

Reinforcement Learning based Swarm Robotics for Rescue Operations

Michael Mu, Romika Sairam and Sachin George

CSE4/546 Reinforcement Learning (Instructor: Alina Vereshchaka)



Introduction

Multi-agent reinforcement learning is a new field being actively researched. We propose the usage of deep reinforcement learning (DQN, A2C) in the field of swarm robotics, controlling many agents to contribute to a singular goal. We test different approaches on a custom environment using MuJoCo.

Applications

In swarm robotics, many simple, small agents work together to achieve much greater tasks. This could be applicable in fields such as disaster search and rescue. Small robots can separately search cramped and dangerous environments and then reconvene to move a large object.



An example scenario is a recent event when an unstable building in Miami collapsed. The building was too unsafe for people to enter, but hurt or unconscious individuals may be stuck inside and need retrieval.

Another example was when children were stuck in a Thai cave. The only path to them was cramped and dangerous for humans to undertake. However, a robot would not be restricted in the same way and is disposable if damaged.



Project Overview

Our goal is to demonstrate the feasibility of deep RL in an environment with many simple agents to accomplish cooperative tasks. The objective is to train the agents to navigate an environment and to find and retrieve a target.

Problem

- Swarm robotics usually requires agents to be hardcoded in order to accomplish tasks, requiring labor-intensive work and is not adaptable
- The act of discerning and reacting to other agents in deep RL is very complicated

Our Approach

We propose simplistic agents with few possible actions so the burden of comprehension is small and the major issue becomes the ability to contribute to a greater task

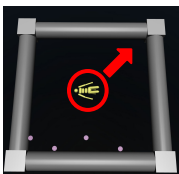
Environment

A custom 3D environment with realistic physics was developed using MuJoCo.

- Target:** humanoid ragdoll
- Agent:** cylindrical object
- Destination:** Top-right corner of environment

Agents are initialized to random locations in the lower region of the environment.

Goal: Find and go to the target and move it to the destination.



Deep Q-Network (DQN)

DQN is our baseline deep RL approach where a neural network predicts the potential value of future states and tries to pick the action with the best future state.

```
Algorithm 1: deep Q-learning with experience replay
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function Q̄ with weights θ̄ = θ
For episode = 1, M do
    Initialize sequence s₁ = {s₁} and preprocessed sequence φ₁ = φ(s₁)
    For t = 1, T do
        With probability ε select a random action aₜ
        otherwise select aₜ = argmaxₐ Q(φₜ, a; θ)
        Execute action aₜ in emulator and observe reward rₜ and image sₜ₊₁
        Set sₜ₊₁ = aₜ, aₜ₊₁ and preprocess φₜ₊₁ = φ(sₜ₊₁)
        Store transition (φₜ, aₜ, rₜ, φₜ₊₁) in D
        Sample random minibatch of transitions (φᵢ, aᵢ, rᵢ, φᵢ₊₁) from D
        Set yᵢ = {
            rᵢ + γ maxₐ Q(φᵢ₊₁, a; θ̄) if episode terminates at step j + 1
            otherwise
            rᵢ + γ Q(φᵢ, aᵢ; θ̄)
        }
        Perform a gradient descent step on (yᵢ - Q(φᵢ, aᵢ; θ))² with respect to the network parameters θ
    Every C steps reset Q̄ = Q
```

Advantage Actor-Critic (A2C)

A2C is an actor-critic algorithm which possesses two neural networks, the actor and the critic.

Actor - Produces the probability of taking each action

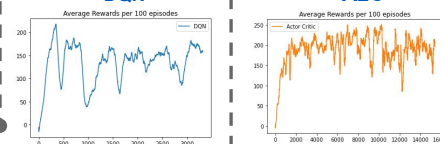
Critic - Rates the (advantage) value of the state and action taken by the actor

The actor is updated by maximizing the advantages according to the following objective function:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) A_w(s, a)]$$

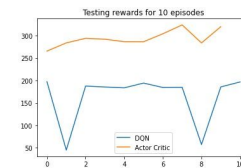
To simplify the critic, we predict state values to derive advantages, so the update function is the same as the value based DQN.

DQN Training A2C



Results

- A2C consistently solves the environment, but occasionally the would completely fail and all agents would stop moving
- DQN was more consistent in its scoring and movement, but it performed worse than A2C on average.



Conclusion

The A2C successfully solves the environment and is able to find the target and bring it to the destination.

The DQN initially performs well and converges faster than A2C, but then performance declines faster before the algorithm can successfully solve the environment

In both algorithms, a successful attempt at cooperation was completed. The agents were able to collaborate to discover and begin moving the target towards the destination, though only A2C was able to complete the task.

References

- <https://www.npr.org/sections/live-updates-miami-area-condo-collapse>
- CSE 4/546 Class Slides
- <https://github.com/openai/mujoco-py>
- <https://www.express.co.uk/news/world/984556/Thai-cave-rescue-Thailand-football-team-latest-update-rescue-options-video>