Team Documentation- GameAI Pclub Project -mentored by Prannay Khosla

July 8, 2017

Team Members- Anurag Chanani, Rahul Saxena, Tushar Gurjar

Contents

1 Introduction

Battleships is a board game involving two players, each having fictional ships placed on a grid, which is hidden from the other player. A player needs to guess the location of the other players ships, and sink them. The first player to sink all his/her opponents ships wins the game. The objective will be to design such an AI which can do this efficiently.

2   Implementation Details

We will implement a text-based version (the visual board will be coded afterward, in common for the multiple teams). Initially, our program will arrange the ships on our grid and print the same.. After this, we will expect that given a 2D array representing a board (with three kinds of markers, Hit, Miss and Unguessed), we will be able to output the best possible next move. We should also be able to detect when a win/loss condition is attained.

3   Week1:

1. Learned python by LPTHW and Codeacadamy. All sample files that were used for learning python are added to the github repositories.

2. Gone through Commandine and Git tutorials from Codeacedemy.

3. Learned python libraries Numpy, Scipy and MatplotLib that will be used for implementation from
   http://cs231n.github.io/python-numpy-tutorial/ and other resources.

4. Created a basic interface for the game in python. Tested it using a random player (use python random function to make next move in game).

5. Completed tasks given by Mentor

Code for the above in github repository-
https://github.com/rahul7iitk/Game-API

4   Week2:

1. Understood Machine learning idea and how to implement in our problem.

2. Learned machine learning from Andrew Ng online course on coursera (till week 2) -
   https://www.coursera.org/learn/machine-learning/home/welcome

3. Completed tasks given by Mentor-

   (a) Calculating graphs for number of games won v/s number of moves taken.
   (b) Calculating graph for time taken for given number of games.
   (c) Calculating number of games won by comp1 vs comp2 for 10000 games.
   (d) Calculating reward (hit difference) for 10000 games with diff boards and with same board(from start).

Code for all the above tasks is stored in github repository-
https://github.com/rahul7iitk/Game-API

5   Week3:

1. Removed the multiple chance bug from main code.

2. Made alterations to main code for creating no ships region in the board.

3. Studied about python library tensorflow for implementation of Neural Networks from-
   https://pythonprogramming.net/tensorflow-introduction-machine-learning-tutorial/

4. Going through Andrew Ng course week 4-6 for Neural Networks.

6   Week4

1. Learned about Reinforcement Learning from Davil Silver youtube lectures till lecture 3-
   https://www.youtube.com/watch?v=2pWv7GOvuf0&list=PLzuuYNsE1EZAXYR4FJ75jcJseBmo4KQ

2. Completed Reinforcement Learning from Davil Silver youtube lectures till lecture 7

3. Learned about implementation of RL from online resources.

7 Week5

1. Implemented a mini version of Battleship game (10*1 board with one ship of 3 size)

2. Trained an agent using Reinforcement learning monte carlo to play on mini version of battleship game.

3. Successfully trained the above agent and got average of 4 moves to complete the game.

8 Week6

1. Extended the mini battleship game to the actual battleship game (10*10 board with 5 ships)

2. Trained an agent using Monte Carlo Reinforcement learning policy to play on the actual battleship game.

3. Trained our agent for 50k games (trail and error learning) and got an average of 76 moves.

4. Improvised the reward policy taking distance of last move into count and got an average of 74 moves after training 50k games.

Code for all the above tasks is stored in github repository-
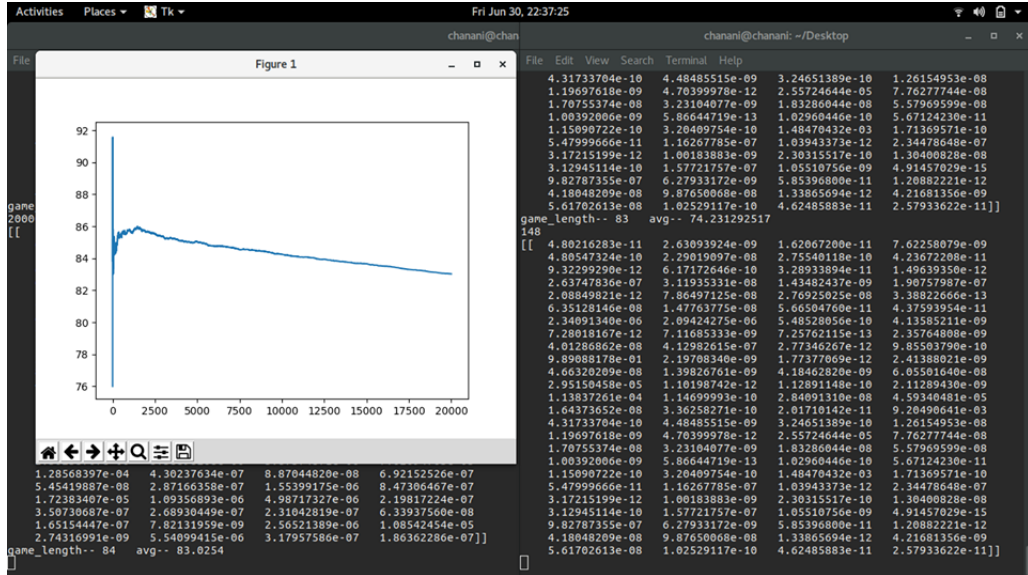https://github.com/rahul7iitk/Game-API
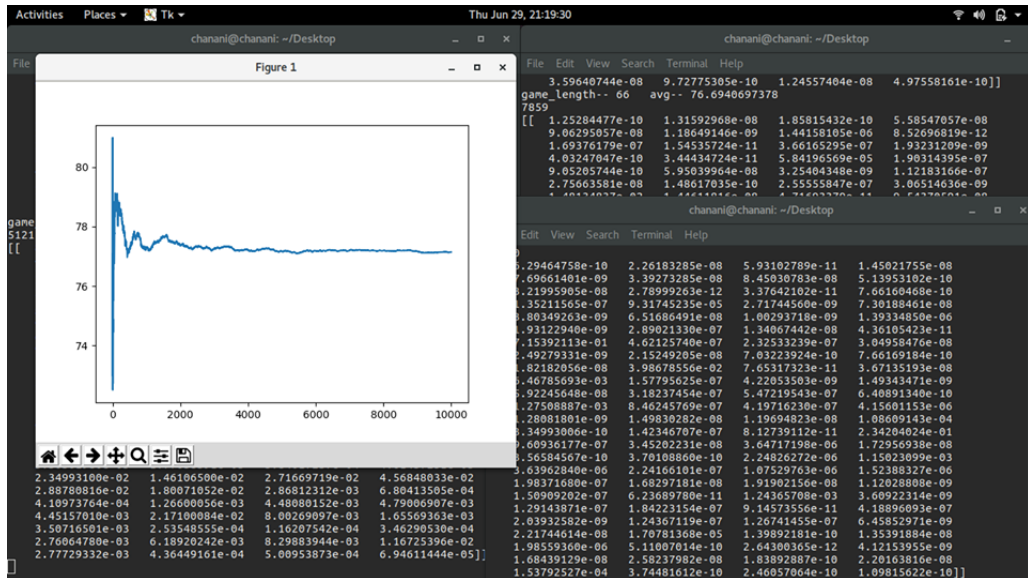
9   Experiments:
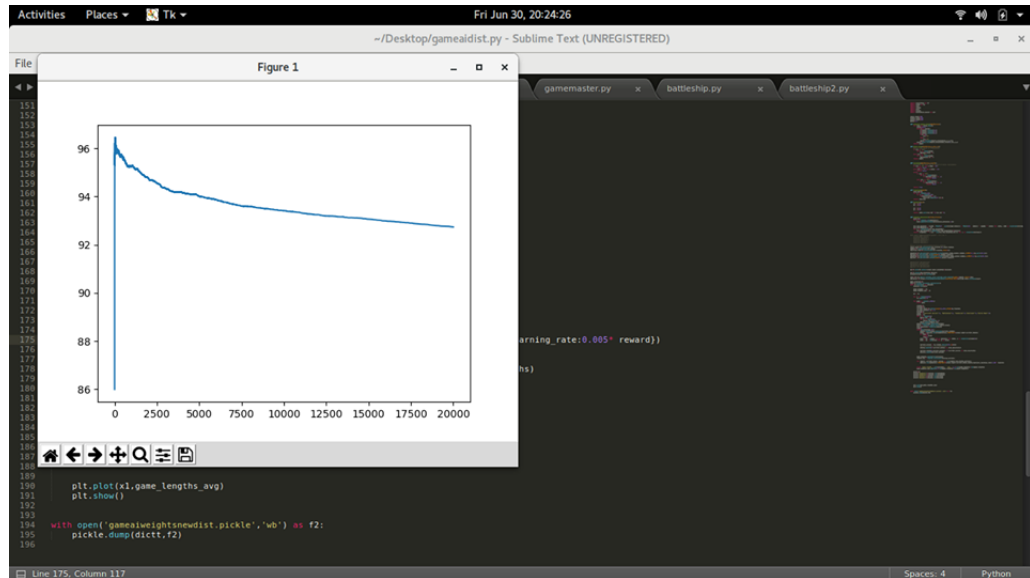
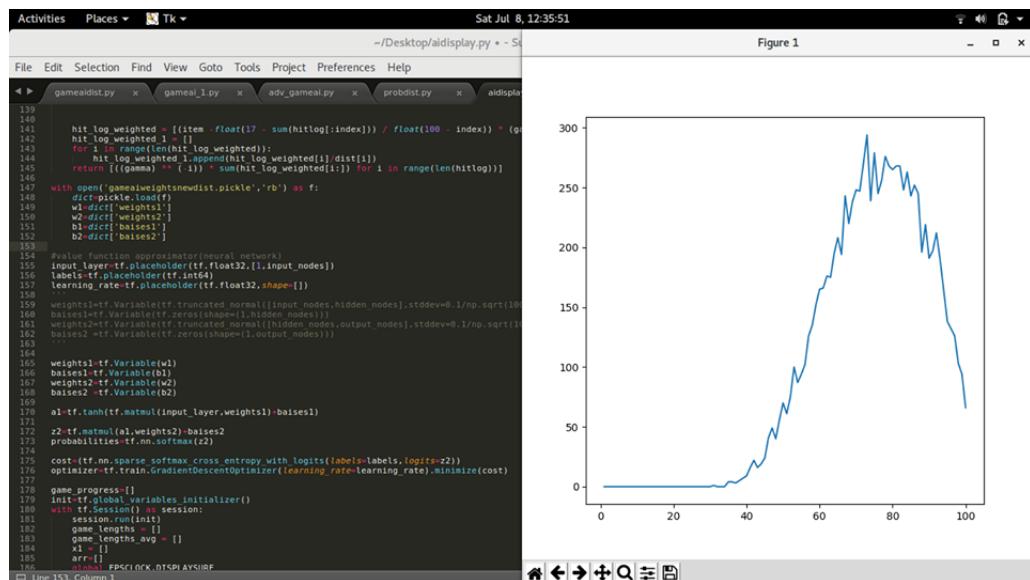Training the agent for first 20k games.
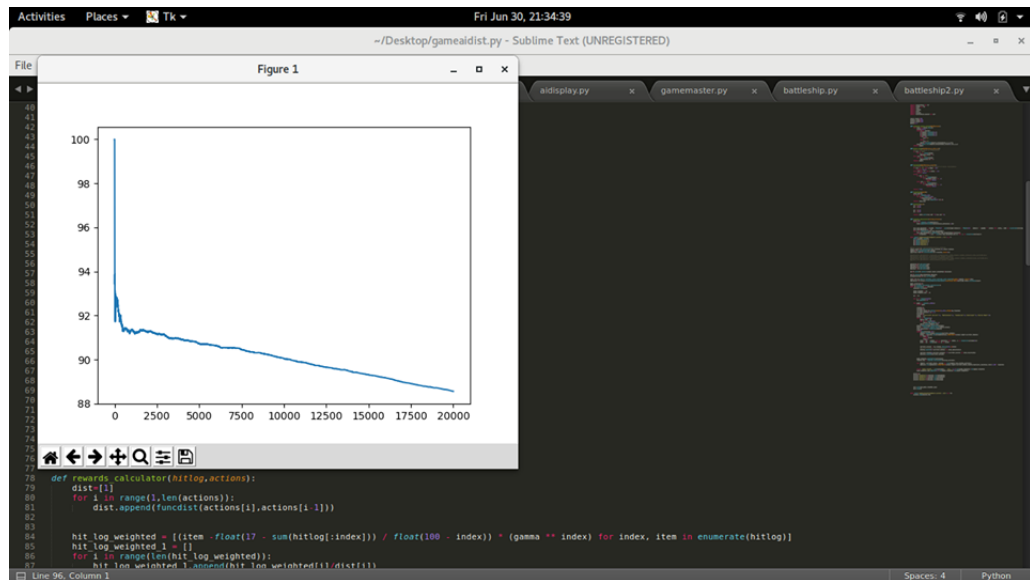
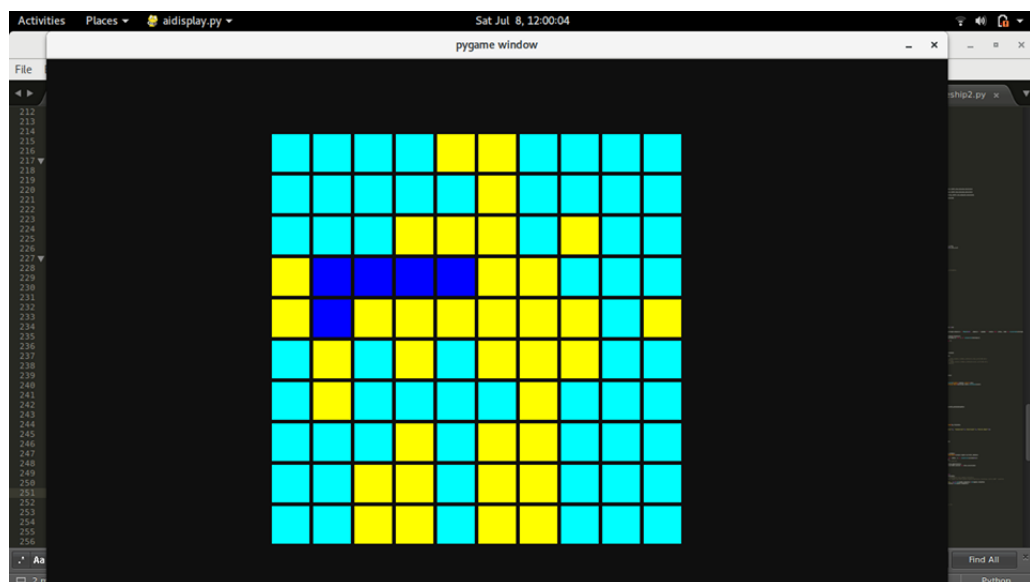Training the agent for next 20k games.

Training the agent for next 20k games.

Training the agent for last 20k games.

Graph showing the average moves our agent takes on the board.

Our interface for the game.