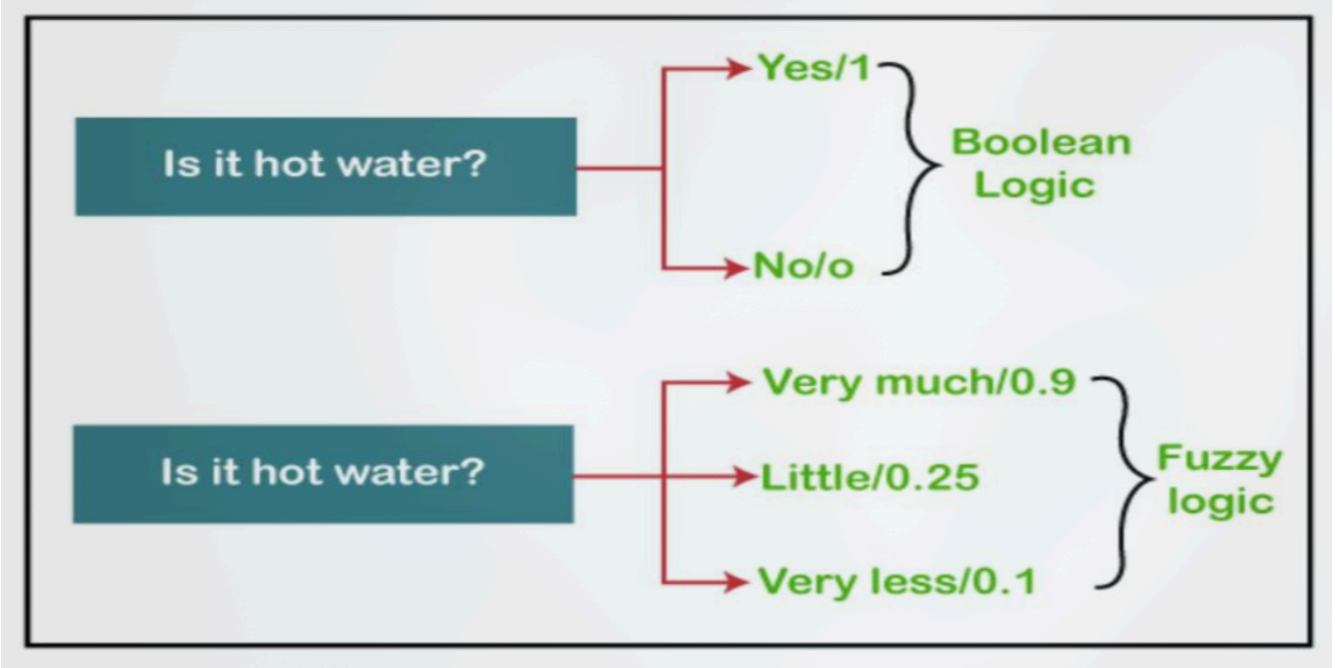
AI UNIT - 4 CONCISE

UNIT 4: Clustering, Uncertainty, Expert Systems, Fuzzy Logic & ML Intro

Part 1: Fuzzy Logic `[PYQ]`

1. Introduction to Fuzzy Logic `[PYQ]`

- **"Fuzzy" Meaning:** Things that are not clear, vague, or imprecise.
- **Purpose:** Handles situations where decisions/statements are not strictly true or false.
- Provides many values **between true (1) and false (0)**, offering flexibility.
- Allows representation of human-like linguistic uncertainty. `[FIG]` (Boolean Yes/1, No/0 vs. Fuzzy Very much/0.9, Little/0.25, Very less/0.1)



- **Introduced by:** Lotfi Zadeh in **1965** based on **Fuzzy Set Theory**.
- **Comparison with Boolean Logic:**
  - Boolean: Only two possibilities (0 or 1 - absolute false/true).
  - Fuzzy: Multiple possibilities between 0 and 1 (partially false/true).
- **Implementation:** Can be in hardware or software (micro-controllers, workstations, network systems).

1.1. Characteristics of Fuzzy Logic `[PYQ]`

- Flexible and easy to understand/implement.
- Helps minimize human-created logic complexity.

- Best for problems suitable for approximate or uncertain reasoning.
- Often offers two values (denoting possible solutions for a problem/statement).
- Allows users to build non-linear functions of arbitrary complexity.
- Everything is a **matter of degree**.
- Any logical system can be "fuzzified."
- Based on **natural language processing**.
- Used by quantitative analysts for algorithm execution improvement.
- Allows users to integrate with programming.

1.2. Fuzzy Thinking & Fuzzy Sets

- **Fuzzy Thinking:** Deals with concepts that are not precise or crisp, but rather "fuzzy" or vague (e.g., "tall man," "high profit"). Classical set theory (elements either belong or don't) struggles with this. `[PYQ]`
- **Fuzzy Set Theory:** Formalized by Lotfi Zadeh (1965) to generalize classical set theory.
- **Crisp Set:** `[FIG]`
  - Elements have either full (1) or no (0) membership.
  - Defined by a **characteristic function** $\chi A: X \rightarrow \{0, 1\}$.
- **Fuzzy Set:** `[PYQ]` `[FIG]`
  - Elements have a **degree of membership** between 0 and 1.
  - Defined by a **membership function (MF)** $\mu A: X \rightarrow [0, 1]$. `[PYQ]`
    - $\mu A(x) = 1$: x is fully in A.
    - $\mu A(x) = 0$: x is not in A.
    - $0 < \mu A(x) < 1$: x is partly in A.
  - Formal definition: $A = \{(x, \mu A(x)) \mid x \in X\}$.
  - Totally characterized by its MF.

1.4. Fuzzy Terminology `[PYQ]`

- **Universe of Discourse (U):** The range of all possible values for an input to the fuzzy system.
- **Fuzzy Set:** `[PYQ]` A set whose elements have degrees of membership (between 0 and 1) in that set.
  - Empowers members to have different grades of membership.
- **Membership Function (μ):** (Defined above) Forms the basis of a fuzzy set.
- **Support of a fuzzy set (S_f):** `[PYQ]` The set of all elements in the universal set U that have a non-zero membership value in fuzzy set `f`.
  - Example: `S_adult = {21,30,35,40,45,60,70}` for adult ages.

- **Depiction of a fuzzy set:** Often represented as a sum of (membership_value / element_value) terms, e.g., `Old = 0.1/21 + 0.3/30 + ... + 1/70`. (Note: slash is not algebraic division).

2. Fuzzy Membership Functions (MF) `[PYQ]` `[FIG]`

- **Definition:** Defines the fuzzy set; provides a measure of similarity of an element to the fuzzy set.
- **Determination:**
  - Chosen by user arbitrarily (based on experience).
  - Designed using machine learning methods (ANNs, GAs).
- **Shapes of MFs:** `[PYQ]` `[FIG]`
  - **Triangular MF:** Defined by 3 points (a, b, c) forming a triangle.
  - **Trapezoidal MF:** Defined by 4 points (α, β, γ, δ) forming a trapezoid.
  - **Gaussian MF:** Bell-shaped curve defined by center (c), width (s), and fuzzification factor (m). `μA(x,c,s,m) = exp[-0.5 * (|x-c|/s)^m]` (Note: formula in notes has 'm' inside exponent, commonly it's 2 for Gaussian, or general form is slightly different).
  - Other shapes: Piecewise-linear, bell-shaped, etc.

3. Crisp Sets vs. Fuzzy Sets `[PYQ]` `[FIG]`

- **Crisp Sets:** Mutually exclusive membership (either in or out).
- **Fuzzy Sets:** Define a **degree of membership**, allowing an element to belong to multiple fuzzy sets simultaneously with different degrees.
- **Example: "Tall Men"** `[FIG]`
  - Crisp: A fixed height (e.g., 180cm) is the cutoff.
  - Fuzzy: Men have a degree of "tallness" (e.g., 175cm might be 0.6 tall, 185cm might be 0.9 tall).

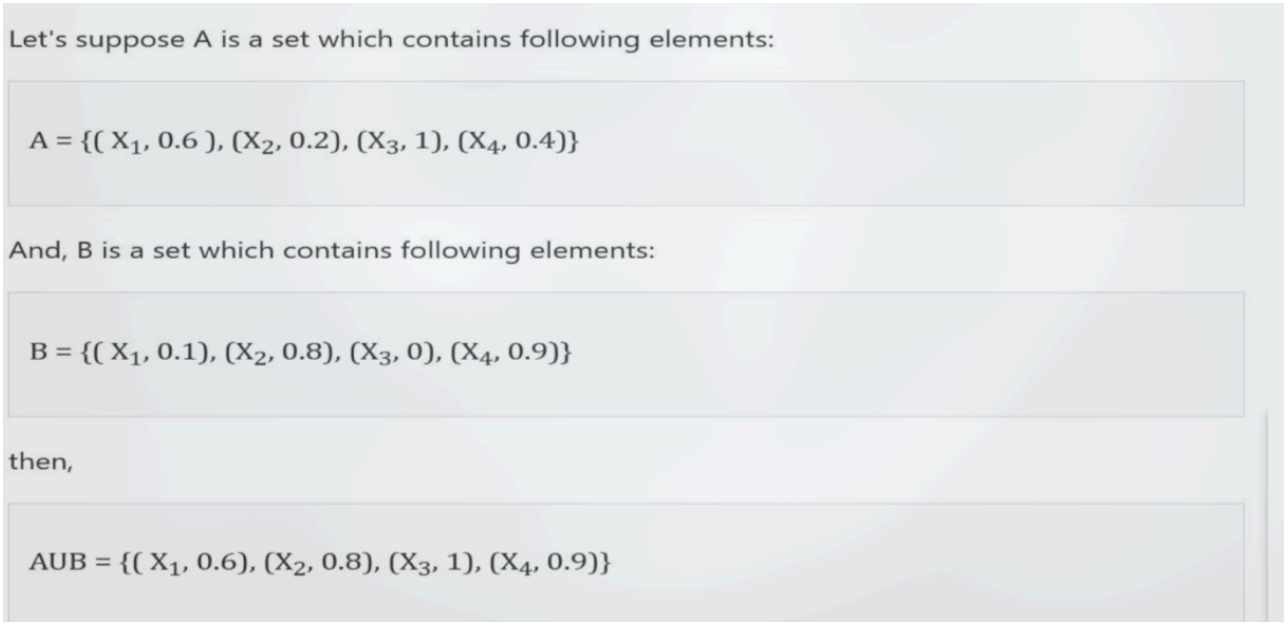
3.1. Difference Between Fuzzy and Classical (Crisp) Set Theory `[PYQ]`

- **Boundaries:**
  - Classical: Sharp boundaries (element is either in or out). Defined by exact boundaries 0 and 1.
  - Fuzzy: Un-sharp boundaries (degrees of membership). Defined by ambiguous boundaries.
- **Uncertainty:**
  - Classical: No uncertainty about boundary location.
  - Fuzzy: Always exists uncertainty about boundary location.
- **Usage:**
  - Classical: Widely used in digital system design.
  - Fuzzy: Mainly used for fuzzy controllers.

4. Fuzzy Set Operations `[PYQ]` `[FIG]` (for Union, Intersection, Complement)

- Defined in terms of membership functions (Standard Fuzzy Operations - Min/Max):

- **Union (A ∪ B):** μA∪B(x) = max(μA(x), μB(x)) (Corresponds to OR) `[PYQ]`

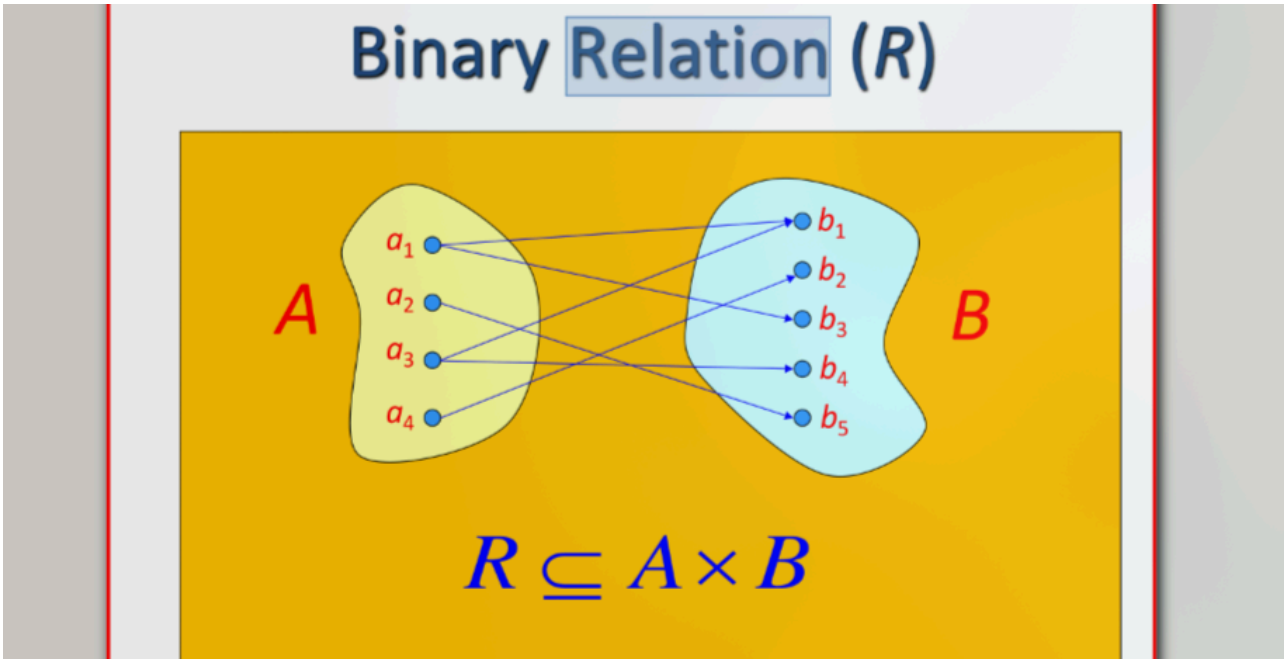


  - **Intersection (A ∩ B):** μA∩B(x) = min(μA(x), μB(x)) (Corresponds to AND) `[PYQ]`
  - **Complement (Ā or not A):** μĀ(x) = 1 - μA(x) `[PYQ]`
- **Fuzzy Set Inclusion (A ⊂ B):** If and only if ∀x, μA(x) ≤ μB(x).
- **Fuzzy Set Equality (A = B):** If and only if ∀x, μA(x) = μB(x).
- **Properties:** Commutativity, associativity, idempotency, distributivity hold. De Morgan's laws hold.
- **Laws that FAIL in Min-Max Fuzzy Logic:** `[PYQ]`
  - **Law of Excluded Middle:** A ∪ Ā ≠ X (Universal Set), because max{μA(x), 1-μA(x)} is not always 1 (it is, unless μA(x)=0.5). *Correction: max is always >= 0.5. The issue is it's not necessarily the full universe in a multi-valued sense. More accurately for fuzzy: The crisp law A ∪ Ā = X doesn't hold in the same way because an element can be "somewhat A" and "somewhat not A".*
  - **Law of Contradiction:** A ∩ Ā ≠ Ø (Empty Set), because min{μA(x), 1-μA(x)} is not always 0 (it is, unless μA(x)=0.5). *Correction: min is always <= 0.5. The issue is an element can have non-zero membership in both.*

5. Fuzzy Relations `[PYQ]` `[FIG]`

- **Definition:** A fuzzy relation R on X × Y is a fuzzy subset of X × Y, characterized by a 2D membership function μR(x,y).
- **Representations:**

- List of ordered pairs with membership grades: {((x,y), μR(x,y))}.



- **Matrix Representation:** Rows for X, columns for Y, cell (i,j) contains μR(xi, yj). `[FIG]`

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ r_{n1} & r_{n2} & \cdots & r_{nm} \end{bmatrix}$$

- **Operations on Fuzzy Relations:** Similar to fuzzy sets (Union, Intersection, Complement, Containment).
- **Composition of Fuzzy Relations (e.g., R on X×Y, S on Y×Z → T on X×Z):** `[PYQ]` `[FIG]`
  - **Max-Min Composition (T = R ∘ S):** `[PYQ]`
    - μT(x,z) = max_y {min(μR(x,y), μS(y,z))}
    - Essentially, for each (x,z) pair, find the "strongest path" through an intermediate y, where path strength is the minimum of the two relation memberships, and then take the maximum of these path strengths.
  - **Max-Product Composition:**
    - μT(x,z) = max_y {μR(x,y) * μS(y,z)}
    - Used when max-min is not mathematically tractable.

6. Linguistic Variables and Hedges `[PYQ]` `[FIG]`

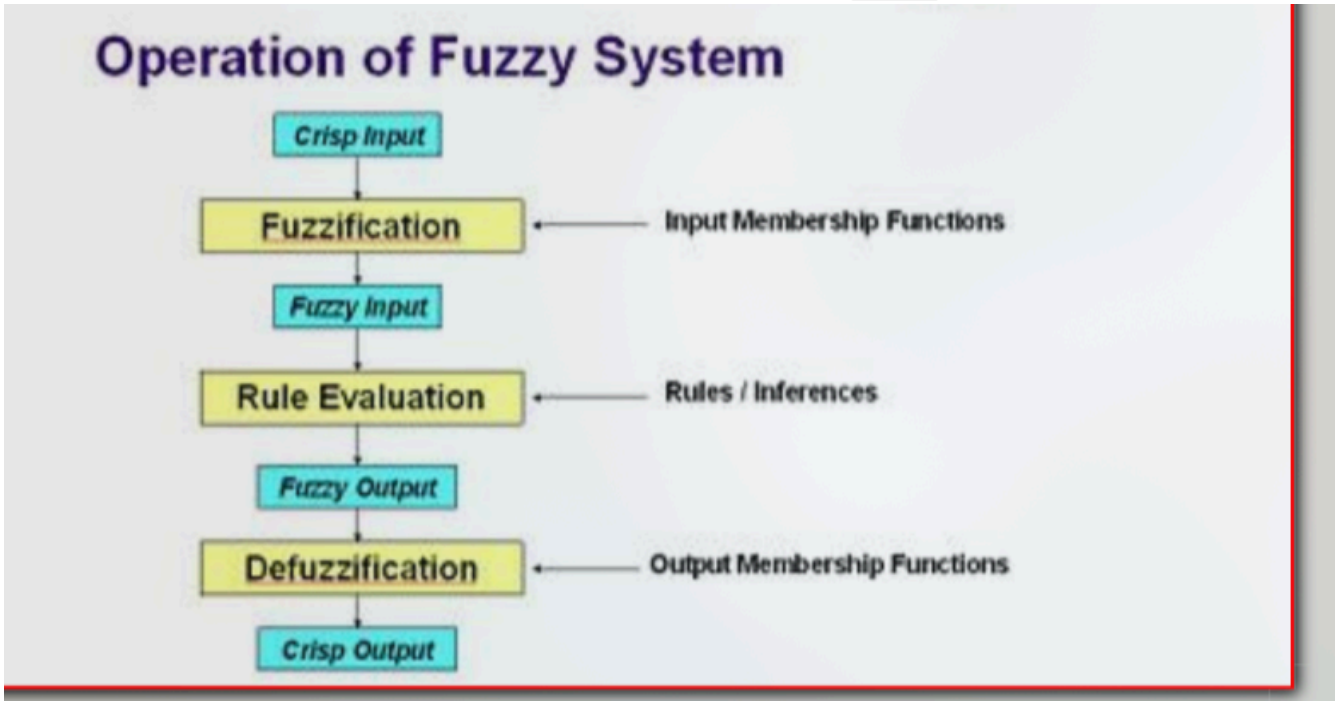
- **Linguistic Variable:** `[PYQ]` A variable whose values are words or sentences in a natural or artificial language (i.e., fuzzy sets).

- E.g., "Temperature" is a linguistic variable; its values can be "cold," "warm," "hot" (which are fuzzy sets).
- "Ram is tall": Linguistic variable "Ram's height" takes linguistic value "tall".

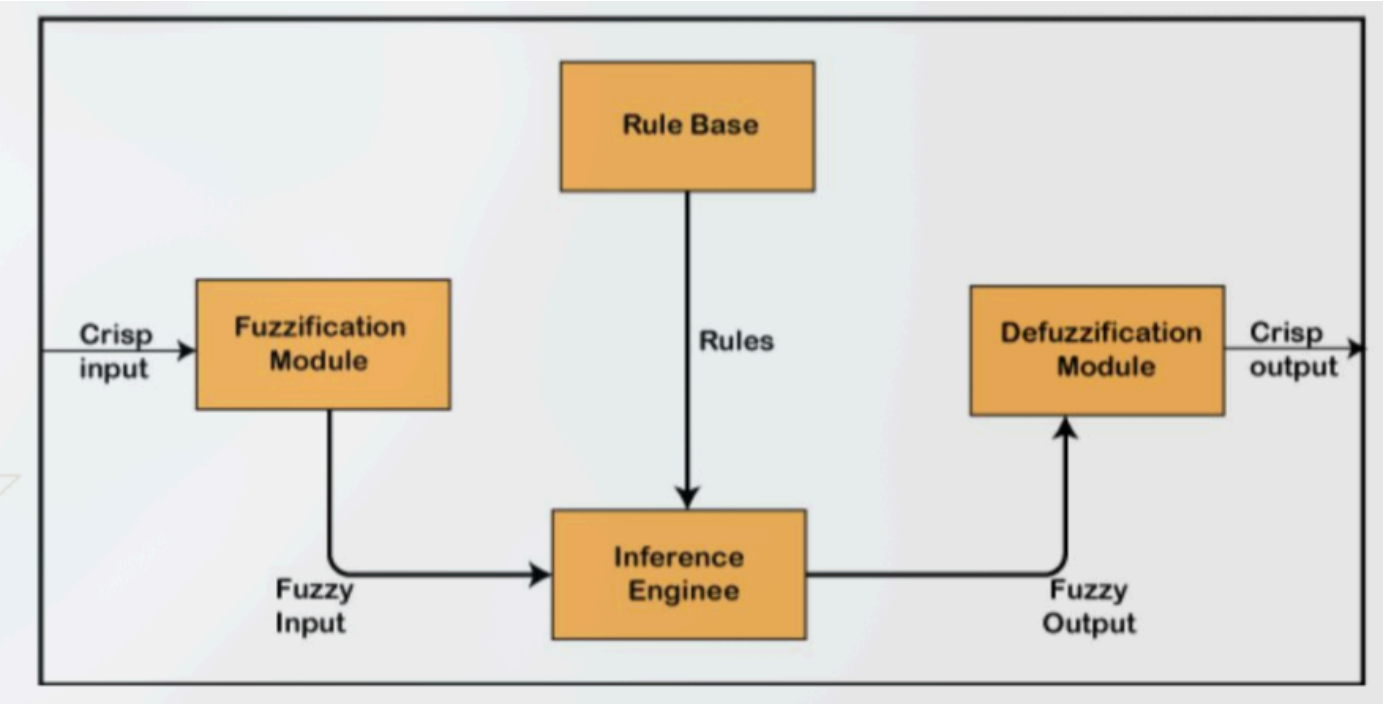
- **Hedges (Fuzzy Set Qualifiers):** `[PYQ]` `[FIG]`
  - Adverbs that modify the shape of fuzzy sets (linguistic values).
  - Examples: *very, somewhat, quite, more or less, slightly*.
  - Mathematical operations on MFs:
    - `very A`: μvery A(x) = (μA(x))^2 (concentration)
    - `slightly A` / `somewhat A`: μslightly A(x) = sqrt(μA(x)) (dilation)
    - `A little`: (μA(x))^1.3
    - `Extremely A`: (μA(x))^3

7. Fuzzification and Defuzzification `[PYQ]` `[FIG]`

- **Fuzzification:** `[PYQ]` `[FIG]`
  - Transforming crisp (numerical) inputs into fuzzy sets (degrees of membership to linguistic terms).
  - Maps input values to membership functions.
- **Defuzzification:** `[PYQ]` `[FIG]`
  - Mapping fuzzy output sets (from inference process) back to a single crisp (numerical) output value.
  - Needed for real-world actuators.
  - Methods: Centroid (Center of Gravity), Max-Membership, Weighted Average, etc.
- **Operation of Fuzzy System:** Crisp Input → Fuzzification → Fuzzy Input → Rule Evaluation (Inference) → Fuzzy Output → Defuzzification → Crisp Output. `[FIG]`



7.1. Architecture of a Fuzzy Logic System (FLS) `[PYQ]` `[FIG]`



- **Four Main Components:**
  - **1. Rule Base (Knowledge Base):** `[PYQ]`
    - Stores a set of rules (often IF-THEN conditions) given by experts.
    - Used for controlling decision-making systems.
    - Modern fuzzy theory offers methods for designing/tuning fuzzy controllers, reducing fuzzy set rules.
  - **2. Fuzzification Module:** `[PYQ]`
    - Transforms crisp system inputs (numerical values measured by sensors) into fuzzy sets (linguistic variables with degrees of membership).
    - Divides input signals into fuzzy states (e.g., Large Positive (LP), Medium Positive (MP), Small (S), Medium Negative (MN), Large Negative (LN)).
  - **3. Inference Engine (Decision-Making Unit):** `[PYQ]`
    - Main component; processes fuzzy input and rules.
    - Finds the matching degree between current fuzzy input and rules.
    - Determines which rule(s) to fire based on input.
    - Combines fired rules to develop control actions.
  - **4. Defuzzification Module:** `[PYQ]`
    - Converts the fuzzy output sets (from inference engine) back into a single crisp numerical value that can be used as a control action.
    - Example methods: Centre of Gravity (Centroid), Maxima methods. `[FIG]` (Defuzzification graphs showing combined output region and centroid)

8. Fuzzy Rule-Based Systems / Fuzzy Classifiers `[PYQ]` `[FIG]`

- Use fuzzy IF-THEN rules.
- **Components of Fuzzy Logic System Architecture:** `[PYQ]` `[FIG]`
  - **Rule Base:** Set of IF-THEN rules provided by experts or learned.
  - **Fuzzification Module:** Converts crisp inputs to fuzzy sets.
  - **Inference Engine:** Applies fuzzy rules to fuzzified inputs to produce fuzzy outputs. Determines matching degree of fuzzy input with rule antecedents. Combines fired rules.
  - **Defuzzification Module:** Converts fuzzy outputs to crisp outputs.
- **Fuzzy IF-THEN Rules:**
  - Antecedent (IF part): `IF x is A AND y is B` (A, B are fuzzy sets).
  - Consequent (THEN part): `THEN z is C` (C is a fuzzy set).
- **Inference Process (Example for AND antecedent):**
  - **Firing Level (αi) of a rule:** min{μA(input_x), μB(input_y)} (for AND). max for OR.
  - If multiple rules have same consequence class, combine firing strengths (usually max/OR).
  - Overall classifier output often selected by choosing class with max combined strength.

9. Applications of Fuzzy Logic `[PYQ]`

- **Businesses:** Decision-making support systems.
- **Automotive systems:** Controlling traffic/speed, transmission efficiency, shift scheduling.
- **Defence:** Target recognition (underwater, thermal infrared).
- **Pattern Recognition and Classification:** Fuzzy logic-based recognition, handwriting recognition, fuzzy image searching.
- **Securities.**
- **Microwave oven:** Setting power/cooking strategy.
- **Modern control systems:** Expert systems.
- **Finance:** Predicting stock market, managing funds.
- **Controlling brakes.**
- **Industries of chemicals:** Controlling pH, chemical distillation.
- **Industries of manufacturing:** Optimization of milk/cheese production.
- **Vacuum cleaners, washing machines** (timings).
- **Heaters, air conditioners, humidifiers.**

10. Differences between Fuzzy Logic and Probability `[PYQ]`

- **Fuzzy Logic:** Deals with **imprecision** or **vagueness** of facts/statements. Membership in vaguely defined sets.
  - E.g., "The water is *hot*." (Degree of hotness).

- **Probability:** Deals with **uncertainty** or **chances** of an event occurring (likelihood of a precise event).
    - E.g., "There is an 80% chance it will rain tomorrow." (Event is precise: rain or no rain).
- Fuzzy sets are based on vague definitions, not randomness.
- Lotfi Zadeh: Fuzzy logic is different in character from probability, not a replacement. Possibility theory is fuzzy alternative to probability.

11. Advantages of Fuzzy Logic `[PYQ]`

- Methodology similar to human reasoning.
- Easily understandable structure.
- Does not need large memory (algorithms can be described with fewer data).
- Widely used, provides effective solutions for high-complexity problems.
- Simple (based on set theory of mathematics).
- Allows control of machines and consumer products.
- Shorter development time compared to conventional methods.
- Flexibility allows easy addition/deletion of rules in FLS.

12. Disadvantages of Fuzzy Logic `[PYQ]`

- Run time can be slow, takes time to produce outputs.
- Users understand it easily only if systems are simple.
- Possibilities produced are not always accurate.
- Multiple ways to solve a statement can lead to ambiguity.
- Not suitable for problems requiring high accuracy.
- Systems need extensive testing for verification and validation.

13. Case Study: Fuzzy Room Cooler `[FIG]`

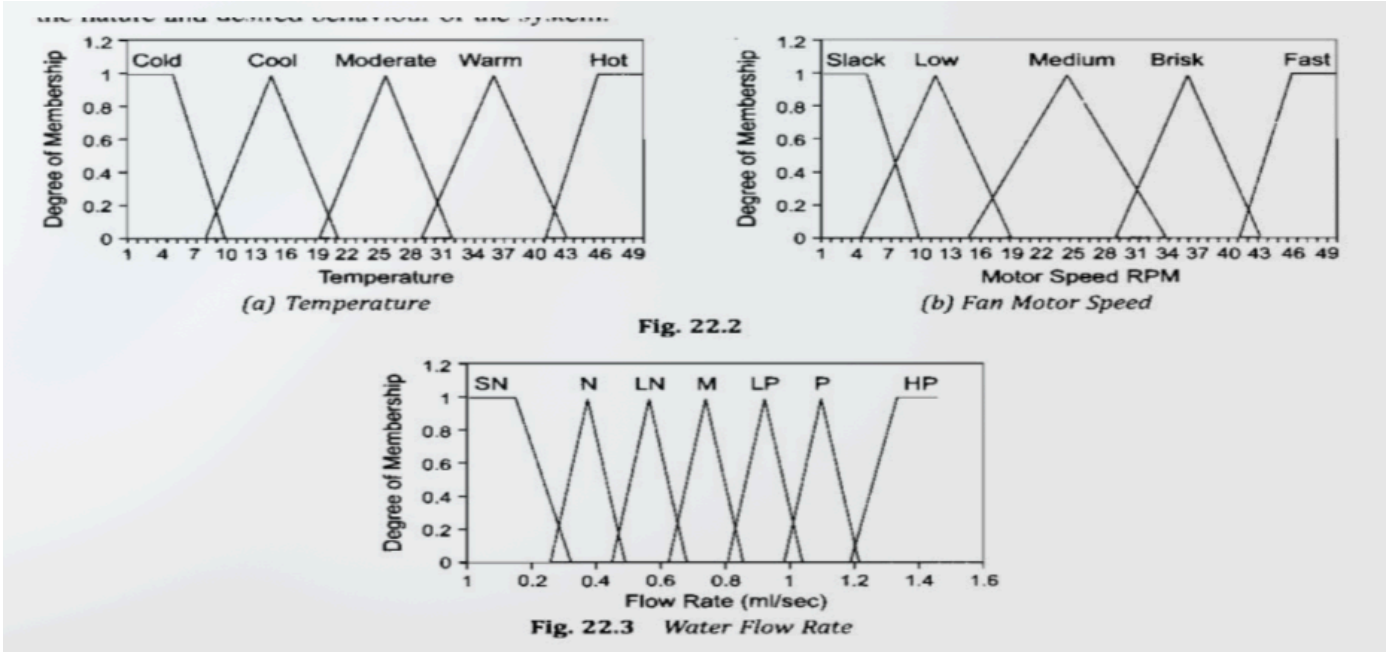


Fig. 22.2

Fig. 22.3 *Water Flow Rate*

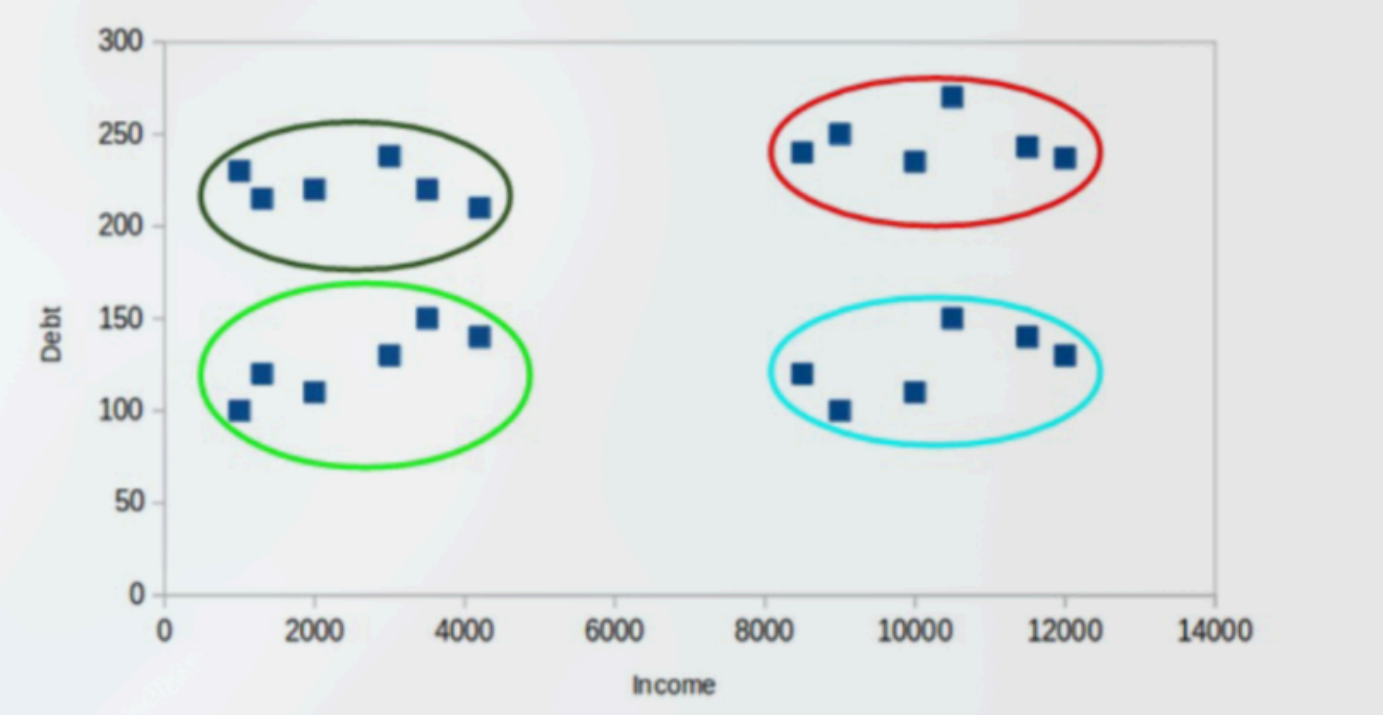
- **Fuzzy Regions/Terms:**
  - **Input Parameters:** Temperature, (Fan Motor Speed/Pressure).
  - **Fuzzy Terms for Temperature:** Cold, Cool, Moderate, Warm, Hot.
  - **Fuzzy Terms for Fan Motor Speed:** Slack, Low, Medium, Brisk, Fast.
  - **Output (Flow-rate):** Strong-Negative, Negative, Low-Negative, Low-Positive, High-Positive.
- **Fuzzy Profiles:** Membership function graphs for each fuzzy term. `[FIG]` (Temp, Motor Speed, Flow Rate graphs)
- **Fuzzy Rules:** Form triggers for the fuzzy engine (linguistic IF-THEN rules). `[FIG]`
  - E.g., R1: IF temperature is HOT AND fan motor speed is SLACK THEN flow-rate is HIGH-POSITIVE.
- **Fuzzification:** Mapping crisp sensor inputs (e.g., temp=42°C, speed=31 rpm) to membership degrees in respective fuzzy regions. `[FIG]`
- **Inference:** (Implicit in combining rules) Rules fire based on fuzzified inputs. If multiple rules apply, their outputs are combined.
- **Defuzzification:** Converting the combined fuzzy output for flow-rate into a single crisp value for the pump. `[FIG]` (Showing how outputs of multiple rules like R3, R4, R5, R6 are combined for a final crisp output using centroid method).
  - Common methods: Centre of Gravity, Composite Maxima.
- **NeuroFuzzy Room Cooler:** `[FIG]` (Diagram showing a neural network structure implementing the fuzzy logic system for the room cooler).

Part 2: K-Means Clustering `[PYQ]`

1. What is K-Means Clustering? `[PYQ]` `[FIG]`

- Popular **unsupervised machine learning algorithm**.
- Partitions a dataset into a pre-defined number (K) of clusters.
- **Goal:** Group similar data points together; discover underlying patterns/structures.
- **Centroid-based / Distance-based:** Calculates distances to assign points to clusters. Each cluster associated with a **centroid**.
- **Property of Clusters:**
  - Points within a cluster should be similar (minimize intra-cluster distance).
  - Points in different clusters should be dissimilar (maximize inter-cluster distance).
- **Main Objective:** Minimize the sum of squared distances between data points and their respective cluster centroids (WCSS - Within-Cluster Sum of Squares).

2. How K-Means Clustering Works? `[PYQ]` `[FIG]`



1. **Initialization:** Randomly select K points from the dataset as initial cluster centroids.
2. **Assignment Step:** For each data point, calculate its distance (e.g., Euclidean) to all K centroids. Assign the point to the cluster whose centroid is closest.
3. **Update Centroids Step:** Recalculate the centroid of each cluster by taking the mean of all data points assigned to that cluster.
4. **Repeat:** Repeat steps 2 and 3 until convergence (centroids no longer change significantly, or max iterations reached).
5. **Final Result:** Algorithm outputs final cluster centroids and data point assignments.

3. What is Clustering (General)? `[PYQ]`

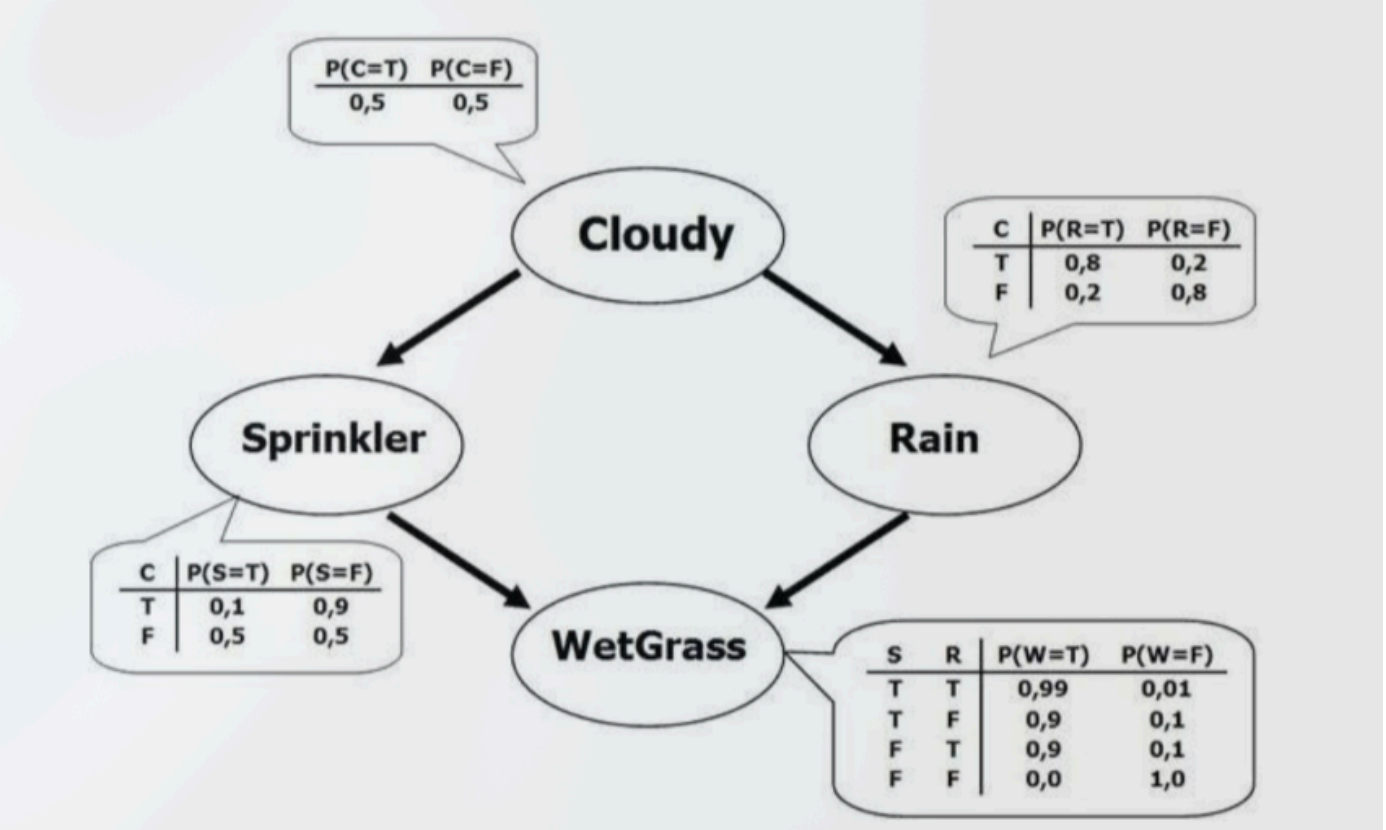
- Technique in data mining & ML that groups similar objects into clusters.
- **Unsupervised Learning Problem:** No target variable to predict. Aims to find inherent structure in data.
- **Example:** Bank segmenting customers based on income for targeted credit card offers. `[FIG]`

4. Properties of K-Means Clustering (Reiteration of Cluster Properties)

- **First Property:** Data points in a cluster should be similar to each other (high cohesiveness).
- **Second Property:** Data points from different clusters should be as different as possible (high separation).

Part 3: Bayesian Networks `[PYQ]`

1. What are Bayesian Networks (BNs)? `[PYQ]` `[FIG]`



- Type of **Probabilistic Graphical Model (PGM)**.
- Represents a set of random variables and their conditional dependencies using a **Directed Acyclic Graph (DAG)**.
- Uses Bayesian inference for probability computations.
- Nodes: Random variables (discrete or continuous).
- Edges (Arcs): Represent conditional dependencies (often interpreted as causal relationships). If edge (A,B) exists, A is a parent of B.
- Also called: Bayes network, belief network, decision network, Bayesian model.

2. Key Concepts of Bayesian Networks

- **Conditional Probability Table (CPT):** Each node has a CPT specifying P(Node | Parents(Node)). For root nodes (no parents), CPT is just P(Node). `[PYQ]` `[FIG]`
- **Local Markov Property:** `[PYQ]` A node is conditionally independent of its non-descendants given its parents.
  - This allows simplification of the Joint Probability Distribution (JPD).
- **Joint Probability Distribution (JPD):** `[PYQ]`
  - BNs provide a compact, factorized representation of the JPD.
  - P(X1, ..., Xn) = Π P(Xi | Parents(Xi)) for all nodes i.
- **Conditional Probability:** P(A|B) = P(A ∩ B) / P(B).

3. Bayesian Network Example (Student Marks) `[PYQ]` `[FIG]`

- Variables: Exam Level (e), IQ Level (i), Aptitude Score (s), Marks (m), Admission (a).

- Dependencies: e,i → m; i → s; m → a. (e and i are parents of m, etc.)
- JPD: P(a,m,i,e,s) = P(a|m) * P(m|i,e) * P(s|i) * P(i) * P(e).
- Given CPTs for each variable, can compute probability of specific events (e.g., P(admission=yes, marks=pass, IQ=low, exam=difficult, aptitude=low)).

4. Bayesian Network Example (Burglary Alarm) `[PYQ]` `[FIG]`

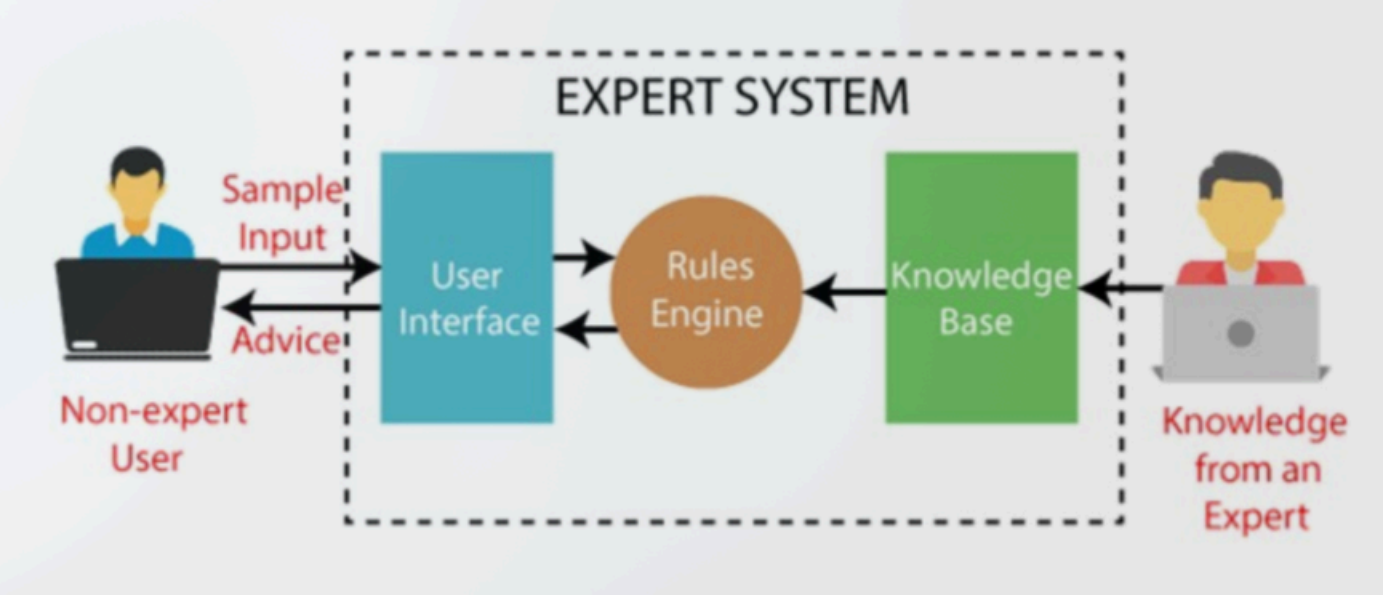
- Variables: Burglary (B), Earthquake (E), Alarm (A), David Calls (D), Sophia Calls (S).
- Dependencies: B,E → A; A → D; A → S.
- JPD: P(D,S,A,B,E) = P(D|A) * P(S|A) * P(A|B,E) * P(B) * P(E) (assuming B and E are independent initially, or P(B|E)P(E)).
- Used to calculate probability of complex events, e.g., P(Alarm | DavidCalls, ¬Burglary, ¬Earthquake).

5. Semantics of Bayesian Network

- **Two ways to understand:**
  1. As a representation of the **Joint Probability Distribution**. (Useful for construction).
  2. As an encoding of a collection of **conditional independence statements**. (Useful for designing inference procedures).

Part 4: Expert Systems `[PYQ]`

1. What is an Expert System (ES)? `[PYQ]`



- Computer program designed to solve complex problems and provide decision-making ability like a human expert within a specific domain.
- Extracts knowledge from its Knowledge Base (KB) using reasoning and inference rules.
- First ES developed in 1970s (e.g., DENDRAL, MYCIN).
- Uses both facts and **heuristics**.
- Designed for specific domains (medicine, science, finance).

- *Note: ES assists human experts, does not replace them.*

2. Examples of Expert Systems `[PYQ]`

- **DENDRAL:** Chemical analysis (organic chemistry).
- **MYCIN:** `[PYQ]` Medical diagnosis (bacterial infections, antibiotic recommendation). Used backward chaining.
- **PXDES:** Determines type/level of lung cancer from images.
- **CaDeT:** Diagnostic support for early cancer detection.

3. Characteristics of Expert Systems

- **High Performance:** Solves complex domain problems with high efficiency/accuracy.
- **Understandable:** Responds in a way easily understandable by user (human language input/output).
- **Reliable:** Generates efficient and accurate output.
- **Highly responsive:** Provides results for complex queries quickly.

4. Components of an Expert System `[PYQ]` `[FIG]`

- **A. User Interface:** `[PYQ]`
  - Allows non-expert user to interact with ES, input queries, receive responses in a readable format.
- **B. Inference Engine (Rule Engine / "Brain"):** `[PYQ]`
  - Main processing unit. Applies inference rules to the KB to derive conclusions or deduce new info.
  - Extracts knowledge from KB.
  - **Types:**
    - *Deterministic Inference Engine:* Conclusions assumed true (based on facts/rules).
    - *Probabilistic Inference Engine:* Contains uncertainty in conclusions (based on probability).
  - **Modes (Chaining):** Forward Chaining, Backward Chaining. `[PYQ]`
- **C. Knowledge Base (KB):** `[PYQ]`
  - Stores knowledge acquired from domain experts.
  - Contains information and rules for a particular domain/subject.
  - **Components of KB:**
    - *Factual Knowledge:* Based on facts, accepted by knowledge engineers.
    - *Heuristic Knowledge:* Based on practice, ability to guess, evaluation, experiences.
  - *Knowledge Representation:* Formalizes knowledge (e.g., IF-ELSE rules). `[PYQ]`
  - *Knowledge Acquisition:* Process of extracting, organizing, structuring domain knowledge, specifying rules, storing in KB. `[PYQ]`

5. Development of Expert System (e.g., MYCIN)

1. Feed ES with expert knowledge (e.g., medical info for MYCIN).
2. KB is updated. Doctor tests with new problem (patient details, symptoms).
3. ES may use questionnaire for general patient info.
4. System finds solution by applying IF-THEN rules (inference engine) to facts in KB.
5. Provides response via user interface.

6. Participants in ES Development

- **Expert:** Provides specialized domain knowledge.
- **Knowledge Engineer:** Gathers knowledge from experts, codifies it into the system.
- **End-User:** Person/group (may not be expert) needing solution/advice from ES.

7. Why Expert Systems? (Capabilities & Need)

- **Capabilities:** Advising, decision-making, demonstrating devices, problem-solving, explaining problems, interpreting input, predicting results, diagnosis.
- **Need / Advantages:**
  - No memory limitations (can store vast data).
  - High Efficiency (if KB is correct).
  - Consolidated Expertise in a domain (mixes knowledge from multiple experts).
  - Not affected by emotions.
  - High security.
  - Considers all facts.
  - Regular updates improve performance.
  - Highly reproducible.
  - Usable in risky places (unsafe for humans).
- **Limitations:** `[PYQ]`
  - Wrong output if KB has wrong info.
  - Cannot produce creative output.
  - High maintenance/development costs.
  - Knowledge acquisition is difficult.
  - Domain-specific (one ES for each domain).
  - Cannot learn from itself; requires manual updates.

8. Applications of Expert Systems `[PYQ]`

- Designing and manufacturing domain (e.g., camera lenses).
- Knowledge domain (publishing relevant knowledge, e.g., tax advisor).

- Finance domain (fraud detection, loan advice).
- Diagnosis and troubleshooting of devices (medical diagnosis was an early area).
- Planning and Scheduling.

Part 5: Machine Learning (Introduction - from Aditya College Notes) `[PYQ]`

1. Introduction: AI, Machine Learning (ML), Deep Learning (DL) `[PYQ]` `[FIG]`

- **Artificial Intelligence (AI):** Branch of CS to create intelligent machines that behave, think, and decide like humans.
- **Machine Learning (ML):** `[PYQ]` Enables computers to learn automatically from past data/experience without explicit programming. Uses algorithms to build models and make predictions.
  - Introduced by Arthur Samuel (1959).
- **Deep Learning (DL):** `[PYQ]` Subset of ML based on Artificial Neural Networks (ANNs), inspired by human brain. Learns feature hierarchies. `[FIG]` (ANN diagram)

2. Types of Machine Learning Systems `[PYQ]` `[FIG]`

- **Categorized by:**
  - Human supervision: Supervised, Unsupervised, Semi-supervised, Reinforcement.
  - Incremental learning: Online vs. Batch.
  - Generalization method: Instance-based vs. Model-based.
- **A. Supervised Learning:** `[PYQ]`
  - Trained on **labeled data** (input-output pairs). Machine predicts output based on training.
  - Goal: Map input x to output y.
  - **Categories:**
    - **Classification:** Output variable is categorical (Yes/No, Spam/Ham). `[PYQ]`
      - Algorithms: Random Forest, Decision Tree, Logistic Regression, SVM.
    - **Regression:** Output variable is continuous (market trends, weather). `[PYQ]`
      - Algorithms: Linear Regression, Decision Tree, Lasso Regression.
  - Advantages: Exact idea of object classes (from labels), predicts based on prior experience.
  - Disadvantages: Can't solve complex tasks, wrong prediction if test data differs from training, high computational time to train.
- **B. Unsupervised Learning:** `[PYQ]`
  - Trained on **unlabeled data**. No explicit supervision.
  - Goal: Find hidden patterns, similarities, structures in data.

- **Categories:**
  - **Clustering:** `[PYQ]` Grouping objects into clusters (similar within, dissimilar between). E.g., K-Means, DBSCAN.
  - **Association Rule Learning:** `[PYQ]` Finding interesting relations/dependencies between variables (e.g., Apriori, Eclat).
- Advantages: Can use for complex tasks, easier to get unlabeled data.
- Disadvantages: Output less accurate, harder to work with (no mapped output).

- **C. Semi-Supervised Learning:** `[PYQ]`
  - Lies between supervised and unsupervised. Uses a combination of labeled and unlabeled data.
  - Overcomes drawbacks of both.
  - Advantages: Simple, efficient, solves drawbacks of purely supervised/unsupervised.
  - Disadvantages: Results may not be stable, not for network-level data, accuracy can be low.
- **D. Reinforcement Learning (RL):** `[PYQ]` `[FIG]`
  - Agent learns by interacting with an environment (trial and error).
  - Receives feedback as **rewards** (for good actions) or **punishments** (for bad actions).
  - Goal: Maximize cumulative reward. No labeled data initially.
  - **Categories:**
    - *Positive RL:* Adding something to increase behavior tendency.
    - *Negative RL:* Avoiding negative condition to increase behavior tendency.
  - Applications: Game theory, robotics, text mining, video games.

3. Main Challenges of Machine Learning `[PYQ]`

- Lack of Quality Data (noisy, incorrect, incomplete).
- Fault in specific applications (e.g., credit card fraud detection nuances).
- Getting Bad Recommendations.
- Talent Deficit (lack of experts).
- Implementation Complexity (integrating new ML with existing systems).
- Making Wrong Assumptions (e.g., handling missing data).
- Deficient Infrastructure (ML needs large data stirring abilities).
- Algorithms Becoming Obsolete as Data Grows.
- Absence of Skilled Resources.
- Customer Segmentation (dynamic behavior).
- Complexity of Models (many still experimental).
- Slow Results (training/inference can be time-consuming).
- Maintenance (models need updates as data/actions change).

- Concept Drift (target variable changes over time).
- Data Bias.
- High Chances of Error (biased programming/datasets).
- Lack of Explainability ("Black box" models).

4. Statistical Learning: Introduction `[PYQ]`

- Mathematical analysis of data, especially to interpret models and quantify uncertainty.
- **Two Major Goals for Modeling Data:**
  1. Accurately predict future quantity of interest.
  2. Discover unusual or interesting patterns.
- **Feature, Response:**
  - Input/feature vector **x**, predict output/response variable **y**.
  - Prediction function **g(x)** is the guess for y.
- **Regression, Classification:**
  - Regression: y is real-valued.
  - Classification: y is from a finite set (categories).
- **Loss Function:** Measures accuracy of prediction g(x) w.r.t. true response y. E.g., Squared error loss (y - g(x))^2 for regression.

5. Training and Test Loss `[PYQ]`

- **Risk ℓ(g):** Expected loss of a prediction function g. Hard to compute.
- **Empirical Risk / Training Loss ℓT(g):** Average loss over the training sample T. Unbiased estimator of risk if g is fixed.
  - `ℓT(g) = (1/n) Σ Loss(Yi, g(Xi))`
- *Optimal Prediction Function g:** Minimizer of true risk ℓ(g).
- *Learner gT:** Minimizer of training loss ℓT(g).
- *Generalization Risk / Test Loss ℓ(gT):** `[FIG]` Accuracy of learner on *new, unseen* data. Measured on a fixed test set τ.
  - `ℓτ(gT*) = E[Loss(Y, gT*(X))]` (expected loss on new data)
  - Often estimated by average loss on a test sample T'.

6. Tradeoffs in Statistical Learning `[PYQ]`

- Goal: Make generalization risk small.
- Decomposition of generalization risk (for squared error loss):
  - `E[ (Y - gT*(X))^2 ] = E[ (Y - f(X))^2 ]` **(Irreducible Error / Bayes Error)**: Variance of Y given X. Optimal possible error. Cannot be reduced by any model.

    - `+ E[ (f(X) - E[gT*(X)])^2 ]` **(Squared Bias / Approximation Error)**: How much the *average* model `E[gT*(X)]` (averaged over training sets) differs from the true function `f(X)`. Due to model being too simple.
    - `+ E[ (gT*(X) - E[gT*(X)])^2 ]` **(Variance / Estimation Error)**: How much a specific model `gT*(X)` (trained on one set T) varies around its average. Due to model being too complex and fitting noise.
- **Bias-Variance Tradeoff:** `[PYQ]` `[FIG]`
    - Simple models: High bias, low variance.
    - Complex models: Low bias, high variance.
    - Need to find a balance for optimal generalization.

7. Estimating Risk `[PYQ]`

- **1. In-Sample Risk:** `[FIG]`
    - Training loss (ℓT(g)) is not a good estimate of generalization risk due to overfitting.
- **2. Cross-Validation (CV):** `[PYQ]` `[FIG]`
    - Method to estimate generalization risk by splitting data.
    - **K-Fold CV:** Data split into K folds. Train on K-1, test on 1. Repeat K times. Average results.
    - **Leave-One-Out CV (LOOCV):** K=N.
    - Helps in model selection and hyperparameter tuning.

8. Sampling Distributions of Estimators

- Estimators (e.g., model parameters) are statistics (functions of random data), so they have sampling distributions.
- Distribution depends on sample size.
- Used to:
    - Calculate probability of estimator differing from true parameter.
    - Obtain interval estimates (confidence intervals).
    - Compare estimators (e.g., using Mean Squared Error).

9. Empirical Risk Minimization (ERM) `[PYQ]` `[FIG]`

- **Principle:** Find the model/hypothesis `h` from a hypothesis class H that minimizes the empirical risk (average loss on the training sample).
    - `h_ERM = argmin_h∈H { (1/N) Σ Loss(h(Xi), Yi) }`
- Fundamental concept in ML.
- Theory behind ERM explains VC-dimension, PAC Learning.
- *ERM is a nice idea, if used with care* (can lead to overfitting if hypothesis class is too complex or data is noisy).

Part 6: Perception and Action (General AI Concept) `[PYQ]`

- **Perception:** `[PYQ]`
  - The process by which an agent acquires information about its environment through its **sensors**.
  - Involves sensing raw data (light, sound, touch, etc.) and interpreting it into a meaningful representation of the state of the environment.
  - Examples: Computer vision (interpreting images), speech recognition (interpreting audio).
- **Action:** `[PYQ]`
  - The process by which an agent affects its environment through its **actuators**.
  - Decisions made by the agent (based on its goals, perceived state, and internal knowledge/reasoning) lead to actions.
  - Examples: Robot moving its arm, a software agent displaying information, self-driving car turning the wheel.
- **Perception-Action Cycle:** `[PYQ]` `[FIG]`
  - Fundamental loop in intelligent agents:
    1. Agent **perceives** the environment.
    2. Agent **reasons/thinks/plans** based on percepts and internal state/goals.
    3. Agent **acts** upon the environment.
    4. The environment changes (partly due to agent's action), leading to new percepts.
  - This cycle is continuous and forms the basis of agent behavior.
  - Learning often occurs by evaluating the outcomes of actions based on new perceptions.
- **Link to Learning Agents:** The "Performer" component of a learning agent is responsible for selecting actions based on its knowledge and current perception. The "Critic" evaluates these actions, and the "Learner" updates the knowledge or performer based on this evaluation, improving future perception-action mappings.