Reinforcement Learning: Mario Bros

Rafael Moreno Cañas[1], Eric Gómez Vázquez[1]

Instituto Tecnológico y de Estudios Superiores de Monterrey,

Campus Estado de México

Abstract: Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation.

I. Introduction.

Reinforcement learning in formal terms is a method of machine learning wherein the software agent learns to perform certain actions in an environment which leads it to maximum reward. It does so by exploration and exploitation of knowledge it learns by repeated trials of maximizing the reward.

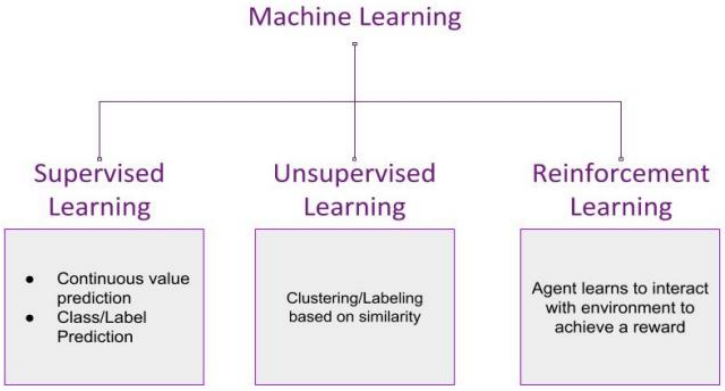


Fig 1. Reinforcement Method vs other Machine Learning methods.

Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it, so the model is trained with the correct answer itself

whereas in reinforcement learning, there is no answer, but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

Main points in Reinforcement Learning.

- *Input:* The input should be an initial state from which the model will start.
- *Output:* There are many possible outputs as there are variety of solution to a particular problem.
- *Training:* The training is based upon the input; The model will return a state and the user will decide to reward or punish the model based on its output.
- The model continues to learn.
- The best solution is decided based on the maximum reward.

II. Problem Statement.

Super Mario Bros.

Super Mario Bros is a platform video game developed and published by Nintendo. In Super Mario Bros, the player takes on the role of Mario, the protagonist of the series.

Objective.

The objective is to race through the Mushroom Kingdom, survive the main antagonist Bowser's forces, and save Princess Toadstool. The game is a side-scrolling platformer; the player moves from the left side of the screen to the right side in order to reach the flagpole at the end of each level.

Techniques used.

At the start of an episode an agent will begin at the start state, and it will keep making actions until it arrives at the end state. Once the agent makes it to the end state the episode will terminate, and a new one will begin with the agent once again beginning from the start state.

Q-Learning.

A state-action value is the quality of being on a particular state and taking a particular action off of that state.

We denote these state-action values as Q(s, a) (the quality of the state-action pair), and all state-action values together form something called a Q-table.

Mathematically, if an agent is on state "s", it will take argmaxaQ(s, a). But how are these values learned? Q-learning uses a variation of the Bellman-update equation and is more specifically a type of temporal difference learning.

Q-Learning Equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma max_a Q(s_{t+1}, a_t) - Q(s_t, a_t))$$

Double Q-Learning.

There is one major issue with Q-learning that we need to deal with: over-estimation bias, which means that the Q-values learned are actually higher than they should be.

To get more accurate Q-values, we use something called double Q-learning. In double Q-learning we have two Q-tables: one which we use for taking actions, and another specifically for use in the Q-update equation.

III. Proposal Approach.

Now that we have the approach and tools of reinforcement learning, so we can start developing our agent, that will be capable to go thorough the first level of Super Mario Bros.

Core Libraries:

- gym-super-mario-bros : Python library that provides ML enthusiasts with a set of environments for reinforcement learning.
- Pytorch: Python library that have functions for ML techniques.
- nes-py: is an NES emulator and OpenAI Gym interface for MacOS, Linux, and Windows based on the SimpleNES emulator.

Environment.

The environment that we will use for this development are all the levels from the Super Mario Bros Game. Which consists of 8 worlds, each world with 4 levels.

For this specific study, we only train our agent for the World 2 – level 2.

Fig. 2. Super Mario Bros Game.

States.

A state is a list of 4 contiguous 84×84-pixel frames, and we have 5 possible actions. If we were to make a Q-table for this environment, the table would have $5x256^{84x84x4}$values, since there are 5 possible actions for each state, each pixel has intensities between 0 and 255, and there are 84×84×4 pixels in a state.

Therefore, we will have the following Q Equation.

$$Q^*(s_t, a_t) \leftarrow Q^*(s_t, a_t) + \alpha(r_{t+1} + \gamma\, max_a\, Q_\theta(s_{t+1}, a) - Q^*(s_t, a_t))$$

Where:

rt+1 + γmaxaQθ(st+1, a) is considered our target, and Q*(st, at) is the value predicted by our network.

The DQN we develop is a convolutional neural net with 3 convolutional layers and two linear layers. It takes two arguments: input_shape(4x84x84) and n_actions(5).

Rewards.

This is the score that is given to the agent, it achieves more score depending on the right distance it travels, so more to the right the agent goes in the map, the more score it will achieve.

IV. Experimental Results.

We train our agent with the pytorch library, we use 1000 episodes where our agent learns with the regards formula that we describe in the above step.

Some of the results we achieve are better explained with the following graphs:

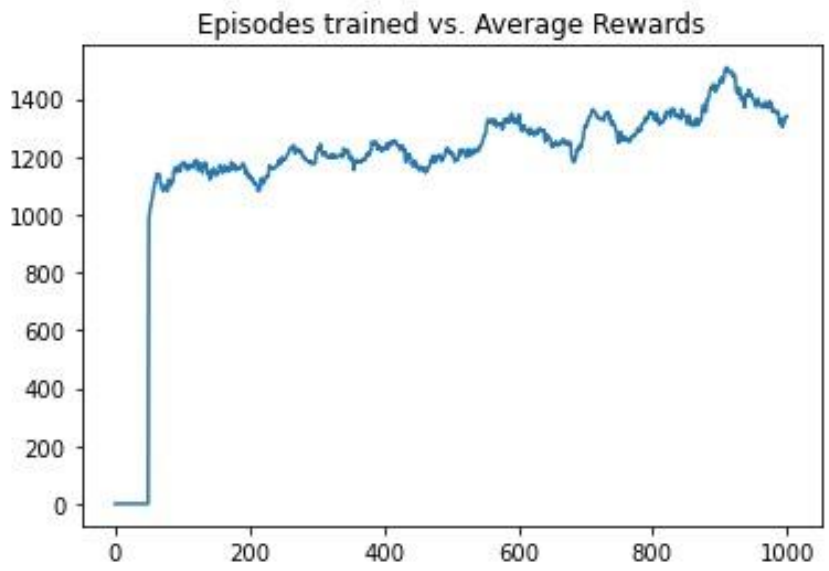


Fig 3. Episodes trained(1000) vs Average Regards.

In the Fig3, we can appreciate the performance of this training, where we noticed the constant increase of the rewards, among the episodes goes on.

Fig4. Episodes trained(500) vs Average Regards.

Here we can compare the average of rewards obtained with the half of the training.

At the end we can conclude this experiment successfully, we decided to train our agent with 1000 episodes and to test it in the World 2 – level 2 of the Mario game, all these with the help of the environment available thanks to gym-super-mario-bros and to nes-py that was the library in charge of moving Mario.

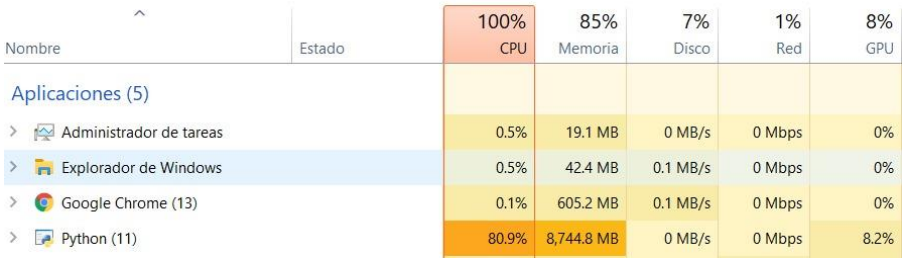
V. Conclusions.

Its interesting the results we achieve from this practice, we were able to appreciate all the tools of reinforcement learning that we covered in class, this combined with our abilities of programming, and the open-source knowledge, all this brings together an agent capable to learn constantly from its mistakes, with a reward system.

The importance of this practice is that we were able to put in practice all the theory about ML and most important reinforcement learning.

Some of the conclusions that we noticed, about the Mario agent, were that at the beginning of the training process the agent spends a lot of episodes just to figure out how to jump to another site where a hole was. This was a flag to us, that our agent will need more than our initial episodes, that were 500 episodes. At the end, we decided to go on a thousand episodes and to train the agent on a lower complex level, which was the water level of world 2.

To close conclusions, we also noticed that implementing this kind of solutions of Machine Learning, consumes a lot of the resources of the computer, so in order to implement bigger solutions we will need a better equipment or in the other hand, develop this solution on a Cloud Environment.



Nombre	Estado	100% CPU	85% Memoria	7% Disco	1% Red	8% GPU
Aplicaciones (5)						
Administrador de tareas		0.5%	19.1 MB	0 MB/s	0 Mbps	0%
Explorador de Windows		0.5%	42.4 MB	0.1 MB/s	0 Mbps	0%
Google Chrome (13)		0.1%	605.2 MB	0.1 MB/s	0 Mbps	0%
Python (11)		80.9%	8,744.8 MB	0 MB/s	0 Mbps	8.2%

Fig 5. Memory occupation.

VI. Appendix.

a. User Manual.

VII. References.

1. Reinforcement learning - GeeksforGeeks. (2020). Retrieved 18 November 2020, from https://www.geeksforgeeks.org/what-is-reinforcement-learning/
2. 10 Real-Life Applications of Reinforcement Learning - neptune.ai. (2020). Retrieved 18 November 2020, from https://neptune.ai/blog/reinforcement-learning-applications
3. Safety-first AI for autonomous data centre cooling and industrial control. (2020). Retrieved 18 November 2020, from https://deepmind.com/blog/article/safety-first-ai-autonomous-data-centre-cooling-and-industrial-control
4. blog, D. (2020). Asynchronous Deep Reinforcement Learning from pixels. Retrieved 18 November 2020, from https://dbobrenko.github.io/2016/11/03/async-deeprl.html
5. Building a Deep Q-Network to Play Super Mario Bros | Paperspace Blog. (2020). Retrieved 25 November 2020, from https://blog.paperspace.com/building-double-deep-q-network-super-mario-bros/
6. Super Mario Bros. (2020). Retrieved 25 November 2020, from https://en.wikipedia.org/wiki/Super_Mario_Bros.