CAB320 - 2023 Exam Prep Guide

Jaden Ussher

July 25, 2024

Question 1: Edit Distance

The edit distance problem involves finding the minimum number of operations required to transform one string into another. The allowed operations are insertions, deletions, and substitutions (mismatches).

Steps to Solve Edit Distance using Dynamic Programming

1. **Define the DP Table:** Let $dp[i][j]$ represent the minimum edit distance between the first i characters of string $s1$ and the first j characters of string $s2$.

2. **Initialize the DP Table:** - $dp[0][0] = 0$ - $dp[i][0] = i$ for all i (cost of deleting all characters from $s1$) - $dp[0][j] = j$ for all j (cost of inserting all characters into $s1$)

$$dp[i][0] = i \quad \text{for } 0 \leq i \leq m$$
$$dp[0][j] = j \quad \text{for } 0 \leq j \leq n$$

3. **Fill the DP Table:** For each pair of indices (i, j): - If $s1[i-1] == s2[j-1]$, then $dp[i][j] = dp[i-1][j-1]$ (no operation needed) - Else, choose the minimum cost among:

$$dp[i][j] = \min \begin{cases} dp[i-1][j] + 1 & \text{(deletion)} \\ dp[i][j-1] + 1 & \text{(insertion)} \\ dp[i-1][j-1] + 1 & \text{(substitution)} \end{cases}$$

```
for i from 1 to m:
    for j from 1 to n:
        if s1[i-1] == s2[j-1]:
            dp[i][j] = dp[i-1][j-1]
        else:
            dp[i][j] = min(dp[i-1][j] + 1, dp[i][j-1] + 1, dp[i-1][j-1] + 1)
```

4. **Extract the Result:** The value $dp[m][n]$ contains the minimum edit distance between $s1$ and $s2$.

Example Problem

Calculate the edit distance between the strings "kitten" and "sitting".

Solution

1. **Initialization:**

	$\emptyset$	s	i	t	t	i	n	g
$\emptyset$	0	1	2	3	4	5	6	7
k	1	0	0	0	0	0	0	0
i	2	0	0	0	0	0	0	0
t	3	0	0	0	0	0	0	0
t	4	0	0	0	0	0	0	0
e	5	0	0	0	0	0	0	0
n	6	0	0	0	0	0	0	0

2. **Filling the DP Table:**

	$\emptyset$	s	i	t	t	i	n	g
$\emptyset$	0	1	2	3	4	5	6	7
k	1	1	2	3	4	5	6	7
i	2	2	1	2	3	4	5	6
t	3	3	2	1	2	3	4	5
t	4	4	3	2	1	2	3	4
e	5	5	4	3	2	2	3	4
n	6	6	5	4	3	3	2	3

3. **Result:** The minimum edit distance is $dp[6][7] = 3$.

Question 2: Search Problems

Defining State Representation

When defining a state representation for a problem: - Identify all the relevant attributes that uniquely determine the state. - Ensure the state representation is complete and minimal.

Example: For a puzzle game, a state might include: - The position of each piece on the board. - The number of moves taken so far.

Example Problem

Define a state representation for the 8-puzzle problem.

Solution

A state can be represented by: - A 3x3 matrix (or a 1D array of length 9) representing the position of each tile. - The position of the blank tile.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & \emptyset \end{bmatrix}$$

State representation: - Array: [1, 2, 3, 4, 5, 6, 7, 8, 0] - Blank tile position: 8 (indexing from 0)

Admissible Heuristics

An admissible heuristic $h(s)$ is one that never overestimates the cost to reach the goal from state s.

Example: For the 8-puzzle problem, the Manhattan distance is an admissible heuristic.

Example Problem

Prove that the Manhattan distance is an admissible heuristic for the 8-puzzle problem.

Solution

The Manhattan distance is the sum of the absolute values of the differences in the horizontal and vertical positions of the tiles from their goal positions.

For each tile i:

$$h_i(s) = |x_i - x_i^*| + |y_i - y_i^*|$$

where (x_i, y_i) is the current position of tile i and (x_i^*, y_i^*) is the goal position of tile i.

Since each move of a tile to its goal position contributes exactly 1 to the Manhattan distance and no move can decrease the distance by more than 1, the Manhattan distance never overestimates the cost.

Question 3: Reinforcement Learning

Tabular Methods

Understand the Bellman equations and how they apply to value functions and Q-functions.

Value Function:

$$V(s) = \max_a Q(s,a)$$

Q-Learning Update Rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

SARSA Update Rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma Q(s',a') - Q(s,a) \right]$$

Example Problem

Consider a simple MDP with three states: A, B, C. The transition and reward structure is as follows: - From A, you can go to B with reward -1. - From B, you can go to C with reward 2. - From C, you can go to A with reward 0. - From any state, you can stay in the same state with reward -1.

Apply one step of the Q-learning update rule with $\alpha = 0.1$ and $\gamma = 0.9$ for the transition from state A to state B.

Solution

Assume current Q-values are:

$$Q(A,B) = 0.5, \quad Q(B,C) = 0.6, \quad Q(C,A) = 0.4$$

Step-by-step: 1. Observe the transition from A to B with reward $r = -1$. 2. Calculate the updated Q-value for $Q(A,B)$:

$$Q(A,B) \leftarrow 0.5 + 0.1 \left[-1 + 0.9 \cdot \max_{a'} Q(B,a') - 0.5 \right]$$

Assume $\max_{a'} Q(B,a') = Q(B,C) = 0.6$:

$$Q(A,B) \leftarrow 0.5 + 0.1 \left[-1 + 0.9 \cdot 0.6 - 0.5 \right]$$

$$Q(A,B) \leftarrow 0.5 + 0.1 \left[-1 + 0.54 - 0.5 \right]$$

$$Q(A,B) \leftarrow 0.5 + 0.1 \left[-0.96 \right]$$

$$Q(A,B) \leftarrow 0.5 - 0.096$$

$$Q(A,B) \leftarrow 0.404$$

Question 4: Bayes Rule

Bayes Rule is used to update the probability estimate for a hypothesis given new evidence.

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

Example Problem

A medical test for a disease is 99% sensitive (true positive rate) and 95% specific (true negative rate). The disease prevalence is 1

Solution

1. Define the events: - D: Having the disease - T: Positive test result
2. Given: - $P(T|D) = 0.99$ - $P(T|\neg D) = 1 - 0.95 = 0.05$ - $P(D) = 0.01$ - $P(\neg D) = 0.99$
3. Use Bayes' Rule:
$$P(D|T) = \frac{P(T|D) \cdot P(D)}{P(T)}$$
4. Calculate $P(T)$:
$$P(T) = P(T|D) \cdot P(D) + P(T|\neg D) \cdot P(\neg D)$$
$$P(T) = 0.99 \cdot 0.01 + 0.05 \cdot 0.99$$
$$P(T) = 0.0099 + 0.0495 = 0.0594$$
5. Calculate $P(D|T)$:
$$P(D|T) = \frac{0.99 \cdot 0.01}{0.0594}$$
$$P(D|T) = \frac{0.0099}{0.0594}$$
$$P(D|T) \approx 0.1667$$

So, the probability that a person has the disease given a positive test result is approximately 16.67

Q-Learning Example

Problem Description:

Given the following map where the player starts on tile 2 and ends on tile 6. The player receives 10 points for reaching tile 6 and -1 points for all other tiles. From each tile, the player can choose to go (up, down, left, right), but cannot move outside the given tiles. The $Q(s, a)$ values have been provided:

- $Q(1, \text{down}) = 4$
- $Q(1, \text{right}) = 6$
- $Q(2, \text{left}) = 2$
- $Q(2, \text{right}) = 4$
- $Q(2, \text{down}) = 3$
- $Q(3, \text{left}) = 5$
- $Q(3, \text{down}) = 7$
- $Q(4, \text{right}) = 4$
- $Q(4, \text{up}) = 8$
- $Q(5, \text{left}) = 5$
- $Q(5, \text{right}) = 6$
- $Q(5, \text{up}) = 8$

Grid Layout:

1	2	3
4	5	6

Task:

1. Set up a Q-table with the corresponding values.
2. With learning rate $\alpha = 0.25$ and discount rate $\gamma = 0.75$, if the player travels down first then right, what are the updated values in the Q-table?

Solution

1. Q-Table Setup

$Q =$

State	Action	Q-Value
1	down	4
1	right	6
2	left	2
2	right	4
2	down	3
3	left	5
3	down	7
4	right	4
4	up	8
5	left	5
5	right	6
5	up	8

2. Q-Learning Update Process

Definitions:

- Learning rate $\alpha = 0.25$
- Discount rate $\gamma = 0.75$

Observations:

1. Initial State: State 2
2. Action: Down (to State 5)
3. Reward: -1 (since it's not tile 6)
4. Next State: State 5
5. Next Action: Right (to State 6)
6. Next Reward: +10 (for reaching tile 6)

Step 1: From State 2, Action Down

$$Q(2, \text{down}) \leftarrow Q(2, \text{down}) + \alpha \left[\text{Reward} + \gamma \max_{a'} Q(5, a') - Q(2, \text{down})\right]$$

Where:

- Reward $= -1$
- $\max_{a'} Q(5, a') = \max(5, 6, 8) = 8$ (from the Q-values of state 5)

$$Q(2, \text{down}) \leftarrow 3 + 0.25 \left[-1 + 0.75 \cdot 8 - 3\right]$$
$$Q(2, \text{down}) \leftarrow 3 + 0.25 \left[-1 + 6 - 3\right]$$
$$Q(2, \text{down}) \leftarrow 3 + 0.25 \left[2\right]$$
$$Q(2, \text{down}) \leftarrow 3 + 0.5 = 3.5$$

Step 2: From State 5, Action Right

$$Q(5, \text{right}) \leftarrow Q(5, \text{right}) + \alpha \left[\text{Reward} + \gamma \max_{a'} Q(6, a') - Q(5, \text{right})\right]$$

Where:

- Reward $= 10$
- $\max_{a'} Q(6, a') = 0$ (assuming Q-values for state 6 are initially 0 since it's the goal)

$$Q(5, \text{right}) \leftarrow 6 + 0.25 \left[10 + 0.75 \cdot 0 - 6\right]$$
$$Q(5, \text{right}) \leftarrow 6 + 0.25 \left[10 - 6\right]$$
$$Q(5, \text{right}) \leftarrow 6 + 0.25 \left[4\right]$$
$$Q(5, \text{right}) \leftarrow 6 + 1 = 7$$

Final Updated Q-Table

State	Action	Q-Value
1	down	4
1	right	6
2	left	2
2	right	4
2	down	3.5
3	left	5
3	down	7
4	right	4
4	up	8
5	left	5
5	right	7
5	up	8

Summary

The Q-Learning algorithm updates the Q-values based on the observed transitions and rewards. By iterating through multiple episodes and updating the Q-table, the agent learns the optimal policy to maximize the cumulative reward.