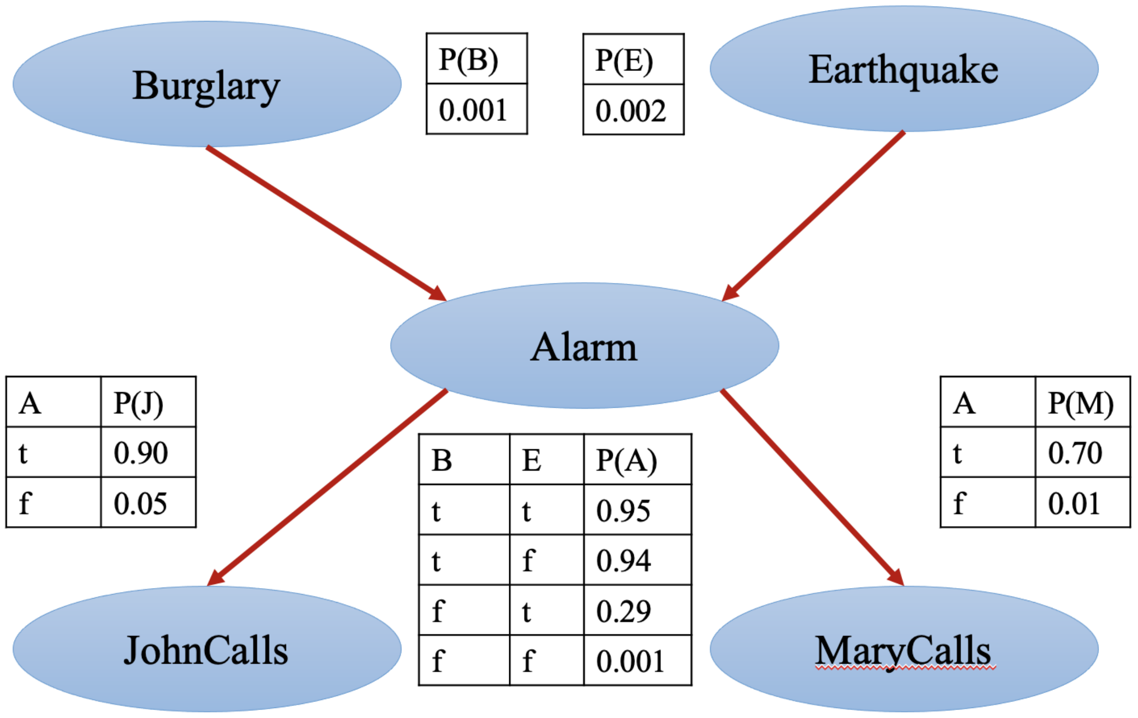
Inference using Belief Networks

$$P(B|J) = \frac{P(BJ)}{P(J)}$$
$$P(BJ) = P(BJA) + P(BJA')$$
$$P(BJA) = P(J|AB)P(AB) + P(J|A'B)P(A'B)$$

$$P(AB) = P(ABE) + P(ABE')$$
$$P(AB) = P(A|BE)P(BE) + P(A|BE')P(BE')$$
$$P(AB) = 0.00095$$

$$P(A'B) = 0.00005$$

$$P(BJ) = 0.90 * 0.00095 + 0.05 * 0.00005$$



P(B)
0.001

P(E)
0.002

B	E	P(A)
t	t	0.95
t	f	0.94
f	t	0.29
f	f	0.001

A	P(J)
t	0.90
f	0.05

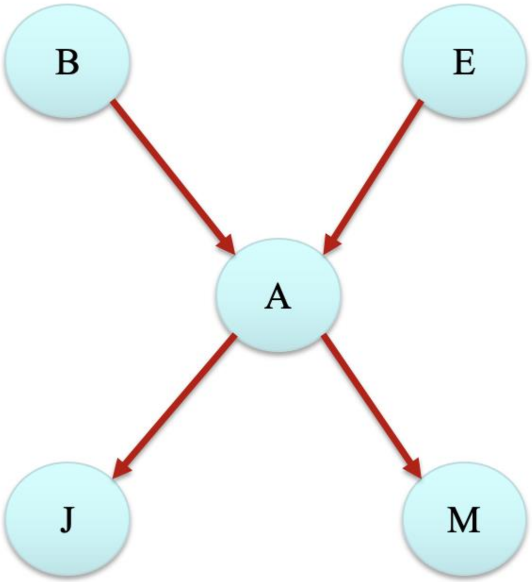
A	P(M)
t	0.70
f	0.01

Bayesian Network: Inference

- Compute posterior probability distribution of a set of query variables
  - Given some observed event [evidence variables]
  - Some unobserved events [Hidden variables]

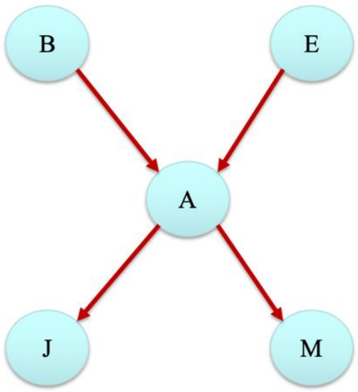
- P(B|J=True, M=True)

- Hidden variables:
  - Earthquake
  - Alarm



Bayesian Network: Inference by Enumeration

- P(B|J=True, M=True)



- Hidden variables:
  - Earthquake
  - Alarm

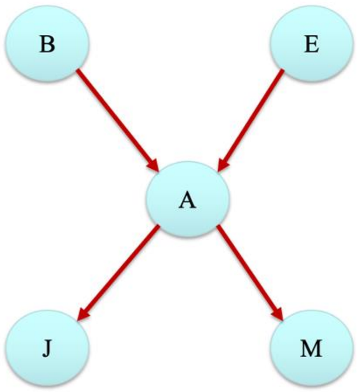
- P(b|j,m) = $\sum_e \sum_a P(b)P(e)P(a|b,e)P(j|a)P(m|a)$

- Add four terms each computed by multiplying five numbers

- If network contains n variables, complexity O($n2^n$)

Bayesian Network: Inference by Enumeration

- P(B|J=True, M=True)



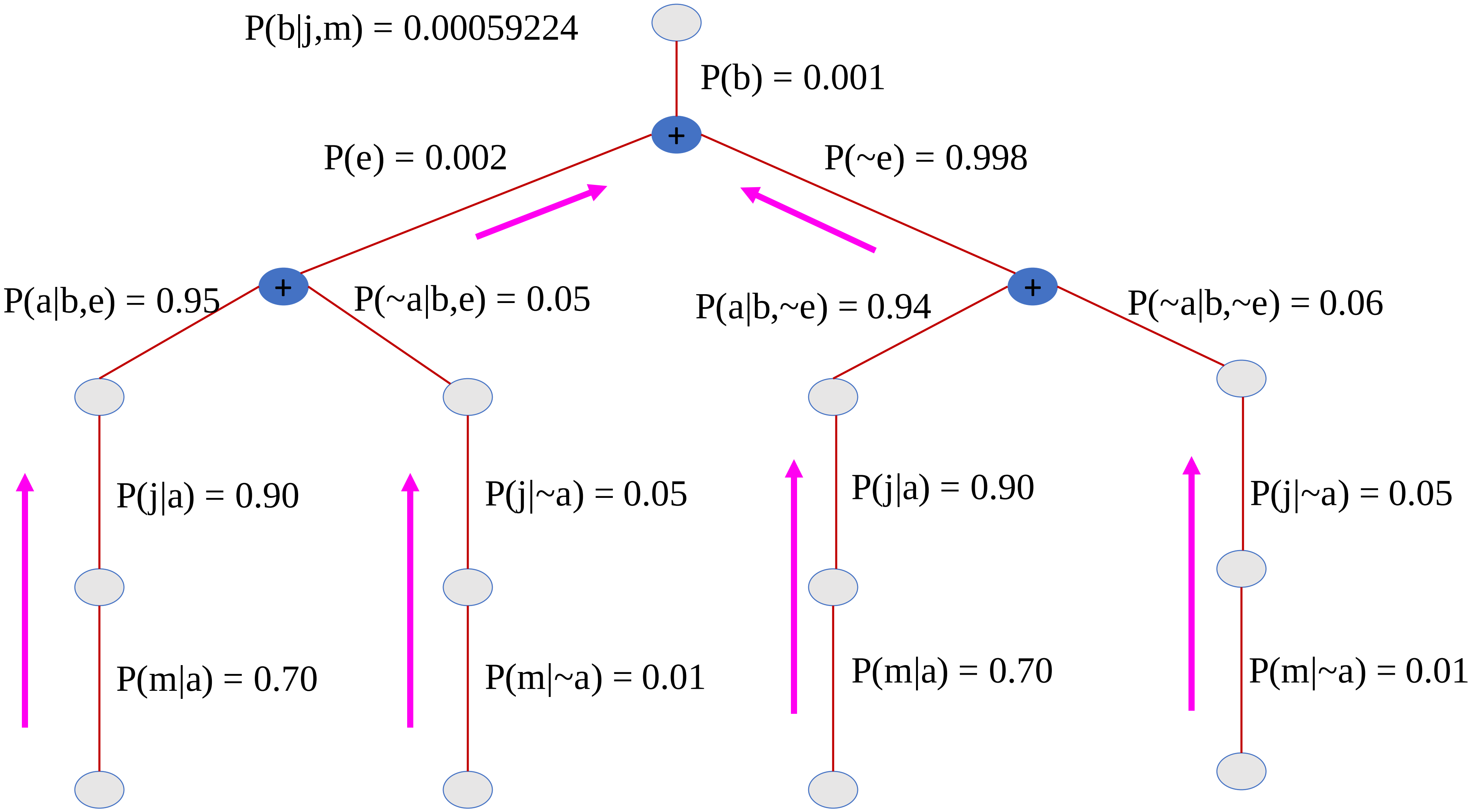
- Hidden variables:
  - Earthquake
  - Alarm

- P(b|j,m) = $\sum_e \sum_a P(b)P(e)P(a|b,e)P(j|a)P(m|a)$

- P(b|j,m) =$P(b) \sum_e \sum_a P(e)P(a|b,e)P(j|a)P(m|a)$

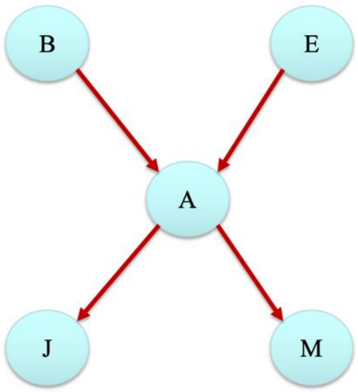
- P(b|j,m) =$P(b) \sum_e P(e)\ \sum_a P(a|b,e)P(j|a)P(m|a)$

Bayesian Network: Inference by Enumeration



Bayesian Network: Inference by Enumeration

- P(B|J=True, M=True)

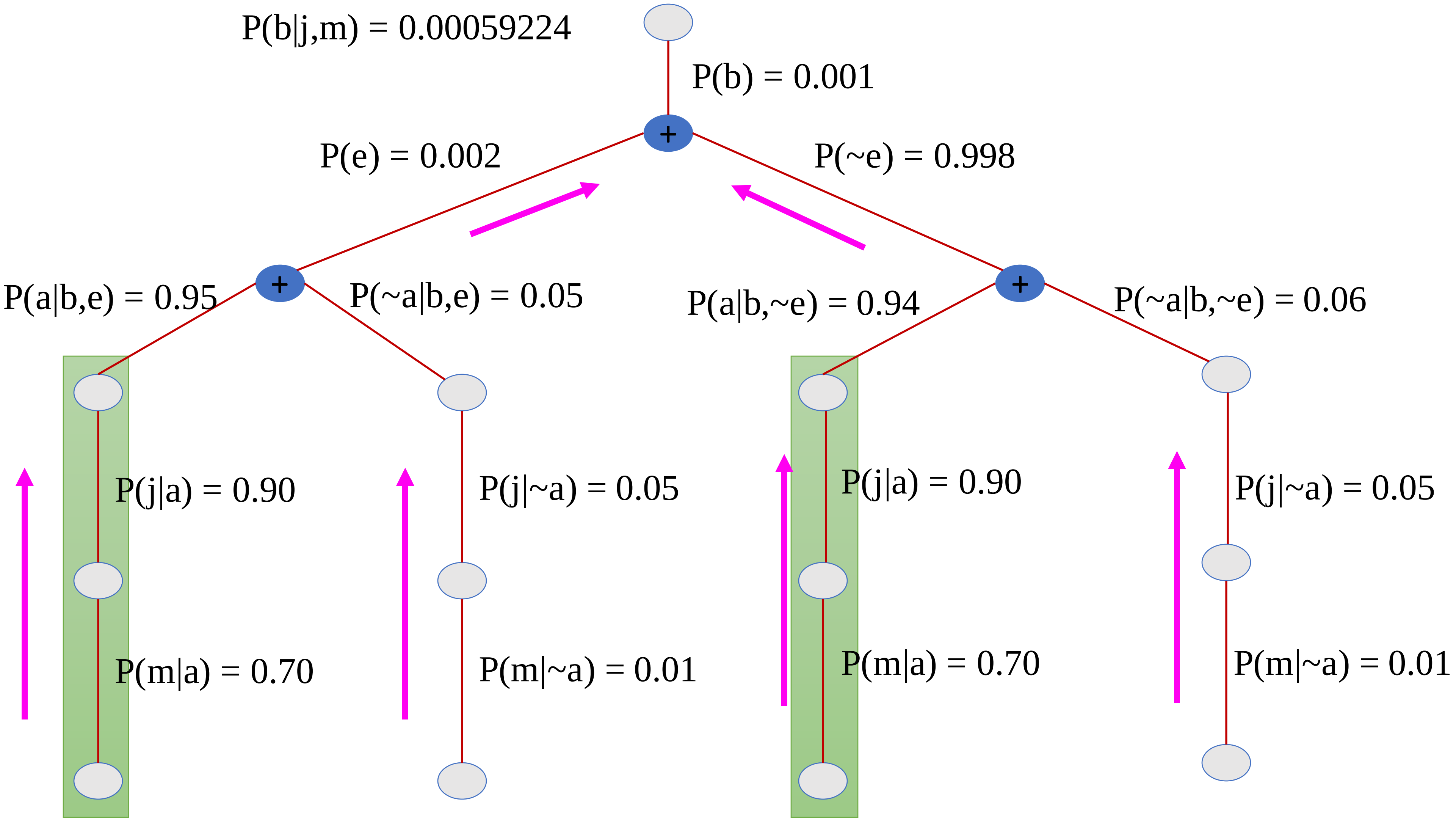


- Hidden variables:
  - Earthquake
  - Alarm

- P(b|j,m) =$P(b)\sum_e P(e)\ \sum_a P(a|b,e)P(j|a)P(m|a)$
- P(b|j,m) = <0.00059224,0.0014919> = <0.284,0.716>

- Chance of a burglary given calls from both neighbours is 28%
- Complexity: O((2^n)

Bayesian Network: Inference by Enumeration



Inference by Variable Elimination

Problem of Inference: Example

- Consider a chain Bayesian network
  - $p(y = 1, x_1, x_2, \dots, x_n) = p(x_1) \prod_{i=2}^{n} p(x_i | x_{i-1})$

- We are interested in computing the marginal probability $p(x_n)$

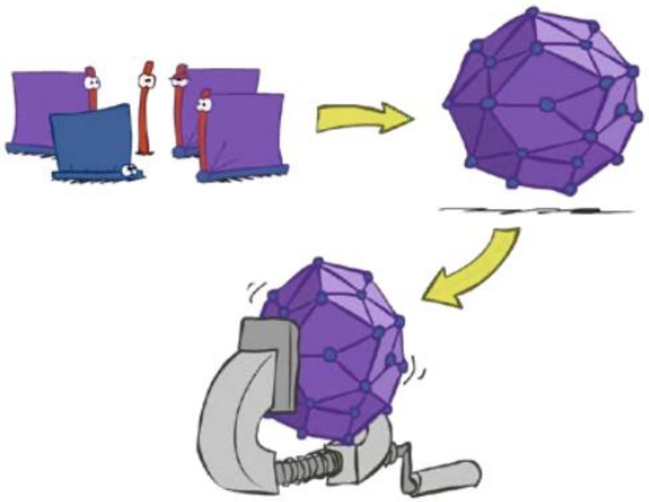
- Enumeration:
  - $p(x_n) = \sum_{x_1} \sum_{x_2} \dots \sum_{x_{n-1}} p(x_1, x_2, \dots, x_n)$
  - Complexity:
    - Sum the probability over all assignments of $x_1, x_2, \dots, x_{n-1}$
    - k^{n-1}

- Can we do better?

Inference by Elimination

- Why is inference by enumeration so slow?

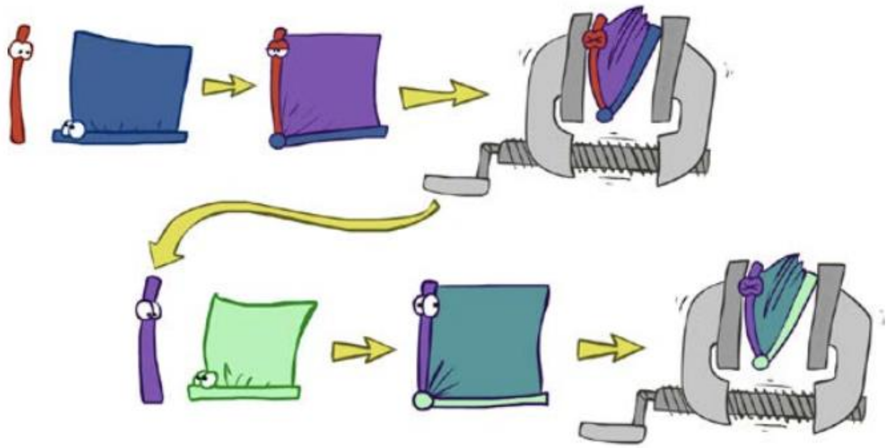
- We join up the whole joint distribution before you sum out the hidden variables



Variable Elimination

- Idea: interleave joining and marginalizing

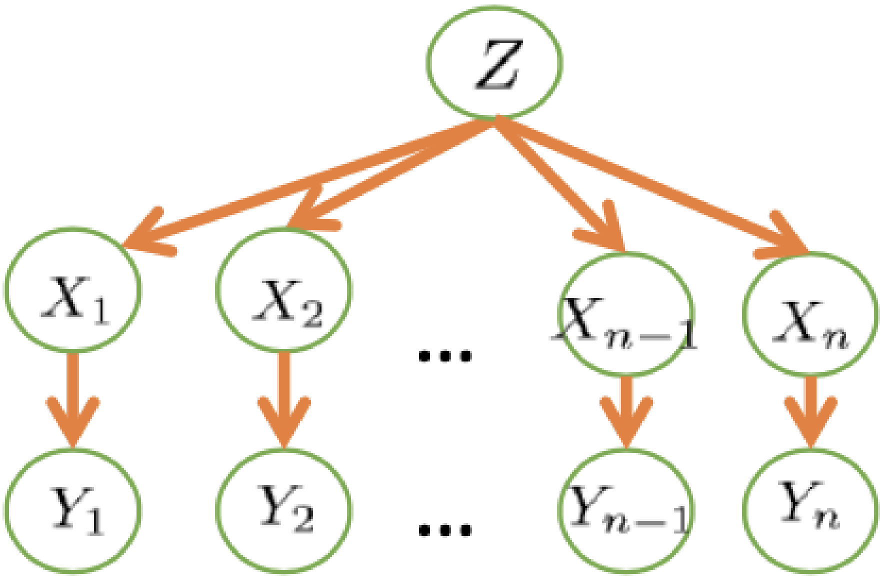
- Called “Variable Elimination”
- Still NP-hard, but usually much faster than inference by enumeration

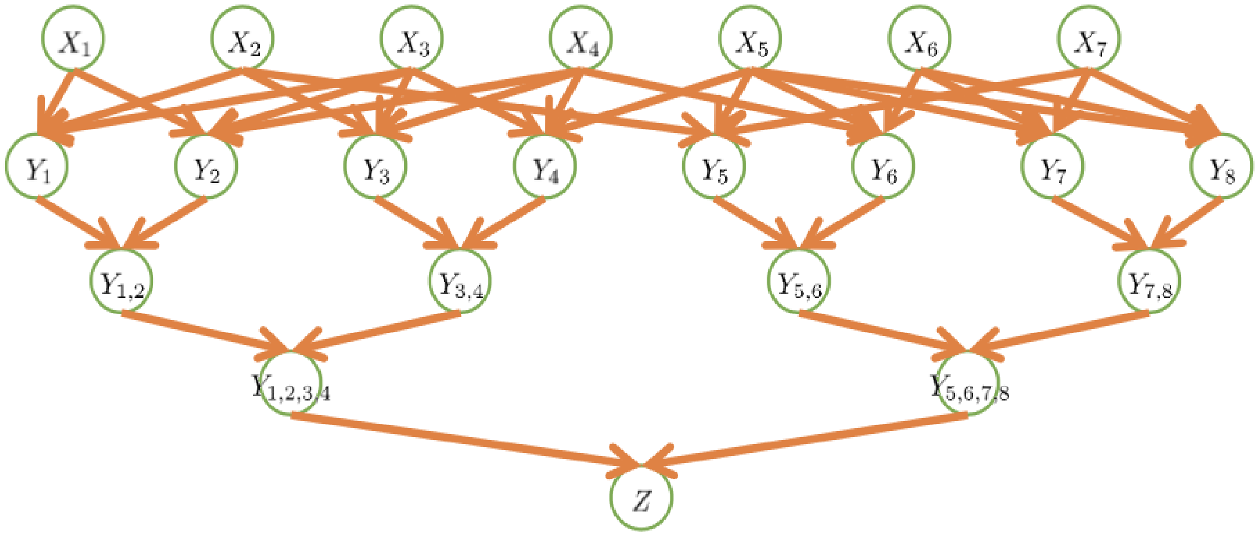


Variable Elimination

- Interleave joining and marginalizing
- d^k entries computed for a factor over k variables with domain sizes d
- Ordering of elimination of hidden variables can affect size of factors generated
- Worst case: running time exponential in the size of the Bayes' net

- Interleave joining and marginalizing
- d^k entries computed for a factor over k variables with domain sizes d
- Ordering of elimination of hidden variables can affect size of factors generated
- Worst case: running time exponential in the size of the Bayes' net





Problem of Inference: Example

- We may rewrite the sum in a way that "pushes in" certain variables deeper into the product
- $p(x_n) = \sum_{x_1} \sum_{x_2} \dots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i|x_{i-1})$
- $p(x_n) = \sum_{x_{n-1}} p(x_n|x_{n-1}) \sum_{x_{n-2}} p(x_{n-1}|x_{n-2}) \dots \sum_{x_1} p(x_2|x_1) p(x_1)$

- We sum the inner terms first, starting from x_1 and ending with x_{n-1}
- We start by computing an intermediary factor
  - $\tau(x_2) = \sum_{x_1} p(x_2|x_1) p(x_1)$ by summing out x_1
  - This takes $O(k^2)$ time because we must sum over x_1 for each assignment to x_2
  - The resulting factor $\tau(x_2)$ can be thought of as a table of k values
    - with one entry for each assignment to x_2
    - Not necessarily probability values

- $p(x_n) = \sum_{x_{n-1}} p(x_n|x_{n-1}) \sum_{x_{n-2}} p(x_{n-1}|x_{n-2}) \dots \sum_{x_2} p(x_3|x_2) \tau(x_2)$

Problem of Inference: Example

- $p(x_n) = \sum_{x_{n-1}} p(x_n|x_{n-1}) \sum_{x_{n-2}} p(x_{n-1}|x_{n-2}) \ldots \sum_{x_1} p(x_2|x_1) p(x_1)$
- $p(x_n) = \sum_{x_{n-1}} p(x_n|x_{n-1}) \sum_{x_{n-2}} p(x_{n-1}|x_{n-2}) \ldots \sum_{x_2} p(x_3|x_2) \tau(x_2)$

- This has the same form as the initial expression, except that we are summing over one fewer variable
- Compute another factor: $\tau(x_3) = \sum_{x_2} p(x_3|x_2)\tau(x_2)$
- Repeat the process until we are only left with x_n
- Each step takes $O(k^2)$ time
- Time Complexity: $O(nk^2)$

Eliminating Variables

- Factors
  - $p(x_1, \dots, x_n) = \prod_{c \in C} \varphi_c(x_c)$
  - Factor as a multi-dimensional table assigning a value to each assignment of a set of variables x_c
  - In a Bayesian network, the factors correspond to conditional probability distributions

- Factor Operations
  - The variable elimination algorithm repeatedly performs two factor operations:
    - **product** and
    - **marginalization**
  - We have been implicitly performing these operations in our chain example

Factor Product

- The factor product operation simply defines the product ϕ3:=ϕ1×ϕ2 of two factors ϕ1,ϕ2 as
  - $\varphi_3(x_c) = \varphi_1(x_c^{(1)}) \times \varphi_2(x_c^{(2)})$
  - The scope of ϕ3 is defined as the union of the variables in the scopes of ϕ1,ϕ2;
  - Also $x_c^{(i)}$ denotes an assignment to the variables in the scope of ϕi defined by the restriction of x_c to that scope
  - $\varphi_3(a, b, c) = \varphi_1(a, b) \times \varphi_2(b, c)$

Factor Product & Marginalization

- Next, the **marginalization** operation “locally” eliminates a set of variables from a factor
- If we have a factor $\phi(X,Y)$ over two sets of variables X,Y, marginalizing Y produces a new factor
  - $\tau(x) = \sum_y \varphi(x, y)$
  - where the sum is over all joint assignments to the set of variables Y
- We use τ to refer to the marginalized factor
- It is important to understand that this factor does not necessarily correspond to a probability distribution, even if ϕ was a CPD

Orderings

- Finally, the variable elimination algorithm requires an ordering over the variables according to which variables will be “eliminated.”
- In our chain example, we took the ordering implied by the DAG

- Notes:
  - Different orderings may dramatically alter the running time of the variable elimination algorithm.
  - It is NP-hard to find the best ordering.

Variable Elimination Algorithm

- Essentially, we loop over the variables as ordered by O
- Eliminate them in that ordering
- Intuitively, this corresponds to choosing a sum and “pushing it in” as far as possible inside the product of the factors

- For each variable X_i (ordered according to O)
  - Multiply all factors Φ_i containing X_i
  - Marginalize out X_i to obtain a new factor τ
  - Replace the factors Φ_i with τ

VE: Example

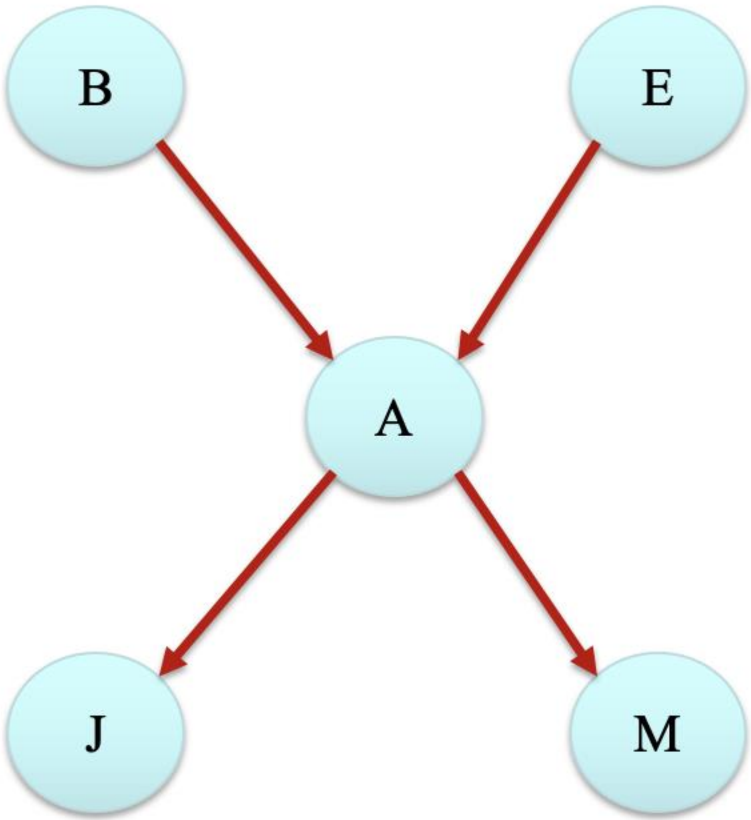
$$p(j, m, a, b, e) = p(j|a)p(m|a)p(a|b, e)p(b)p(e)$$

$$\tau_1(a, e) = \sum_b p(a|b, e)p(b)$$

$$\tau_2(a) = \sum_e \tau_1(a, e)p(e)$$

$$\tau_3(j, m) = \sum_a \tau_2(a)p(j|a)p(m|a)$$

…



Variable Elimination Algorithm

- Evaluate expressions from right to left
- Summations over each variable are done only for those portions of the expression that depend on the variable
- $P(B|j,m) = \alpha \overset{B}{\boxed{P(B)}} \sum_e \overset{E}{\boxed{P(e)}} \sum_a \overset{A}{\boxed{P(a|B,e)}} \overset{J}{\boxed{P(j|a)}} \overset{M}{\boxed{P(m|a)}}$
- Each part of the expression is annotated with the name of the associated variable
    - These parts are called factors
- $f_M(A) = \begin{pmatrix} P(m|a) \\ P(m|{\sim}a) \end{pmatrix}$
- $f_J(A) = \begin{pmatrix} P(j|a) \\ P(j|{\sim}a) \end{pmatrix}$

Variable Elimination Algorithm

B E A J M

- $P(B|j,m) = \ \alpha P(B) \sum_e P(e) \sum_a P(a|B,e) P(j|a) P(m|a)$

- Sum out A from the product of three factors
- $f_{\bar{A}JM}(B,E) = \sum_a f_A(a,B,E) \times f_J(a) \times f_M(a)$
- $f_{\bar{A}JM}(B,E) = f_A(a,B,E) \times f_J(a) \times f_M(a) + f_A(\sim a,B,E) \times f_J(\sim a) \times f_M(\sim a)$
- Pointwise product

- $f_{\bar{E}\bar{A}JM}(B) = f_E(e) \times f_{\bar{A}JM}(B,e) + f_E(\sim e) \times f_{\bar{A}JM}(B,\sim e)$

- $P(B|j,m) = \alpha f_B(B) \times f_{\bar{E}\bar{A}JM}(B)$

VE: Running Time

- Clearly some orderings are more efficient than others

- In fact, the running time of Variable Elimination is $O(nk^{M+1})$
  - where M is the maximum size of any factor τ formed during the elimination process and
  - n is the number of variables

Choosing variable elimination orderings

- Choosing the optimal VE ordering is an NP-hard problem
- Heuristics:
  - *Min-neighbors*: Choose a variable with the fewest dependent variables.
  - *Min-weight*: Choose variables to minimize the product of the cardinalities of its dependent variables
  - *Min-fill*: Choose vertices to minimize the size of the factor that will be added to the graph.

Variable Elimination Algorithm

- function ELIMINATION-ASK(X,e,bn) returns a distribution over X
  - inputs: X, the query variable
    - e, evidence specified as an event
    - bn, a Bayesian network specifying joint distribution P(X1,X2,…,Xn)

  - factors←[]; vars ←REVERSE(VARS[bn])
  - for each var in vars do
    - factors ←[MAKE-FACTOR(var,e)|factors]
    - if var is a hidden variable then factors ←SUM-OUT(var,factors)
  - return NORMALIZE(POINTWISE-PRODUCT(factors))

Thank You