Playing Doom using Deep Q Learning

[Term Paper Spring 2018]

Muhammad Abdullah Abu Bakar
FAST - National University of Computing and Emerging Sciences
Peshawar, Pakistan
p156043@nu.edu.pk

ABSTRACT

With the advent of reinforcement learning, we have seen the advent of machines being able to video-games, continuous environments and learn tasks via experience.

In this paper I have successfully used Deep Q Learning along with Memory Replay to make a machine traverse a stage of the Doom game with minimum use of processing power.

1. INTRODUCTION

Why Doom? Doom has been the centerpiece of reinforcement learning for a while now. There have been Global Reinforcement Learning Events conducted for playing in both Player vs Player Environment and Player vs Enemy(Non AI) Environments, with the latter being much more complex as it has a multi-agent environment with different players doing different things, just as it is in online games.

Why Deep Q Learning? Deep Q Learning has shown great promise in video-game reinforcement learning specially on Atari games and gaming overall. As the basic concepts in gaming is to get better at one specific task(for us humans) through learning, deep q learning fulfills that philosophy really well!

2. THE TECHNOLOGIES AND TECHNIQUES USED

The Main Technologies used in this semester experiment of mine are OpenAIGym, PyTorch and VizDoom(upto a very limited degree).

The Machine Learning Techniques that I have used in this are Convolutional Neural Networks, Q-Learning, Memory Replay and Eligibility Trace.

2.1 OpenAIGym and the Environment

So the main question is how do I receive the state of the game,get input, press action and all that stuff. Logically if I custom made all of that stuff then it would take me decades to read the memory buffers of Doom inorder to get the Score and stuff. Thankfully that is not the case, as OpenAI gym does that for us. It returns us 4 variables after completing each "step" of the environment:

observation (object): an environment-specific object representing your observation of the environment. For example, pixel data from a camera, joint angles and joint velocities of a robot, or the board state in a board game.

reward (float): amount of reward achieved by the previous action. The scale varies between environments, but the goal is always to increase your total reward.

done (boolean): whether itâĂŹs time to reset the environment again. Most (but not all) tasks are divided up into well-defined episodes, and done being True indicates the episode has terminated.

info (dict): diagnostic information useful for debugging. It can sometimes be useful for learning (for example, it might contain the raw probabilities behind the environmentâĂŹs last state change).

Every environment comes with an actionspace and an observationspace. These attributes are of type Space, and they describe the format of valid actions and observations.

Thus that part of the Equation is solved. That is practically getting the reward and doing the actions etc. I have used OpenAI Gym 0.9.2 as it contained the older version of Doom, and the levels aka Doom Corridor level was provided by VizDoom since the input of the algorithm is visual frames.

2.2 Deep Q Learning and Memory Replay

I have implemented the Deep Q Learning using a convolutional neural network as the inputs I take are frames of the game, plus I take an action after 4 frames rather than after every frame as it would just be a fidgeting mess after every frame.

The Convolutional NN takes the input in the forms of frames and then calculates the Q-Values(which are equal to the number of actions we can take in this environment) and from there we select the best Q value(best action).

I'm keeping the details of the DeepQLearning in this paper short as it will bloat it, that is why I will keep the github updated regarding relevant QLearning Information.

DeepQLearning on it's own has shown abysmal results in doom on it's own. Thus we have memory replay or experience replay as it's called. The idea behind experience replay is quite simple: at each Q-learning iteration, you play one step in the game, but instead of updating the model based on that last step, you add all the relevant information from the step you just took to a sized memory , and then call random-batch on a sample of that memory. Before doing anything on the model, we fill the memory with random actions. The reason why experience replay is helpful has to do with the fact that in reinforcement learning, successive states are highly similar. This means that there is a significant risk that the network will completely forget about what itâĂŹs like to be in state it hasnâĂŹt seen in a while.

2.3 Eligibility Trace

When the agent calculates the reward for "N-Steps" for playing the games and deciphers whether that set of moves were beneficial or not, then that is called N-Step QLearning or Eligibility Trace. In simple Q Learning we calculate the rewards for each q-value or each action predicted. In this we predict a series of actions without calculating their reward and then calculate the reward at the end of the "N-Steps". I implemented this concepted via udemy course on artificial intelligence and it has surprisingly really well.

2.4 Graphs

tobeaddedsoon

2.5 References

tobeaddedverysoon.

3. CONCLUSION

From my experiments conducted throughout the semester with different technologies and different environments for making intelligent agents. I conclude that it is possible to create intelligent agents for playing human video-games and learning them through a multitude of machine learning techniques, of which one I presented in this paper. The main obstacle in terms of achieving singularity; that is a machine that plays all and any computer games; is that making an interface to interact with the computer games and gain information from them. In my experiment I got all that through open ai gym, in modern AAA games that is not the case at all.