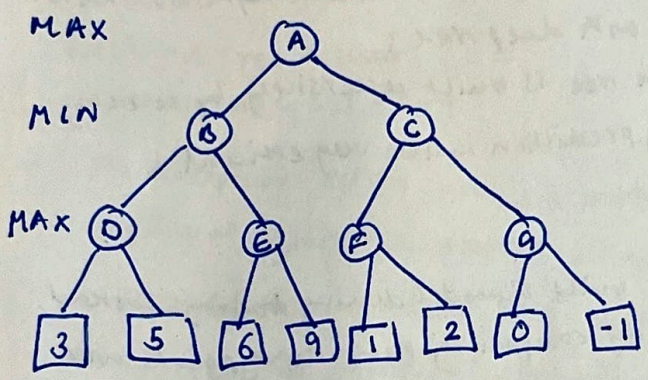
ARTIFICIAL INTELLIGENCE

HARMAN SETHI
CSE-A
0461702722

1. Alpha-beta pruning is a technique used to improve the efficiency of the minimax algorithm by reducing the number of nodes that need to be evaluated in a game tree.

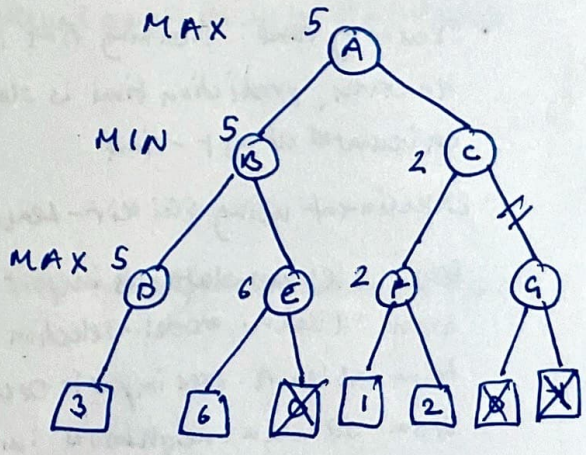
Alpha: Represents the best value the maximizing player can guarantee so far.

Beta: Represents the best value the minimizing player can guarantee so far.



1. Start at Node A (MAX):
   - Alpha = $-\infty$, Beta = ∞
   - A will explore its left child B first
2. Node B (MIN):
   - Alpha = $-\infty$, Beta = ∞ (inherited from A)
   - B explores D first
3. Node D (MAX):
   - Left child returns 3 ⇒ alpha = max($-\infty$, 3) = 3
   - Right child returns 5 ⇒ alpha = max(3, 5) = 5
   - D returns 5 to B
4. Back to Node B
   - Beta = min(∞, 5) = 5
   - Now, B explores E
5. Node E (MAX):
   - Alpha = $-\infty$, Beta = 5 (inherited from B)
   - Left child returns 6 ⇒ alpha = max($-\infty$, 6) = 6
   - Since alpha (6) > beta (5), pruning happened
   - E returns 6 to B
6. Back to Node B:
   - Beta = min(5, 6) = 5
   - B returns 5 to A
7. Back to Node A
   - Alpha = max($-\infty$, 5) = 5
   - A now evaluates right child C
8. Node C (MIN):
   - Alpha = 5, Beta = ∞
   - C explores F
9. Node F (MAX):
   - Left child returns 1 ⇒ alpha = MAX(5, 1) = 5
   - Right child returns 2 ⇒ alpha remains 5
   - F returns 2 to C
10. Back to Node C:
    - Beta = min(∞, 2) = 2
    - Now Beta (2) ≤ Alpha (5) ⇒ pruning happens
11. Final comparison at Node A
    - A chooses max (5, 2) = 5

Hence, the optimal value for the maximizer is 5



2. A decision tree is a supervised learning algorithm that splits data into branches based on features values. Each external node represents a decision on an attribute, each branch represents the outcome of the decision, and each leaf node represent a class label

In terms of

- Interpretability: Decision Tree are highly interpretable because they can be visualized as a flowchart. Each decision is easy to follow and explain, making them useful for domains where explanation is critical
- Performance: They generally perform well on both classification and regression tasks. However, they are prone to overfitting especially with deep trees
- Training Time: It is relatively fast because the tree is built recursively by selecting the best features of each node. Once built, prediction is also very efficient

K-Nearest Neighbours (KNN)

KNN is a lazy learning algorithm that does not build a model during training. Instead, it stores the training data and makes prediction by comparing new input points with its k-nearest neighbours using a distance metric

In terms of

- Interpretability: It is not very interpretability. It does not provide clear reasoning of a model - decisions are made based on proximity to other data points
- Performance: KNN can perform well on small dataset but struggles with high-dimensional data due to the curse of dimensionality. It is also sensitive to noisy data and irrelevant features
- Training time: Training time is almost zero since the model simply stores data. However, prediction time is slower because distances to all training points need to be calculated at test-time

Experiment using Scikit-Learn

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbours import kNeighborsClassifier
from sklearn.metrics import accuracy_score

iris = load_iris()
x = iris.data
y = iris.target

# Split into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)

# decision tree
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)
```

dt_pred = dt.predict(x_test)

knn

knn = kNeighboursClassifier(n_neighbors = 3)

knn.fit(x_train, y_train)

knn_pred = knn.predict(x_test)

Accuracy

print("Decision Tree Accuracy", accuracy_score(y_test, dt_pred))

print("KNN Accuracy:", accuracy_score(y_test, knn_pred))

OUTPUT

Decision Tree Accuracy: 1.0

KNN Accuracy: 0.9777

3. The Reception-action cycle is a fundamental concept in AI, especially in the context of intelligent agents and robotics. It refers to the continuous loop through which an agent:

1. Receives the environment using its sensors
2. Processes the perceptual information to make decisions
3. Acts upon the environment using its actuators

Components of the reception-action cycle

1. Sensors (Reception):
   These are used to gather data from the environment
2. Processing / Decision making unit:
   Based on the sensed information and the agents internal goals or rules. It decides the next action to take
3. Actuators (Actions)
   These execute the decision, made by the agents
4. Environment:
   The external world in which the agents operates. It continuously changes as the agent performs actions or new data become available

Use in Robotics Systems:

In robotics, the perception action cycle is crucial for autonomy

- Reception: A robot uses a camera to detect an obstacle in its path
- Decision-making: The robot determines that it must turn to avoid the obstacle
- Action: It moves its wheel or arms to steer around the object
- Cycle Repeats: After moving, it's perceives again to check it's position and surroundings

Example: Consider a vaccum cleaning robot:

- It senses dirt and obstacles using in built sensors
- It decides the direction to move in real-time
- It acts by moving forward, turning or avoiding furniture. This process happens continuously, allowing the robot to adapt to a changing environment

4. A Fuzzy set is a set without a crisp, clearly defined boundary. It allows elements to have partial membership, described by a membership function ranging from 0 to 1

- In a fuzzy set, an element can partially belong to a set

For ex: a temperature of 25°C can be 0.6 warm and 0.4 cold

Feature	Crisp Sets	Fuzzy Sets
Membership	Binary (either 0 or 1)	Gradual (values b/w 0 and 1)
Boundary	Sharp	Undefined
Logic Used	classical Boolean Logic	Fuzzy logic
Example	Temperature > 30°C = Hot	Temperature 28°C = (Warm, 0.3 Hot)

Real-world Scenario

- Crisp logic:

If temperature > 30°C → fan at full speed

Else → fan off

Problem: This causes abrupt changes

- Fuzzy logic:

Fan speed increases gradually from 'low' to 'high' as temperature rises

Benefit: Smooth transitions, more natural and energy-efficient control

5. A Bayesian Network is a type of probabilistic graphical model that represents a set of variables and their conditional dependencies via a directed acyclic graph (DAG).

Each node represents a random variable, and edges represent direct dependencies

Conditional independence refers to a situation where two random variables are independent of each other given the value of the third variable

Why is conditional independence important in Bayesian Networks

- Reduces complexity: without conditional independence, the full joint probability distribution grows exponentially with the no. of variables. So, it helps factors the joint distribution into smaller, manageable parts
- Efficient Inference: It allows algorithms to make probabilistic predictions more efficiently by ignoring irrelevant variables once certain conditions are known
- Simplifies Modeling: The network structure captures conditional independencies, simplifying the design of intelligent systems

Example: the sprinkler model

Consider the following variables:

- Rain (R)
- Sprinkler (S)
- Wet Grass (W)
- Rain and Sprinkler are both parents of Wet Grass

This means wet grass is influenced by both Rain and the Sprinkler

Now, analyze conditional independence

- Unconditionally, Rain and sprinkler might appear related: If it rains, the sprinkler is less likely to be turned on
- But, conditionally, given the knowledge that the grass is wet (W), the events rain and sprinkler becomes conditionally independent

$$P(R|S,W) = P(R|W)$$