Q7. [12 pts] Games and ML: BeatBlue

Pacman wants to design an agent that can play chess and beat his older (more successful) brother, DeepBlue.

First, Pacman would like to design a machine learning algorithm that takes in a board state and outputs a real number between 0.00 (for states where Pacman is losing) and 1.00 (for states where Pacman is winning).

(a) [2 pts] Pacman starts by asking experts to manually assign values to board states. Select all true statements.

- [x] We can use the manually-assigned values as our training dataset.
- [x] We can use the manually-assigned values as our testing dataset.
- [ ] Since we have values assigned by experts, the machine learning algorithm provides no additional benefit.
- ○ None of the above

First two options are true, because we use hand-labeled data for both training and testing.

Third option is false, because the machine learning algorithm might be able to help us assign values to states that we've never seen before, and we know from class that the state space of a game like chess is so large that it would be impossible to enumerate and manually assign values to every possible board state.

(b) [1 pt] Pacman suggests using a perceptron for this problem. Is it reasonable to use a perceptron for this problem?

- ○ Yes, because this is a classification problem.
- ○ Yes, because the training data is linearly separable.
- ● No, because this is a regression problem, and perceptrons output discrete classes, not continuous numbers.
- ○ No, because perceptrons should never be used when the data is not linearly separable.

The last option is false because perceptrons can still be used when data is not linearly separable; they just wouldn't have perfect training accuracy.

Pacman decides to represent the chess board state as a 64-dimensional vector. Pacman designs a fully-connected, feed-forward neural network with the following architecture:

$$h = \text{ReLU}(x \cdot W_1 + b_1)$$
$$\hat{y} = \text{ReLU}(h \cdot W_2 + b_2)$$

This network takes in a 1×64 vector, x, and outputs a scalar real number, $\hat{y}$.

Pacman sets the hidden layer size to be 128. In other words, h has dimensions 1×128.

(c) [1 pt] What are the dimensions of W_1?

○ 1×1	○ 128×1	○ 8192×1	○ 128×128
○ 64×1	○ 192×1	○ 64×64	● 64×128

From the question: x has dimension 1×64, and h has dimension 1×128.

In order to make the dimensions line up, W_1 must have dimension 64×128, so that we have $(1, 64) \times (64, 128) \rightarrow (1, 128)$.

Intuitively, the 64×128 weight array is converting our 64-dimensional input vector into a 128-dimensional hidden layer vector. In other words, if you think of the hidden layer output as the output of 128 different perceptrons, then the weight array contains 128 different weight vectors, where each weight vector is 64-dimensional.

(d) [1 pt] What are the dimensions of W_2?

- ○ 1×1
- ● 128×1
- ○ 8192×1
- ○ 128×128
- ○ 64×1
- ○ 192×1
- ○ 64×64
- ○ 64×128

From the question: h has dimension 1×128, and $\hat{y}$ is a scalar, so it has dimension 1×1

In order to make the dimensions line up, W_2 must have dimension 128×1, so that we have $(1, 128) \times (128, 1) \rightarrow (1, 1)$.

Intuitively, this layer is a single neuron taking the 128-dimensional hidden layer vector and outputting a single scalar as output. This neuron requires a 128-dimensional weight vector, so that we can take a dot product between the weight vector and hidden layer vector.

(e) [1 pt] Pacman considers changing the second layer of the neural network. Which of these proposed changes is best for this problem?

Reminders: $\text{ReLU}(x) = \max(x, 0)$ $\sigma(x) = \frac{1}{1+e^{-x}}$ $\text{sgn}(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$

- ○ $\hat{y} = \text{ReLU}(h \cdot W_2 + b_2)$
- ○ $\hat{y} = \text{sgn}(h \cdot W_2 + b_2)$
- ● $\hat{y} = \sigma(h \cdot W_2 + b_2)$
- ○ $\hat{y} = h \cdot W_2 + b_2$

The output of the machine learning algorithm needs to be a real number between 0 and 1.

The sigmoid function is the only function in the answer choices that always outputs a real number between 0 and 1.

ReLU could output positive numbers greater than 1. The sgn function could output -1. Without any non-linearity at the last layer, the outputs could be outside of the range of 0 to 1.

Pacman now has a neural network $\hat{y} = f(x)$ that takes in board states (x) and outputs values ($\hat{y}$), and would like to use the network to select actions in the chess game.

(f) [1 pt] Let $s' = g(s, a)$ represent the successor function that takes in a state s and action a, and outputs a successor state s'.

Which of the following expressions represents a reflex agent's optimal action from state s, based on the values outputted by the neural network?

- ○ $\arg\max_s f(s)$
- ○ $\max_a g(s, a)$
- ● $\arg\max_a f(g(s, a))$
- ○ $\sum_a f(g(s, a))$

For every action a, generate a successor state $g(s, a)$, then use f to evaluate this successor state. Use argmax to pick the action that led to the successor state with the highest value.

Finally, Pacman decides to run depth-limited minimax search and use the neural network as an evaluation function.

(g) [1 pt] Is it reasonable to use the neural network as an evaluation function?

- ● Yes, because the neural network maps states to real-numbered values.
- ○ Yes, because the network is guaranteed to correctly identify terminal states where the game is over.
- ○ No, because it would be more efficient and accurate to expand the entire minimax tree to the terminal states.
- ○ No, because evaluation functions should map states to actions.

The other Yes option is false because depending on the weights in the neural network, it might not correctly identify terminal states. There is also no requirement that evaluation functions need to correctly identify terminal states.

(h) [1 pt] Is it possible to use alpha-beta pruning in this problem?

- ● Yes, pruning works even if the neural network output was unbounded.
- ○ Yes, but only because the neural network only outputs numbers between 0 and 1.
- ○ No, because pruning requires you to know the values at all leaf nodes in advance.
- ○ No, because the values outputted by the neural network are not guaranteed to be correct.

In this question, we're running standard minimax (with a specific evaluation function learned from a neural network), so standard alpha-beta pruning works as well.

(i) [1 pt] Is it better to spend more time on training the neural network, or expanding more layers of the game tree?

- ○ Training the neural network
- ○ Expanding the game tree
- ● Not enough information

This depends on how accurate the neural network is, and in practice, it would probably require empirical studies to determine where the time is better spent.

(j) [2 pts] Recall that in Monte Carlo tree search, we allocate more rollouts to game states that are more promising (i.e. better utility for Pacman).

Inspired by this idea, Pacman considers running gradient descent for a longer time when using the neural network to evaluate game states that are more promising. Would this idea work?

- ○ Yes, because this causes the neural network to focus its training on promising game states.
- ○ Yes, because training for a longer time leads to better evaluations.
- ○ No, because running gradient descent for too long always leads to overfitting.
- ● No, because gradient descent runs during training, not evaluation.

The third option is false because running gradient descent for too long doesn't necessarily lead to overfitting.

Q2. [15 pts] Search & Games: Shelving Books

At the library, Carolyn plans to place 36 books on a bookshelf. The bookshelf consists of 6 rows, labeled Row 0 to Row 5, and each row can hold exactly 6 books.

The 36 books are labeled from b_0 to b_{35}. Furthermore, each spot on the bookshelf is labeled from 0 to 35. Here is a diagram of what the bookshelf looks like with all the books in their correct positions:

Row 0	b_0	b_1	b_2	b_3	b_4	b_5
Row 1	b_6	b_7	b_8	b_9	b_{10}	b_{11}
Row 2	b_{12}	b_{13}	b_{14}	b_{15}	b_{16}	b_{17}
Row 3	b_{18}	b_{19}	b_{20}	b_{21}	b_{22}	b_{23}
Row 4	b_{24}	b_{25}	b_{26}	b_{27}	b_{28}	b_{29}
Row 5	b_{30}	b_{31}	b_{32}	b_{33}	b_{34}	b_{35}

Carolyn first places all 36 books onto the bookshelf in a random order.

Carolyn can move books around using these three actions (each action costs 1):

1. Shift all books in any row to the right by 1, with the rightmost book moving to the first spot on that row.
   - For instance, using this action on Row 3, $\{b_{18}, b_{19}, b_{20}, b_{21}, b_{22}, b_{23}\}$, turns the row into $\{b_{23}, b_{18}, b_{19}, b_{20}, b_{21}, b_{22}\}$.
2. Shift all books in any row to the left by 1, with the leftmost book moving to the last spot on that row.
   - For instance, using this action on Row 4, $\{b_{24}, b_{25}, b_{26}, b_{27}, b_{28}, b_{29}\}$, turns the row into $\{b_{25}, b_{26}, b_{27}, b_{28}, b_{29}, b_{24}\}$.
3. Swap any two books that are on **different** rows.
   - For instance, the books at positions 6 and 22 can be swapped, and so can the books at positions 25 and 31, but the books at positions 24 and 26 **cannot** be swapped.

Carolyn's goal is to **sort** the books by putting all the books in their correct positions. In other words, for $0 \leq i \leq 35$, Carolyn wants book b_i to be placed on the ith position on the bookshelf.

Carolyn decides to model this as a search problem.

(a) [2 pts] Carolyn claims that the size of the state space for this problem is 2^{36}, while a coworker Julia claims that it is 36! instead. Who is correct, and why? Explain your choice in two sentences or fewer.

○ Carolyn is correct ● Julia is correct ○ Neither are correct

Julia is correct, every book's specific position must be known in order to perform the right actions.

(b) [4 pts] Derive the maximum branching factor for this problem. Your answer must written as a single integer.

Hint 1: How many ways are there to swap two books on different rows? How many possible ways are there to shift any of the rows?

Hint 2: The choose function, $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, may be useful here (though you can also solve this question without it).

552

One way to go about this problem is by first accounting for all possible swaps. The way this is done is by computing all possible swaps between two books, $\binom{36}{2} = 630$, and then subtracting all the ways we can swap two books from the same row, $6 \cdot \binom{6}{2} = 90$, which will give us all possible swaps between two books on **different** rows, resulting in 540. Then, we add 12, accounting for all the shifts that can be done (one forward and one backward shift for each of the 6 rows), resulting in 552.
Furthermore, you could also approach this problem by choosing any book, so 36 choices, and then choosing any of the 30 books that book can successfully swap with (because we don't want our book to swap with a book from the same row), so 1080. We then divide by 2 to prevent double-counting, and adding 12 for the same reason as before yields 552.

(c) [1 pt] Another coworker Nawoda claims that the following goal test is valid for this search problem.

(Reminder: the goal state is when every book is at its correct position on the bookshelf.)

- For the leftmost book b_i of every row, book b_{i+1} is to its direct right.
- For the rightmost book b_i of every row, book b_{i-1} is to its direct left.
- For each book b_i on **neither** end of a row, book b_{i-1} is to its direct left, and book b_{i+1} is to its direct right.

○ This is a valid goal test.

● This is **not** a valid goal test.

Consider the case in which almost every book is sorted, but two complete rows of books are swapped. In the goal test stated, it will incorrectly say that we have reached the goal state.

(d) [3 pts] Which of the following heuristics are admissible for this search problem? Select all that apply.

☐ The minimum number of **swaps** needed to sort all the books, assuming you can swap **any** two books, regardless of whether they are on the same row.

☐ The number of books that are **not** in their correct position.

☐ For some book b_i, define dist(b_i) to be the number of rows plus the number of columns book b_i is from its correct position. The heuristic is max(dist(b_i)) for all books b.

● None of the above.

The first heuristic is inadmissible, as we can consider the case in which almost all the books are at their correct position, but the first row's books are all shifted one to the right of where they're supposed to be. The minimum number of swaps would give a cost larger than the true cost, which would just be shifting the books in the first row to the left once.
The second heuristic is also inadmissible. We can picture the exact case talked about above. If one row is shifted to the right and everything else is correct, the number of books that aren't in their correct shelf position would be an overestimate of the true cost (just 1, shift all the books in that row to the left by 1).
Finally, the third is inadmissible. Manhattan distance is irrelevant here, as you can swap two books from different rows that are really far away from each other at only the cost of 1, so the distance a book is from where it's supposed to be does not matter for this search problem.

Now consider the following game: Julia and Carolyn take turns making moves on the same bookshelf until all the books are in their correct position. The person who makes the move that results in the bookshelf being completely sorted wins. Assume both players are playing optimally.

(e) [2 pts] Julia and Carolyn decide to model this game with a game tree. Select all true statements.

■ This game tree is infinite in depth.

☐ This game tree requires expectation nodes.

■ This is a zero-sum game.

■ Every non-terminal node's branching factor will be equal to the game's maximum branching factor.

○ None of the above.

This game tree is infinite in depth because since both players are acting optimally, no player will ever make a move that causes the bookshelf to be ONE move away from being completely sorted, because then the other player automatically wins the game.
The game tree does NOT require expectation nodes, as minimizer, maximizer, and terminal nodes are all we need.
This indeed is a zero-sum game, as each player's gain is directly equivalent to the other player's loss.
Since there are no state-dependent restrictions on what moves can be played, we can perform any of the possible actions on any of the non-terminal states.

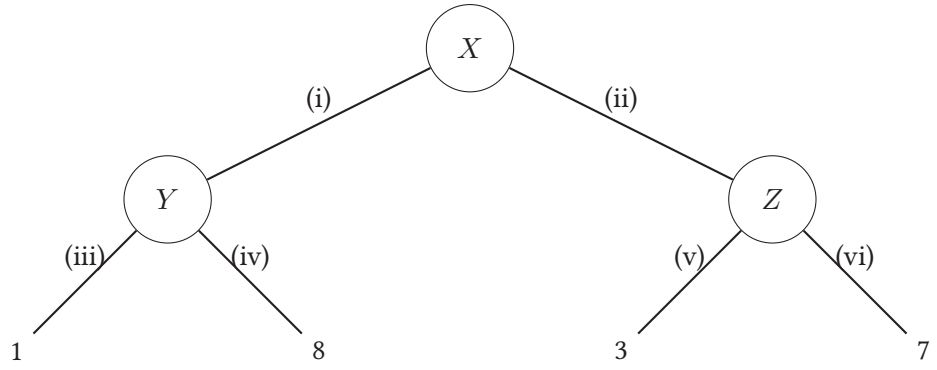
(f) [3 pts] Julia is trying to maximize her utility. Fill in the boxes below with integer values such that $f(s)$ is an evaluation function that causes Julia to act optimally.

$$f(s) = \begin{cases} \boxed{-1} & \text{if state } s \text{ is } \boxed{1} \text{ move(s) away from a state where all the books are sorted.} \\ \boxed{0} & \text{otherwise.} \end{cases}$$

The main idea for this problem is that since Carolyn is playing optimally, Julia should NEVER move into a state that is one move away from being a terminal state, because then Carolyn will make the move that causes the bookshelf to be completely sorted. Therefore, we want our evaluation function to equate to some value a when we are at a state that is one move away from being terminal, and some other value b otherwise, where $a < b$. Therefore, -1 and 0 could both work, but all we need is that the first value is strictly less than the second.

Q3 *Friend or Foe?* **(13 points)**

Robotron is a latest-generation robot. Robotron is loaded into a world where Robotron chooses one action, then a human chooses one action, then Robotron receives a reward. We can model this as a game tree, shown below.



At node X, Robotron selects action (i) or (ii). At node Y, the human selects action (iii) or (iv). At node Z, the human selects action (v) or (vi). The circles do not necessarily represent chance nodes.

Q3.1 (1 point) Suppose the human acts adversarially, and Robotron knows this. If Robotron acts optimally, what reward will Robotron receive?

○ (A) 1 ○ (B) 8 ● (C) 3 ○ (D) 7 ○ (E) 4.5 ○ (F) 5

Solution: This is the minimax outcome of this game tree. We use minimax here because the human is acting adversarially.

Q3.2 (1 point) Suppose the human acts cooperatively with Robotron to maximize Robotron's reward, and Robotron knows this. If Robotron acts optimally, what reward will Robotron receive?

○ (A) 1 ● (B) 8 ○ (C) 3 ○ (D) 7 ○ (E) 4.5 ○ (F) 5

Solution: Because the human is acting cooperatively, the human will choose action (iv) from Y to get reward 8, and action (vi) from Z to get reward 7. Robotron will choose action (i) to reach Y and maximize its reward.
Sometimes this is called MaxiMax, because both agents are maximizing agents trying to maximize the value of the game.

Q3.3 (3 points) For the rest of the question, suppose Robotron doesn't know the human's behavior. Robotron knows that with probability p, the human will act adversarially, and with probability $1-p$, the human will act cooperatively. For what p will Robotron be indifferent about Robotron's choice of action?

○ (A) $1/8$ ○ (B) $1/6$ ● (C) $1/3$ ○ (D) $1/2$ ○ (E) $2/3$ ○ (F) $5/6$

Solution: The expected utility of taking action (i) is $1p + (1-p)8$. The expected utility of taking action (ii) is $3p + (1-p)7$. Set the two equal to each other and solve to see when the robot finds the expected utility of each action to be equal.

The problem above (where Robotron doesn't know the human's behavior) can also be modeled as an MDP.
In this MDP, T_a refers to a terminal state (no more actions or rewards are available from that state) when the human is adversarial, and T_c refers to a terminal state when the human is cooperative.
For the rest of the question, fill in the blanks in the table, or mark "Invalid" if the provided row does not belong in the MDP's transition function and reward function. Not all rows of the table are shown.

s	a	s'	$T(s,a,s')$	$R(s,a,s')$
X	(i)	Q3.4	p	Q3.5
X	Q3.6	Q3.7	Q3.8	7
Y	(iii)	T_a	p	Q3.9

Q3.4 (1 point) Q3.4

○ (A) Y ○ (F) (ii) ○ (K) p ○ (P) 7

○ (B) Z ○ (G) (iii) ○ (L) $1-p$ ○ (Q) 4.5

● (C) T_a ○ (H) (iv) ○ (M) 1 ○ (R) 5

○ (D) T_c ○ (I) (v) ○ (N) 8 ○ (S) 0

○ (E) (i) ○ (J) (vi) ○ (O) 3 ○ (T) Invalid

Solution: Robotron only gets to choose one action, so after taking that action, Robotron immediately transitions to the terminal state, and since transition probability is p, we know human is adversarial so terminal state is T_a.