Graphical Models

Module 5: Machine Learning Course

Why Graphical Models Matter

The Real Problem

Imagine you're a fraud detection analyst at a major bank. Every second, thousands of transactions flow through the system.

Business Impact: Banks lose $28 billion annually to fraud

$4,000

Average Fraud Cost

Per fraudulent transaction

$118

False Positive Cost

Per blocked legitimate transaction

The Challenge: How do you model complex relationships between transaction amount, location, time, merchant type, customer behaviour, and device patterns? Graphical Models save the day!

Learning Like Humans Do

Solving a Mystery: Did Someone Attend the Party?

Trial 1: Evidence Found

"I see empty beer bottles"

Belief Update: 60% chance they went

Trial 2: More Evidence

"Car wasn't in driveway at 9 PM"

Belief Update: 80% chance they went

Trial 3: Definitive Proof

"Friend posts photo with them"

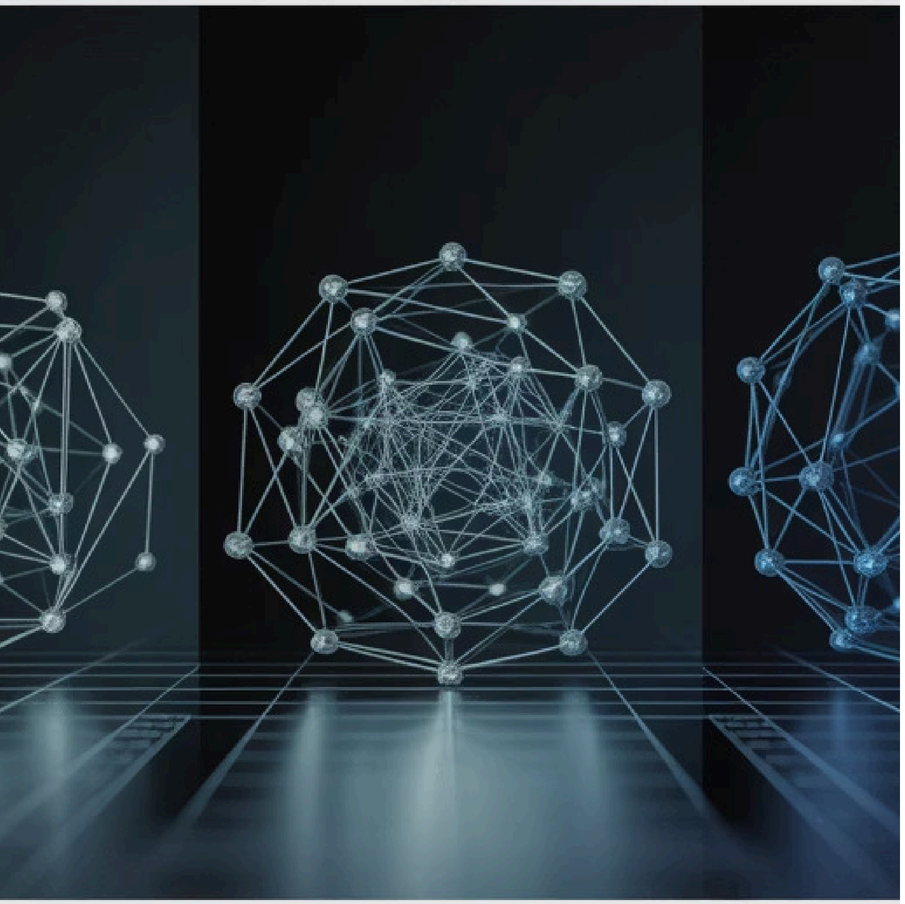
Belief Update: 95% certain

Human Approach

- Gather evidence incrementally
- Update beliefs with each clue
- Store relationship patterns
- Make probabilistic judgements

Neural Network Parallel

- **Trial** = Model iteration
- **Evidence** = Input observations
- **Belief Update** = Probability computation
- **Memory** = Network structure



The Graphical Models Landscape

Bayesian Network

Directed graph with arrows showing cause → effect relationships

- Nodes represent variables
- Arrows show direct influence
- Captures causal structure

Markov Random Field

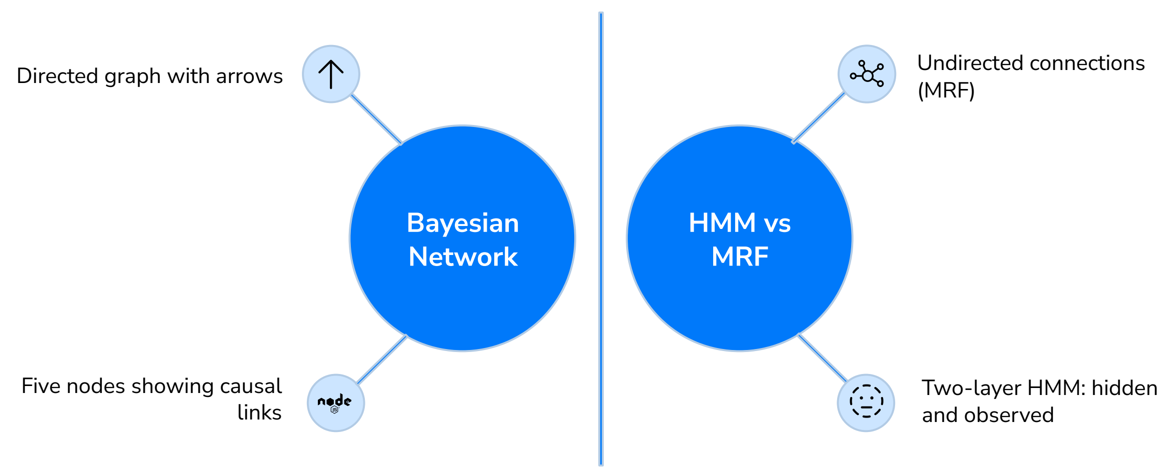
Undirected graph with symmetric relationships

- No directional arrows
- Bidirectional connections
- Perfect for spatial data

Hidden Markov Model

Sequential temporal models with hidden states

- Two layers: hidden and observed
- Horizontal state transitions
- Vertical observations



Learning Objectives

01

Conceptual Understanding

Explain graphical models, distinguish between types, and understand conditional independence

02

Technical Skills

Build Bayesian Networks, perform inference, apply HMMs, and implement algorithms

03

Practical Application

Choose the right model, troubleshoot issues, and implement tracking algorithms

04

Career Readiness

Answer interview questions and apply concepts to real-world scenarios

What Are Graphical Models?

Simple Definition

Graphical models **represent complex probability distributions** using graphs:

- **Nodes** = Random variables
- **Edges** = Relationships between variables

Everyday Analogy

Think of Facebook's friend network:

- Each person = Node
- Friendship = Edge
- Friends influence each other
- Non-friends are independent

Key Insight: Instead of a giant probability table with $2^5 = 32$ entries, we break it down: $P(A,B,C,D,E) = P(A) \times P(B|A) \times P(C|A) \times P(D|B,C) \times P(E|D)$. This is much more efficient!

Industry Applications

Healthcare Diagnosis

$50B market - Improves accuracy by **23%**, reduces unnecessary tests by **31%**

Fraud Detection

Catches **87%** of fraud cases, reduces false positives by **40%**

Autonomous Vehicles

Processes **100 objects/second** in real-time for safe navigation

Speech Recognition

Powers Siri, Alexa, Google Assistant - **$10B+ market** with 97%+ accuracy

Recommendation Systems

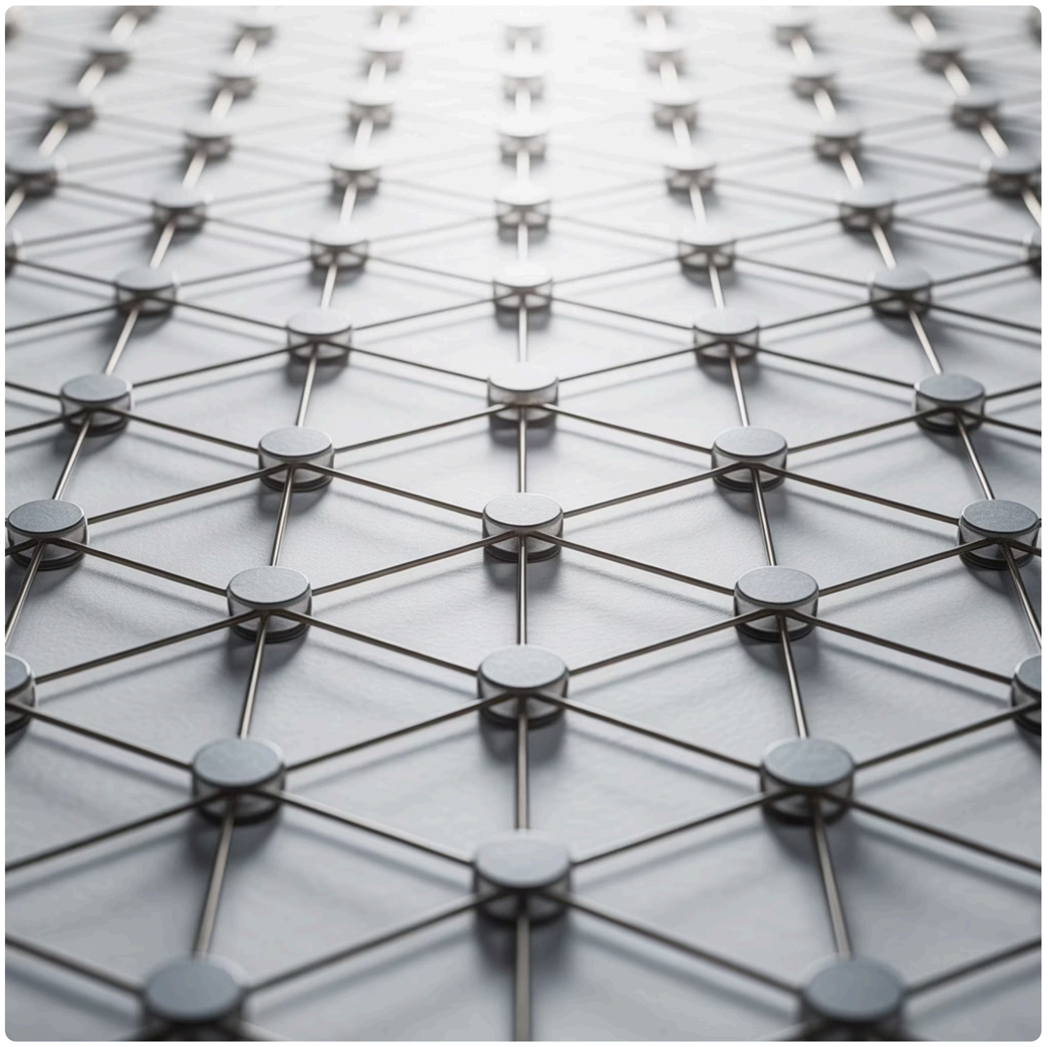
Increases conversion rates by **15-25%** for Amazon and Netflix

Bayesian Networks Explained

Definition

A **directed acyclic graph (DAG)** where:

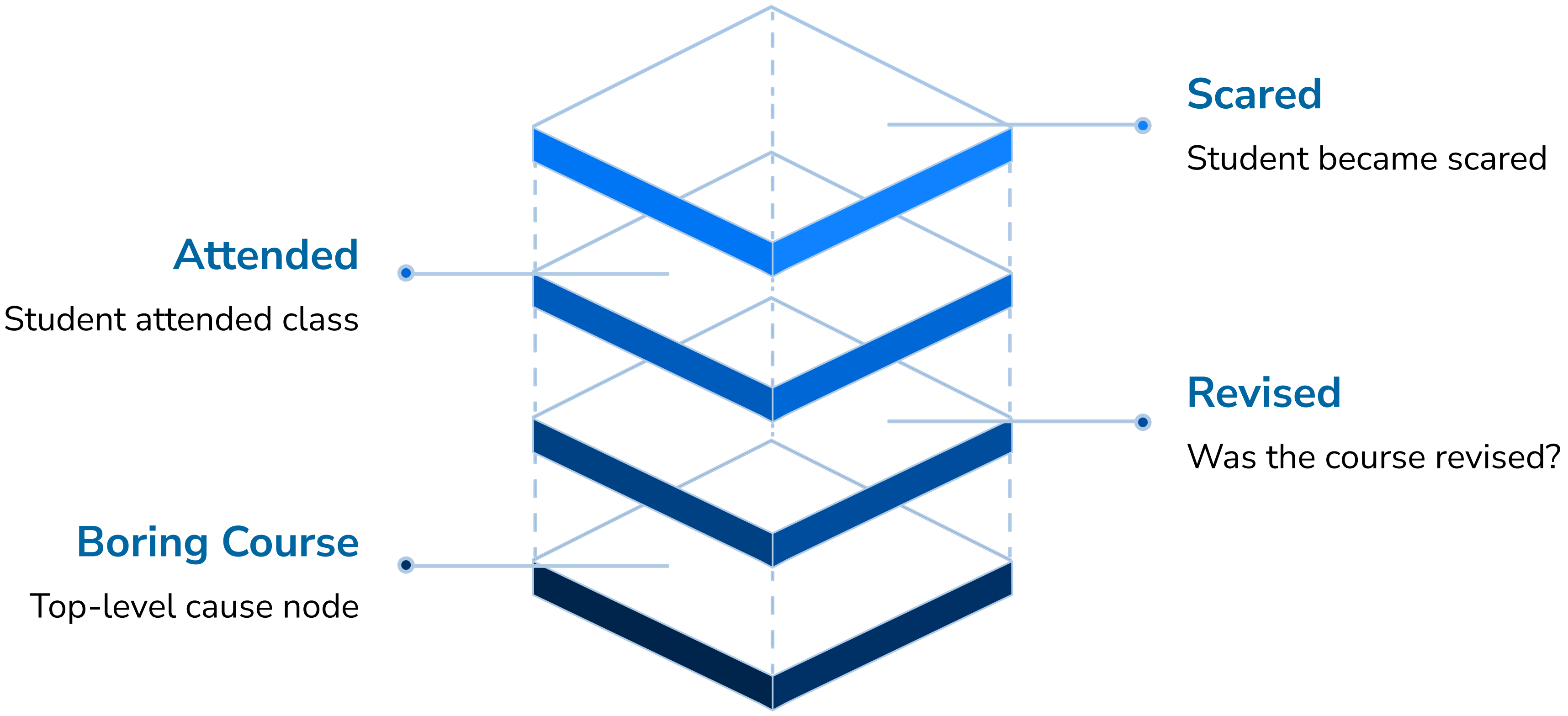
- Each node = random variable
- Directed edges = direct influence
- Each node has a conditional probability table



Key Properties

1. **Directed:** Arrows show cause → effect
2. **Acyclic:** No loops allowed
3. **Probabilistic:** Captures uncertainty

The Exam Fear Example



Variables

- **B:** Is the course boring?
- **R:** Did you revise?
- **A:** Did you attend lectures?
- **S:** Are you scared before exam?

What Each Node Stores

- B: P(Boring) = 0.5
- R: P(Revised | Boring)
- A: P(Attended | Boring)
- S: P(Scared | Revised, Attended)

How Inference Works

Problem: Given observations, compute probability of unknown variables

Step 1: Identify Structure

Observed: S = Scared

Query: R = Revised?

Path: R → S

Step 2: Apply Bayes' Rule

P(R|S) = P(S|R) × P(R) / P(S)

Step 3: Marginalize Hidden Variables

Sum over all values of B and A

Step 4: Use Probability Tables

Look up conditional probabilities

Step 5: Calculate Answer

P(Revised | Scared) = **39%**

Complexity Challenge: For N nodes with K values each: O(K^N) - exponential! Solution: Use approximation algorithms for large networks.

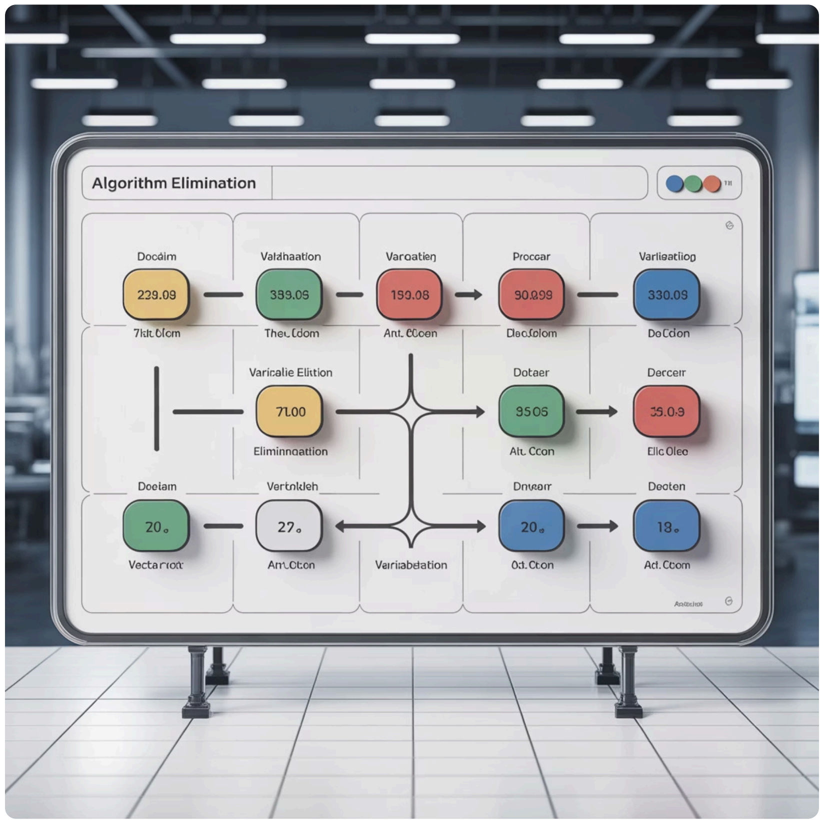
Variable Elimination Algorithm

The Problem

Computing exact probabilities is exponentially expensive

The Solution

Eliminate variables one at a time, from leaves to root



01

Create λ Tables

For each variable with all possible value combinations

02

Eliminate Observed Variables

Remove incorrect rows and the variable column

03

Eliminate Hidden Variables

Multiply tables, sum out variables, create new tables

04

Normalize

Divide by sum to get final probabilities

Example: Eliminating R (Revised)

R	A	S	λ
T	T	T	0.05
T	T	F	0.31
F	T	T	0.37
F	T	F	0.63

Computational Benefit: Reduces O(K^N) to O(N × K^w) where w is tree-width

Quick Check

1

Conceptual Question

In a Bayesian Network, if there is NO edge between nodes A and B, what does this mean?

- a) A and B are independent
- b) A and B are conditionally independent given their parents
- c) A causes B
- d) We have no information

2

Predictive Question

In the exam fear network, you observe: Course is boring (B=T), Student attended (A=T), Student revised (R=T). What about S (Scared)?

- a) High probability of being scared
- b) Low probability of being scared
- c) Equal probability
- d) Cannot determine

3

Practical Question

You're building a spam filter using a Bayesian Network. Which variables would you include as nodes?

- a) Email text only
- b) Sender, subject, body, links, attachments
- c) Just spam/not-spam label
- d) User's inbox history only

Answers: 1. b - Conditional independence given parents | 2. b - Low probability (attended AND revised) | 3. b - Multiple relevant features

Real Project Story

Medical Diagnosis at Mayo Clinic (2019)

The Challenge

Business Problem: Fast, accurate heart attack risk assessment needed

- 1,200 patients per day across 5 hospitals
- Unnecessary admissions cost **$2.1M annually**
- 15% of low-risk patients admitted unnecessarily

Initial Approach Failed

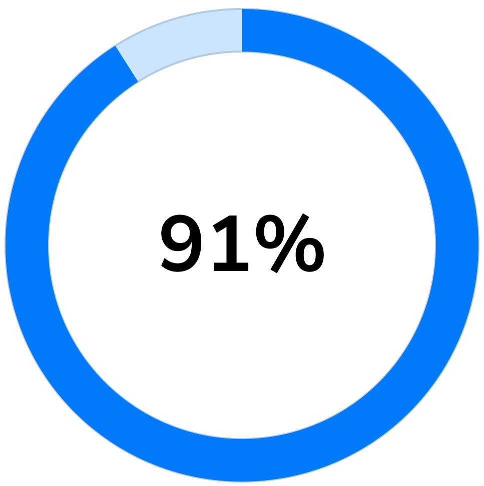
Simple decision tree: 78% accuracy, poor with missing data

The Solution

Bayesian Network: 35-node network with expert-defined structure

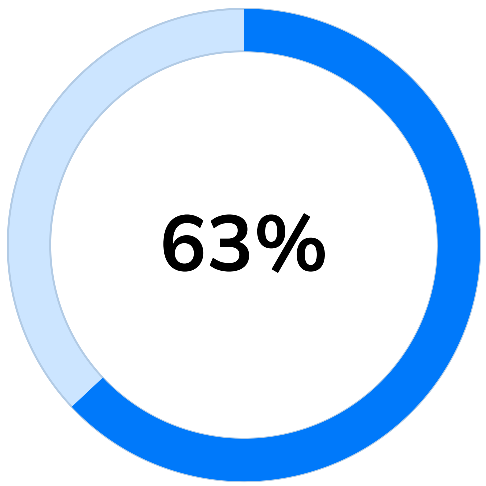
Key Innovation: Handled missing lab results through probabilistic inference

Implementation: 3 months, Python with pgmpy, 5-person team



Accuracy Achieved

Up from 78%



Reduction in Unnecessary Admissions

From 22% to 8%

Annual Savings

Business impact

Key Lesson: Start with domain expertise, then fine-tune with data. Fully-automated structure learning gave poor results initially.

Markov Random Fields (MRFs)

What's Different from Bayesian Networks?

5 nodes with directed arrows

Shows causal A → B relations

Bayesian Network

Markov Random Field

Same 5 nodes with undirected links

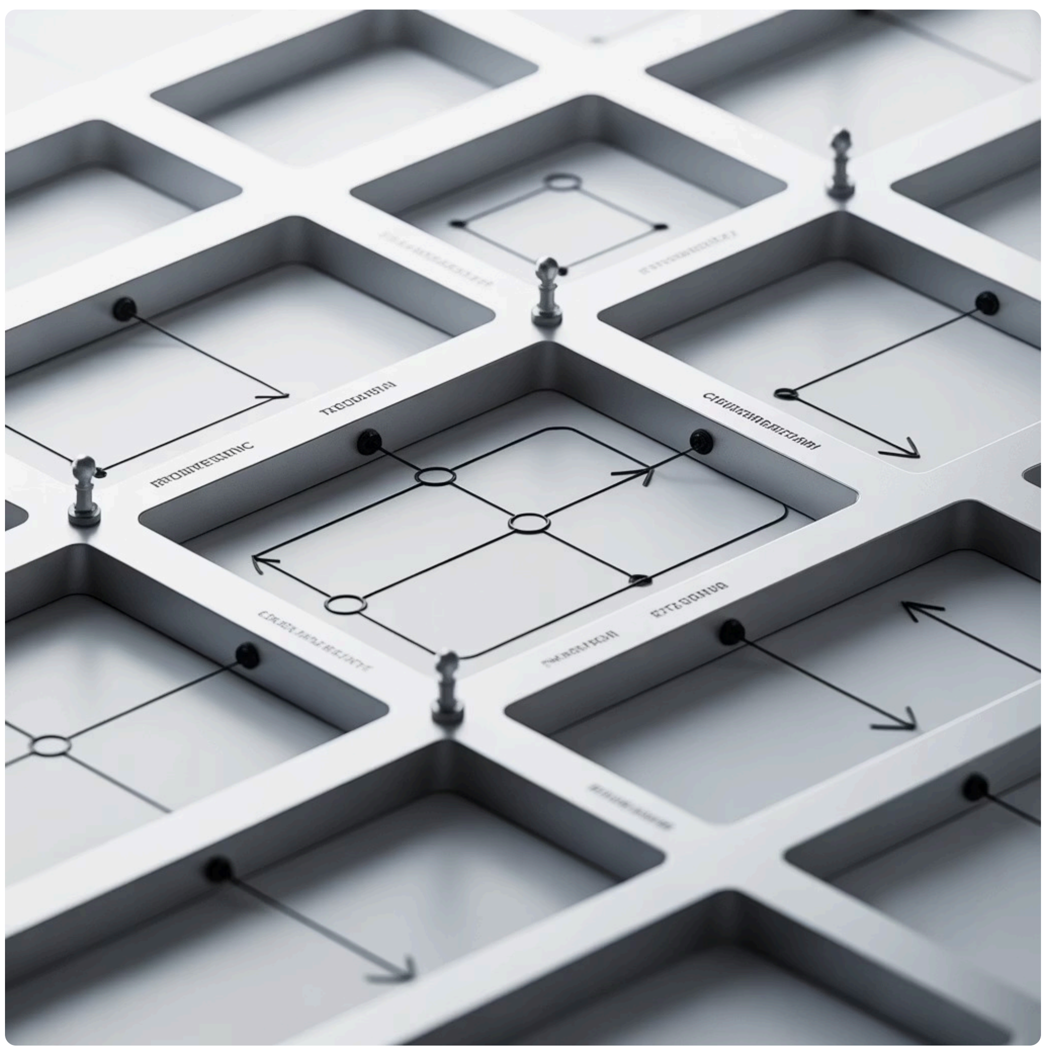
Highlights symmetric relationships

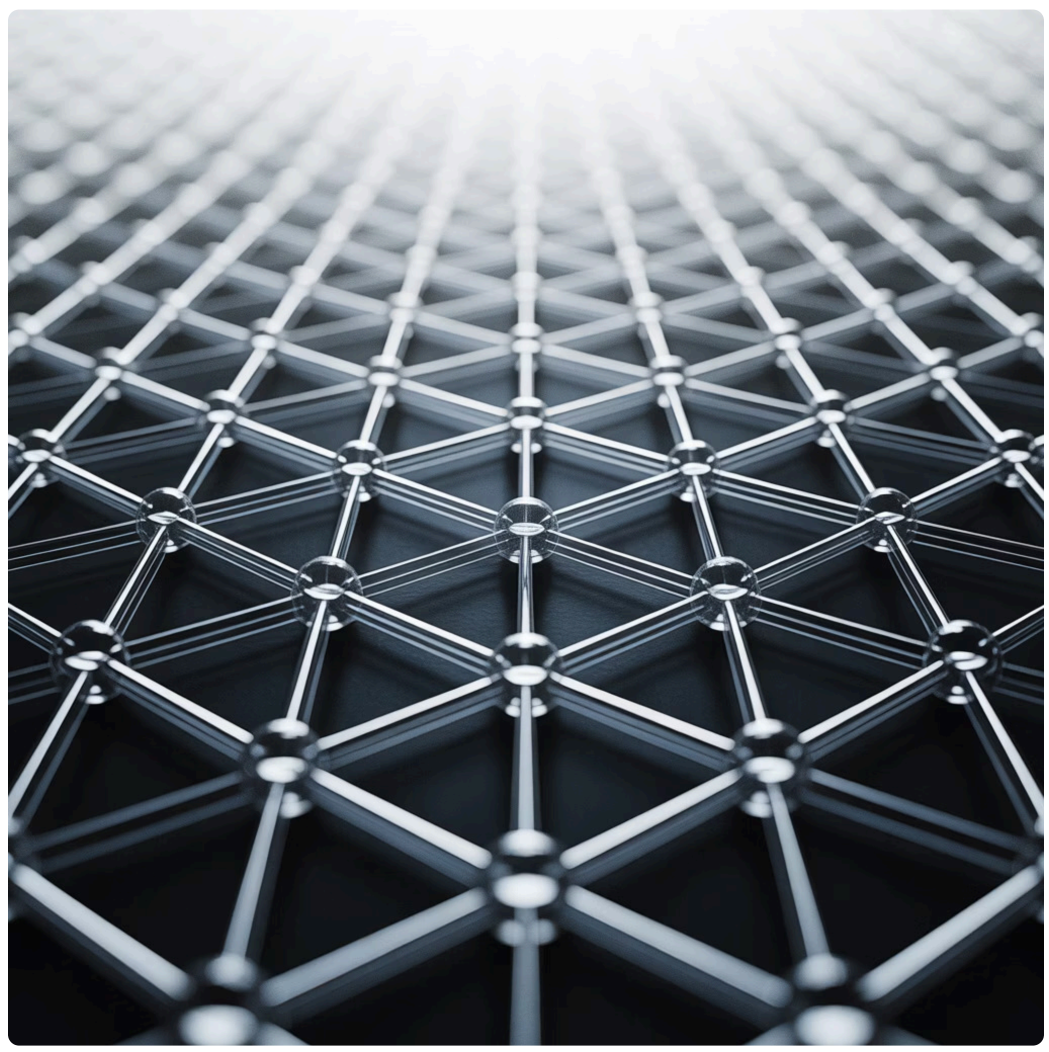
Bayesian Networks

- Directed edges (A → B)
- A causes B
- Clear cause-effect

Markov Random Fields

- Undirected edges (A — B)
- A and B are related
- Symmetric relationships





When to Use MRFs

Symmetric Relationships

Friendship networks where relationships are mutual with no clear direction

Spatial Data

Images, maps, grids where neighboring elements influence each other

Pairwise Interactions

When pairwise interactions matter more than directed causation

The Energy Function: MRFs use energy instead of probabilities. Lower energy = More likely configuration. Perfect for image denoising where neighboring pixels should have similar colors.

MRF Image Denoising

Real Example: Removing Noise from Images

The MRF Model

$$
\begin{aligned}
\text{Energy } E(I) &= \Sigma\,[\text{node energy}] + \Sigma\,[\text{edge energy}] \\
&= -\zeta\, \Sigma\, I(i,j) \times I'(i,j) \\
&\quad -\eta\, \Sigma\, I(i,j) \times I(\text{neighbor})
\end{aligned}
$$

Parameters

- **ζ (zeta) = 1.5:** Trust noisy observation
- **η (eta) = 2.1:** Neighboring pixels should agree

01

Start

Begin with noisy image

02

Compute

Energy for each pixel value

03

Update

Pick lower energy value

04

Repeat

Until convergence

10%

Initial Noise

Random pixel flips

<1%

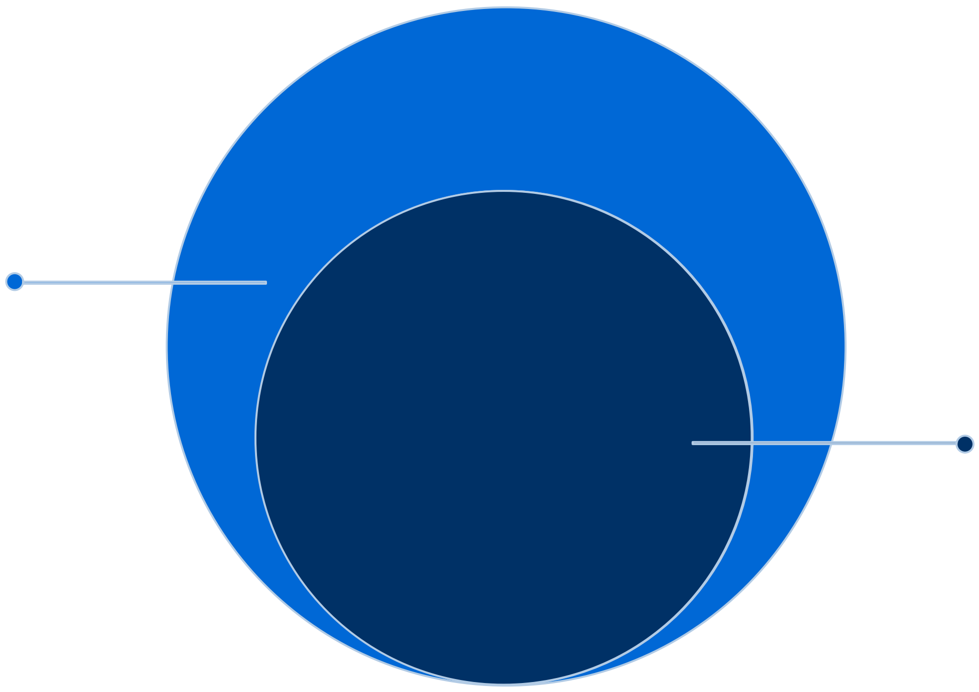
Final Error

After reconstruction

Why It Works: The MRF enforces spatial coherence - neighboring pixels should be similar!

Hidden Markov Models (HMMs)

Sequential Patterns with Hidden States



Observations

O1, O2, O3, O4 linked vertically to states

Hidden States

S1, S2, S3, S4 connected horizontally

The Student Behaviour Example

Hidden States (What Student Did)

- **TV:** Watched TV
- **Party:** Went to party
- **Pub:** Went to pub
- **Study:** Actually studied

Observations (What Professor Sees)

- Tired
- Hungover
- Scared
- Fine

1

Transition Probabilities

How states change over time

- P(Study → Pub) = 0.25
- P(Study → Study) = 0.05
- P(Party → Pub) = 0.05

2

Emission Probabilities

What we observe from each state

- P(Tired | TV) = 0.2
- P(Hungover | Party) = 0.4
- P(Scared | Study) = 0.3

The Markov Property: Next state depends ONLY on current state, not entire history: $P(S_3 \mid S_2, S_1, S_0) = P(S_3 \mid S_2)$

HMM Three Fundamental Problems

Problem 1: Evaluation

Forward Algorithm

Question: Given observation sequence, what's its probability?

Example: See "tired, tired, fine" - how likely is this?

Use Case: Speech recognition - is this audio English?

Problem 2: Decoding

Viterbi Algorithm

Question: Given observations, what's the most likely hidden state sequence?

Example: See "tired, hungover, fine" - what did student do?

Use Case: Part-of-speech tagging, gene prediction

Problem 3: Learning

Baum-Welch Algorithm

Question: Given observations, learn the best model parameters

Example: Observe many students, learn probabilities

Use Case: Training speech recognition systems

Complexity Comparison

Method	Naive	Efficient Algorithm
Evaluation	$O(N^T \times T)$	**$O(N^2 \times T)$** - Forward
Decoding	$O(N^T)$	**$O(N^2 \times T)$** - Viterbi
Learning	$O(N^T)$	**$O(N^2 \times T \times iter)$** - Baum-Welch

Where N = # states, T = sequence length

The Forward Algorithm

Computing P(observations | model) Step by Step

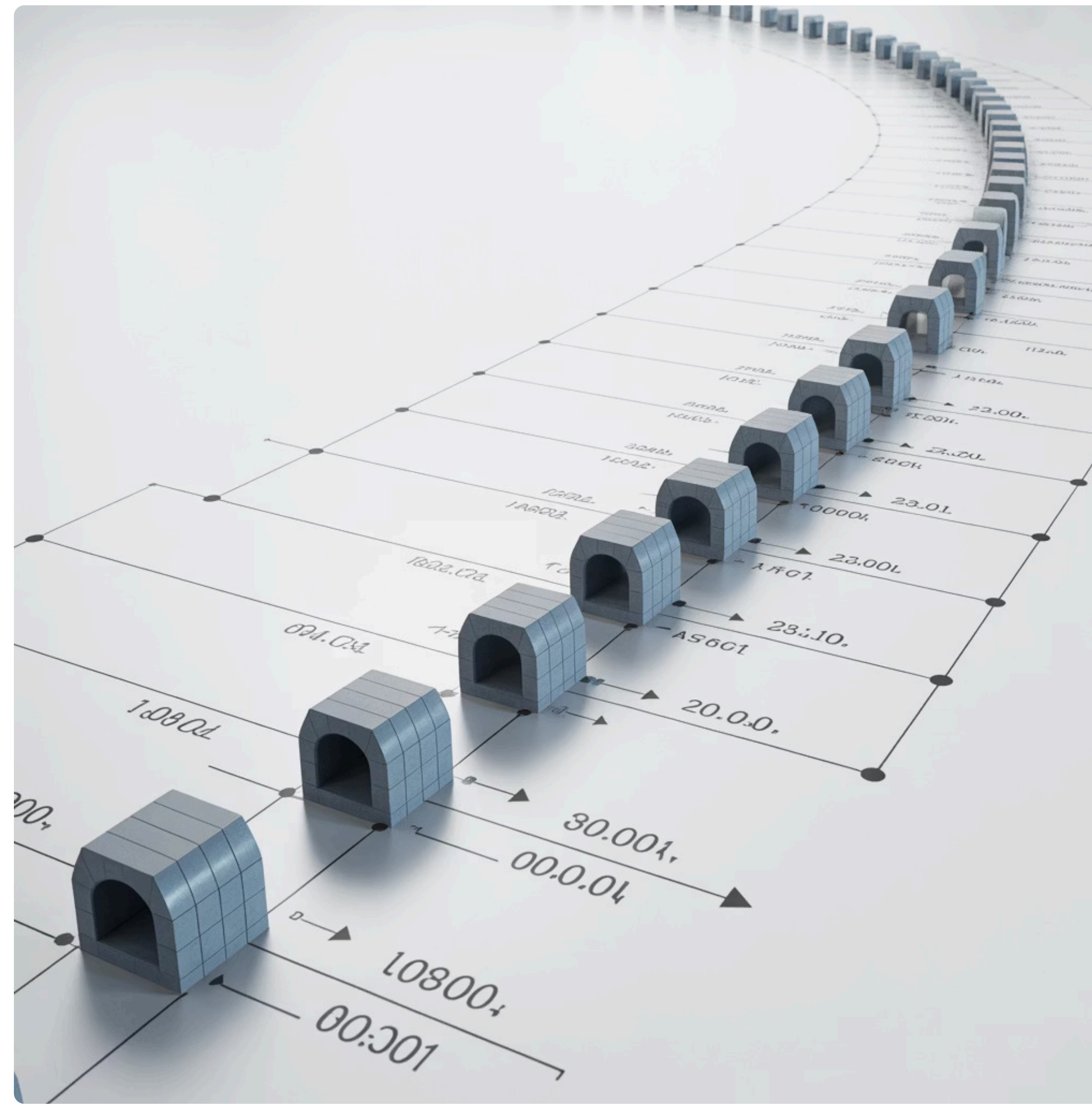
Setup

- **Observations:** (tired, tired, fine)
- **States:** {TV, Party, Pub, Study}
- **Goal:** P(O | model)

Key Idea

Build up probabilities incrementally using **α values**

α(i, t) = Probability of observing sequence up to time t AND being in state i at time t



Step 1: Initialize (t=0)

α(TV, 0) = π(TV) × P(tired | TV) = 0.05
α(Pub, 0) = π(Pub) × P(tired | Pub) = 0.1
α(Party, 0) = 0.075
α(Study, 0) = 0.075

Step 2: Forward Recursion (t=1)

α(Pub, 1) = P(tired | Pub) ×
Σ[α(i,0) × P(i→Pub)]
= 0.4 × (0.05×0.6 + 0.1×0.4 + ...)
= 0.022

Step 3: Repeat for t=2

Continue recursion for observation "fine"

Step 4: Final Probability

P(O) = Σ α(i, T)
Sum over all final states

Computational Savings: $O(N^2T)$ instead of $O(N^T)$ - HUGE difference! For N=4 states and T=10 time steps: 160 operations vs 1,048,576 operations!

The Viterbi Algorithm

Finding the Best Path Through Hidden States

The Problem

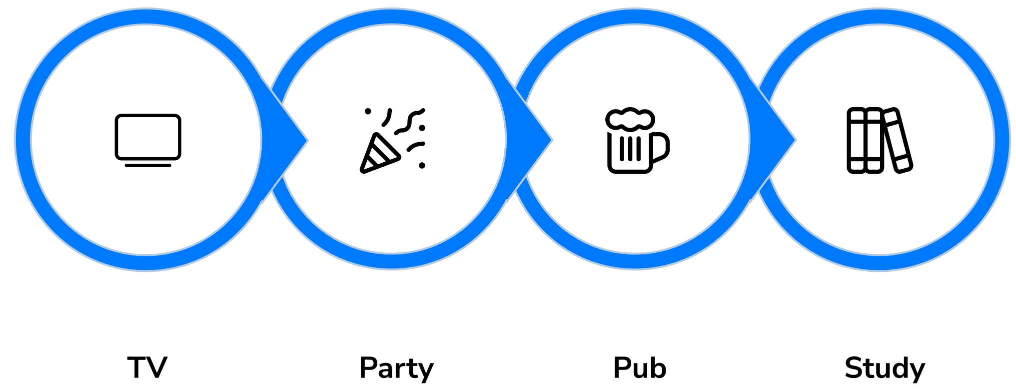
Observations: (fine, hungover, hungover, fine, tired, fine, fine, hungover)

Question: What did the student do each night?

Key Idea

Track the **best path** to each state at each time

$\delta(i, t)$ = Probability of most likely path ending in state i at time t



1

Initialize

$\delta(i, 0) = \pi(i) \times P(o_0|i)$

$\phi(i, 0) = 0$ [backpointer]

2

Recursion

$\delta(i, t) = \max[\delta(j, t\text{-}1) \times P(j \rightarrow i)] \times P(o_t|i)$

$\phi(i, t) = \operatorname{argmax}[\delta(j, t\text{-}1) \times P(j \rightarrow i)]$

3

Termination

$q^* = \operatorname{argmax}[\delta(i, T)]$

Find best final state

4

Backtrack

Follow ϕ pointers backwards

Reconstruct optimal path

Result for Example

1 — **Night 1:** Study

2 — **Night 2:** Pub

3 — **Night 3:** Pub

4 — **Night 4:** Study

5 — **Night 5:** TV

6 — **Night 6:** Study

7 — **Night 7:** Study

8 — **Night 8:** Party

Probability: 7.65×10^{-9} (very small, but best among all possibilities!)

Tracking Methods: Kalman Filter

Estimating State from Noisy Measurements

GPS Navigation

$75B market - Smooths noisy GPS signals for accurate positioning

Robotics

Tracks robot position and velocity for precise movement control

Aerospace

Guides missiles and spacecraft with high precision navigation

Finance

Tracks and predicts stock prices from noisy market data

State Equation (How Object Moves)

$x(t+1) = A \times x(t) + B \times u(t) + w(t)$

x = state (position, velocity)
u = control input (acceleration)
w = process noise

Measurement Equation (What We Observe)

$y(t) = H \times x(t) + v(t)$

y = measurement (e.g., GPS reading)
v = measurement noise

Key Insight: The Kalman Filter is **optimal** for linear systems with Gaussian noise! It's the best possible estimator under these conditions.