Project 4: Reinforcement learning (Bonus)
Introduction to Machine Learning

Yash Narendra Saraf
UB Person No. 50290453
MS Computer Science and Engineering
School of Engineering and Applied Science
ysaraf@buffalo.edu

Abstract

Train an atari environment to play games. Understanding how the Deep Q learning networks behave and what policies can be used to design a DQN. Implementing it all using a open source library stabel_baselines by Open AI.

1 Introduction

Atari games have been very popular in our childhood days and are very simple games which can be played using a joystick with less than 16 actions. So here we have a simple environment and limited actions and we need to train the network to play games.

2 Problem Statement

The project combines reinforcement learning and deep learning. Task is to teach the agent to navigate in the grid-world environment. We have modeled Tom and Jerry cartoon, where Tom, a cat, is chasing Jerry, a mouse. In our modeled game the task for Tom (an agent) is to find the shortest path to Jerry (a goal), given that the initial positions of Tom and Jerry are deterministic. To solve the problem, we would apply deep reinforcement learning algorithm - DQN (Deep Q-Network), that was one of the first breakthrough successes in applying deep learning to reinforcement learning.

3 Game - 1

Used the DQN library present in the stable_baseline and trained the game of **Demon Attack** over 10000 time-steps. Now the DQN library provides methods to use different types of policy networks based on the requirement.

Now for atari games there are a lot of states possible and training using a Convolutional neural network makes more sense as the CNN preserve the correlation between adjacent pixels. So for this I have used CNN policy.

After training the network the model starts to play a little which can be seen from the added video, this is because the game is quite simple and has very less actions so the model starts to show improvement at this earlier stage. Training games like breakout show no improvement whatsoever even after 25000 time-steps as the game is complicated and requires about 20 million time-steps to train.

The paper states the DQN implements an epsilon-greedy mechanism to decide whether to exploit or explore, and uses experience replay, i.e. storing all the experience in replay memory and applying Deep Q learning to mini batches in loop.

The DQN library uses RMSprop algorithm with mini batches of size 32. The epsilon greedy policy reduces the epsilon from 1 to 0.1 in the first million and then keeps it at 0.1. Also frame skipping can be used where instead of every sample every k^{th} sample can be taken and repeating last action on the skipped frame. This allows network to play game k times more than the standard method. Also the value of k should be selected properly so as the image data does not get distorted.

4 Game - 2

For experimentation purposes the MLPpolicy was used for 100,000 number of timesteps on the game of Atlantis which does not perform well even after 100,000 timesteps. The MLP policy implements a 2 layer neural network using tensorflow in the backend which is not sufficient to model such a complex game. Here while training the average reward did not seem to increase.

Conclusion

Here we implemented 2 games using 2 different policies, went through the deep mind's papaer on how reinforcement learning was implemented for the atari games and what were the core technologies and concepts used.

References

[1] Stable Baselines Documentation,
`https://media.readthedocs.org/pdf/stable-baselines/v1.0.7/stable-baselines.pdf`

[2] Playing Atari with Deep Reinforcement Learning
`https://arxiv.org/pdf/1312.5602.pdf`

[3] Stable Baseline User Guide
`https://stable-baselines.readthedocs.io/en/master/_modules/stable_baselines/common/cmd_util.html`