UVA CS 6316
– Fall 2015 Graduate:
Machine Learning

Lecture 25: Graphical models and Bayesian networks

Dr. Yanjun Qi

University of Virginia

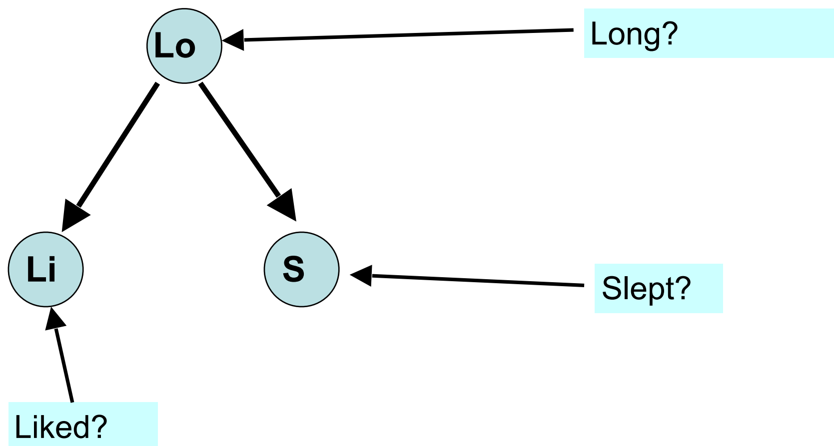
Department of
Computer Science

Independence

- Independence allows for easier models, learning and inference
- For example, with 3 binary variables we only need 3 parameters rather than 7.
- The saving is even greater if we have many more variables …
- In many cases it would be useful to assume independence, even if its not the case
- Is there any middle ground?

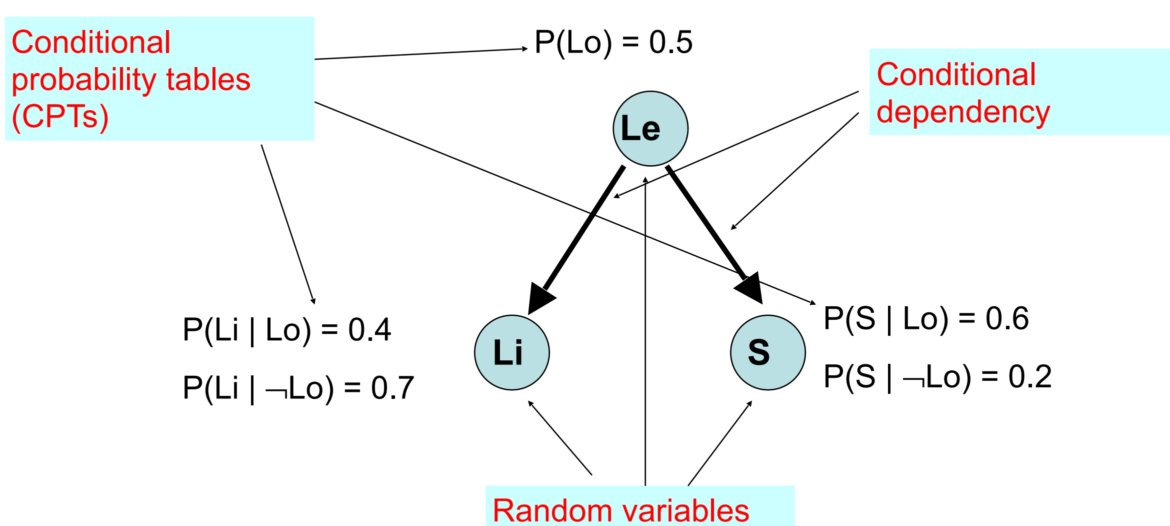
Bayesian networks

- Bayesian networks are *directed graphs* with nodes representing *random variables* and edges representing *dependency assumptions*
- Lets use a movie example: We would like to determine the joint probability for length, liked and slept in a movie



Bayesian networks: Notations

Bayesian networks are directed acyclic graphs.



Bayesian networks: Notations

The Bayesian network below represents the following joint probability distribution:

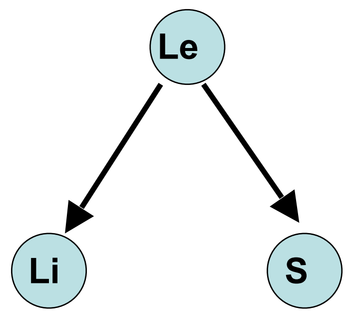
$$p(Le, Li, S) = P(Le)P(Li \mid Le)P(S \mid Le)$$

More generally Bayesian network represent the following joint probability distribution:

$$p(x_1 \ldots x_n) = \prod_i p(x_i \mid Pa(x_i))$$

The set of parents of x_i in the graph

P(Lo) = 0.5



P(Li | Lo) = 0.4

P(Li | ¬Lo) = 0.7

P(S | Lo) = 0.6

P(S | ¬Lo) = 0.2

Network construction and structural interpretation

Constructing a Bayesian network

- How do we go about constructing a network for a specific problem?
- Step 1: Identify the random variables
- Step 2: Determine the conditional dependencies
- Step 3: Populate the CPTs

Can be learned from observation data!

A example problem

- An alarm system
  B – Did a burglary occur?
  E – Did an earthquake occur?
  A – Did the alarm sound off?
  M – Mary calls
  J – John calls

- How do we reconstruct the network for this problem?

Factoring joint distributions

- Using the chain rule we can always factor a joint distribution as follows:

  P(A,B,E,J,M) =

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

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

  P(A | B,E,J,M) P(B | E, J,M) P(E | J,M) P(J,M)

  P(A | B,E,J,M) P(B | E, J,M) P(E | J,M)P(J | M)P(M)
- This type of conditional dependencies can also be represented graphically.

A Bayesian network

P(A | B,E,J,M) P(B | E, J,M) P(E | J,M)P(J | M)P(M)

Number of parameters:

A: 2^4

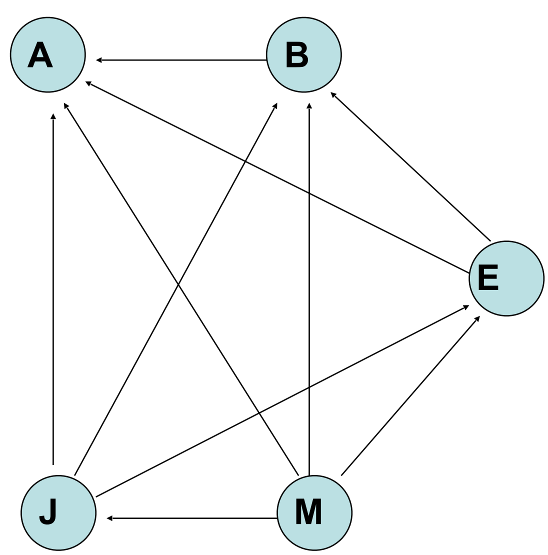
B: 2^3

E: 4

J: 2

M: 1

A total of 31 parameters



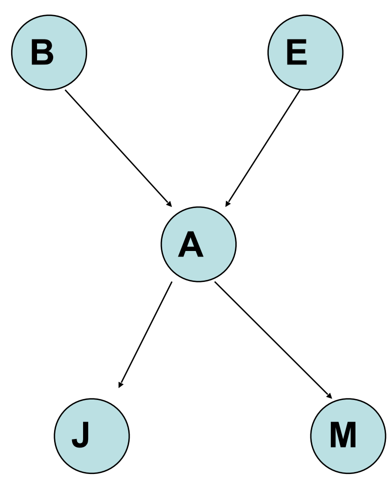
A better approach

- An alarm system
  B – Did a burglary occur?
  E – Did an earthquake occur?
  A – Did the alarm sound off?
  M – Mary calls
  J – John calls

- Lets use our knowledge of the domain!

Reconstructing a network

B – Did a burglary occur?
E – Did an earthquake occur?
A – Did the alarm sound off?
M – Mary calls
J – John calls



Reconstructing a network

Number of parameters:

A: 4

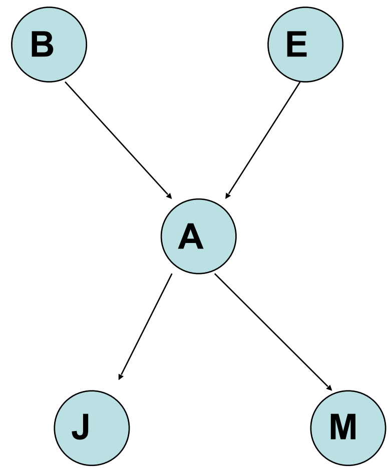
B: 1

E: 1

J: 2

M: 2

A total of 10 parameters

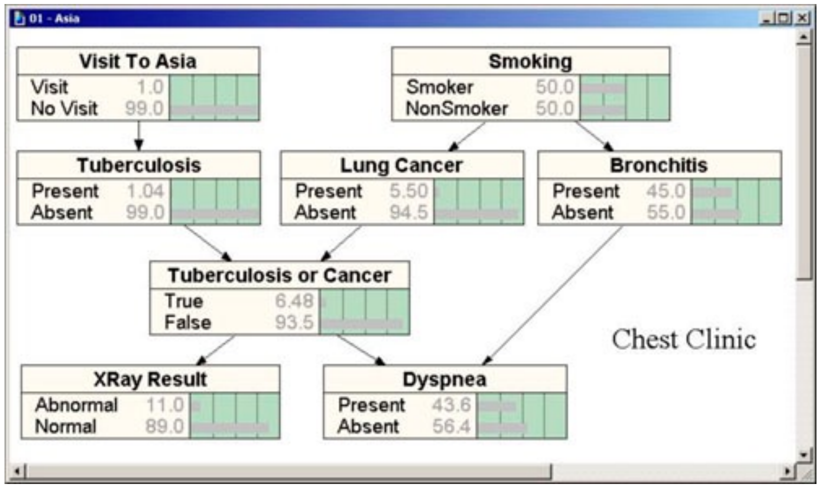


By relying on domain knowledge we saved 21 parameters!

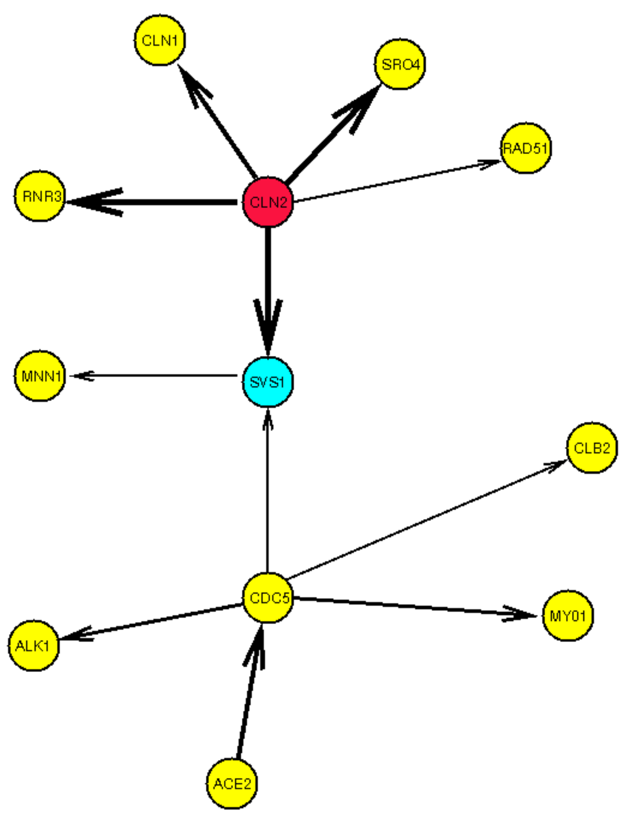
Constructing a Bayesian network: Revisited

- Step 1: Identify the random variables
- Step 2: Determine the conditional dependencies
  - Select on ordering of the variables
  - Add them one at a time
  - For each new variable X added select the minimal subset of nodes as parents such that X is independent from all other nodes in the current network given its parents.
- Step 3: Populate the CPTs
  - From examples using density estimation

Example: Bayesian networks for cancer detection



Example: Gene expression network

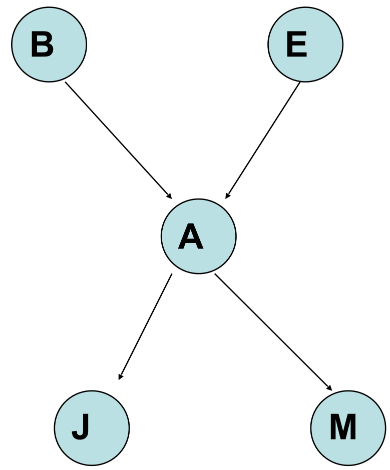


Conditional independence

• Two variables x,y are said to be conditionally independent given a third variable z if p(x,y|z) = p(x|z)p(y|z)

• In a Bayesian network a variable is conditionally independent of all other variables given it Markov blanket

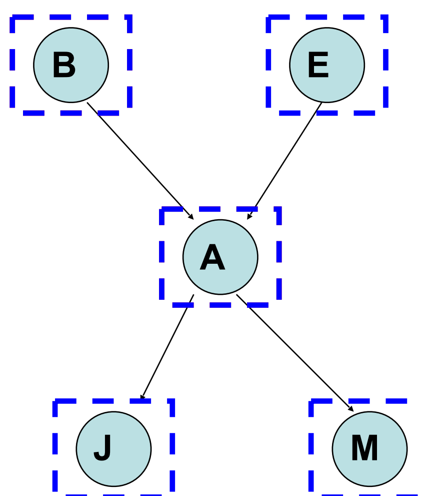
Markov blanket: All parent, children's and co-parents of children



Markov blankets: Examples

Markov blanket for B:
E, A

Markov blanket for A:
B, E, J, M



d-separation

• In some cases it would be useful for us to know under which conditions two variables are independent of each other

- Helps when trying to do inference
- Can help determine causality from structure

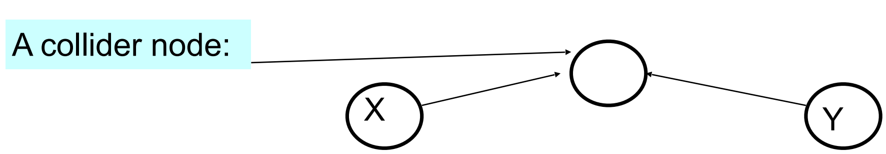
• Two variables x and y are d-separated given a set of variables Z (which could be empty) if x and y are conditionally independent given Z

• We denote such conditional independence as I(x,y|Z)

d-separation

- We will give rules to identify d-connected variables. Variables that are not d-connected are d-separated.
- The following three rules can be used to determine if x and y are d-connected given Z:

1. If Z is empty then x and y are d-connected if there exists a path between them does not contain a collider.
2. x and y are d-connected given Z if there exists a path between them that does not contain a collider and does not contain any member of Z
3. If Z contains a collider or one of its descendents then if a path between x and y contains this node they are d-connected



Inference in BN's

Bayesian network: Inference

- Once the network is constructed, we can use algorithms for inferring the values of unobserved variables.
- For example, in our previous network the only observed variables are the phone call and the radio announcement. However, what we are really interested in is whether there was a burglary or not.
- How can we determine that?

Inference

- Lets start with a simpler question
  - How can we compute a joint distribution from the network?
  - For example, P(B,¬E,A,J, ¬M)?
- Answer:
  - That's easy, lets use the network

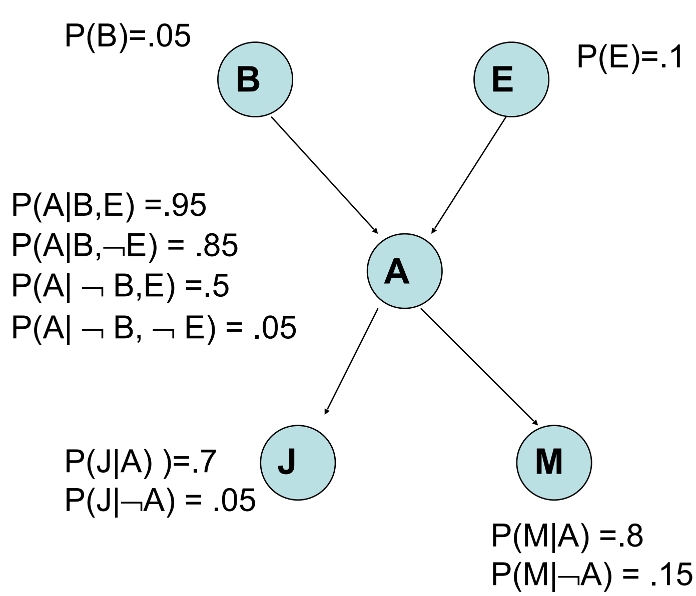
Computing: P(B,¬E,A,J, ¬M)

P(B,¬E,A,J, ¬M) =

P(B)P(¬E)P(A | B, ¬E) P(J | A)P(¬M | A)

= 0.05*0.9*.85*.7*.2

= 0.005355



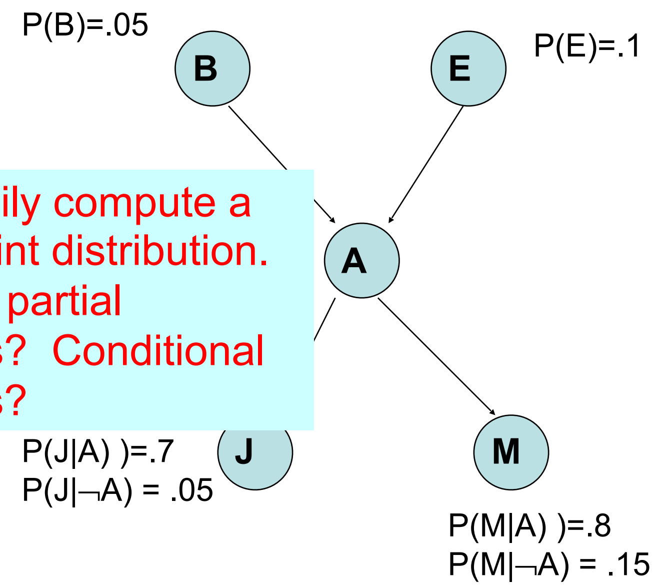
Computing: P(B,¬E,A,J, ¬M)

P(B,¬E,A,J, ¬M) =

P(B)P(¬E)P(A | B, ¬E) P(J | A)P(¬M | A)

= 0.05*0.9*.85*.7*.2

= 0.005355



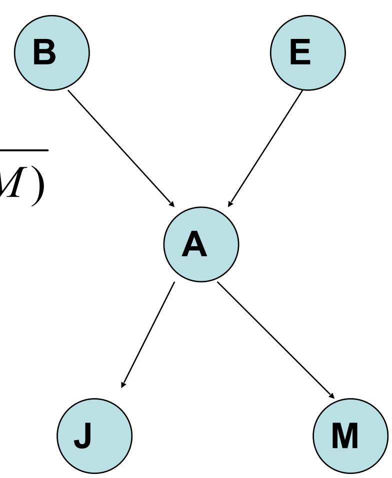
We can easily compute a complete joint distribution. What about partial distributions? Conditional distributions?

Inference

- We are interested in queries of the form: P(B | J,¬M)
- This can also be written as a joint:

$$P(B \mid J, \neg M) = \frac{P(B, J, \neg M)}{P(B, J, \neg M) + P(\neg B, J, \neg M)}$$

- How do we compute the new joint?



Inference in Bayesian networks

- We will discuss three methods:

1. Enumeration
2. Variable elimination
3. Stochastic inference

Computing partial joints

$$P(B \mid J, \neg M) = \frac{P(B, J, \neg M)}{P(B, J, \neg M) + P(\neg B, J, \neg M)}$$

Sum all instances with these settings (the sum is over the possible assignments to the other two variables, E and A)

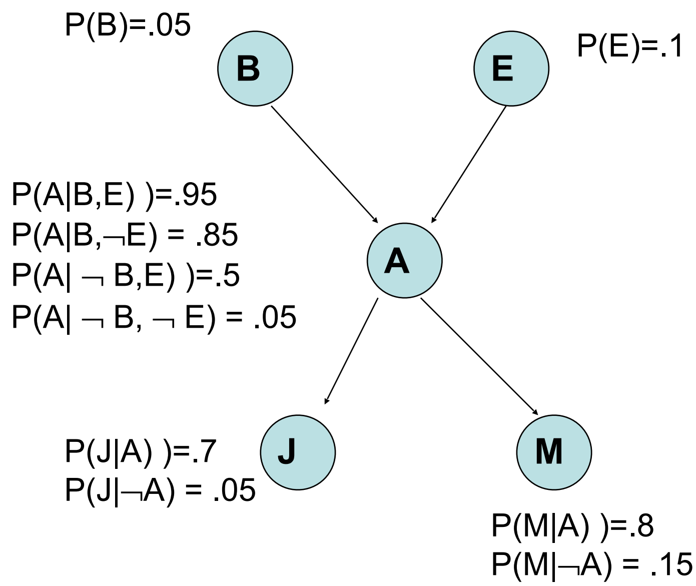
Computing: P(B,J, ¬M)

P(B,J, ¬M) =

P(B,J, ¬M,A,E)+

P(B,J, ¬M, ¬ A,E) + P(B,J, ¬M,A, ¬ E) + P(B,J, ¬M, ¬ A, ¬ E) =

0.0007+0.00001+0.005+0.0003 = 0.00601



Computing partial joints

$$P(B \mid J, \neg M) = \frac{P(B, J, \neg M)}{P(B, J, \neg M) + P(\neg B, J, \neg M)}$$

Sum all instances with these settings (the sum is over the possible assignments to the other two variables, E and A)

- This method can be improved by re-using calculations (similar to dynamic programming)
- Still, the number of possible assignments is exponential in the unobserved variables?
- That is, unfortunately, the best we can do. General querying of Bayesian networks is NP-complete

Inference in Bayesian networks if NP complete (sketch)

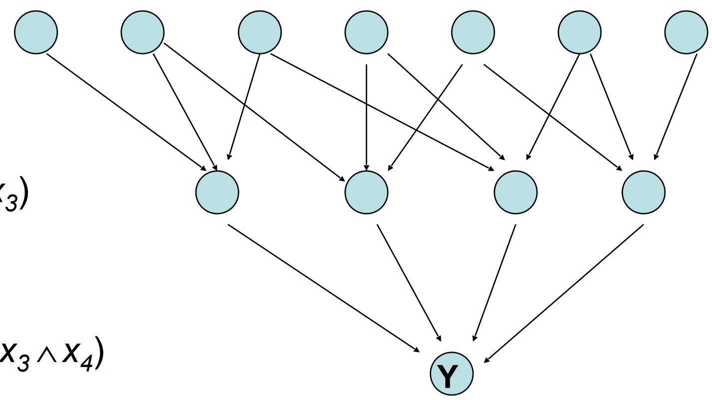
- Reduction from 3SAT
- Recall: 3SAT, find satisfying assignments to the following problem: $(a \vee b \vee c) \wedge (d \vee \neg b \vee \neg c) \ldots$

What is P(Y=1)?

$P(x_i=1) = 0.5$

$P(x_i=1) = (x_1 \vee x_2 \vee x_3)$

$P(Y=1) = (x_1 \wedge x_2 \wedge x_3 \wedge x_4)$



Inference in Bayesian networks

- We will discuss three methods:

1. Enumeration
2. Variable elimination
3. Stochastic inference

Variable elimination

P(B,J, ¬M) =

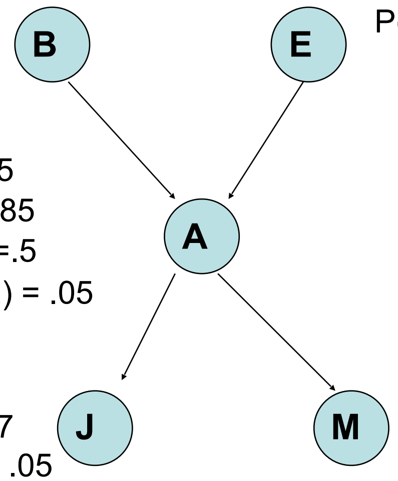
P(B,J, ¬M,A,E)+

P(B,J, ¬M, ¬ A,E) +
P(B,J, ¬M,A, ¬ E) + P(B,J, ¬M,
¬ A, ¬ E) =

0.0007+0.00001+0.005+0.0003
= 0.00601

P(B)=.05

P(E)=.1



P(A|B,E))=.95
P(A|B,¬E) = .85
P(A| ¬ B,E))=.5
P(A| ¬ B, ¬ E) = .05

P(J|A))=.7
P(J|¬A) = .05

P(M|A))=.8
P(M|¬A) = .15

Reuse computations rather than recompute probabilities

Computing: P(B,J, ¬M)

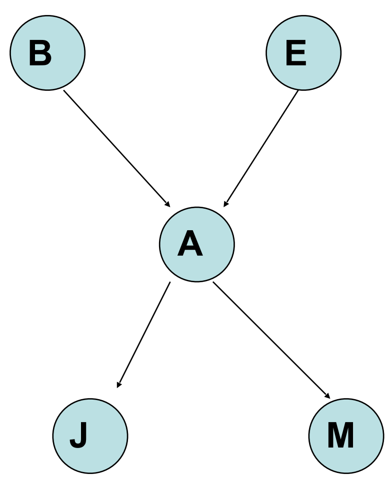
P(B,J, ¬M) =

P(B,J, ¬M,A,E)+

P(B,J, ¬M, ¬ A,E) + P(B,J,
¬M,A, ¬ E) + P(B,J, ¬M, ¬
A, ¬ E) =

$$\sum_a \sum_e P(B)P(e)P(a \mid B,e)P(M \mid a)P(J \mid a)$$

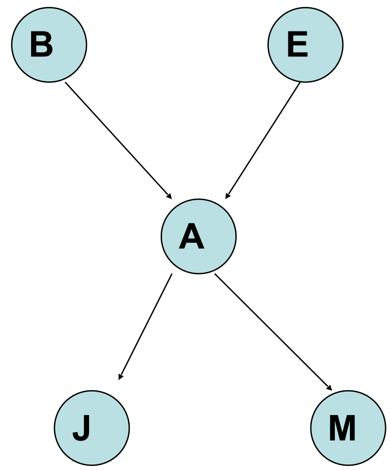
Store as a function of a and use whenever necessary (no need to recompute each time)



Variable elimination

$$P(B,J,M) = \sum_a \sum_e P(B)P(e)P(a \mid B,e)P(M \mid a)P(J \mid a)$$

$$= P(B)\sum_e P(e)\sum_a P(a \mid B,e)P(M \mid a)P(J \mid a)$$



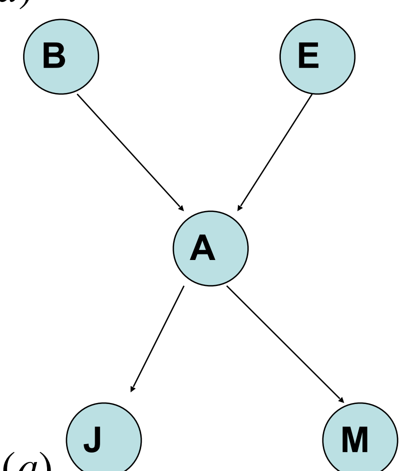
Set: $f_M(A) = \begin{pmatrix} P(M \mid A) \\ P(M \mid \neg A) \end{pmatrix}$

$$f_J(A) = \begin{pmatrix} P(J \mid A) \\ P(J \mid \neg A) \end{pmatrix}$$

Variable elimination

$$P(B,J,M) = \sum_a \sum_e P(B)P(e)P(a \mid B,e)P(M \mid a)P(J \mid a)$$

$$= P(B)\sum_e P(e)\sum_a P(a \mid B,e)P(M \mid a)P(J \mid a)$$



Set: $f_M(A) = \begin{pmatrix} P(M \mid A) \\ P(M \mid \neg A) \end{pmatrix}$

$$f_J(A) = \begin{pmatrix} P(J \mid A) \\ P(J \mid \neg A) \end{pmatrix}$$

$$P(B,J,M) = P(B)\sum_e P(e)\sum_a P(a \mid B,e) f_M(a) f_J(a)$$

Variable elimination

$$= P(B)\sum_e P(e)\sum_a P(a \mid B,e) f_M(a) f_J(a)$$

Lets continue with these functions:

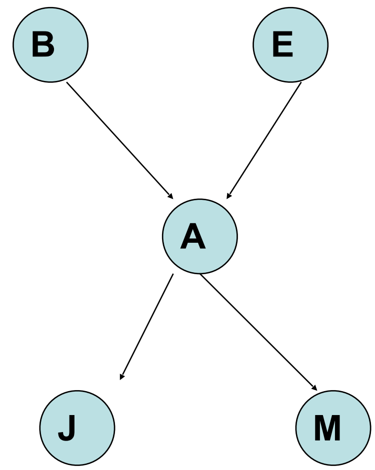
$$f_A(a,B,e) = P(a \mid B,e)$$

We can now define the following function:

$$f_{A,J,M}(B,e) = \sum_a f_A(a,B,e) f_J(a) f_M(a)$$

And so we can write:

$$P(B,J,M) = P(B)\sum_e P(e) f_{A,J,M}(B,e)$$



Variable elimination

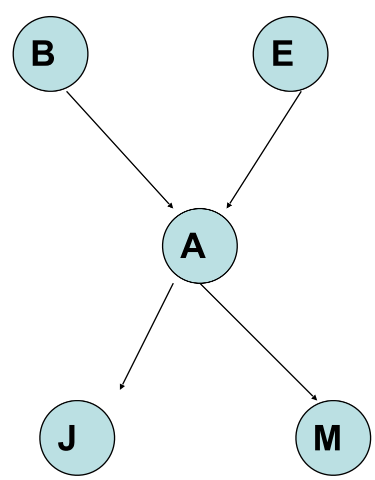
$$P(B,J,M) = P(B)\sum_e P(e) f_{A,J,M}(B,e)$$

Lets continue with another function:

$$f_{E,A,J,M}(B) = \sum_e P(e) f_{A,J,M}(B,e)$$

And finally we can write:

$$P(B,J,M) = P(B) f_{E,A,J,M}(B)$$

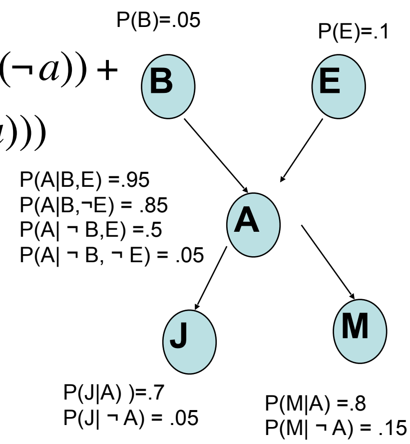


Example

$$P(B,J,M) = P(B) f_{E,A,J,M}(B)$$

$$= 0.05 \sum_e P(e) f_{A,J,M}(B,e) = 0.05(0.1 f_{A,J,M}(B,e) + 0.9 f_{A,J,M}(B,\neg e))$$

$$0.05(0.1(0.95 f_J(a) f_M(a) + 0.05 f_J(\neg a) f_M(\neg a)) +$$
$$0.9(.85 f_J(a) f_M(a) + .15 f_J(\neg a) f_M(\neg a)))$$



Calling the same function multiple times

Final computation (normalization)

$$P(B \mid J, \neg M) = \frac{P(B, J, \neg M)}{P(B, J, \neg M) + P(\neg B, J, \neg M)}$$

Algorithm

- *e* - evidence (the variables that are known)
- *vars* - the conditional probabilities derived from the network in reverse order (bottom up)
- For each *var* in *vars*

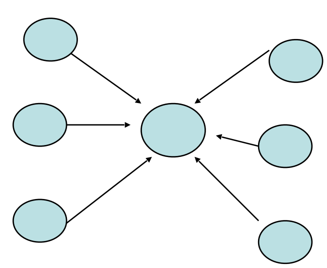
  - *factors* <- make_factor (*var,e*)

  - if *var* is a hidden variable then create a new factor by summing out *var*
- Compute the product of all factors
- Normalize

Computational complexity

- We are reusing computations so we are reducing the running time.
- However, there are still cases in which this algorithm we lead to exponential running time.
- Consider the case of $f_x(y_1 \dots y_n)$. When factoring x out we would need to account for all possible values of the y's.

Variable elimination can lead to significant costs saving but its efficiency depends on the network structure



Inference in Bayesian networks

- We will discuss three methods:

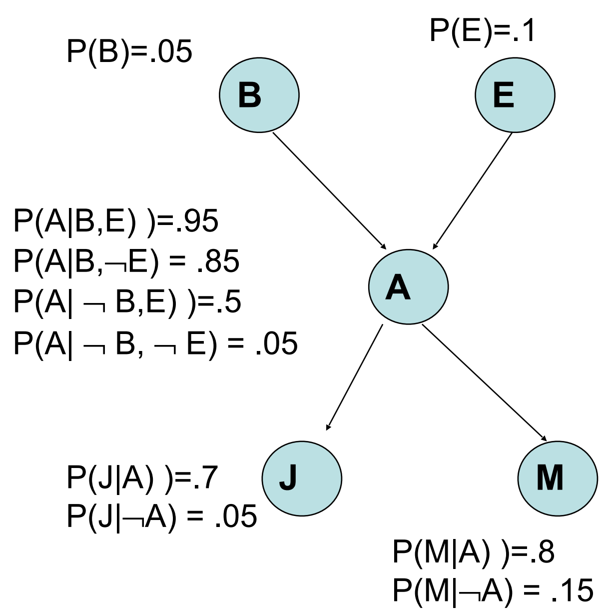
1. Enumeration
2. Variable elimination
3. Stochastic inference

Stochastic inference

- We can easily sample the joint distribution to obtain possible instances

1. Sample the free variable
2. For every other variable:
   - If all parents have been sampled, sample based on conditional distribution

We end up with a new set of assignments for B,E,A,J and M which are a random sample from the joint

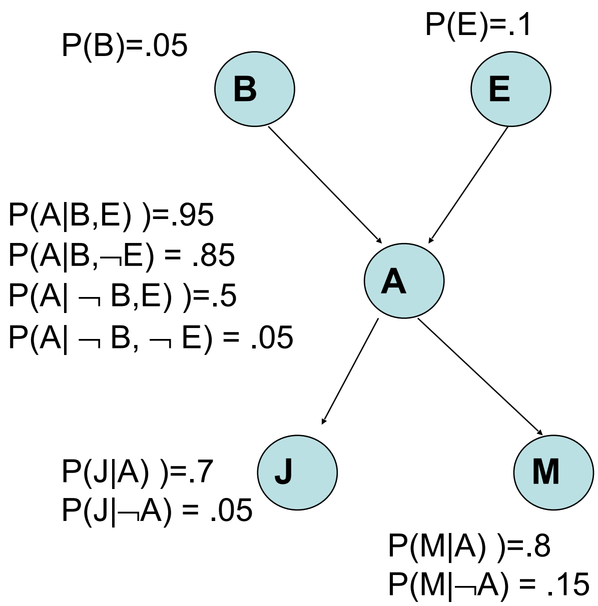


Stochastic inference

- We can easily sample the joint distribution to obtain possible instances

1. Sample the free variable
2. For every other variable:
   - If all parents have been sampled, sample based on conditional distribution

Its always possible to carry out this sampling procedure, why?



Using sampling for inference

- Lets revisit our problem: Compute P(B | J,¬M)
- Looking at the samples we can count:
  - N: total number of samples
  - N_c : total number of samples in which the condition holds (J,¬M)
  - N_B: total number of samples where the joint is true (B,J,¬M)
- For a large enough N
  - N_c / N $\approx$ P(J,¬M)
  - N_B / N $\approx$ P(B,J,¬M)
- And so, we can set

$$P(B \mid J,\neg M) = P(B,J,\neg M) / P(J,\neg M) \approx N_B / N_c$$

Using sampling for inference

- Lets revisit our problem: Compute P(B | J,¬M)
- Looking at the s

Problem: What if the condition rarely happens?

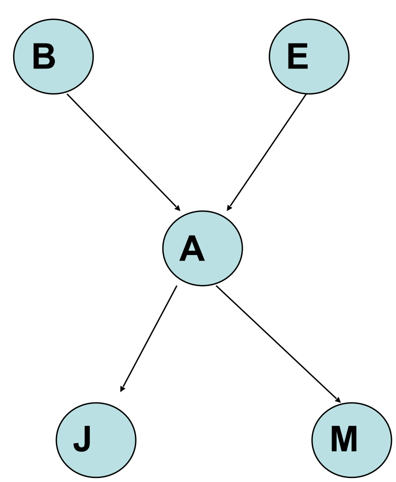
We would need lots and lots of samples, and most would be wasted

  - N: total number of
  - N_c : total number
  - N_B: total number
- For a large enough N
  - N_c / N ≈ P(J,¬M)
  - N_B / N ≈ P(B,J,¬M)
- And so, we can set

P(B | J,¬M) = P(B,J,¬M) / P(J,¬M) ≈ N_B / N_c

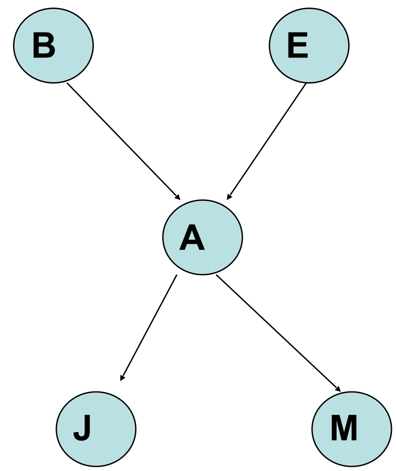
Weighted sampling

- Compute P(B | J,¬M)
- We can manually set the value of J to 1 and M to 0
- This way, all samples will contain the correct values for the conditional variables
- Problems?



Weighted sampling

- Compute P(B | J,¬M)
- Given an assignment to parents, we assign a value of 1 to J and 0 to M.
- We record the *probability* of this assignment ($w = p_1 * p_2$) and we weight the new joint sample by w



Weighted sampling algorithm for computing P(B | J,¬M)

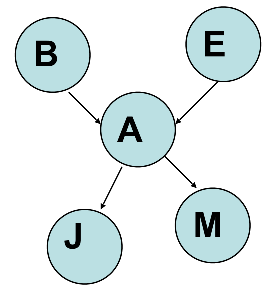
- Set $N_B, N_c = 0$
- Sample the joint setting the values for *J* and *M*, compute the weight, *w,* of this sample
- $N_c = N_c + w$
- If $B = 1$, $N_B = N_B + w$

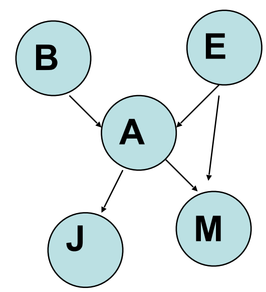
- After many iterations, set

P(B | J,¬M) = N_B / N_c

Other inference methods

- Convert network to a polytree
  - In a polytree no two nodes have more than one path between them
  - We can convert arbitrary networks to a polytree by clustering (grouping) nodes. For such a graph there is a algorithm which is linear in the number of nodes
  - However, converting into a polytree can result in an exponential increase in the size of the CPTs





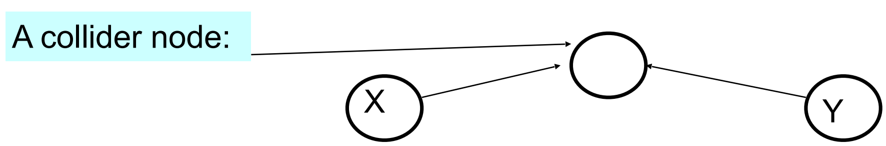
Bayesian networks: Inference

d-separation

- We will give rules to identify d-connected variables. Variables that are not d-connected are d-separated.
- The following three rules can be used to determine if x and y are d-connected given Z:

1. If Z is empty then x and y are d-connected if there exists a path between them does not contain a collider.
2. x and y are d-connected given Z if there exists a path between them that does not contain a collider and does not contain any member of Z
3. If Z contains a collider or one of its descendents then if a path between x and y contains this node they are d-connected

3. (revised) If all colliders on an undirected path between x and y are in Z or have a descendent in Z, then they are d-connected



Variables

- An alarm system
  B – Did a burglary occur?
  E – Did an earthquake occur?
  A – Did the alarm sound off?
  M – Mary calls
  J – John calls

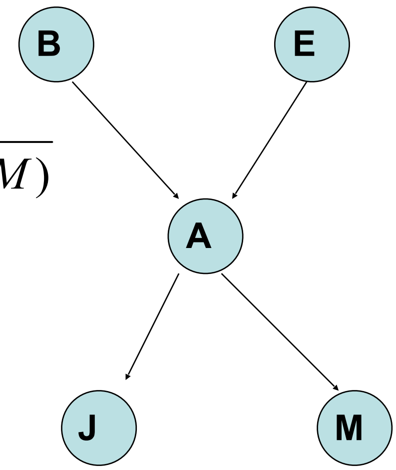
- Lets use our knowledge of the domain!

Inference

- We are interested in queries of the form:
  P(B | J,¬M)
- This can also be written as a joint:

$$P(B \mid J, \neg M) = \frac{P(B, J, \neg M)}{P(B, J, \neg M) + P(\neg B, J, \neg M)}$$

- How do we compute the new joint?



Inference in Bayesian networks

- We will discuss three methods:

1. Enumeration
2. Variable elimination
3. Stochastic inference

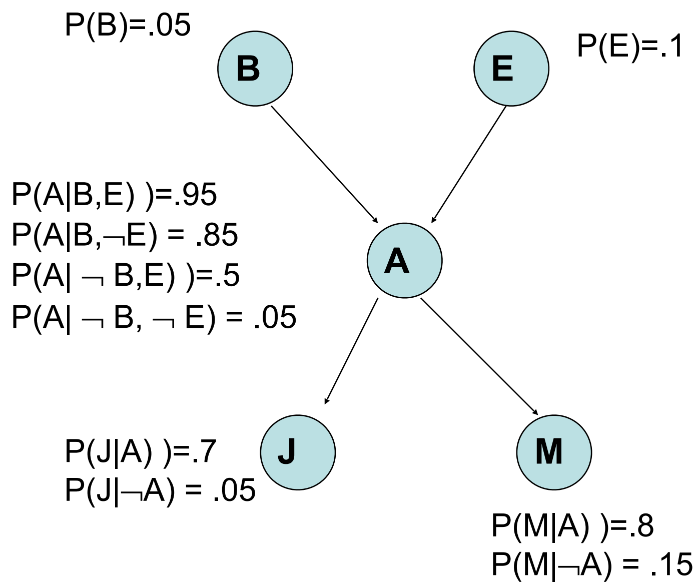
Computing: P(B,J, ¬M)

P(B,J, ¬M) =

P(B,J, ¬M,A,E)+

P(B,J, ¬M, ¬ A,E) + P(B,J, ¬M,A, ¬ E) + P(B,J, ¬M, ¬ A, ¬ E) =

0.0007+0.00001+0.005+0.0003 = 0.00601



Computing partial joints

$$P(B \mid J, \neg M) = \frac{P(B, J, \neg M)}{P(B, J, \neg M) + P(\neg B, J, \neg M)}$$

Sum all instances with these settings (the sum is over the possible assignments to the other two variables, E and A)

• This method can be improved by re-using calculations (similar to dynamic programming)

• Still, the number of possible assignments is exponential in the unobserved variables?

• That is, unfortunately, the best we can do. General querying of Bayesian networks is NP-complete

Inference in Bayesian networks if NP complete (sketch)

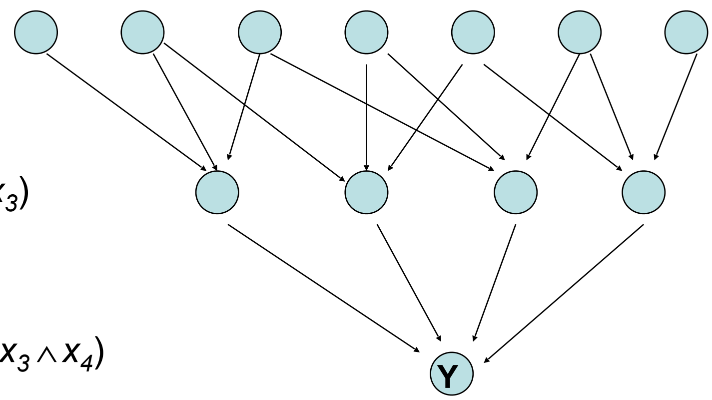
- Reduction from 3SAT
- Recall: 3SAT, find satisfying assignments to the following problem: $(a \vee b \vee c) \wedge (d \vee \neg b \vee \neg c)$ …

What is P(Y=1)?

$P(x_i=1)$ = 0.5

$P(x_i=1) = (x_1 \vee x_2 \vee x_3)$

$P(Y=1) = (x_1 \wedge x_2 \wedge x_3 \wedge x_4)$



Inference in Bayesian networks

- We will discuss three methods:

1. Enumeration
2. Variable elimination
3. Stochastic inference

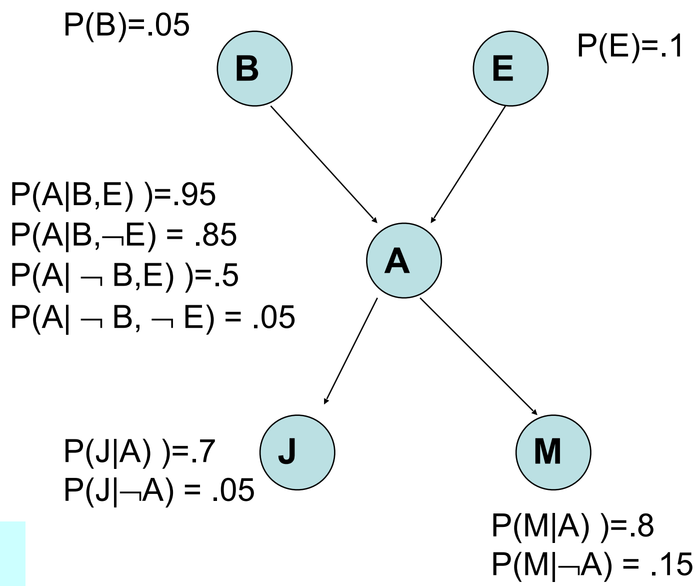
Variable elimination

P(B,J, ¬M) =

P(B,J, ¬M,A,E)+

P(B,J, ¬M, ¬ A,E) +
P(B,J, ¬M,A, ¬ E) + P(B,J, ¬M, ¬ A, ¬ E) =

0.0007+0.00001+0.005+0.0003
= 0.00601



Reuse computations rather than recompute probabilities

Computing: P(B,J, ¬M)

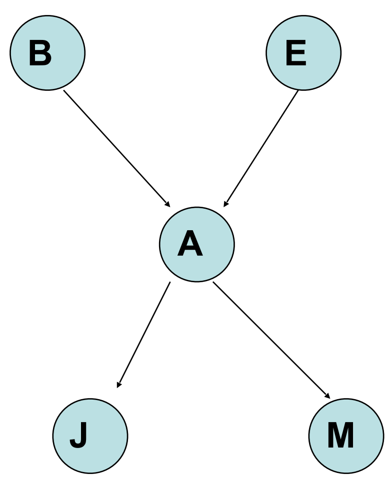
P(B,J, ¬M) =

P(B,J, ¬M,A,E)+

P(B,J, ¬M, ¬ A,E) + P(B,J, ¬M,A, ¬ E) + P(B,J, ¬M, ¬ A, ¬ E) =

$$\sum_a \sum_e P(B)P(e)P(a \mid B,e)P(M \mid a)P(J \mid a)$$

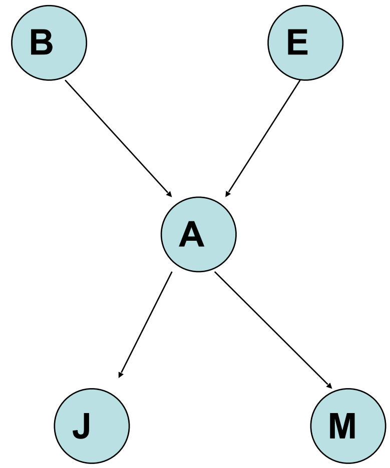
Store as a function of a and use whenever necessary (no need to recompute each time)



Variable elimination

$$P(B,J,M) = \sum_a \sum_e P(B)P(e)P(a \mid B,e)P(M \mid a)P(J \mid a)$$

$$= P(B)\sum_e P(e)\sum_a P(a \mid B,e)P(M \mid a)P(J \mid a)$$



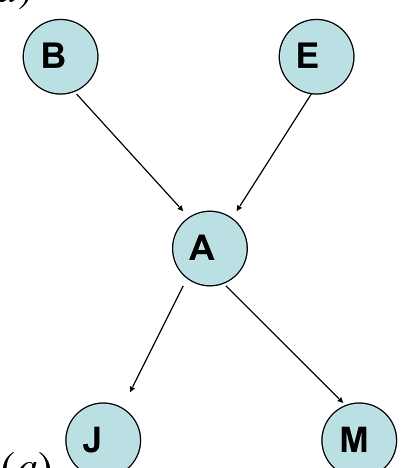
Set: $f_M(A) = \begin{pmatrix} P(M \mid A) \\ P(M \mid \neg A) \end{pmatrix}$

$$f_J(A) = \begin{pmatrix} P(J \mid A) \\ P(J \mid \neg A) \end{pmatrix}$$

Variable elimination

$$P(B,J,M) = \sum_a \sum_e P(B)P(e)P(a \mid B,e)P(M \mid a)P(J \mid a)$$

$$= P(B)\sum_e P(e)\sum_a P(a \mid B,e)P(M \mid a)P(J \mid a)$$



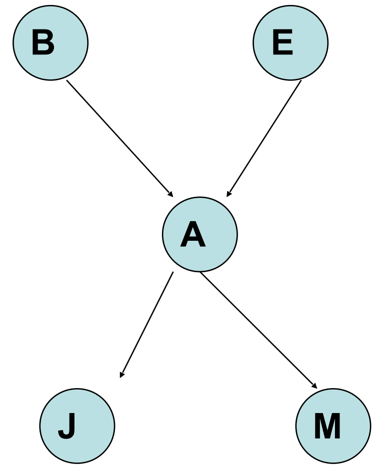
Set: $f_M(A) = \begin{pmatrix} P(M \mid A) \\ P(M \mid \neg A) \end{pmatrix}$

$$f_J(A) = \begin{pmatrix} P(J \mid A) \\ P(J \mid \neg A) \end{pmatrix}$$

$$P(B,J,M) = P(B)\sum_e P(e)\sum_a P(a \mid B,e) f_M(a) f_J(a)$$

Variable elimination

$$= P(B)\sum_e P(e)\sum_a P(a \mid B,e) f_M(a) f_J(a)$$



Lets continue with these functions:

$$f_A(a,B,e) = P(a \mid B,e)$$

We can now define the following function:

$$f_{A,J,M}(B,e) = \sum_a f_A(a,B,e) f_J(a) f_M(a)$$

And so we can write:

$$P(B,J,M) = P(B)\sum_e P(e) f_{A,J,M}(B,e)$$

Variable elimination

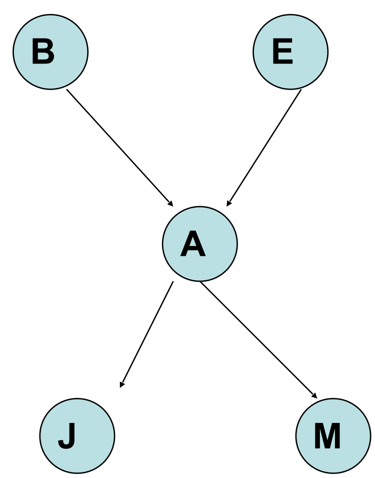
$$P(B,J,M) = P(B)\sum_e P(e) f_{A,J,M}(B,e)$$

Lets continue with another function:

$$f_{E,A,J,M}(B) = \sum_e P(e) f_{A,J,M}(B,e)$$

And finally we can write:

$$P(B,J,M) = P(B) f_{E,A,J,M}(B)$$

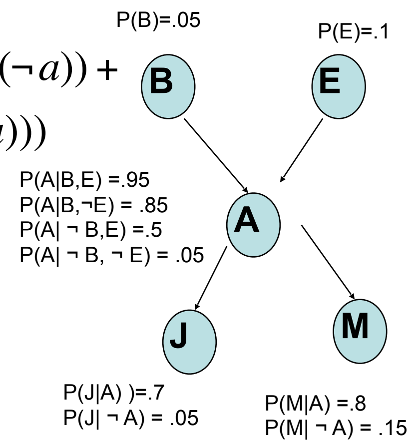


Example

$$P(B,J,M) = P(B) f_{E,A,J,M}(B)$$

$$= 0.05 \sum_e P(e) f_{A,J,M}(B,e) = 0.05(0.1 f_{A,J,M}(B,e) + 0.9 f_{A,J,M}(B,\neg e))$$

$$0.05(0.1(0.95 f_J(a) f_M(a) + 0.05 f_J(\neg a) f_M(\neg a)) +$$
$$0.9(.85 f_J(a) f_M(a) + .15 f_J(\neg a) f_M(\neg a)))$$



Calling the same function multiple times

Final computation (normalization)

$$P(B \mid J, \neg M) = \frac{P(B, J, \neg M)}{P(B, J, \neg M) + P(\neg B, J, \neg M)}$$

Algorithm

- *e* - evidence (the variables that are known)
- *vars* - the conditional probabilities derived from the network in reverse order (bottom up)
- For each *var* in *vars*

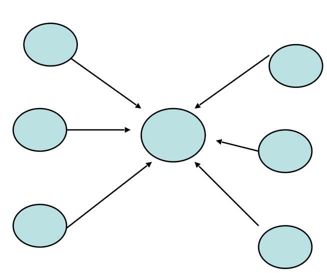
  - *factors* <- make_factor (*var,e*)

  - if *var* is a hidden variable then create a new factor by summing out *var*
- Compute the product of all factors
- Normalize

Computational complexity

- We are reusing computations so we are reducing the running time.
- However, there are still cases in which this algorithm we lead to exponential running time.
- Consider the case of $f_x(y_1 \ldots y_n)$. When factoring x out we would need to account for all possible values of the y's.

Variable elimination can lead to significant costs saving but its efficiency depends on the network structure



Inference in Bayesian networks

- We will discuss three methods:

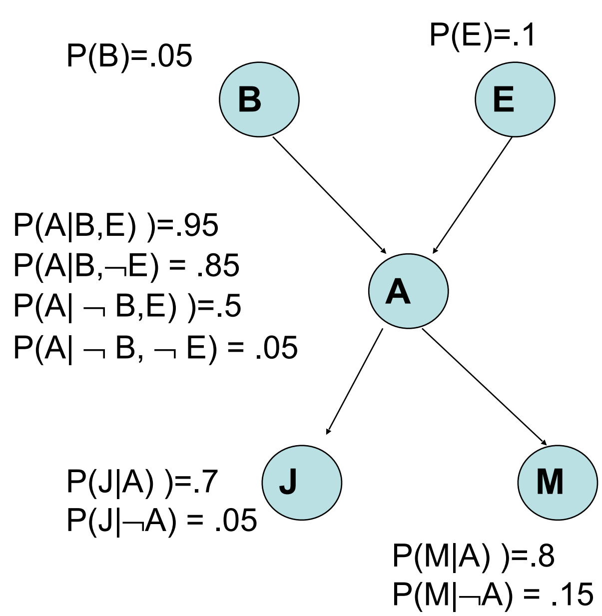
1. Enumeration
2. Variable elimination
3. Stochastic inference

Stochastic inference

- We can easily sample the joint distribution to obtain possible instances

1. Sample the free variable
2. For every other variable:
   - If all parents have been sampled, sample based on conditional distribution

We end up with a new set of assignments for B,E,A,J and M which are a random sample from the joint

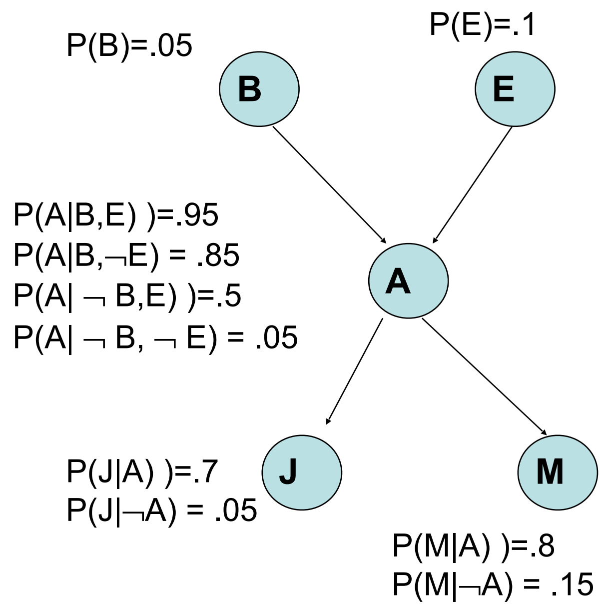


Stochastic inference

- We can easily sample the joint distribution to obtain possible instances

1. Sample the free variable
2. For every other variable:
   - If all parents have been sampled, sample based on conditional distribution

Its always possible to carry out this sampling procedure, why?



Using sampling for inference

- Lets revisit our problem: Compute P(B | J,¬M)
- Looking at the samples we can count:
  - N: total number of samples
  - N_c : total number of samples in which the condition holds (J,¬M)
  - N_B: total number of samples where the joint is true (B,J,¬M)
- For a large enough N
  - N_c / N $\approx$ P(J,¬M)
  - N_B / N $\approx$ P(B,J,¬M)
- And so, we can set

$$P(B \mid J,\neg M) = P(B,J,\neg M) / P(J,\neg M) \approx N_B / N_c$$

Using sampling for inference

- Lets revisit our problem: Compute P(B | J,¬M)
- Looking at the s
  - N: total number of
  - N_c : total number
  - N_B: total number
- For a large enough N
  - N_c / N ≈ P(J,¬M)
  - N_B / N ≈ P(B,J,¬M)
- And so, we can set

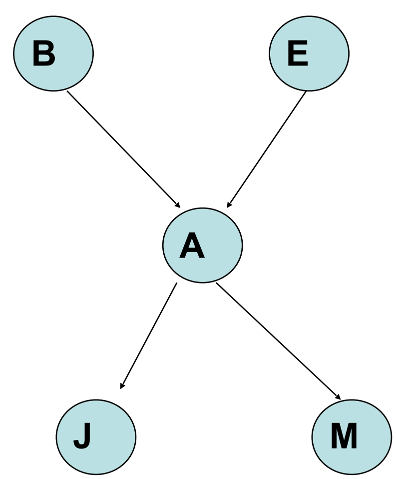
P(B | J,¬M) = P(B,J,¬M) / P(J,¬M) ≈ N_B / N_c

Problem: What if the condition rarely happens?

We would need lots and lots of samples, and most would be wasted

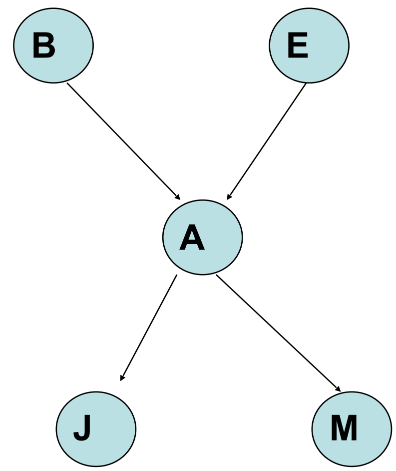
Weighted sampling

- Compute P(B | J,¬M)
- We can manually set the value of J to 1 and M to 0
- This way, all samples will contain the correct values for the conditional variables
- Problems?



Weighted sampling

- Compute P(B | J,¬M)
- Given an assignment to parents, we assign a value of 1 to J and 0 to M.
- We record the *probability* of this assignment ($w = p_1 * p_2$) and we weight the new joint sample by w



Weighted sampling algorithm for computing P(B | J,¬M)

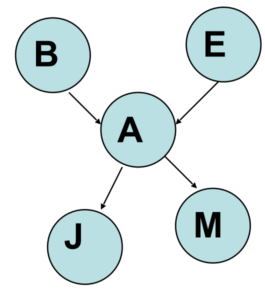
- Set $N_B, N_c = 0$
- Sample the joint setting the values for *J* and *M*, compute the weight, *w,* of this sample
- $N_c = N_c + w$
- If $B = 1$, $N_B = N_B + w$

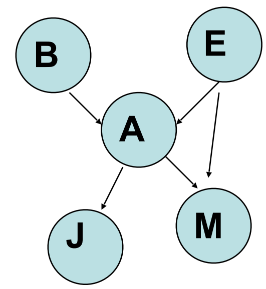
- After many iterations, set

P(B | J,¬M) = N_B / N_c

Other inference methods

- Convert network to a polytree
  - In a polytree no two nodes have more than one path between them
  - We can convert arbitrary networks to a polytree by clustering (grouping) nodes. For such a graph there is a algorithm which is linear in the number of nodes
  - However, converting into a polytree can result in an exponential increase in the size of the CPTs





Important points

- Bayes rule
- Joint distribution, independence, conditional independence
- Attributes of Bayesian networks
- Constructing a Bayesian network
- Inference in Bayesian networks

References

- Bishop 8.1 and 8.2.2
- Big thanks to Prof. Ziv Bar-Joseph @ CMU for allowing me to reuse some of his slides