tNational University of Singapore

CS2109S—Introduction to AI and Machine Learning

Midterm Assessment - Solution for Examplify Questions V2

Semester 1, 2025/2026

Time allowed: 1 hour 30 minutes

	Number of questions	Total marks
Uninformed Search	6	9
Informed Search	4	8
Local Search	1	4
Adversarial Search	7	10
Decision Trees	6	7
Linear/Logistic Regression	6	12
Total	30	50

Part 1: Uninformed Search

Context

The Elevator Problem

A single elevator serves a 20-floor building. At time zero, a fixed and finite set of N passengers appears, each passenger has a known source floor and destination floor (1-20), where source floor ≠ destination floor. The elevator starts at floor 1, empty. By the most recent elevator inspection, the certified capacity of the elevator, i.e., the number of passengers that it is allowed to carry is up to C passengers at a time, where C ≥ 0 is fixed.

At each step, the elevator chooses one of three actions: move up, move down, or open on the current floor. Passengers only enter or exit the elevator when the action open is explicitly chosen, including in the initial state. Passengers cannot exit at non-destination floors.

Each action costs 1. The goal is to deliver all passengers to their destinations using the fewest actions. The elevator's final floor does not matter.

Problem Formulation

- **States**:
  - Elevator's current floor (1-20)
  - Status of each passenger (waiting, in elevator, delivered)

- **Initial State**:
  - Elevators' current floor = 1
  - All passengers are waiting
- **Actions**:
  - Move up (condition: current floor < 20)
  - Move down (condition: current floor > 1)
  - Open
- **Transition Function**:
  - Move up/down move the elevator up/down one floor
  - Open:
    - All passengers in the elevator whose destination floor is the current floor will be delivered.
    - All passengers waiting on the current floor will enter the elevator until the elevator's capacity is full.
    - These movements of the passengers are mandatory.
- **Goal Test**: All passengers: status = delivered.
- **Path Cost**: Sum of the costs of the actions taken.

Note:

- A solution may not be optimal.
- An optimal solution is the least-cost solution.

Analyze the search formulation above and answer questions 1A-1F.

Questions

1A. [1 mark] Is it true that the search formulation results in a state space where a state **is reachable through more than one sequence of actions**? Note: we do not care about the search algorithms in this question since we are asking about the state space, not the search tree.

A. Yes
B. No

1B. [1 mark] Does the search formulation result in **more than one goal states**?

A. Yes
B. No

1C. [1 mark] Is there **always** a solution when using the search formulation?

A. Yes
B. No

1D. [2 marks] Suppose that you use search (without visited memory). Which search algorithm(s) always terminate? Select all that apply.

a. Queue-based search (i.e., BFS, UCS)
b. Depth-First Search (DFS)
c. None of the above

1E. [2 marks] Which of the following search (without visited memory) algorithm(s) can we employ such that the search **always finds an answer** (valid solution) if a solution exists? Select all that apply.

a. Breadth-First Search (BFS)
b. Depth-First Search (DFS)
c. Uniform-Cost Search (UCS)
d. Depth-Limited Search (DLS) with DFS and max-depth (40+C)*N/C

e. None of the above

1F. [2 marks] Suppose that we use **search with visited memory**. Which of the following search algorithm(s) is/are the best for the problem? Select all that apply.
Best means the algorithm(s) should be complete, optimal, efficient (in terms of big O worst-case space and time complexity), and aware if there is no solution.

a. Depth-First Search (DFS)
b. Uniform-Cost Search (UCS)
c. Depth-Limited Search (DLS) with DFS and max-depth (20+C)*N/C+1
d. Iterative Deepening Search (IDS) with DFS
e. None of the above

Solution

Note: During the midterm exam, we clarified that we define (implementation of) division by zero to equal infinity. In other words, our implementation of division is similar to this in pseudocode:

```
def divide(numerator, denominator):
    if denominator == 0:
        return ∞
    return numerator / denominator
```

1A. A. Yes

The same state (same floor and same passenger statuses) can be reached via different action sequences due to cycles (e.g., moving up then down vs. staying and opening at different times), so multiple distinct paths can lead to the same state.

1B. A. Yes

A goal state requires all passengers delivered, but the elevator's final floor is irrelevant. Therefore, there are multiple goal states—one for each possible final floor (1–20).

1C. B. No

In this problem, the certified capacity of the elevator can be zero, indicating that it cannot carry any passengers, for instance, due to elevator maintenance.

If capacity C = 0 and there is at least one passenger (N > 0), no passenger can ever enter, so no solution exists. Hence, not always solvable under the formulation.

1D. C. None of the above

Without visited memory, all these algorithms can revisit states indefinitely due to reversible actions which cause cycles. BFS/UCS can keep generating longer paths to the same states if there is no solution; DFS can get stuck down an infinite cycle. Therefore, none of the given algorithms "always terminate" in general.

1E. A. Breadth-First Search (BFS); c. Uniform-Cost Search (UCS)

The question addresses scenarios where a solution exists, indicating cases where C > 0.

Without visited memory,

- BFS is complete on finite branching with unit step cost and will find a solution if one exists because the number of sequences up to the solution depth is finite.
- UCS with positive action costs (here all 1) will also find a solution if one exists, as it explores increasing path cost and will eventually expand the optimal path.
- DFS is not complete without visited (can loop).
- The proposed DLS bound $(40 + C) \cdot \frac{N}{C}$ is not sufficient in simple, allowed cases. Let the capacity be C=100, and N=1, then (40+100)/100= 1.4, which is not enough to deliver the 1 passenger.

1F. D. Iterative Deepening Search (IDS) with DFS

With visited memory:

- UCS is complete, optimal for positive costs, and will detect no-solution by exhausting reachable states. However, the space complexity is exponential.
- IDS with DFS is complete and optimal for unit step costs, and significantly more space-efficient (polynomial) while still detecting no-solution by exhausting depths across the finite state space.
- DFS alone is not optimal;
- DLS bound is not sufficient. See 1E for explanation. Also, when C=0, the bound is infinite using our definition of divide by 0.

Part 2: Informed Search

Context

A drone operates in a three-dimensional (3D) environment modeled as a 3D grid. The environment is described by coordinates (x, y, z), where (x, y) is the position relative to the ground surface and z is the height of the drone. The positions are represented by integers on a $N \times N$ square grid, i.e., $x \in \{1,2,\dots,N\}$ and $y \in \{1,2,\dots,N\}$. The heights are represented by $z \in \{1,2,\dots,M\}$. The initial state is a position $(x_{start}, y_{start}, 1)$.

At each position (x, y, z), there are the six possible actions: move up, move down, move left, move right, move forward, move backward, with the corresponding deterministic transitions, e.g., move up moves the drone from (x, y, z) to $(x, y, z + 1)$ for $z < M$. Actions that move outside the 3D grid are not possible. Due to the presence of static obstacles, some actions may not be allowed at certain positions. The cost of each valid action is 1.

Based on this description, answer questions 2A-2D. Notice that the questions include further assumptions on the problem setting.

Questions

2A: [2 marks]

In addition to the setting described in the Context, the drone must perform a set of tasks. These tasks are presented in a queue $Q = [(start\_i, end\_i)]$, which contains **distinct** 3D coordinates representing the start and end points of each task. Once a task is complete by reaching the end point, the task is removed from the queue.

The goal state is to complete all the tasks in sequence and return to the initial position. Select admissible heuristics to use for the A* algorithm. Here, L is the size of the current task queue. In addition, the Manhattan distance is $d_{MD}(a,b) = |x_a - x_b| + |y_a - y_b| + |z_a - z_b|$. Select all that apply.

A. $d_1((x,y,z),(x_{start},y_{start},1))$.
B. L
C. The sum $\sum_{i=1}^{L} D_i$ where D_i is $d_{MD}(start\_i, end\_i)$ of the i-th element of the current queue.
D. The sum $\sum_{i=2}^{L} D_i$ where D_i is $d_{MD}(start\_i, end\_i)$ of the i-th element of the current queue.
E. None of the above.

2B: [2 marks]

For this question, let the task queue be empty. In addition, the costs for each valid action at any valid position (x,y,z) are as follows:

- Move left/right/forward/backward = 1.
- Move up/down = 2.

Let the goal state be defined by reaching (x_1,y_1,z_1) from the initial state. Select **all** heuristics to use for the A* algorithm <u>with</u> visited memory such that it returns the optimal solution? Select all that apply.

A. $h\big((x,y,z)\big) = x + y + 2z$.
B. $h((x,y,z)) = x + y + z^2$.
C. $h((x,y,z)) = |x - x_1| + |y - y_1| + 2|z - z_1|$.
D. $h((x,y,z)) = |x - x_1| + |y - y_1| + |z - z_1|$.
E. $h((x,y,z)) = |x - x_1| + |y - y_1| + |z - z_1|^2$.
F. None of the above.

2C: [2 marks] For this question, let the task queue be empty. Consider the variable cost of valid actions and the goal described in Question **2B**. Select **the best** heuristics to use for the A* algorithm <u>with</u> visited memory such that it returns the optimal solution? Select one that applies.

A. $h\big((x,y,z)\big) = x + y + 2z$.
B. $h((x,y,z)) = x + y + z^2$.
C. $h((x,y,z)) = |x - x_1| + |y - y_1| + 2|z - z_1|$.
D. $h((x,y,z)) = |x - x_1| + |y - y_1| + |z - z_1|$.
E. $h((x,y,z)) = |x - x_1| + |y - y_1| + |z - z_1|^2$.
F. None of the above.

2D: [2 marks] For this question, let the task queue be empty. Consider the variable cost of valid actions as follows:

- Move left/right/forward/backward = 0.
- Move up/down = $2z$.

Let the goal state be defined by reaching $(x_1,y_1,1)$. The initial state is any valid state. Select **all** heuristics to use for the A* algorithm <u>without</u> visited memory such that it returns the optimal solution? Select all that apply.

A. $h((x,y,z)) = x + y + 4z^2$
B. $h((x,y,z)) = x + y + 2z^2 - 2$
C. $h((x,y,z)) = x + y + z^2 + z - 2$
D. $h((x,y,z)) = 4z^2 - 4$
E. $h((x,y,z)) = 2z^2 - 2$
F. $h((x,y,z)) = z^2 + z - 2.$
G. $h((x,y,z)) = z^2 - 1.$
H. None of the above.

Solution

2A. ABD

Any obstacles only increase the true cost, because the drone must navigate around them. For all questions consider the relaxed problem as well without any obstacles.

A is a solution because the goal state is returning to the initial state. At any current point (x, y, z), the drone must at least return to the initial state, hence the Manhattan distance from the current state to the initial state is an admissible heuristic.

B: First, notice that the question says, "**Once** a task is complete by reaching the **end point**, the task is removed from the queue."
The drone must solve all the tasks, which consists of **distinct** 3D start and end coordinates. Hence the cost of solving a single task is at least 1. Even if the drone is inside the first/current task, to get to the end point and removing the task from the queue, takes at least cost 1. Hence, L is an admissible heuristic.

C is not an admissible heuristic. Note that the queue element is only popped when the endpoint is reached, as the question says. The drone could be inside task 1 and already have accomplished most of that task. The task L could have an end point that is very close (say, one move away) from the initial state. Hence, there are states where the heuristic overcounts the cost to achieve the tasks and to return to the initial state, and the heuristic is not admissible.

D is admissible as it removes the first task from estimating the cost (see discussion for Option C).

2B. CD

A and B will overcount in cases when the current position is close to the goal state, hence they are not admissible and hence not consistent, i.e., we cannot expect them to produce an optimal solution in A* with visited memory.

For C and D, we check consistency by noticing that an up/down move changes $|z - z_1|$ by at most 1, either making it smaller or larger. The same holds for the other coordinates. For z, the move comes with a cost of 2, hence we always have that

$$h((x,y,z)) \leq 2 + h\big((x,y,z \pm 1)\big),$$

and a similar argument holds for the other moves.

For E, for $z = 1$ and $z_1 = 4$, the term $|z - z_1|^2 = 9$, while the cost of moving 3 steps is 6. Hence, the heuristic is not admissible and cannot be consistent.

2C. C

Choose the dominant heuristics among the consistent heuristics, which is C. It is always true that $|x - x_1| + |y - y_1| + |z - z_1| \leq |x - x_1| + |y - y_1| + 2|z - z_1|$.

2D. H -- UPDATED

Updated: We made a mistake in the previous version of the solution script. Recall that in this question, x, y moves are without cost and A* is without visited memory. Consider an example where the optimal solution requires a path that moves from z to $z + 1$, i.e., the drone must go up first before being able to land at $z = 1$. The up move will never be first in the priority queue, if there are any x, y moves available. The algorithm may cycle between the x, y coordinates indefinitely. It will never perform the necessary move to $z + 1$. Hence, for none of the heuristics will A* return the optimal solution, and selecting H gives full marks.

Local Search

Context

Refer to the description of the Elevator Problem.

Questions

3A. [4 marks] Given the Elevator Problem, which of the following local search formulation(s) is/are reasonable? Select all that apply.

In this context, we consider the formulation reasonable if hill-climbing with an infinitely large number of random restarts can return the optimal sequence of actions for the problem. Note that, at every restart, the initial state and the output of the successor function may differ from the previous run if there is randomness involved.

A.
State: (current floor, total cost so far).
Initial State: (floor 1, cost = 0).
Goal Test: cost ≥ 40.
Evaluation Function: negative of the path cost.
Successors: move up or open.

B.
State: (current floor, how many floors have been visited).
Initial State: (floor 1, visited = 1).
Goal Test: visited = 20.
Evaluation Function: number of “visited” floors.
Successors: next floor = current floor ± 1.

C.
State: (current floor, state of each passenger: waiting, in elevator, or delivered).
Initial State: elevator on floor 1, all passengers waiting.
Goal Test: all passengers are delivered; floor of elevator does not matter.
Evaluation Function: number of passengers delivered.
Successors: all states you get by either moving the elevator one floor up/down (if within 1–20) or opening the elevator (allowing boarding/alighting).

D.
State: (current floor, number of passengers in elevator).
Initial State: (floor 1, zero passengers in elevator).

Goal Test: elevator at floor 20.
Evaluation Function: number of visited floors.
Successors: move 1–3 floors up in a single step.

E.
None of the above

Solution

3A. E. None of the above

All the state representations do not include the information needed to extract the optimal solution (i.e., the optimal sequence of actions).

In local search, we start from a random state rather than the actual initial state, and the optimization steps explore neighbors of the current state without explicitly modeling the problem's actions. Therefore, the steps taken during optimization to reach an optimal final state are not the same as the sequence of actions from the actual initial state to the optimal goal state. Moreover, local search returns only the final state, not the path leading to it. Even if the path is returned, it also does not correspond the optimal sequence of actions.

Updated:
In the lecture slides, we have clearly stated that in local search, a state is a candidate solution and that hill climbing returns a state, not a path to a state. We have also given an example where we apply local search to a path finding problem. Notice that we had to modify the state formulation to encode the path so that it can work with local search. If the local search solution is a sequence of actions and that somehow this encodes the optimal solution, then there is no need for us to modify the formulation for path finding.

In the AY2024/2025 Sem 2 midterm, Option D was accepted as correct even though it did not explicitly include the path. This was based on an additional (but unstated) assumption in the solution following post-midterm queries: we allow an assumption that the MRT graph is mostly chains with no cycles. Under this assumption, the set of visited states uniquely determines the path, which is why we allowed Option D as correct. We will update the past paper and solution to make this assumption explicit and avoid future confusion. However, it is important to note that the correctness of answers in this semester's paper is independent from previous semesters.

In this semester's question, no optimal sequence of actions can be extracted from any of the state representations provided in the options, so the correct answer is E. It is illogical to accept clearly incorrect options.

Option C is also clearly incorrect, not only because the state lacks path information, but also because the formulation will always get stuck at a local optimum and return a poor final state. Consider a scenario where all passengers are initially at the 20th floor. The initial state in Option C is that the elevator starts at floor 1. Hill climbing will consider the successor states, which in this formulation are deterministic and they are: elevator at level 2 (move up) and elevator open (open). Here, the current state has a value of 0, and all the successors have an evaluation value of 0, so the hill climbing terminates and returns the initial state as the final state.

Adversarial Search

Minimax

Context

Parity Zig Zag

Players A and B take turns modifying a shared number that starts at 5. On each turn, the current player can make a move by adding or multiplying the number. If the current number is even, the player can either add 2 or 5 to it. If the current number is odd, the player can either add 4 or multiply it by 2.

The only way a player can win is by landing exactly on 20, causing the opponent to lose. In all other scenarios, the game ends in a draw. If a player cannot make a valid move—meaning all available moves exceed 20—the other player also cannot make a move, and the game ends in a draw.

We will use the **Minimax with cutoff** algorithm to solve the game by constructing the complete game tree up to a cutoff depth of 4. At the cutoff depth, if the node is non-terminal, the evaluation function will return -1 if the number is even, and +1 otherwise.

Notes:

- We assume that player A is the first player and is the max player.
- The value of a state is evaluated from the perspective of the max player.
- In terminal states, a win is valued at +1, a loss at -1, and a draw at 0.
- The game tree is *tree* (a child can only have one parent).
- Root node is at depth 0.

Construct the game tree and answer questions 4A-4E.

Questions

4A. [1 mark] How many terminal nodes are there in the game tree generated by the algorithm specified in the context?

4B. [1 mark] How many nodes in the **full** game tree generated by Minimax without cutoff exceed the cutoff depth specified in the context?

4C. [1 mark] What is the game tree's maximum depth if we don't perform cutoff? Note: please write "infinity" if the depth is infinite.

4D. [1 mark] If Player A moves to number 9 on their first turn, which player can win the game if they both play optimally in the subsequent turns?

A. A
B. B
C. Game is draw
D. Game continues indefinitely
E. None of the above

4E. [1 mark] Which sequences (from the first turn) guarantee a Player A victory if both players play optimally in the subsequent turns? Select all that is/are true.

A. A: x2 → B: +5
B. A: x2 → B: +2

C. A: x2 → B: +2 → A: +2
D. A: +4 → B: x2
E. None of the above

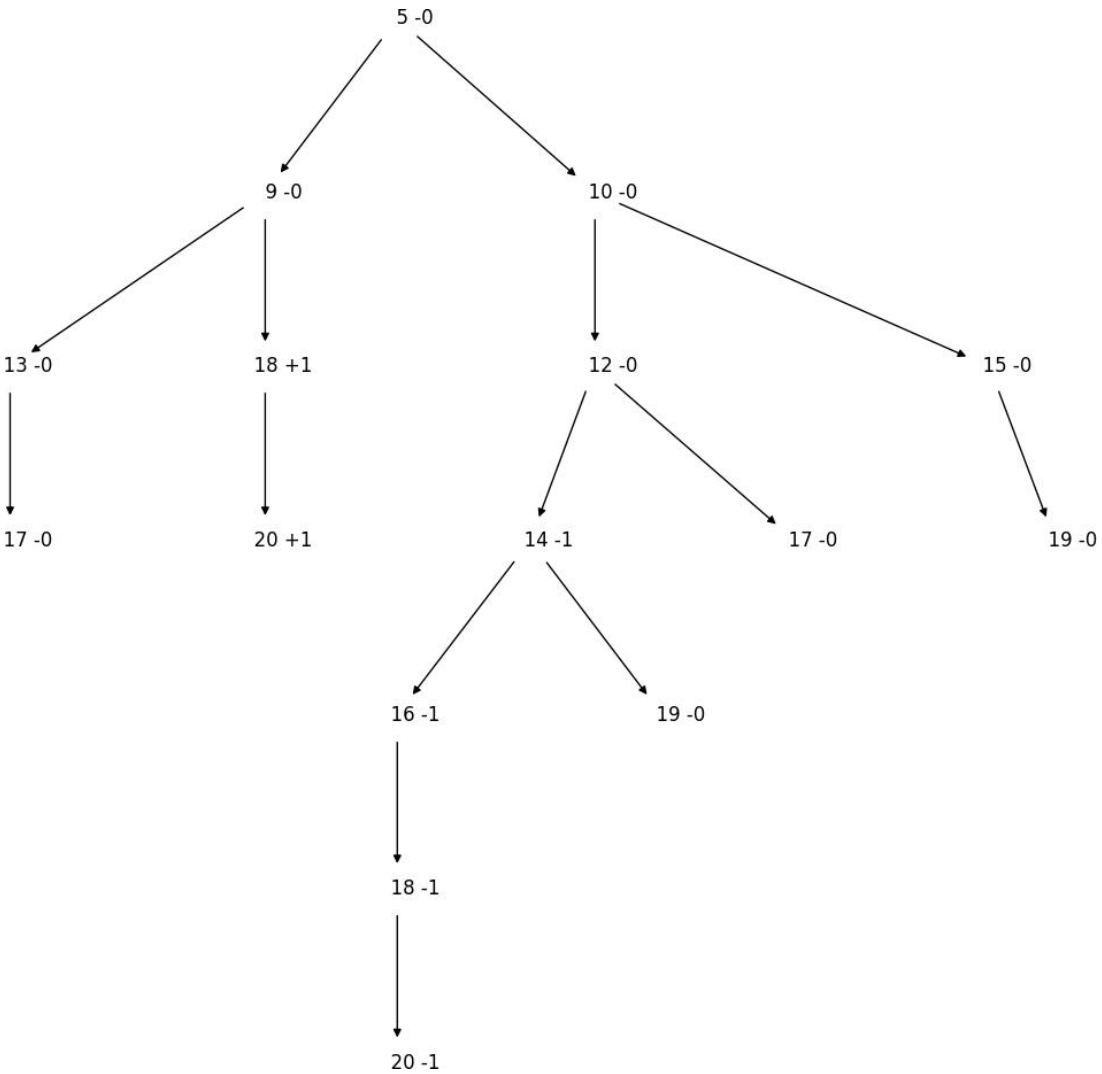
4F. [1 mark] Which of the following action(s) should the first player take on their first turn? Assume all players play optimally in the subsequent turns. Select all that is/are true.

A. +4
B. x2
C. All actions result in losing the game
D. All actions result in a draw
E. Game continues indefinitely

Solution

Notes:

- Player can't make a valid move means all moves are invalid, i.e., all moves result in a number > 20.
  $\forall_{move} \neg valid(move) = \forall_{move} result(move) > 20 = \forall_{move} \neg (result(move) \leq 20)$.
- A valid move is a move that results in a number <= 20.



4A. 5 (Tree) OR 4 (Graph) -- UPDATED

Terminal nodes reached within the cutoff: 20 (A wins), 17 (draw) from 13, 17 (draw) from 12+5, 19 (draw) from 15, and 19 (draw) from 14+5.

Updated:

In the lecture slides, we have clearly defined what terminal states and terminal nodes are:

- “End State (Terminal State): A state where the game ends, with no further moves possible.”
- “The outcome of the game (such as monetary payoff, win, loss, or draw) is displayed beneath each terminal node (end state).”

We have also explicitly stated that the cutoff is performed on mid-game states. If the state in node “16 (-1)” is a terminal state, there is no need to apply the cutoff to that state and use evaluation function which uses heuristic instead of utility function. The heuristic value coinciding with the actual utility value has nothing to do with the definition of terminal nodes.

Regarding the definition of a valid move, we have made it clear above in the original solution.

Regarding the game tree, we have explicitly stated in the context that "The game tree is tree (a child can only have one parent).". However, we will be lenient and accept a graph representation with four terminal nodes as an answer.

4B. 2

Without cutoff, only the branch through 16 continues: 16 (depth 4, non-terminal) → 18 (depth 5) → 20 (depth 6, terminal). So there are 2 nodes beyond depth 4.

4C. 6

The deepest terminal path without cutoff is 5 → 10 → 12 → 14 → 16 → 18 → 20, which reaches depth 6.

4D. C. Game is draw

If A goes to 9, B will optimally choose +4 → 13 (since x2 → 18 lets A win immediately), leading to 13 → 17 where B has no moves, so draw.

4E. D. A: +4 → B: x2

- A: x2 → B: +5: B would avoid +5 (to 15) as it gives A a safe draw; optimal is +2 → 12 leading to draw if A plays optimally.
- A: x2 → B: +2: Leads to 12 where A can choose +5 → 17 for a draw.

- A: x2 → B: +2 → A: +2: This line actually lets B eventually win (ends in 20 for B), so not an A win.
- A: +4 → B: x2: This leads to Player A victory.

Updated:

The question asks for the sequence of moves presented, which would lead to player A's victory if both players play optimally after that.

A different interpretation for the phrase "in the subsequent turns" was suggested in the forum. Let us inspect the question statement: "Which sequences (from the first turn) guarantee a Player A victory if both players play optimally in the subsequent turns?"

The parenthesis allows for two interpretations:

1. "sequences that start from the first turn and that guarantee [...] in subsequent […]"
2. "sequences that guarantee from the first turn that […] in subsequent […]"

Interpretation 2. is wrong, or less likely to be correct from a test-takers point of view, in several aspects. First, it is undefined what is the starting point of the given sequences. Second, in the question, "first turn" is written inside the parenthesis, hence it is semantically not correct for "subsequent" to refer to the first turn. Third, it does not make sense to introduce "sequences" if the question only cares about the first move. Fourth, if one arrived at the correct game tree, then one will notice that there is a sequence of depth 3 that leads to a Player A victory. It should then become immediately clear that Option D is a sequence leading towards that victory sequence.

Overall, the question is a test if the correct game tree was found, and if the suboptimal winning path was identified.

4F. D. All actions result in a draw or A,B,D -- UPDATED

From 5, both A's options (+4 → 9 or x2 → 10) lead to positions where, with optimal play, the result is a draw.

Updated:

We will allow two answers:

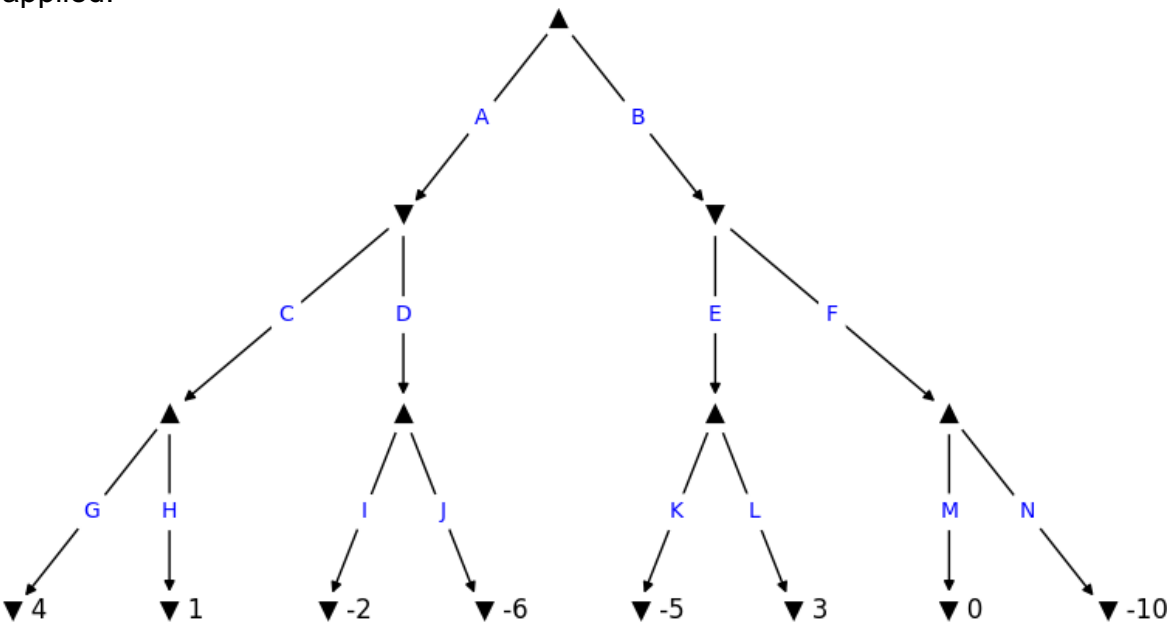
- D. All actions result in a draw. Thus, the agent does not have any action that it should take.
- A,B,D. The agent is indifferent to both actions (A and B), thus can select both, and at the same time, it's aware that both actions result in a draw.

Note that the question assumes that the opponent plays optimally in the subsequent turns; thus, it doesn't matter whether the action leads to a higher or lower probability of winning.

Alpha-Beta Pruning

Context

Consider the following game tree for a two-player game where alpha-beta pruning is applied.



Note: The symbol ▲ represents the max player's turn, while ▼ indicates the min player's turn.
Answer question 4G.

Questions

4G. [4 marks] Suppose we traverse this tree using (depth-first) alpha-beta pruning from **right to left**. Select **all** the link(s) that would be pruned by alpha-beta pruning algorithm. Select only the links that are directly pruned and not those that are indirectly pruned because they are in a subtree of a pruned link.

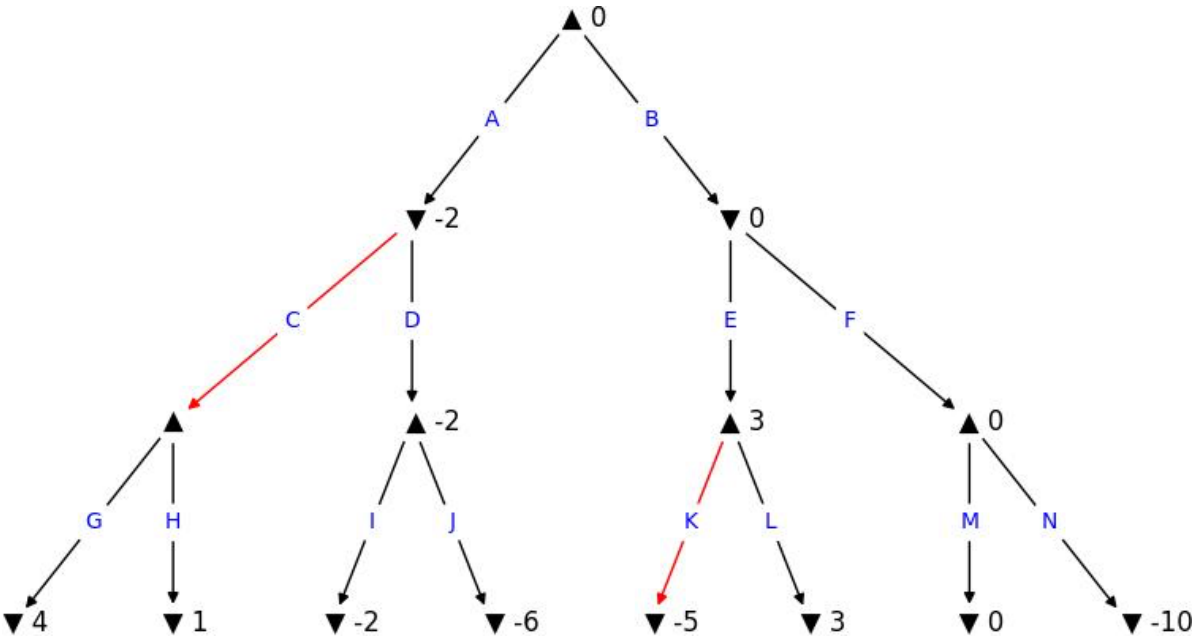
Which of the following link(s) is/are pruned? Select all that is/are true.

A. A
B. B
C. C
D. D
E. E
F. F
G. G
H. H
I. I
J. J
K. K
L. L

M. M
N. N
O. None of the above

Solution

4G. C and K



Part 5: Decision trees

You have been tasked with finding a restaurant for a conference. Leveraging your expertise, you create a decision tree based on data from past conferences given in Table 1.

	Restaurant Price (per pax)	Review of restaurant	Distance to conference center	Restaurant decision
1	100-200	Average	Long	No
2	100-200	Very good	Short	No
3	100-200	Very good	Long	Yes
4	20-100	Average	Short	Yes
5	20-100	Very good	Short	Yes
6	20-100	Very good	Long	Yes

Consider the decision tree learning algorithm using information gain. In the event of a tie in information gain, the priority for selecting attributes to construct the tree is as follows: Restaurant Price (most preferred), followed by Distance to conference center, and finally Review of restaurant (least preferred). In a case of a tie, return “Undecided”. The default value at the initial DTL call is “No”.

Based on this description, answer questions 5A-5D.

Notes:

- The entropy for a given probability distribution p_i, for $i = 1, \ldots, n$ is given by:

$$Entropy = -\sum_{i=1}^{n} p_i \log_2(p_i).$$

- The conditional entropy of a random variable Y given X is given by:

$$H(Y|X) = \sum_{x \in X} P(X = x) H(Y|X = x).$$

- Remember the log is in base 2!
- Values for the log: $\log_2(1) = 0$; $\log_2(2) = 1$; $\log_2(3) = 1.585$; $\log_2(4) = 2$; $\log_2(5) = 2.322$; $\log_2(6) = 2.585$; $\log_2(7) = 2.807$; $\log_2(8) = 3$; $\log_2(9) = 3.170$; $\log_2(10) = 3.322$.
- For rounding to two decimal places, round to the closest number with two decimal places. For example, 1.234 is rounded to 1.23 and 1.237 is rounded to 1.24.

Questions

5A: [1 mark] What is the entropy of the Restaurant Decision (yes/no) in the table, rounded to two decimal places?

5B: [1 mark] What is the information gain of selecting "Price" as the root node of the Restaurant Decision (yes/no) in the table, rounded to two decimal places?

5C: [1 mark] What is the information gain of selecting "Review" as the root node of the Restaurant Decision (yes/no) in the table, rounded to two decimal places?

5D: [1 mark] What is the information gain of selecting "Distance" as the root node of the Restaurant Decision (yes/no) in the table, rounded to two decimal places?

5E: [2 marks] Build the decision tree given the context information. What is/are the last attribute(s) in the longest path(s) of the decision tree? Select all that applies.

A. Price.
B. Review.
C. Distance.
D. None of the above.

5F: [1 mark] According to the decision tree built in Question 5E, consider the Restaurant Decision for the following restaurants.

Select the restaurant(s) with a Yes decision. Select all that apply.

A. Price: 100-200, Review: Average, and Distance: Short.
B. Price: 20-100, Review: Average, and Distance: Long.
C. None of the above.

Solution

5A. 0.92

Explanation: $-\frac{4}{6}\log_2\left(\frac{4}{6}\right)-\frac{2}{6}\log_2\left(\frac{2}{6}\right)=0.918$.

5B. 0.46

The conditional entropy when splitting according to price is *H(Y|Price)*= $\frac{3}{6}H(1,0)+\frac{3}{6}H\left(\frac{2}{3},\frac{1}{3}\right)$ =0.4575. Hence the information gain is $0.918-0.457=0.461$.

5C. 0.04

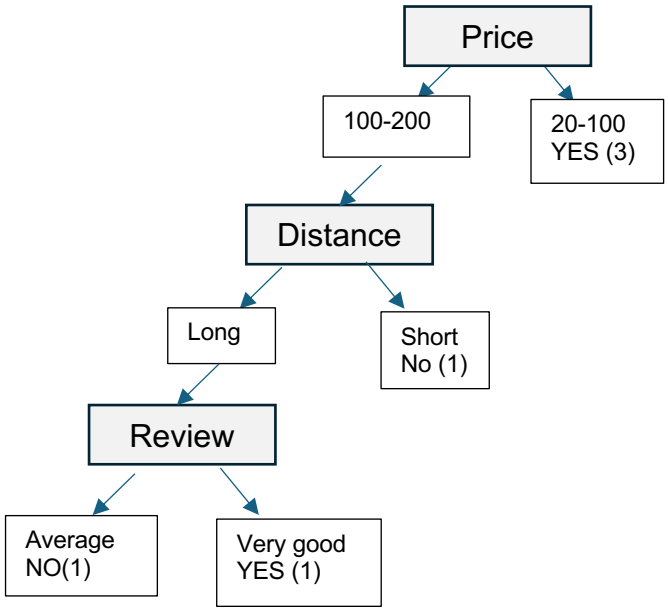
The conditional entropy when splitting according to Review is *H(Y|Review)*= $\frac{2}{6}H\left(\frac{1}{2},\frac{1}{2}\right)+\frac{4}{6}H\left(\frac{1}{4},\frac{3}{4}\right)$ =0.8742. Hence the information gain is $0.918-0.874=0.044$.

5D. 0.00

The conditional entropy when splitting according to Distance is *H(Y|Dist)*= $\frac{3}{6}H\left(\frac{1}{3},\frac{2}{3}\right)+\frac{3}{6}H\left(\frac{2}{3},\frac{1}{3}\right)$ =0.915. Hence the information gain is $0.918-0.915=0.003$.

5E. B

The decision tree is as follows:



5F. B

A. Price: 100-200, Review: Average, and Distance: Short. This example leads to No.
B. Price: 20-100, Review: Average, and Distance: Long. This example leads to Yes.

Part 6: Linear regression and logistic regression.

Context for Questions 6A and 6B

Consider the linear regression model $h_w(x) = w_0 + w_1 x_1$ where $w = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$. Consider the MSE loss function $J_{MSE}(w) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_w\left(x^{(i)}\right) - y^{(i)}\right)^2$.
Consider the following training data.

Training point	x_1	y
$x^{(1)}$	0	0
$x^{(2)}$	2	1
$x^{(3)}$	3	2

Context for Questions 6C and 6D

Consider the following customer data. The customers are described by the following features and a target variable.

x_1	x_2	x_3	y
0.3	0.9	1000	$y^{(1)}$
0.2	0.6	3400	$y^{(2)}$
0.7	2.1	500	$y^{(3)}$
0.4	1.2	1900	$y^{(4)}$

The values $y^{(1)}, \ldots, y^{(4)}$ are real numbers.

Questions

6A: [2 marks] Using the given data, evaluate the hypothesis and the loss function. Fill in the following blanks:

$h_w\left(x^{(1)}\right)$ $[BLANK1]$
$h_w\left(x^{(2)}\right)$ $[BLANK2]$
$h_w\left(x^{(3)}\right)$ $[BLANK3]$
$J_{MSE}(w)$ $[BLANK4]$

6B: [2 marks] Perform a single update to the vector $\begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$ using the **stochastic gradient descent (SGD)** update rule with the learning rate $\gamma = 1/3$. The SGD algorithm uses a random number generator that generates a random integer from 1 to n. Assume that for this update, the random generator outputs 2. Fill into the blank the resulting weights $\begin{bmatrix} w_0{}' \\ w_1{}' \end{bmatrix}$ as follows: $\begin{bmatrix} BLANK1 \\ BLANK2 \end{bmatrix}$

6C: [2 marks] Consider the data given in the Context for this question. You consider two sets of features, $x = [x_1, x_2, x_3]^T$ and $x' = [x_1, x_3]^T$. You have asked two friends to train linear regression models using gradient descent. They both use appropriate learning rates and number of steps until no more change of weights are observed. Friend A returns with results h^A and $h^{A\prime}$, corresponding to the use of x and x', respectively. Friend B returns with results

h^B and $h^{B\prime}$, corresponding to the use of x and x', respectively. Select correct statements. Select all that apply.

A. $J_{MSE}(h^A) = J_{MSE}(h^B)$ with respect to the given data in the Context.
B. $J_{MSE}(h^{A'}) = J_{MSE}(h^{B\prime})$ with respect to the given data in the Context.
C. The weights of h^A are equal to the weights of h^B.
D. The weights of $h^{A\prime}$ are equal to the weights of $h^{B\prime}$.
E. None of the above.

6D: [2 marks] Suggest changes to improve the learning stability on the data set in the Context using all the features. Select all that apply.

A. Subtract the feature standard deviation from x_3 and divide by the feature mean.
B. Subtract the feature mean from x_3 and divide by the standard deviation.
C. Min-max scaling of feature x_3
D. Fix the gradient descent learning rate to a single value.
E. Use the normal equation to obtain the weights.
F. None of the above.

6E: [2 marks] We would like to construct a machine learning model to predict the chance of a thunderstorm y in the next hour. Let there be three attributes $x = [x_1, x_2, x_3]^T$ used for prediction. In addition, it was observed that when $x_1 = x_2 = x_3 = 0$, the chance of a thunderstorm was greater than 50%. Which of the following model(s) is/are correct? Hint: consider conceptual correctness as well. Select all that apply.

A. $p(y|x) = \sigma(w_1x_1 + w_2x_2 + w_3x_3)$.
B. $p(x|y) = \sigma(w_1x_1 + w_2x_2 + w_3x_3)$.
C. $p(y|x) = w_0 + w_1x_1 + w_2x_2 + w_3x_3$.
D. $p(x|y) = w_0 + w_1x_1 + w_2x_2 + w_3x_3$.
E. $p(y|x) = \sigma(w_0 + w_1x_1 + w_2x_2 + w_3x_3)$.
F. $p(x|y) = \sigma(w_0 + w_1x_1 + w_2x_2 + w_3x_3)$.
G. $p(y|x) = w_1x_1 + w_2x_2 + w_3x_3$.
H. $p(x|y) = w_1x_1 + w_2x_2 + w_3x_3$.
I. None of the above.

6F [2 marks]

Let $X = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$ and $Y = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$. Find the best weights using the normal equation. Fill in the blanks with the results w = [Blank1, Blank2]^T.

Hint: Invert a diagonal matrix by inverting the diagonal entries, i.e., the i-th diagonal entry di becomes 1/di.

Solution

6A.

$h_w\left(x^{(1)}\right) = 2 + 0$

$h_w(x^{(2)}) = 2 + 1 * 2 = 4$
$h_w(x^{(3)}) = 2 + 1 * 3 = 5$
$J_{MSE}(w) = \frac{1}{6}[2^2 + 3^2 + 3^2] = \frac{11}{3}$

6B.

$h_w(x^{(2)}) - y^{(2)} = 4 - 1 = 3.$
$\begin{bmatrix} w_0' \\ w_1' \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} - \gamma \begin{bmatrix} g_0 \\ g_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} - \frac{1}{3}\left(h(x^{(2)}) - y^{(2)}\right)\begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} - \frac{1}{3}(3)\begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$

6C. ABD.

The set of features $x = [x_1, x_2, x_3]^T$ contains the linearly dependent feature $x_2 = 3 * x_1$. The loss function J_{MSE} will be convex in the corresponding weights w when features x are used and strictly convex in the corresponding weights w' when features x' are used. For a convex function, a local minimum is a global minimum, i.e., there can be multiple equivalent minima. For a strictly convex function, there is a single global minimum. Training with features x can lead to different weights w_A or w_B, depending on the initialization of gradient descent. Both will minimize the loss function, achieving the same value $J_{MSE}(h^A) = J_{MSE}(h^B)$.

Note that they both use appropriate learning rates and number of steps until no more change of weights are observed, hence the theorem conditions mentioned in lecture are satisfied.

6D. BC or BCE -- UPDATED

For learning stability, we notice that x_3 has a greatly different scale than the other features. Option A is non-sensical, while B is the standardization of feature x_3, and C the appropriate min-max scaling of feature x_3. Regarding D, a *variable* learning rate may help, say different learning rates for the different features.

Updated:
Option E does not work when we use the matrix inverse since we have the linearly dependent feature which makes the X^TX matrix in the normal equation not invertible. However, using the pseudoinverse, which was hinted in tutorial, can make the normal equation work. Hence, we allow both selections for full marks.

6E. E

From the question it is evident that logistic regression is required, since “chance of thunderstorm” is equivalent to “probability of thunderstorm”. In addition, it was observed that at $x_1 = x_2 = x_3 = 0$ the chance of a thunderstorm was greater than 50%. Hence, we require an offset, since $\sigma(0) = 0.5$. Lastly, notice that logistic regression expresses the conditional probability of a thunderstorm given that an observation x is made. The correct mathematical expression for this conditional probability is $p(y|x)$ and not $p(x|y)$.

6F. 5, 3

First, compute $X^TX = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$. Then invert, by inverting the diagonals, $(X^TX)^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1/4 \end{bmatrix}$.
Also, $X^TY = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} 5 \\ 6 \end{bmatrix} = \begin{bmatrix} 5 \\ 12 \end{bmatrix}$. Thus, $(X^TX)^{-1}X^TY = \begin{bmatrix} 1 & 0 \\ 0 & 1/4 \end{bmatrix}\begin{bmatrix} 5 \\ 12 \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \end{bmatrix}$.