Multi-agent reinforcement learning on 2D movement

Luca Serfilippi
luca.serfilippi@studio.unibo.it

Abstract

Creation of a simulation with multiple agents that move in a 2D world using Mesa library and using reinforcement learning techniques with the TensorFlow library to control their movements to reach swarm configurations.

1 Introduction

1.1 Mesa library

The Mesa library [2] gives basic classes and methods to create custom multi-agent simulations. It has been selected since it is one of the best libraries with these capabilities in python. Mesa is modular, the various components of the simulation: modeling , analysis and visualization are separated but work together.

1. Modeling: modules used to build the models are agent classes, a scheduler do determine the sequence in which the agents act and the space for them to move around.
2. Analysis: tools to collect data from the model, to run it multiple times with different parameters
3. Visualization: classes to create and launch an interactive model visualization

All of these functionalities are needed to simulate the problem and retrieve data to train an ANN.

1.2 Reinforcement learning

Reinforcement learning is learning what to do (how to map situations to actions) so as to maximize a numerical reward signal [4]. To write the code the library used is TensorFlow [1]. We have an agent that acts in an environment and receives rewards, then an optimizer changes the agent parameters to maximize the the accumulation of rewards. Rewards have to be selected carefully to obtain the desired outcome. To extend it to a multi-agent system we can repeat the same algorithm for each agent looking at the others as a part of the environment. The rewards can be given individually or as a group to facilitate cooperation.

1.3 The problem to solve

The problem is inspired by the drones used at the Olympics to create images in the sky [3]. We have N-agents that have to reach N-positions to create a configuration, we want them to reach the positions with the least amount of time. With a series of positions also a video could be represented. This problem has a perfect solution that can be calculated a priory by a single machine, but by distributing the computation and using reinforcement learning we have a robust swarm and any image can be queried in real time since the calculation is done step by step by moving.

2 Simulation

The Simulation is composed of three parts, the definition of the space, the definition of the agents and the environment, the place where we put agents and spaces to interact.

2.1 Space

The Space is a n-dimensional toroidal multi-grid, meaning that it is a grid of height and width n where multiple agents can be in the same square and by exiting the grid the agent will come up in the opposite side of the grid. This space has been selected so that the movement of the agent are as free as possible, all directions are always available at any given position. The Space has the information of the positions of the agents and the positions that agents have to occupy for a successful configuration.

2.2 Agent

The agent is the entity that moves into space.

2.2.1 Agent Perception

The agent sees the space 2.1 through the observation space $O = M_n(\{0, 1, 2, 3\})$ that contains the information about the position of himself, the other agents and the positions to reach. Each value in the matrix $o(i, j)$ indicates the object present in the position $(i, j) \in n \times n$ of the simulation space. 0 represents the empty position, 1 another agent, 2 the agent observing the space and 3 the position to reach. Multiple objects might be present in the same position, if this happens the object present in that position is selected based on priority {2,3,1,0} from the higher priority to the lowest 1.

```
[[1 2 0]
 [0 3 0]
 [3 0 0]]
```

(a) in this state every information is maintained since the agents are not in the same square of each other or in a terminal position

```
[[0 0 2]
 [0 3 0]
 [1 0 0]]
```

(b) in this state we lose the information of the terminal position since the other agent is on the right spot

Figure 1: example of states representations in 3x3 matrix form

2.2.2 Agent Actions

The agent movement is discrete, it can move in the space in any of his eight neighbouring positions or staying still. The action space $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ comprehends all those options: 0 no movement, 1 to 8 goes from up to up-left clockwise.

2.2.3 Agent Decision

Given a perception ($P \in O$) the agent decides which action ($a \in A$) to take. The agent has an ANN that takes as input the perception and gives as a result a probability distribution of the action space ($P_a = \{p(a) : a \in A\}$). The actual action that the agent takes in the space is extracted from A based on P_a.

2.3 Environment

The environment can be created differently based on training or visualization phases.

2.3.1 Training

The Environment is set up to generate multiple experiences to train the agents. Since the TensorFlow library supports parallel calls on the network, multiple spaces can be built to exploit it. Also the experience of each agent can be used to train every other agent since they all have the same objective. So in the simulation is faster to use a single ANN that decides the movement of every agent in a single pass and adjourns itself on every agent experience calculating the gradient as a batch of $N\_simulations * N\_agents$.

2.3.2 Visualization

Once the ANN is trained the result can be visualized using MESA visualization tools. A single simulation can be set up to check out the results of the training 3.

3 Deep Reinforcement Learning

As the ANN it has been used a stack of three Dense layers 1 and for the calculation of the gradient the policy method Reinforce [4]. The reward at each step is calculated as +1 if the agent goes in a position to reach or if it remains in it, otherwise it gets -N_agents, that is because we want the reward to still be negative even if all the agents except one are in position. So in this way maximizing the cumulative reward gets the agents to get as fast as possible in each final position.

Table 1: Policy Model

Layer (type)	Output Shape	parameters	Connected to
input (InputLayer)	[(32, 3, 3)]	0	
flatten (Flatten)	(32, 9)	0	input[0][0]
dense0 (Dense)	(32, 32)	320	flatten[0][0]
dense1 (Dense)	(32, 32)	1560	dense0[0][0]
policy (Dense)	(32, 9)	297	dense1[0][0]

total parameters : 1673

4 Results

Mesa and TensorFlow are great libraries that used together can let us train multi-agents systems. For an example of a 3x3 space with 2 agents the reinforce algorithm is able to optimize the reward 4, the agents are able to reach the desired positions between one 3 or two moves 2, the optimal solution with a 3x3 matrix space should have a particular move for each scenario, but the network is an approximation and can have sub-optimal actions, this is helpful when the state is so big that a tabular solution is not an option.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] J. Kazil, D. Masad, and A. Crooks. Utilizing python for agent-based modeling: The mesa framework. In R. Thomson, H. Bisgin, C. Dancy, A. Hyder, and M. Hussain, editors, *Social, Cultural, and Behavioral Modeling*, pages 308–317, Cham, 2020. Springer International Publishing.

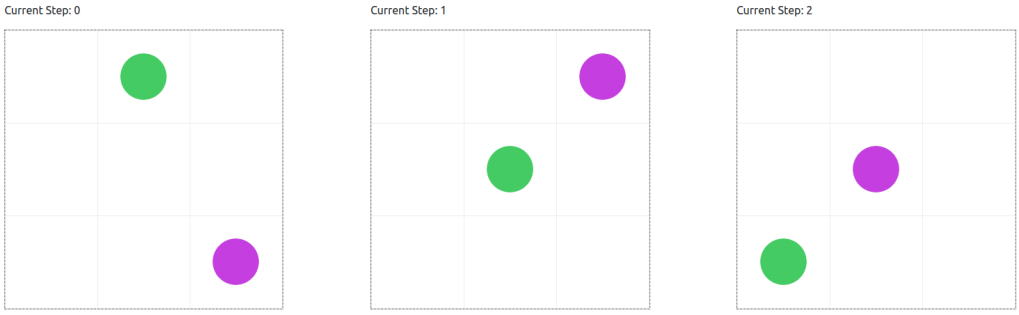

Figure 2: example of a two moves solution (desired positions center and down-left), the policy is not optimal.

Figure 3: example of a single move solution (desired positions center and down-left), where the purple agent even if it is in a goal position moves to the other position to let the green agent to take is position and reach the final goal, even if the rewards are selfish they learn collaborate.

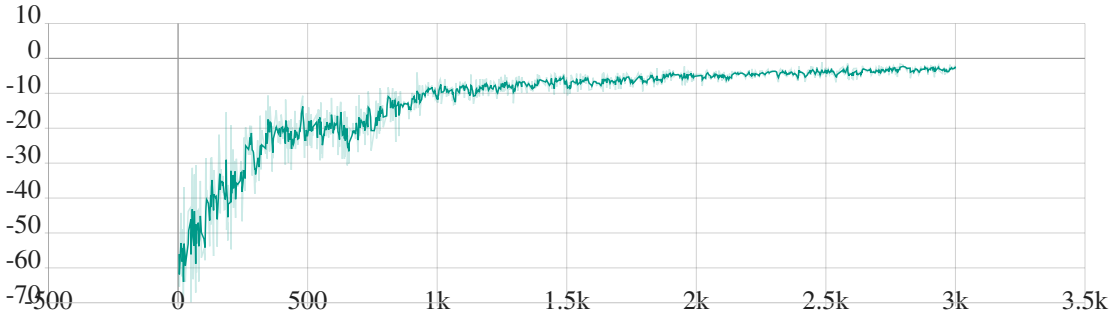

Figure 4: Mean Cumulative Reward at each epoch of 2 agents in a 3x3 toroidal space with fixed positions to reach

[3] Reuters. Tokyo olympics: Constellation of drones form globe at opening ceremony.

[4] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.