Lecture 11: Bayesian Networks

Probabilistic Graphical Models for Uncertainty

Professor Anis Koubaa

SE 444
Alfaisal University

December 1, 2025

Outline

Why Do We Need Bayesian Networks?

A Bayesian network is a map that shows how things influence each other and helps us handle uncertainty in a structured way.

The Problem:

- Full joint distributions are exponentially large
- For n binary variables: 2^n entries
- 20 variables $\rightarrow$ 1,048,576 entries!
- Impossible to store or learn

The Solution:

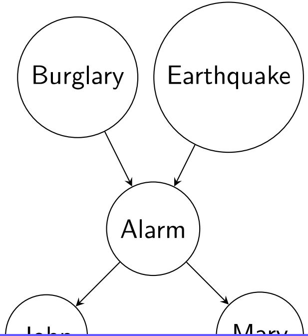
- Exploit conditional independence
- Graph structure shows dependencies
- Complexity: $O(n \cdot 2^k)$ where $k =$ max parents
- Massive reduction

What is a Bayesian Network?

Formal Definition

A Bayesian Network is a pair (G, Θ) where:

- G = Directed Acyclic Graph (DAG)
- Θ = Set of Conditional Probability Tables (CPTs)



The Three Major Advantages

1. **Reduction of Complexity**
   - Only compute direct relationships between variables
   - Breaks huge problem into small local probability relationships
2. **Powerful Inference (Reasoning)**
   - Forward reasoning: Prediction (e.g., "If rain $\rightarrow$ P(wet grass)?")
   - Backward reasoning: Diagnosis (e.g., "If wet grass $\rightarrow$ rain or sprinkler?")
3. **Causal Modeling**
   - Explicitly shows what causes what
   - Ideal for causal reasoning in uncertain environments

The Chain Rule of Probability

General Chain Rule

$$P(X_1, X_2, \ldots, X_n) = P(X_1) \times P(X_2|X_1) \times P(X_3|X_1, X_2) \times \cdots \times P(X_n|X_1, \ldots, X_{n-1})$$

Problem: Each variable is conditioned on all previous variables

Question: Does every variable really depend on *all* previous variables?

Usually not! This is where Bayesian Networks help us simplify.

Bayesian Network Factorization

The Key Simplification

Each variable depends only on its **direct parents**, not on all previous variables:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \text{Parents}(X_i))$$

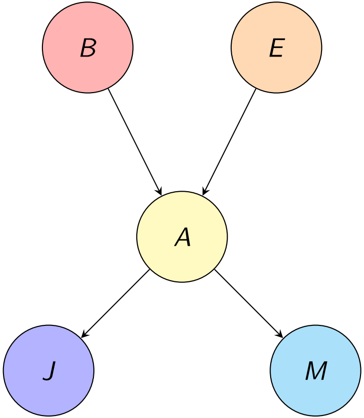
Chain Rule (General):

- $P(X_1)P(X_2|X_1)$
- $P(X_3|X_1, X_2)$
- $P(X_4|X_1, X_2, X_3)$
- Many dependencies!

BN Factorization:

- $P(X_1|\text{Parents}(X_1))$
- $P(X_2|\text{Parents}(X_2))$
- $P(X_3|\text{Parents}(X_3))$
- Only local dependencies!

Example: Burglary-Earthquake-Alarm Network



5 binary variables:
B = Burglary
E = Earthquake
A = Alarm
J = JohnCalls
M = MaryCalls

What is Conditional Independence?

Formal Definition

Two variables X and Y are **conditionally independent** given Z if:

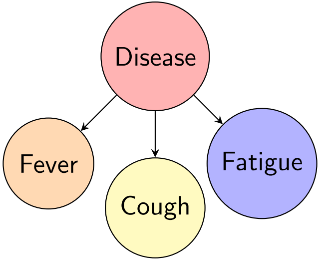
$$X \perp Y \mid Z$$

$$P(X|Y, Z) = P(X|Z)$$

"Once we know Z, learning Y tells us nothing new about X"

Intuitive Explanation: Z screens off the relationship between X and Y. All information that Y provides about X flows through Z.

Real-World Example: Medical Diagnosis



Without Knowing Disease:

- Symptoms are correlated
- If fever, more likely cough
- Symptoms appear together

Given Disease = Influenza:

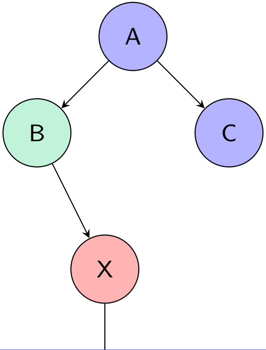
- Fever ⊥ Cough — Disease ✓
- Fever ⊥ Fatigue — Disease ✓
- Symptoms become independent

The Local Markov Property

Fundamental Principle

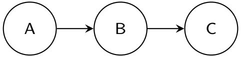
Each node is conditionally independent of all its non-descendants, given its parents.

$$X_i \perp \text{NonDescendants}(X_i) \mid \text{Parents}(X_i)$$



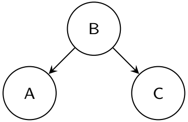
The Three Rules of d-Separation

Rule 1: Chain



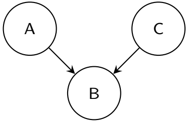
B unobserved: Active
B observed: Blocked

Rule 2: Fork



B unobserved: Active
B observed: Blocked

Rule 3: Collider



B unobserved: Blocked
B observed: Active

Key Insight

Collider rule is opposite! Unobserved collider blocks the path.

The d-Separation Algorithm

Goal: Determine if $X \perp Y \mid Z$

1. **Find All Paths:** Between X and Y (ignoring arrow directions)
2. **Check Each Path:** For every triple of consecutive nodes:
   - Identify if it's a Chain, Fork, or Collider
   - Apply the appropriate blocking rule
3. **Determine Path Status:**
   - Path is BLOCKED if at least one triple is blocked
   - Path is ACTIVE if all triples are active
4. **Final Decision:**
   - If ALL paths blocked $\Rightarrow X \perp Y \mid Z$ ✓
   - If at least one path active $\Rightarrow$ X and Y are dependent

The 5-Step Construction Process

1. **Identify Variables**
   - What can we observe?
   - What do we want to infer?
   - What hidden causes exist?
2. **Choose Variable Ordering**
   - Arrange in causal order
   - Causes before effects
   - Root causes first, observations last
3. **Add Edges (Dependencies)**
   - For each variable, add edges from its direct influences
   - Only add necessary edges
   - Check for cycles (must be DAG)

The 5-Step Construction Process (cont.)

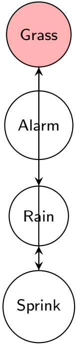
4. **Specify CPTs**
   - From data (if available)
   - From expert knowledge
   - Must sum to 1.0 per row
5. **Validate & Test**
   - Check independencies make sense
   - Test with known scenarios
   - Refine if needed

Pro Tip

Following causal ordering in Step 2 makes Step 3 much easier!

Variable Ordering: The Key to Success

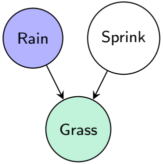
BAD: Random Ordering



Problems:

- Confusing direction
- Many unnecessary edges
- Hard to interpret

GOOD: Causal Ordering



Benefits:

- Intuitive direction
- Minimal edges
- Easy to interpret

The Inference Problem

Central Question: How do you use a BN to answer questions?

Inference

Computing conditional probabilities from the BN:

$$P(\textit{Query} \mid \textit{Evidence}) = ?$$

Query
What we want to know
e.g., "Burglary?"

Evidence
What we observed
e.g., "John & Mary called"

Hidden
What we don't know
e.g., "Earthquake, Alarm"

The Enumeration Algorithm

The Formula

$$P(Q|E) = \frac{P(Q,E)}{P(E)} = \frac{\sum_h P(Q,E,h)}{\sum_{q,h} P(q,E,h)}$$

where:

- Q = Query variable
- E = Evidence (observed)
- h = Hidden variables

Three Steps:

1. Select: Filter entries consistent with evidence
2. Sum Out: Marginalize over hidden variables
3. Normalize: Make probabilities sum to 1.0

Example: P(Burglary — Alarm)

Given: Alarm = true. **Query:** P(Burglary = true — Alarm = true)?

Step 1: Compute Numerator

$$P(B=T, A=T) = \sum_{e} P(B=T, A=T, e)$$
$$= P(B=T, A=T, E=T) + P(B=T, A=T, E=F)$$
$$= 0.0000019 + 0.00093812$$
$$= 0.00094002$$

Step 2: Compute Denominator

$$P(A=T) = \sum_{b,e} P(b, A=T, e) = 0.00251644$$

Step 3: Normalize

$$P(B=T|A=T) = \frac{0.00094002}{0.00251644} = 0.374 \; (37.4\%)$$

Interpretation

Result

Even though the alarm went off, there's only a **37.4% chance of burglary**.

Why?

- Burglaries are rare (0.1
- Alarm can be triggered by other causes:
  - Earthquake
  - Random false alarms
- Must consider base rates and alternative explanations

Key Lesson

Bayesian inference properly accounts for prior probabilities and multiple explanations!

The Best Intuition: VE = Cleaning Up Before You Calculate

VE is like simplifying an algebra expression before computing it

Enumeration (Brute Force):

Expand everything first

$$(a + b)(c + d)(e + f)$$

$$\downarrow$$

$$= ace + acf + ade + adf$$
$$+bce + bcf + bde + bdf$$

8 terms to compute!

Variable Elimination (Smart):

Simplify & eliminate first

$$(a + b)(c + d)(e + f)$$

$$\downarrow$$

Let $x = (e + f)$
Let $y = (c + d)x$
Result $= (a + b)y$

Compute once, reuse!

Key Insight

VE is literally the "factor & simplify" trick from high school algebra, but applied to

From Enumeration to Variable Elimination

Example Query: Compute $P(B|A)$ in network $A \to B \to C \to D$

The 4-Step Transformation:

1. **Enumeration:** $P(B, A) = \sum_{C,D} P(A, B, C, D)$
   - Sum over all combinations of C and D
2. **Factor by Chain Rule:** $= \sum_{C,D} P(A) \cdot P(B|A) \cdot P(C|B) \cdot P(D|C)$
   - Break joint into conditional probabilities
3. **Push Summations Inside:** $= P(A) \cdot P(B|A) \cdot \sum_C P(C|B) \cdot [\sum_D P(D|C)]$
   - Group terms that don't depend on outer variables
4. **Result:** $= f_1 \times f_2 \times f_3$
   - Each f is a factor after eliminating one variable

The Variable Elimination Algorithm

Goal: Compute $P(Q|e)$ efficiently by eliminating variables one at a time

Step 1: Initialize Factors & Set Evidence

Create one factor for each CPT, and instantiate evidence

Example: For $A \rightarrow B \rightarrow C$

- $f_A = P(A)$
- $f_B = P(B|A)$
- $f_C = P(C|B)$

Benefit: Each factor represents a piece of the joint probability that we'll combine strategically

The VE Algorithm (cont.)

Step 2: Eliminate Hidden Variables (One at a Time)

For each hidden variable X:

1. **Multiply** all factors that mention X
2. **Sum out** X from the product
3. Store the result as a new factor

Mini Example: Eliminating C from $f_1(B, C)$ and $f_2(C, D)$:

$$f_{new}(B, D) = \sum_C [f_1(B, C) \times f_2(C, D)]$$

Step 3: Multiply & Normalize

1. Multiply all remaining factors
2. Normalize to get probability distribution: $P(Q|e) = \alpha \cdot [f_1 \times f_2 \times \ldots]$

Complete Example: P(Flu — Fever=Yes)

Network: Flu $\rightarrow$ Fever, Flu $\rightarrow$ Cough

Given CPTs:

Flu	P
Yes	0.1
No	0.9

Flu	P(Fever=Yes)
Yes	0.8
No	0.2

Query: $P(Flu|Fever = Yes)$

Evidence: Fever = Yes
Hidden: Cough

Step 1: Initialize Factors & Set Evidence

$f_1 = P(Flu)$

Flu	Value
Yes	0.1
No	0.9

$f_2 = P(Fever = Yes|Flu)$

Evidence set!

Flu	Value
Yes	0.8
No	0.2

$f_3 = P(Cough|Flu)$

Flu	Cough	Value
Yes	Yes	0.7
No	Yes	0.3

Note

We set Fever=Yes in factor f_2, so it becomes a function only of Flu

Step 2: Eliminate Cough (Hidden Variable)

Only f_3 contains Cough, so we sum it out:

$$f_4(Flu) = \sum_{Cough} [f_3(Cough, Flu)]$$

Result

$f_4(Flu) = 1$ for all Flu values (probabilities sum to 1)

Intuition: Since Cough is not observed and not in the query, summing over all its values gives us 1.0

Steps 3 & 4: Multiply & Normalize

Step 3: Multiply Remaining Factors

$$P(Flu, Fever = Yes) = f_1(Flu) \times f_2(Flu) \times f_4(Flu)$$

Flu=Yes:

$$P(Flu) \times P(Fever = Yes|Flu) \times 1$$
$$= 0.1 \times 0.8 \times 1 = 0.08$$

Flu=No:

$$P(Flu) \times P(Fever = Yes|Flu) \times 1$$
$$= 0.9 \times 0.2 \times 1 = 0.18$$

Step 4: Normalize

$$P(Flu = Yes|Fever = Yes) = \frac{0.08}{0.08+0.18} = \mathbf{0.308\ (30.8\%)}$$

Why Variable Elimination Wins

Enumeration:

- Complexity: $O(2^n)$ — exponential in ALL variables
- Repeats same calculations many times
- Simple but extremely inefficient

Variable Elimination:

- Complexity: $O(2^w)$ — exponential in tree-width only
- Computes each sub-expression exactly once
- Much more efficient in practice!

Key Takeaway

Elimination ordering matters!

Good ordering can make the difference between tractable and intractable inference.

Summary: Why Bayesian Networks Matter

1. **Compact Representation**
   - Reduce exponential to manageable complexity
   - Store only local dependencies
2. **Principled Reasoning**
   - Grounded in probability theory
   - Handle uncertainty systematically
   - Support both prediction and diagnosis
3. **Causal Interpretation**
   - Graph shows causal relationships
   - Intuitive and interpretable
   - Mirror how humans think about causation
4. **Practical Applications**
   - Medical diagnosis, spam filtering, robotics
   - Decision support systems
   - Risk assessment, fault diagnosis

Key Takeaways

What You Should Remember

1. BN = DAG + CPTs
2. Factorization: $P(X_1, \ldots, X_n) = \prod_i P(X_i | \text{Parents}(X_i))$
3. Conditional independence is the key to compact representation
4. d-Separation: 3 rules (Chain, Fork, Collider)
5. Construction: causal ordering is critical
6. Inference: enumeration is exact but expensive

Questions?

References & Further Reading

- Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.
- Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.
- Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

Online Resources:

- Course materials: SE444 Lecture 11
- Interactive demos available on course website