Term Paper Draft

Topic:”Automating Pac-man with Deep Q learning”

Name :Prajwal Mani

UCID:pbm6

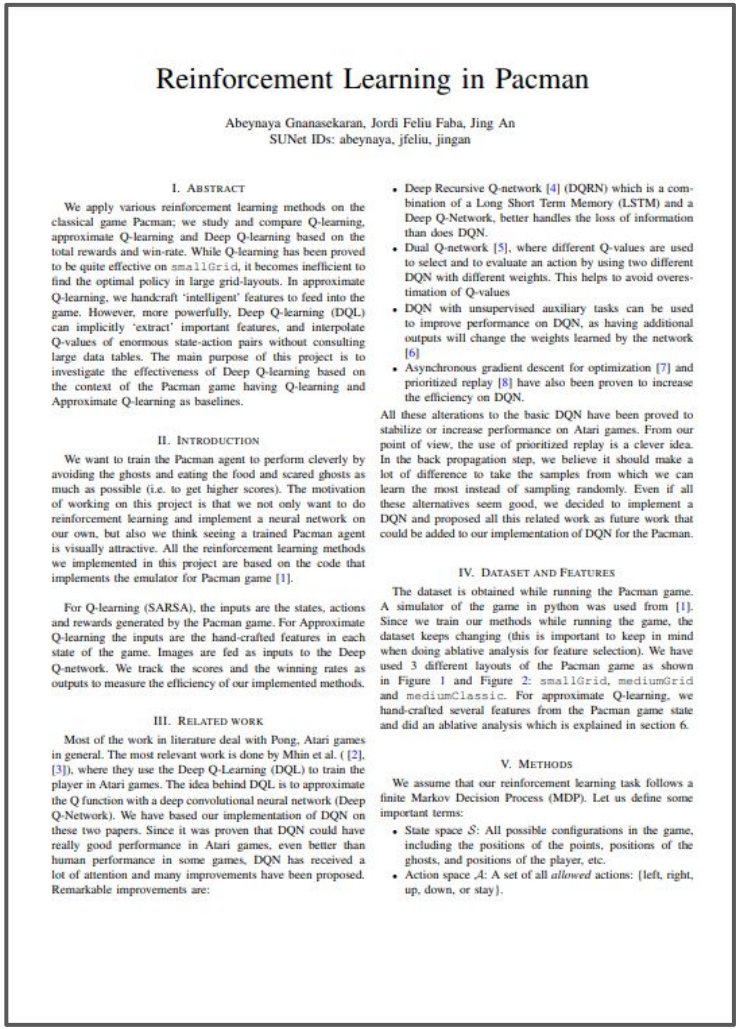
Email:pbm6@njit.edu

Subject:CS 634 104

Base paper

Paper link:
http://cs229.stanford.edu/proj2017/final-reports/5241109.pdf

The base paper is a course paper under stanford university written by Abeynaya, Jordi and Jing. The paper talks about the process of combing Q learning, Deep Learning, and the game itself with all the possible strategies to get the best outcome of the entire project. After understanding these strategies and checking out other reference papers mentioned in this paper. I will be turning them into the code with some changes adapted to the requirements and the technology available now.



Reinforcement Learning in Pacman

Abeynaya Gnanasekaran, Jordi Feliu Faba, Jing An
SUNet IDs: abeynaya, jfeliu, jingan

I. Abstract

We apply various reinforcement learning methods on the classical game Pacman; we study and compare Q-learning, approximate Q-learning and Deep Q-learning based on the total rewards and win-rate. While Q-learning has been proved to be quite effective on `smallGrid`, it becomes inefficient to find the optimal policy in large grid-layouts. In approximate Q-learning, we handcraft 'intelligent' features to feed into the game. However, more powerfully, Deep Q-learning (DQL) can implicitly 'extract' important features, and interpolate Q-values of enormous state-action pairs without consulting large data tables. The main purpose of this project is to investigate the effectiveness of Deep Q-learning based on the context of the Pacman game having Q-learning and Approximate Q-learning as baselines.

II. Introduction

We want to train the Pacman agent to perform cleverly by avoiding the ghosts and eating the food and scared ghosts as much as possible (i.e. to get higher scores). The motivation of working on this project is that we not only want to do reinforcement learning and implement a neural network on our own, but also we think seeing a trained Pacman agent is visually attractive. All the reinforcement learning methods we implemented in this project are based on the code that implements the emulator for Pacman game [1].

For Q-learning (SARSA), the inputs are the states, actions and rewards generated by the Pacman game. For Approximate Q-learning the inputs are the hand-crafted features in each state of the game. Images are fed as inputs to the Deep Q-network. We track the scores and the winning rates as outputs to measure the efficiency of our implemented methods.

III. Related work

Most of the work in literature deal with Pong, Atari games in general. The most relevant work is done by Mhin et al. ([2], [3]), where they use the Deep Q-Learning (DQL) to train the player in Atari games. The idea behind DQL is to approximate the Q function with a deep convolutional neural network (Deep Q-Network). We have based our implementation of DQN on these two papers. Since it was proven that DQN could have really good performance in Atari games, even better than human performance in some games, DQN has received a lot of attention and many improvements have been proposed. Remarkable improvements are:

- Deep Recursive Q-network [4] (DQRN) which is a combination of a Long Short Term Memory (LSTM) and a Deep Q-Network, better handles the loss of information than does DQN.
- Dual Q-network [5], where different Q-values are used to select and to evaluate an action by using two different DQN with different weights. This helps to avoid overestimation of Q-values
- DQN with unsupervised auxiliary tasks can be used to improve performance on DQN, as having additional outputs will change the weights learned by the network [6]
- Asynchronous gradient descent for optimization [7] and prioritized replay [8] have also been proven to increase the efficiency on DQN.

All these alterations to the basic DQN have been proved to stabilize or increase performance on Atari games. From our point of view, the use of prioritized replay is a clever idea. In the back propagation step, we believe it should make a lot of difference to take the samples from which we can learn the most instead of sampling randomly. Even if all these alternatives seem good, we decided to implement a DQN and proposed all this related work as future work that could be added to our implementation of DQN for the Pacman.

IV. Dataset and Features

The dataset is obtained while running the Pacman game. A simulator of the game in python was used from [1]. Since we train our methods while running the game, the dataset keeps changing (this is important to keep in mind when doing ablative analysis for feature selection). We have used 3 different layouts of the Pacman game as shown in Figure 1 and Figure 2: `smallGrid`, `mediumGrid` and `mediumClassic`. For approximate Q-learning, we hand-crafted several features from the Pacman game state and did an ablative analysis which is explained in section 6.

V. Methods

We assume that our reinforcement learning task follows a finite Markov Decision Process (MDP). Let us define some important terms:

- State space $\mathcal{S}$: All possible configurations in the game, including the positions of the points, positions of the ghosts, and positions of the player, etc.
- Action space $\mathcal{A}$: A set of all *allowed* actions: {left, right, up, down, or stay}.

Contents

Abstract

In this paper the authors talk about different reinforcement learning methods to make the agent play the classical pacman game.The three methods they compare are Q learning,Q learning approximate and Deep Q learning based on win rates and total rewards Most of time the paper talks about how to make Deep Q learning method more effective while keeping Q learning and Q learning approximate as baselines.

Introduction

The agent is trained to play cleverly by avoiding the ghosts and eating food and scared ghosts as much as possible. The entire game was emulated.

For Q learning the inputs are states, actions and rewards generated by the game.

For Q learning approximate the inputs are preprocessed features in each state of the game.

For Deep Q learning the images are feed as the inputs.

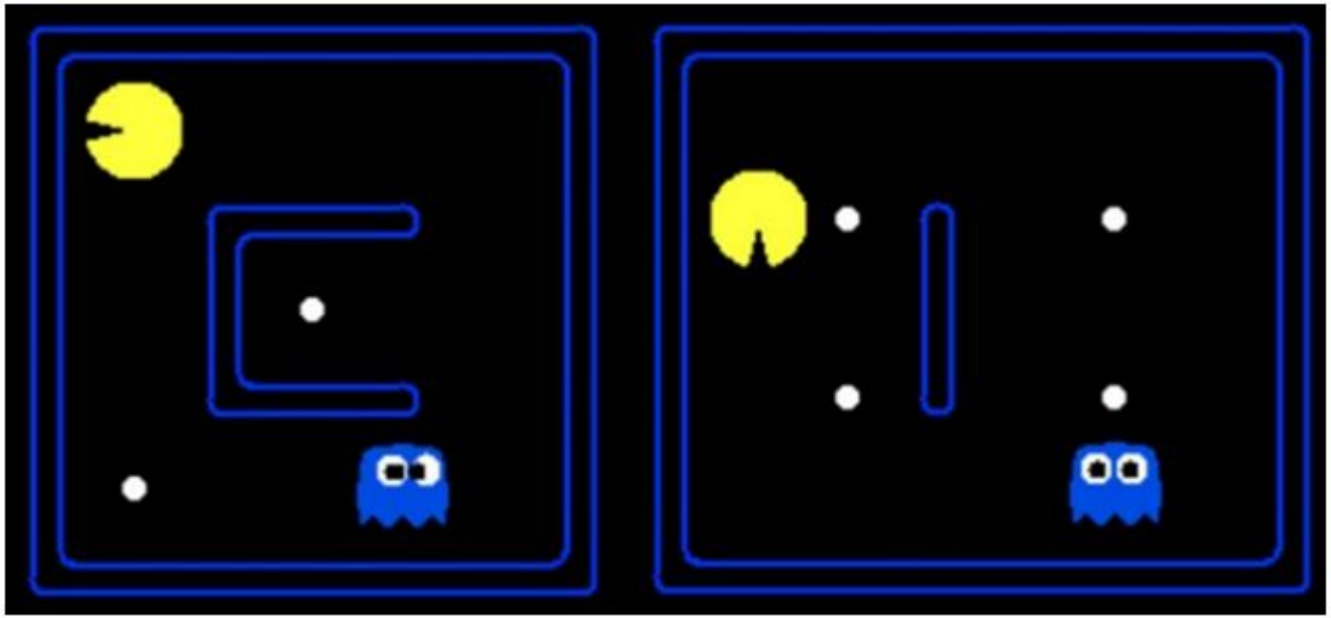
To measure the efficiency they use win rates and scores of each method.

Dataset

The entire dataset is obtained while running the pacman game using the simulator which is available in python. Since the data is captured while running the model the data will not be static based on how the agent plays the game. The pacman games come with different layouts compare to the classic game they have used three of the popular ones for reinforcement learning smallGrid, mediumGrid and mediumClassic.

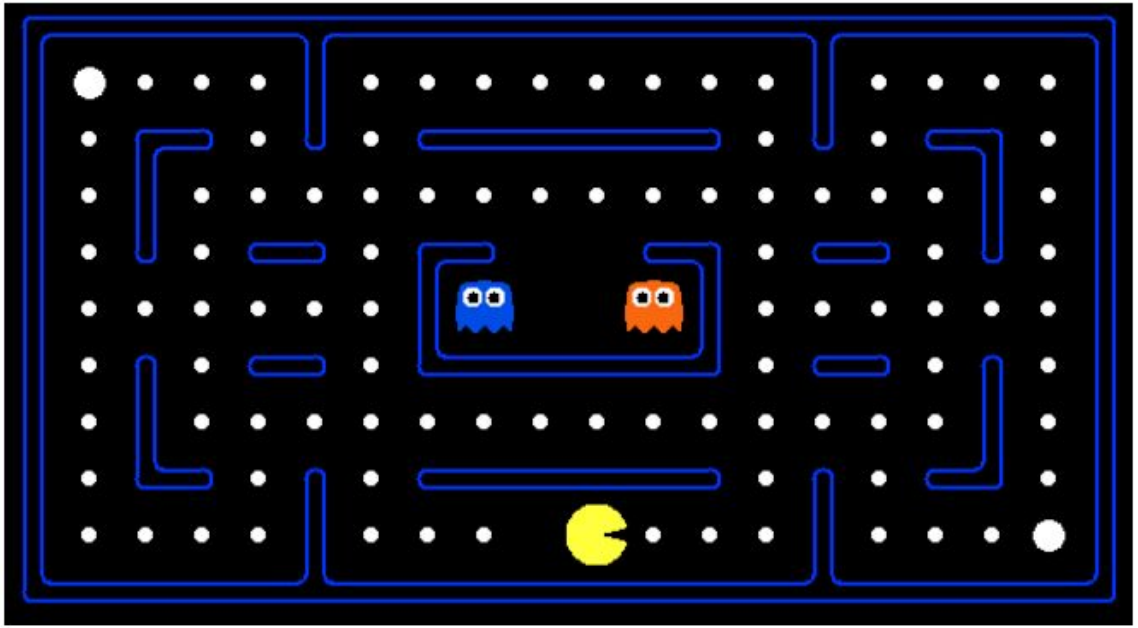
SmallGrid and MediumGrid

The below picture shows a snapshot the smallgrid and medium grid from the simulation



MediumClassic

As the previous slide showed us the 2 of layouts here the medium classic layout is show.



Method

The reinforcement learning tasks follows finite Markov Decision Process

Some of the basic terms used in paper is

- State space S: This is all the possible configuration possible in the game like the position of all the elements in game
- Action Space A: A set of all the actions allowed in the game eg left,right,up etc
- Policy π:S->a: A mapping function from state to action space

The main goal here is to make the agent to take optimal actions to maximize the total reward or we can tells this has the total point earned after playing the game.