

Lecture 7: Bayesian Networks



Gabriel Poesia

Assistant Professor, CSE

University of Michigan, Ann Arbor

poesia@umich.edu

<https://gpoesia.com>

- 0. Administrivia
- 1. Naïve Bayes
- 2. Bayesian networks
- 3. Independence in Bayesian Networks
- 4. Exact inference
- 5. Approximate inference

HW1 due tonight, 11:59pm on Gradescope!

New GSI in the staff: Zhengjie (Jerry) Xu

Jerry will lead the discussion section this week

Jerry's office hours will be on Tuesday 4-5pm (same room as Seung's, 1637 BBB - see course website)

Feedback form for +0.5% bonus point (see instructions on Ed)

RECAP

Joint distributions and inference

The joint distribution $P(X_1, \dots, X_n)$ had everything we needed to answer probabilistic queries.

A "query" has some variable (or many) Q we want to know about.

We observe some evidence $E = e$

There might be other hidden variables H in our model (neither observed nor in the question).

With the joint, for each possible answer q , we can add up the joint probabilities over all values of H :

$$\text{score}(q) = \sum_h P(q, e, h), \quad P(Q = q \mid e) \propto \text{score}(q)$$

$\propto$ ("proportional to") means the scores still need normalization:

$$P(Q = q \mid e) = \frac{\text{score}(q)}{\sum_{q'} \text{score}(q')}$$

But the normalization constant is the same for any q

To find only the most likely answer, we can compare scores without normalizing.

The joint is large: If X_i has r_i possible values, the full joint has

$$\prod_i r_i \text{ entries.}$$

When there are many variables, it's impractical to have (or learn) combinatorially many parameters.

Conditional independence

A and B are independent ($A \perp B$) if their joint factorizes as the marginals:

$$P(A = a, B = b) = P(A = a)P(B = b)$$

Otherwise they're dependent.

Recall the friend's unknown coin from last time: they chose coin A with probability .8 or coin B with probability .2.

Coin A was fair ($P(A = \text{Head}) = 0.5$), while coin B was biased ($P(B = H) = 0.75$).

They gave us one of them, and we flipped it many times.

The results are not independent for us: why?

Seeing heads makes another head more likely because it made coin B more plausible.

But, if they tell us which coin it is, with that the results become independent.

So they are conditionally independent given we know the coin.

DEFINITION (CONDITIONAL INDEPENDENCE)

$X \perp Y \mid Z$ if, for every z with $P(z) > 0$,

$$P(x, y \mid z) = P(x \mid z)P(y \mid z) \quad \text{for all } x, y.$$

Independence lets us factorize the joint and need potentially much fewer parameters.

Often a useful, practical modeling choice even if not actually true!

Naïve Bayes

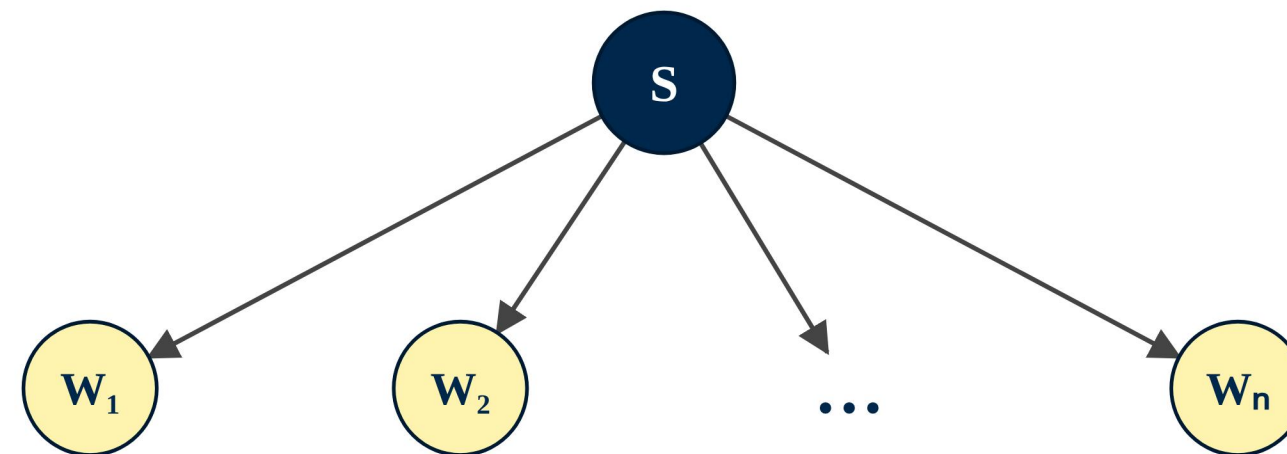
A model for generating messages

Suppose we want to classify emails as spam or non-spam.

Suppose our vocabulary contains $V = 100,000$ words, and we're given email length n .

For simplicity, assume messages are generated as follows:

1. Choose a class $S = s$ (spam or not) from the class distribution $P(S)$.
2. Fixing S , sample some W_1 from a class-specific word distribution $P(W_1 | s)$.
3. Repeat from the same distribution, independently, for $W_2, \dots, W_n$.
4. This gives a message at the end.



Our goal is: given a message $w_1, \dots, w_n$, we want to predict its unobserved class s .

Let W_i be the word at position i and $\theta_{w,s} = P(W_i = w | S = s)$. From independence we get:

$$P(w_1, \dots, w_n | s) = \prod_{i=1}^n \theta_{w_i, s}$$

Bag of words

The email is a sequence of word tokens. But our classifier will only need how many times each word appeared. This is the **bag of words** text representation:

$$c = (c_1, \dots, c_V), \quad c_w = \text{count of word } w$$

For “free free winner,” $c_{\text{free}} = 2$, $c_{\text{winner}} = 1$, and every other count is zero.

A **bag of words** keeps these counts and discards order. “Winner free free” gives the same c .

Given this bag, we want the probability that the message is spam:

$$P(S = \text{spam} \mid c)$$

To compute it, we need the probability of observing a bag under each class.

From word sequences to counts

Given one bag of words we see, what emails could have generated it?

The bag for “free free winner” can be generated in three different orders:

Generated sequence	Probability given class s
free, free, winner	$\theta_{\text{free},s}^2 \theta_{\text{winner},s}$
free, winner, free	$\theta_{\text{free},s}^2 \theta_{\text{winner},s}$
winner, free, free	$\theta_{\text{free},s}^2 \theta_{\text{winner},s}$

These are three mutually exclusive ways to observe the same count vector. We add their probabilities:

$$P(c \mid s) = 3\theta_{\text{free},s}^2 \theta_{\text{winner},s}$$

In general, we have $n! / \prod_w c_w!$ ways to generate the word counts $c_1, \dots, c_V$ (if counts sum to n)

Multinomial Naïve Bayes

This leads to the main equation defining our Naïve Bayes classifier.

The general probability of a count vector is therefore:

DEFINITION (MULTINOMIAL NAÏVE BAYES)

$$P(c \mid s) = \frac{n!}{\prod_w c_w!} \prod_{w=1}^V \theta_{w,s}^{c_w}, \quad n = \sum_w c_w.$$

When we vary the class s here (e.g., spam or not), the count vector c the same for both.

So the number of word orders is also the same for every class.

We can thus ignore it, since all it gives us is the normalization constant.

Bayes' rule gives the unnormalized class score:

$$P(s \mid c) = \frac{P(s)P(c|s)}{P(c)} \propto \text{score}(s \mid c) = P(s) \prod_{w:c_w>0} \theta_{w,s}^{c_w}.$$

To obtain posterior probabilities, compute this numerator for each class, add and normalize.

Naïve Bayes training

Where do our θ parameters come from? Or the class prior $P(s)$?

We will *learn them* from N labeled messages in a *training set*.

We can estimate them by counting and normalizing. In the training set, we count:

- N_s : number of training messages with class s
- C_{ws} : occurrences of word w in messages with class s
- $T_s = \sum_w C_{ws}$: total word tokens in class s

Then:

$$\hat{P}(s) = \frac{N_s}{N}, \quad \hat{\theta}_{w,s} = \frac{C_{ws}}{T_s}$$

The prior counts *messages*. The word distributions count *word tokens* (occurrences of each word).

Class	Messages N_s	Tokens T_s	“free”	“meeting”	“winner”
Spam	40	1000	100	5	50
non-spam	60	2000	10	100	2

For long messages, better to multiply many small probabilities in log space:

$$\log \text{score}(s \mid c) = \log P(s) + \sum_{w:c_w>0} c_w \log \theta_{w,s}$$

Naïve Bayes prediction

Now say we receive the message “**free winner**”

From the training counts:

Class	Prior	$P(\text{free} \mid s)$	$P(\text{winner} \mid s)$	Score
Spam	.4	100/1000	50/1000	.002
Non-spam	.6	10/2000	2/2000	.000003

The spam score is larger, we predict spam.

To get probabilities, just normalize the scores:

$$P(S = \text{spam} \mid c) = \frac{.002}{.002 + .000003} \approx .9985$$

Normalization makes the class probabilities sum to one.

Laplace smoothing

Suppose we never saw “GPU” in spam during training, then receive “free GPU now.”

$$C_{\text{GPU},S} = 0 \implies \hat{\theta}_{\text{GPU},S} = 0$$

The whole spam score becomes zero, regardless of the other words.

In log space, $\log 0 = -\infty$

Not seeing a word in the training set shouldn't make it impossible.

We can fix it with *smoothing*

DEFINITION (LAPLACE SMOOTHING)

Add $\alpha > 0$ to every word count in every class:

$$\tilde{\theta}_{w,s} = \frac{C_{ws} + \alpha}{T_s + V\alpha}.$$

The denominator adds $V\alpha$ because we added one α count for each of V words.

If “GPU” also never showed up in non-spam, this just relies on other words (rather than getting $-\infty$ for every class)

In homework 2, you will train and test Naïve Bayes on news classification

Naïve Bayes and assumptions

This is a very efficient and scalable algorithm

Surprisingly accurate for text classification, and good when efficiency matters

For k classes and a dictionary of size V , we only store $O(kV)$ word probabilities.

Training is one linear pass through the labeled data, counting classes and tokens.

With k classes, scoring a message of n tokens takes $O(kn)$.

The model is small and fast because it makes strong assumptions:

- Prediction does not use word order (bag of words)
- The tokens are conditionally independent given the class

In many settings, several variables are (roughly) independent, but the dependencies that exist are all important.

We can drastically generalize the kinds of dependencies we support with Bayesian Networks!

BAYESIAN NETWORKS

An alarm at home

Suppose you're away from home, and your house has an alarm.

The alarm might sound because of a burglary, or because of an earthquake.

Your two neighbors, John and Mary, might call you if they hear the alarm.

Everything is probabilistic! The alarm might fail, John might be away too, etc

Let's model these five events:

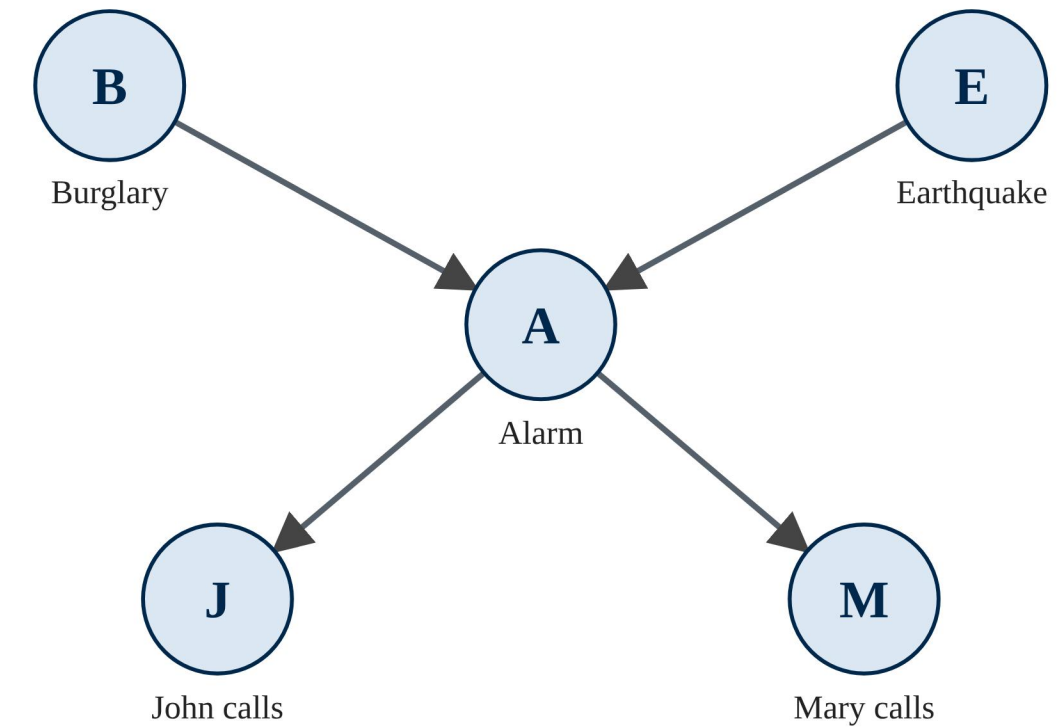
Variable	Event
B	A burglary happens
E	An earthquake happens nearby
A	The alarm sounds
J	John calls you
M	Mary calls you

We might want to ask questions like: if you see a missed call from John, how likely is a burglary?

To build a useful model, let's make a few independence assumptions:

- Burglaries and earthquakes happen independently.
- Either of the two can cause the alarm to sound.
- John's and Mary's call you based on them hearing the alarm.
- John and Mary don't communicate between themselves.

We can represent these direct dependencies in a directed graph like the following:



A Bayesian network

The idea is that this graph will precisely represent our independence assumptions.

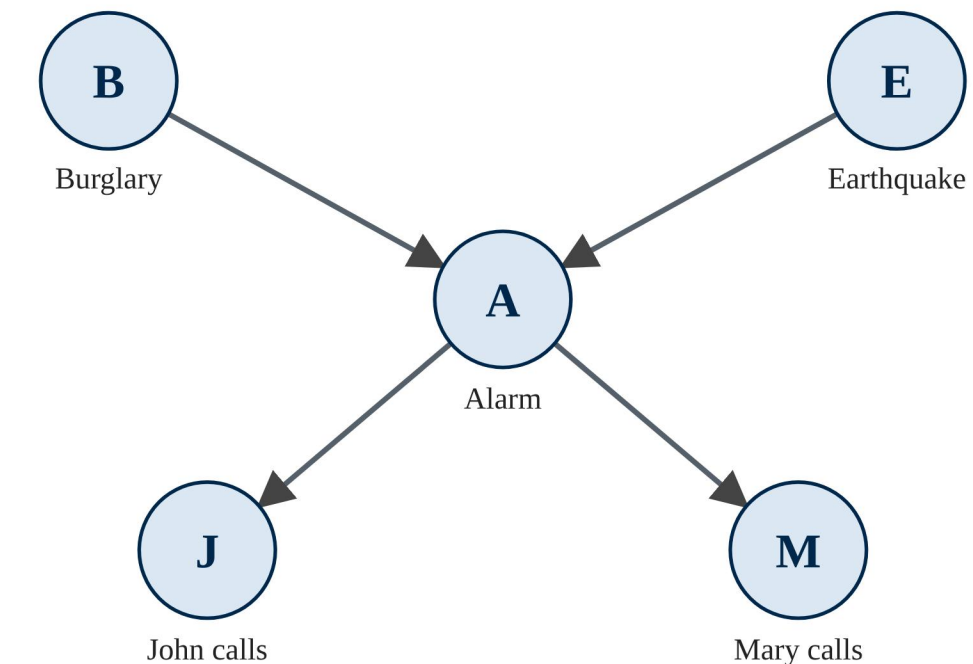
We will describe a large joint distribution through a few local relationships.

A Bayesian network is a precise representation for this idea.

DEFINITION (BAYESIAN NETWORK)

A Bayesian network consists of

1. a directed acyclic graph whose nodes are random variables
2. one conditional distribution $P(X_i \mid \text{Parents}(X_i))$ for every node



Together, these pieces represent the joint distribution

$$P(x_1, \dots, x_n) = \prod_i P(x_i \mid \text{Parents}(X_i)).$$

Note: The arcs only imply a factorization of the joint, not causality necessarily.

Here, $\text{Parents}(A) = \{B, E\}$, $\text{Parents}(J) = \{A\}$, $\text{Parents}(B) = \{\}$, ...

Conditional probability tables

The graph only tells us which variables each local probability depends on.

For each node, we still need a probability distribution for every assignment to its parents.

We organize those numbers in a **conditional probability table**, or CPT.

Here the five variables here are Boolean.

Root probabilities

	Probability
B	.001
E	.002

Roots have no parents, so they need prior probabilities.

Alarm

B	E	$P(A \mid B, E)$
T	T	.95
T	F	.94
F	T	.29
F	F	.001

Calls

A	$P(J \mid A)$	$P(M \mid A)$
T	.90	.70
F	.05	.01

For each fixed parent assignment, the probabilities for the node's possible values must add to one.

The nodes are Boolean, so $P(\neg A \mid b, e) = 1 - P(A \mid b, e)$. The same rule fills the false probabilities in the other tables.

Chain rule for probabilities: $P(X, Y) = P(X)P(Y|X)$

The chain rule always gives, in this topological order:

$$\begin{aligned}
 &P(B, E, A, J, M) \\
 &= P(B)P(E \mid B)P(A \mid B, E) \\
 &P(J \mid B, E, A)P(M \mid B, E, A, J)
 \end{aligned}$$

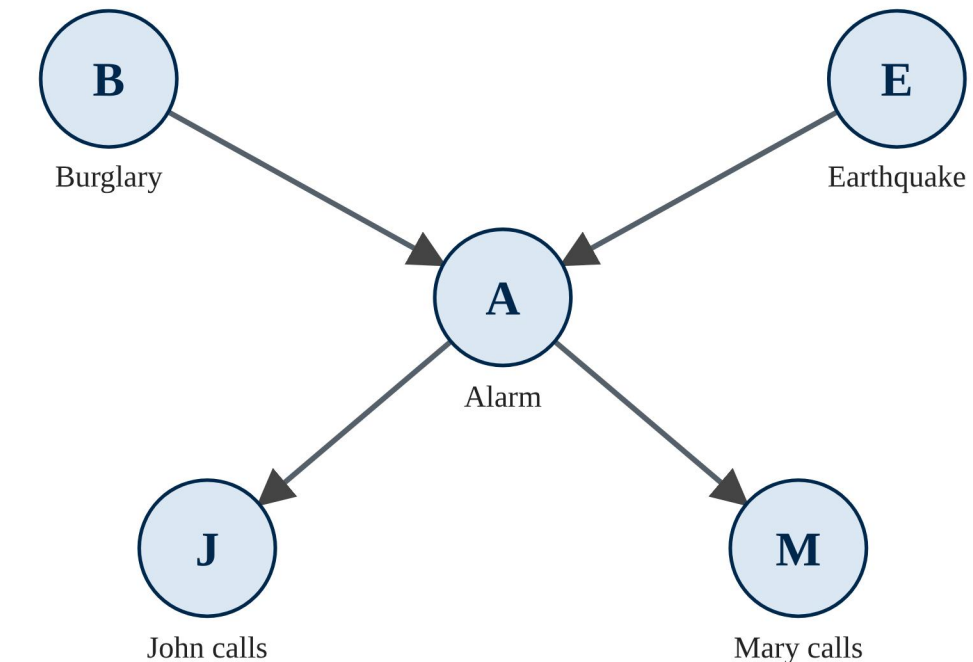
The graph's independence assumptions simplify these terms:

$$P(E \mid B) = P(E)$$

$$P(J \mid B, E, A) = P(J \mid A)$$

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

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



Probability of one world

The network defines the probability of every complete assignment.

For example, consider $b, \neg e, a, j, m$:

$$\begin{aligned} P(b, \neg e, a, j, m) &= P(b)P(\neg e)P(a \mid b, \neg e)P(j \mid a)P(m \mid a) \\ &= .001 \cdot .998 \cdot .94 \cdot .90 \cdot .70 \approx 0.00059 \end{aligned}$$

We selected one entry from each local table to find the probability of this complete assignment.

How compact is it?

Arbitrary joint

Five Boolean variables give $2^5 = 32$ entries.

After normalization:

31 free parameters

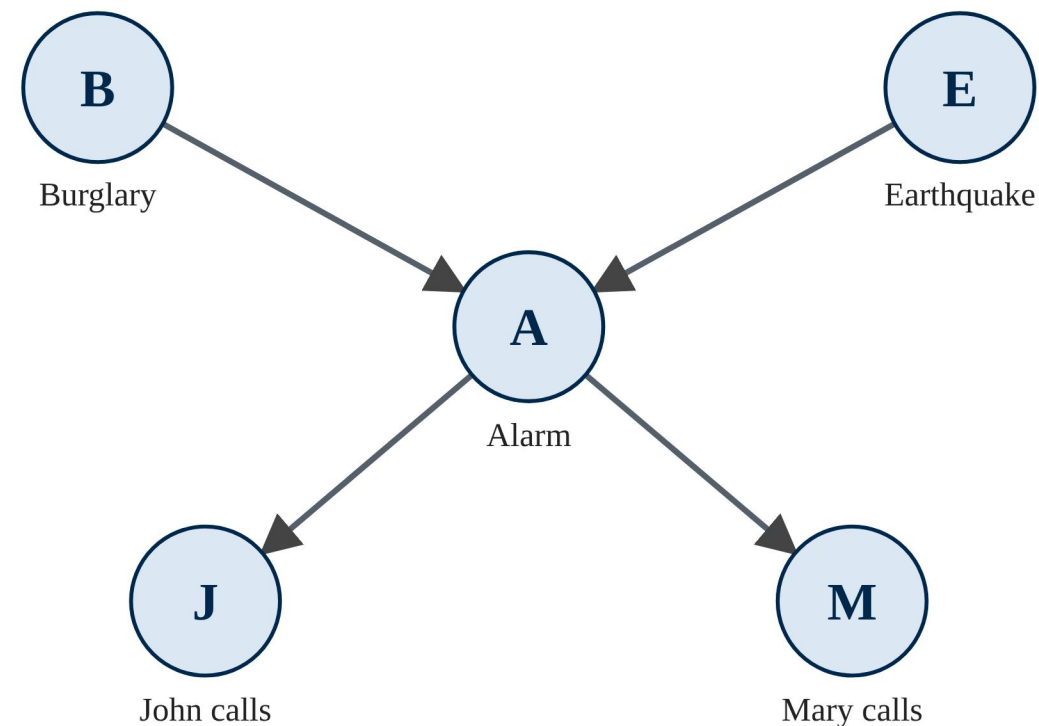
Alarm network

Node	Free parameters
<i>B</i>	1
<i>E</i>	1
<i>A</i>	4
<i>J</i>	2
<i>M</i>	2
Total	10

A Boolean node with k Boolean parents needs 2^k free parameters: one for each parent assignment.

The graph saves space only when nodes have relatively few parents.

One model, many questions



The graph and CPTs specify one joint distribution over all five variables.

We can condition on any observed variables and ask about any others:

- $P(B \mid j, m)$: how likely is a burglary given both calls?
- $P(J \mid b)$: how likely is John to call if there was a burglary?
- $P(A \mid m)$: how likely is the alarm given only Mary's call?
- $P(B \mid a, e)$: how does an earthquake change the burglary probability?

Many successful practical applications: medical diagnosis, fault diagnosis in spacecrafts, Windows troubleshooting helper...

INDEPENDENCE IN BAYESIAN NETWORKS

Implied independences

Given a query in a large network, we want to ignore all the irrelevant variables

But how do we know which variables cannot affect the answer?

There are many properties implied by the independence assumptions in Bayesian Nets that help. Example:

THEOREM (LOCAL MARKOV PROPERTY)

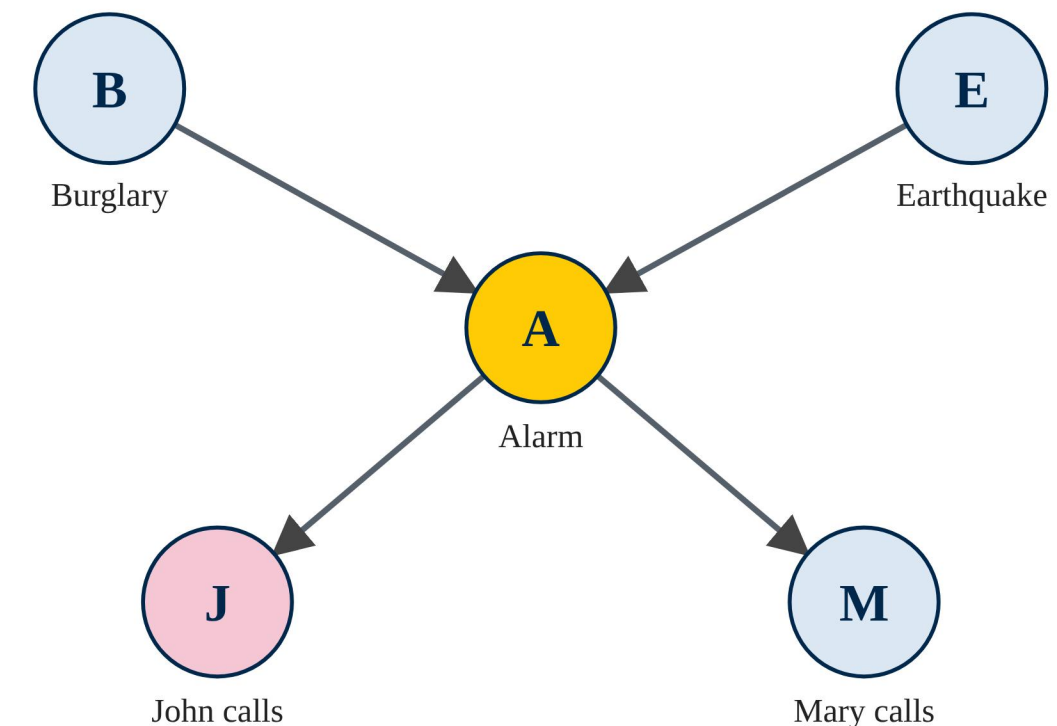
Given its parents, each node is independent of its non-descendants.

Here, a is a descendant of b if we can reach a from b using the arcs (directed edges)

For example:

$$J \perp \{B, E\} \mid A$$

Once we know whether the alarm sounded, learning about a burglary or earthquake gives no more information about whether John will call or not.



How can evidence flow in a path?

Even if variables are far away, they can still influence each other through long paths

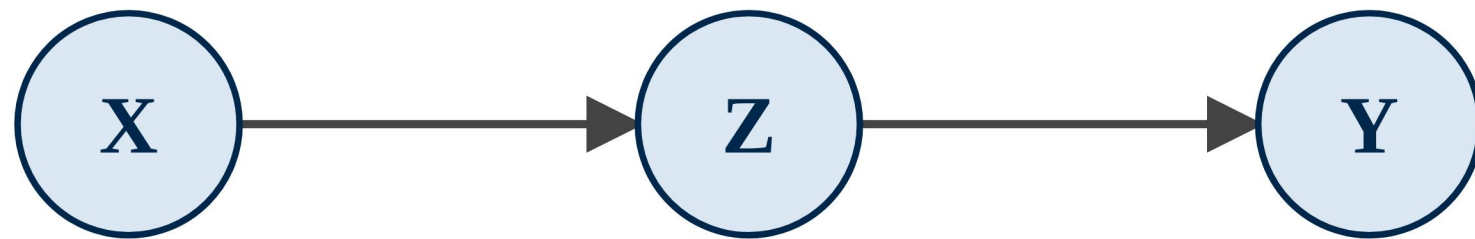
In fact, it's useful to reason about paths regardless of the directions of the edges.

We call such path here a **trail** (e.g., $a \rightarrow b \leftarrow c$ is still a trail between a and c)

An **active** trail allows the endpoints to remain dependent given what we observed. A **blocked** trail does not.

Does observing the middle node of a three-variable trail block it?

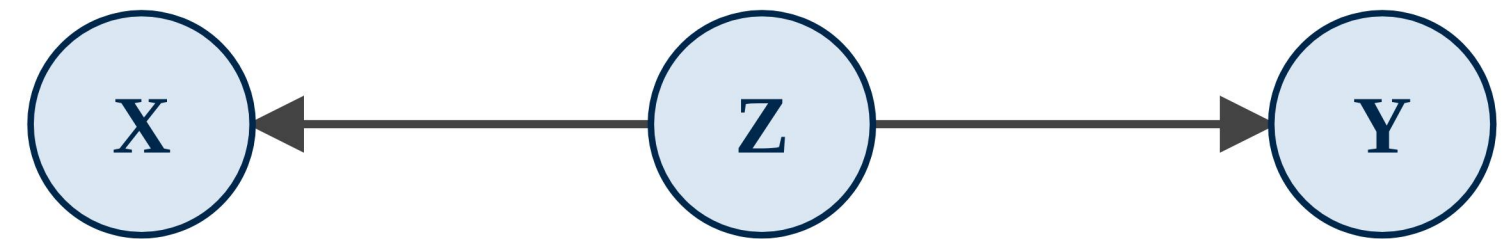
Chain



X can tell us about Y through Z .
After observing Z , the trail is blocked.

$$X \perp Y \mid Z$$

Fork



Z is a shared source of information about X, Y .
After observing Z , the trail is blocked.

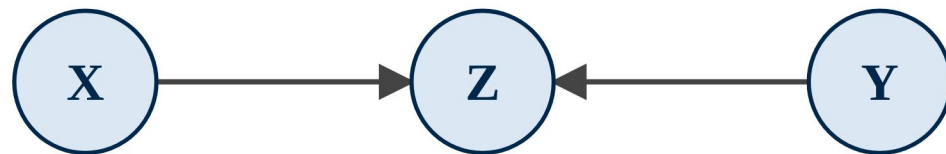
$$X \perp Y \mid Z$$

In the alarm graph, $J \leftarrow A \rightarrow M$ is a fork: knowing A makes the two calls independent.

Colliders behave differently

There is one more way two arrows can meet. This time observing the middle node has the opposite effect.

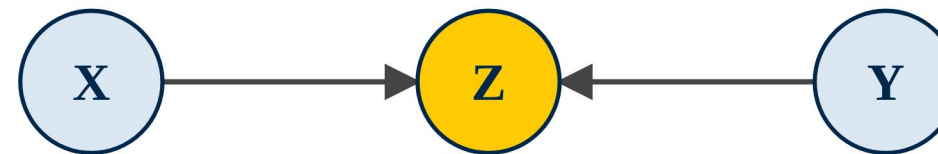
Nothing observed



The trail is blocked.

$$X \perp Y$$

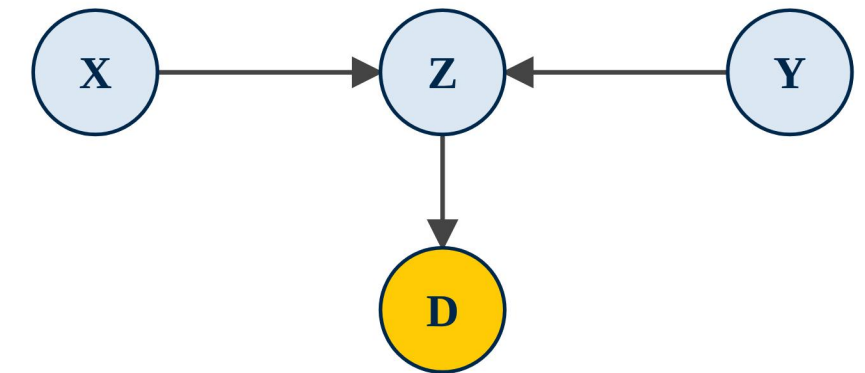
Observe the collider



The trail becomes active.

X and Y can become dependent given Z .

Observe a descendant



This also activates the collider.

For $B \rightarrow A \leftarrow E$, evidence about A , J , or M can make burglary and earthquake dependent.

Consider an undirected trail between two variables. Evidence can make that trail active or blocked.

Chain or fork node

The trail is blocked if the middle node is observed.

Collider node

The trail is blocked unless the collider or one of its descendants is observed.

In general, we will have a set of variables in Q , a set in E we condition on, and we want to know what can't influence Q given E

Beyond individual paths, this will be given by a property called d -separation.

DEFINITION (D-SEPARATION)

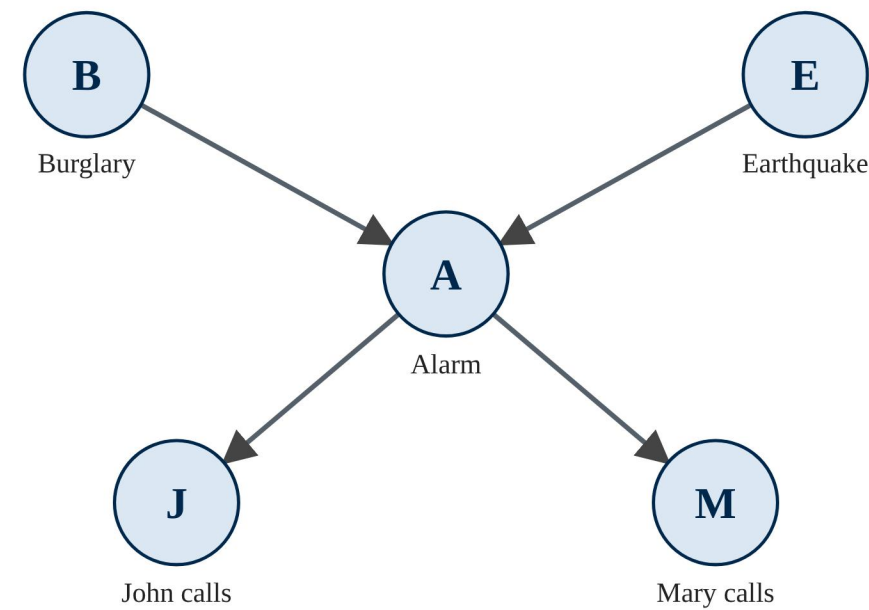
Sets X and Y are d -separated by evidence Z if every trail between them is blocked by Z .

If the graph d -separates X and Y given Z , then every distribution represented by the graph satisfies

$$X \perp Y \mid Z.$$

A particular choice of CPTs may have other independences. d -separation gives us the ones guaranteed by the graph alone.

UNIVERSITY OF
MICHIGAN



Claim	True?	Why?
$B \perp E$	Yes	The only trail, $B \rightarrow A \leftarrow E$, has an unobserved collider.
$J \perp M \mid A$	Yes	Middle of the fork A blocks $J \leftarrow A \rightarrow M$.
$B \perp E \mid A$	No	Observing collider A activates $B \rightarrow A \leftarrow E$.
$B \perp M \mid A$	Yes	Observed middle of the chain A blocks $B \rightarrow A \rightarrow M$.
$B \perp E \mid J$	No	Observed descendant J activates the collider at A .

The Markov blanket

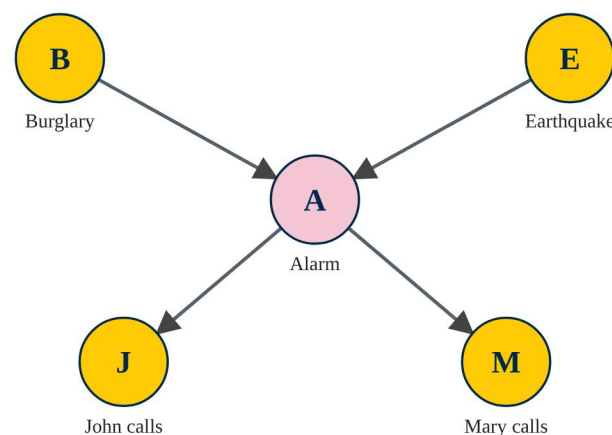
Suppose we want to learn about one variable X . What is the smallest set of observations that makes everything else irrelevant to X ?

The graph identifies a set around X that blocks every trail to the remaining variables.

DEFINITION (MARKOV BLANKET)

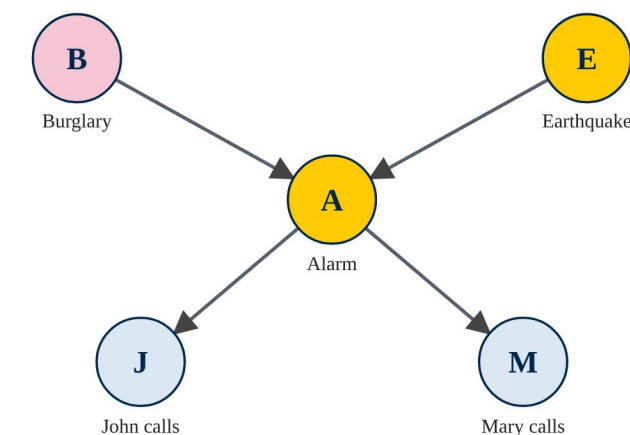
The Markov blanket of X contains its parents, its children, and its children's other parents.

Blanket of A



$$MB(A) = \{B, E, J, M\}$$

Blanket of B



$$MB(B) = \{A, E\}$$

Given its blanket, a node is independent of every other node in the network.

For B , we need co-parent E because observing their common child A can make B and E dependent.

EXACT INFERENCE

What is an inference query?

Let's now actually answer the probabilistic queries algorithmically.

Suppose John and Mary are both calling you. How likely is a burglary?

$$P(B \mid J = \text{true}, M = \text{true})$$

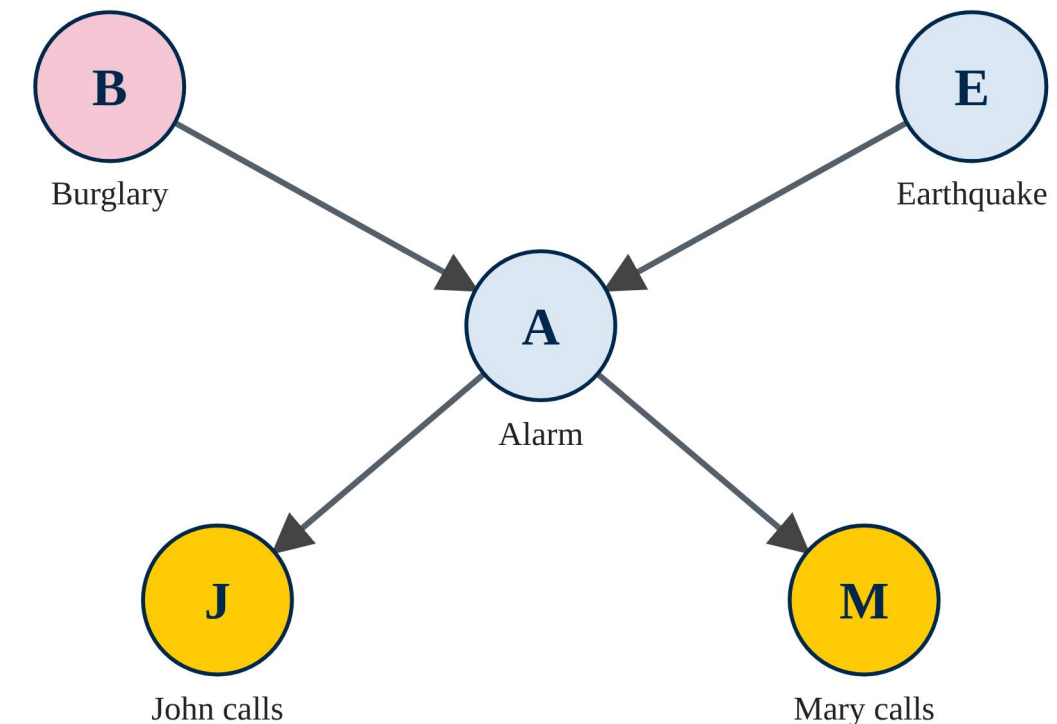
We'll abbreviate a true value with a lowercase letter:

$$P(B \mid j, m)$$

Here B is a random variable, while j denotes the event $\{J = \text{true}\}$.

- **Query:** B
- **Evidence:** $J = j, M = m$
- **Hidden:** E, A

The answer is a normalized distribution over both values of B .



Inference by enumeration

We want $P(B \mid j, m)$. The hidden variables are E, A .

Use the network factorization and sum over every hidden assignment:

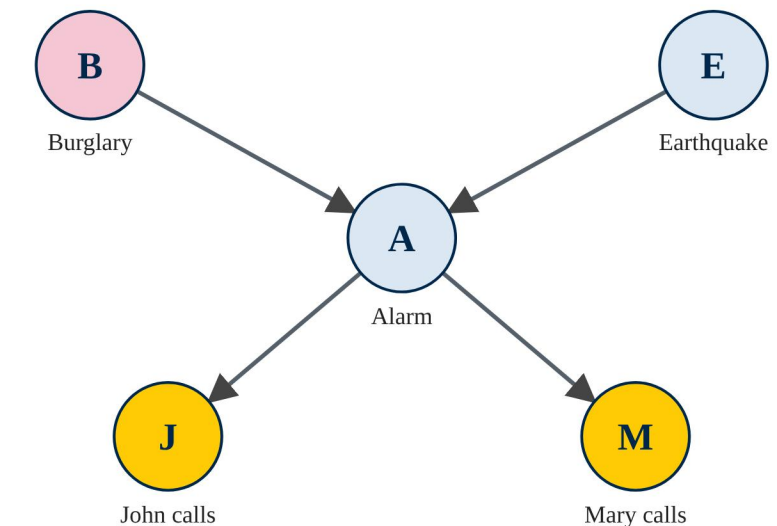
$$P(B \mid j, m) \propto \sum_e \sum_a P(B)P(e)P(a \mid B, e)P(j \mid a)P(m \mid a)$$

For each value of B , this enumerates four assignments to (E, A) .

Normalize the two resulting numerators to get the distribution over B .

This is exact, but it repeats some work:

- $P(j \mid a)P(m \mid a)$ is recomputed for both earthquake values
- Larger networks will repeat the same partial sums lots of times.



Can we save and reuse partial computations?

Query: B **Evidence:** j, m

Our enumeration multiplies and adds entries from the same CPTs many times. Let's save the smaller tables we compute along the way.

These tables may not sum to one, so we need a name more general than “probability distribution.”

DEFINITION (FACTOR)

A factor is a table of nonnegative numbers, with one entry for each assignment to its variables. Its entries need not sum to one.

Each CPT is already a factor. We name them so we can track the tables in the calculation:

$$\begin{array}{c} f_B(B) \\ P(B) \end{array}$$

$$\begin{array}{c} f_E(E) \\ P(E) \end{array}$$

$$\begin{array}{c} f_A(A, B, E) \\ P(A \mid B, E) \end{array}$$

$$\begin{array}{c} f_J(J, A) \\ P(J \mid A) \end{array}$$

$$\begin{array}{c} f_M(M, A) \\ P(M \mid A) \end{array}$$

How will we use these factors to answer $P(B \mid j, m)$?

The inference plan

We want $P(B \mid j, m)$. We will simplify the factors in two phases.

We'll restrict factors using evidence, then multiply factors and sum out hidden variables.

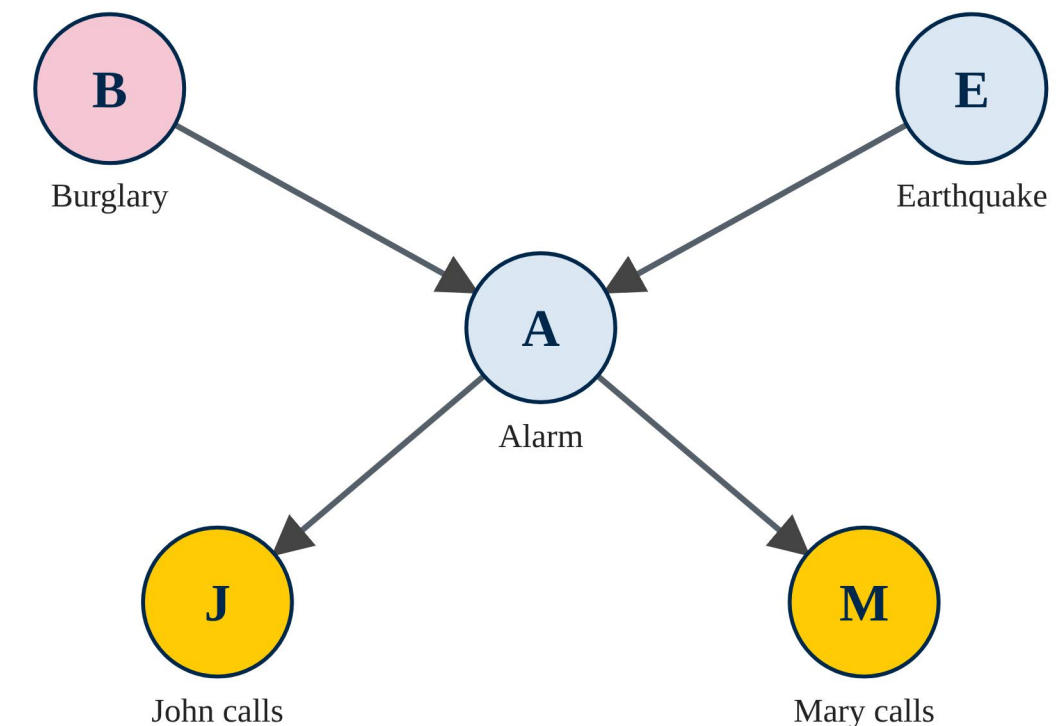
1. Restrict using the evidence

Set $J = j$ and $M = m$ wherever they appear.
Afterwards no factor still varies with J or M .

2. Eliminate the hidden variables one by one

Choose E or A . Multiply the current factors containing it, sum over its values, and replace those factors with the result. Repeat for the other hidden variable.

Finally, we multiply the factors that remain. This gives an unnormalized factor over B . Normalize to get $P(B \mid j, m)$.



Query: B **Evidence:** j, m
Hidden: E, A

Phase 1: restricting

We know $J = M = \text{true}$. In each call table, keep the rows where that call happened.

John called

J	A	$P(J A)$
T	T	.90
T	F	.05
F	T	.10
F	F	.95

This is $f_J(j, A) = P(j | A)$: .90 if $A = \text{true}$, .05 if $A = \text{false}$.

Mary called

M	A	$P(M A)$
T	T	.70
T	F	.01
F	T	.30
F	F	.99

This is $f_M(m, A) = P(m | A)$: .70 if $A = \text{true}$, .01 if $A = \text{false}$.

We call this **restricting a factor by evidence**. The values j, m are fixed, so both smaller factors vary only with A . The other three factors are unchanged. No current factor has J or M as a variable. Phase 1 is done.

Which sum should we do first?

We have used the observed calls. The earthquake E and alarm A are still unknown.

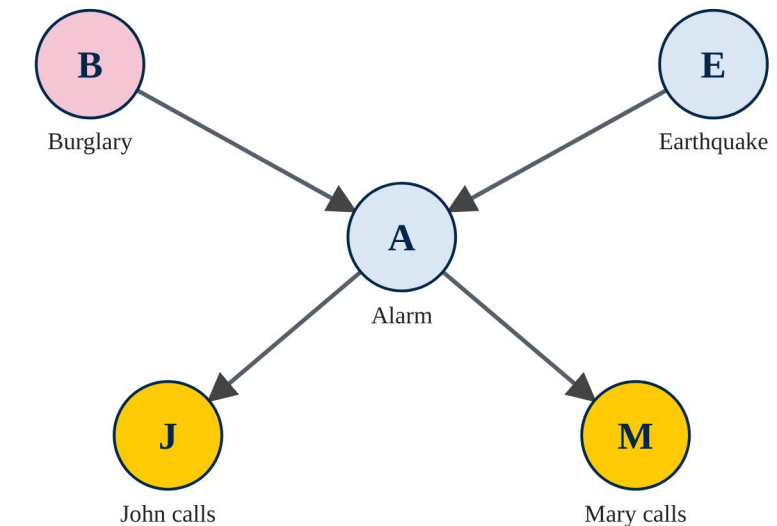
Let's start with E . Only two of our current factors contain it:

$$f_E(E) = P(E), \quad f_A(A, B, E) = P(A \mid B, E)$$

To multiply factors, use entries that agree on shared variables. For each (A, B) , multiply once with $E = \text{true}$ and once with $E = \text{false}$, then add.

$$g(A, B) = \sum_E f_E(E) f_A(A, B, E)$$

Do this for all four choices of (A, B) . The results make a four-entry table $g(A, B)$, with no E left in it.



Query: B Evidence: j, m
Unknown: E, A

Variable elimination: eliminate earthquake

Let's fill the table $g(A, B)$. Start with $A = B = \text{true}$:

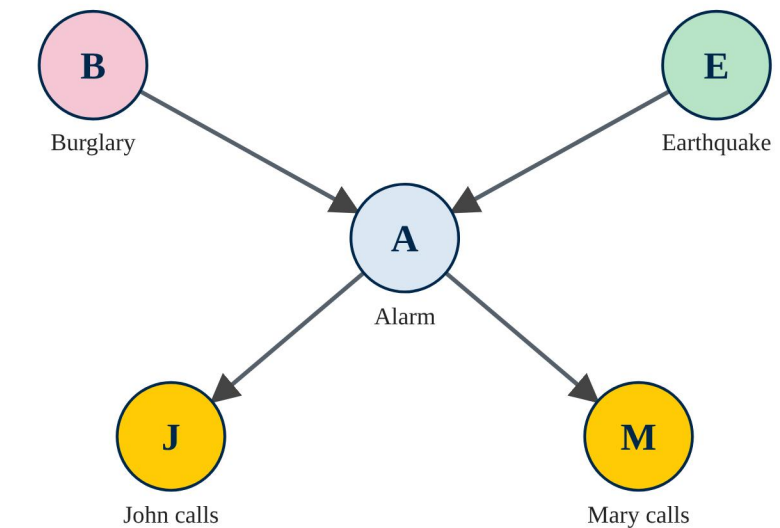
$$g(a, b) = .002(.95) + .998(.94) = .94002$$

The first term uses $E = \text{true}$; the second uses $E = \text{false}$. Repeat for the other three pairs:

A	B	$g(A, B)$
T	T	.94002
F	T	.05998
T	F	.001578
F	F	.998422

This factor $g(A, B)$ replaces both f_E and f_A . Now just need to eliminate A .

Eliminate E



Variable elimination: eliminate alarm

The remaining factors involving A are $g(A, B)$, $f_J(j, A)$, and $f_M(m, A)$.

Eliminate A

For each B , multiply these factors for $A = \text{true}$ and $A = \text{false}$, then add. The two call factors multiply to .63 if $A = \text{true}$ and .0005 if $A = \text{false}$.

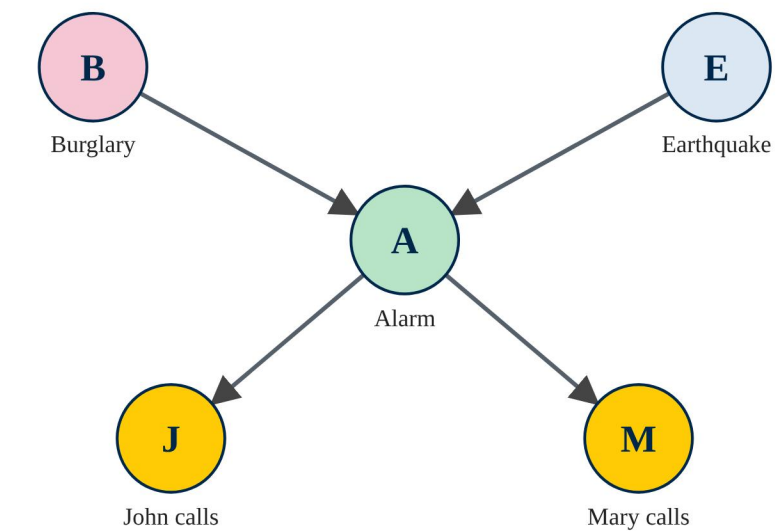
$$\text{score}(B) = f_B(B) \sum_A g(A, B) f_J(j, A) f_M(m, A)$$

B	$A = \text{true}$	$A = \text{false}$	Score
T	.001(.94002)(.63)	.001(.05998)(.0005)	$\approx .000592$
F	.999(.001578)(.63)	.999(.998422)(.0005)	$\approx .001492$

The scores don't yet add to one. Divide each by their sum:

$$P(b \mid j, m) \approx \frac{.000592}{.000592 + .001492} \approx .284$$

Two calls raise the burglary probability from .001 to about 28%.



Variable elimination: the procedure

Our calculation followed the same two phases we planned.

DEFINITION (VARIABLE ELIMINATION)

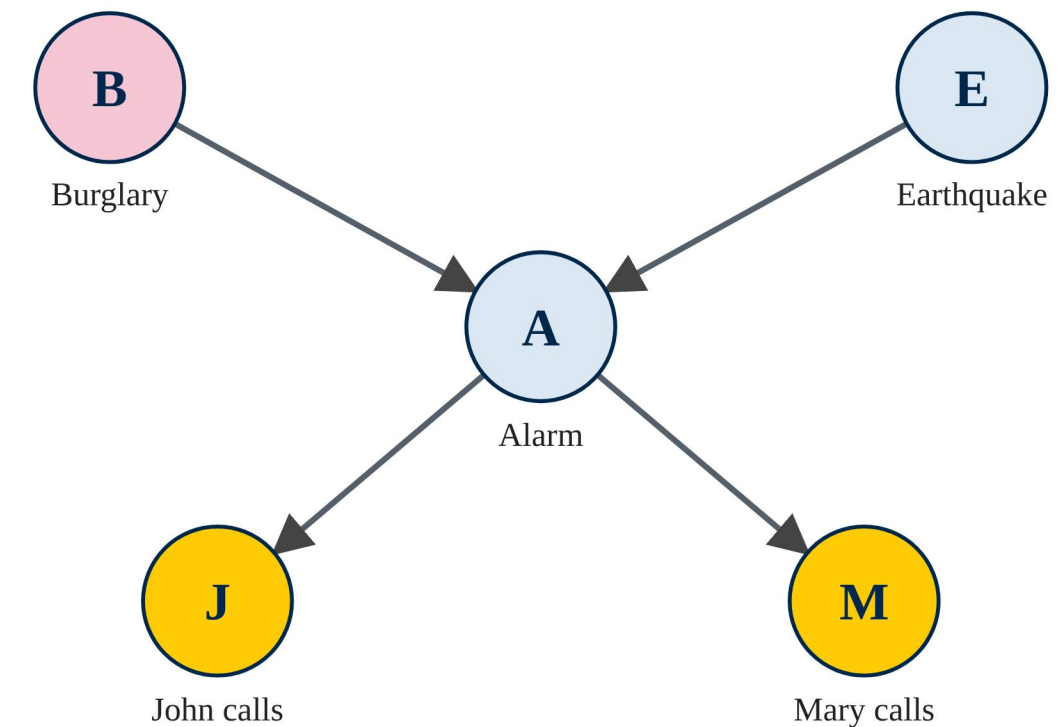
1. **Use evidence:** fix observed values in every factor. No factor still varies with an evidence variable.
2. **Eliminate hidden variables:** choose a hidden variable X . Multiply every current factor containing X , sum over X , and replace those factors with the result. Repeat for each hidden variable.

Multiply all factors left over. The result has only query variables and may not sum to one. Normalize it to get the answer.

Here we eliminated E to make $g(A, B)$, then eliminated A to get two scores for B .

This is exact. The order matters a lot for efficiency!

Evidence j, m fixed. Hidden variables eliminated: E , then A .



Before hearing anything

Variable elimination can answer any query: e.g., how likely is a burglary before any news arrives?

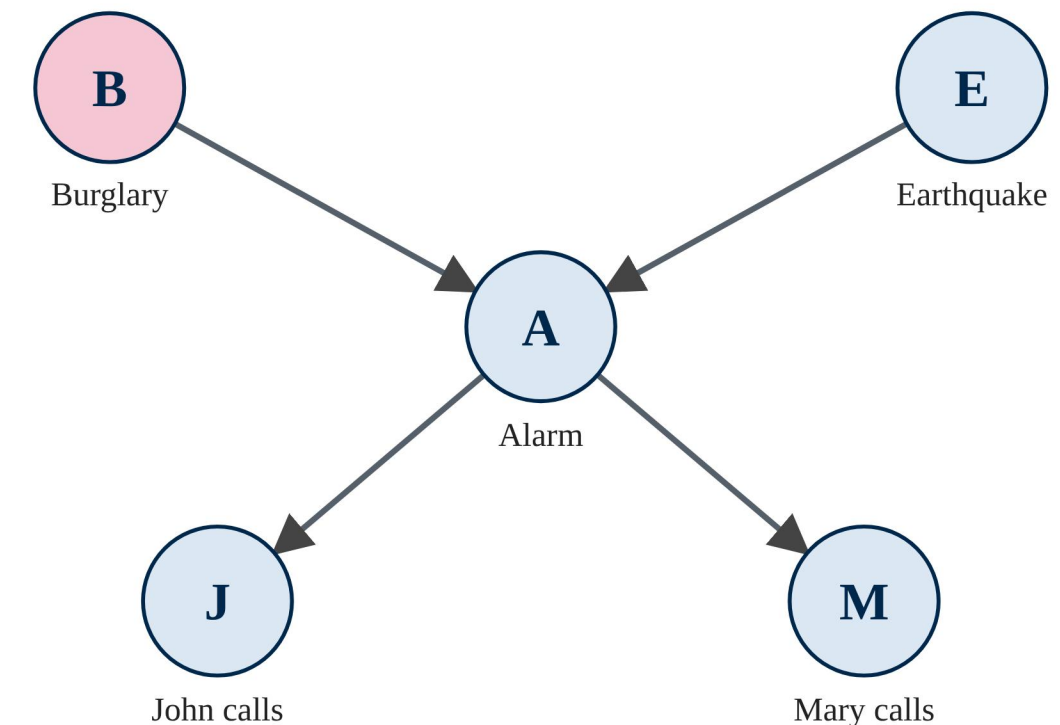
$$P(B) = \sum_{e,a,j,m} P(B, e, a, j, m)$$

We haven't observed either call. For any alarm value A , their CPTs sum to one:

$$\sum_{j,m} P(j \mid A)P(m \mid A) = 1$$

So we can sum out J, M first. Their factors disappear. Then summing out A, E leaves the burglary prior.

$$P(b) = .001$$



The alarm sounds

Now suppose someone other than John and Mary tells you your alarm sounded. Your phone was off, so you don't know if you got a call.

What is $P(b \mid a)$ now?

John's and Mary's calls are still unobserved. Summing them out gives 1 again. The only unknown variable left is E .

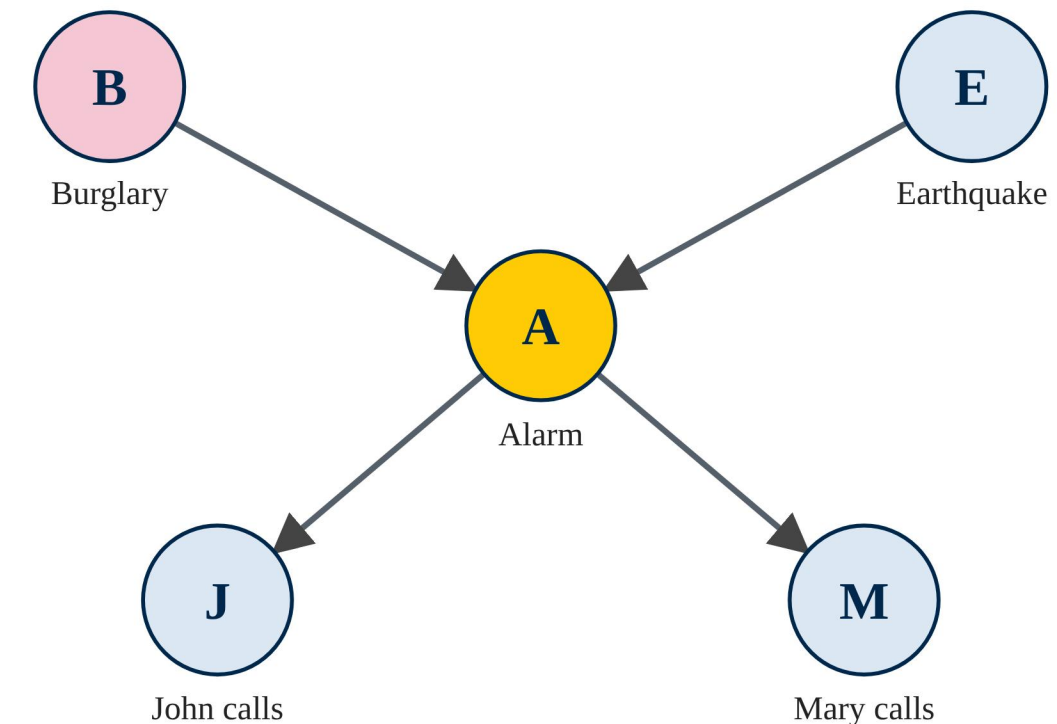
Eliminate the unobserved earthquake:

$$g(a, B) = \sum_e P(e)P(a \mid B, e)$$

Multiply by the burglary prior. The two scores are about .00094 for b and .00158 for $\neg b$.

$$P(b \mid a) = \frac{.00094}{.00094 + .00158} \approx .37$$

The alarm raises burglary from .001 to about .37.



Then an earthquake

A friend tells you there was an earthquake nearby. We now know a and e .

Does this make burglary more or less likely?

The calls are still unobserved and sum out to 1. With A , E known, no hidden variables remain. Compare the two scores:

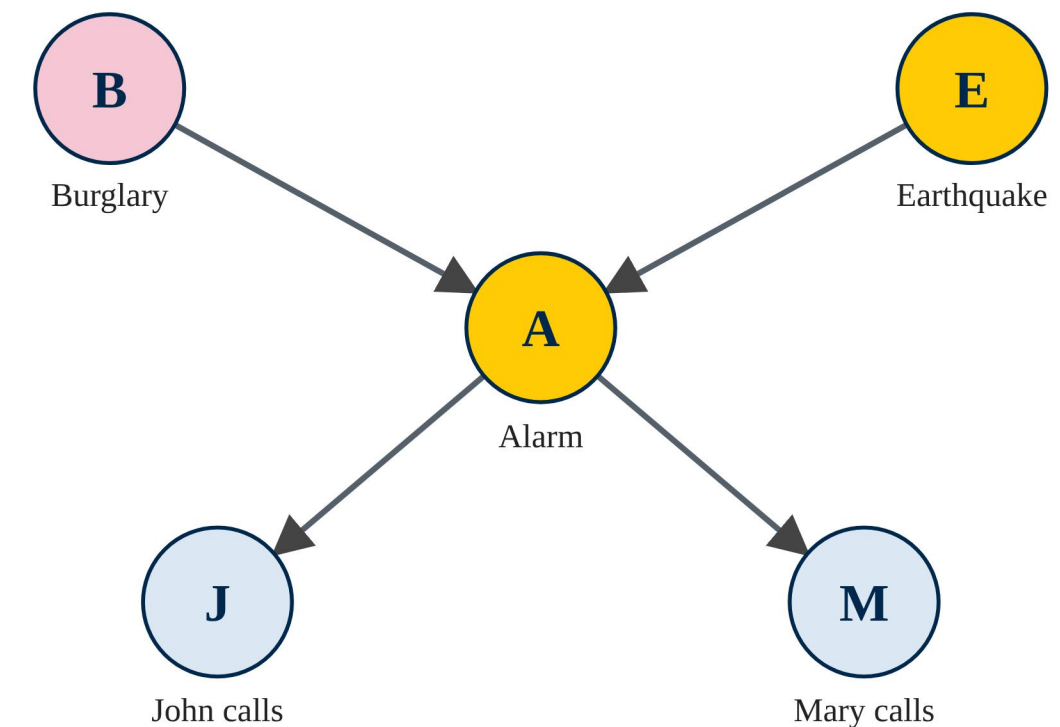
$$\text{score}(b) = .001(.95) = .00095$$

$$\text{score}(\neg b) = .999(.29) \approx .29$$

$$P(b \mid a, e) = \frac{.00095}{.00095 + .29} \approx .0033$$

The earthquake provides another explanation for the alarm. Learning one cause lowers our belief in the other.

This is **explaining away**: conditioning on a shared effect makes its causes dependent.



Ordering, relevance, and complexity

Variable elimination is exact, but its cost depends on the elimination order.

Computing exact probabilities in a general Bayesian network is #P-hard.

Finding the best order is itself hard. There are general heuristics for choosing good orders.

Before eliminating anything, we can remove irrelevant structure:

- Prune nodes that are neither query/evidence nodes nor ancestors of them
- Use d-separation to detect evidence that cannot affect the query.

Exact inference can be expensive in practice. But not the only option!

APPROXIMATE INFERENCE

Probabilities from samples

When exact inference creates factors that are too large, we can estimate probabilities from samples.

If samples $x^{(1)}, \dots, x^{(N)}$ come from $P(X)$, estimate

$$P(X = x) \approx \frac{\#\{i: x^{(i)} = x\}}{N}$$

For a conditional query, we need samples that tell us about the evidence:

$$P(b \mid j, m)$$

How we handle evidence determines whether those samples are useful.

Independent samples

Forward and rejection sampling generate a fresh world each time.

Markov chain samples

Start with one world, then make a small change to get the next.

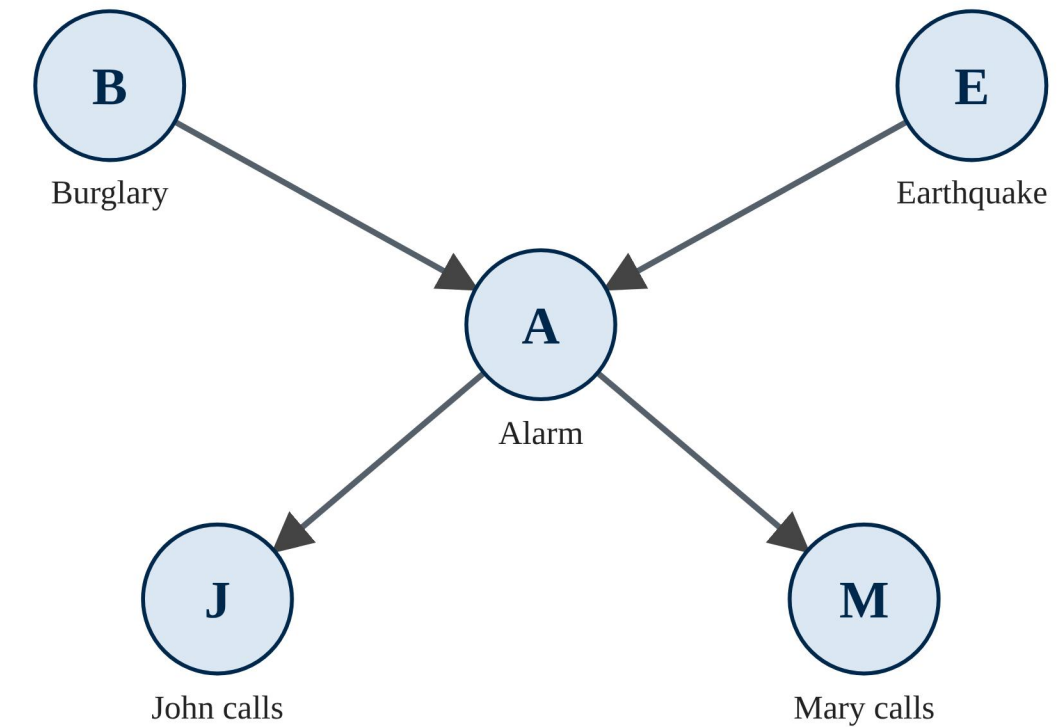
Forward sampling

Visit the nodes in topological order.

At each node, sample from its CPT using the already sampled parent values.

One possible run:

Node	CPT probability	Random r	Value
B	.001	.42	F
E	.002	.001	T
A	.29	.10	T
J	.90	.83	T
M	.70	.76	F



The resulting sample is $(\neg b, e, a, j, \neg m)$.

Repeating this procedure produces samples from the joint distribution represented by the network.

Rejection sampling

To estimate $P(B \mid j, m)$:

1. generate a full sample by forward sampling
2. reject it unless $J = M = \text{true}$
3. among accepted samples, count the fraction with $B = \text{true}$.

Sample	B	E	A	J	M	Use?
1	F	F	F	F	F	reject
2	F	T	T	T	F	reject
3	T	F	T	T	T	accept

Here $P(j, m) \approx .0020841$, so only about two out of every thousand prior samples are not rejected.

Rejection sampling is correct in the limit

But conditioning on rare evidence can make we reject most samples.

Rejection sampling wastes samples when the evidence is rare. Another approach is to keep the evidence fixed and start with one assignment to the other variables.

A **Markov chain** repeatedly changes that assignment. Each new state depends only on the current state.

With j, m fixed, the chain might visit these values of (B, E, A) :

$$(\neg b, \neg e, a) \longrightarrow (\neg b, e, a) \longrightarrow (b, e, a) \longrightarrow \dots$$

The aim is to choose these transitions so that, after enough initial steps, the samples we get approximate the posterior.

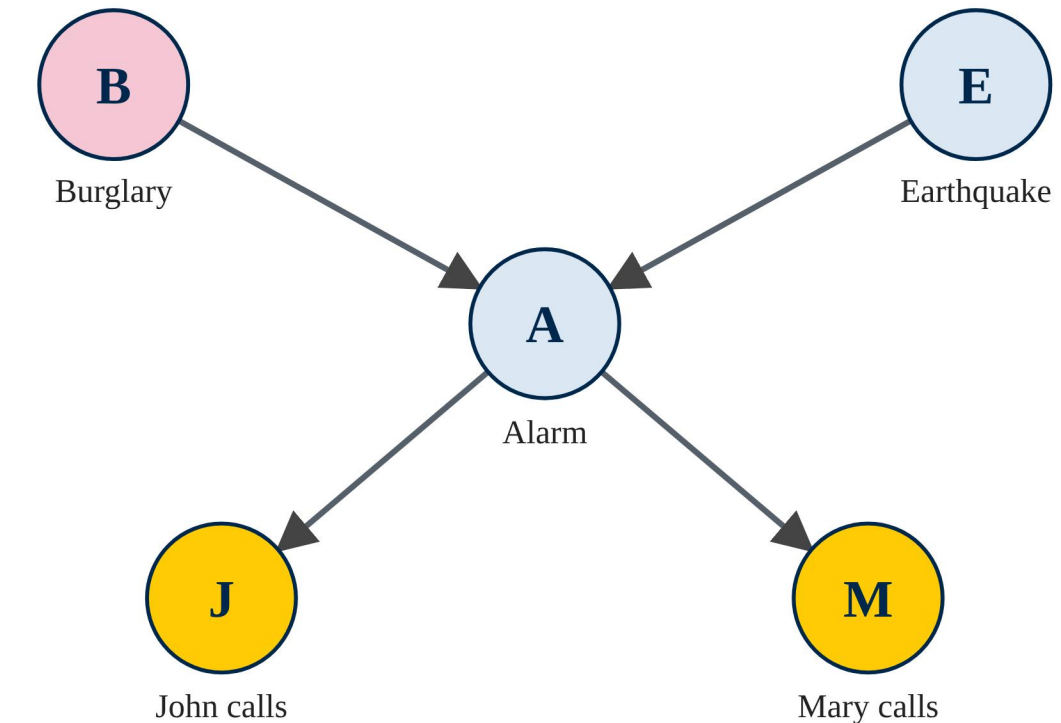
Gibbs sampling gives us one way to make these transitions.

1. Keep $J = M = \text{true}$ fixed and choose initial values for B, E, A .
2. Cycle through B, E, A . Resample one variable at a time using the current values in its Markov blanket.
3. After an initial "burn in" period, estimate $P(b \mid j, m)$ by the fraction of samples with $B = \text{true}$.

At most one variable changes per update.

You'll explore Gibbs sampling in Homework 2 for denoising an image.

MCMC and Gibbs sampling have a very rich theory and many practical uses beyond Bayesian Nets.



One update: resample B using its blanket $\{A, E\}$; keep the other values as they are.

- **Naïve Bayes:** a text classifier assuming conditional independence of words given the class
- **Bayesian network:** a DAG plus local conditional distributions that factor a joint
- **d-separation:** gives the conditional independences guaranteed by the DAG
- **Markov blanket:** the local information needed to update one node
- **Exact inference:** variable elimination multiplies factors and sums out hidden variables
- **Approximate inference:** sampling gives up exact answers but can scale better

Today, we had to specify a dependency structure, hidden variables, etc

Next lecture: We'll start with supervised learning by predicting Y from X directly, without specifying a structure for hidden variables.