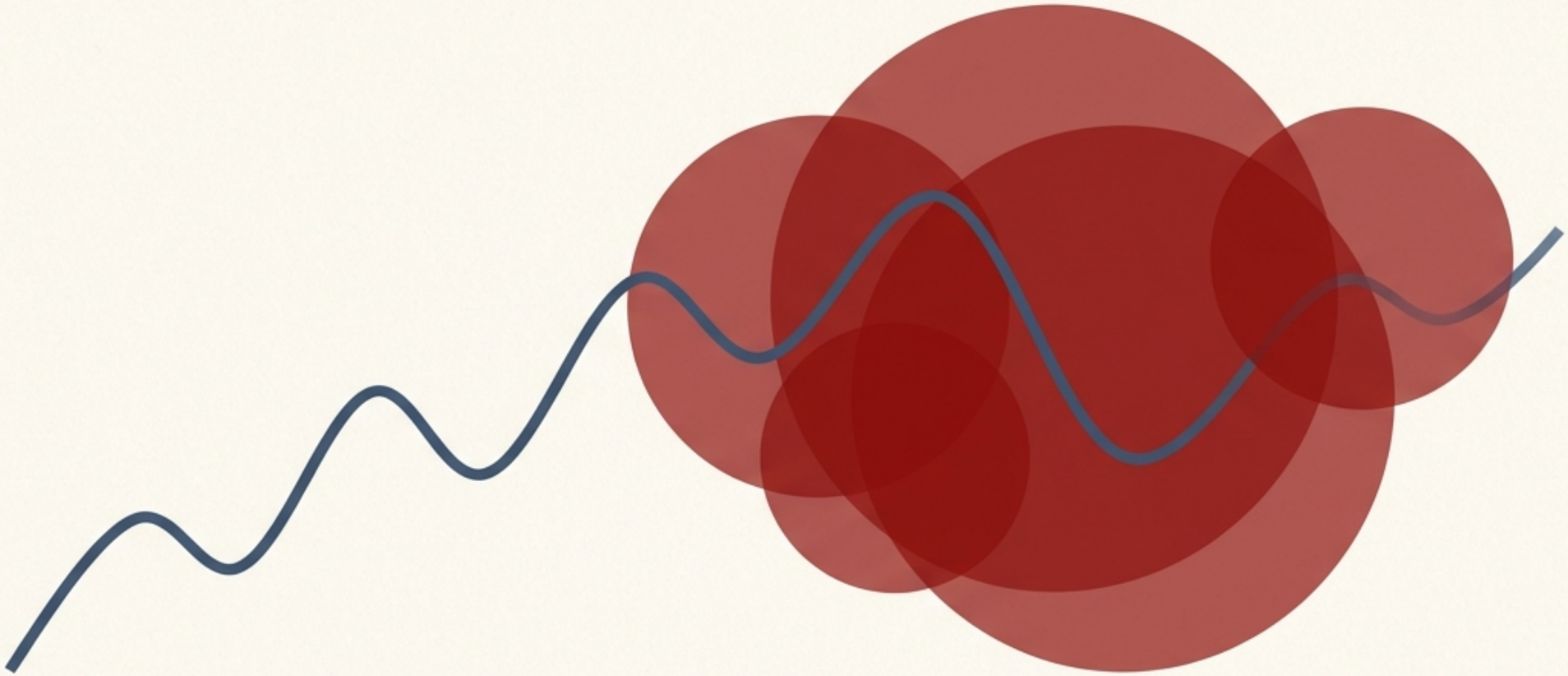
From Observable Patterns to Hidden Realities

A Journey into Markov Chains and Hidden Markov Models

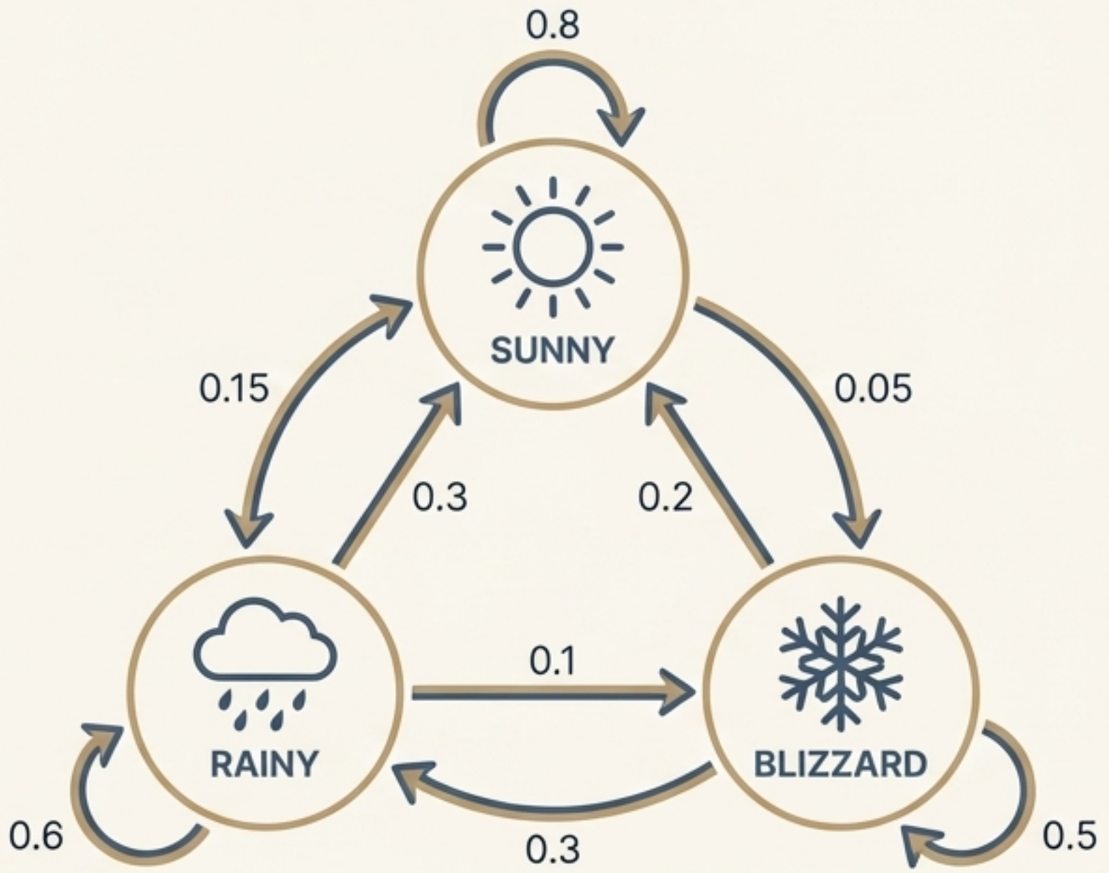


The World We Can See: When the Past Informs the Future

We begin with a simple idea: the future state of a system often depends on its current state. This is the essence of a **Markov Chain**. It's a stochastic model where the probability of a future event depends *only* on the current event, a property called 'memorylessness.'

Example: Let's model the weather. The chance of rain tomorrow is heavily influenced by whether it's raining today, not by the weather a week ago.

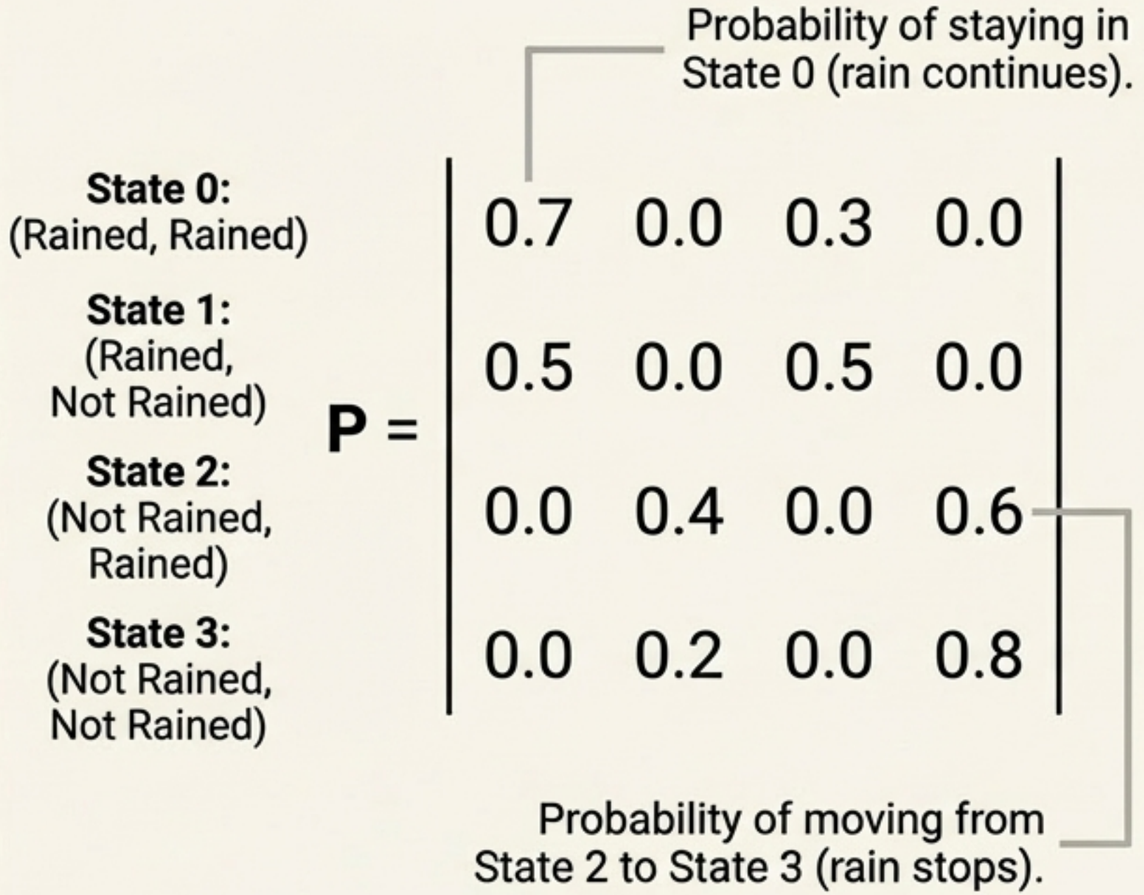
Key Property: The Markov property states: $P(X_{t+1} \mid X_t, X_{t-1}, \ldots, X_1) = P(X_{t+1} \mid X_t)$. Given the present, the future is independent of the past.



Formalizing Memory: States and Transitions

A Markov Chain is defined by two key components:
1. **States (Q):** A set of distinct conditions the system can be in. (e.g., {Sunny, Rainy, Blizzard}).
2. **Transition Probability Matrix (A):** A matrix where element a_{ij} represents the probability of moving from state i to state j. Each row must sum to 1.

Example: Weather Transition Matrix: Let's say our states are {State 0: Rained today & yesterday, State 1: Rained today, not yesterday, State 2: Rained yesterday, not today, State 3: No rain either day}. The transition matrix `$P$` captures the probabilities, such as "if it rained the last two days days (State 0), there's a 0.7 probability it will rain tomorrow."



The Plot Twist: What if We Can Only See the Effects?

Markov Chains are powerful, but they assume we can directly observe the state of the system. What happens when we can’t?

The Mystery: The Occasionally Dishonest Casino

The sequence of numbers rolled. This is what we can see.

The Unseen Cause

**Fair Die (F)

**Loaded Die (L)

The casino might be switching between a Fair Die and a Loaded Die.
We can’t see which die is being used for any given roll.

The true "state" (Fair or Loaded die) is **hidden**. We must **infer** it from the sequence of observations (the dice rolls). This is the problem that a **Hidden Markov Model (HMM)** is designed to solve.

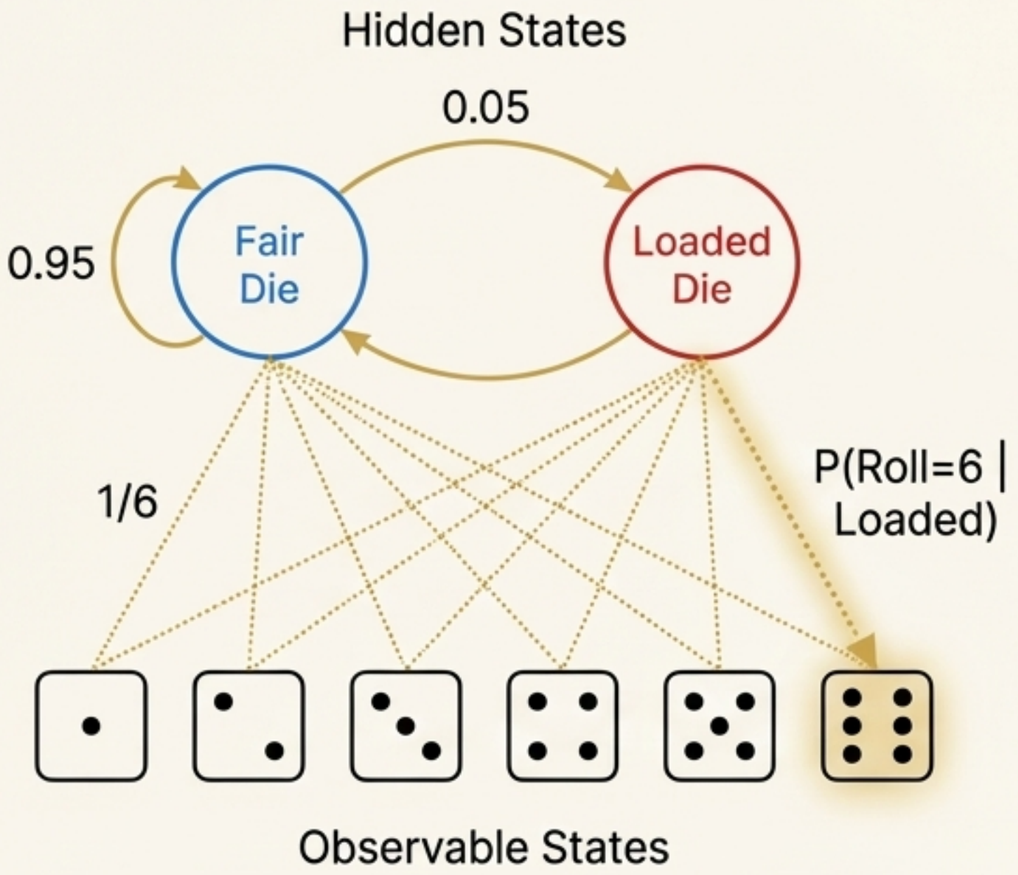
Anatomy of a Mystery: Defining the Hidden Markov Model

An HMM adds two new layers to our model:

- **Hidden States (Q)**: The underlying, unobservable states. In our casino: { **Fair Die** (blue), **Loaded Die** (red)}.
- **Observations (O)**: The visible outputs. In our casino: `{1, 2, 3, 4, 5, 6}`.

The model is governed by three sets of probabilities:

1. **Transition Probabilities (A)**: The probability of switching between hidden states. e.g., `P(Loaded | Fair) = 0.05`.
2. **Emission Probabilities (B)**: The probability of an observation being generated from a hidden state. e.g., `P(Roll=6 | Loaded Die)`.
3. **Initial Probabilities (π)**: The probability of starting in a particular hidden state. e.g., `P(Start with Fair Die) = 0.5`.



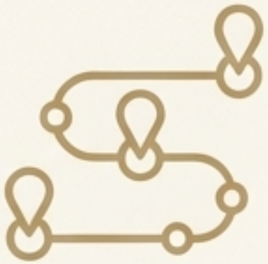
The Gambler’s Questions: Unraveling the Casino’s Secret

The structure of an HMM naturally leads to three fundamental questions we might ask to solve our mystery. The algorithms of HMMs are simply the tools to answer them.

Question 1: Evaluation

Given the sequence of rolls I've seen, what is the total probability of it happening with this casino's system?

The Forward Algorithm



Question 2: Decoding

Okay, I saw these rolls. What's the single most likely sequence of Fair and Loaded dice the dealer actually used?

The Viterbi Algorithm

Question 3: Learning

I have no idea what the casino's strategy is. Can I watch the rolls and reverse-engineer their probabilities for switching dice and how biased their loaded die is?

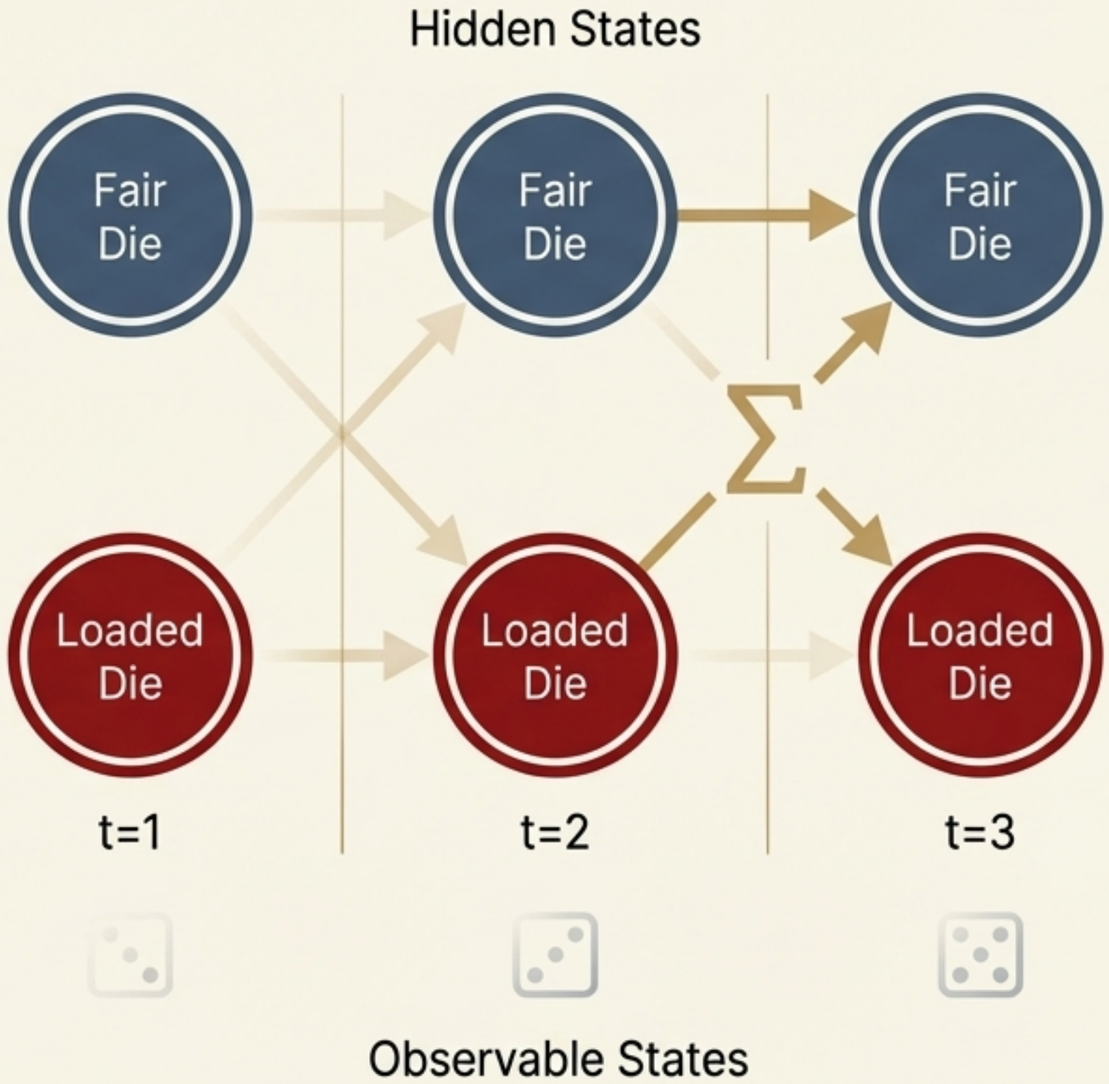
The Baum-Welch Algorithm

Question 1: How Likely Is What I Saw? (Evaluation)

The Challenge
Given an HMM model $\lambda = (A, B)$ and an observation sequence O, we want to determine the likelihood $P(O|\lambda)$. We can't just multiply probabilities because we don't know the hidden path.

The Solution: The Forward Algorithm
This algorithm efficiently computes the total probability of the observed sequence by summing the probabilities of *all possible* hidden state paths that could have generated it. It uses dynamic programming, calculating the probability of being in each hidden state at each time step, given the observations so far.

Analogy
It's like calculating the odds of a specific poker hand by considering every possible way the cards could have been dealt to produce that hand.



Question 2: What Was the Hidden Path? (Decoding)

The Challenge

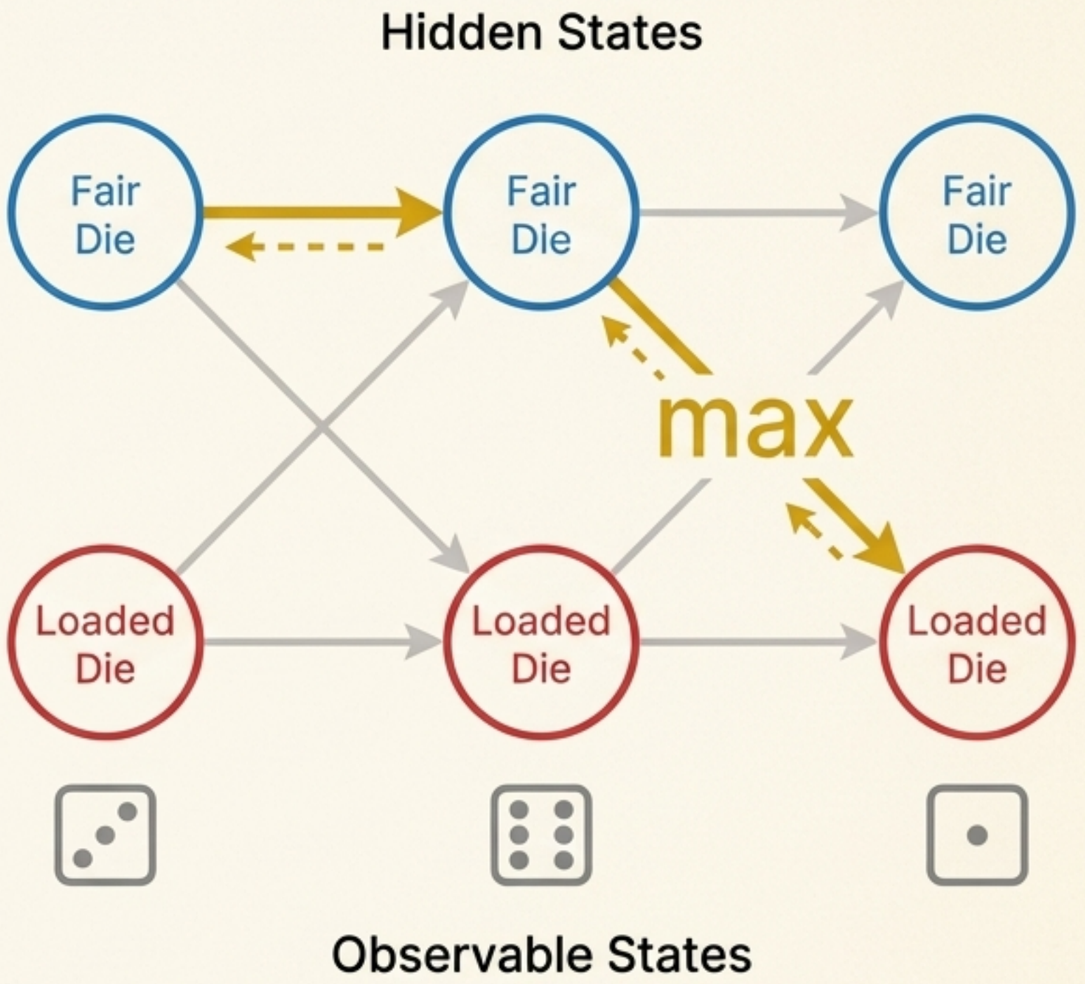
We don’t want the total probability of the sequence; we want the single *best* explanation. What was the most probable sequence of hidden states?

The Solution: The Viterbi Algorithm

Finds the optimal sequence of hidden states. It’s nearly identical to the Forward algorithm but replaces the **sum** with a **max**. At each step, instead of summing the probabilities of all paths leading to a state, it chooses the path with the highest probability and discards the others. It also stores “backpointers” to reconstruct the single most likely path at the end.

Use Case

This is decoding. It's used in speech recognition to find the most likely sequence of words (hidden) from an audio signal (observed).



Question 3: How Does the Casino Operate? (Learning)

The Challenge

What if we don't know the casino's transition (A) and emission (B) probabilities? Can we learn them just by observing the dice rolls? This is the HMM training problem.

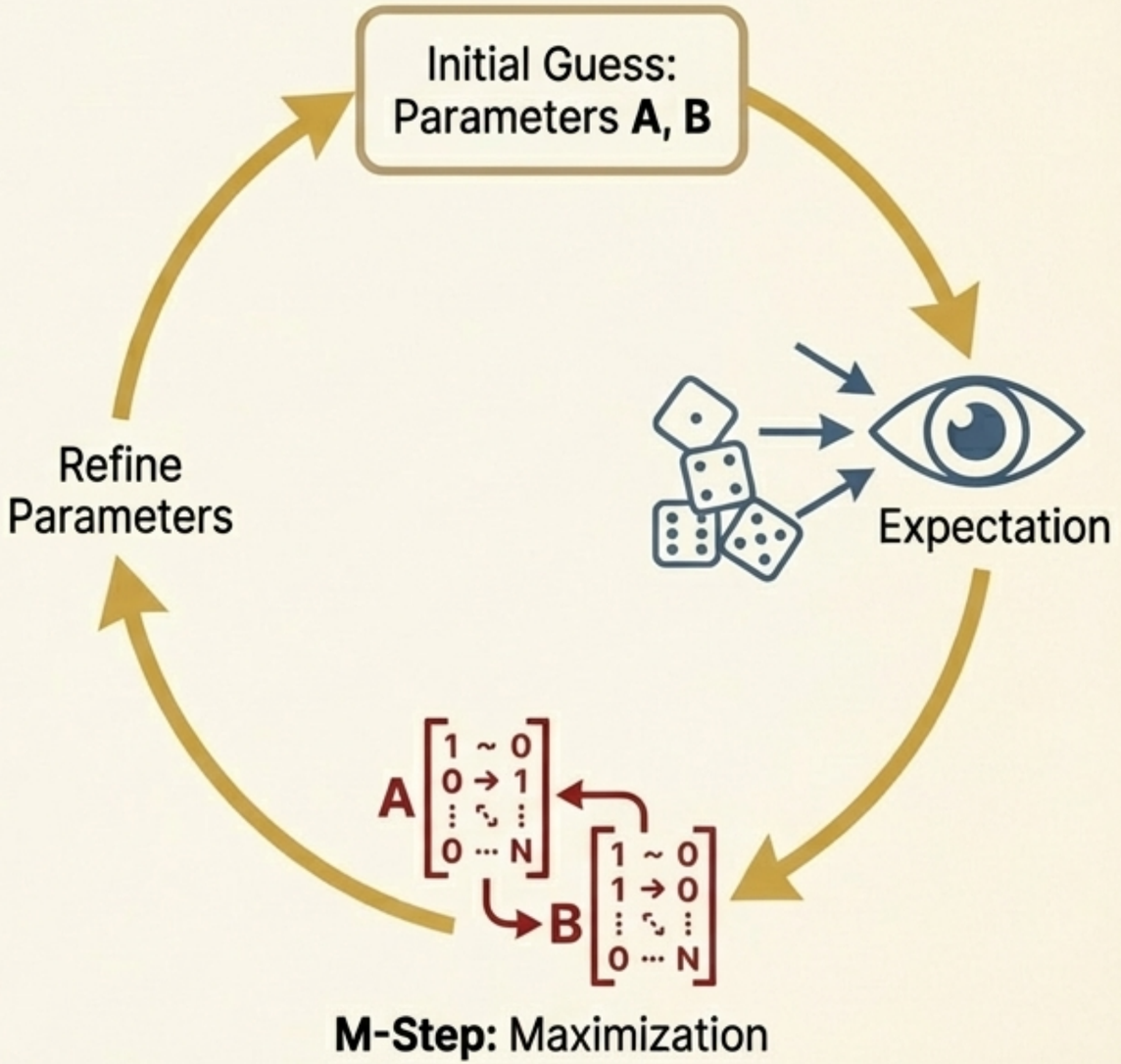
The Solution: The Baum-Welch Algorithm

This is an unsupervised learning algorithm, a special case of the Expectation-Maximization (EM) algorithm. It starts with an initial guess for A and B. It then iteratively refines these probabilities until they converge on values that best explain the observed data.

- **E-Step:** Computes the expected number of times each transition and emission is used.
- **M-Step:** Updates the A and B matrices based on these expected counts.

Analogy

It's like a detective who has a list of crime scene clues (observations) and iteratively develops and refines a theory of the culprit's methods (the model parameters) until it perfectly fits the evidence.



From Theory to Practice: Implementing HMMs

The power of HMMs is accessible through powerful libraries. While the underlying math is complex, fitting a model to data can be done in just a few lines of code. Below is an example using R's depmixS4 package to fit a two-state model to a stream of returns.

```r
# Create and fit the Hidden Markov Model
# family = gaussian -> assumes observations are normally distributed
# nstates = 2 -> we are looking for two hidden regimes
hmm <- depmix(returns ~ 1, family = gaussian(), nstates = 2, data=data.frame(returns=returns))

# Fit the model using the EM algorithm (Baum-Welch)
hmmfit <- fit(hmm, verbose = FALSE)

# Extract the posterior probabilities of being in each state
post_probs <- posterior(hmmfit)
```

Defining the number of hidden states.

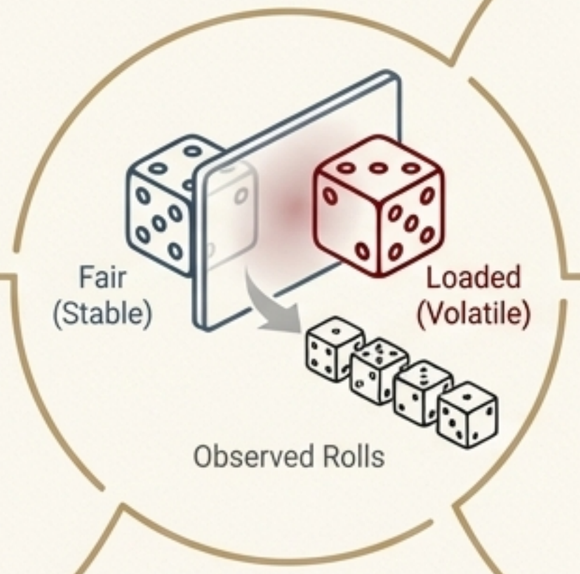
Training the model with the Baum-Welch algorithm

Briefly mention that similar libraries exist in Python, like hmmlearn.

```python
# Conceptually similar in Python
from hmmlearn import hmm
model = hmm.GaussianHMM(n_components=2, covariance_type="full")
model.fit(returns)
```

The Reveal: The Casino is Everywhere

The “Occasionally Dishonest Casino” is a powerful metaphor. The same framework of inferring hidden states from observed data is used to solve fundamental problems across multiple domains.



Natural Language Processing: Part-of-Speech Tagging

N-V-Adj

What is the hidden grammatical structure (noun, verb, adj) of a sentence? The observations are the words themselves.

Finance: Market Regime Detection

Are we in a hidden "Bull" or "Bear" market? The observations are daily asset returns.

Computational Biology: Gene Sequencing & Analysis

Identifying hidden protein-coding regions within an observed DNA sequence.

Engineering: Predictive Maintenance

Is a piece of equipment in a "healthy" or "failing" state? The observations are sensor readings.

Speech Recognition

A B
C
C...

Inferring a hidden sequence of words from an observed acoustic signal.

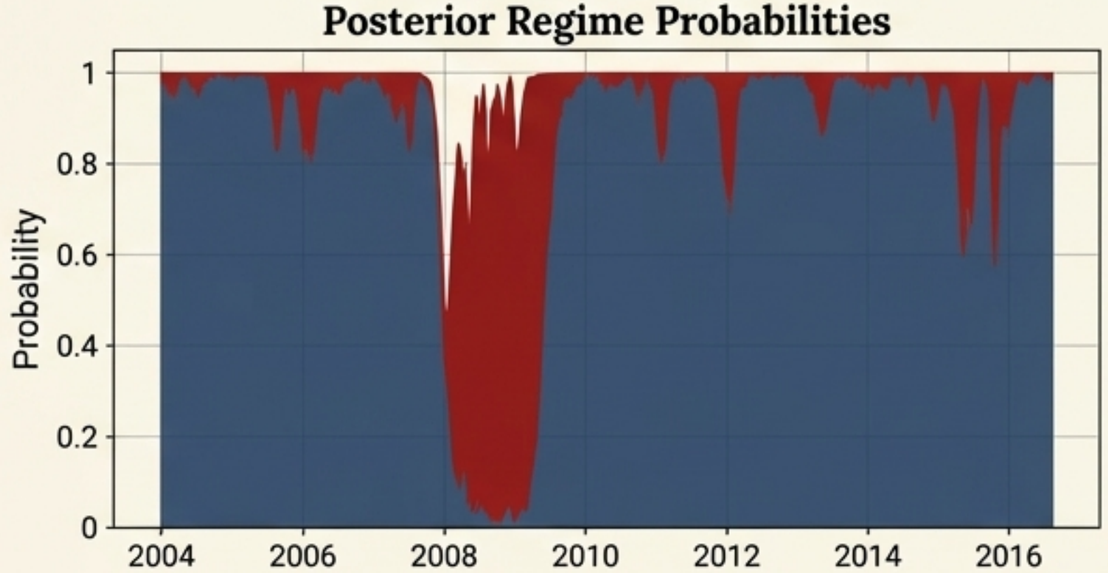
Case Study: Uncovering Hidden Market Regimes

The Problem: Financial markets exhibit different behaviors (regimes) like high-volatility ‘bear’ markets and low-volatility ‘bull’ markets. These are not directly observable but have huge implications for trading strategy. An HMM can identify these hidden states.

The Approach:

1. **Observations:** Use a time series of S&P500 daily returns (`^GSPC`).
2. **Model:** Fit a 2-state HMM (assuming two regimes: low-volatility and high-volatility). The model learns the mean and variance for each regime and the probabilities of transitioning between them.

The Result: The model outputs the probability of being in each regime at any given time.



The Hidden Markov Model Toolkit

The Components

Q : Hidden States ({Fair, Loaded})

O : Observations ({1, 2, ... 6})

A : Transition Matrix ($P(state_t|state_{t-1})$)

B : Emission Matrix ($P(observation_t|state_t)$)

π : Initial State Distribution

The Problem: Evaluation

Question: What is $P(Observations)$?

Algorithm: Forward Algorithm

Use: Model Scoring

The Problem: Decoding

Question: What is the most likely sequence of hidden states?

Algorithm: Viterbi Algorithm

Use: Inference (e.g., POS Tagging)

The Problem: Learning

Question: What are the best model parameters A and B?

Algorithm: Baum-Welch Algorithm

Use: Model Training (Unsupervised)

Beyond the Dice Roll: A Framework for Inference

Core Idea:
Hidden Markov Models provide a robust and mathematically sound framework for modeling dynamic systems where the underlying process is
underlying process is unobservable.

Key Takeaway:
By separating the hidden causes from the visible effects, HMMs allow us to look through the noise of data and infer the hidden structure of the world—from the strategy of a casino to the state of financial markets and the grammar of human language. They transform uncertainty into probabilistic insight.