Reinforcement Learning applied in Monopoly

A deep investigation of concepts for the use of Reinforcement Learning in a board game with proximate application and objectives.
Selected Topics in Artifitial Intelligence 2021. Professor: Gissel Velarde

Roberto Pablo Cuevas Pinto
Computer System Engineering
Universidad Privada Boliviana
Nuestra Señora de La Paz, Bolivia
robertopablocp1@gmail.com

Cesar Ernesto Illanes Argote
Computer System Engineering
Universidad Privada Boliviana
Nuestra Señora de La Paz, Bolivia
cesarill99@gmail.com

Luciana Avelina Nuñez Linares
Computer System Engineering
Universidad Privada Boliviana
Nuestra Señora de La Paz, Bolivia
lucianaunezl99@gmail.com

***Abstract*—This document describes the important concepts to understand the implementation of Reinforcement Learning (RL) in the Monopoly board game [1]. The main purpose of this project is to analyze and investigate the behavior of the RL Agent implemented in .NET by making various experiments changing values of some parameters and hyperparameters.**

***Keywords*—Reinforcement Learning, Q-Learning, Q(λ)-Learning, Markov Decision Process, Monopoly.**

I. Introduction

One of the most important areas developed in recent years within Machine Learning is Reinforcement Learning (RL) applied for areas such as board or video games, where learning in an unknown environment is more important than gathering lots of information. Monopoly is an example where this type of learning is applied.

II. Literature Review

The project uses a RL approach to model the board game Monopoly as a Markov Decision Process (MDP). The implementation of an environment and a basic Q-Learning agent is necessary to train with another RL agent to evaluate it with two players with defined characteristics.

A review of the concepts referenced in [1] was made in order to learn these concepts and to have a better idea of the proposed project. A synthesis of each concept is presented below:

A. *Markov Decision Process*

The MDP [2] is a sequence of decisions that go through a time *t* and is controlled by an agent. It is composed of a set of states, a set of actions, a reward function, a probability function and a transition function.

B. *Reinforcement Learning*

Reinforcement Learning [3] bases its decisions on trial and error, learning which actions give it a higher reward and assigning situations to certain actions to maximize a numerical reward signal.

C. *Q-Learning / Q(λ)-Learning*

Q-Learning [4] is a type of model-free Reinforcement Learning, which focuses on finding the Q-value, which is the maximum reward that can be obtained by applying an action while in a certain state.

Q(λ)-Learning [5] is an improved model of Q-Learning and TD(λ)-Learning where the use of λ, an eligibility trace, significantly improves the model.

III. Model

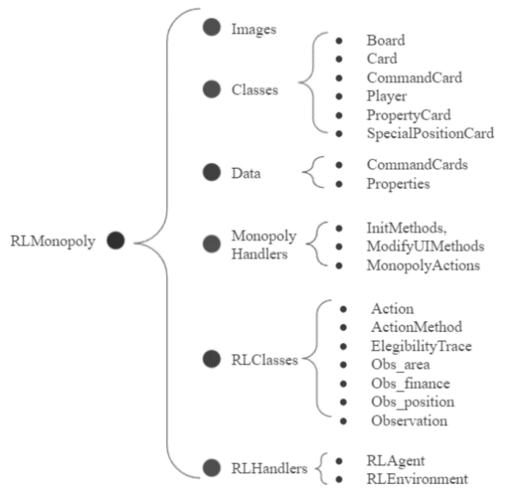
As in [1], the State Space defines a state in certain moment, as a tridimensional vector where it is defined area, position and finances. The Action Space presents all possibilities the agent can perform: buy, sell or do nothing. Finally, the Reward Model limits the output, allowing it to discriminate state-actions positives and negatives, in order to give all available information to players.

The RL Agent is a version of Q(λ)-Learning and it also makes the use of a 3-Layer Backpropagation Network.

IV. Experiments

The Project presented in [1] was developed in .NET and written in C# in 2013. Our plan was to study the code presented in order to understand the functioning, change different parameters and hyperparameters and see how they affect the model performance. For a better execution of the program, we had to rollback to an older version of it with the help of Git. The structure of this version can be shown in *Figure I*.

Figure I. Structure of the project developed in .NET



We executed 6 experiments, each one with different purposes for observations and analysis. First, the parameters modified in this implementation were: the MaxSteps to limit a Monopoly Game and the TotalGames that will be played by the agents. Second, the hyperparameters modified in each iteration were λ and γ.

A. *Experiments 1 and 2*

The first two experiments were made to check if the project was running properly. Both failed, for different reasons, being too long to execute and erroneous changes in the code, so we did not get any results.

B. *Experiment 3*

The number of games to be played was reduced and that error in the code was fixed, in order to know if the project already run properly and can really save results.

C. *Experiments 4 and 5*

They were created to learn more about the importance of hyperparameters if they are manipulated, in the last one giving more emphasis on the Gamma value and its relation with the MaxSteps.

D. *Experiment 6*

We wanted to have some more executions with the goal of comparing the results between each other and the rest of the experiments.

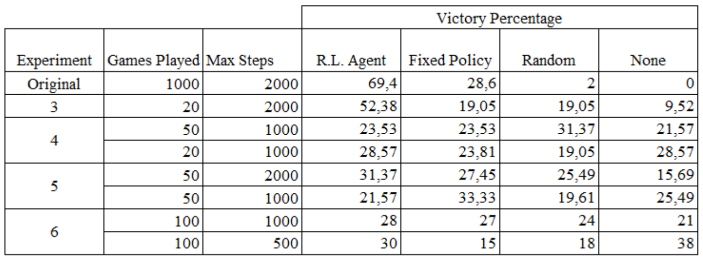
V. Obtained Results

Table 1 shows the original results presented in [1]. *Table 2* shows us the results obtained at each experiment. The changes made in λ and γ vary at each iteration, with λ going from 0.8 and 0.85, and γ going from 0.5 and 0.95. It is important to remark that in all our iterations, there was at least a 9.52% of times that none of the agents wins, so the agent learned slower.

TABLE I. Results presented in [1]

Player	**N° Victories**	**Percentage**
Random	20	2%
Fixed Policy	286	28.6%
RL Agent	694	69.4%
Total	1000	100%

TABLE II. Percentage of wins for each experiment



			Victory Percentage			
Experiment	Games Played	Max Steps	R.L. Agent	Fixed Policy	Random	None
Original	1000	2000	69,4	28,6	2	0
3	20	2000	52,38	19,05	19,05	9,52
4	50	1000	23,53	23,53	31,37	21,57
	20	1000	28,57	23,81	19,05	28,57
5	50	2000	31,37	27,45	25,49	15,69
	50	1000	21,57	33,33	19,61	25,49
6	100	1000	28	27	24	21
	100	500	30	15	18	38

VI. Conclusions

Training the agent further is mandatory, since the is room for improvement. However, so far, there is no mechanism implemented in the .NET implementation that would allow to use an already-trained agent and improve its performance.

This project helped us realize the importance that Reinforcement Learning will take in the near future. Agents will help humans to achieve larger or more general problems, first in the virtual world as now, and then in reality, when they will complement us for a better tomorrow.

VII. Contribution

All participants contributed equally in the research of the basic concepts for the subsequent implementation of the project, as well as the preparation of this document and its presentation.

VIII. References

[1] P. Bailis, A. Fachantidis, I. Vlahavas. "Learning to play Monopoly: A Reinforcement Learning approach". 2014. Available: http://doc.gold.ac.uk/aisb50/AISB50-S02/AISB50-S2-Bailis-paper.pdf

[2] L. Sucar, E. Morales. "Procesos de Decisión de Markov y Aprendizaje por Refuerzo". 2018. Available: http://ccc.inaoep.mx/~emorales/Papers/2018/2018ESucar.pdf

[3] R. Sutton, A. Barto. "Reinforcement Learning, An Introduction". 2018. Available: https://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

[4] C. Watkins, P. Dayan. "Q-Learning". 1992. Available: https://www.gatsby.ucl.ac.uk/~dayan/papers/cjch.pdf

[5] J. Peng, R. Williams. "Incremental Multi-Step Q-Learning". 1994. Available: https://link.springer.com/content/pdf/10.1007/BF00114731.pdf