CS 182 FALL 2022, PROBLEM SET 3

Due: November 7, 2022 11:59pm

This problem set covers Lectures 10 through 14. The topics include AI game playing, Stackelberg security games, social choice, Bayesian networks, and hidden Markov models.

1. *Stackelberg Security Game.* (10 points) Consider a 2-player game in normal form. Let (x_1, x_2) be a (possibly mixed) Nash equilibrium. Show that if $(x_1^\star, s_2)$ is a strong Stackelberg equilibrium (where player 1 is the leader) then $u_1(x_1^\star, s_2) \geq u_1(x_1, x_2)$.

2. *Social Choice.* (20 Points) Recall that a voting rule is Condorcet consistent if the rule selects an alternative that is a Condorcet winner whenever there exists a Condorcet winner in the given preference profile. As in class we denote the set of voters by $N = \{1, \ldots, n\}$ and the set of alternatives by A, where $|A| = m$.

(1) (10 points) The minimax voting rule (not to be confused with the minimax algorithm) picks an alternative that minimizes the maximum number of voters by which another alternative beats this one, i.e.,

$$\operatorname*{argmin}_{x \in A} \max_{y \in A \setminus \{x\}} \text{Score}(y, x),$$

where $\text{Score}(y, x) = \sum_{i=1}^{n} \mathbb{1}[y \succ_i x]$ counts the number of voters who prefer y over x, with $\succ_i$ denoting preference according to the ranking of voter i. For example, in the profile on slide 5 of Lecture 11, $\text{Score}(b, c) = 4$, as there are 4 voters who prefer b to c. The minimax rule would choose b, as $\max_{y \in A \setminus \{b\}} \text{Score}(y, b) = \text{Score}(a, b) = 2$.
Show that the minimax rule is Condorcet consistent. You may assume that the number of voters n is odd.

> First, suppose there is a condorcet winner, x^*, in the profile. This means that x^* beats all other alternates in a head-to-head plurality vote, so
>
> $$\text{Score}(y, x*) < \text{Score}(x^*, y)$$
>
> for all $y \in A \setminus \{x^*\}$

(2) (10 points) A *positional scoring rule* using a score vector $(s_1, \ldots, s_m)$, where $s_1 \geq \cdots \geq s_m$, is one that determines a winner by picking an alternative with maximum score, where scores are determined as follows:

- Each voter i gives s_j points to the alternative that is the jth highest on their ranking (e.g., the highest alternative in σ_i receives s_1 points and the lowest alternative σ_i receives s_m points).
- Each alternative's score is the sum of the points given over all voters.

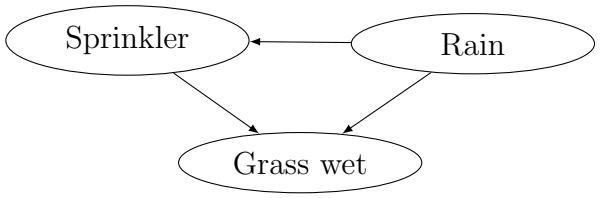
Answer the following questions about positional scoring rules.

(a) (4 points) Explain why both Plurality voting and Borda voting are examples of positional scoring rules.

(b) (6 points) Under the assumption that $s_1 > s_2 > \cdots > s_m$, prove that no positional scoring rule is Condorcet consistent.
Hint: Construct a preference profile with three alternatives and a smallish number of voters such that there is a Condorcet winner but any positional scoring rule with $s_1 > s_2 > s_3$ would select a different alternative.

3. *Bayesian Networks.* (10 points) Consider the following Bayes net:

	Sprinkler	
Rain	T	F
F	0.4	0.6
T	0.1	0.9



Rain	
T	F
0.2	0.8

		Grass wet	
Sprinkler	Rain	T	F
F	F	0.1	0.9
F	T	0.2	0.8
T	F	0.3	0.7
T	T	0.4	0.6

In each table, the value of the conditioned variables are presented on the leftmost columns.

(1) (3 points) What is $\mathbb{P}(\text{Grass wet} = \text{True} \mid \text{Rain} = \text{False})$?

By the definition of conditional probability,

$$\mathbb{P}(\text{Grass wet} = \text{True} \mid \text{Rain} = \text{False}) = \frac{\mathbb{P}(\text{Grass wet} = \text{True} \cap \text{Rain} = \text{False})}{\mathbb{P}(\text{Rain} = \text{False})}$$

The denominator is given, but the numerator can be found using LOTP

$$\mathbb{P}(\mathrm{G} \cap \mathrm{R}^c) = \mathbb{P}(\mathrm{G} \cap \mathrm{R}^c \cap \mathrm{S}) + \mathbb{P}(\mathrm{G} \cap \mathrm{R}^c \cap \mathrm{S}^c)$$

where G, R, and S are used as shorthand for Grass = True,... and so on and G^c, R^c, and S^c are shorthand for Grass = False... and so on.
By the chain rule, we have

$$\begin{aligned}\mathbb{P}(\mathrm{G} \cap \mathrm{S} \cap \mathrm{R}^c) &= \mathbb{P}(\mathrm{G} \mid \mathrm{R}^c \cap \mathrm{S}) * \mathbb{P}(\mathrm{S} \mid \mathrm{R}^c) * \mathbb{P}(\mathrm{R}^c) \\ &= 0.3 * 0.4 * 0.8 \\ &= 0.096\end{aligned}$$

Similarly, we have

$$\begin{aligned}\mathbb{P}(\mathrm{G} \cap \mathrm{S}^c \cap \mathrm{R}^c) &= \mathbb{P}(\mathrm{G} \mid \mathrm{R}^c \cap \mathrm{S}^c) * \mathbb{P}(\mathrm{S}^c \mid \mathrm{R}^c) * \mathbb{P}(\mathrm{R}^c) \\ &= 0.1 * 0.6 * 0.8 \\ &= 0.048\end{aligned}$$

Thus,

$$\begin{aligned}\mathbb{P}(\mathrm{G} \mid \mathrm{R}^c) &= \frac{0.096 + 0.048}{0.8} \\ &= 0.18\end{aligned}$$

(2) (3 points) What is $\mathbb{P}(\text{Grass wet} = \text{True})$?

Using LOTP and the chain rule we get

$$\begin{aligned}\mathbb{P}(\mathrm{G}) &= \mathbb{P}(\mathrm{G} \cap \mathrm{R} \cap \mathrm{S}) + \mathbb{P}(\mathrm{G} \cap \mathrm{R}^c \cap \mathrm{S}) + \mathbb{P}(\mathrm{G} \cap \mathrm{R} \cap \mathrm{S}^c) + \mathbb{P}(\mathrm{G} \cap \mathrm{R}^c \cap \mathrm{S}^c) \\ &= \mathbb{P}(\mathrm{G} \mid \mathrm{R} \cap \mathrm{S}) * \mathbb{P}(\mathrm{S} \mid \mathrm{R}) * \mathbb{P}(\mathrm{R}) + \mathbb{P}(\mathrm{G} \mid \mathrm{R}^c \cap \mathrm{S}) * \mathbb{P}(\mathrm{S} \mid \mathrm{R}^c) * \mathbb{P}(\mathrm{R}^c) \\ &\quad + \mathbb{P}(\mathrm{G} \mid \mathrm{R} \cap \mathrm{S}^c) * \mathbb{P}(\mathrm{S}^c \mid \mathrm{R}) * \mathbb{P}(\mathrm{R}) + \mathbb{P}(\mathrm{G} \mid \mathrm{R}^c \cap \mathrm{S}^c) * \mathbb{P}(\mathrm{S}^c \mid \mathrm{R}^c) * \mathbb{P}(\mathrm{R}^c) \\ &= 0.4 * 0.1 * 0.2 + 0.3 * 0.4 * 0.8 + 0.2 * 0.9 * 0.2 + 0.1 * 0.6 * 0.8 \\ &= 0.188\end{aligned}$$

(3) (4 points) Suppose we did likelihood weighting on this network, where we observe evidence $Sprinkler = True$. What is the weight of a sample

$$\boldsymbol{x} = (Rain = True, Sprinkler = True, GrassWet = False)$$

obtained from the WEIGHTED-SAMPLE algorithm?

In the algorithm, Sprinkler = True is set first without changing w. Then, the algorithm sees that $X_i = \text{S}$ is an evidence variable and adjusts w as $w \leftarrow w * \mathbb{P}(\text{Sprinkler} = \text{True} \mid \text{parents(Sprinkler)}) = 1 * \mathbb{P}(\text{Sprinkler} = \text{True} \mid \text{Rain} = \text{True}) = 0.1$

4. *Hidden Markov Models.* (20 Points) Alice and Bob are living in a house. Bob never leaves the house or checks the weather outside, so he never knows for sure if it is rainy or sunny (assume for simplicity these are the only two weather phenomena). However, Alice does leave the house, and her mood is determined stochastically by the status of the weather that day. Bob is able to make inferences about the weather based on Alice's mood when she gets home, so he uses a hidden markov model in which the weather describes the underlying hidden states and Alice's mood are his observed states. He knows that in his city of Markovtown,

- If it is raining today, then it will rain with probability 0.8 on the next day.
- If it is currently sunny, then it will be sunny with probability 0.6 on the next day.

He also knows that Alice's mood depends on the weather in the following process:

- If it is raining, then Alice will be in a bad mood with probability 0.7.
- If it is sunny, then Alice will be in a good mood with probability 0.6.

One day, Bob checks the weather channel, so he knows for sure that it rained. He then recorded Alice's mood the next three days (not including the day he checked the weather channel): good, good, bad.

(1) (10 points) What is the probability that it will be rainy on day 4?

(2) (10 points) Under Bob's model, what is the most likely sequence of weather states over the three days?

5. *AI Game Playing.* (35 points)*AI Game playing in Ghost.* In this problem, you will solve the game of *Ghost*[1] using alpha-beta pruning. This game is played with two players, as follows. The first player chooses an English letter, the second player adds another letter to the end of the first player's letter, the first player then adds another letter to the end of that, and so on. The key is that, at any given point in time, the growing string must have the ability to form a word, but the first player who is forced to make the string into a word loses. For example, an example game might play out as follows:

First Player: W *// Growing string: "W"*
Second Player: A *// Growing string: "WA"*
First Player: T *// Growing string: "WAT"*
Second Player: E *// Growing string: "WATE"*
First Player: R *// Game ends, "WATER" forms a word*

Note that the first player played an R at the end, so since "WATER" is a word, the first player loses.

We have modified the game in the following way: the value of a player's is the reciprocal of the length of the word in the terminal state, flipped with the appropriate sign depending on which player's turn it is. This has the effect that players aren't merely interested in winning, but they are also interested in winning quickly whenever they can guarantee their victory, and they want to make the game last as long as possible if their opponent can guarantee victory instead,
The code file is located in `pset3.py`. We have included a text file of English words in `dictionary.txt`. Note that you only have to implement the alpha-beta pruning agent, and not any of the Ghost game details. Our autograder will run your agent against ours and check that your agent wins in all the situations in which it should win. Note that the Gradescope Autograder corresponds to the point values in parts (1), (2), and (4) of this problem. **How to tiebreak: For all agents, if several actions result in states with the same value, then we pick the action that is first in our `get_actions()` list (e.g. in the Ghost game, this should just be alphabetical order). Additionally, do not make any unnecessary calls to `get_actions()` or the autograder will fail the test cases.**

(1) (5 points) Implement the `MinimaxAgent`.
(2) (10 points) `AlphaBetaAgent`.

[1]The authors of this problem are unclear whether this is the universal name for this game, but we shall go with it.

(a) (5 Points) Paul and Fiona decide to play one final game, in which Paul is attempting to maximize his score, while Fiona attempts to minimize Paul's score. Both Paul and Fiona play optimally, and the game tree is shown in the following figure. Perform the minimax algorithm with alpha-beta pruning to find the alpha and beta values passed to each node when the node is first called. Include an image with the alpha beta values at each node, as well as an indication of which branches are pruned. Assume that the algorithm visits each child from left to right. What are the actions that each player takes?

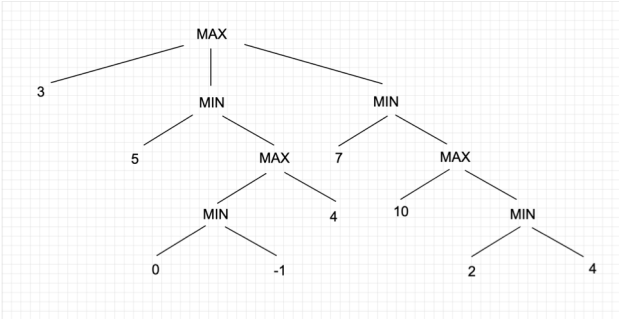


FIGURE 1. Game Tree for Problem 1

(b) (5 Points) Implement the `AlphaBetaAgent`. You should find that the number of calls to `getSuccessor` is smaller than before (which would indicate that pruning has improved our search). **You should prune on equality $\alpha = \beta$ in order for the autograder to recognize your submission.**

(3) (5 points) Recall that the minimax algorithm helps you find the optimal value against an opponent who is also playing optimally. Now suppose that you know that your opponent is not necessarily playing smart; instead, your opponent simply just picks a random letter that is a valid move. Describe the strategy that maximizes or minimizes (depending on whether you are the max or the min agent) the expected value of the terminal game state.

(4) (10 points) Implement the agent in part (3) in `OptimizedAgainstRandomAgent`. You should implement both cases, whether the agent wants to maximize or minimize the terminal value of the game, depending on the agent's index.

(5) (5 points) In `simulate_versus_random`, we compare the performance of the agent from part (4) against a random agent with the performance of a minimax agent against a random agent. We take the random agent to be the minimizer. We compute the average value that each agent wins over `k=10000` trials. By about how much does your agent from (4) beat the minimax agent on the initial game state prefixes [`beh`, `feb`, `gw`]? Explain your intuition for why the `OptimizedAgainstRandomAgent` beats the `MinimaxAgent` on average against a random agent (there is more than one possible answer to this question).

6. Collaboration, Calibration and References (5 points).

(1) With whom did you work on this problem set? What (if any) references and/or resources did you use beyond the course lecture slides and textbook?

(2) (5 points) Approximately how long did it take you to complete this problem set? Please complete this brief survey worth 5 points, graded for completion.