ÉCOLE
D'INGÉNIEURS
DE L'AIR ET DE L'ESPACE

Introduction to Machine Learning

From a childhood game to reinforced learning

Authors :
Matthieu **Thomas**
Julien **Huynh**

Supervised by :
Mme **Frontera-Pons**

Version 0.1
April 13, 2020

Contents

Chapter 1

The legendary PinBall 3D game

Back in the old days when multiplayer gaming and unlimited internet connection was not as widespread, as kids, some of us eventually turned to a Windows pre-installed game, PinBall 3D.

1.1 Objective

Score as many points as possible

Chapter 2

Reinforcement learning

2.1 Why reinforcement learning ?

While thinking about the way to use Machine Learning to play PinBall, we thought of various possibilities but the one that stood out from the rest was reinforcement learning. The advantage of this method is that it works well when there are various states through time which is the case in a game of Pinball. Moreover, we can fairly easily say that the reward involved in this method will be related to the score and the lives.

2.2 Basics of reinforcement learning

2.2.1 Concepts

Simply put, reinforcement learning is a method that takes advantage of the interactions between :

- Agent
- Actions
- Rewarding system
- Environment
- States
- Policy
- Value

Other concepts are also used but the ones mentioned above are the most essential ones.

2.2.2 Main idea

In reinforcement learning methods, the agent does a set of actions in order to maximize the reward it will get. At each given time, the agent will compute an action being given the state and this will then have a reaction from the environment surrounding it as a feedback which is the reward and will put it in a new state.

One representation of some of those interactions is the following loop :

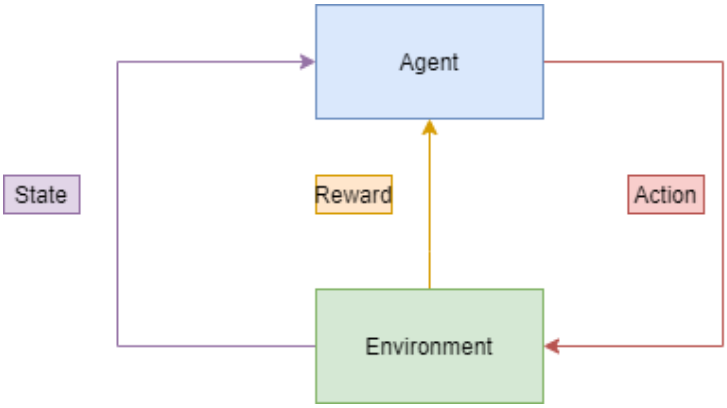

Figure 2.1: Reinforcement learning concept interactions

On this loop, at a given time t, the agent knows the state s_t and the reward r_t at time t and then computes an action a_t with respect to s_t and r_t. Then, the environment will give a new state s_{t+1} and a new reward r_{t+1} as a response to a_t. Obviously, the better the action taken was with respect to the objective, the higher the reward will be.

While it is debatable, reinforcement learning does not exactly fall into Supervised or Unsupervised learning because when the agent does the action, there are no labels. However, we could consider the reward as a time delayed label.

2.2.3 Policy, Value and Trajectory

While they did not appear in the loop, the trajectory, policy and value are essential in reinforcement learning.

Policy

The policy is the strategy the agent will use to reach its objective. If we refer to path-finding, it could be the different ways to go from a starting point to the objective point using different algorithms (Dijkstra, A*, RRT, potential field, ...). Simply put, it

defines how the agent will choose its action to have an impact on the environment.

The learning part is done at the **Agent** part which, given the state and the previous reward, will learn the best policy to maximize the value.

Value

In reinforcement learning, the value is basically the long term reward, it is what we would expect as an overall reward while being in a certain state and following a certain policy. The difference with the reward is that the latter is instantaneous, meaning that it is simply a response at a given time while the value is a much more global concept. Mathematically, it is defined as :

$$V(s) = \mathbb{E}\left[\sum_n \gamma^n r_n\right]$$

With n the states, r_n the reward for the state and $\gamma \in [0, 1]$ a kind of damping factor for the reward, called discount factor. This factor helps giving a balance in the importance between the current reward and the ones to be expected in the "future". This is important in some situations when for example, an action is taken, giving a high reward at the given time but will also greatly reduce the expected future rewards thus, the value.

Trajectory

Roughly speaking, the trajectory τ is the set of states the agent went through after interacting with the environment.

2.3 Model-based vs model-free

One of the two important separations that we will see in Reinforcement learning is model-based versus model-free.

Model-based algorithms learn a model meaning that they rely on states, actions and the **environment**. This means that if after the learning, our agent can predict the next state and reward related to it before even doing the action, it is a model-based algorithm. Those algorithms approximate a model of the environment.

On the other hand, model-free algorithms rely only on the states and the actions. They can **not** predict what the model-based algorithms can. The best action is not chosen using the environment. Those are "explicit" trial-and-error algorithms which are done using different methods such as the Monte Carlo method or temporal difference learning for

example. One of the most popular reinforcement learning algorithms is Q-learning which is a model-free, off-policy algorithm.

2.4 Q-Learning

We choose to take a closer look at Q-Learning as it is one of the most popular reinforcement learning algorithms. Its objective is to maximize a value called the Q-value which is similar to the previously mentioned value but also takes the current action as an additional parameter.

An appropriate description for this Q value would be quality of that given action and that would then allow us to estimate the score at the end without knowing what will come right after that given action (because we're model-free).

Obviously different actions will give different Q-values so in order to maximize our result, all we have to do is to choose the action that will give the highest Q-value. This implies learning the function, the mapping from the action to the Q-value and this process is Q-learning. This is done by saving and updating values in tables while approximating on the trial-and-error approach through the learning process.

While a basic Q-learning algorithm will do well on relatively small problems, issues will start rising once we approach bigger problems with more sophisticated and complex problems. This is especially true in our case with PinBall 3D where the actions might be limited but the number of states is gigantic. Needless to say, such a large amount of data to store and update is inefficient and requires a huge computational power thus, the need for a more efficient technique.

2.5 Deep reinforcement learning

Deep reinforcement learning is a solution to our previously stated problem. As we mentioned, updating values and storing them is costly so we can try to approximate the function of this Q-value using Deep Neural Networks. As mentioned during our class, there is this fundamental theorem saying that Neural Networks can approximate about any function given the right number of neurons and layers. Thus, we could approximate the Q-value for each possible action given the state. This combination of learning a Q-function with a Deep Neural Network is called a Deep Q-Network (DQN)

With DQNs, the entry layer corresponds to the state (pixels for example) and the last layer is the different Q-values, one for each possible action. The so called labels used for propagation in this case is the targeted Q-values which can be computed from the Bellman Equation :

$$Q_{target}(s_t,\ a_t) = \mathbb{E}\left[r_{t+1} + \gamma \max_{a'} Q_{target}(s',\ a')\right]$$

We could then consider the loss as a comparison between this targeted Q-value and the resulting output of our network and then tune as we would do with any typical neural network.

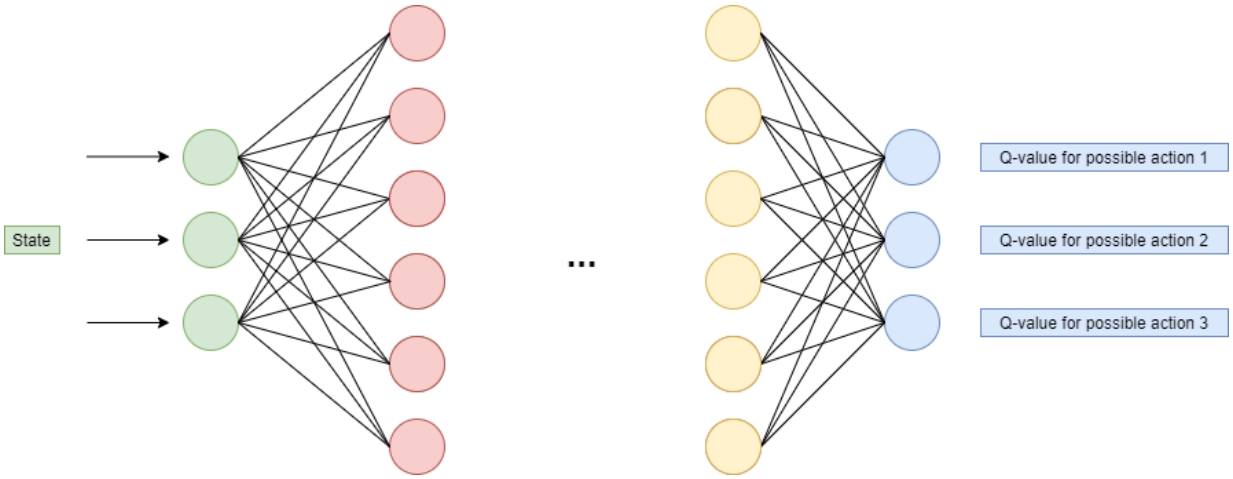

Figure 2.2: Deep Q Network input/output

The idea behind this is still to solve the same problem as with Q-Learning but with a different method to make it more efficient.

This deep reinforcement learning method is used in a lot of games where people (either players or companies) want to create AIs to play the game and it is also valid for PinBall 3D.

Chapter 3

Reinforcement learning - Application to PinBall 3D

blabla applicatif + git

Sources

- https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56
- https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4
- https://www.geeksforgeeks.org/what-is-reinforcement-learning/
- https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html
- https://towardsdatascience.com/keras-transfer-learning-for-beginners-6c9b8b7143e
- https://pathmind.com/wiki/deep-reinforcement-learning
- Various online courses/videos