HIT Summer Course 2025 - AI/ML Assignment Solutions

Part A: Search Algorithms

Problem Formulation

Initial State: Starting at city Arad
Goal State: Reaching city Bucharest
Solution: A sequence of cities from Arad to Bucharest
Transition Model: Moving from one city to an adjacent city along a road

Example of Transition Model:

- From Arad, we can move to: Sibiu (140 km), Timisoara (118 km), or Zerind (75 km)
- From Sibiu, we can move to: Arad (140 km), Fagaras (99 km), Oradea (151 km), or Rimnicu Vilcea (80 km)

1) Problem Formulation

States: Cities in Romania (Arad, Bucharest, Sibiu, etc.)
Actions: Drive from one city to an adjacent city
Transition Model: result(state, action) = new_state
Goal Test: state == Bucharest
Path Cost: Sum of road distances

Initial State: Arad
Goal State: Bucharest
Solution: A path from Arad to Bucharest
Example Transition: result(Arad, "drive to Sibiu") = Sibiu

2) Breadth-First Search Solution

BFS explores all nodes at the current depth before moving to the next level.

Step-by-step execution:

1. **Level 0:** Start with Arad
    - Frontier: [Arad]
    - Explored: []
2. **Level 1:** Expand Arad
    - From Arad: Sibiu (140), Timisoara (118), Zerind (75)
    - Frontier: [Sibiu, Timisoara, Zerind]
    - Explored: [Arad]
3. **Level 2:** Expand Sibiu, Timisoara, Zerind
    - From Sibiu: Fagaras (99), Oradea (151), Rimnicu Vilcea (80)
    - From Timisoara: Lugoj (111)
    - From Zerind: Oradea (71)
    - Frontier: [Fagaras, Oradea, Rimnicu Vilcea, Lugoj]
    - Explored: [Arad, Sibiu, Timisoara, Zerind]

4. **Level 3:** Expand Fagaras, Oradea, Rimnicu Vilcea, Lugoj
    - From Fagaras: Bucharest (211) ← **GOAL FOUND!**
    - From Oradea: Sibiu (151) [already explored]
    - From Rimnicu Vilcea: Pitesti (97), Sibiu (80) [already explored]
    - From Lugoj: Mehadia (70)

Solution Path: Arad → Sibiu → Fagaras → Bucharest
Total Distance: 140 + 99 + 211 = 450 km

3) A* Search Solution

A* uses $f(n) = g(n) + h(n)$ where $g(n)$ is path cost and $h(n)$ is heuristic (straight-line distance to Bucharest).

Step-by-step execution:

1. **Start:** Arad
    - $f(Arad) = g(Arad) + h(Arad) = 0 + 366 = 366$
    - Frontier: [Arad(366)]
    - Explored: []
2. **Expand Arad:**
    - Sibiu: $g = 140$, $h = 253$, $f = 140 + 253 = 393$

- Timisoara: $g = 118$, $h = 329$, $f = 118 + 329 = 447$
- Zerind: $g = 75$, $h = 374$, $f = 75 + 374 = 449$
    - Frontier: [Sibiu(393), Timisoara(447), Zerind(449)]
    - Explored: [Arad]

3. **Expand Sibiu (lowest f-value):**
    - Fagaras: $g = 140 + 99 = 239$, $h = 176$, $f = 239 + 176 = 415$

- Oradea: $g = 140 + 151 = 291$, $h = 380$, $f = 291 + 380 = 671$
- Rimnicu Vilcea: $g = 140 + 80 = 220$, $h = 193$, $f = 220 + 193 = 413$
    - Frontier: [Rimnicu Vilcea(413), Fagaras(415), Timisoara(447), Zerind(449)]
    - Explored: [Arad, Sibiu]

4. **Expand Rimnicu Vilcea (lowest f-value):**
    - Pitesti: $g = 220 + 97 = 317$, $h = 100$, $f = 317 + 100 = 417$
    - Frontier: [Fagaras(415), Pitesti(417), Timisoara(447), Zerind(449)]
    - Explored: [Arad, Sibiu, Rimnicu Vilcea]
5. **Expand Fagaras (lowest f-value):**
    - Bucharest: $g = 239 + 211 = 450$, $h = 0$, $f = 450 + 0 = 450$ ← **GOAL FOUND!**

Solution Path: Arad → Sibiu → Fagaras → Bucharest
Total Distance: $140 + 99 + 211 = 450$ km

Note: A* found the same path as BFS in this case, but A* is more efficient as it uses heuristic information to guide the search.

Part B: Propositional Logic

Analysis of Well-Formed Formulas

Tautology: A formula that is always true regardless of the truth values of its variables.
Contradiction: A formula that is always false regardless of the truth values of its variables.
Neither: A formula that can be either true or false depending on the truth values of its variables.

Formula Analysis:

1) (P ∧ Q) → (P ∨ Q)

- This is a **TAUTOLOGY**
- **Explanation:** If both P and Q are true, then at least one of them must be true. This is always true.

2) (P ∨ Q) ∧ (¬P ∧ ¬Q)

- This is a **CONTRADICTION**
- **Explanation:** This says "either P or Q is true" AND "neither P nor Q is true" - impossible!

3) (P → Q) ↔ (¬Q → ¬P)

- This is a **TAUTOLOGY**
- **Explanation:** This is the logical equivalence of contrapositive - if P implies Q, then not Q implies not P.

4) (P ∧ Q) ∨ (¬P ∧ ¬Q)

- This is **NEITHER** (it's a contingency)
- **Explanation:** This is true when P and Q have the same truth value, false when they differ.

5) (P → Q) ∧ (Q → R) → (P → R)

- This is a **TAUTOLOGY**
- **Explanation:** This is the logical principle of transitivity - if P implies Q and Q implies R, then P implies R.

Part C: Decision Tree with ID3 Algorithm

ID3 Algorithm Steps

Dataset Analysis:

- Target variable: "Play" (yes/no)
- Features: Outlook, Temperature, Humidity, Wind
- Total instances: 14
- Positive cases (yes): 9
- Negative cases (no): 5

Step 1: Calculate Entropy of the Target Variable

$Entropy(S) = -p_{yes} \times \log_2(p_{yes}) - p_{no} \times \log_2(p_{no})$

- $p_{yes} = 9/14 = 0.643$
- $p_{no} = 5/14 = 0.357$
- $Entropy(S) = -0.643 \times \log_2(0.643) - 0.357 \times \log_2(0.357)$
- $Entropy(S) = -0.643 \times (-0.637) - 0.357 \times (-1.485)$
- $Entropy(S) = 0.410 + 0.530 = 0.940$

Step 2: Calculate Information Gain for Each Feature

For Outlook:

- Sunny: 5 instances (2 yes, 3 no)
- Overcast: 4 instances (4 yes, 0 no)
- Rainy: 5 instances (3 yes, 2 no)

$$Entropy(Sunny) = -\frac{2}{5} \times \log_2(\frac{2}{5}) - \frac{3}{5} \times \log_2(\frac{3}{5}) = 0.971$$
$$Entropy(Overcast) = 0(allsameclass)$$
$$Entropy(Rainy) = -\frac{3}{5} \times \log_2(\frac{3}{5}) - \frac{2}{5} \times \log_2(\frac{2}{5}) = 0.971$$
$$InformationGain(Outlook) = Entropy(S) - \sum(\frac{|S_v|}{|S|} \times Entropy(S_v))$$
$$IG(Outlook) = 0.940 - (\frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971)$$
$$IG(Outlook) = 0.940 - (0.347 + 0 + 0.347) = 0.246$$

For Temperature:

- Hot: 4 instances (2 yes, 2 no)
- Mild: 6 instances (4 yes, 2 no)
- Cool: 4 instances (3 yes, 1 no)

$$Entropy(Hot) = -\frac{2}{4} \times \log_2(\frac{2}{4}) - \frac{2}{4} \times \log_2(\frac{2}{4}) = 1.000$$
$$Entropy(Mild) = -\frac{4}{6} \times \log_2(\frac{4}{6}) - \frac{2}{6} \times \log_2(\frac{2}{6}) = 0.918$$
$$Entropy(Cool) = -\frac{3}{4} \times \log_2(\frac{3}{4}) - \frac{1}{4} \times \log_2(\frac{1}{4}) = 0.811$$
$$IG(Temperature) = 0.940 - (\frac{4}{14} \times 1.000 + \frac{6}{14} \times 0.918 + \frac{4}{14} \times 0.811)$$
$$IG(Temperature) = 0.940 - (0.286 + 0.393 + 0.232) = 0.029$$

For Humidity:

- High: 7 instances (3 yes, 4 no)
- Normal: 7 instances (6 yes, 1 no)

$$Entropy(High) = -\frac{3}{7} \times \log_2(\frac{3}{7}) - \frac{4}{7} \times \log_2(\frac{4}{7}) = 0.985$$
$$Entropy(Normal) = -\frac{6}{7} \times \log_2(\frac{6}{7}) - \frac{1}{7} \times \log_2(\frac{1}{7}) = 0.592$$
$$IG(Humidity) = 0.940 - (\frac{7}{14} \times 0.985 + \frac{7}{14} \times 0.592)$$
$$IG(Humidity) = 0.940 - (0.493 + 0.296) = 0.151$$

For Wind:

- Weak: 8 instances (6 yes, 2 no)
- Strong: 6 instances (3 yes, 3 no)

$$Entropy(Weak) = -\frac{6}{8} \times \log_2(\frac{6}{8}) - \frac{2}{8} \times \log_2(\frac{2}{8}) = 0.811 Entropy(Strong) = -\frac{3}{6} \times \log_2(\frac{3}{6}) - \frac{3}{6} \times \log_2(\frac{3}{6}) = 1.000 IG(Wind) = 0.940 - (\frac{8}{14} \times 0.811 + \frac{6}{14} \times 1.000) IG(Wind) = 0.940 - (0.463 + 0.429) = 0.048$$

Step 3: Select Root Node

Information Gains:

- Outlook: 0.246 (highest)
- Humidity: 0.151
- Wind: 0.048
- Temperature: 0.029

Root Node = Outlook

Step 4: Build Tree Recursively

Outlook = Overcast branch:

- All instances are "yes" → Leaf node with "yes"

Outlook = Sunny branch:

- Subset: 5 instances (2 yes, 3 no)
- Calculate IG for remaining features (Temperature, Humidity, Wind)
- Humidity has highest IG → Split on Humidity
- High: 3 instances (0 yes, 3 no) → Leaf "no"
- Normal: 2 instances (2 yes, 0 no) → Leaf "yes"

Outlook = Rainy branch:

- Subset: 5 instances (3 yes, 2 no)
- Calculate IG for remaining features
- Wind has highest IG → Split on Wind
- Weak: 3 instances (3 yes, 0 no) → Leaf "yes"
- Strong: 2 instances (0 yes, 2 no) → Leaf "no"

Final Decision Tree:

```
Outlook
├── Sunny
|   └── Humidity
|       ├── High → No
|       └── Normal → Yes
├── Overcast → Yes
└── Rainy
    └── Wind
        ├── Weak → Yes
        └── Strong → No
```

Classification Rules:

1. If Outlook = Overcast → Play = Yes
2. If Outlook = Sunny AND Humidity = High → Play = No
3. If Outlook = Sunny AND Humidity = Normal → Play = Yes
4. If Outlook = Rainy AND Wind = Weak → Play = Yes
5. If Outlook = Rainy AND Wind = Strong → Play = No