UNIT 3

Monotonic Reasoning vs Non-Monotonic Reasoning

Monotonic reasoning is when conclusions always stay the same or only get stronger as more information is added. In other words, once you know something is true, adding more facts won't change it. For example, if you know "All apples are fruits," and then you learn more about different types of apples, your conclusion (that apples are fruits) won't change.

Non-monotonic reasoning, on the other hand, means that new information can change or reverse your conclusions. It's like reasoning that can "unlearn" or adapt. A real-life example is if you initially think someone is at home because their car is in the driveway, but later you find out the car is broken down, and now you conclude they're not home. Here, new information changes your previous assumption.

In summary, monotonic reasoning builds on facts, while non-monotonic reasoning adjusts conclusions as new facts or information is added.

Non-Monotonic Reasoning:

Non-monotonic reasoning refers to a type of logical thinking where conclusions can change when new information is added. Unlike monotonic reasoning, which always leads to stronger conclusions, non-monotonic reasoning allows conclusions to be revised or even rejected when additional, sometimes conflicting, facts come into play.

Example:

Let's say you have a knowledge base with the following information:

1. **Typically, birds fly.**
2. **Penguins do not fly.**
3. **Tweety is a bird.**

Based on the first statement ("typically, birds fly"), you might conclude that Tweety, being a bird, should fly. This is a common assumption based on the generalization that most birds fly. However, when you add the second piece of information ("Penguins do not fly"), you encounter a contradiction. Penguins are also birds, but they don't fly.

Now, the conclusion about Tweety can change, because we know not all birds fly—especially if Tweety turns out to be a penguin. This is where non-monotonic reasoning comes in: your conclusion (that Tweety can fly) is revised based on the exception. You would now reason that "Maybe Tweety doesn't fly, because Tweety could be a penguin." The addition of new information has led to a change in the conclusion, which makes this reasoning non-monotonic.

In everyday life, we often use non-monotonic reasoning because the world is full of exceptions. We adapt our thinking as new, more specific information becomes available.

Methods for Handling Uncertain Knowledge:

1. **Non-Monotonic Reasoning:** Non-monotonic reasoning helps handle uncertainty by revising conclusions when new, conflicting information is added. This method doesn't assume that once something is concluded, it will always remain true. It's dynamic and adaptable, reflecting how human reasoning works in uncertain situations. For instance, if a new fact about Tweety is revealed (e.g., Tweety is a penguin), the previous conclusion that "Tweety flies" is revised. This flexibility is essential when dealing with real-world uncertainty, where facts change and exceptions arise.
2. **Statistical Learning:** Statistical learning involves using data and probabilities to handle uncertain knowledge. Instead of relying on strict logical rules, statistical learning looks at patterns in data to make predictions or decisions. It's particularly useful in situations where there's a lot of data but incomplete or uncertain information.

Example of Statistical Learning:

Imagine you have data about thousands of birds, and you know some fly while others don't. With statistical learning, you can build a model that, based on characteristics like species, weight, and wing size, predicts whether a bird is likely to fly or not. For instance, based on the data, the model might predict that "birds with larger wings are more likely to fly" and "penguins are less likely to fly."

If you input Tweety's characteristics into the model (say, Tweety is a small, winged bird), the model might predict that Tweety can fly, but if Tweety is identified as a penguin with specific features, the model might predict no flight. Unlike traditional logical reasoning, statistical learning doesn't rely on rigid rules; it uses observed patterns and probabilities to make educated guesses, handling uncertainty by considering a range of possibilities.

Summary:

Non-monotonic reasoning helps us adjust conclusions as new, conflicting information arises, and statistical learning uses data patterns to predict and handle uncertain knowledge. Both methods are valuable for navigating real-world complexity, where certainty is rare and facts can change.

Logics for Non-Monotonic Reasoning

In non-monotonic reasoning, the aim is to handle situations where conclusions might change when new, conflicting information is introduced. Several logical frameworks are used to manage uncertainty, and each has its own way of handling exceptions or missing information.

1. Default Logic:

Default logic allows us to assume things based on typical scenarios, but it can be retracted when new information contradicts the assumption.
The formula **A:B/C** can be read as: "If A is true, then typically B is true, unless we have a reason to think otherwise (C)."

Example:

- "Birds fly: Birds typically fly unless they are penguins."
- If Tweety is a bird, we *assume* Tweety can fly, but this can be revised if we learn that Tweety is a penguin (the exception).

2. Abduction:

Abduction is about making the best guess or hypothesis based on available evidence. It's like solving a mystery by proposing the simplest explanation that fits the facts.

Example:

- **Evidence:** Tweety is a bird, and it isn't flying.
- **Hypothesis (Abduction):** The best explanation might be that Tweety is a penguin (since penguins don't fly).

3. Inheritance:

Inheritance logic is used when a property or characteristic is passed down from a general category to a specific instance. It allows conclusions to be inherited from more general information.

Example:

- "All birds can fly" (generalization).
- If Tweety is a bird, we inherit the assumption that Tweety can fly, unless we find an exception like penguins.

4. Closed World Assumption (CWA):

The Closed World Assumption assumes that if something is not known to be true, it is false. It's a way of dealing with incomplete knowledge by assuming that anything not explicitly stated does not exist.

Example:

- "Tweety is a bird, and it doesn't fly" implies that Tweety can't fly in this world because the CWA says we only know what is stated, and nothing more.

5. Circumscription:

Circumscription is a method of limiting the set of possible scenarios to only the most typical ones, ignoring less likely possibilities.

Example:

- "Most birds fly" and "penguins don't fly." If we circumscribe the flying bird category to typical birds, we infer that Tweety, as a bird, likely flies unless specified otherwise (like penguin traits).

Summary:

These logics are ways of handling uncertainty and exceptions in real-world reasoning. Default logic assumes typical cases unless proven wrong, abduction makes educated guesses, inheritance allows general rules to apply to specific cases, CWA assumes the unknown is false, and circumscription narrows down possibilities to the most typical ones. All of these approaches help us reason in situations where knowledge is incomplete or changing.

Implementation Issues

Non-monotonic reasoning means that our conclusions can change when new information is added. Unlike normal logic (where once something is true, it stays true), in non-monotonic reasoning, earlier conclusions might be revised or rejected when new facts appear.

Implementation Issues

1. **Backtracking and Re-Evaluation**
    - If a system makes a decision based on current knowledge and later gets new information, it has to go back and correct itself.
    - This can be slow and inefficient, especially in large or complex systems.
2. **Handling Defaults and Exceptions**
    - Many AI systems use general rules (e.g., "Birds can fly"), but they must adjust when exceptions arise (e.g., "Penguins cannot fly").
    - Managing such changes dynamically is a challenge.
3. **Computational Cost**
    - Constantly checking and updating knowledge requires more memory and processing power, making it harder for real-time applications.

Real-Time Example: Self-Driving Cars

Imagine a self-driving car using AI to navigate roads. Initially, it assumes all traffic lights work correctly. However, if a camera detects that a signal is broken, the AI must **revise its conclusion** and decide based on other factors (e.g., stopping based on other cars' movement). This constant adjustment is non-monotonic reasoning.

If the AI struggles to update its beliefs efficiently, it could **cause accidents or delays**, showing why proper implementation is critical.

Statistical Reasoning: Probability & Conditional Probability in AI

Statistical reasoning helps AI make decisions based on uncertainty. **Probability** and **Conditional Probability** are essential for AI systems to predict events and make informed choices. Let’s break these concepts down simply.

1. What is Probability?

- Probability measures how likely an event is to happen.
- It ranges from **0 (impossible event)** to **1 (certain event)**.
- Formula: **P(A)= Number of ways A can happen / Total possible outcomes**

Example:

- If you roll a fair dice, the probability of getting a 3 is **1/6** because there are six possible outcomes.

2. What is Conditional Probability?

- Conditional probability tells us the likelihood of an event occurring **given that another event has already happened**.
- It is written as **P(A | B)**, meaning "Probability of A happening **given** B has already occurred".
- Formula: $\mathbf{P(A \mid B) = P(A \cap B)/P(B)}$

Where $\mathbf{P(A \cap B)}$ is the probability of both A and B happening together.

Example:

- Suppose a person is chosen randomly from a group where **40% are women** and **10% of all people are doctors**.
- If you know the chosen person is a woman, the probability that she is a doctor might **change** based on available data (e.g., 25% of women are doctors).
- So, **P(Doctor | Woman) = 0.25** (25%).

3. How AI Uses Probability & Conditional Probability?

A. Spam Email Detection

- AI analyzes whether an email is spam based on words used.
- **P(Spam | "Win a prize")**: If "Win a prize" appears, AI checks how often such emails were spam in past cases.
- If the probability is high, AI marks the email as spam.

B. Medical Diagnosis

- AI predicts diseases based on symptoms.
- **P(Disease | Fever, Cough)**: If a patient has a fever and cough, AI looks at past data to estimate how likely it is that the patient has flu, COVID-19, etc.

C. Self-Driving Cars

- AI calculates the probability of a pedestrian crossing based on movement.
- **P(Crossing | Person near road)**: If someone is standing near the road, AI checks past data to predict if they will cross.

4. Why is Conditional Probability Important in AI?

- Helps AI make **smarter decisions** by considering **prior knowledge**.
- Reduces **uncertainty** in predictions.
- Used in **Bayesian Networks, Machine Learning, and AI-based forecasting**.

By applying probability and conditional probability, AI can **adapt, learn, and improve decision-making** in real-world scenarios.

Statistical Reasoning: Bayes' Theorem with a Simple Example

- Bayes' Theorem is a powerful tool in **AI, statistics, and decision-making**. It helps update probabilities when new information is available.

1. What is Bayes' Theorem?

Bayes' Theorem calculates the probability of an event based on prior knowledge of related conditions. It is written as:

$$P(A|B) = P(B|A) \cdot P(A) / P(B)$$

Where:

- **P(A | B)** = Probability of **A happening given** B has occurred.
- **P(B | A)** = Probability of **B happening given** A has occurred.
- **P(A)** = Prior probability of **A happening**.
- **P(B)** = Probability of **B happening**.

Example:

A person takes a medical test for a rare disease. We need to find the probability that they **actually have the disease given that they tested positive**.

Given Data:

- **P(Disease) = 0.001** → Probability of having the disease (1 in 1,000 people).
- **P(No Disease) = 1 - P(Disease) = 0.999** → Probability of not having the disease.
- **P(Positive | Disease) = 0.9** → Probability of testing positive if the person has the disease (90% sensitivity).
- **P(Positive | No Disease) = 0.05** → Probability of testing positive if the person does **not** have the disease (5% false positive rate).

Step 1: Compute Total Probability of Testing Positive (P(Positive))

A person can test positive in two ways:

1. **They actually have the disease** and test positive:

$$P(\text{Positive} \mid \text{Disease}) \times P(\text{Disease}) = 0.9 \times 0.001 = 0.0009$$

2. **They do not have the disease** but still test positive (false positive):

$$P(\text{Positive} \mid \text{NoDisease}) \times P(\text{NoDisease}) = 0.05 \times 0.999 = 0.04995$$

Now, summing both cases:

$$P(\text{Positive}) = 0.0009 + 0.04995 = 0.05085$$

Step 2: Apply Bayes' Theorem

$$P(\text{Disease} \mid \text{Positive}) = P(\text{Positive} \mid \text{Disease}) \times P(\text{Disease}) / P(\text{Positive})$$

$$P(\text{Disease} \mid \text{Positive}) = 0.0009 / 0.05085$$

$$P(\text{Disease} \mid \text{Positive}) = 0.0177 \text{ or } 1.77\%$$

Final Answer:

If a person tests **positive**, the probability that they actually **have the disease** is only **1.77%**, despite the test being 90% accurate.

Why is this Important?

- Even though the test is **90% accurate**, false positives make the real probability of having the disease **much lower**.
- This is why doctors often require **multiple tests** before confirming a diagnosis.

Bayes' Theorem is widely used in **AI, medical diagnosis, spam filters, and risk assessment**.

Certainty Factor (CF)

1. What is a Certainty Factor (CF)?

- The **Certainty Factor (CF)** is a measure of how confident we are about a fact or event.
- It is used in expert systems and AI to **handle uncertainty** when making decisions.
- CF values range from **-1 (definitely false)** to **+1 (definitely true)**, with **0 meaning complete uncertainty**.

2. Where is Certainty Factor Used?

- **Medical Diagnosis Systems** (e.g., Expert systems like MYCIN).
- **AI-Based Decision Making** (e.g., Chatbots, Virtual Assistants).
- **Fraud Detection** (e.g., Identifying suspicious transactions).
- **Industrial Fault Detection** (e.g., Predicting machine failures).

3. How is Certainty Factor Calculated?

CF is calculated using:

CF = MB − MDCF

Where:

- **MB (Measure of Belief)** = How strongly we believe in the hypothesis (0 to 1).
- **MD (Measure of Disbelief)** = How strongly we disbelieve the hypothesis (0 to 1).

4. Real-Time Example: Medical Diagnosis (Flu Detection)

Scenario:
A doctor uses an AI system to diagnose flu based on symptoms like fever and cough.

- The AI system checks **Fever**: **MB = 0.8**, MD = 0.1
    - **CF = 0.8 - 0.1 = 0.7** (High confidence in flu).
- The AI checks **Cough**: **MB = 0.6**, MD = 0.2
    - **CF = 0.6 - 0.2 = 0.4** (Moderate confidence).

The system combines these factors and determines **flu is likely** but not 100% certain.

Certainty Factor helps AI make better, flexible decisions even when information is incomplete.

Rule-Based Systems in AI

1. Definition

- A **rule-based system** is an AI system that uses **predefined rules** to make decisions or solve problems.
- It models **human expertise** using **if-then** logic statements.
- These systems were among the earliest AI approaches and are still used in specific applications.

2. Rule Representation

- Rules are written in the form:
  IF condition THEN action
- These rules are derived from **domain knowledge or expert insights**.

Example:

- **IF** temperature > 100 **THEN** activate_cooling_system.

3. Inference Engine

- The component that applies rules to **available data** to derive conclusions.
- Uses two main reasoning methods:
  - **Forward Chaining (Data-Driven)**:
    - Starts with known **facts** and applies rules to infer **new facts**.
  - **Backward Chaining (Goal-Driven)**:
    - Starts with a **goal** and works **backward** to check if known facts support it.

4. Working Memory

- Stores **current facts or data** related to the problem.
- The **inference engine** uses this memory to apply relevant rules.

Applications of Rule-Based Systems

A. Expert Systems

- Used in **medical diagnosis**, troubleshooting, and **legal advisory systems**.
- Example: **MYCIN** (a medical expert system).

B. Configuration Systems

- Automates **software/hardware configuration** based on user needs.
- Example: **Custom PC builder** that selects parts based on user preferences.

C. Process Automation

- Monitors **system conditions** and triggers actions if thresholds are met.
- Example: **Smart monitoring systems** that send alerts if **server CPU usage exceeds 90%**.

Conclusion

- Rule-based systems are **simple, explainable, and effective** for structured problems.
- However, they struggle with **complex or evolving** problems where learning is required.

Bayesian Network

1. What is a Bayesian Network?

A **Bayesian Network (BN)** is a **graphical model** that represents relationships between different variables using **probability theory**. It helps in reasoning under **uncertainty**, which is common in AI, medical diagnosis, and decision-making systems.

A Bayesian Network consists of:

- **Nodes (Variables):** Represent different factors or events.
- **Edges (Arrows):** Show dependencies between variables.
- **Conditional Probability Tables (CPTs):** Store probability values to define relationships.

Example:
A **medical diagnosis system** may use a Bayesian Network to predict **flu** based on symptoms like fever and cough.

Different Modes of Representation

A. Graphical Representation

- The **Bayesian Network** is shown as a **graph** where:
  - Each **node** represents a variable (e.g., Fever, Flu, Cough).
  - Each **arrow** represents a conditional dependence (e.g., Flu → Fever).
- The structure **reduces complexity** by avoiding unnecessary dependencies.

Example Graph Structure:

Flu → Fever

Flu → Cough

If someone has **Flu**, there is a high probability they will have **Fever** and **Cough**.

B. Directed Acyclic Graph (DAG)

- Bayesian Networks must be a **Directed Acyclic Graph (DAG)**.
- **"Directed"** means each arrow points in one direction (cause → effect).
- **"Acyclic"** means there are no loops (i.e., a variable cannot be its own ancestor).

Example DAG:

Smoking → Lung Cancer → Cough

Smoking **causes** lung cancer, which **leads to** coughing.

C. Variable Types in Bayesian Networks

- **Discrete Variables**: Fixed values (e.g., "Yes/No", "High/Low").
- **Continuous Variables**: Numeric values (e.g., temperature, blood pressure).

Applications of Bayesian Networks

A. Medical Diagnosis

- AI-based **disease prediction** using symptoms.
- **Example:** If a patient has **chest pain and sweating**, AI estimates the probability of a **heart attack**.

B. Spam Email Detection

- AI analyzes words in an email and predicts if it's spam.
- **Example:** If an email contains "free money" and "win now," AI estimates its probability of being spam.

C. Self-Driving Cars

- AI predicts **traffic conditions** and driver behavior.
- **Example:** If a pedestrian is near a crossing, AI estimates whether they will **cross** the road.

D. Fraud Detection

- AI identifies **unusual transactions** in banking.
- **Example:** If a credit card is used in **two countries within an hour**, AI detects potential fraud.

Conclusion:

- **Bayesian Networks** are powerful **AI tools** for decision-making under uncertainty.
- They use **graphs, probabilities, and inference** to model real-world scenarios.
- **Applications** include **medical AI, cybersecurity, autonomous systems, and risk analysis**.

Bayesian Networks **help AI make smart, data-driven decisions** in uncertain environments!

Dempster-Shafer Theory (DST)

Introduction to Dempster-Shafer Theory (DST)

- **Dempster-Shafer Theory (DST)** is a mathematical framework for **reasoning under uncertainty**.
- **Dempster-Shafer Theory (DST)** is also known as **Theory of evidence or Theory of Belief Functions,** is a mathmatical framework for reasoning under uncertainty in AI and decision theory.
- It is developed by **Arthur Dempster** and **Glenn Shafer** in 1960s as an alternative to classical probability theory, especially when dealing with uncertain or incomplete information**.**
- It is an **alternative to probability theory**, allowing AI systems to make decisions even when data is **incomplete or imprecise**.

- DST assigns **belief** to sets of possibilities instead of individual events, making it more **flexible** than traditional probability.

Dempster-Shafer Rule of Combination

- DST combines **evidence from multiple sources** to calculate the overall belief in an event.
- It helps in situations where different pieces of evidence provide **partial knowledge**.
- The **Dempster's Rule of Combination** updates belief functions using multiple sources of information.

Applications of Dempster-Shafer Theory

A. Medical Diagnosis

- AI combines symptoms from different tests to estimate the probability of a disease.
- Example: If **Test 1** suggests 60% flu and **Test 2** suggests 50%, DST combines them to improve diagnosis accuracy.

B. Autonomous Vehicles

- Self-driving cars use DST to **combine data** from sensors (camera, radar, lidar) to detect obstacles.
- If **camera sees a pedestrian** (80% confidence) and **radar detects an object** (70% confidence), DST merges this data for better decisions.

C. Cybersecurity & Fraud Detection

- AI systems analyze multiple **security alerts** to detect cyber threats.
- Example: If **Network Activity** suggests a 50% attack chance and **User Behavior** suggests 40%, DST helps decide if a security breach is likely.

D. Decision Making in AI & Robotics

- Used in expert systems where AI **doesn't have complete knowledge**.
- Example: AI assistants use DST to **merge uncertain information** (e.g., weather prediction combining multiple reports).

Real-Life Example for Better Understanding

Scenario: Diagnosing COVID-19 Based on Symptoms & Test Reports

1. **Doctor 1 (Symptom Analysis)**:
    - Believes 70% that a patient has **COVID-19** based on fever, cough.
2. **Doctor 2 (Test Results)**:
    - Believes 60% in COVID-19 after analyzing a rapid test result.
3. **Using Dempster-Shafer Rule**:
    - DST combines **both opinions** to refine the belief and make a more **accurate diagnosis**.
    - Final belief may be **higher than either doctor's belief alone**, increasing confidence in the decision.

Conclusion

- **DST is powerful in uncertain situations**, where traditional probability fails.
- It helps **AI, robotics, medicine, and security systems** combine uncertain data to make **better, reliable** decisions.
- Unlike probability theory, DST allows handling of **conflicting evidence** without requiring complete data.

DST is widely used in AI for decision-making under uncertainty!

Important Questions

1. Explain about Non-monotonic Reasoning and its implementation issues.
2. Explain Bayes' Theorem and its application in updating probabilities.
3. Explain about the Dempster Shafer theory.
4. Write about Monotonic versus Non monotonic Reasoning.
5. Explain the concept of the certainty factor in AI and its role in decision-making.
6. Explain about Bayesian theorem and Bayesian Networks.