REPORT
ON

Algorithms to find Dynamic Best Response in a UN Voting Session

Under the guidance of

Dr. Himadri Mukherjee

BY

Names of the Students	**ID Numbers**
Arkil Patel	2016A7PS0665G
Hritika Suneja	2016A7PS0093G
Vedashree Kulkarni	2016B4TS0637G

Prepared in partial fulfillment of the
Course No. BITS F314
Course Title: Game Theory

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
(DECEMBER, 2018)

Acknowledgements

We are extremely pleased to acknowledge the Game Theory Course Instructor-In-Charge, Dr. Himadri Mukherjee, Assistant Professor, BITS Goa, for his invaluable guidance during the course of the project work. Without his constant guidance, this project would have been an impossible task.

The concepts taught by him during the semester August-December, 2018, for Game Theory and the research papers we looked up for inspiration of this project helped build this model.

We are also grateful to our friends, family and many other resources which constantly helped us regarding some issues that we faced and thereby, we were successfully able to fix them.

6, Dec 2018

Arkil Patel
Hritika Suneja
Vedashree Kulkarni

TABLE OF CONTENTS

Acknowledgements ______ *01*
Table of Contents ______ *02*
1. Introduction ______ 03
2. Repeated Games ______ 04
    - 2.1. Types of Repeated Games ______ 04
    - 2.2. Strategies in a Repeated Game ______ 04
3. Problem Statement ______ 05
    - 3.1. Game Description ______ 05
    - 3.2. Dataset ______ 06
4. Payoff Matrices ______ 08
    - 4.1. Goodwill Matrix ______ 08
    - 4.2. Profit Matrix ______ 09
    - 4.3. Bayesian Verification ______ 10
    - 4.4. Reward Matrix ______ 11
5. Algorithms ______ 12
    - 5.1. Q-Learning ______ 12
    - 5.2. Dynamic Reward ______ 17
6. Conclusion ______ 19
7. Future Scope ______ 20
    - 7.1. Improvements in the UN Voting Simulation ______ 20
    - 7.2. Environment Pollution Control: A Case Study ______ 20

References ______ *24*

1. INTRODUCTION

Game Theory

Game theory is the study of mathematical models of strategic interaction between rational decision-makers. It has applications in all fields of social science, as well as in logic and computer science. Originally, it addressed zero-sum games, in which one person's gains result in losses for the other participants. Today, game theory applies to a wide range of behavioral relations, and is now an umbrella term for the science of logical decision making in humans, animals, and computers.

Types of Games

There can be various kinds of games: cooperative, non-cooperative games; Repeated games, non-repeated games and many more. In cooperative game, also called coalitional game players form group and play as a team so it is effectively a game between two teams. In non-cooperative game there is no possibility of forming an alliance. For solving any game, we have a set of players and a set of actions for each player. Corresponding to these are respective payoff values which we use to find the Nash Equilibrium.

Repeated Games

Repeated games are the kind of games that are repeated many times. The interesting thing about repeated games is that the action of a player in a game will depend on the actions of the players in the previous games and the fact that they might have to meet in near future.

Problem Statement

Modeling the United Nations Voting Session as an Infinitely Repeated Game or a Finitely Repeated Game where the players do not have the knowledge of the termination of the game. On the basis of their past and the future, we deduce algorithms to obtain the Best Response over a period of time.

Algorithms

Two algorithms developed by us will be explained in detail – (i) The Q-Learning Algorithm and (ii) The Dynamic Reward Algorithm.

2. REPEATED GAMES

Repeated games consist of some base game that is played repeatedly by the players. At each point of time, the players can be seen opting any of the base game strategies available to them.

2.1. TYPES OF REPEATED GAMES

A. **FINITELY REPEATED:** In finitely repeated games the game terminates after a finite number of times. Finitely repeated games often reduce to a simple game. E.g. Let us consider an iterated prisoner's dilemma. So, in this we iterate Prisoners dilemma many numbers of times, and every time the strategy of the player will depend on the past responses. But in the last game irrespective of the past moves both the players will choose to defect. So, this further can be reduced to the second last strategy and so on.

B. **INFINITELY REPEATED:** In infinitely repeated games there are further two types. One is that the game repeats infinite times and does not terminate and the other is when the game terminates at a point in future but the players don't know when.

2.2. STRATEGIES IN A REPEATED GAME

There are generally two strategies in a repeated game:

1. **Tit for Tat:** Every time play the opponent's previous move.
2. **Grim Trigger Strategy:** Cooperate until the opponent defects and when the opponent defects you defect forever.

Grim trigger is generally not a wise strategy to be followed in real life examples.

3. PROBLEM STATEMENT

In our project we try to model the voting in a UN General Assembly as an Infinitely Repeated Game or a Finitely Repeated Game with infinite horizon, i.e., where the players do not have the knowledge of the termination of the game.

Consider two countries that are competitive with each other (e.g. United States and China). They do not wish good things for each other. So ideally, in any voting scenario, US will vote negatively for resolutions that benefit China and vice versa. But they cannot do this in every session as they also need their opponent's vote for their own resolutions to get passed. So, their needs to be some equilibrium between the two different mindsets of the countries.

We have used learning algorithms to study the past responses and then predict what might be the best strategy to be taken in the given set of situations.

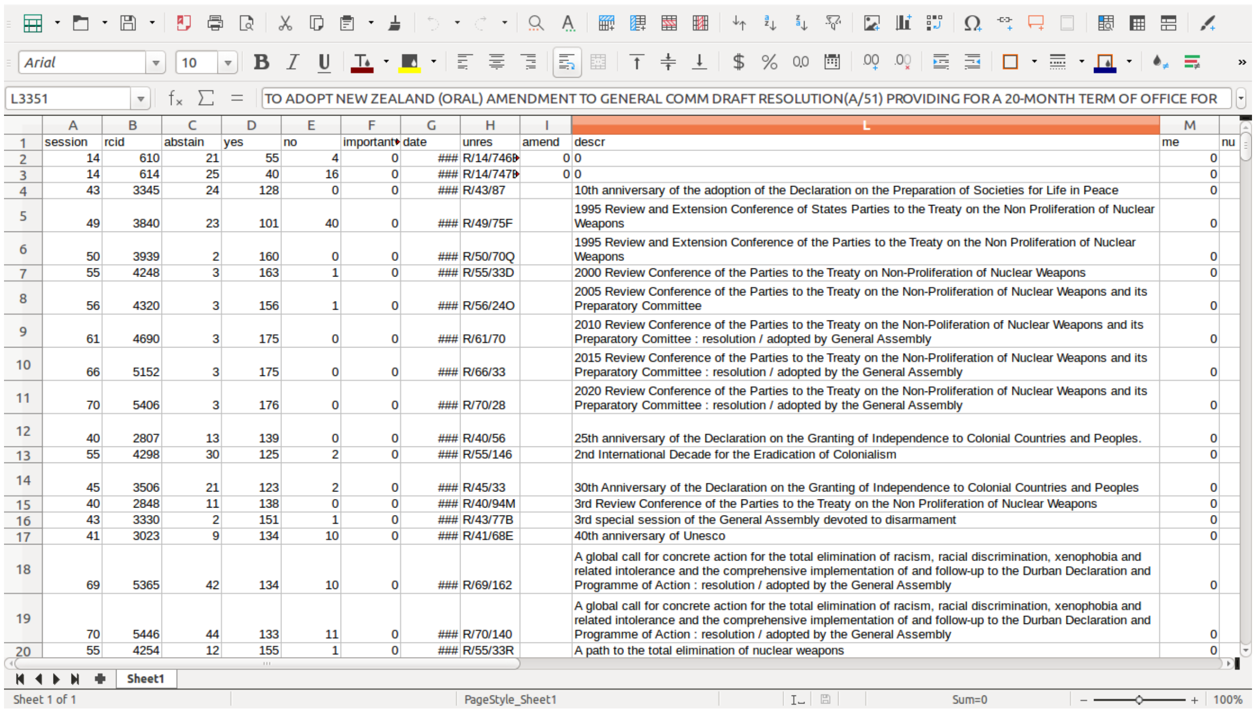
3.1. GAME DESCRIPTION

We are considering a 2-player game to model two countries that tend to be competitive with each other.

Every country representative has 2 moves, that is to either be in favor of the country (saying Yes to them: Y) or to go against them by not being in their favor (saying No to them: N).

Hence each group of countries has 2 moves: Y or N.

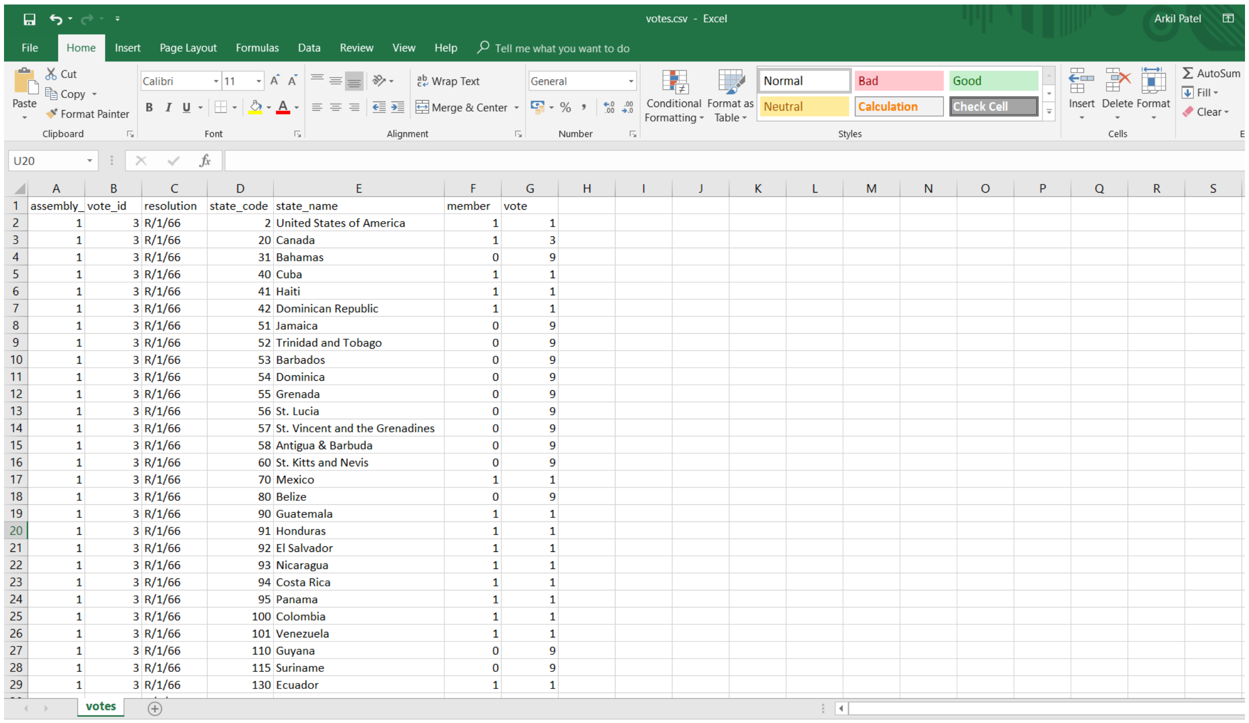
3.2. DATASET



L3351 TO ADOPT NEW ZEALAND (ORAL) AMENDMENT TO GENERAL COMM DRAFT RESOLUTION(A/51) PROVIDING FOR A 20-MONTH TERM OF OFFICE FOR

session	rcid	abstain	yes	no	important	date	unres	amend	descr	me
14	610	21	55	4	0	###	R/14/746	0	0	0
14	614	25	40	16	0	###	R/14/747	0	0	0
43	3345	24	128	0	0	###	R/43/87		10th anniversary of the adoption of the Declaration on the Preparation of Societies for Life in Peace	0
49	3840	23	101	40	0	###	R/49/75F		1995 Review and Extension Conference of States Parties to the Treaty on the Non Proliferation of Nuclear Weapons	0
50	3939	2	160	0	0	###	R/50/70Q		1995 Review and Extension Conference of the Parties to the Treaty on the Non Proliferation of Nuclear Weapons	0
55	4248	3	163	1	0	###	R/55/33D		2000 Review Conference of the Parties to the Treaty on Non-Proliferation of Nuclear Weapons	0
56	4320	3	156	1	0	###	R/56/24O		2005 Review Conference of the Parties to the Treaty on the Non-Proliferation of Nuclear Weapons and its Preparatory Committee	0
61	4690	3	175	0	0	###	R/61/70		2010 Review Conference of the Parties to the Treaty on the Non-Poliferation of Nuclear Weapons and its Preparatory Comittee : resolution / adopted by General Assembly	0
66	5152	3	175	0	0	###	R/66/33		2015 Review Conference of the Parties to the Treaty on the Non-Proliferation of Nuclear Weapons and its Preparatory Committee : resolution / adopted by the General Assembly	0
70	5406	3	176	0	0	###	R/70/28		2020 Review Conference of the Parties to the Treaty on the Non-Proliferation of Nuclear Weapons and its Preparatory Committee : resolution / adopted by the General Assembly	0
40	2807	13	139	0	0	###	R/40/56		25th anniversary of the Declaration on the Granting of Independence to Colonial Countries and Peoples.	0
55	4298	30	125	2	0	###	R/55/146		2nd International Decade for the Eradication of Colonialism	0
45	3506	21	123	2	0	###	R/45/33		30th Anniversary of the Declaration on the Granting of Independence to Colonial Countries and Peoples	0
40	2848	11	138	0	0	###	R/40/94M		3rd Review Conference of the Parties to the Treaty on the Non Proliferation of Nuclear Weapons	0
43	3330	2	151	1	0	###	R/43/77B		3rd special session of the General Assembly devoted to disarmament	0
41	3023	9	134	10	0	###	R/41/68E		40th anniversary of Unesco	0
69	5365	42	134	10	0	###	R/69/162		A global call for concrete action for the total elimination of racism, racial discrimination, xenophobia and related intolerance and the comprehensive implementation of and follow-up to the Durban Declaration and Programme of Action : resolution / adopted by the General Assembly	0
70	5446	44	133	11	0	###	R/70/140		A global call for concrete action for the total elimination of racism, racial discrimination, xenophobia and related intolerance and the comprehensive implementation of and follow-up to the Durban Declaration and Programme of Action : resolution / adopted by the General Assembly	0
55	4254	12	155	1	0	###	R/55/33R		A path to the total elimination of nuclear weapons	0

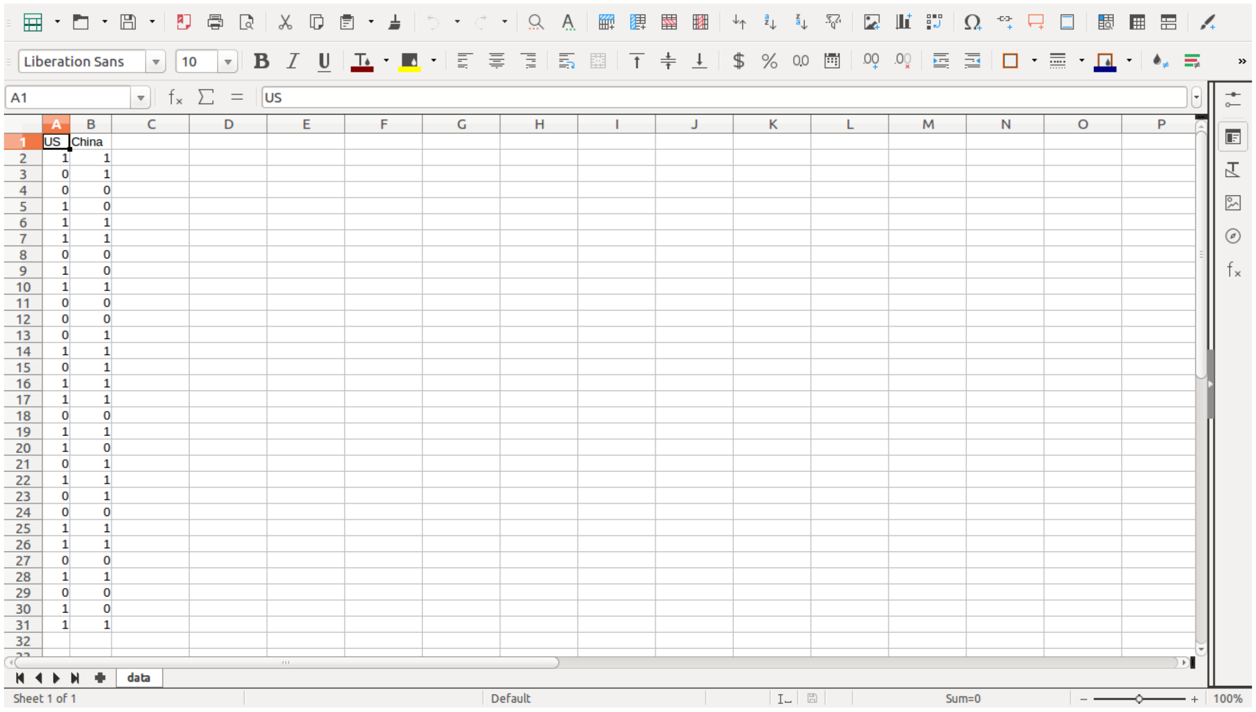
Fig 1.



votes.csv - Excel

assembly_	vote_id	resolution	state_code	state_name	member	vote
1	3	R/1/66	2	United States of America	1	1
1	3	R/1/66	20	Canada	1	3
1	3	R/1/66	31	Bahamas	0	9
1	3	R/1/66	40	Cuba	1	1
1	3	R/1/66	41	Haiti	1	1
1	3	R/1/66	42	Dominican Republic	1	1
1	3	R/1/66	51	Jamaica	0	9
1	3	R/1/66	52	Trinidad and Tobago	0	9
1	3	R/1/66	53	Barbados	0	9
1	3	R/1/66	54	Dominica	0	9
1	3	R/1/66	55	Grenada	0	9
1	3	R/1/66	56	St. Lucia	0	9
1	3	R/1/66	57	St. Vincent and the Grenadines	0	9
1	3	R/1/66	58	Antigua & Barbuda	0	9
1	3	R/1/66	60	St. Kitts and Nevis	0	9
1	3	R/1/66	70	Mexico	1	1
1	3	R/1/66	80	Belize	0	9
1	3	R/1/66	90	Guatemala	1	1
1	3	R/1/66	91	Honduras	1	1
1	3	R/1/66	92	El Salvador	1	1
1	3	R/1/66	93	Nicaragua	1	1
1	3	R/1/66	94	Costa Rica	1	1
1	3	R/1/66	95	Panama	1	1
1	3	R/1/66	100	Colombia	1	1
1	3	R/1/66	101	Venezuela	1	1
1	3	R/1/66	110	Guyana	0	9
1	3	R/1/66	115	Suriname	0	9
1	3	R/1/66	130	Ecuador	1	1

Fig 2.



US	China
1	1
0	1
0	0
1	0
1	1
1	1
0	0
1	0
1	1
0	0
0	0
0	1
1	1
0	1
1	1
1	1
0	0
1	1
1	0
0	1
1	1
0	1
0	0
1	1
1	1
0	0
1	1
0	0
1	0
1	1

Fig 3

The dataset we found most relevant for our project work was UN Voting dataset on Kaggle. But we had to modify it for getting the information relevant to our project. So, from the dataset 'DescriptionsUN1-72.csv' shown in Fig1. we searched the 'descr' column and found the resolutions involving the two countries and their corresponding sessions. Then from the dataset 'votes.csv' shown in Fig2 we matched the session and found the vote of the respective other country either in favour of or against the first country. Using this information, we made our own dataset shown in Fig 3. Since our need for the dataset is only to initialise the values in the Q-Table, we didn't feel the need to get more data. The algorithm prediction in dynamic time happens independent of data.

4. PAYOFF MATRICES

For a voting session in UN, a particular country has two mindsets - Profit and Goodwill. Profit mindset is obvious which is to get higher profit than other countries hence not voting in the favor of other countries is the most optimal choice to make but this cannot be the case as the game is not played in only one session. Since the countries have to meet again in the next UN session, repercussions of being self-minded could be harmful to them itself. So, the players need to keep a Goodwill mindset as well. Goodwill mindset basically represents being supportive towards other countries and voting in their favor. We have quantified these mindsets as two payoff matrices.

The payoffs in each of the cells in the matrices are designed qualitatively that will be proven mathematically further.

4.1. GOODWILL MATRIX

$$\begin{array}{c|c|c|}
 & Y & N \\ \hline
Y & (1,1) & (1,-1) \\ \hline
N & (-1,1) & (-1,-1) \\ \hline
\end{array}$$

Qualitative Description of Payoffs:

In this matrix, the payoffs are assigned on the basis of whether a player is in favor or not. A player is awarded a ‘+1’ point if he wishes for the other player’s good that is by saying Yes to him and a ‘-1’ if he says No.

Consider YY Payoffs that comply with the decided rule for Goodwill Matrix, that is both the players favor each other and hence both get ‘+1’ each. So for NN, both the players get ‘-1’ each.

For YN Payoffs, P1 is awarded ‘+1’ and P2 is awarded ‘-1’ for obvious reasons that are P1 has favored P2 hence it gets a point and P2 did not do the same, hence it doesn’t. Similarly, the payoffs of NY can be explained.

4.2. PROFIT MATRIX

	Y	N
Y	(0,0)	(-2,2)
N	(2,-2)	(0,0)

Qualitative Description of Payoffs:

The Payoffs in this matrix are decided on the basis of profit/loss. Profit counts as a ‘+1’ and loss counts as a ‘-1’.

Consider the YY Payoff which is (0,0). When the Player1 (P1) favors Player2 (P2) relatively P1 loses its profitable situation hence a P1 is assigned ‘-1’ by its own move but at the same time P2 also favors P1 which assigns a ‘+1’ to P1. This makes the total payoff for P1 to be 0 and similarly for P2. A similar explanation holds for the NN payoffs.

For the YN Payoffs the P1 is assigned a ‘-2’ and P2 is assigned a ‘+2’. The reason being that P1 says Yes to P2, hence lowering its position in the competition (so a ‘-1’ for P1) whereas P2 is betraying P1 by saying No to it which further assigns a ‘-

1', making the total '-2'. This also describes the payoff of P2 which is '2' in this case. A similar explanation holds for the NY case.

4.3. BAYESIAN VERIFICATION

In order to verify the payoffs obtained qualitatively we tried modelling the game as a Bayesian game and the payoff matrices thus obtained were:

Goodwill world:

	YY	YN	NY	NN
Y	1	1	1	1
N	-1	-1	-1	-1

Profit World:

	YY	YN	NY	NN
Y	0	2q-2	-2q	-2
N	2	2q	2-2q	0

(here q is the probability of player 2 playing in the goodwill world).

So clearly, we can see from the payoff matrices that the equilibrium is (YN, YN). And it is justified because the best option for both the players would be to vote yes in the goodwill world and vote no in the profit world. So, the payoff values we have considered seems justified.

4.4. REWARD MATRIX

The two Payoff matrices are combined to give a Reward Matrix that eventually will depict both mindsets together.

The dynamic way to combine both is to assign a pair of complementary functions with one for each of the matrices such that one increases and the other one decreases and vice versa.
Let w be the variable to control the complementary weight functions for the matrices.

Let f(w) and g(w) be the two functions assigned to the Goodwill Matrix and the Profit Matrix respectively such that for a particular range of w when f(w) increases, g(w) decreases and when f(w) decreases, g(w) increases.

To learn the pairs of functions, we tried non-linear functions like the exponential functions (e^w, e^{-w} for $-1 \le w \le 1$) or (2^w, 2^{-w} for $-1 \le w \le 1$) or (w^2, $(1-w)^2$ for $-1 \le w \le 1$) or linear functions (w, 1-w for $0 \le w \le 1$) and so on and so forth.

Analyzing the pattern, the Reward Matrix follows based on different functions, we concluded that all of them behave in a similar way to give the best response finally. Due to which sticking to the easiest option that was the linear function (w, 1-w for $0 \le w \le 1$) would be the optimal choice.

$$\begin{array}{c|c|c|}
 & \text{Y} & \text{N} \\
\hline
\text{Y} & (w,w) & (3w-2, 2-3w) \\
\hline
\text{N} & (2-3w, 3w-2) & (-w,-w) \\
\hline
\end{array}$$

Best Response obtained from different values of w:

1. For $0 \leq w \leq 1/3$, NN is the equilibrium
2. For $1/3 < w < 2/3$ YY, YN, NY, NN all can be the equilibrium
3. For $2/3 \leq w \leq 1$, YY is the equilibrium

Based on the dynamic value of ‘w’, we will find out what is the best response for the two players.

5. ALGORITHMS

For the purpose of obtaining the best response of a country dynamically, i.e. at any given point in time, we consider the history of its association with the other player meanwhile keeping in mind the fact that its actions will have consequences in the future. For this purpose, we developed two different algorithms which are distinct in their ideology.

5.1. The Q-Learning Algorithm

The Q-Learning algorithm works on the concepts of Reinforcement Learning. In this section we shall explain the underlying concepts of the algorithm and what motivated us to use it.

Reinforcement Learning is a field of study which seeks to understand the dynamics behind decision making. It seeks to enable independent agents to make intelligent decisions for them. A lot of research is being carried out to implement Reinforcement Learning algorithms to make agents capable of exhibiting Artificial Intelligence.

In our simulation, we seek to develop a model that is able to give the best response that a player should move given a set of conditions and history. Hence, at the core, we are just trying to make an intelligent and educated decision of which strategy to take. Thus, it was only natural that we explored and implemented Reinforcement learning concepts and algorithms to carry out our simulation.

5.1.1. The Q-Value, the Q-Table and the Policy

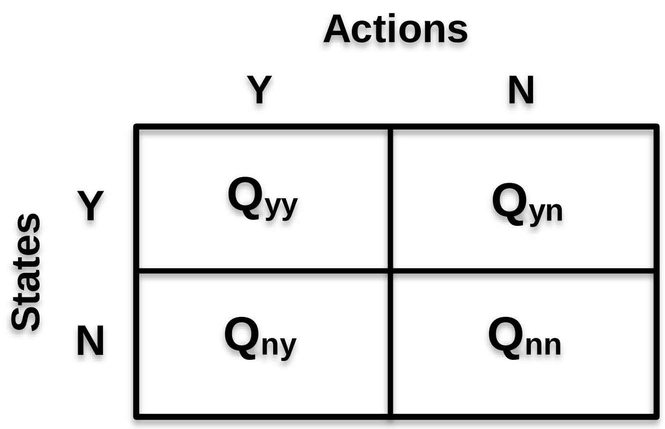
The Q-Value is a value that denotes long term return of taking any specific action in any given state by following some policy 'π'. It is denoted as '$Q_\pi(S,a)$' where 'S' denotes the state and 'a' denotes the action taken in the state 'S'.

The Policy 'π' is nothing but a mapping of states to actions. It is used to determine the next action based on current state.

The Q-Table is a table that holds the Q-Values of all the State-action pairs.

Our objective is to develop a policy that gives us the best response to a given strategy of the opponent player.

We consider all the possible opponent moves to be the states and all possible player strategies to be the actions. Hence, our policy will tell us that given a particular move being undertaken by the opponent, what would be the best strategy that the player must take in order to maximize long term rewards. Thus, our Q-table would look something like this:



States \ Actions	Y	N
Y	Q_{yy}	Q_{yn}
N	Q_{ny}	Q_{nn}

The Policy followed to determine the Best Response is:
For any given state S, the action to be chosen A is given by:

$$A = max\ (Q_{Sa1}, Q_{Sa2}, \ldots, Q_{SaN})$$

Where $Q_{Sa'}$ denotes the Q-Value obtained by taking an action a' in a state S and N is the number of actions.

Hence in our case,
$BR_1(A') = max\ (Q_{A'Y}, Q_{A'N})$
Where A' is the strategy of the opponent player.

The next section talks about how exactly the Q-Values are updated.

5.1.2. The Q-learning Equation

The Q-learning Equation is obtained by iteratively approximating the value obtained by Bellman equation. The equation is:

$$Q_{t+1}(S_t, a_t) = Q_t(S_t, a_t) + \alpha * (r_{t+1} + \gamma * max_a(Q_t(S_{t+1},a)) - Q_t(S_t, a_t))$$

Where ‘α’ is the learning rate and ‘γ’ is the discount factor. Both these are hyperparameters and can be tuned accordingly.

Let us try to understand the equation. The ‘r_{t+1}’ term denotes the reward that the player will get by choosing that action in that state. This is nothing but the reward value obtained from the reward matrix. Hence every time, the equation is updating the Q-value by using the reward as well as the maximum Q-value that it shall get in the future if it pursues this action. This takes care of both our points of concern – (i) The reward term takes care of past history since it makes use of the parameter ‘w’ which is dependent on history and (ii) The next state Q-Value term takes care of the future.

5.1.3. Control of the weight ‘w’

For every time we update the Q-table, we need a value of ‘w’ which signifies what nature the player should take – (i) Goodwill or (ii) Profit. We declared four variables that represent the history of moves:

‘yy’ – The number of times both players opted ‘y’
‘yn’ – The number of times opponent opted ‘n’ and player opted ‘y’
‘ny’ – The number of times opponent opted ‘y’ and player opted ‘n’
‘nn’ – The number of times both players opted ‘n’

Qualitatively we can see that if ‘yn’ is more than ‘ny’, it means that the opponent has betrayed the player a greater number of times than the player has betrayed the opponent. Hence in this case we must make the player’s nature to tend towards profit. For this purpose, we increase the value of ‘w’ with a quantity ‘*λ*(yn – ny)*’ where ‘λ’ is a hyperparameter. Such a quantity makes sure that if the difference is more, then the value must also increase more rapidly.

Similarly, we can see that if ‘ny’ is more than ‘yn’, it means that the player has betrayed the opponent a greater number of times than the opponent has betrayed the player. Hence in this case we must make the player’s nature to tend towards goodwill. For this purpose, we decrease the value of ‘w’ with a quantity ‘*λ*(ny - yn)*’ where ‘λ’ is a hyperparameter. Such a quantity makes sure that if the difference is more, then the value must also decrease more rapidly.

5.1.4. The Algorithm

The following are the various stages of the algorithm:

1. The Dataset is loaded.
2. The variables are initialized. The parameter ‘w’ is initialized to 0.5 (denoting a neutral nature in the beginning).
3. Reading from the data, the opponent’s moves are taken as states and player’s moves as actions.
4. The reward is calculated and the Q-Value in the Q-Table is updated.
5. The ‘w’ value is updated according to the history.
6. Once all the data has been used once, the opponent moves are predicted using the probability estimation.
7. These moves are passed as states and the best action is selected according to the policy.
8. The reward is calculated and the Q-Value in the Q-Table is updated.
9. The ‘w’ value is updated according to the history.
10. At any point of time, the action outputted by the algorithm can be taken to be the dynamic best response.

5.1.5. Results

The algorithm is successful in outputting an action given a set of history. The values of ‘yn’ and ‘ny’ are always very close implying that the algorithm is not allowing any player to keep betraying the other without retaliation.

It is observed that the players get locked in a cycle of betraying and getting betrayed thereby ensuring that no player has the clear advantage over other.

5.2. The Dynamic Reward Algorithm

The dynamic reward algorithm is entirely similar to the Q-Learning algorithm except that now there is no Q-Learning involved. Instead, at every point of time, the player just chooses the action that gives it maximum payoff. This results in this algorithm only focusing on the history of association and not the future possibility of rewards. The calculation of reward and control of weight ‘w’ remains the same.

5.2.1. The Algorithm

The following are the various stages of the algorithm:

1. The Dataset is loaded.
2. The variables are initialized. The parameter ‘w’ is initialized to 0.5 (denoting a neutral nature in the beginning).
3. Reading from the data, the opponent’s moves are taken as states and player’s moves as actions.
4. The reward is calculated.
5. The ‘w’ value is updated according to the history.
6. Once all the data has been used once, the opponent moves are predicted using the probability estimation.
7. These moves are passed as states and the best action is selected according to the maximum reward given the opponent move and the value of ‘w’.
8. The ‘w’ value is updated according to the history.
9. At any point of time, the action outputted by the algorithm can be taken to be the dynamic best response.

5.2.2. Results

The algorithm is successful in outputting an action given a set of history. The values of ‘yn’ and ‘ny’ are always very close implying that the algorithm is not allowing any player to keep betraying the other without retaliation.

A difference in result as compared to the Q-Learning algorithm is that it does not get locked in the betrayal and getting betrayed cycle.

6. CONCLUSION

In this project, we have learned about Repeated Games and the various strategies used to solve them. We took a particular application case – The United Nations Voting Scenario to develop a learning algorithm that can find the best response in dynamic time. We modelled the situation considering two competitive countries exhibiting two natures – (i) Goodwill and (ii) Profit. We then qualitatively decided the payoffs of the decisions involved. We developed two algorithms which can determine the best response strategy using the payoffs and the history of association.

The Q-Learning Algorithm applies the concepts of Reinforcement Learning to make intelligent decision as to which strategy to take in dynamic time. The long-term reward obtained by taking a strategy was characterized by the parameter 'w' which influenced the reward matrix thereby indicating how much a player would benefit. The algorithm caused the players to become locked in a sequence of betrayal and getting betrayed and was successful in assuring that the player was not doing worse than the opponent.

The Dynamic Reward Algorithm only takes the strategy that gives the player the maximum reward at any given point of time. It considered history of association since it was dependent on the parameter 'w' but did not consider future long-term benefits. The algorithm did not get locked in the betrayal sequence like the Q-Learning algorithm and was successful in assuring that the player was not doing worse than the opponent.

There is a lot of scope for improvement in our approach as mentioned in the future scope. The approaches that we have developed may be used in other real-life application scenarios such as Bargaining, Market strategy decision making, etc. We also explored the scope for future work in modelling the Environment Pollution Situation as a repeated game as will be explained in the next section.

7. FUTURE SCOPE

7.1. Improvements in the UN Voting simulation

Various improvements could be done in the project later on in the future.

1. We have considered a two-country competitive model. Later on, we can extend the model to multiple countries which would make the simulation more realistic.

2. Here while filtering the dataset we have not considered the percentage of votes, we have considered the majority of votes. So, let's say if US has voted majority of the times in favour of China so we have considered it to be a yes, and not the percentage of the number of times it voted yes. Later on, we could consider the percentage also.

3. We have considered the reward function to be linear. Later on, we can think of a more complex reward function.

4. We have just considered two natures being exhibited – Goodwill and Profit. More qualitative natures and features may be added in order to mimic real life scenario.

We believe that with the above-mentioned changes, our model can be extended to work accurately on real life situations involving multiple entities engaged in an infinite horizon Repeated Game.

7.2. Environment Pollution Control – A Case Study

Environment Pollution has become a major source of concern in the past few decades. Cities like New Delhi are on the verge of becoming uninhabitable in the very near future. A lot of studies have been carried out to suggest solutions to curb this menace.

While exploring the possible applications of repeated games in real life, we thought of the case of environmental pollution. In this section, we shall briefly talk about how Environmental Pollution is related to Game theory and propose a small model similar to our voting scenario.

If we look at the main causes of environmental pollution, the first thing that comes to our mind is Industries. Industries are the major sources of air and water pollution in any country. However, we cannot simply shut them all down as they are an essential part of our life. There are ways and methods of controlling the pollutants emitted by any industry but these are costly and instead of contributing towards the profit of the industry, they may cause the cost of maintenance to go up thereby leading to losses. Hence there is no encouragement for industries to use these ‘Pollution Control Methods’.

The current approach used by the Government to make the industries use these Pollution Control Methods is to routinely carry out surprise inspections and if the industry is not found to keep the required standards, then it is fined some amount of money.

We believe that this approach is not very effective in curbing environmental pollution. The problems with this approach are listed as:

1. There is no incentive for the industry to employ the pollution control methods. They only do so because they are forced by the government.
2. In a country like India, there are millions of industries and hence it is impossible to keep them all in check by doing surprise inspection of all of them.
3. Some level of Corruption may be involved wherein the cost of paying the inspector is much less than the cost of using the environment control methods.

In order to address these issues, we turn towards Game Theory. This entire situation can be thought of as a repeated game between two players – Industry and Government. Here by ‘Industry’ we are talking about one particular Industry. The

Government can be seen as playing multiple repeated games with all the industries in the country.

The Industry has the strategies as – (i) Employ Pollution Control Methods, (ii) Don’t employ Pollution Control Methods and (iii) Shut down all operations. The Government has the strategies – (i) Reward the industry and (ii) Punish the industry. Here reward is not necessarily monetary; It may be anything beneficial to the industry such as favorable policy changes, lowering taxes, etc. The punishment also is not necessarily a fine and may include sanctions being placed on the industry.

At any point of time, based on the Government’s past behavior and its affect on the industry, the industry can quantitatively decide the best strategy that it should take. Similarly, by examining the industry’s past behavior, the government can decide its own course of action.

This model is very similar to the one proposed by us for the UN Voting scenario. Firstly, they involve two players playing an infinite horizon repeated game. Secondly, both the players have a history of association with the other and knows the past outcomes of each other’s actions. Thirdly, if both the industry and the Government want to succeed in the long run, they need to give thought to the possible outcomes of the strategies they want to take and how it will benefit them. Lastly, a dual-nature can be seen being exhibited by the players; Both the players have a nature of Goodwill and Profit. The industry exhibits profit nature wherein it will prefer NOT to employ the Pollution Control Methods in order to maximize its monetary profit but it also exhibits a goodwill nature which will prefer to keep the pollution under control so as to keep good relations with the government which can help it in with regards to Government Policy implementation, taxes, etc. The Government has a goodwill nature towards the environment and will prefer to enforce fines and punishment on industries which do not follow the rules but it also has a profit nature wherein it understands the importance of industries for the country’s economy and hence wouldn’t wish to antagonize them with punishments to avoid them shutting down their operations altogether.

Such a model addresses the various issues of the currently employed model. The possibility of rewards from the government provides the industries with an actual incentive to show a positive attitude towards the environment which in itself may be enough to actually control the pollution levels.

8. REFERENCES

1. Osborne, M. J., & Rubinstein, A. (1994). *A course in game theory*. MIT press.
2. Agrawal, A., & Jaiswal, D. (1981). When machine learning meets ai and game theory.
3. Wolpert, D. H. (2005). Predictive game theory.
4. Voeten, E., & Merdzanovic, A. (2009). United Nations general assembly voting data.
5. Cesa-Bianchi, N. (2011, September). The game-theoretic approach to machine learning and adaptation. In *ICAIS* (p. 1).
6. Huang, P., & Sycara, K. (2003, July). Multi-agent learning in extensive games with complete information. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems* (pp. 701-708). ACM.
7. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
8. Dataset Reference: https://www.kaggle.com/unitednations/general-assembly#votes.csv

Code - Q-Learning Algorithm

December 6, 2018

```
In [1]: import numpy as np
        import pandas as pd
        import random as rd

In [2]: data = pd.read_csv("data.csv", sep=',')

In [3]: data.head()

Out[3]:    US  China
        0   1      1
        1   0      1
        2   0      0
        3   1      0
        4   1      1

In [4]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
US       30 non-null int64
China    30 non-null int64
dtypes: int64(2)
memory usage: 560.0 bytes


In [5]: # Inititializations
        train_num = 29
        test_num = 100
        y = 0
        n = 1
        alpha = 0.5 # Learning Rate
        gamma = 0.2 # Discount Rate
        yy = 0
        ny = 0
        yn = 0
        nn = 0
        w = 0.5 # Weight for payoffs
        lamda = 0.01 # Rate for updation of w
        qtable = [[0 for x in range(2)] for y in range(2)] # Q-table
```

```
In [6]: # Reward Matrix
        def reward(state,action):

            if state == y: # Opponent move is y
                if action == y:
                    return w
                else:
                    return (2-3*w)
            else: # Opponent move is n
                if action == y:
                    return (3*w-2)
                else:
                    return -w
```

```
In [7]: def opponent_Move():

            prob_y = (yy+ny/yy+ny+yn+nn)
            prob_n = (yn+nn/yy+ny+yn+nn)

            Est_y = max(w,3*w-2)*prob_y
            Est_n = max(2-3*w,-w)*prob_n

            if Est_y-Est_n > 0.1:
                return n;
            elif Est_n-Est_y > 0.1:
                return y;

            if Est_y>Est_n:
                return y
            elif Est_n>Est_y:
                return n
            else:
                return rd.randint(0,1)
```

1 Train

```
In [8]: temp1 = 1
        tempyn = yn
        tempny = ny

        for i in range(train_num):

            state = data['China'][i]
            nextstate = data['China'][i+1]
            action = data['US'][i]

            r = reward(state,action)
            qtable[state][action] = qtable[state][action] + \
            alpha*(r + gamma*max(qtable[nextstate][y],qtable[nextstate][n]) - \
                   qtable[state][action])

            if action == y and state == y:
                yy += 1
            elif action == y and state == n:
                yn += 1
            elif action == n and state == y:
                ny += 1
            else:
                nn +=1

            if(tempyn - tempny != yn - ny):
                temp1 = 1

            tempyn = yn
            tempny = ny

            if temp1 == 1:
                if yn>ny:
                    w = w - (lamda*(yn-ny))
                elif yn<ny:
                    w = w + (lamda*(ny-yn))
                temp1 = 0

        print("yy = ",yy," ny = ",ny," yn = ",yn," nn = ",nn)
```

```
yy =  8  ny =  4  yn =  5  nn =  12
```

2 Independent

```
In [9]: for i in range(test_num):

            state = nextstate
            nextstate = opponent_Move()

            if qtable[state][y] > qtable[state][n]:
                action = y
            elif qtable[state][n] > qtable[state][y]:
                action = n
            else:
                action = rd.randint(0,1)

            r = reward(state,action)
            qtable[state][action] = qtable[state][action] + \
            alpha*(r + gamma*max(qtable[nextstate][y],qtable[nextstate][n]) - \
                   qtable[state][action])

            if action == y and state == y:
                yy += 1
            elif action == y and state == n:
                yn += 1
            elif action == n and state == y:
                ny += 1
            else:
                nn +=1

            if(tempyn - tempny != yn - ny):
                temp1 = 1

            tempyn = yn
            tempny = ny

            if temp1 == 1:
                if yn>ny:
                    w = w - (lamda*(yn-ny))
                elif yn<ny:
                    w = w + (lamda*(ny-yn))
                temp1 = 0

        print("yy = ",yy," ny = ",ny," yn = ",yn," nn = ",nn)
```

```
yy =  8  ny =  52  yn =  54  nn =  15
```

Code - Dynamic Reward

December 6, 2018

```
In [1]: import numpy as np
        import pandas as pd
        import random as rd

In [2]: data = pd.read_csv("data.csv", sep=',')

In [3]: data.head()

Out[3]:    US  China
        0   1      1
        1   0      1
        2   0      0
        3   1      0
        4   1      1

In [4]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
US       30 non-null int64
China    30 non-null int64
dtypes: int64(2)
memory usage: 560.0 bytes


In [5]: # Inititializations
        train_num = 29
        test_num = 100
        y = 0
        n = 1
        yy = 0
        ny = 0
        yn = 0
        nn = 0
        w = 0.5 # Weight for payoffs
        lamda = 0.01 # Rate for updation of w
```

```
In [6]: # Reward Matrix
        def reward(state,action):

            if state == y: # Opponent move is y
                if action == y:
                    return w
                else:
                    return (2-3*w)
            else: # Opponent move is n
                if action == y:
                    return (3*w-2)
                else:
                    return -w
```

```
In [7]: def opponent_Move():

            prob_y = (yy+ny/yy+ny+yn+nn)
            prob_n = (yn+nn/yy+ny+yn+nn)

            Est_y = max(w,3*w-2)*prob_y
            Est_n = max(2-3*w,-w)*prob_n

            if Est_y-Est_n > 0.1:
                return n;
            elif Est_n-Est_y > 0.1:
                return y;

            if Est_y>Est_n:
                return y
            elif Est_n>Est_y:
                return n
            else:
                return rd.randint(0,1)
```

1 Train

```
In [8]: temp1 = 1
        tempyn = yn
        tempny = ny

        for i in range(train_num):

            state = data['China'][i]
            nextstate = data['China'][i+1]
            action = data['US'][i]

            if action == y and state == y:
                yy += 1
            elif action == y and state == n:
                yn += 1
            elif action == n and state == y:
                ny += 1
            else:
                nn +=1

            if(tempyn - tempny != yn - ny):
                temp1 = 1

            tempyn = yn
            tempny = ny

            if temp1 == 1:
                if yn>ny:
                    w = w - (lamda*(yn-ny))
                elif yn<ny:
                    w = w + (lamda*(ny-yn))
                temp1 = 0

        print("yy = ",yy," ny = ",ny," yn = ",yn," nn = ",nn)
```

```
yy =  8  ny =  4  yn =  5  nn =  12
```

2 Independent

```
In [9]: for i in range(test_num):

            state = nextstate
            nextstate = opponent_Move()

            if reward(state,y) > reward(state,n):
                action = y
            elif reward(state,n) > reward(state,y):
                action = n
            else:
                action = rd.randint(0,1)

            if action == y and state == y:
                yy += 1
            elif action == y and state == n:
                yn += 1
            elif action == n and state == y:
                ny += 1
            else:
                nn +=1

            if(tempyn - tempny != yn - ny):
                temp1 = 1

            tempyn = yn
            tempny = ny

            if temp1 == 1:
                if yn>ny:
                    w = w - (lamda*(yn-ny))
                elif yn<ny:
                    w = w + (lamda*(ny-yn))
                temp1 = 0

        print("yy = ",yy," ny = ",ny," yn = ",yn," nn = ",nn)
```

```
yy =  30  ny =  44  yn =  41  nn =  14
```