Reinforcement Learning in Video Games

Valentí Torrents Vila

21 d'abril de 2024

Resum– Resum del projecte, màxim 10 línies. ....
.. ..
... .
....
....
....
....
....
....
....
....

Paraules clau– Paraules clau del treball, màxim 2 línies . ...
....

Abstract– Versió en anglès del resum . ...
....
....
....
....
....
....
....
....
....
....

Keywords– Versió en anglès de les paraules clau. ....
....

✦

1 Introduction

1.1 Preliminary Information

REINFORCEMENT learning, or RL, is a type of machine learning that revolves around how an intelligent agent should behave in an environment in order to achieve an specific goal. Unlike other machine learning paradigms, like supervised learning or unsupervised learning, RL's goal is to generate an intelligent agent that learns on its own while interacting with a dynamic environment. Through trial and error, the agent must chose which of the possible actions yield the most reward(both immediate and long-term), until achieving the necessary ”knowledge”to go through the environment without any problems and solving it.

1.2 Objectives and expected results

The goal of this project is to dive into the world of Reinforcement Learning while using the Gym library effectively. Going from the fundamentals of RL, the project aims to understand and implement different RL algorithms. Starting with a quick introduction to this world, and going through

- E-mail: valentovi55@gmail.com
- Specialisation: Computation
- Work tutored by: Jordi Casas Roma
- Year 2023/2024

the basics with different tabular methods, our final objective is to go deep into the most complex methods applicable to video games. The following objectives mark the trajectory of this project:

- Develop a foundational understanding of Reinforcement Learning and proper use of the Gym library.
- Study and implement tabular methods such as Dynaim Programming, Monte Carlo, or Q-learning to create an agent capable of solving simple environments.
- Study and comprehend some non-tabular methods, like policy-based methods and Deep Q Networks (DQN).
- Apply the acquired knowledge to solve more complex and sophisticated environments, enhancing the agents' capabilities.
- Generate a series of agents, each more complex, capable of achieving desired results in various games, while documenting findings and utilizing visualization tools to ensure a thorough exploration of Reinforcement Learning in video games.

1.3 Metodology

I have chosen to work using the Agile methodology for my project, working in iterative sprints, each lasting two weeks, which will be shown in the Gantt Diagram. Using this methodology allows continual reassessment and adjustment if necessary since reinforcement learning is a dynamic field and it will need some adaptability. The tasks shown in the Gantt Diagram will be accomplished in sprints of up to two weeks, depending on the workload. Through the regular review sessions with my tutor, we will check the gradual progression of the project, and provide checkpoints for evaluation, adaptation, and resolution of new obstacles as they appear.

1.4 Planning

The project planning is visualized through a Gantt diagram. This will be a dynamic tool to help the development of the project. At first it shows the main objectives and milestones necessary and, over time, using the Agile methodology, the Gantt diagram will show more, depending on the obstacles faced and the necessary changes that appear in the process. [Fig. 1]

2 STATE OF THE ART

Having talked about the basics of this project, we can now get into a more detailed explanation of what is the State of the art like, and how our first implementations of reinforcement learning methods has been like.

2.1 Introduction to RL

As we have just explained in the preliminary information, RL stands at an in-between point of artificial intelligence and decision-making. Thanks to that, it offers a powerful structure for creating and teaching agents to navigate through sequential decisions in dynamic environments. Simply put, RL revolves around the interaction between an agent and a specific environment. Through observations, this agent can perceive its environment and, based on its current state, selects one action or another. Furthermore, and depending on the specifics of the environment, the agent receives feedback in the form of rewards. Depending on the actions taken by the agent in each situation, these rewards can vary, indicating the desirability of said actions. Through these rewards, the agent is guided towards learning optimal strategies, which let it maximize these rewards over time. To further understand how Reinforcement Learning works, we have to talk about its components:

- The environment is, basically, the problem space with which the agent interacts. All the states, possible actions, rewards and rules that the agent has to abide come from the environment. Its dynamics dictate how the agent's actions modify future states and rewards, and even though we usually see dynamic environments in RL, there can also be static ones.
- The agent is the principal entity in RL, and the one that takes the decisions, earns the rewards, and goes through the environment. By navigating through the environment, and recieving different inputs from it, the agent learns to take better decisions. Its goal is to follow a behaviour(policy) which maximizes the rewards over time, solving then the task at hand.
- Actions are the choices available to the agent at each step within the environment. Depending on which action the agent takes, its surroundings will be influenced one way or another. Actions can be discrete(finite options) or continuous.
- The states repressent the current position or configuration of the environment. Depending on the moment and the position of the agent the state changes, and with it all the relevant information used by the agent to make decisions.
- Rewards, as we said before, are provided by the environment to the agent, and are signals that evaluate its actions. They can present inmediate or delayed feedback, and are the resposibles for the changes in the agent's behaviour over time, since its objective is to maximize these rewards in order to solve the environment.
- Policies define the agent's behavior by relating all states to actions. They mark the strategy followed by the agent to achieve its objectives. These policies can be deterministic, where each state has a specific action, or stochastic, where all actions have a probability of happening depending on the state and policy parameters. The objective of all Reinforcement Learning algorithms is to achieve the optimal policy that maximizes long-term rewards, and leads the agent to solving the environment.

By interacting iteratively with its environment, and learning from its experience, RL algorithms generates agents that learn autonomously how to behave, solving different tasks, from video games to controlling robots. By learning through trial and error, and without supervision, RL ends up being perfectly adequate to solving scenarios where the agent must adapt to dynamic and complex environments.

2.2 OpenAI Gymnasium

OpenAI Gym serves as a versatile platform for the development and evaluation of reinforcement learning algorithms. Offering a diverse array of environments spanning from classic control problems to complex simulated scenarios, Gym provides a standardized interface for interacting with different tasks. Each environment in Gym is encapsulated as a Markov decision process (MDP), allowing agents to interact with states, take actions, receive rewards, and transition between states based on probabilistic dynamics. With its user-friendly API and extensive documentation, OpenAI Gym empowers researchers and practitioners to prototype, benchmark, and iterate on various reinforcement learning techniques with ease. Moreover, Gym's compatibility with popular RL libraries and frameworks fosters collaboration and accelerates the advancement of RL research and applications.

2.3 RL Methods

In the current state of reinforcement learning (RL), researchers employ tabular and non-tabular methods. Tabular methods, like dynamic programming and Monte Carlo, excel in small-scale tasks with discrete state and action spaces. They offer theoretical guarantees but struggle with larger, continuous spaces. Non-tabular methods, including deep reinforcement learning (DRL), leverage function approximation, particularly neural networks, to handle complex, high-dimensional environments. DRL algorithms like deep Q-networks (DQN) and policy gradients have demonstrated success in diverse applications, from robotics to gaming. Both approaches complement each other, with tabular methods providing theoretical foundations and non-tabular methods offering scalability and adaptability to real-world problems.

3 Tabular Methods

To get to understand the basics of RL we first have to talk about the tabular methods. These are the simplest ones in RL, since they store a specific action for each state in a table-like structure. The tabular methods we will talk about are Dynamic Programming, Monte Carlo methods, and Q-learning. To do so, we will work with finite Markov Decission Processes(MDP), trying to estimate value functions, which are functions of states that tell use how positive is for the agent to be in a specific state.

3.1 Markov Decission Processes

A Markov Decission Process(MDP) is a mathematical model used in RL to make sequential decisions in a stochastic environment. Basically, an MDP describes a system(or environment) where an agent makes decisions which results are subject to uncertainty. Just like the parts of a RL environment, a MDP contains states, actions and rewards. However, a MDP also contains a Transition Model, which describes the transition probabilities from one state to another after taking a certain action. In order to solve a finite MDP, we can use different tabular methods, which we are going to talk about now.

3.2 Dynamic Programming

Dynamic Programming(DP) is used to find the optimal policies to follow in order to solve a finite MDP. In a finite MDP, states, actions, and rewards are finite, and their dynamics are represented by transition probabilities. In order to achieve a good policy in RL in general, we use value functions. With DP we can compute these value functions by satisfying the Bellman optimality equations.
(IMPORTANT PARLAR AQUÍ DE LES EQUACIONS DE BELLMAN)
In our project, we employed policy iteration to find this optimal value function. First, we initialized a policy, and then evaluated and improved it iteratively until it converged. To keep evaluating the policy we had, we estimated the value function with the current policy using the Bellman Equations. Afterwards, we selected the actions that maximized the expected returns in order to update the policy. After repeating this 2 steps several times and observing no further improvements, we had the optimized value function and policy that let us guide the decision-making through the MDP.

3.3 Monte Carlo Methods

Even though Monte Carlo and DP are both tabular methods, Monte Carlo methods do not require a model of the environment to operate. Instead, they rely on trial and error, sampling trajectories by interacting with the environment.
In our Monte Carlo policy evaluation process, we began by executing episodes according to the current policy, collecting state-action-reward trajectories. After each episode, we calculated returns for each visited state, providing estimates of their values. These returns were then averaged over multiple episodes to update the value function incrementally. Optionally, we improved the policy based on the estimated values, possibly selecting actions greedily to maximize returns. This iterative refinement continued over multiple episodes until convergence, where the estimated value function approached the optimal one as the number of episodes increased.

3.4 Q-learning

4 Conclusions

....

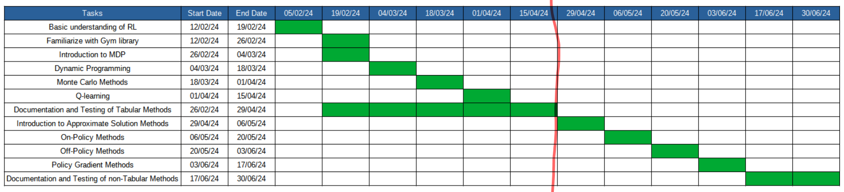
Special Thanks

... .
....
....
....

Referències

Appendix

Informe 1



Tasks	Start Date	End Date	05/02/24	19/02/24	04/03/24	18/03/24	01/04/24	15/04/24	29/04/24	06/05/24	20/05/24	03/06/24	17/06/24	30/06/24
Basic understanding of RL	12/02/24	19/02/24	🟩											
Familiarize with Gym library	12/02/24	26/02/24		🟩										
Introduction to MDP	26/02/24	04/03/24		🟩										
Dynamic Programming	04/03/24	18/03/24			🟩									
Monte Carlo Methods	18/03/24	01/04/24				🟩								
Q-learning	01/04/24	15/04/24					🟩							
Documentation and Testing of Tabular Methods	26/02/24	29/04/24		🟩	🟩	🟩	🟩	🟩						
Introduction to Approximate Solution Methods	29/04/24	06/05/24							🟩					
On-Policy Methods	06/05/24	20/05/24								🟩				
Off-Policy Methods	20/05/24	03/06/24									🟩			
Policy Gradient Methods	03/06/24	17/06/24										🟩		
Documentation and Testing of non-Tabular Methods	17/06/24	30/06/24											🟩	🟩

Fig. 1: Gantt Diagram