Fundamentals of Artificial Intelligence and Knowledge Rapresentation - Module 3

Chiara Sivieri

December 2025

Contents

1 Introduction to uncertainty and probabilistic reasoning

1.1 Introduction and Logistics

1.1.1 Problem-solving logical agents

The limitation of Logic

We begin by considering standard problem-solving logical agents. Typically, these agents maintain a belief state and attempt to generate a contingency plan to reach a goal. However, when applied to real-world scenarios, handling uncertainty creates significant obstacles. The belief-state representations become prohibitively large and complex, and contingency plans can grow arbitrarily large. More critically, in uncertain environments, there may be simply **no plan** that is guaranteed to achieve the goal.

Puzzle 1: The Switchboard

To illustrate the limits of logical reasoning, consider a switchboard with three switches: A, B, and C. The rules are as follows: 1. When A is on, B is also on. 2. When C is on, B is off. The question posed is: "If both A and C are on, the switchboard melts down. Can that ever happen?" Logic struggles here because it detects a contradiction (B must be both on and off) but cannot easily model the likelihood or the temporal dynamics of such an event without complex extensions.

Puzzle 2: The 12 Coins

Consider a scenario with 12 coins distributed in piles of 4, 1, and 7 coins. We are allowed to move coins from a pile A to a pile B, but with a constraint: we can only move coins to pile B if we double the number of coins currently in B. The question is: "Can we distribute coins evenly among the three piles?" This is a deterministic search problem, but it highlights the complexity of state-space search which becomes unmanageable when uncertainty is added.

Puzzle 3: The Monty Hall Problem

We are guests on a TV game show standing in front of three closed doors. A prize hides behind one of them. We choose the door on the left. At this point, the host—who knows where the prize is—opens the middle door to reveal it is empty. We are then offered the chance to modify our choice. Should we? This puzzle demonstrates that new information changes probabilities, a concept that classical logic does not handle naturally.

1.1.2 Handling uncertainty

Why uncertainty exists

Agents are forced to handle uncertainty primarily due to three factors:

- **Partial observability**: The agent cannot perceive the entire state of the world (e.g., the state of the road ahead).

- **Nondeterminism**: Actions do not always have the intended effect (e.g., trying to drive might fail due to engine trouble).
- A combination of both.

Application areas

Uncertainty management is central to many fields in AI, including Robotics, Medical diagnosis, Troubleshooting, Decision-making, Risk assessment, Automated monitoring, Predictions, Image/speech synthesis and recognition, Computational biology, and Economics.

1.2 Acting under uncertainty

1.2.1 The Airport Scenario

The Goal

Consider a classic planning problem: we need to reach the airport on time. Let us define the action A_t as "leaving for the airport t minutes before the flight." The fundamental question is: *Will A_t get me there on time?*

Problems

Answering this question with certainty is impossible due to: 1. **Partial observability** (we do not know the plans of other drivers). 2. **Noisy sensors** (traffic reports may be inaccurate). 3. **Uncertainty in action outcomes** (unexpected events like a flat tire). 4. The **immense complexity** of modeling and predicting traffic flow.

Failure of Logic

If we rely on a purely logical approach, we face a dilemma. We either: 1. **Risk falsehood** by asserting "A_{25} will get me there on time," which might turn out to be false. 2. **Reach weak conclusions** that are useless for decision making, such as "A_{25} will get me there on time *if* there is no accident, *and* it doesn't rain, *and* my tires remain intact, etc." Even an extreme action like A_{1440} (leaving 24 hours early) might guarantee arrival, but it leads to the absurd outcome of sleeping overnight at the airport.

1.2.2 Methods for handling uncertainty (M2)

Default or nonmonotonic logic

Historically, one approach was to use default logic, which relies on assumptions. For example, we might "assume my car does not have a flat tire" or "assume A_{25} works unless contradicted by evidence." The critical failure of this method is determining exactly *what* assumptions are reasonable to make.

Rule-based systems with fudge factors

Another attempt involved attaching numerical confidence values ("fudge factors") to rules:

- $A_{25} \rightarrow_{0.3} AtAirportOnTime$

- Causal reasoning: $FaultyPowerCord \rightarrow_{0.99} DisplayOff$
- Diagnostic reasoning: $DisplayOff \rightarrow_{0.7} SleepMode$

This approach fails due to problems with **locality** (how can a local number like 0.3 account for all global evidence?) and **combination** (if a faulty cord causes the display to turn off, and a display being off suggests sleep mode, does a faulty cord cause sleep mode? The math often leads to incorrect conclusions).

1.2.3 Probability

The Probabilistic Approach

The modern solution is Probability. Instead of true/false, we state: "Given the available evidence, A_{25} will get me there on time with probability 0.04."

Remark on Fuzzy Logic

It is crucial to distinguish Probability from Fuzzy Logic. Fuzzy logic deals with **degrees of truth**, not uncertainty.

- **Fuzzy Logic**: "$TrafficCongested$ is true to degree 0.8" means the traffic is heavy (a fact about the world).
- **Probability**: "There is a 0.8 probability of traffic" means we are 80% sure there is traffic (a fact about our belief).

1.2.4 Probability and decision making

Subjective Probability

Probabilistic assertions summarize uncertainty arising from **laziness** (it is too much work to list all exceptions) and **ignorance** (we lack all the facts). We use **Subjective or Bayesian probability**, where probabilities relate propositions to the agent's own state of knowledge. For example, $P(A_{25}|\text{no reported accidents}) = 0.06$. These are not claims about the physical world, but about our beliefs. As such, probabilities change when new evidence arrives: $P(A_{25}|\text{no reported accidents, 5 a.m.}) = 0.15$.

Making decisions

Suppose I hold the following beliefs regarding leaving times:

- $P(A_{25} \text{ success}) = 0.04$
- $P(A_{90} \text{ success}) = 0.70$
- $P(A_{120} \text{ success}) = 0.95$
- $P(A_{1440} \text{ success}) = 0.9999$

Which action should I choose? The answer depends on my **preferences** (e.g., how much I dislike missing the flight versus how much I dislike eating bad airport food). To be **Rational**, an agent must follow the principle of **Maximum Expected Utility (MEU)**. Thus, **Decision Theory** is defined as Utility Theory combined with Probability Theory.

1.3 Basic probability notation

1.3.1 Probability basics

Sample Space and Events

While logical assertions simply tell us which worlds are impossible, **probabilistic assertions** tell us how probable the remaining worlds are. The set of all possible worlds is called the **sample space**, denoted by Ω.

- Any subset $A \subseteq \Omega$ is called an **event**.
- Any single element $\omega \in \Omega$ is called a **sample point**, possible world, or atomic event.

Example: If we roll a die, the sample space is $\{1, 2, 3, 4, 5, 6\}$. The event "die roll ¡ 4" corresponds to the set $\{1, 2, 3\}$.

Probability space

A probability model assigns a number $P(\omega)$ to every sample point ω such that: 1. $0 \leq P(\omega) \leq 1$ 2. The sum of all probabilities is 1: $\sum_\omega P(\omega) = 1$. The probability of any event A is simply the sum of the probabilities of the sample points contained in it: $P(A) = \sum_{\omega \in A} P(\omega)$. For a fair die, $P(1) = \cdots = P(6) = 1/6$. Therefore, $P(\text{die roll} < 4) = 1/6 + 1/6 + 1/6 = 1/2$.

1.3.2 Random variables

Definition

A **random variable** is formally defined as a **function** that maps sample points to some range, such as the real numbers or Boolean values. For example, Odd is a random variable where $Odd(1) = true$, $Odd(2) = false$, etc. The probability distribution for a random variable X is induced by the underlying sample points: $P(X = x_i) = \sum_{\omega : X(\omega) = x_i} P(\omega)$.

1.3.3 Propositions

Logic and Events

We can think of a proposition as the event (set of sample points) where that proposition is true. Given Boolean random variables A and B:

- Event a is the set of points where $A(\omega) = true$.
- Event $a \wedge b$ is the set of points where both $A(\omega) = true$ and $B(\omega) = true$.

A proposition is equivalent to the disjunction of the atomic events in which it holds true. For instance: $P(a \vee b) = P(\neg a \wedge b) + P(a \wedge \neg b) + P(a \wedge b)$.

Syntax for propositions

Variables can be of different types:

- **Boolean**: e.g., $Cavity$. The proposition $Cavity = true$ is often shortened to just $cavity$.

- **Discrete**: e.g., $Weather$ with domain $\langle sunny, rain, cloudy, snow \rangle$. The values must be exhaustive and mutually exclusive.
- **Continuous**: e.g., $Temp = 21.6$.

1.3.4 Distributions

Prior probability

Prior or unconditional probabilities represent the agent's belief *before* any evidence arrives. Example: $P(Cavity = true) = 0.1$. A **probability distribution** lists the values for all possible assignments. For example, the distribution for Weather might be $\mathbf{P}(Weather) = \langle 0.72, 0.1, 0.08, 0.1 \rangle$. This vector is normalized (sums to 1).

Joint Probability Distribution

The **Joint Probability Distribution (JPD)** is the most comprehensive representation. For a set of random variables, it gives the probability of every atomic event (every combination of values). Example: $\mathbf{P}(Weather, Cavity)$ is a 4×2 matrix:

Weather	**sunny**	**rain**	**cloudy**	**snow**
$Cavity = true$	0.144	0.02	0.016	0.02
$Cavity = false$	0.576	0.08	0.064	0.08

Key Concept: Every question about a domain can be answered by the joint distribution because every event is just a sum of sample points (entries in this table).

Probability for continuous variables

For continuous variables, we cannot list probabilities for every point. Instead, we use a **probability density function (pdf)**, $p(x)$, such that the integral over the domain is 1. Common distributions include:

- **Uniform distribution**: $Unif[a, b](x) = \frac{1}{b-a}$.
- **Gaussian (Normal) distribution**: Defined by mean μ and variance σ^2.

Note that for a continuous variable, $P(X = 20.5) = 0.125$ actually refers to the limit of the density over an infinitesimal interval dx.

1.3.5 Conditional probability

Definition

Conditional (or posterior) probability, denoted $P(X|Evidence)$, represents the updated belief after observing new Evidence. Example: $P(cavity|toothache) = 0.8$. This means: "Given that toothache is the **only** information I have, the probability of a cavity is 0.8." If we acquire more information (e.g., we see the cavity), the probability updates to 1. Note that new evidence can also be irrelevant (e.g., knowing the "49ers won" does not change the probability of a cavity).

Mathematical Rules

The definition is given by:

$$P(a|b) = \frac{P(a \wedge b)}{P(b)} \quad \text{(provided } P(b) \neq 0\text{)}$$

From this, we derive the **Product Rule**: $P(a \wedge b) = P(a|b)P(b)$. This allows us to decompose a joint distribution using the **Chain Rule**:

$$\mathbf{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbf{P}(X_i | X_1, \ldots, X_{i-1})$$

1.4 Inference using full joint distributions

1.4.1 Inference by enumeration

The Joint Distribution Table

Let us apply the concept of Joint Probability Distribution to a specific example involving three variables: *Toothache*, *Catch* (a dentist's probe catching on a tooth), and *Cavity*.

	toothache		**¬toothache**	
	catch	**¬catch**	**catch**	**¬catch**
cavity	0.108	0.012	0.072	0.008
¬cavity	0.016	0.064	0.144	0.576

Summing Atomic Events

We can calculate the probability of any proposition ϕ by summing the probabilities of the atomic events (cells in the table) where ϕ is true.

- Example 1: What is $P(toothache)$? We sum all entries in the "toothache" columns:

$$0.108 + 0.012 + 0.016 + 0.064 = 0.2$$

- Example 2: What is $P(cavity \vee toothache)$? We sum all cells where either is true:

$$0.108 + 0.012 + 0.072 + 0.008 + 0.016 + 0.064 = 0.28$$

Computing conditional probabilities

We can also compute conditional probabilities using the formula. For instance, to find $P(\neg cavity|toothache)$:

$$P(\neg cavity|toothache) = \frac{P(\neg cavity \wedge toothache)}{P(toothache)}$$

1. Numerator: Sum of cells where both are true ($\neg cavity$ AND $toothache$) $= 0.016 + 0.064 = 0.08$. 2. Denominator: $P(toothache) = 0.2$ (calculated above). 3. Result: $0.08/0.2 = 0.4$.

1.4.2 Normalization

The constant α

Instead of computing the denominator separately, we can view it as a **normalization constant**, denoted by α. To find the distribution of $Cavity$ given $toothache$, we calculate the joint probabilities for both cases ($Cavity = true$ and $Cavity = false$) while keeping $toothache$ fixed, and then normalize:

$$\begin{aligned}\mathbf{P}(Cavity|toothache) &= \alpha\mathbf{P}(Cavity, toothache) \\ &= \alpha[\mathbf{P}(Cavity, toothache, catch) + \mathbf{P}(Cavity, toothache, \neg catch)] \\ &= \alpha[\langle 0.108, 0.016\rangle + \langle 0.012, 0.064\rangle] \\ &= \alpha\langle 0.12, 0.08\rangle\end{aligned}$$

The sum of the vector is $0.12 + 0.08 = 0.2$. Thus, $\alpha = 1/0.2 = 5$. Final result: $5 \times \langle 0.12, 0.08\rangle = \langle 0.6, 0.4\rangle$.

General Idea

This process generalizes to any inference problem: to answer a query, we compute the distribution on the **query variable** by fixing the **evidence variables** and summing over the **hidden variables**.

1.4.3 Operations on CPDs

Marginalization and Conditioning

Two key operations are defined on Conditional Probability Distributions (CPDs):

- **Marginalization** (or Summing Out): This eliminates a variable from the distribution. E.g., summing $P(Weather, Cavity)$ over all cavity values gives $P(Weather)$.
- **Conditioning**: This fixes a variable to a specific value (e.g., $Weather = sunny$). This reduces the dimensionality of the table and requires renormalization.

1.4.4 Probability queries

The Query

Formally, a probability query is denoted as $\mathbf{P}(Y|e)$, where Y is the set of query variables and e is the evidence. The answer is obtained by summing out all hidden variables H:

$$\mathbf{P}(Y|E = e) = \alpha \sum_h \mathbf{P}(Y, E = e, H = h)$$

Obvious problems

While this method (inference by enumeration) is theoretically sound, it suffers from major scalability issues: 1. **Time complexity**: In the worst case, it is $O(d^n)$, where d is the number of values per variable and n is the number of variables. 2. **Space complexity**: Storing the full joint distribution also requires $O(d^n)$ space. 3. **Data acquisition**: It is practically impossible to estimate accurate probabilities for $O(d^n)$ specific entries.

1.5 Recap

From Logic to Probability

In this module, we explored why classical logic is often insufficient for real-world AI agents. While logic handles deterministic puzzles well, it fails in scenarios like the Airport problem due to "Laziness" (too many exceptions to list) and "Ignorance" (missing information). This necessitates the use of **Probability Theory,** which quantifies degrees of belief (uncertainty) rather than degrees of truth.

The Power and Limits of the JPD

We introduced the **Joint Probability Distribution (JPD),** a table that assigns a probability to every possible state of the world (atomic event). We demonstrated that if an agent possesses the JPD, it can answer *any* probabilistic query (such as marginals or conditionals) by simply summing appropriate entries (inference by enumeration) and normalizing the result.

The Bottleneck

However, the module concludes with a critical realization: the JPD is **computationally intractable.** Its size grows exponentially with the number of variables ($O(d^n)$). For a problem with many variables, we cannot store the JPD, nor can we fill it with data. This fundamental limitation sets the stage for the next module, which will introduce **Bayesian Networks** as a solution to efficiently represent and reason with uncertainty by exploiting independence relationships.

2 Bayesian network representation

2.1 Independence

2.1.1 Absolute Independence

Definition and Decomposition

Two random variables A and B are defined as **independent**, denoted as $P \models (A \perp B)$, if and only if the knowledge of one does not change the probability of the other. Mathematically:

$$P(A|B) = P(A) \quad \text{or} \quad P(B|A) = P(B) \quad \text{or} \quad P(A, B) = P(A)P(B)$$

Independence allows us to decompose the full joint distribution into smaller components. For example, consider the domain: $Toothache$, $Catch$, $Cavity$, and $Weather$. The full joint distribution $P(Toothache, Catch, Cavity, Weather)$ would normally require $2^4 = 32$ entries. However, if we assume $Weather$ is independent of the dental variables, we can write:

$$P(Toothache, Catch, Cavity, Weather) = P(Toothache, Catch, Cavity)P(Weather)$$

This reduces the number of required entries from 32 to 12. In the extreme case of n independent biased coins, the complexity reduces from 2^n to just n. However, absolute (marginal) independence is powerful but rare in complex domains like dentistry.

2.1.2 Conditional Independence

The Concept

A more robust and common form of knowledge is **conditional independence**. Consider the variables $Toothache$, $Cavity$, and $Catch$ (the dentist's probe catching). The full joint distribution has $2^3 - 1 = 7$ independent entries. However, we can observe that: 1. If I have a cavity, the probability that the probe catches ($Catch$) does not depend on whether I have a toothache.

$$P(catch|toothache, cavity) = P(catch|cavity)$$

2. The same holds if I do not have a cavity.

$$P(catch|toothache, \neg cavity) = P(catch|\neg cavity)$$

Thus, we say that $Catch$ is **conditionally independent** of $Toothache$ given $Cavity$. Notation: $P \models (Catch \perp Toothache|Cavity)$.

Reduction of Complexity

Using the chain rule and the conditional independence assertion, we can write the full joint distribution as:

$$P(Toothache, Catch, Cavity) = P(Toothache|Cavity)P(Catch|Cavity)P(Cavity)$$

This representation requires defining only $2 + 2 + 1 = 5$ independent numbers, compared to the original 7. In most cases, the use of conditional independence reduces the size of the representation from **exponential** in n to **linear** in n.

2.2 Bayes' Rule

2.2.1 Derivation and Usage

The Formula

From the product rule $P(a \wedge b) = P(a|b)P(b) = P(b|a)P(a)$, we derive Bayes' rule:

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$

In distribution form, useful for inference:

$$\mathbf{P}(Y|X) = \frac{\mathbf{P}(X|Y)\mathbf{P}(Y)}{\mathbf{P}(X)} = \alpha\mathbf{P}(X|Y)\mathbf{P}(Y)$$

This rule is crucial for assessing **diagnostic probability** ($P(Cause|Effect)$) starting from **causal probability** ($P(Effect|Cause)$), which is often easier to estimate.

Example: Meningitis

Consider a medical diagnosis scenario. - M: Meningitis. $P(m) = 1/50,000$. - S: Stiff neck. $P(s) = 0.01$ (1% of people have a stiff neck). - Causal knowledge: Meningitis causes a stiff neck 70% of the time, so $P(s|m) = 0.7$. We want to find the probability that a patient with a stiff neck has meningitis ($P(m|s)$):

$$P(m|s) = \frac{P(s|m)P(m)}{P(s)} = \frac{0.7 \times (1/50,000)}{0.01} = 0.0014$$

Note that despite the stiff neck, the posterior probability of meningitis remains very small.

2.2.2 Naive Bayes

Model Structure

Bayes' rule combined with conditional independence leads to the **Naive Bayes model**. If we have a single Cause and multiple Effects ($Effect_1, \ldots, Effect_n$), and we assume the effects are conditionally independent given the cause, the joint distribution is:

$$P(Cause, Effect_1, \ldots, Effect_n) = P(Cause)\prod_i P(Effect_i|Cause)$$

This is called "naive" because it assumes strong independence, but it is very efficient: the total number of parameters is linear in n.

2.3 Bayesian network representation

2.3.1 Syntax and Topology

Definition

A Bayesian network is a simple, graphical notation for conditional independence assertions and compact specification of full joint distributions. Syntax:

- A set of nodes, one per variable.

- A directed, acyclic graph (DAG), where a link $X \to Y$ loosely means "X directly influences Y".
- A conditional distribution for each node given its parents: $\mathbf{P}(X_i|Parents(X_i))$.

In the simplest case, these distributions are represented as Conditional Probability Tables (CPTs).

Example Topology

Consider the variables $Weather$, $Cavity$, $Toothache$, and $Catch$. - $Weather$ is independent of the others (isolated node). - $Cavity$ directly influences both $Toothache$ and $Catch$. - Therefore, $Toothache$ and $Catch$ are conditionally independent given $Cavity$.

2.3.2 The Burglary Example

Scenario

I am at work. Neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. The alarm is sometimes set off by minor earthquakes. Is there a burglar? Variables: $Burglar(B)$, $Earthquake(E)$, $Alarm(A)$, $JohnCalls(J)$, $MaryCalls(M)$. Topology: - $B \to A$ and $E \to A$ (Burglar and Earthquake cause Alarm). - $A \to J$ and $A \to M$ (Alarm causes neighbors to call).

CPTs (Specific Numbers)

The network is defined by the following probabilities:

- Priors: $P(B) = 0.001$, $P(E) = 0.002$.
- Alarm CPT ($P(A|B, E)$):
  - $B = T, E = T \Rightarrow 0.95$
  - $B = T, E = F \Rightarrow 0.94$
  - $B = F, E = T \Rightarrow 0.29$
  - $B = F, E = F \Rightarrow 0.001$
- JohnCalls CPT ($P(J|A)$): If $A = T \Rightarrow 0.90$; If $A = F \Rightarrow 0.05$.
- MaryCalls CPT ($P(M|A)$): If $A = T \Rightarrow 0.70$; If $A = F \Rightarrow 0.01$.

2.3.3 Reasoning Patterns (The Student Network)

The Network

Consider a network describing a student's performance: - Difficulty (D) and Intelligence (I) influence Grade (G). - Intelligence (I) influences SAT (S) score. - Grade (G) influences the recommendation Letter (L).

Causal Reasoning (Prediction)

Question: "Will George get a strong reference letter?" We start with priors. Suppose $P(Letter = strong) = 0.50$ (derived from priors). If we observe that the course is easy ($Difficulty = low$), the probability of a strong letter increases. Example update: $P(L = strong|D = easy) = 0.51$ (vs 0.49 for weak). If we observe $Intelligence = normal$, the probability might drop.

Evidential Reasoning (Explanation)

Question: "Is George a good potential recruit?" (Inferring cause from effect). Start: $P(Intelligence = high) = 0.30$. Evidence: George received a grade of C ($Grade = low$). Update: The probability of High Intelligence drops significantly (e.g., to 0.08).

Intercausal Reasoning (Explaining Away)

Question: "Why did George score low/high?" Suppose George got a Grade C ($G = low$). As seen, $P(Intelligence = high)$ drops to 0.08. Now, suppose we observe that the course was extremely hard ($Difficulty = high$). Update: The probability of High Intelligence recovers (e.g., rises to 0.14 or similar). Reasoning: The difficulty of the exam "explains away" the bad grade, so the bad grade provides less evidence against his intelligence.

2.4 Compactness and Semantics

2.4.1 Compactness

Parameter Complexity

A CPT for a Boolean variable X_i with k Boolean parents has 2^k rows. Each row requires one number p (since the false case is $1 - p$). If each variable has no more than k parents, the complete network requires $O(n \cdot 2^k)$ numbers. Comparison with Full Joint Distribution ($O(2^n)$): - Burglary Net: $1 + 1 + 4 + 2 + 2 = 10$ numbers (vs $2^5 - 1 = 31$). - Student Net: $1 + 1 + 8 + 2 + 3 = 15$ numbers (vs $2^4 \times 3 - 1 = 47$). The growth is linear in n (assuming fixed k), which allows scaling to large domains.

2.4.2 Global Semantics

Definition

Global semantics defines the full joint distribution as the product of the local conditional distributions:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|parents(X_i))$$

Example calculation from Burglary Net ($j \wedge m \wedge a \wedge \neg b \wedge \neg e$):

$$P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e)$$

Using the table values:

$$0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 \approx 0.00063$$

2.4.3 Local Semantics and Markov Blanket

Local Semantics

Each node is conditionally independent of its nondescendants given its parents. This theorem is equivalent to the global semantics product rule.

Markov Blanket

Each node is conditionally independent of **all other nodes** in the network given its Markov blanket, which consists of: 1. Its parents. 2. Its children. 3. Its children's parents.

2.5 Flow of probabilistic influence

2.5.1 Active Trails

Types of connections

When can a variable X influence Y via Z? 1. $X \to Z \to Y$ (Causal trail). 2. $X \leftarrow Z \leftarrow Y$ (Evidential trail). 3. $X \leftarrow Z \to Y$ (Common cause). 4. $X \to Z \leftarrow Y$ (Common effect / V-structure).

Definition of Active Trail

Let $\mathbf{Z}$ be a subset of observed variables. A trail $X_1 \rightleftharpoons \cdots \rightleftharpoons X_n$ is active given $\mathbf{Z}$ if: - For every V-structure $X_{i-1} \to X_i \leftarrow X_{i+1}$ along the trail, X_i or one of its descendants is in $\mathbf{Z}$. - No other node along the trail is in $\mathbf{Z}$.

2.5.2 d-separation

Definition

Two sets of nodes $\mathbf{X}$ and $\mathbf{Y}$ are d-separated given $\mathbf{Z}$ if there is no active trail between any $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$ given $\mathbf{Z}$. If they are d-separated, they are conditionally independent given $\mathbf{Z}$.

Examples

Consider the complex graph with nodes A, B, C, D, E, F, G, H, I, J (where A and B feed into C and D, which feed into E and F, etc.). - Is $P \models (C \perp D)$? Yes. The path goes through A and B (common causes, unobserved) or E/F (common effects, unobserved). - Is $P \models (C \perp D|A)$? No. Observing A opens a path if A is a common cause, but here observing a parent generally blocks the flow. However, analyzing the specific V-structures is key. - Is $P \models (C \perp D|A, B)$? Yes, if A and B block the common cause trails. - Is $P \models (C \perp D|A, B, J)$? No. J is a descendant of the V-structures below. Observing a descendant of a collider (like those leading to G and H) activates the V-structure, potentially opening a path.

Recap of the Module

The Solution to Exponential Complexity

We started with the problem of the Full Joint Distribution being too large ($O(2^n)$). We introduced Independence and Conditional Independence as the mathematical tools to break down this complexity.

Bayesian Networks

A Bayesian Network uses a DAG to represent these independencies visually. The Joint Distribution is reconstructed by multiplying local CPTs (Global Semantics). The complexity drops to $O(n \cdot 2^k)$, making reasoning feasible for large systems.

Reasoning

We saw that BNs support various reasoning patterns (Causal, Evidential, Intercausal). Crucially, the concept of d-separation allows us to determine purely from the graph structure whether two variables are independent given some evidence, without doing any numerical calculation.

2.6 Recap

The Solution to Exponential Complexity

We started with the problem of the Full Joint Distribution being too large ($O(2^n)$). We introduced **Independence** and **Conditional Independence** as the mathematical tools to break down this complexity.

Bayesian Networks

A Bayesian Network uses a DAG to represent these independencies visually. The Joint Distribution is reconstructed by multiplying local CPTs (Global Semantics). The complexity drops to $O(n \cdot 2^k)$, making reasoning feasible for large systems.

Reasoning

We saw that BNs support various reasoning patterns (Causal, Evidential, Intercausal). Crucially, the concept of **d-separation** allows us to determine purely from the graph structure whether two variables are independent given some evidence, without doing any numerical calculation.

3 Building Bayesian networks

3.1 Constructing Bayesian networks

3.1.1 The Construction Algorithm

Methodology

To ensure that a Bayesian Network correctly represents the global semantics (the full joint distribution), we need a systematic method to build it. The core requirement is that locally testable assertions of conditional independence must guarantee the global property. The algorithm proceeds as follows: 1. **Choose an ordering** of variables $X_1, \ldots, X_n$. 2. For $i = 1$ to n:

- Add node X_i to the network.
- Select a minimal set of parents from $X_1, \ldots, X_{i-1}$ such that the conditional independence property holds:

$$\mathbf{P}(X_i|Parents(X_i)) = \mathbf{P}(X_i|X_1, \ldots, X_{i-1})$$

This construction guarantees that the network topology satisfies the chain rule:

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

3.1.2 The Importance of Ordering

The Burglary Example Revisited

The choice of variable ordering is critical for the compactness of the network. Let us analyze the Burglary network with a non-causal ordering: M, J, A, B, E.

- **Step 1**: Add MaryCalls (M).
- **Step 2**: Add JohnCalls (J). Is $P(J|M) = P(J)$? **No**. If Mary calls, it is likely the alarm is ringing, so John is also likely to call. We must draw a link $M \to J$.
- **Step 3**: Add Alarm (A). Is $P(A|J, M) = P(A|J)$? **No**. We need links from both M and J to A.
- **Step 4**: Add Burglary (B). We need links from A to B.
- **Step 5**: Add Earthquake (E). We need links from A and B to E.

Compactness Comparison

Deciding conditional independence is difficult for humans in non-causal directions. Furthermore, the resulting network is much denser and less efficient.

- **Causal Ordering** (B, E, A, J, M): Requires $1 + 1 + 4 + 2 + 2 = \mathbf{10}$ numbers.
- **Non-Causal Ordering** (M, J, A, B, E): Requires $1 + 2 + 4 + 2 + 4 = \mathbf{13}$ numbers.

The standard causal model (Burglary causes Alarm) aligns with how humans perceive the world, leading to sparse graphs and fewer parameters.

Example: Car Diagnosis

A practical application is car diagnosis (initial evidence: "car won't start"). The network structure typically distinguishes between: 1. Testable variables (green nodes, e.g., "battery meter"). 2. "Broken" variables (orange nodes, e.g., "alternator broken"). 3. Hidden variables (gray nodes, e.g., "no charging"). Hidden variables are crucial because they ensure a sparse structure and reduce the number of parameters required.

3.1.3 Structure Learning

Approaches

If manual design is not possible, we can learn the network structure from data using two main approaches: 1. Constraint-based: Perform independence tests to identify edge constraints, then search for a graph that satisfies them. 2. Score-based: Define a criterion (score) to evaluate how well a network fits the data, then search to maximize this score.

3.2 Causal networks

3.2.1 Causality vs Probability

Causal Asymmetry

In principle, any ordering permits a consistent construction of a Bayesian network (as seen with the Burglary example). However, **Causal Networks** are a restricted class where arrows specifically represent causal influence. Consider the relationship between Fire and Smoke:

- Model (a): $Fire \rightarrow Smoke$.
- Model (b): $Smoke \rightarrow Fire$.

Probabilistically, equally good distributions can be defined for both. However, they are not equivalent when we consider **interventions**. The question is: "Which responds to which?". Rule: Draw $X \rightarrow Y$ if nature "assigns" a value to Y based on X.

Structural Equations

We can represent variables using structural equations of the form $x_i = f_i(Parents(X_i), U_i)$, where U_i represents unmodeled variables (noise). Example: The Lawn Example.

$$Cloudy \rightarrow \{Sprinkler, Rain\} \rightarrow WetGrass \rightarrow GreenerGrass$$

These equations describe mechanisms in nature that are invariant to local changes.

3.2.2 The do-operator

Interventions

Stability is key for representing interventions. The semantics of standard Bayes nets gives us $P(c, r, s, w, g)$. However, if we intervene, e.g., turn the sprinkler on ($do(Sprinkler = True)$), the network structure changes.

- Observation ($P(W|S = true)$): We observe the sprinkler is on. This provides evidence about the Cloudiness (it's likely dry), which affects the probability of Rain.
- Intervention ($P(W|do(S = true))$): We force the sprinkler on. This cuts the link $Cloudy \rightarrow Sprinkler$. The value of Sprinkler is now fixed by us, not by the clouds. Therefore, our action tells us nothing about whether it rained.

The do-operator allows us to predict the consequences of actions, which differs from simply observing a state.

3.3 Representing conditional distributions

3.3.1 Compact conditional distributions

The Problem of CPT size

The Conditional Probability Table (CPT) grows exponentially with the number of parents (2^k). If parents are continuous, the CPT becomes infinite. Solution: Use canonical distributions (standard patterns) or deterministic nodes.

- Deterministic: $X = f(Parents(X))$. Example: $NorthAmerican \iff Canadian \vee US \vee Mexican$.
- Continuous: Numerical relationships, e.g., $\frac{\partial Level}{\partial t} = inflow - outflow$.

Noisy-OR

The Noisy-OR model is used for multiple non-interacting causes. It assumes that each cause has an independent probability q_i of *failing* to trigger the effect.

$$P(Effect|U_1 \dots U_j, \neg U_{j+1} \dots) = 1 - \prod_{i=1}^{j} q_i$$

where $U_1 \dots U_j$ are the active causes. Example: Causes of Fever ($Cold, Flu, Malaria$). If $P(\neg Fever|Malaria) = 0.1$ and $P(\neg Fever|Flu) = 0.2$:

$$P(Fever|Malaria, Flu) = 1 - (0.1 \times 0.2) = 1 - 0.02 = 0.98$$

This reduces the number of parameters from exponential to linear in the number of parents.

Real-world Example: CPCS Network

The effectiveness of these methods is shown in the CPCS network for internal medicine (Pradhan et al., 1994). This massive network contains 448 nodes and 908 arcs. Instead of filling a giant table, it uses predisposing factors and leak probabilities, requiring the assessment of only about 560 probabilities to fully specify the network.

3.3.2 Hybrid Networks

Discrete and Continuous Variables

Real-world networks often mix variable types. Example: $Harvest$ (Continuous) and $Subsidy$ (Discrete) influence $Cost$ (Continuous), which influences $Buys$ (Discrete).

Continuous Child (Linear Gaussian)

For a continuous child with continuous and discrete parents (e.g., Cost), we often use the Linear Gaussian (LG) model.

$$P(c|h, subsidy) = \mathcal{N}(a_t h + b_t, \sigma_t^2)$$

The mean of the child varies linearly with the continuous parent (h), but the parameters (a_t, b_t, σ_t) depend on the value of the discrete parent ($subsidy$). This results in a Conditional Gaussian network: a multivariate Gaussian for each combination of discrete values.

Discrete Child (Soft Threshold)

For a discrete child with continuous parents (e.g., Buys given Cost), the probability represents a "soft" threshold. We use functions like:

- Probit: Based on the integral of the Gaussian.
- Sigmoid (Logit): $P(Buys = true|Cost = c) = \frac{1}{1+\exp(-2\frac{-c+\mu}{\sigma})}$.

3.3.3 Density Estimation and Learning

Density Estimation

Parameters (conditional distributions) are not always elicited from experts; they can be learned from data. This process is called density estimation. Conceptually: - Data are viewed as evidence. - Hypotheses are probabilistic theories about the domain. - Bayesian learning calculates the probability of each hypothesis given the data.

Learning Methods

Predictions are made using all hypotheses, weighted by their probabilities. This is optimal but computationally expensive. Approximations include: 1. MAP hypothesis: Keeps only the most probable theory. 2. Maximum-likelihood hypothesis: Assumes a uniform prior. 3. EM algorithm: Used for learning when there are hidden variables.

3.3.4 Other Probabilistic Models

Dynamic Bayesian Networks (DBNs)

Used for reasoning about time. We define template variables $X_i^{(t)}$ instantiated at each time step. The goal is to represent a joint distribution over trajectories (e.g., Weather and Location evolving over time steps $0, 1, 2 \dots$).

Markov Networks (Undirected)

Bayesian Networks are a type of Probabilistic Graphical Model (PGM). Another class is **Markov Networks.**

- Undirected graphs: Edges have no arrows.

- Factors: Instead of CPTs (probabilities), they use compatibility factors (potentials, ϕ) which denote "affinity" between values but do not necessarily sum to 1.
- They naturally capture conditional independence but require a normalization constant (partition function) to become probabilities.

3.4 Recap

Building the Network

We learned that the order of node insertion matters. Constructing the network using a causal ordering (Causes $\rightarrow$ Effects) minimizes the number of edges and parameters, making the network sparse and efficient. Non-causal orderings lead to dense, complex networks.

Causality vs. Correlation

We introduced the distinction between simple probabilistic dependence and causality. This is formalized by the do-operator. Observing a variable ($X = x$) allows inference flow in all directions; Intervening on a variable ($do(X = x)$) cuts the links from its parents, preventing inference from flowing back to causes.

Handling Complexity

To represent complex real-world distributions (like the CPCS medical network with 448 nodes), we need efficient representations for the Conditional Probabilities.

- Discrete: Noisy-OR reduces parameters from exponential to linear.
- Continuous: Linear Gaussian models allow mixing discrete and continuous variables.
- Thresholds: Sigmoid/Probit functions handle continuous parents for discrete children.
- Learning: When experts are unavailable, parameters can be learned from data using Density Estimation (MAP, EM).

4 Exact inference

4.1 Inference tasks

4.1.1 Types of Queries

Simple and Conjunctive Queries

The primary goal of a Bayesian network is to answer queries about the domain.

- Simple queries: Compute the posterior marginal probability of a query variable X_i given some evidence $\mathbf{E} = \mathbf{e}$.
$$P(X_i|\mathbf{E} = \mathbf{e})$$
Example: $P(NoGas|Gauge = empty, Lights = on, Starts = false)$.

- Conjunctive queries: Compute the posterior probability of a set of variables.
$$\mathbf{P}(X_i, X_j|\mathbf{E} = \mathbf{e}) = \mathbf{P}(X_i|\mathbf{E} = \mathbf{e})\mathbf{P}(X_j|X_i, \mathbf{E} = \mathbf{e})$$

Advanced Tasks

Beyond simple probabilities, inference supports decision making:

- Optimal decisions: In decision networks (which include utility information), we need to calculate $P(outcome|action, evidence)$ to choose the action that maximizes expected utility.

- Value of information: Determining which piece of evidence is most valuable to acquire next.

- Sensitivity analysis: Identifying which probability parameters most critically affect the query result.

- Explanation: Answering "why" questions (e.g., "Why do I need a new starter motor?").

4.2 Inference by enumeration

4.2.1 The Concept

Definition

Inference by enumeration is a slightly intelligent way to sum out variables from the joint distribution without actually constructing the full explicit representation (which would be huge). Consider a query on the Burglary network: $\mathbf{P}(B|j, m)$. Using the definition of conditional probability and normalization:

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

We can rewrite the full joint entries as a product of CPT entries based on the network structure:

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

Evaluation Tree

The computation can be visualized as a depth-first recursive enumeration. The algorithm loops over the possible values of the hidden variables (e and a). - Space complexity: $O(n)$, because we only need to store the current path in the tree. We do *not* construct the full joint distribution. - Time complexity: $O(d^n)$, where d is the number of values per variable. Inefficiency: The main problem is repeated computation. For example, the product $P(j|a)P(m|a)$ is computed multiple times for different branches of the tree.

4.2.2 The Algorithm (Pseudo-code)

Enumeration-Ask

This function initializes the process. It iterates over every possible value of the query variable X, extends the evidence set, and calls the recursive enumeration. Finally, it normalizes the result.

```
function ENUMERATION-ASK(X, e, bn) returns a distribution over X
    inputs: X, the query variable
            e, observed values for variables E
            bn, a Bayesian network with variables {X} U E U Y
    Q(X) <- a distribution over X, initially empty
    for each value xi of X do
        extend e with X = xi
        Q(xi) <- ENUMERATE-ALL(Vars[bn], e)
    return NORMALIZE(Q(X))
```

Enumerate-All

This is the recursive core. It processes variables one by one. 1. If the list of variables is empty, it returns 1.0 (base case). 2. If the current variable Y is an evidence variable (has a value y in e), it multiplies the probability $P(y|Parents(Y))$ by the recursive call on the rest. 3. If Y is a hidden variable, it sums the results of the recursive calls for all possible values of Y.

```
function ENUMERATE-ALL(vars, e) returns a real number
    if EMPTY?(vars) then return 1.0
    Y <- FIRST(vars)
    if Y has value y in e then
        return P(y|Parents(Y)) * ENUMERATE-ALL(REST(vars), e)
    else return SUM_{y} P(y|Parents(Y)) *
                        ENUMERATE-ALL(REST(vars), e U {Y=y})
```

4.3 Inference by variable elimination

4.3.1 The Concept

Key Idea

Variable Elimination improves upon enumeration by carrying out summations right-to-left and storing intermediate results (called factors) to avoid recomputation. Let's rewrite

the Burglary summation:

$$\mathbf{P}(B|j,m) = \alpha \underbrace{\mathbf{P}(B)}_{f_1} \sum_e \underbrace{P(e)}_{f_2} \sum_a \underbrace{\mathbf{P}(a|B,e)}_{f_3} \underbrace{P(j|a)}_{f_4} \underbrace{P(m|a)}_{f_5}$$

We process the sum over a first. The terms involving a are f_3, f_4, f_5. We multiply them and sum out a, creating a new factor $f_{\bar{A}}(b,e)$ that depends only on B and E. Then we process the sum over e using f_2 and the new factor $f_{\bar{A}}$, creating a factor $f_{\bar{E}}(b)$. Finally, we multiply by f_1 and normalize.

4.3.2 Basic Operations

Pointwise Product

The pointwise product of two factors f_1 and f_2 yields a new factor whose variables are the union of the variables in f_1 and f_2. Example: $f_1(A,B) \times f_2(B,C) = f(A,B,C)$. For each combination of (a,b,c), the value is $f_1(a,b) \times f_2(b,c)$.

Summing Out

Summing out a variable X from a product of factors involves: 1. Collecting all factors that depend on X. 2. Multiplying them together. 3. Summing the results over all values of X. The result is a factor that no longer depends on X.

4.3.3 The Algorithm (Pseudo-code)

Elimination-Ask

This algorithm treats the network as a list of factors. It iterates through the variables based on a specific ordering. 1. Make-Factor: Converts CPTs into factors, fixing evidence variables where appropriate. 2. Sum-Out: If a variable is hidden, it is summed out from the product of factors that contain it. 3. Normalize: The final product is normalized to get a probability distribution.

```
function ELIMINATION-ASK(X, e, bn) returns a distribution over X
    factors <- []
    for each var in ORDER(VARS[bn]) do
        factors <- [MAKE-FACTOR(var, e) | factors]
        if var is a hidden variable then
            factors <- SUM-OUT(var, factors)
    return NORMALIZE(POINTWISE-PRODUCT(factors))
```

Efficiency and Ordering

Any ordering yields a valid result, but the goal is to minimize the size of the largest intermediate factor constructed. Finding the optimal ordering is NP-hard, but heuristics work well in practice.

4.3.4 Complexity Results

Irrelevant Variables

A variable Y is irrelevant to the query $\mathbf{P}(X|E)$ if summing over it yields a factor of 1. Theorem: Y is irrelevant unless $Y \in Ancestors(\{X\} \cup E)$. More generally, if it is d-separated from X by E.

Topology Matters

- Singly connected networks (Polytrees): Graphs with no undirected cycles. There is at most one path between any two nodes. Time and space complexity: $O(d^k n)$, which is linear in n.
- Multiply connected networks: Graphs with loops. Inference is NP-hard. This means essentially that in the worst case, complexity is exponential $O(d^n)$.

Clustering Algorithms

Variable elimination solves one query at a time. If we need to compute posteriors for *all* variables, running it n times takes $O(n^2)$. Clustering (Join Tree) algorithms can do this in $O(n)$ by joining nodes to form "meganodes" until the network becomes a polytree. However, the worst-case complexity remains exponential due to the size of the CPTs within meganodes.

4.4 Exercise: Online Shop

4.4.1 Problem Description

Scenario

You are designing an e-commerce system to sell products. You can display ads for Books, Toys, or Holidays. You want to guess which ad a customer will click. You don't have profile data, but you infer features from behavior. Profile Features (Hidden/Inferred): - $Young(y)$: Is the customer young? - $Married(m)$: Is the customer married? - $Wealthy(w)$: Is the customer wealthy? - $Children(c)$: Does the customer have children? Behaviors (Observables): - $Book(b)$: Clicks on book ad. - $Toy(t)$: Clicks on toy ad. - $Holiday(h)$: Clicks on holiday ad.

Network Topology

The proposed network structure is:

- Roots: $Young$ and $Married$.
- Intermediate: - $Young \rightarrow Wealthy$ (Age affects wealth). - $Young \rightarrow Children$ and $Married \rightarrow Children$ (Age and marriage status affect having children).
- Leaves (Observables): - $Wealthy \rightarrow Holiday$ (Wealth affects holiday interest). - $Wealthy \rightarrow Book$ and $Children \rightarrow Book$ (Wealth and kids affect book interest). - $Children \rightarrow Toy$ (Kids affect toy interest).

4.4.2 Questions and Reasoning

Reasoning Patterns

Q1: You display ads, record clicks, and infer profile features. Which pattern is this? Answer: Diagnostic (or Evidential) reasoning. You observe the Effects (clicks) to infer the Causes (profile). Query: $\mathbf{P}(Young|Book = true, Toy = false)$.

Parameters

Q2: How many independent values are needed? We sum the parameters for each CPT: - Roots (Y, M): 1 each. - W (parent Y): 2. - C (parents Y, M): 4. - H (parent W): 2. - B (parents W, C): 4. - T (parent C): 2. Total $= 1 + 1 + 2 + 4 + 2 + 4 + 2 = 16$.

Inference Calculation

Q4: Calculate $P(c|h)$ using variable elimination. Network part: $Y \to W \to H$. Also M and $Y \to C$. The query involves C and H.

$$P(c|h) = \alpha \sum_{y} \sum_{m} \sum_{w} P(y)P(m)P(w|y)P(c|y, m)P(h|w)$$

We compute factors by summing out variables one by one.

Extensions and d-separation

Q6: You introduce a new node Male and Skateboard ($Male \to Skateboard$). Also $Male \to Wealthy$. Scenario: You observe the customer does *not* click on a skateboard ($S = false$). Does this help predict if they are $Married$? - Path: $S \leftarrow Male \to W \leftarrow Y \to C \leftarrow Married$. - Analysis: The path is blocked at the V-structures (W and C) because we have not observed W, C or their descendants. So, observing S does not influence M. Independence holds.

Q7: New evidence: Customer is a female parent. - Evidence: $Male = false$, $Children = true$. - Now C is observed. This activates the V-structure at C. - Is there an active path from S to M? - Path segment: $S \leftarrow Male \to W \leftarrow Y \to C \leftarrow M$. - C is observed (open). W is blocked (V-structure not observed). - However, $Male$ is observed ($Male = false$). Observing a node on the path blocks the flow. - So S is d-separated from M by $Male$. The observation of S adds no new info about M because we already know the gender perfectly.

4.5 Recap

Algorithms

We explored two main exact inference algorithms: 1. Enumeration: Conceptual simple, sums over the full joint via recursion. Good for space $O(n)$, bad for time $O(d^n)$. 2. Variable Elimination: More efficient dynamic programming approach. Saves intermediate factors. Efficiency depends on node ordering.

Complexity Limits

Inference is efficient ($O(n)$) only for Polytrees (singly connected networks). For general multiply connected networks (with loops), exact inference is NP-hard. This suggests that for very complex networks, we might need Approximate Inference (next module).

5 Approximate inference

5.1 About the exam

5.1.1 Format and Rules

Options

Students can choose between two exam formats: 1. Written Exam: Consists of one exercise (8 points, style of Russel Norvig) and one or more open theory questions (3 points). Duration: 1 hour. Registration via AlmaEsami. 2. Project: Can be a Case Study (implementing a network using libraries like 'pgmpy') or an Investigation (theoretical study/experiment). Requires code, a 2-page report, and an oral discussion (presentation + QA).

5.2 Approximate inference

5.2.1 Motivation

Complexity

As seen in the previous module, exact inference is efficient for polytrees ($O(n)$) but becomes NP-hard for multiply connected networks (general graphs with loops). Example: The Car Insurance Network. It has hidden variables (e.g., "AntiTheft", "DrivingSkill") and many connections. Exact inference might take 10^8 operations, while approximate methods or optimized elimination might reduce this to 10^5. However, when the network is very large or dense, we must rely on approximation.

The Stochastic Simulation Idea

The basic idea is to draw N samples from a sampling distribution and compute an approximate posterior probability $\hat{P}$ based on the frequencies in the sample. We aim to show that as $N \to \infty$, the estimate converges to the true probability P.

Theorem (Dagum and Luby, 1993)

It is important to note that even approximate inference is hard. Both absolute approximation ($|P - \hat{P}| \leq \epsilon$) and relative approximation are NP-hard for any $\epsilon, \delta < 0.5$. However, in practice, these methods often work well.

5.2.2 Direct Sampling Methods

Sampling from an Empty Network (Prior Sampling)

This is the simplest method. We sample each variable in topological order. 1. Sample root nodes based on their priors $P(X)$. 2. Sample children based on the values assigned to their parents: $P(X_i|Parents(X_i))$. This generates samples from the full joint distribution $P(X_1, \ldots, X_n)$.

```
function PRIOR-SAMPLE(bn) returns an event sampled from bn
    x <- an event with n elements
    for i = 1 to n do
        xi <- a random sample from P(Xi | parents(Xi))
              given the values of Parents(Xi) in x
    return x
```

Rejection Sampling

Used to compute conditional probabilities $P(X|e)$. Algorithm: 1. Generate samples using Prior Sampling. 2. If a sample is inconsistent with the evidence e (e.g., we observe $Rain = true$ but sampled $Rain = false$), we reject (discard) it. 3. Count the remaining samples to estimate the probability.

```
function REJECTION-SAMPLING(X, e, bn, N) returns estimate of P(X|
   e)
    N_counts <- vector of counts over X, initially zero
    for j = 1 to N do
        x <- PRIOR-SAMPLE(bn)
        if x is consistent with e then
            N_counts[x] <- N_counts[x] + 1
    return NORMALIZE(N_counts)
```

Example: Estimate $P(Rain|Sprinkler = true)$. Suppose we draw 100 samples. - 73 samples have $Sprinkler = false \rightarrow$ Rejected. - 27 samples have $Sprinkler = true$. Of these: - 8 have $Rain = true$. - 19 have $Rain = false$. Result: $\hat{P} = Normalize(\langle 8, 19 \rangle) = \langle 0.296, 0.704 \rangle$. Problem: If the evidence is rare, we reject almost all samples.

Likelihood Weighting

Idea: Avoid rejecting samples by fixing the values of the evidence variables. To correct the bias, we weight each sample by the likelihood it accords to the evidence.

```
function LIKELIHOOD-WEIGHTING(X, e, bn, N) returns estimate
    W <- vector of weighted counts
    for j = 1 to N do
        x, w <- WEIGHTED-SAMPLE(bn, e)
        W[x] <- W[x] + w
    return NORMALIZE(W)
```

Weight calculation: $w(\mathbf{z}, \mathbf{e}) = \prod P(e_i|parents(E_i))$. This returns consistent estimates, but performance degrades if evidence variables are "downstream" and unlikely.

5.2.3 Markov Chain Monte Carlo (MCMC)

Concept

Instead of generating independent samples from scratch, MCMC generates a sequence of samples where each one depends on the previous one (a Markov Chain). We wander through the state space, and the fraction of time spent in each state is proportional to its posterior probability.

Gibbs Sampling

Gibbs Sampling is a specific MCMC method for Bayesian Networks. It samples one variable at a time conditioned on its Markov Blanket (parents, children, children's parents).

```
function MCMC-ASK(X, e, bn, N) returns estimate of P(X|e)
    N_counts <- vector of counts
    Z <- nonevidence variables
    x <- current state, initialized randomly
    for j = 1 to N do
        for each Zi in Z do
            sample value of Zi from P(Zi | MarkovBlanket(Zi))
            update x
        N_counts[x] <- N_counts[x] + 1
    return NORMALIZE(N_counts)
```

Numerical Example

Estimate $P(Rain|Sprinkler = true, WetGrass = true)$. We have 4 possible states (since S and W are fixed): $(C, R) \in \{TT, TF, FT, FF\}$. We wander for a while. Suppose we visit 100 states. - 31 times the state has $Rain = true$. - 69 times the state has $Rain = false$. Result: $\hat{P} = Normalize(\langle 31, 69 \rangle) = \langle 0.31, 0.69 \rangle$.

Compiling Approximate Inference

To speed up sampling, we can compile the network into model-specific code. For example, to sample $Burglary$ given $Alarm$, instead of generic graph lookups, we compile:

```
r <- random()
if Alarm == true then
    if Burglary == true return (r < 0.002)
    else return (r < 0.367)
...
```

This is 2-3 orders of magnitude faster than generic MCMC.

5.3 Recap

Exact vs. Approximate

Exact inference (Variable Elimination) is precise but fragile: its complexity explodes with network connectivity. Approximate inference offers a trade-off: we trade precision for speed and feasibility.

Evolution of Methods

We moved from naive Rejection Sampling (correct but wasteful) to Likelihood Weighting (uses evidence, better but still struggles with rare events) to MCMC/Gibbs Sampling (smart local updates using the Markov Blanket).

The Reality Check

Even approximate inference is theoretically hard (NP-hard bounds). However, in practice, methods like MCMC combined with compilation allow us to reason about massive networks (like the Car Insurance example) that would be impossible to solve exactly.

6 Exercises & wrap-up

6.1 Simple case studies

6.1.1 Case Study 1: Bayes Stop

Scenario

You are waiting at a bus stop and notice a crowd. Why? Variables (Boolean): - RushHour (r): Is it rush hour? - Traffic (t): Is there heavy traffic? - Late (l): Is the bus delayed? - Ban (b): Is there a car ban (due to pollution)? - Crowd (c): Is there a crowd? Initial causal links: - RushHour causes Traffic. - Traffic causes Late. - Late causes Crowd. - Ban causes Crowd (more people take the bus).

Questions and Reasoning

1. Define CPTs: How many independent parameters? (Depends on topology, e.g., $1 + 2 + 2 + 1 + 4 = 10$). 2. Explain the crowd: Query $P(Late = true|Crowd = true)$. This is Diagnostic reasoning (Effect $\rightarrow$ Cause). 3. Intercausal Reasoning: You observe $RushHour = false$ but there is a Crowd. Does this affect belief in Ban? - Yes. If it's not rush hour, Traffic and Late are less likely. So "Late" is a less likely explanation for "Crowd". This makes the alternative explanation "Ban" more likely (Explaining Away).

Extensions (Football Match)

New variables: Football, Deviation (bus route change), Police. - Football causes Traffic. - Football causes Deviation. - Deviation causes Late. - Football causes Police. Is the resulting network a polytree? No, because there are multiple undirected paths (e.g., Football $\rightarrow$ Traffic $\rightarrow$ Late $\leftarrow$ Deviation $\leftarrow$ Football). Loops exist.

Independence and Markov Blanket

- Is $P \models (Police \perp Ban|Crowd, Deviation, RushHour)$? Check d-separation. The path from Police to Ban goes through Football $\rightarrow$ Traffic $\rightarrow$ Late $\rightarrow$ Crowd $\leftarrow$ Ban. $Crowd$ is a collider and is observed $\rightarrow$ path open. $Late$ is not observed. $Traffic$ is not observed. This requires careful d-separation analysis. - Markov Blanket of Deviation: Parents ($Football$), Children ($Late$), Children's other parents ($Traffic$).

Inference

- Query: $P(Traffic|\neg RushHour, Deviation)$. - Calculate using Variable Elimination (show first step summing out a variable). - Calculate using Sampling: Generate one sample given random numbers (e.g., 0.32, 0.92...).

6.1.2 Case Study 2: Bookworms

Scenario

You want to classify books by genre (Spy, Cooking, AI) based on words found on a random page ("agent", "oven", "mind"). Two possible network structures: - (a) Words $\rightarrow$ Book $\rightarrow$ Genre: This implies Genre depends on the specific Book. - (b) Genre $\rightarrow$ Words: This implies words appear based on the Genre directly (Naive Bayes assumption).

Analysis

- Q4: Which option serves the purpose of estimating genre better? Option (b) is the standard Naive Bayes classifier. It allows learning $P(word|genre)$ across all books. - Q5: What independence assumption is implied by (b)? Conditional independence of words given Genre. This is a strong assumption (e.g., "oven" and "bake" are correlated even within Cooking books), but simplifies computation. - Q6: Evaluate $P(spy|mind, \neg oven)$ using variable elimination.

6.1.3 Case Study 3: Predictive Justice

Scenario

Forensic medics model the risk of violent reoffending. Variables: Social, Family, Network (criminal network), Gang, Victimisation, Attitude, Aggression, Reconviction. Structure shows complex causal chains, e.g., $Family \rightarrow Network \rightarrow Gang \rightarrow Aggression$.

Interventions

Experts propose interventions: - Psychiatric therapy: Reduces Victimisation. - Employment policies: Improves Social factors. Modelling interventions: Add decision nodes or fix values (do-operator). Q4: Which intervention reduces Reconviction risk for a subject in a criminal Network? - Tracing the paths: Network affects Gang and Attitude. - Employment affects Social $\rightarrow$ Network (but Network is already "known"/fixed for this subject, so Employment might not help downstream if we already know the state of Network). - Psychiatric affects Victimisation $\rightarrow$ Aggression $\rightarrow$ Reconviction. This seems effective.

Probability Calculation

Q5: Compute $P(Victimisation|\dots)$. Requires using the specific CPTs provided (e.g., $P(v|p, g)$) and summing out hidden variables (like Gang, if not observed).

6.1.4 Case Study 4: Energy Efficiency

Scenario

A network for home energy consumption. Variables: $TimeOfDay, Weather, Occupants, SolarPanels, A$
Query: $P(Heating|SolarPanels = high, Dehumidifiers = low)$. This exercises d-separation: Heating and SolarPanels might be independent given Weather? Or connected via a V-structure at Consumption?

6.2 Concluding remarks

6.2.1 What we didn't cover

Advanced Topics

The field of Probabilistic Graphical Models is huge. We omitted:

- Representation: Plate models, Undirected graphs (Markov Random Fields).
- Inference: Belief propagation (Loopy BP), Junction Tree algorithm (exact inference for general graphs).

- Dynamic Belief Networks: Hidden Markov Models (HMM), Kalman Filters, Conditional Random Fields (CRF) for time-series data.
- Probabilistic Logic Programming (PLP): Combining logic (Prolog) with probability. Example: Monty Hall in LPAD syntax.
- Learning: Learning structure and parameters from data (beyond simple density estimation).

Example: Monty Hall in PLP

Here is how the Monty Hall problem looks in Probabilistic Logic Programming (LPAD syntax):

```
prize(1): 1/3 ; prize(2) : 1/3 : prize(3) : 1/3.
open_door(2): 0.5 : open_door(3) : 0.5 :- prize(1).
open_door(2) :- prize(3).
open_door(3) :- prize(2).
win_keep :- prize(1).
win_switch :- prize(2), open_door(3).
win_switch :- prize(3), open_door(2).
?- prob(win_keep, Prob). % returns 0.333
```

6.2.2 Software

Libraries

For practical implementation, Python libraries are standard. - pgmpy: A popular library for learning and inference. Example code structure:

```
from pgmpy.models import BayesianModel
from pgmpy.factors.discrete import TabularCPD
from pgmpy.inference import VariableElimination

# Define structure
model = BayesianModel([('C', 'H'), ('P', 'H')])

# Define CPDs
cpd_c = TabularCPD('C', 3, [[0.33], [0.33], [0.33]])
# ... define other CPDs ...

# Add to model and infer
model.add_cpds(cpd_c, ...)
infer = VariableElimination(model)
print(infer.query(['P'], evidence={'C': 0, 'H': 2}))
```

7 Course Summary and Synthesis

The Epistemological Shift: From Logic to Probability

This module marked a fundamental shift in how we design intelligent agents. We moved from the rigid world of logic,where facts are either true or false,to the nuanced world of uncertainty. We established that in real-world scenarios (like the Airport problem), strictly logical plans fail due to **laziness** (the impossibility of listing every exception) and **ignorance** (the lack of complete information). To cope with this, we adopted **Subjective Probability**. We no longer model the "truth" of the world, but rather the agent's **degree of belief** in a proposition given the available evidence.

The Fundamental Challenge: The JPD Bottleneck

The theoretical core of probabilistic reasoning is the **Joint Probability Distribution (JPD)**. This "Holy Grail" contains the probability of every possible atomic event (every combination of variables). Ideally, if an agent possesses the JPD, it can answer *any* query (via marginalization and normalization). However, we encountered a critical computational barrier: the JPD grows exponentially with the number of variables ($O(d^n)$). For any non-trivial domain, the JPD is impossible to store, learn, or compute with.

The Solution: Bayesian Networks

The breakthrough solution is the **Bayesian Network**. A BN is not merely a diagram; it is a compact, factored representation of the JPD. By exploiting **Conditional Independence**, a BN breaks the massive global distribution into small, manageable local Conditional Probability Tables (CPTs). The topology of the network (the DAG) explicitly encodes these independence assertions (Theorem: Local Semantics $\iff$ Global Semantics). We learned that a **sparse** network is an efficient network, and that respecting the causal order (Cause $\rightarrow$ Effect) is crucial for keeping the graph sparse.

Exact Inference: Precision at a Cost

Once the model is built, we need algorithms to extract answers (Posterior Probabilities). We explored **Exact Inference** methods.

- **Enumeration** is the brute-force approach: mathematically sound but computationally redundant.
- **Variable Elimination** optimizes this by using dynamic programming. By summing out variables and storing intermediate factors, it avoids recalculating the same values.

However, the "No Free Lunch" theorem applies: exact inference remains **NP-hard** for general (multiply connected) graphs. Efficiency ($O(n)$) is only guaranteed for **Polytrees**.

Approximate Inference: The Practical Trade-off

When networks become too complex for exact math, we resort to **Approximate Inference** via Stochastic Simulation. Instead of calculating the exact probability mass, we generate samples to estimate it.

- We started with simple methods like **Rejection Sampling** (wasteful) and **Likelihood Weighting** (better, but biased by downstream evidence).
- We arrived at the state-of-the-art: **Markov Chain Monte Carlo (MCMC)** and **Gibbs Sampling**. By wandering through the state space and utilizing the **Markov Blanket** (local neighborhood), these algorithms converge to the true posterior distribution over time, making inference feasible even in massive networks.

Causality and Beyond

Finally, we refined our understanding of the arrows. While probabilistic dependence is symmetric, **Causality** is asymmetric. We introduced the **do-operator** to distinguish between *observing* an event (inference flows everywhere) and *intervening* to cause an event (which cuts links to parents). This distinction is vital for planning and predictive justice. In conclusion, Bayesian Networks provide the rigorous mathematical framework to build rational agents that can perceive, reason, and decide optimally (MEU) even in an uncertain, chaotic world.