Article

Not peer-reviewed version

Which Machine-Learning Model Do You Want In Your Ocean's Eleven: A Computational Prisoner's Dilemma Simulation

Jayden Bai *

Posted Date: 23 July 2024

doi: 10.20944/preprints202407.1759.v1

Keywords: Game Theory; Prisoner's Dilemma; Machine Learning

Article

Which Machine-Learning Model Do You Want In Your Ocean's Eleven: A Computational Prisoner's Dilemma Simulation

Jayden Bai

mailto:jayden.bai@emory.edu; jayden.bai@emory.edu; mailto:jayden.bai@gmail.com; jayden.bai@gmail.com

Abstract: Which machine-learning model is the best at winning the prisoner's dilemma? Which models create the best cumulative outcomes? Is there a model that perfectly captures both winning and cumulative points? These are questions generated from a simple 2 x 2 payoff matrix of the prisoner's dilemma. Imagine you and your partner in crime are caught and sent independently into questioning. You can either collaborate or defect, but you don't know what your partner will do. If you both collaborate, cumulatively, you'll each get one year in jail. If you defect and your partner collaborates, you'll serve no time and they will serve 10 years, and vice versa. If you both defect, you'll both serve 5 years in jail. Placing AIs against each other to play just one round doesn't reveal much about their code, strategy, and end goal. So the Reinforcement Learning, Pattern Learning, Tit For Tat, and other models were put up against each other in a 100 round game where their behavior, convergence, and learning were analyzed to reveal the most effective ways strategies to beat the prisoner's dilemma. All code and data is open sourced here.

Keywords: game theory; prisoner's dilemma; machine learning

1. Introduction

Research surrounding the implications of machine-learning on the Prisoner's Dilemma has been primarily focused on Reinforcement Learning and the Tit-for-Tat strategy. The intention of this project is to analyze head-to-head match-ups between various machine-learning and game-theoretic algorithms and provide a statistical lens to view the 70-year-old dilemma.

The simulation in this project is a 100-round iterated Prisoner's Dilemma game. The game was programmed in Java and takes two Players and simulates the game based on each player's pre-programmed strategy while their learning models actively gain memory and experience that inform their moves. Some strategies are simple like AlwaysDefect and some are more complicated like the PatternLearner. Each player will gain points based on the reward matrix below.

Table 1. Reward Matrix

	C	**D**
C	+3 +3	+5 +0
D	+0 +5	+1 +1

2. Meet The Players

The first player is the QLearningPlayer, a Reinforcement Learning model that makes decisions based on real-time Q-value updates. This model is fitting because Q-Learning allows the player to balance exploration (moving randomly) and exploitation (comparing Q-values) and eventually converge to an optimal policy (C or D) based on the reward given per round. Q-values are updated and calculated using the Bellman equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_b Q(s',b) - Q(s,a)) \quad (1)$$

where α is the learning rate, γ is the discount factor, r is the reward, s' is the next state, and b is the next action. In the implementation of the game, the player class records the current state with each

 doi:10.20944/preprints202407.1759.v1

reward and updates its Q-values based on each move. α is set to 0.1, γ to 0.9, and exploration rate is set to 0.6 with a decay rate of 0.99 per round. These values are arbitrarily set but enable the model to learn and decrease exploration as the model is trained. Each Q-value for each move C or D is tracked in a HashMap with a double array and the player will collaborate or defect depending on which Q-value is higher.

In the first model, the reward system is set to account only for the player's personal reward (which would be values of 1, 3, or 5). This could lead to the player commonly converging to defection to maximize its own reward of 5. To explore the differences between personal reward and cumulative reward, a model called the JointPQLearningPlayer was created. This player considers the state of all players' moves (e.g., CC if both players collaborated) as opposed to the traditional account of one move (C if opponent collaborated). Additionally, this model's reward updated in the Bellman equation is the cumulative reward which would be values of 5, 6, or 2 for states CD or DC, CC, and DD respectively.

The next strategy is the BayesianInferencePlayer (BIP). This model is programmed to calculate the probabilities of each of the opponents' moves to structure the BIP's next move.

$$P_C = \frac{M_C}{t} \tag{2}$$

$$P_D = \frac{M_D}{t} \tag{3}$$

The model uses the equations above, where P_C and P_D represent the BIP's probability of C and D respectively, with M_C and M_D representing the number of times the opponent has C and D and t representing the total amount of moves. For example, if the model predicts that there is an 80% chance that the opponent will C, there will also be an 80% chance that the BIP will also C. The values of M_C and M_D are initialized to 1 and t to 2 for Laplace Smoothing to avoid the case of dividing by zero.

But what if we took the base idea from the BIP and took it a step further? The next strategy is the PatternLearningPlayer (PLP) which makes predictions based on the history of the opponent's moves. This player uses a simple n-gram model for sequence prediction and is visually depicted in Figure 1.

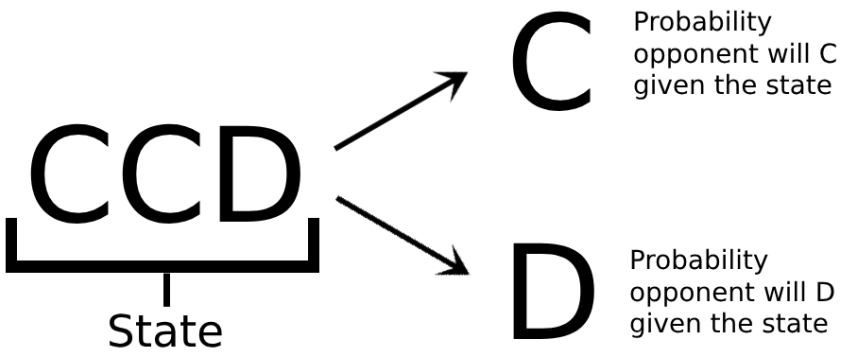

Figure 1. PatternLearning Algorithm

With a set history of 3, the PLP moves randomly for 3 rounds to collect data on the opponent's strategy. Each pattern is stored in a HashMap and then the model can predict, for example, if the opponent will C or D given a history of CCD based on the amount of the times the opponent's next move has been C or D given that state. The player will play whichever move is predicted of the opponent.

The next strategy is a game-theoretic strategy called TitForTat. This player's strategy revolves around doing the same move that the opponent had previously done. This strategy is especially interesting because it can "train" other models by giving them a taste of their own medicine. This gives them a punishment if they previously defected and a reward if they previously cooperated.

 doi:10.20944/preprints202407.1759.v1

Additionally, a reverse Tit-for-tat (revTitForTat) player was created who plays the opposite move that the opponent previously played.

The next player is the Upper Confidence Bound (UCB) Player which creates a bound for each move C or D and moves based on the larger UCB bound. This is a reinforcement learning algorithm that balances exploration and exploitation with the following equation:

$$UCB(a) = \mu_a + \sqrt{\frac{2 \ln t}{N_a}} \tag{4}$$

where μ_a is the mean reward for action a, t is the total number of times all actions have been played, and N_a is the number of times action a has been played. μ_a represents exploitation and the confidence interval term $\sqrt{\frac{2 \ln t}{N_a}}$ represents exploration and ensures that fewer-played actions get the chance to the played. By increasing the confidence interval for less played, the UCB ensures each action is explored before converging to a sub-optimal C or D.

Another game-theoretic player was added to test the limits and the cooperation levels of the different models. The Grudge Player begins with collaboration but once the opponent defects, the Grudge Player defects twice before going back to collaboration.

The next player is the PatternPlayer. Rather than C or D using a mathematical expression, the player moves based on a predetermined pattern. The numerical pattern is 01121220 and if there is a zero, the player C and if there is a one, the player D. When a two is encountered, those roles swap. The pattern for one complete iteration is shown below.

$$CDDCDC \tag{5}$$

Finally, a couple of other basic player models were created to diversify the player pool. A player that always collaborates, a player that always defects, a player that alternates between the two, and one that moves completely randomly were added.

3. Predictions

This results in five categorizations of the models. Firstly, the game-theoretic models that make decisions based on their opponents' moves. Next, the learning models who prefer cumulative reward like how the PatternLearner's and BayesianInference's models collaborate if they predict an opponent collaboration. The JointPQLearner updates its Q-values based on combinations of points which allows it to move as such. Next, the machine learners who are rewarded based on their own points gained and make decisions based on those. Next, the unchanging patterns which is self-explanatory. Lastly, the random player who is just random.

Table 2. Model Categorizations

Game-Theoretic	Learners (cumulative reward)	Learners (individual reward)	Unchanging Patterns	Random
TitForTat	PatternLearner	QLearner	Alternates	Random
revTitForTat	JointPQLearner	UCBPlayer	PatternPlayer	
grudgePlayer	BayesianInference		AlwaysC	
			AlwaysD	

By the end of the data processing stage, we will want to know the average cumulative scores to see how collaborative each match-up is, the number of CCs to better understand how each point total was played, and the win rates to see how competitive each player is. An initial prediction from looking at the categories is that the Game-Theoretic strategies will have the highest win rates (as they are programmed to be reactive rather than based on points), cumulative reward learners will have the highest cumulative scores and CCs, and individual reward learners will have a lower cumulative score but higher win rate.

 doi:10.20944/preprints202407.1759.v1

4. The Game

Displayed above is the terminal output of one 100-round game of the Pattern Player vs. the Pattern Learning Player. While not all 100 rounds are displayed, the program runs for a set number of iterations and tracks all point totals and times the players both collaborated and both defected to track cooperation levels. Each will play every player for a calculated number of games (not to be confused with rounds; each game has 100 rounds). The number of games will be calculated based on the sample size for a mean with a margin of error of 1 and a confidence interval of 95%. After each ten-round pilot study, the total amount of games is calculated for each player using:

$$n = \left(\frac{Z \cdot \sigma}{E}\right)^2 \tag{6}$$

where Z, the Z-value corresponding to the desired confidence level, is 1.96, E, the desired margin of error, is 1, and σ, the standard deviation, is estimated per match-up. The game model will take the standard deviation of the cumulative scores in the pilot study to estimate the standard deviation. After the pilot study, the model will dynamically resize the number of games needed for each player and will run those games into the generated database. The greatest number of rounds played between two players was 11,869,200 by the PatternLearningPlayer and the GrudgePlayer! The CSV dataset for each player will include the variables: Game, Opponent, PlayerA_Score, PlayerB_Score, Cumulative_Score, CCs, DDs, and Winner.

```
Round 1 | patternPlayer: C | pLearner: D | patternPlayer's points: 0 | pLearner's points: 5 | Cumulative: 5 | CCs: 0 | DDs: 0
Round 2 | patternPlayer: D | pLearner: C | patternPlayer's points: 5 | pLearner's points: 5 | Cumulative: 10 | CCs: 0 | DDs: 0
Round 3 | patternPlayer: D | pLearner: D | patternPlayer's points: 6 | pLearner's points: 6 | Cumulative: 12 | CCs: 0 | DDs: 1
Round 4 | patternPlayer: C | pLearner: C | patternPlayer's points: 9 | pLearner's points: 9 | Cumulative: 18 | CCs: 1 | DDs: 1
Round 5 | patternPlayer: D | pLearner: C | patternPlayer's points: 14 | pLearner's points: 9 | Cumulative: 23 | CCs: 1 | DDs: 1
Round 6 | patternPlayer: C | pLearner: C | patternPlayer's points: 17 | pLearner's points: 12 | Cumulative: 29 | CCs: 2 | DDs: 1
Round 7 | patternPlayer: C | pLearner: C | patternPlayer's points: 20 | pLearner's points: 15 | Cumulative: 35 | CCs: 3 | DDs: 1
Round 8 | patternPlayer: D | pLearner: C | patternPlayer's points: 25 | pLearner's points: 15 | Cumulative: 40 | CCs: 3 | DDs: 1
Round 9 | patternPlayer: D | pLearner: C | patternPlayer's points: 30 | pLearner's points: 15 | Cumulative: 45 | CCs: 3 | DDs: 1
Round 10 | patternPlayer: C | pLearner: C | patternPlayer's points: 33 | pLearner's points: 18 | Cumulative: 51 | CCs: 4 | DDs: 1
Round 11 | patternPlayer: D | pLearner: D | patternPlayer's points: 34 | pLearner's points: 19 | Cumulative: 53 | CCs: 4 | DDs: 2
Round 12 | patternPlayer: C | pLearner: C | patternPlayer's points: 37 | pLearner's points: 22 | Cumulative: 59 | CCs: 5 | DDs: 2
Round 13 | patternPlayer: C | pLearner: C | patternPlayer's points: 40 | pLearner's points: 25 | Cumulative: 65 | CCs: 6 | DDs: 2
Round 14 | patternPlayer: D | pLearner: D | patternPlayer's points: 41 | pLearner's points: 26 | Cumulative: 67 | CCs: 6 | DDs: 3
Round 15 | patternPlayer: D | pLearner: D | patternPlayer's points: 42 | pLearner's points: 27 | Cumulative: 69 | CCs: 6 | DDs: 4
Round 16 | patternPlayer: C | pLearner: C | patternPlayer's points: 45 | pLearner's points: 30 | Cumulative: 75 | CCs: 7 | DDs: 4
Round 17 | patternPlayer: D | pLearner: D | patternPlayer's points: 46 | pLearner's points: 31 | Cumulative: 77 | CCs: 7 | DDs: 5
Round 18 | patternPlayer: C | pLearner: C | patternPlayer's points: 49 | pLearner's points: 34 | Cumulative: 83 | CCs: 8 | DDs: 5
Round 19 | patternPlayer: C | pLearner: C | patternPlayer's points: 52 | pLearner's points: 37 | Cumulative: 89 | CCs: 9 | DDs: 5
Round 20 | patternPlayer: D | pLearner: D | patternPlayer's points: 53 | pLearner's points: 38 | Cumulative: 91 | CCs: 9 | DDs: 6
Round 21 | patternPlayer: D | pLearner: D | patternPlayer's points: 54 | pLearner's points: 39 | Cumulative: 93 | CCs: 9 | DDs: 7
Round 22 | patternPlayer: C | pLearner: C | patternPlayer's points: 57 | pLearner's points: 42 | Cumulative: 99 | CCs: 10 | DDs: 7
Round 23 | patternPlayer: D | pLearner: D | patternPlayer's points: 58 | pLearner's points: 43 | Cumulative: 101 | CCs: 10 | DDs: 8
Round 24 | patternPlayer: C | pLearner: C | patternPlayer's points: 61 | pLearner's points: 46 | Cumulative: 107 | CCs: 11 | DDs: 8
Round 25 | patternPlayer: C | pLearner: C | patternPlayer's points: 64 | pLearner's points: 49 | Cumulative: 113 | CCs: 12 | DDs: 8
Round 26 | patternPlayer: D | pLearner: D | patternPlayer's points: 65 | pLearner's points: 50 | Cumulative: 115 | CCs: 12 | DDs: 9
Round 27 | patternPlayer: D | pLearner: D | patternPlayer's points: 66 | pLearner's points: 51 | Cumulative: 117 | CCs: 12 | DDs: 10
Round 28 | patternPlayer: C | pLearner: C | patternPlayer's points: 69 | pLearner's points: 54 | Cumulative: 123 | CCs: 13 | DDs: 10
Round 29 | patternPlayer: D | pLearner: D | patternPlayer's points: 70 | pLearner's points: 55 | Cumulative: 125 | CCs: 13 | DDs: 11
Round 30 | patternPlayer: C | pLearner: C | patternPlayer's points: 73 | pLearner's points: 58 | Cumulative: 131 | CCs: 14 | DDs: 11
Round 31 | patternPlayer: C | pLearner: C | patternPlayer's points: 76 | pLearner's points: 61 | Cumulative: 137 | CCs: 15 | DDs: 11
Round 32 | patternPlayer: D | pLearner: D | patternPlayer's points: 77 | pLearner's points: 62 | Cumulative: 139 | CCs: 15 | DDs: 12
Round 33 | patternPlayer: D | pLearner: D | patternPlayer's points: 78 | pLearner's points: 63 | Cumulative: 141 | CCs: 15 | DDs: 13
Round 34 | patternPlayer: C | pLearner: C | patternPlayer's points: 81 | pLearner's points: 66 | Cumulative: 147 | CCs: 16 | DDs: 13
Round 35 | patternPlayer: D | pLearner: D | patternPlayer's points: 82 | pLearner's points: 67 | Cumulative: 149 | CCs: 16 | DDs: 14
Round 36 | patternPlayer: C | pLearner: C | patternPlayer's points: 85 | pLearner's points: 70 | Cumulative: 155 | CCs: 17 | DDs: 14
Round 37 | patternPlayer: C | pLearner: C | patternPlayer's points: 88 | pLearner's points: 73 | Cumulative: 161 | CCs: 18 | DDs: 14
Round 38 | patternPlayer: D | pLearner: D | patternPlayer's points: 89 | pLearner's points: 74 | Cumulative: 163 | CCs: 18 | DDs: 15
Round 39 | patternPlayer: D | pLearner: D | patternPlayer's points: 90 | pLearner's points: 75 | Cumulative: 165 | CCs: 18 | DDs: 16
Round 40 | patternPlayer: C | pLearner: C | patternPlayer's points: 93 | pLearner's points: 78 | Cumulative: 171 | CCs: 19 | DDs: 16
Round 41 | patternPlayer: D | pLearner: D | patternPlayer's points: 94 | pLearner's points: 79 | Cumulative: 173 | CCs: 19 | DDs: 17
Round 42 | patternPlayer: C | pLearner: C | patternPlayer's points: 97 | pLearner's points: 82 | Cumulative: 179 | CCs: 20 | DDs: 17
Round 43 | patternPlayer: C | pLearner: C | patternPlayer's points: 100 | pLearner's points: 85 | Cumulative: 185 | CCs: 21 | DDs: 17
Round 44 | patternPlayer: D | pLearner: D | patternPlayer's points: 101 | pLearner's points: 86 | Cumulative: 187 | CCs: 21 | DDs: 18
Round 45 | patternPlayer: D | pLearner: D | patternPlayer's points: 102 | pLearner's points: 87 | Cumulative: 189 | CCs: 21 | DDs: 19
Round 46 | patternPlayer: C | pLearner: C | patternPlayer's points: 105 | pLearner's points: 90 | Cumulative: 195 | CCs: 22 | DDs: 19
Round 47 | patternPlayer: D | pLearner: D | patternPlayer's points: 106 | pLearner's points: 91 | Cumulative: 197 | CCs: 22 | DDs: 20
Round 48 | patternPlayer: C | pLearner: C | patternPlayer's points: 109 | pLearner's points: 94 | Cumulative: 203 | CCs: 23 | DDs: 20
Round 49 | patternPlayer: C | pLearner: C | patternPlayer's points: 112 | pLearner's points: 97 | Cumulative: 209 | CCs: 24 | DDs: 20
Round 50 | patternPlayer: D | pLearner: D | patternPlayer's points: 113 | pLearner's points: 98 | Cumulative: 211 | CCs: 24 | DDs: 21
Round 51 | patternPlayer: D | pLearner: D | patternPlayer's points: 114 | pLearner's points: 99 | Cumulative: 213 | CCs: 24 | DDs: 22
Round 52 | patternPlayer: C | pLearner: C | patternPlayer's points: 117 | pLearner's points: 102 | Cumulative: 219 | CCs: 25 | DDs: 22
```

Figure 2. Game Terminal Output

The terminal output for the simulation ran for the QLearningPlayer (where the QLearner is playerA) is shown in Figure 3 and the beginning of the 17529-row dataset is shown in Figure 4. Most of the names should be self-explanatory, but Gullible is Always Collaborate, Sly Fox is Always Defect,

 doi:10.20944/preprints202407.1759.v1

Randy is Random, and Evan is Alternate Collaboration. All data structures containing "memory" of the learning players were reset from game to game, ensuring that the models are only learning between rounds, not independent games – otherwise, data points would repetitively converge to one score.

```
Playing against Gullible
Playing 93 games against Gullible
Finished playing against Gullible. Games played: 93
Playing against Sly Fox
Playing 993 games against Sly Fox
Finished playing against Sly Fox. Games played: 993
Playing against Randy
Playing 630 games against Randy
Finished playing against Randy. Games played: 630
Playing against Evan
Playing 423 games against Evan
Finished playing against Evan. Games played: 423
Playing against ReinforcementLearningPlayer
Playing 488 games against ReinforcementLearningPlayer
Finished playing against ReinforcementLearningPlayer. Games played: 488
Playing against pLearner
Playing 5198 games against pLearner
Finished playing against pLearner. Games played: 5198
Playing against titForTat
Playing 1472 games against titForTat
Finished playing against titForTat. Games played: 1472
Playing against Bayesian
Playing 1220 games against Bayesian
Finished playing against Bayesian. Games played: 1220
Playing against UCBPlayer
Playing 3907 games against UCBPlayer
Finished playing against UCBPlayer. Games played: 3907
Playing against JointPQPlayer
Playing 1188 games against JointPQPlayer
Finished playing against JointPQPlayer. Games played: 1188
Playing against revtitForTat
Playing 136 games against revtitForTat
Finished playing against revtitForTat. Games played: 136
Playing against patternPlayer
Playing 311 games against patternPlayer
Finished playing against patternPlayer. Games played: 311
Playing against Grudge
Playing 2429 games against Grudge
Finished playing against Grudge. Games played: 2429
File generated
```

Figure 3. Simulation Terminal Output

 doi:10.20944/preprints202407.1759.v1

Game	Opponent	PlayerA_Scor	PlayerB_Scor	Cumulative_	CCs	DDs	Winner
0	Guillible	410	135	545	45	0	Player A
1	Guillible	406	141	547	47	0	Player A
2	Guillible	388	168	556	56	0	Player A
3	Guillible	404	144	548	48	0	Player A
4	Guillible	404	144	548	48	0	Player A
5	Guillible	396	156	552	52	0	Player A
6	Guillible	388	168	556	56	0	Player A
7	Guillible	396	156	552	52	0	Player A
8	Guillible	410	135	545	45	0	Player A
9	Guillible	398	153	551	51	0	Player A
10	Guillible	398	153	551	51	0	Player A
11	Guillible	388	168	556	56	0	Player A
12	Guillible	394	159	553	53	0	Player A
13	Guillible	396	156	552	52	0	Player A
14	Guillible	404	144	548	48	0	Player A
15	Guillible	392	162	554	54	0	Player A
16	Guillible	402	147	549	49	0	Player A
17	Guillible	386	171	557	57	0	Player A
18	Guillible	384	174	558	58	0	Player A
19	Guillible	386	171	557	57	0	Player A
20	Guillible	412	132	544	44	0	Player A
21	Guillible	398	153	551	51	0	Player A
22	Guillible	404	144	548	48	0	Player A

Figure 4. QLearner Dataset

This process created nine datasets with each playing player (all players except the four noted not to play each other) which were processed and merged to create a database with usable numbers.

5. Processing

Once all nine datasets were created, they entered a procedural stage where they were processed and merged into 3 data frames which show the results of each match-up with our variables of interest: Cumulative Scores (Figure 5), Win Rates (Figure 6), and CC Rates (Figure 7).

✓ #merged Cumulative Score dataframe ···

	QLearningPlayer	PatternLearningPlayer	TitForTat	Bayesian	UCBPlayer	JointPQPlayer	revTitforTat	PatternPlayer	GrudgePlayer	Gullible	Sly Fox	Evan	Randy
QLearningPlayer	449.795122	471.086263	449.201592	449.299926	358.747928	469.234660	450.555556	450.093373	398.941345	550.098592	349.647303	449.845411	450.185687
PatternLearningPlayer	471.393345	595.413043	586.519967	596.665816	244.917339	494.328575	444.759615	410.750000	384.920795	598.636364	207.536585	405.090909	471.912472
TitForTat	450.507032	587.140818	600.000000	450.990358	229.628205	476.377402	450.000000	466.538462	600.000000	600.000000	200.272727	499.727273	450.210643
Bayesian	448.830731	521.936701	522.513644	524.189474	404.056725	483.009313	451.967742	467.050847	422.630834	551.745098	354.330435	452.334071	451.388060
UCBPlayer	359.690566	246.446809	229.750000	233.070248	356.862477	408.358893	496.363636	353.000000	218.000000	501.909091	217.750000	353.000000	359.038860
JointPQLearner	469.845538	493.567640	475.844235	481.934059	409.963094	484.755233	459.000000	467.711538	459.511322	557.501266	403.003610	469.276403	469.553186
revTitforTat	449.701613	443.391813	450.000000	449.117647	496.000000	460.321429	400.000000	433.461538	460.000000	500.000000	499.727273	400.272727	449.958333
PatternPlayer	449.397183	410.409091	466.666667	449.752381	367.000000	469.507628	433.461538	466.727273	400.272727	550.000000	350.000000	433.454545	450.259587
GrudgePlayer	399.887156	386.888952	200.272727	200.400000	217.727273	458.541882	460.000000	400.272727	200.272727	599.818182	200.272727	350.000000	399.896650
Gullible	549.670520	599.000000	600.000000	599.272727	502.000000	557.416836	500.090909	550.000000	600.000000	600.000000	500.000000	550.000000	550.660550
Sly Fox	350.550409	206.954545	200.272727	200.545455	217.727273	404.837156	500.000000	350.000000	200.272727	500.000000	200.000000	350.000000	349.544944
Evan	450.569647	405.772727	500.000000	450.923513	353.000000	468.493511	400.272727	433.454545	350.272727	550.000000	350.000000	400.000000	450.826840
Randy	449.290196	471.456602	450.013839	450.349157	359.437495	469.445971	449.138462	450.257261	400.095572	551.238095	350.075435	449.945055	450.225000

Figure 5. Cumulative Scores

 doi:10.20944/preprints202407.1759.v1

✓ #merged WR dataframe ···

	QLearningPlayer	PatternLearningPlayer	TitForTat	Bayesian	UCBPlayer	JointPQPlayer	revTitforTat	PatternPlayer	GrudgePlayer	Gullible	Sly Fox	Evan	Randy
QLearningPlayer	0.443206	0.867706	0.247569	0.475130	0.000691	0.771245	0.555556	0.475904	0.000000	1.000000	0.0	0.475845	0.473888
PatternLearningPlayer	0.109532	0.326087	0.118989	0.354592	0.135333	0.195382	0.471154	0.000000	0.000000	0.818182	0.0	0.000000	0.107103
TitForTat	0.256847	0.110371	0.000000	0.240358	0.076923	0.230019	0.000000	0.384615	0.000000	0.000000	0.0	0.090909	0.270510
Bayesian	0.474938	0.968506	0.200642	0.494737	0.004575	0.625067	0.016129	0.039548	0.000000	1.000000	0.0	0.369469	0.436567
UCBPlayer	0.998383	0.680851	0.166667	0.471074	0.000000	0.992818	1.000000	1.000000	0.000000	1.000000	0.0	1.000000	0.998850
JointPQLearner	0.221721	0.792354	0.233117	0.338194	0.008089	0.491917	0.285714	0.228365	0.000000	1.000000	0.0	0.229065	0.225645
revTitforTat	0.443548	0.432749	0.000000	0.478992	0.000000	0.678571	0.000000	0.307692	0.000000	1.000000	0.0	0.000000	0.666667
PatternPlayer	0.443662	1.000000	0.333333	0.460317	0.000000	0.777223	0.384615	0.000000	0.000000	1.000000	0.0	0.000000	0.480826
GrudgePlayer	1.000000	0.947712	0.090909	0.066667	1.000000	1.000000	1.000000	1.000000	0.090909	0.090909	0.0	1.000000	1.000000
Gullible	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
Sly Fox	1.000000	1.000000	0.090909	0.121212	1.000000	1.000000	1.000000	1.000000	0.090909	1.000000	0.0	1.000000	1.000000
Evan	0.461538	0.954545	0.000000	0.450425	0.000000	0.760970	0.090909	0.000000	0.000000	1.000000	0.0	0.000000	0.515152
Randy	0.439216	0.873038	0.249097	0.480479	0.002259	0.773783	0.476923	0.446058	0.000000	1.000000	0.0	0.417582	0.496875

Figure 6. Win Rates

✓ #merged CCs dataframe ···

	QLearningPlayer	PatternLearningPlayer	TitForTat	Bayesian	UCBPlayer	JointPQPlayer	revTitforTat	PatternPlayer	GrudgePlayer	Gullible	Sly Fox	Evan	Randy
QLearningPlayer	0.249477	0.303784	0.249469	0.248255	0.021533	0.299302	0.247222	0.250030	0.123577	0.500986	0.0	0.248671	0.251219
PatternLearningPlayer	0.303951	0.977609	0.882632	0.967500	0.032933	0.449515	0.276346	0.491250	0.398437	0.986364	0.0	0.492273	0.305165
TitForTat	0.251229	0.889020	1.000000	0.250937	0.000128	0.384667	0.250000	0.166154	1.000000	1.000000	0.0	0.000000	0.251840
Bayesian	0.249129	0.486149	0.493210	0.499516	0.033504	0.343143	0.240161	0.291907	0.166995	0.517451	0.0	0.255719	0.255345
UCBPlayer	0.023903	0.039362	0.000000	0.003306	0.171568	0.039806	0.018182	0.000000	0.000000	0.019091	0.0	0.000000	0.022497
JointPQLearner	0.299421	0.447335	0.383173	0.339597	0.044146	0.340877	0.205195	0.296731	0.295697	0.575013	0.0	0.299023	0.298312
revTitforTat	0.251613	0.283041	0.250000	0.254286	0.020000	0.198393	0.500000	0.333846	0.200000	0.000000	0.0	0.500000	0.250000
PatternPlayer	0.248239	0.493636	0.166667	0.249905	0.050000	0.298958	0.333846	0.166364	0.000909	0.500000	0.0	0.332727	0.251091
GrudgePlayer	0.125343	0.403844	0.000000	0.000000	0.000000	0.293593	0.200000	0.000909	0.000000	0.998182	0.0	0.000000	0.124875
Gullible	0.496705	0.990000	1.000000	0.992727	0.020000	0.574168	0.000909	0.500000	1.000000	1.000000	0.0	0.500000	0.506606
Sly Fox	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
Evan	0.250395	0.496364	0.000000	0.247762	0.000000	0.297557	0.500000	0.332727	0.000000	0.500000	0.0	0.500000	0.253723
Randy	0.248078	0.302461	0.249471	0.249818	0.022929	0.298968	0.254308	0.252386	0.125761	0.512381	0.0	0.251923	0.254312

Figure 7. CC Rates

Each of these data frames include all relevant analysis from the large datasets that included scores from each game.

6. Results

After the processing stage, the result of the AlwaysCollaborate (Gullible) player encouraging the highest average score total is not a surprise – it collaborates every time! Additionally, the prediction of the Cumulative Reward Learners performing well in point totals held up in Figure 8 – with the JointPQLearner creating the highest score out of any non-pattern player. This is logical because the model gains the most personal reward when both players collaborate and will get lower reward if there is any defection. The surprise is the PatternLearningPlayer's algorithmic complexity under-performing compared to the Bayesian player's straightforward calculations. This reveals that specificity doesn't always result in more allocative results.

For the Game-Theoretic players, an interesting pattern will be noticed within the next coming figures. The revTitForTat consistently shows up between TitForTat and the GrudgePlayer seemingly acting as a median of the two. This contextually makes sense in this visualization because TitForTat encourages collaboration more than the reverse model and definitely more than the GrudgePlayer who often likes defection twice as much as its opponent. It is interesting to also conceptually think of revTitForTat as the middle ground between the two because they all handle defection and collaboration in similar ways.

 doi:10.20944/preprints202407.1759.v1

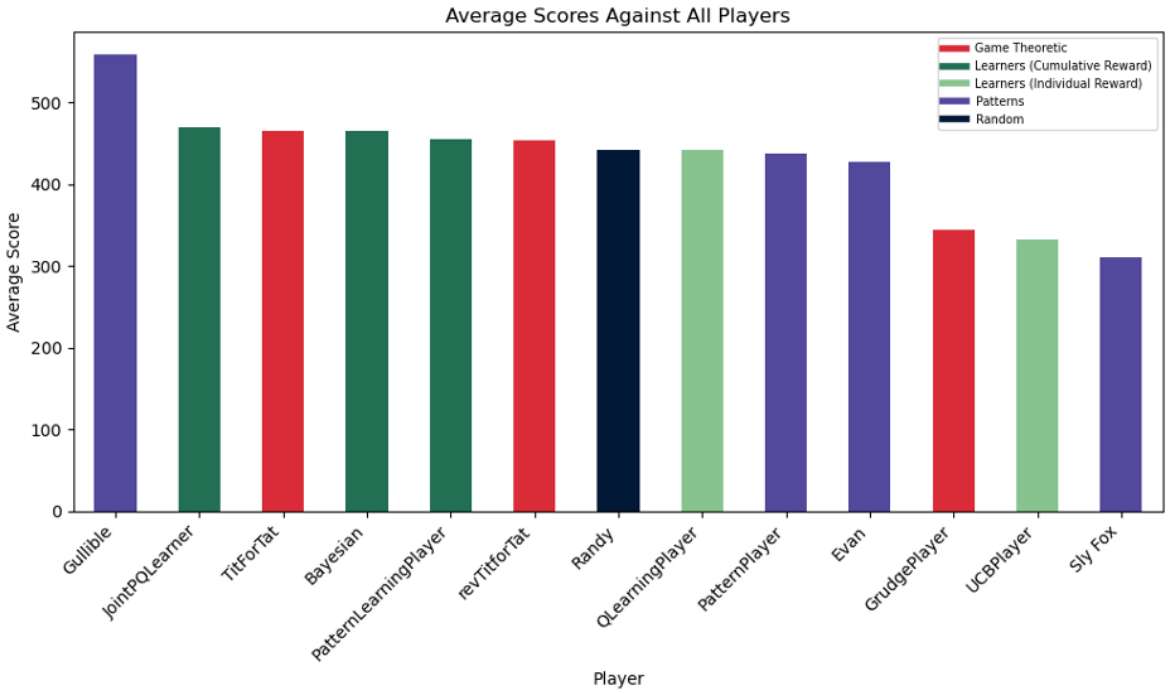

Figure 8. Results Overview

Obviously, the AlwaysDefect (Sly Fox) remains on top in Win Rates – constant defection is the dominant strategy which means it literally cannot lose (it can tie which is why the win rate is not 100%). Figure 9 further supports the prediction of Individual Reward Learners performing well in win rates; players like the UpperConfidenceBound can identify that defection is key to higher point totals. However, the surprise is in successful Game Theoretic GrudgePlayer. The rate makes sense as it punishes its opponents for defection with double the loss (by defecting twice when receiving a defection) which may result in constant defection similar to the Sly Fox's game model.

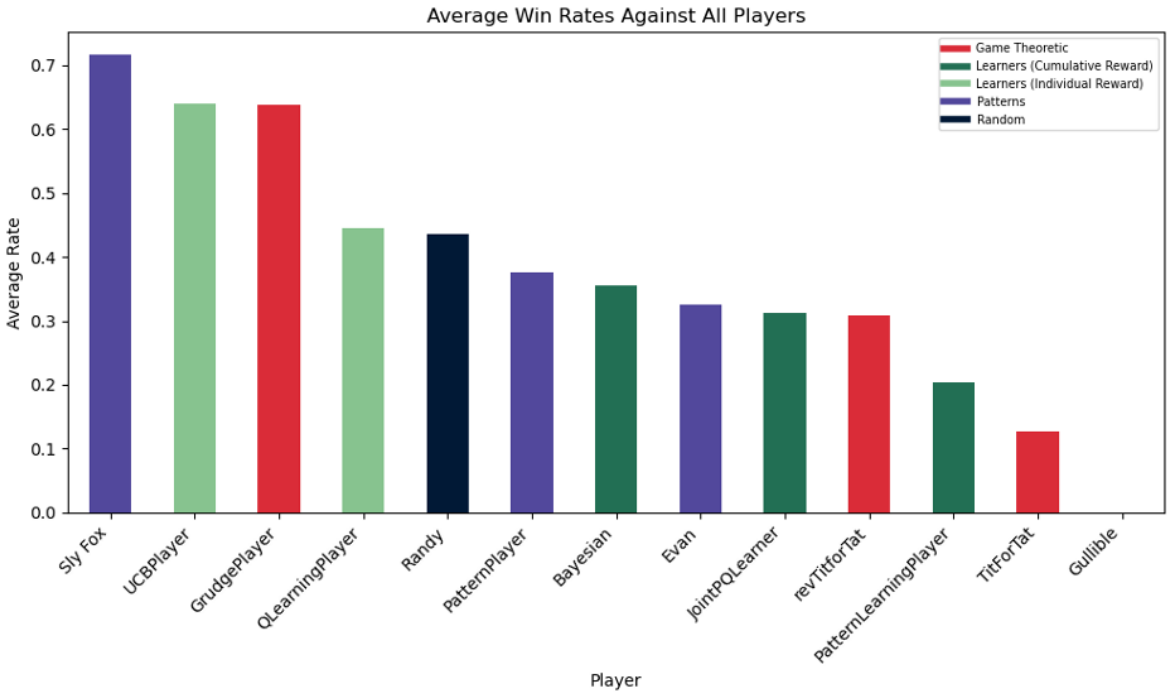

Figure 9. Win Rates of Different Strategies

 doi:10.20944/preprints202407.1759.v1

Despite the hype, the TitForTat player curiously under-performed in this category – proving that the player may not be the most offensive. Upon further inspection, it was because the TitForTat player has a tie rate of 77% which reveals the efficacy of the TitForTat player in accumulating a lot of points but not winning in a margin over the opponent. This also makes sense because the TitForTat player makes decisions reactive to the opponent and will copy the opponent's pattern.

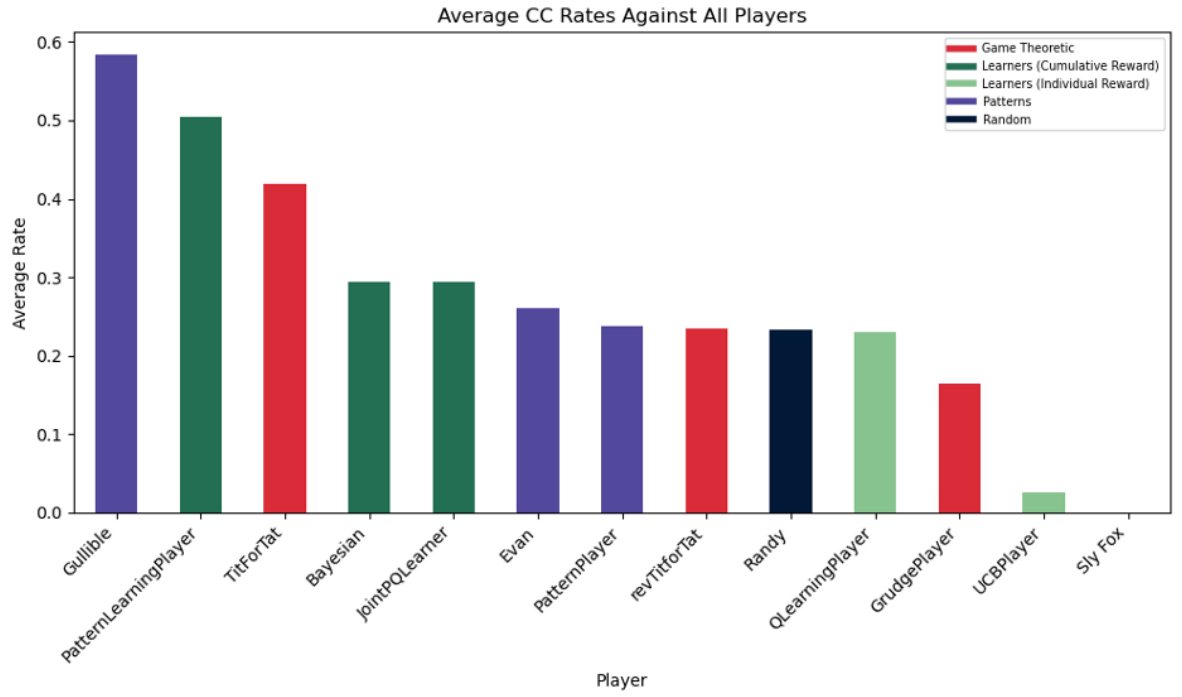

Figure 10. CC Rates of Different Strategies

Again, the AlwaysCollaborate (Gullible) player being the most collaborative is not a surprise because it always collaborates. Expectedly, the PatternLearningPlayer (a popular choice now) takes the spot as the most effective at predicting and making CCs happen. Interestingly, TitForTat, a model that is able to train the opponent to collaborate, performs better than the JointPQLearner who is rewarded when both models collaborate. This reveals that punishing the opponent for defecting may be more effective than only one of the two players encouraging collaboration.

The behavior of the Evan (Alternate) PatternPlayer, revTitForTat, and Randy (random) statistically make sense, hovering around the 25% mark. Out of the four possible outcomes (CC, CD, DC, DD), it would be a 25% chance to CC given that the strategies are mostly random and pattern-based. This implicates how the QLearningPlayer moves as well, suggesting that its behavior behaves similarly to a simple ¼ probability. Upon further investigation, it was found that the DD rate was also 25%. It makes sense that the CC rate would match the DD rate since the QLearningPlayer balances both exploitation and exploration. However, I don't believe that there is as much significance that the rates are both around 25% since the numerical values are based on the player pool and which strategies the model is competing against.

The player combinations that had the highest cumulative scores of 600 points (the max points they could attain) were the GrudgePlayer with TitForTat, TitForTat with TitForTat, and those players with AlwaysCollaborate. The best performing combination of learning models was the PatternLearner with the Bayesian player with approximately 596 points. The worst combinations were the GrudgePlayer with itself at a total of approximately 200 and the GrudgePlayer with the UCBPlayer with also 200 points.

To emphasize the game in game theory, the players were placed in a bracket to decide the most ruthless and individualistic prisoner's dilemma winner. AlwaysDefect cannot lose and AlwaysCol-

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 July 2024 doi:10.20944/preprints202407.1759.v1

laborate cannot win, so they were excluded from the bracket. The 11 players were seeded by their win rates and to stay true to the inherent nature of a competition, they were only allowed to play one round which will decide who moves on. Ties will result in the next non-tie game being recorded.

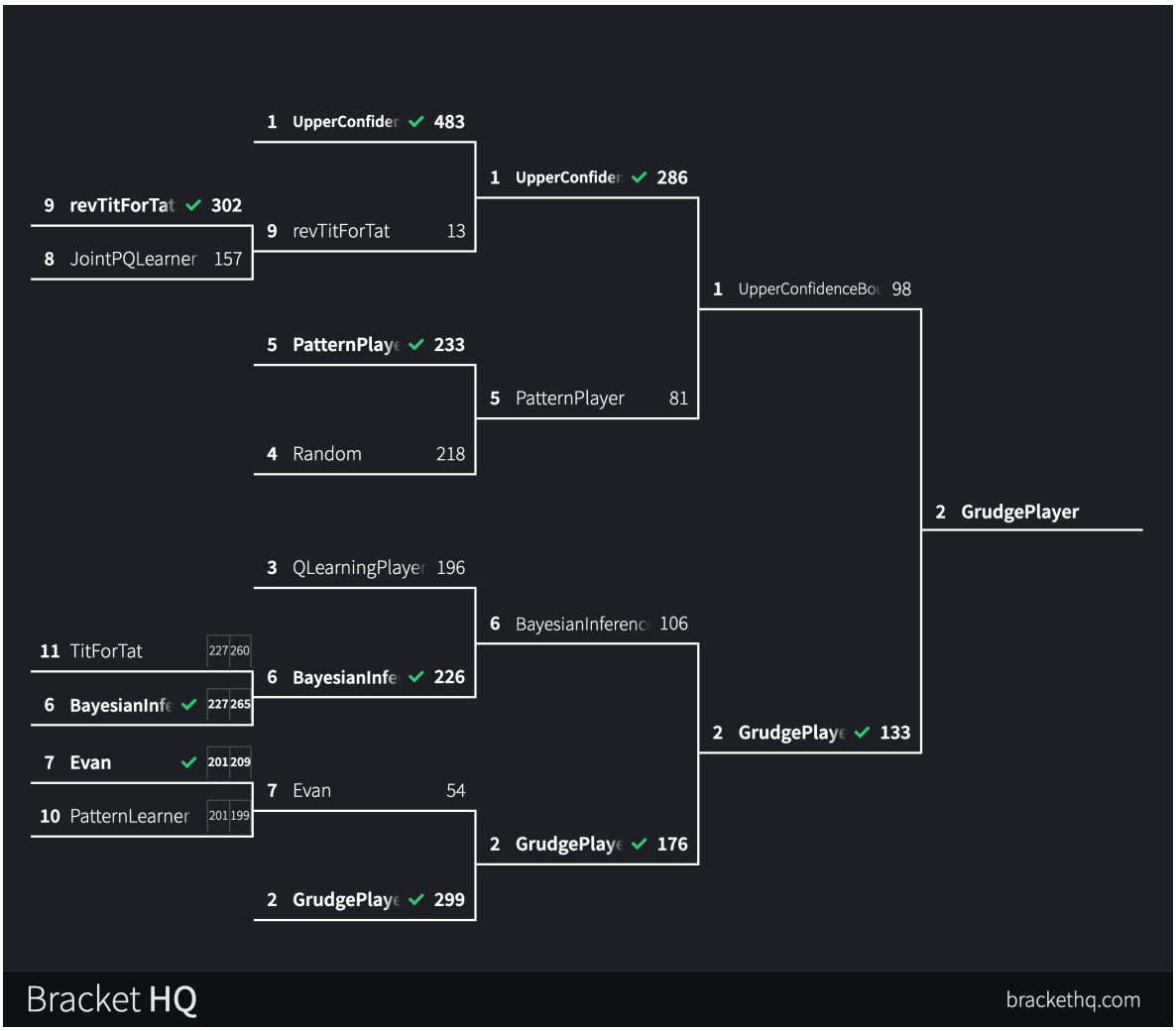

Figure 11. Tournament Bracket

As the game progressed, it looked as if No. 1 UpperConfidenceBound was going to take the game after having a dominant win streak but ultimately it was the No.2 seed GrudgePlayer in a slight upset to run away with the victory in the lowest point game in the entire tournament. The fact that commonly defecting strategies did better in the bracket while decreasing total game points illustrates the payoff matrix's trade-off in prioritizing wins. Interestingly, in the first round both No. 11 TitForTat vs. No. 6 BayesianInference and No. 10 PatternLearner vs. No. 7 Evan both initially tied but it wasn't enough to pull the upset.

7. Conclusion

As the winner of the bracket reveals, in a simple game like the Prisoner's Dilemma where a player can only either collaborate or defect, sometimes a simple strategy with just 27 lines of code beats a complicated mathematical algorithm. Starting all the way back at the matrix, there are only two results the players can have: the allocative result or the selfish result. The simulation holds true that players whose programs are embedded with allocative rewards move for cumulative good while individual reward systems move selfishly.

 doi:10.20944/preprints202407.1759.v1

If a player's only goal is to win the round, it should defect every time as it is the dominant strategy from the payoff matrix. The nuance comes when players want to maximize cumulative good – learning to trust and predict moves is key. This necessitates a model who is encoded with cumulative good within its decision-making process (the cumulative reward learners) or the ability to train other models away from defection (like the TitForTat model).

This may come as an intuitive thought but in the context of broader decisions involving outcomes more than just CC, CD, DC, and DD, this simulation informs us that decisions will be made according to a player's essential nature. So when picking your partner-in-crime make sure that they are not either a Sly Fox, a grudge holder, or someone who operates based on the Upper Confidence Bound.

References

1. Sandholm, Tuomas W, and Robert H Crites. "Multiagent Reinforcement Learning in the Iterated Prisoner's Dilemma." Science Direct, Elsevier Ireland Ltd., 26 Apr. 1999, http://www.sciencedirect.com/science/article/abs/pii/0303264795015515.
2. "Upper Confidence Bound Algorithm in Reinforcement Learning." GeeksforGeeks, GeeksforGeeks, 19 Feb. 2020, http://www.geeksforgeeks.org/upper-confidence-bound-algorithm-in-reinforcement-learning/.
3. Veritasium. "What Game Theory Reveals about Life, the Universe, and Everything." YouTube, YouTube, 23 Dec. 2023, http://www.youtube.com/watch?v=mScpHTIi-kM.
4. Surma, Greg. "Prison Escape - Solving Prisoner's Dilemma with Machine Learning." Medium, Medium, 11 Apr. 2019, http://gsurma.medium.com/ prison-escape-solving-prisoners-dilemma-with-machine-learning-c194600b 0b71.