Variants of Q-Learning in Nim

Strategy Efficiency

Vaibhav Gupta

Adlai E. Stevenson High School
Grade 12

Table of Contents

Abstract 1

Safety Sheet 2

Acknowledgements 5

Purpose 6

Hypothesis 6

Variables 7

Review of Literature 9

Reinforcement Learning 9

Q-Learning 11

Nim 13

Materials 16

Procedure 16

Results 17

Discussion 20

Conclusion 21

Works Cited 24

Appendices I

Appendix A: Complete Dataset I

Acknowledgements

I would like to thank my parents for supporting me throughout this project, as well as my sponsors Mrs. Palffy and Mr. Robinson for their willingness to sacrifice their time to allow me to pursue this project.

Purpose

The purpose of this experiment is to determine the effect of varying learning rate and discount factor pairs on Q-learning efficiency in the game of Nim. Specifically, a Q-learning algorithm was trained with distinct pairs of learning rates and discount factors through self-play before playing against an expert algorithm, where algorithm efficiency was measured by the percentage of games won.

Although machine learning has existed for a long time, recent computational advancements and the rise of big data have increased the need for large-scale data handling and manipulation. While machine learning is currently used in several applications such as social media, virtual assistants, and malware filtering, it also has implications in nearly any data-dependent industry such as medicine, retail, and finance. Thus, the results from this experiment can verify the efficiency of Q-learning at a smaller scale and its applicability to other similar tasks.

Hypothesis

If a Q-learning algorithm is trained against itself with 36 distinct pairs of learning rates and discount factors ranging from 0-1 in 0.2 increments for 200,000 games of Nim before playing 200,000 games against an expert algorithm, then the algorithms with a pair value above (α=0.6, γ=0.6) will win an equal percentage of games (50%) against the expert algorithm.

Q-learning generally functions optimally with a low learning rate (α=0.1-0.2) and a high discount factor (γ=0.8-0.9) to allow the algorithm to converge in a stochastic environment. However, since the environment remains entirely deterministic, a moderate to high learning rate with a similar discount factor should result in the quickest learning.

Variables

Independent Variable: The independent variable in this experiment is the learning rate and discount factor pair (α, γ), with each element varying in value from 0-1 in 0.2 increments for a total of $6^2 = 36$ distinct pairs.

Dependent Variable: The dependent variable in this experiment was the percentage of wins (%) of the Q-learning algorithm against the expert algorithm in 200,000 games of Nim after training for 200,000 games through self-play.

As the game of Nim is completely solved, there exists a strategy such that the second player to move can always win if they follow the optimal moves. Thus, with the first player alternating between every game, the Q-learning algorithm can theoretically win a maximum of 50% of games.

Positive Control: The positive control in this experiment is the expert algorithm against an identical expert algorithm. As the expert algorithm always plays the optimal strategy when it moves second, both algorithms are expected to win 50% of the games.

Negative Control: The negative control in this experiment is the expert algorithm against a random algorithm. As the random algorithm always selects a random move to play, it is expected to win nearly 0% of the games.

Constants: The implementation of the expert, random, and Q-learning algorithm remained constant, with only the learning rate and discount factor coefficients being manipulated. The Q-learning algorithm was trained for 200,000 games before playing against the expert algorithm for 200,000 games, and then repeated 10 times for each pair. Similarly, the game conditions remained constant, with the initial board configuration consisting of 4 piles of 1, 3, 5, and 7 objects respectively, following the rules of misère Nim.[1]

[1] Nim is a two-player game in which players take alternating turns removing objects from multiple piles. A player can remove any number of objects during their turn, but only from a single pile. In the misère variant of Nim, the losing condition is to remove the last object.

Review of Literature

In the modern age, the rise of big data, advanced algorithms, and improved computational power have prompted technological growth, with the emergence of machine learning (ML) as a consequence. Machine learning is currently being developed and integrated in nearly every industry, from self-driving cars to chess-playing computers. Algorithm integration plays a vital role within the function of machine learning, and algorithms such as Q-learning form a basis for more complex types of learning that enable machines to learn from experience and adapt to constantly shifting inputs. This allows machines to imitate human reasoning that both complements and augments human abilities, having the potential to automate tasks with greater efficiency in many industries. Through the implementation of algorithms, machines can be trained to not only process large volumes of data but to produce their own as well (Machine Learning: What it is and why it matters).

Reinforcement Learning

Reinforcement learning is a particular area of machine learning that seeks to maximize the reward given a specific circumstance. By defining the rewards, an algorithm can find the optimal path or behavior under the given conditions simply through exploration. As opposed to other methods such as supervised learning, which requires a preexisting database to be labeled manually, reinforcement learning optimizes its function by learning from experience (Bajaj, 2018).

In order to fully understand the basis of reinforcement learning, several key terms and concepts must be recognized. First, an agent is simply the algorithm of a program which executes a set of possible actions (A) within its environment. The configuration of the agent relative to other factors within the environment at any given instant is known as its state (S), which may include attributes such as position, movement, etc. Every action that the agent performs within its environment determines its next state and the corresponding reward (R), with the environment acting as a function of the agent's current state and action. A reward is the immediate or delayed feedback an agent receives from its environment that evaluates the success or failure of a given action, which influences the actions of the agent in the future (Ashraf, 2010).

The environment represents an unknown function that maps the agent's current state and action to its next state and reward, while the agent is a known function that maps the new state and reward into the next action. This results in a continuous feedback loop, in which each time step is a state-action pair. As such, reinforcement learning is a goal-oriented process in which an agent attempts to learn sequences of actions that maximize its objective function, effectively achieving its goal. Following every iteration, the agent attempts to approximate the environmental function in order to maximize the potential reward (A Beginner's Guide to Deep Reinforcement Learning, n.d.).

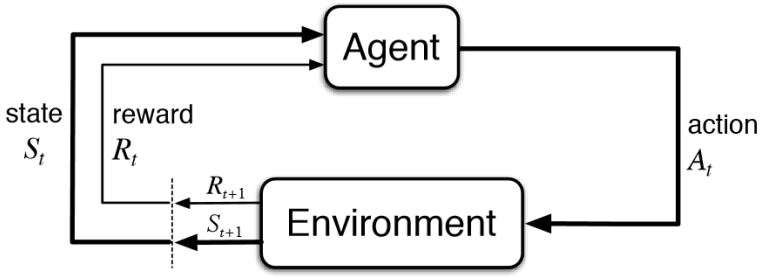

Figure 1. Basic agent-environment interaction in RL

Q-Learning

Q-learning is a method of reinforcement learning that seeks to estimate the value function of all possible state-action pairs. Specifically, Q-learning is a temporal difference algorithm, meaning it updates the estimated reward of each state-action pair at every time step, rather than updating the pairs after the final reward is received. Temporal difference algorithms can be further divided into on and off-policy learning, in which the former seeks to learn the value of the policy while the latter only seeks to estimate the return of a state-action pair regardless of the set of actions. As such, Q-learning can be classified as an off-policy temporal difference algorithm, as opposed to other variations such as SARSA which estimate the return of state-action pairs assuming continuation of the current policy (Eden, Knittel, Uffelen, n.d.).

Additionally, Q-learning requires understanding of more specific terms. The strategy that the agent applies is known as the policy (π), which dictates the agent's next action based upon its current state. It is responsible for mapping states to actions and performing those that maximize the potential reward, an approach known as epsilon-greedy. The influence of rewards on future actions is determined by the discount factor (γ)—a value of 0 ignores the potential of future

rewards entirely, while a value of 1 considers both immediate and future rewards equally. While the short-term return of a reward is known as R, the value (V) of a reward is its expected long-term return. The value of a reward is discounted, or underestimated, to account for its diminishing effect the farther into the future it occurs. The function Vπ(S) considers the value of every possible reward within a state and is defined as the long-term return of the current state S under policy π. The value of a reward can also be extended to an action-value or Q-value (Q), which considers both the state of the agent and its current action. Qπ(S, A) is defined as the long-term return of the current state S, taking an action A under policy π. Whereas a policy maps states to actions, Q extends an additional dimension by mapping state-action pairs to rewards (Huang, 2018).

An agent defines its goal through an objective function, which can be generally defined in reinforcement learning as follows:

$$\sum_{t=0}^{t=\infty} \gamma^t r(x(t), a(t)))$$

Figure 2. Objective function of Q-learning algorithm

in which *t* represents a time step, *x* is the state at a given time step, *a* is the action taken in a given state, *r* is the reward function, and *γ* is the discount factor. Therefore, the objective function is the summation of the reward function at each time step, with each subsequent reward discounted at a greater rate (A Beginner's Guide to Deep Reinforcement Learning, n.d.).

Most importantly, however, is how Q-learning updates the value of each state-action pair. Each state-action pair is usually stored in an array known as a Q-table, with the value of each pair usually known as the Q-value. The formula for updating each value is based on the formula below.

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

Figure 3. Q-learning formula

As shown, the value of each state-action pair is updated based on its current value, the learning rate, the discount factor, the future reward, and the estimated optimal future value. The future reward refers to the immediate reward as a result of the given action, while the estimated optimal value is the maximum value of the next state given any possible action. When a Q-learning algorithm achieves the optimal value for every possible state-action pair, it has successfully convergence, which allows the algorithm to perform the best possible action given a specific state. In a deterministic environment with no randomness, any non-zero learning rate and discount factor will allow the algorithm to converge eventually given a sufficient number of episodes, or trials. However, in a stochastic environment, generally low learning rates are used along with high discount rates to allow the algorithm to successfully converge even under random conditions. Conventionally, the optimal learning rate and discount factor is $\alpha = 0.1$ and $\gamma = 0.9$ for most general tasks, with a learning rate of $\alpha = 1$ more optimal in a deterministic environment (Simonini, 2018).

Nim

Nim is a combinatorial game—a two-player game that is deterministic and contains perfect information. A game is deterministic if there is no randomization, such as rolling a die. Perfect information exists if both players can view all the information regarding the state of the game without anything remaining hidden. In addition, Nim is also impartial, meaning the possible moves available to a player is based upon the position of the game rather than the player's turn, and the value for any given move remains equal for both players (Belwariar, 2018).

In a game of Nim, players alternately remove objects from one of several rows in each turn. During each turn, a player must remove at least one object, but can remove more than one if all are from one row. The end condition occurs when a player is unable to move, in which no objects remain, and the player loses. This is known as normal Nim; however, the variant of Nim used in this experiment is known as misère Nim, in which the player that removes the last object loses instead (Nim, n.d.).

The game of Nim has been entirely solved, meaning a known strategy exists to win for any initial configuration. The strategy is calculated through a Nim-sum, or exclusive-or (XOR) operation, of every row within the game. This is accomplished by representing the sum of objects in every row as sums of distinct powers of two, then eliminating any powers that appear twice before summing the remaining powers. For instance, $3 = 2^1 + 2^0$ and $5 = 2^2 + 2^0$, so the Nim-sum $3 \oplus 5 = 2^2 + 2^1 = 6$ since both sums contain 2^0. In normal Nim, the winning strategy is to ensure the Nim-sum equals zero on the opponent's turn. On the other hand, the winning strategy in misère

Nim is similar but slightly more complex as the game approaches the end. The initial strategy is identical to that of normal Nim, with a shift in strategy occurring when only a single pile of more than one object is remaining, with the other piles having either one or zero objects remaining. If the initial strategy is followed, this board state is guaranteed eventually, and the new strategy is to reduce the pile with more than one object to either one or zero objects, whichever results in an odd number of one-object piles remaining. Therefore, assuming both players play perfectly, the winner of a game is determined based upon the player that moves first. If the initial board configuration has a Nim-sum of zero, the second player is guaranteed to win under the optimal strategy, as seen in the board configuration used in this experiment. On the other hand, the first player is guaranteed to win under the optimal strategy if the Nim-sum of the initial board configuration is zero (Nim, n.d.).

Materials

32-bit/64-bit PC (Mac OS, Windows, Linux)

Internet Access

Procedure

If Python 3.6 is not installed on PC:

1. Download the corresponding package of Python 3.6 for your PC from https://www.python.org.

2. Install Python and configure it by following the prompts.

If IntelliJ PyCharm 2019.2.3 IDE is not installed on PC:

3. Download the corresponding package of IntelliJ PyCharm 2019.2.3 IDE for your PC from https://www.jetbrains.com.

4. Install PyCharm and configure it by following the prompts.

Running Program:

5. Download and import the full project implementation which can be found here.

6. Run the program to test both the positive and negative controls, as well as to train the Q-learning algorithm for 200,000 games and simulate 10 trials of 200,000 games against the expert algorithm for each of the 36 distinct learning rate and discount factor pairs.

7. After each program has completed, CSV files containing the data will be automatically generated and can be found in each agent directory.[2]

[2] The location for each CSV is agents/agent/test_results.csv

Results

For each learning rate and discount factor pair, the average percentage of games won by the Q-learning algorithm was calculated over 10 trials. Each average was graphed with respect to the learning rate rather than each unique pair for simplicity.[3]

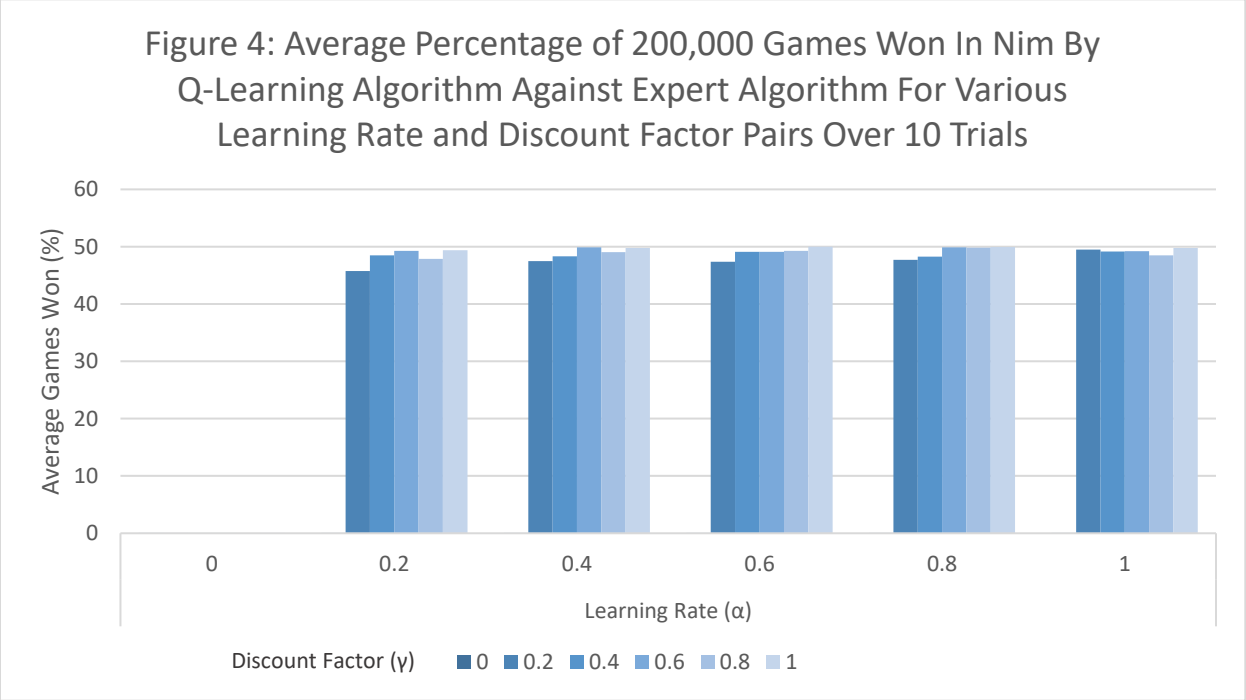

Figure 5: Average Percentage of 200,000 Games Won in Nim by Q-Learning Algorithm Against Expert Algorithm for Various Learning Rate and Discount Factor Pairs Over 10 Trials

Average Percentage of Games Won (%)		Learning Rate (α)					
		0.0	0.2	0.4	0.6	0.8	1.0
Discount Factor (γ)	0.0	0.005	0.002	0.004	0.003	0.004	0.004
	0.2	0.003	45.772	47.508	47.377	47.729	49.480
	0.4	0.003	48.470	48.333	49.092	48.275	49.143
	0.6	0.003	49.256	49.862	49.076	49.910	49.219
	0.8	0.004	47.863	49.047	49.283	49.842	48.480
	1.0	0.004	49.366	49.750	49.969	49.942	49.762

[3] The full dataset containing statistics for each individual trial can be found in Appendix B.

Figure 6: Average Percentage of 200,000 Games Won in Nim by Expert & Random Algorithm Against Expert Algorithm Over 10 Trials			
Average Percentage of Games Won (%)		Algorithm 1	
		Expert	Random
Algorithm 2	Expert	50.000	0.152

A one sample t-test was conducted between the mean of each pair and the hypothetical mean value of 50%. The hypothetical mean value was selected as 50% as it was the maximum percentage of games that could be won by the Q-learning algorithm, as well as the percentage of games won by the expert algorithm against an identical expert algorithm. The α value considered the threshold for significance was 0.05. Pairs with p-values above this value indicate that the average percentage of games won by the Q-learning algorithm was not significantly different than the theoretical maximum of 50%, suggesting the algorithm was as effective as the expert algorithm. On the other hand, p-values below 0.05 suggest the average percentage of games won by the Q-learning differed significantly from 50%, or were not as effective as the expert algorithm.

Discount Factor (γ)	Learning Rate (α)					
	0.0	0.2	0.4	0.6	0.8	1.0
0.0	t = 52699 p<0.0001 SIGNIFICANT	t = 175674 p<0.0001 SIGNIFICANT	t = 83212 p<0.0001 SIGNIFICANT	t = 131754 p<0.0001 SIGNIFICANT	t = 175668 p<0.0001 SIGNIFICANT	t = 143730 p<0.0001 SIGNIFICANT
0.2	t = 112931 p<0.0001 SIGNIFICANT	t = 3.6071 p = 0.0057 SIGNIFICANT	t = 2.2621 p = 0.0500 INSIGNIFICANT	t = 2.4503 p = 0.0367 SIGNIFICANT	t = 1.9688 p = 0.0805 INSIGNIFICANT	t = 1.5000 p = 0.1679 INSIGNIFICANT
0.4	t = 98815 p<0.0001 SIGNIFICANT	t = 1.5361 p = 0.1589 INSIGNIFICANT	t = 2.1107 p = 0.0640 INSIGNIFICANT	t = 1.9543 p = 0.0824 INSIGNIFICANT	t = 1.9935 p = 0.0774 INSIGNIFICANT	t = 1.3684 p = 0.2044 INSIGNIFICANT
0.6	t = 105402 p<0.0001 SIGNIFICANT	t = 1.0976 p = 0.3009 INSIGNIFICANT	t = 2.4283 p = 0.3381 INSIGNIFICANT	t = 1.2484 p = 0.2434 INSIGNIFICANT	t = 1.9016 p = 0.0897 INSIGNIFICANT	t = 0.9999 p = 0.3435 INSIGNIFICANT
0.8	t = 71864 p<0.0001 SIGNIFICANT	t = 2.0179 p = 0.0744 INSIGNIFICANT	t = 1.1063 p = 0.2973 INSIGNIFICANT	t = 1.0946 p = 0.3021 INSIGNIFICANT	t = 2.3282 p = 0.3449 INSIGNIFICANT	t = 1.5139 p = 0.1643 INSIGNIFICANT
1.0	t = 131750 p<0.0001 SIGNIFICANT	t = 1.0000 p = 0.3434 INSIGNIFICANT	t = 0.9998 p = 0.3435 INSIGNIFICANT	t = 1.0005 p = 0.3432 INSIGNIFICANT	t = 0.9998 p = 0.3435 INSIGNIFICANT	t = 0.9999 p = 0.3435 INSIGNIFICANT

Figure 6. One sample t-tests for significance

Discussion

The results of the t-tests suggest that the average percentage of games won by the Q-learning algorithm against the expert algorithm was not significantly different than 50% for nearly every pair beyond (α=0.2, γ=0.2). This suggests that there is no relationship between the efficiency of the Q-learning algorithm and the learning rate/discount factor pair beyond a minimum threshold, performing nearly as well as the expert algorithm.

The one exception is the pair (α=0.6, γ=0.2), which likely failed to win 50% of the games due to variability within the learning phase of the algorithm. The Q-learning algorithm failed to learn when either α=0 or γ=0, performing even worse than the random algorithm with an average of 3-4 games won out of 200,000. This is consistent with theoretical expectations as the algorithm fails to learn any data when α=0 and discounts all data entirely when γ=0.

Furthermore, the positive control functioned as expected, with the expert algorithm consistently winning the 50% of games in which it moved second and played the optimal strategy against an identical algorithm. Similarly, the negative control functioned as expected as well, with the random algorithm winning only an average of 152 games out of 200,000. Further analysis revealed a total of 5920 possible state-action pairs within the Q-table, with each pair allowing the algorithm to nearly reach convergence. Given additional training episodes, eventually the Q-value for each state-action pair would converge regardless of the learning rate and discount factor (ignoring α=0 or γ=0), allowing the Q-learning algorithm to play perfectly as the expert algorithm.

Conclusion

The purpose of this experiment was to determine the relationship between the learning rate and discount factor pair on the efficiency of Q-learning in the game of Nim, which was measured through the percentage of games won against an expert algorithm. The initial hypothesis of the experiment was that any pair with a value greater than (α=0.6, γ=0.6) would result in an equal percentage of games won by the Q-learning algorithm as the expert algorithm (50%).

While the results from the experiment supported the hypothesis, the hypothesis failed to encompass all pairs that performed successfully. The t-test analysis revealed that the Q-learning algorithm performed equally well when trained with nearly any pair with a value greater than (α=0.2, γ=0.2), excluding only the pair (α=0.2, γ=0.6). With p-values consistently above 0.05 for the aforementioned pairs, the percentage of games won by the Q-learning algorithm was not significantly different from 50%, meaning it performed nearly as well as the expert algorithm.

A possible error present in the experiment could be the number of episodes over which the Q-learning algorithm was trained. As nearly every pair with a non-zero learning rate and discount factor approached a win rate of 50% after 200,000 games of self-play, a lower number of episodes would likely allow a greater degree of distinction in the results between each pair, with pairs with higher learning rate and discount factor values performing significantly different than lower value pairs. With such a large number of episodes, the algorithm under each pair was likely near convergence, allowing it to perform equally as well as the expert algorithm.

However, the possibilities of future experimentation are especially expansive, as Q-learning is only a single means of machine learning that has unlimited variations beyond the context of games. For instance, the efficiency of Q-learning under various learning rate and discount factor pairs can be measured through other metrics such as time per move, moves per game, etc. Furthermore, the learning efficiency of the algorithm can be more accurately measured through the number of episodes required for the algorithm to reach convergence, which is guaranteed in a deterministic environment. The algorithm would reach convergence when it plays perfectly and wins exactly 50% of the games against the expert algorithm. Additionally, Q-learning efficiency can be measured over changes in the game conditions themselves, with different initial board configurations, variants of Nim, and variations within the algorithm itself. The application of Q-learning can additionally be extended beyond Nim to other games such as checkers as well, with testing in stochastic environments.

While the results of this experiment are not directly relevant to a practical application, the underlying utilization of Q-learning certainly has implications within several other contexts. The results prove the viability of Q-learning in learning and applying large amounts of data within complex tasks, with distinctions between learning rates and discount factors minimal at smaller scales. Although more complex forms of machine learning exist such as neural networks which are currently employed in technological industries, they expand upon the basis of Q-learning, which can be used in nearly any application consisting of an environment and corresponding state-action pairs. For instance, Q-learning can be integrated within autonomous robots and drones to allow them to learn environmental navigation. While traditionally supervised or unsupervised learning

is used for machine learning, they require the compilation of large, preexisting volumes of data, which may be impractical in certain scenarios in which data is scarce. With the use of Q-learning, however, algorithms are able to produce and use their own data simply through environmental exploration, allowing greater efficiency throughout the process of machine learning.

Works Cited

A Beginner's Guide to Deep Reinforcement Learning. (n.d.). Retrieved from SkyMind: https://skymind.ai/wiki/deep-reinforcement-learning

Ashraf, M. (2010, April 10). *Reinforcement Learning Demystified: Markov Decision Processes (Part 1)*. Retrieved from Towards Data Science: https://towardsdatascience.com/reinforcement-learning-demystified-markov-decision-processes-part-1-bf00dda41690

Bajaj, P. (2018). *Reinforcement learning*. Retrieved from https://www.geeksforgeeks.org/what-is-reinforcement-learning/

Belwariar, R. (2018, February 10). Combinatorial Game Theory | Set 2 (Game of Nim). Retrieved from https://www.geeksforgeeks.org/combinatorial-game-theory-set-2-game-nim/

Eden, T., Knittel, A., & Uffelen, R. V. (n.d.). Reinforcement Learning. Retrieved from https://www.cse.unsw.edu.au/~cs9417ml/RL1/algorithms.html

Huang, S. (2018, January 11). *Introduction to Various Reinforcement Learning Algorithms. Part I (Q-Learning, SARSA, DQN, DDPG)*. Retrieved from Towards Data Science: https://towardsdatascience.com/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dqn-ddpg-72a5e0cb6287

Jarleberg, E. (2011). *Reinforcement learning on the combinatorial game of Nim* (Unpublished bachelor's thesis). KTH Royal Institute of Technology. Retrieved from http://www.csc.kth.se/utbildning/kth/kurser/DD143X/dkand11/Group6Lars/erikjarleberg.pdf

Nim. (n.d.). Retrieved from Brilliant.org: https://brilliant.org/wiki/nim/

Simonini, T. (2018, April 10). Diving deeper into Reinforcement Learning with Q-Learning. Retrieved from https://medium.freecodecamp.org/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe

Appendices

Appendix A: Complete Dataset

	Positive	Negative	(0.0, 0.0)	(0.0, 0.2)	(0.0, 0.4)	(0.0, 0.6)	(0.0, 0.8)
1	50.000	0.154	0.002	0.005	0.002	0.004	0.004
2	50.000	0.159	0.005	0.003	0.002	0.004	0.001
3	50.000	0.149	0.008	0.004	0.003	0.002	0.002
4	50.000	0.150	0.006	0.002	0.001	0.004	0.007
5	50.000	0.149	0.003	0.004	0.004	0.005	0.002
6	50.000	0.155	0.001	0.004	0.007	0.002	0.004
7	50.000	0.136	0.005	0.003	0.002	0.001	0.004
8	50.000	0.148	0.002	0.001	0.004	0.003	0.008
9	50.000	0.163	0.003	0.005	0.002	0.001	0.003
10	50.000	0.159	0.003	0.002	0.002	0.005	0.004
avg	50	0.1524	0.0050	0.0034	0.0032	0.0031	0.0039
std	0	0.00757 4812	0.0030	0.0014	0.0016	0.0015	0.0022
t-value			52699.3572	112930.8080	98814.8523	105402.7200	71864.3410
p-value			<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
SED			0.0010	0.0000	0.0010	0.0000	0.0010

	(0.0, 1.0)	(0.2, 0.0)	(0.2, 0.2)	(0.2, 0.4)	(0.2, 0.6)	(0.2, 0.8)	(0.2, 1.0)	(0.4, 0.0)
1	0.004	0.001	47.750	44.048	49.780	50.000	50.000	0.007
2	0.003	0.003	47.823	41.216	50.000	50.000	50.000	0.002
3	0.004	0.001	50.000	49.737	50.000	43.483	43.660	0.004
4	0.004	0.002	41.333	50.000	43.167	42.207	50.000	0.004
5	0.002	0.004	43.334	49.696	50.000	50.000	50.000	0.001
6	0.004	0.001	40.878	50.000	50.000	49.510	50.000	0.004
7	0.005	0.002	50.000	50.000	50.000	50.000	50.000	0.004
8	0.005	0.002	50.000	50.000	50.000	50.000	50.000	0.001
9	0.007	0.004	43.514	50.000	50.000	43.427	50.000	0.004
10	0.004	0.004	43.095	50.000	49.614	50.000	50.000	0.005
avg	0.0044	0.0023	45.7725	48.4696	49.2560	47.8626	49.3660	0.0036
std	0.0012	0.0009	3.7062	3.1505	2.1435	3.3496	2.0049	0.0019
t-value	131749.9742	175674.0109	3.6071	1.5361	1.0976	2.0179	1.0000	83211.8415
p-value	<0.0001	<0.0001	0.0057	0.1589	0.3009	0.0744	0.3434	<0.0001
SED	0.0000	0.0000	1.1720	0.9960	0.6780	1.0590	0.6340	0.0010

	(0.4, 0.2)	(0.4, 0.4)	(0.4, 0.6)	(0.4, 0.8)	(0.4, 1.0)	(0.6, 0.0)	(0.6, 0.2)	(0.6, 0.4)
1	49.408	50.000	50.000	41.335	50.000	0.001	50.000	45.175
2	49.691	50.000	49.668	50.000	50.000	0.003	49.640	49.692
3	45.130	50.000	49.707	50.000	50.000	0.004	47.742	50.000
4	40.742	46.130	50.000	50.000	47.502	0.002	42.433	49.486
5	50.000	49.263	49.623	50.000	50.000	0.004	49.486	48.200
6	49.268	50.000	50.000	50.000	50.000	0.001	43.473	50.000
7	42.246	50.000	49.620	50.000	50.000	0.002	41.867	49.624
8	49.622	44.252	50.000	50.000	50.000	0.004	49.452	49.621
9	49.637	49.620	50.000	50.000	50.000	0.002	49.680	49.786
10	49.344	44.062	50.000	49.137	50.000	0.005	50.000	49.341
avg	47.5085	48.3326	49.8617	49.0472	49.7502	0.0029	47.3772	49.0924
std	3.4830	2.4981	0.1801	2.7235	0.7901	0.0012	3.3849	1.4686
t-value	2.2621	2.1107	2.4283	1.1063	0.9998	131753.9270	2.4503	1.9543
p-value	0.0500	0.0640	0.3381	0.2973	0.3435	<0.0001	0.0367	0.0824
SED	1.1010	0.7900	0.0570	0.8610	0.2500	0.0000	1.0700	0.4640

	(0.6, 0.6)	(0.6, 0.8)	(0.6, 1.0)	(0.8, 0.0)	(0.8, 0.2)	(0.8, 0.4)	(0.8, 0.6)	(0.8, 0.8)
1	49.536	50.000	50.000	0.004	50.000	49.455	50.000	50.000
2	50.000	50.000	50.000	0.003	47.461	48.745	50.000	50.000
3	49.913	50.000	49.689	0.004	48.993	49.393	50.000	49.456
4	50.000	49.615	50.000	0.004	49.531	49.476	49.613	49.753
5	49.659	50.000	50.000	0.004	49.710	50.000	49.770	49.909
6	49.521	49.817	50.000	0.004	50.000	43.862	50.000	50.000
7	50.000	50.000	50.000	0.004	41.153	49.581	50.000	49.488
8	50.000	43.402	50.000	0.005	49.673	50.000	50.000	50.000
9	49.693	50.000	50.000	0.005	50.000	42.472	50.000	49.819
10	42.436	50.000	50.000	0.003	40.773	49.772	49.713	50.000
avg	49.0757	49.2833	49.9689	0.0041	47.7292	48.2755	49.9095	49.8423
std	2.3414	2.0706	0.0983	0.0009	3.6473	2.7356	0.1505	0.2142
t-value	1.2484	1.0946	1.0005	175667.6863	1.9688	1.9935	1.9016	2.3282
p-value	0.2434	0.3021	0.3432	<0.0001	0.0805	0.0774	0.0897	0.3449
SED	0.7400	0.6550	0.0310	0.0000	1.1530	0.8650	0.0480	2.3282

	(0.8, 1.0)	(1.0, 0.0)	(1.0, 0.2)	(1.0, 0.4)	(1.0, 0.6)	(1.0, 0.8)	(1.0, 1.0)
1	49.423	0.003	47.395	50.000	42.194	50.000	50.000
2	50.000	0.004	50.000	50.000	50.000	50.000	50.000
3	50.000	0.004	50.000	50.000	50.000	50.000	47.624
4	50.000	0.003	47.404	50.000	50.000	50.000	50.000
5	50.000	0.001	50.000	47.436	50.000	50.000	50.000
6	50.000	0.004	50.000	50.000	50.000	39.980	50.000
7	50.000	0.004	50.000	50.000	50.000	47.389	50.000
8	50.000	0.002	50.000	50.000	50.000	47.432	50.000
9	50.000	0.004	50.000	50.000	50.000	50.000	50.000
10	50.000	0.005	50.000	43.994	50.000	50.000	50.000
avg	49.9423	0.0036	49.4798	49.1430	49.2194	48.4801	49.7624
std	0.1825	0.0011	1.0967	1.9805	2.4686	3.1748	0.7514
t-value	0.9998	143729.5444	1.5000	1.3684	0.9999	1.5139	0.9999
p-value	0.3435	<0.0001	0.1679	0.2044	0.3435	0.1643	0.3435
SED	0.0580	0.0000	0.3470	0.6260	0.7810	1.0040	0.2380