Soccer Playing MuJoCo Agent, Group 2

Abhishek Shah (DS), Ankur Gupta (CS), Rajendra Kante (DS) , Sai Vineeth K V (RBE)
Advisor: Prof. Yanhua Li (DS)

I. Introduction

There has been significant work done in playing games such as Go or Atari where there is a clearly identifiable goal of an agent trying to win the game with a well-defined reward function. However, in the case of an agent learning to do actions such as walking and running, the agent will need to learn a much more "implicit" reward function. The behavior is complex and continuous, and the agent needs to acquire special skills. Also, in case of agents containing multiple degrees of freedom, the action space is continuous and hence simple reward signals would not provide the desired learning behavior. Our project will attempt to train an agent to move towards a target, a soccer ball in this case, and move the ball towards a goal.

II. Environmental Setup

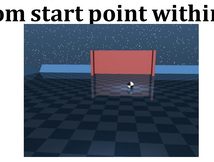
Agent

Our agent is a MuJoCo "Quadruped". The "Quadruped" agent has 4 DoFs and 3 Actuators per leg; making it a total of 12-dimensional action space that is continuous

Environment

The environment is a smooth bounded field containing a target goal post and a soccer ball in which the agent is expected to train and perform the task . Both the ball and the agents are initialized at a random start point within the field

Quadruped Agent



Environment

Reward

Total reward the agent gains is based on the agent's relative position from the ball, ball's position from the goal and also for the agent's upward orientation.

III. Algorithm

We considered 2 approaches which are the variants of Policy Gradient methods. We used an actor-critic method (A2C) and a policy optimization method (PPO).

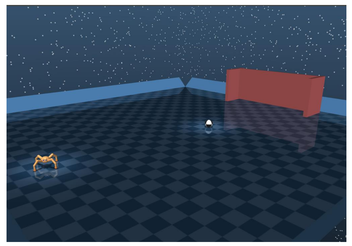
A2C (Advantage Actor-Critic)

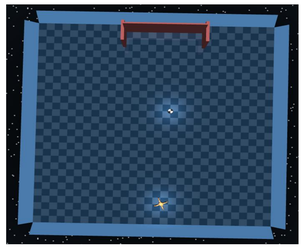
A2C is a synchronous actor-critic method where the agent policy is updated through learnt state-value functions and the policy gradients are updated via an "Advantage" function (captures the relative performance of an action at a particular state).

PPO (Proximal Policy Optimization)

PPO is an iterative procedure that confines the magnitude of the step taken in the optimization procedure within a trust region, given by the KL divergence of the new policy and the old policy.

IV. Rewards



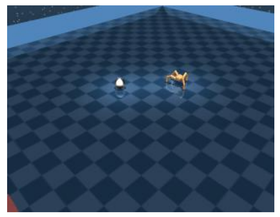


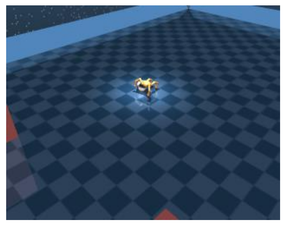
1. Reach reward

Reward obtained by the agent for moving towards the ball position on the field

2. Move reward

This is the reward obtained by the agent for moving the ball towards the goal location





Overall reward

(upright reward) * (*w1** Reach reward + *w2** Move reward),
where *w1* & *w2* are weights assigned to each reward.

V. Tools

1. Mujoco:

- MuJoCo is a physics engine aimed to help researchers create accurate simulations by accounting for joint dynamics and contact forces, leaving the user to only bother about the RL algorithm.
- Interactive 3D visualization support is provided with OpenGL.

2. Deep mind Control Suite:

- The DeepMind Control Suite is a set of continuous control tasks with a standardized structure and interpretable rewards, intended to serve as performance benchmarks for reinforcement learning agents.
- The tasks are written in Python and powered by the MuJoCo physics engine, making them easy to use and modify.
- In our project, we extended the pre-defined task set to include a new "soccer" task for the quadruped agent.

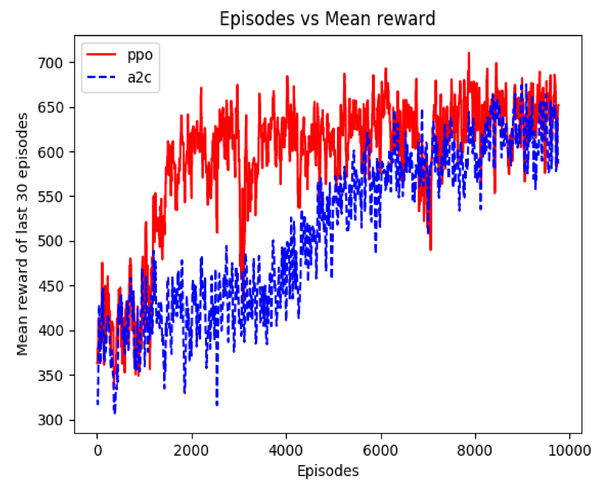
3. Computational / Hardware Resources:

- The Hardware Platform chosen is a Linux (Ubuntu 16.04) system with NVIDIA capabilities to run the training on our RL task .
- The training process with CUDA enabled took as long as 6 hrs in case of PPO and 4 hs in case of A2C to obtain our trained model. The same with CUDA enabled took around 12 hrs for PPO & 9 hrs for A2C.
- PyTorch was used as the learning framework

VI. Results

The average reward (over episodes) versus time were logged for both PPO and A2C to track learning. Unlike typical supervised learning problems, the problem space is largely unexplored, causing the network loss to have a high variance. However, an increase of reward over time demonstrates the emergence of intelligent behavior.

Both the PPO & A2C agents were trained for 10K episodes spanning over 1Million timesteps. A plot of the learning curve from the data logged during the training process gives an analysis of agent's learning ability in both cases. The trained model is also used to assess the agent's ability to complete the task.



VII. Conclusions & Future Work

- From the trained agents, we observed that PPO performed better than A2C w.r.t task fulfillment and there is intelligent movement and trajectory maintained in both PPO & A2C agents
- PPO Agent learned the desired behaviour at a faster rate than compared to its A2C counterpart
- The employed reward function seem to induce the desired agent behaviour and gave successful results and also worked in both A2C and PPO
- Future work would focus on exploring robust behaviour of the agent by adding external bias and noise in the environment and how agent would maintain the desired task behaviour