Introducing Deep Reinforcement Learning to Swarm Robotics

Michael Mu, Romika Sairam and Sachin George
CSE446 Reinforcement Learning (Instructor: Alina Vereshchaka)

Introduction

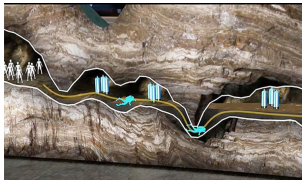
Multi-agent reinforcement learning is a new field being actively researched. We propose the usage of deep reinforcement learning (DQN, A2C) in the field of swarm robotics, controlling many agents to contribute to a singular goal. We test different approaches on a custom environment using MuJoCo.

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/unconscious individuals may be stuck inside and need retrieval.

Another example in recent news 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 of our project 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.

A. Target: humanoid ragdoll
B. Agent: cylindrical object
C. 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.
Initialise replay memory D to capacity N
Initialise action-value function Q with random weights θ
Initialise target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
For episode = 1, M **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For t = 1,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$

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 possible action

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

The actor is updated by trying to maximize 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.

Training

DQN

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

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