ML Assignment 2 - Final Report

Group 03: Eklavya Sharma, Anirudh Kumar Bansal

November 27, 2017

Abstract

We explore apprenticeship learning in this project. Apprenticeship learning is a process where an intelligent agent learns by first observing an expert and then trying out the task on its own. We devise an apprenticeship variant of the Deep Q-learning algorithm. We then test our solution on the OpenAI game 'CartPole'. Unfortunately, our algorithm fails to learn from expert human gameplay.

1 Introduction

1.1 Apprenticeship Learning

Reinforcement learning is a branch of machine learning where an intelligent agent learns from the environment by taking actions based on the current state and then observing rewards and change in state.

An intelligent agent is an entity which has the capability to observe the state of the environment. and the agent is required to take an action at every time instant. For every action, the agent receives a reward. A policy is a function which maps a state to an action. The goal of the agent is to learn a policy π such that on following that policy the agent is able to maximize its rewards.

When the agent interacts with the environment, it observes tuples of the form (s_1, a, r, s_2). This means that taking action a in state s_1 gives a reward r and changes the environment's state to s_2. A collection of such tuples constitutes 'experience'. In reinforcement learning, an agent uses a mixture of trial-and-error and experience to interact with its environment. This gives the agent more experience which helps it refine its policy.

When the intelligent agent interacts with the environment for the first time, it has no experience and must rely solely on hit-and-trial to gain experience. This can make it difficult for the agent to quickly learn a good policy.

In apprenticeship learning, an agent first observes an expert perform a task. It observes what actions the expert takes, how that affects the environment and what reward the expert receives. This technique therefore helps the agent get experience from the expert. The agent can then use this experience as a starting point and begin refining its policy from there onwards instead of starting from scratch.

Apprenticeship learning is different from supervised learning. The agent doesn't learn a policy only by looking at expert actions. It also refines the policy by itself interacting with the environment after observing the expert.

Apprenticeship learning therefore has 2 parts — observation and practice. Observation involves observing an expert, while practice involves learning by interacting with the environment on one's own.

1.2 Deep Q-Learning

Q-learning is a reinforcement learning algorithm in which the agent learns a Q-function. $Q(s, a)$ represents the expected discounted reward which the agent can obtain if it starts from state s and the first action it takes is a.

In deep Q-learning, the agent uses a deep neural network to learn the Q function. Such a neural network is called a 'Deep Q network' (DQN).

A successful variant of Q-learning uses an 'experience replay mechanism' to learn the Q-function [1]. It was the first algorithm which outperformed humans in Atari video games and only used raw images as input. This is the algorithm we use as the basis of our apprenticeship learning variant.

In this project, we compare reinforcement learning and apprenticeship learning based on their performance in 2 games.

2 Deep Q-Learning with Apprenticeship

First, expert human gameplay is recorded for several episodes. Gameplay is stored as a list of tuples of the form (s_1, a, r, s_2, d). Each such tuples means that the environment changed from state s_1 to state s_2 when action a was taken and reward r was received. d is a boolean which indicates whether s_2 is an absorbing state.

In our variant, before the agent is made to interact with the world, tuples from the expert gameplay are added to the agent's experience memory and its Q-network is updated using experience replay after every episode. This means that when the agent begins interacting with the environment, it doesn't start with an empty experience memory and it doesn't start with randomly-assigned weights in the Q-network.

3 Implementation

The agent was tested the 'CartPole-v1' environment from the OpenAI gym. CartPole's state consists of 4 real-numbers and it has 2 possible actions.

10 episodes of observation was generated for CartPole by Eklavya. A CartPole episode runs for approximately 25 seconds.

The deep Q-network uses the multi-layered perceptron network. It has 'relu' as activation in the hidden-layer and has no activation for the output layer. The hidden layers have sizes 12, 16 and 12 respectively.

The discount (γ) is set to 0.95. This parameter measures the ratio by which future rewards should be discounted when computing the 'future discounted reward' for the Q-value at a particular state and action. Experience memory size is 10000 and batch size for experience replay is 64. The agent chooses a random action with probability ϵ and chooses the action with the highest Q otherwise. ϵ is initially set to 0.1 and gets multiplied by 0.995 after each episode. The minimum value of ϵ is set to 0.02.

The network is trained using the Adam algorithm. The learning rate for the Adam algorithm is set to 0.01 and the learning decay rate over each parameter is also set to 0.01.

4 Results

A bot was trained 15 times — 5 times with no observation episodes and 1000 practice episodes, 5 times with only 500 observation episodes and no practice and 5 times with 500 observation episodes and 1000 practice episodes. Since expert gameplay data was scarce, the 10 recorded episodes were fed to the agent 50 times. After training, the agent is tested on 10 episodes.

The score for each practice episode was plotted. There is a lot of fluctuation in the score but it increased in general with practice episodes and after a point stops increasing. The agent performs okay with no observation episodes but performs poorly with observation episodes, regardless of whether it practiced. The practice episodes seem to mislead the agent instead of helping it.

Table 1: CartPole Test Scores

Sno	Pure Observation	Pure Practice	Mixed
1	-0.7	163.1	-0.2
2	-1.1	230.8	-0.5
3	-0.9	109.2	-0.7
4	-0.1	126.6	-0.7
5	-0.3	130.1	-0.5
mean	-0.62	151.96	-0.52

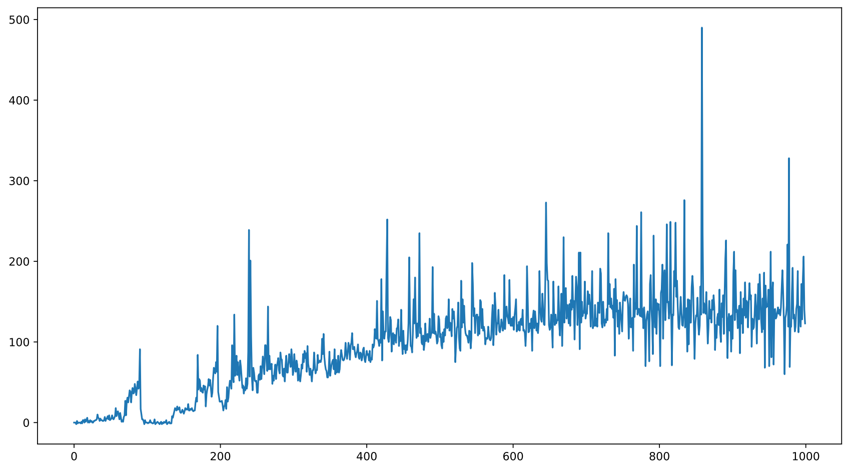

Figure 1: Scores in practice episodes (Pure Practice)

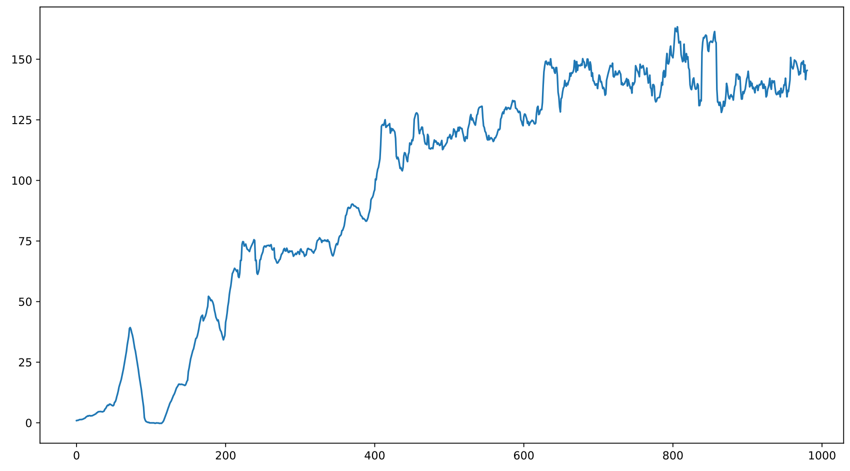

Figure 2: Average scores in sets of 20 practice episodes (Pure Practice)

5 Conclusion

We explored a learning method where experience is used to learn an initial Q function, which is then improved by an agent. This method, however, failed to produce good results. Apart from it being a bad learning technique, there could be several reasons why it didn't work. Expert gameplay data was limited. It's also possible that the neural network architecture was inappropriate.

Another reason could be that the way humans play a game might be very different from the way a computer might play it. Perhaps Deep Q-networks cannot be complex enough to understand human policies and require less data at the same time. A human policy will therefore mislead it instead of helping it.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.