Team Assignment: 1
Playing Rock-Paper-Scissors using Bayesian Networks

24-787 Machine Learning & Artificial Intelligence for Engineers (Spring 2023)
https://piazza.com/cmu/spring2023/24787
OUT: Monday, February 6
DUE: Monday, February 20 @ 11:59 pm EST

Instructor: Prof. Conrad Tucker
TAs/ CAs: Ananya, Aviral, Avi, Jiacheng, Kenrick, Nischal, Sakthi, Tong, Vedant

---

Introduction

Temporal pattern recognition is fundamental to the fields of machine learning and artificial intelligence. Rock-Paper-Scissors (RPS) may seem like a trivial game, but finding an optimal strategy to beat predictable players makes it a challenging temporal pattern recognition problem. When you play RPS against the same opponent for several rounds, it is possible to predict what the opponent will do in order to gain an advantage. However, it is not possible to gain an advantage against an opponent who plays randomly.

The game has three candidate actions R (rock), P (paper) and S (scissors). A random strategy assumes a uniform categorical prior distribution over the action space (i.e., $P(R) = P(P) = P(S) = \frac{1}{3}$). You may wonder what is the the point of the RPS competition if the optimal strategy is to pick a random action? It is actually possible to do much better than a random strategy when the competition involves non-random players (eg. Humans, A.I. Bot, etc.). Human tendencies can be predictable, since a winning player will think that the action they won with will win them the game again, and a losing player will think their action is worse, therefore will most likely change strategies. When mathematically modeling rock-paper-scissors, variables such as human tendencies are not factored into deciding which action to be played against a player. In game theory, this is called as *conditional response*. In the paper titled *Social cycling and conditional responses in the Rock-Paper-Scissors game* published in *Scientific Reports*, the authors have found a strategy through a field study with a large crowd of volunteers. This strategy, coined as *win-stay, lose-shift* by the authors, improves player odds of winning a hand in RPS. In this assignment, the *win-stay, lose-shift* strategies will be used by A.I. agents.

Hopefully, you are beginning to see how conditional probabilities and Bayesian inference can be leveraged to find a winning strategy. During each round of the game, each player must ask themselves the question, "What is the conditional probability of my opponent choosing Rock, Paper, and Scissors, given the choice they made on the last round?". Let's assume that after playing many rounds of RPS with your friend, you find that every time you chose Rock in a previous move, your friend chose Paper in the following round 8 times out of 10. That means the conditional probability of you choosing Paper, given your friend chose Rock on the previous round, is 0.8. If you have identified this pattern, you would pick Scissors and win 8 times out of 10.

Win-Stay, Lose-Shift Strategy

The *win-stay, lose-shift* strategy is actually quite elegant and simple enough to understand. Let's say you play Rock during your first round against your opponent and lose. In the next round, switch to the move that beats your opponent's previous move (in this case play Scissors) as they are likely to continue using that move against you. This strategy greatly improves your odds of winning many games if you are the only person out of the two to employ it. As you can imagine, when this strategy is employed by both teams, both players win equal number of games. Instead, if you win during the first round and you think that your opponent will play the same *win-stay, lose-shift* strategy by choosing to play Paper in anticipation of you continuing to play Rock, switch your move to play Scissors. In other words, play the hand your losing opponent just played. This

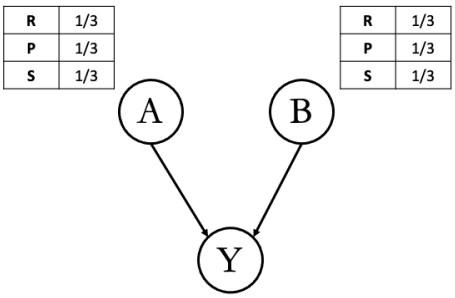
strategy is called *win-shift, lose-shift*. As part of the assignment, you will be required to program A.I. agents that use the *win-***, lose-shift* strategies to test the Bayesian network strategies that your team develops.

Designing a Bayesian Network [100 points]

Your mission, should you choose to accept is to fit Bayesian networks using empirically collected data that can outperform a non-random strategy employed by an A.I. agent or a fellow team. For this assignment, you will be provided with boiler-plate code that implements the game of RPS as well as a vanilla V-DAG Bayesian Network. With the help of the boiler-plate code, investigate the following cases.

1. **Case 1 [25 points]: Human Vs. Human**
   For this case, you will be required to play the game of RPS within your team to collect data. As a warm-up, begin by playing 10 rounds of RPS (you can choose to discard this data). Starting from the 11^{th} round, record the moves played by both the teammates and continue this process for as many rounds as you like. Then, shuffle the player pair and continue data collection. Ideally, in a team of 4, you can make up to 6 pair combinations and collect data based on the rounds played by each pair. To collect data, you can either manually play this game, modify the boiler-plate code to include an additional human player or use any other tool/software. This step is crucial in empirically finding the probability mass values to construct the conditional probability tables of your Bayesian Networks. Please ensure that you are not collecting data by playing against an algorithm if you do use any software/tool.

2. **Case 2 [45 points]: Bayesian Networks Vs. *win-stay, lose-shift* strategy**
   Using data collected in Case 1, construct the marginal and conditional probability tables (CPT) corresponding to the V-DAG Bayesian Network as shown in Fig.1 as well as the inverted V-DAG (Naive Bayes) Bayesian Network shown in Fig.2. Once you obtain the marginal and conditional probability tables, fit the V-DAG Bayesian Network [modify boiler-plate code] and the Naive Bayes network [write a python script]. Using your Bayesian network, you can perform inferences (e.g, What is the joint probability of A's move being "Rock" and B's move being "Scissors" and the predicted/next move of player$A$ (assume your team is player B) being "Paper"? What are the probabilities over the actions for Player A given the previous moves of players A and B). To use your Bayesian network to play against an opponent, you can perform a MAP (Maximum A Posteriori) Inference such that $y^* = \underset{y \in \{R,P,S\}}{\text{argmax}} \; P(Y = y | A = a, B = b)$ and choose the move that beats y^* to play against other strategies. You will need to write python functions to implement both the *win-stay, lose-shift* and *win-shift, lose-shift* strategies. Write a python function to play 30 games of RPS, with 3 rounds per game.



	A=R			A=P			A=S		
	B=R	B=P	B=S	B=R	B=P	B=S	B=R	B=P	B=S
Y=R	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3
Y=P	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3
Y=S	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3

Figure 1: A Bayesian Network in which the dependencies between random variables are represented by a V-shaped Directed Acylic Graph (V-DAG). The V-DAG factorizes as follows: $P(Y, A, B) = P(Y|A, B) \cdot P(A) \cdot P(B)$, where the random variable A denotes Player A's move in the previous round and random variable B denotes Player B's move in the previous round.

3. **Case 3 [10 points]: Team X Bayesian Network Vs. Team Z Bayesian Network**
   The face-off! For this case, your team's Bayesian Network will play against all other teams from the course. Since there are 10 teams, each team will compete against 9 other teams. The team with the most

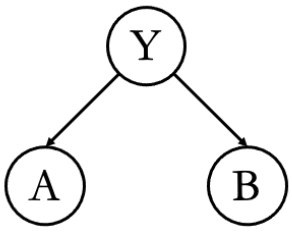


Figure 2: A Bayesian Network with Naive Bayes assumption in which the dependencies between random variables are represented by a inverted V-shaped Directed Acylic Graph (Invereted V-DAG). The Inverted V-DAG factorizes as follows: $P(Y, A, B) = P(A|Y) \cdot P(B|Y) \cdot P(Y)$, where the random variable A denotes Player A's move in the previous round and random variable B denotes Player B's move in the previous round.

wins will emerge as the winner.
[BONUS][10 points] In addition to data collected from plays between human players, employ the *win-***, lose-shift* strategies to augment your dataset. Beware, your fellow teams are also aware of these strategies and they too may augment their datasets. Feel free to use/develop any other strategies for the data augmentation.
[BONUS][10 points] Choose the Bayesian network that shows superior prediction accuracy and implement a real-time variant that constantly keeps updating its priors and posteriors while playing against an opponent.

4. **Report [20 points]:**
Write a report that contains an abstract summarizing your case study findings and results of your implementation. In addition, provide a brief description of your methodology and implementation with supporting images and tables if any in separate sections. In your results section, report dataset statistics (i.e., number of samples, number of games played, etc.), marginal and conditional probability tables for both Bayesian networks as well as the success rate against the A.I. agents and fellow team. If your team attempted any of the bonus cases, please report those findings in a separate section. Finally, discuss your findings, limitations of your models and possible techniques to improve the performance of your models. If your team has used any software/tool for data collection, please state and provide reference. The team report will be graded based on the conciseness and clarity of the requested information and not on the performance of your team's strategies. The maximum page limit of the report is 4 pages.

Playing the game

To play RPS, ensure you have a working anaconda/python 3.xx environment. Apart from Numpy, you will need to install the Tkinter library using your conda or pip manager. Retrieve the file titled "rps_game.py" from the zip folder provided to you as part of the assignment and execute the file using your terminal. You should see a RPS GUI pop-up. To work with the "bayes_net.py", you will need to install the Pomegranate library using your conda or pip manager. Have fun playing RPS!

Submitting your work

There are two steps to submitting your assignment on Gradescope:

1. **Writeup:** Submit a combined .pdf file containing the findings from the case study. Submissions can be written in LaTeX or typed (try to ensure the file size is a maximum of 50 MB). The file name (FILE) for naming should be saved as TeamAssignment-*ASSIGNMENT-NUMBER*-*TEAM-NUMBER*.pdf. For example for team assignment 1, my FILE = "TeamAssignment-1-TEAM-CONRAD.pdf". Submit only one file per team and anyone from the team can make the submission.

2. **Team Assignment 01 Code:** Submit a .zip file containing the .py files. The ZIP folder containing your .py files should be named as TeamAssignment-*ASSIGNMENT-NUMBER*-*TEAM-NUMBER*.zip. Only submit the required files that you edit. All other files, including the ones we provide as supporting functionality is unnecessary because we already have copies of them. Unless we tell you otherwise, make sure that you only submit the file you edited/changed.