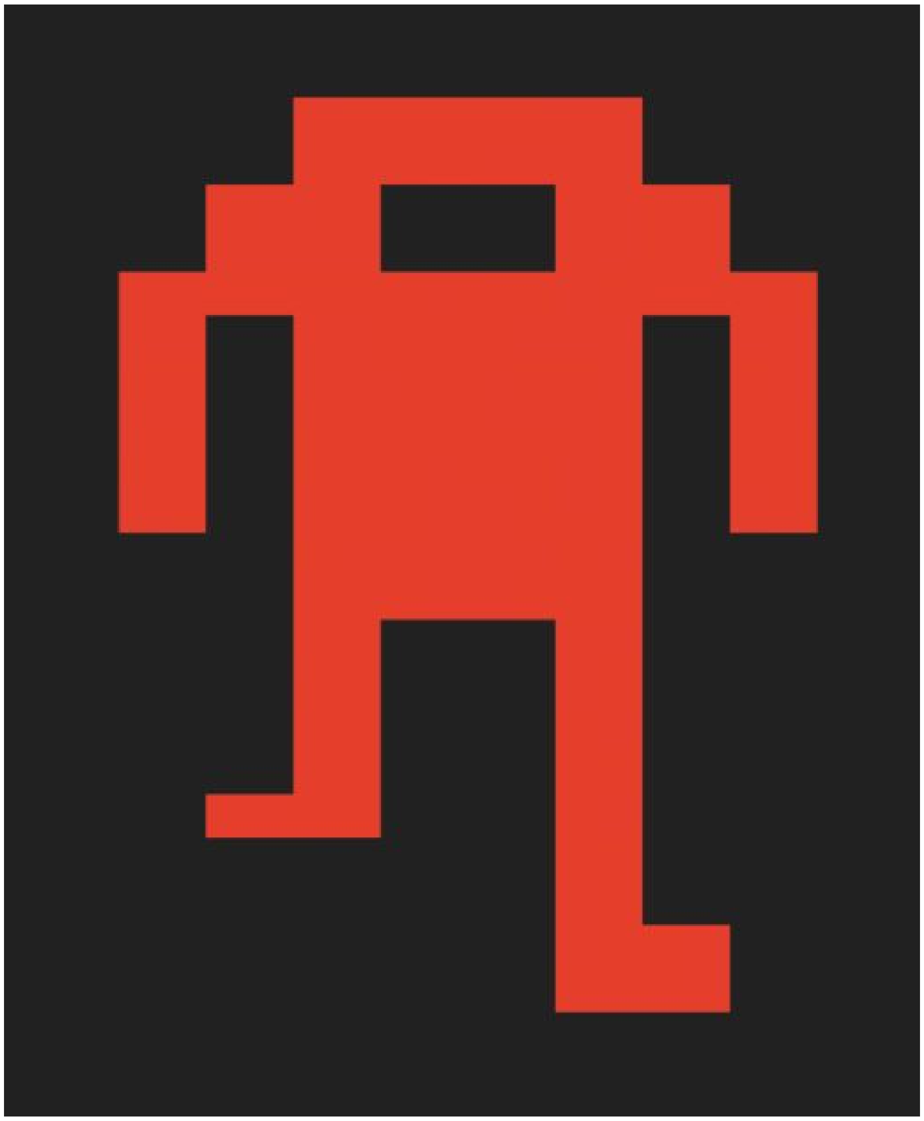
BERZERK REPORT



Nihal Parchand **Rohit Kunjilikattil**

03.02.2019

FOUNDATIONS OF INTELLIGENT SYSTEMS

INTRODUCTION

Atari 2600 was a popular second generation game console, released in 1977. With a 1.19 MHz CPU, 128 bytes of RAM , and no frame buffer, it offers minimal hardware capabilities compared to a modern game console. Nonetheless, Atari 2600 programmers learned to push the capabilities of this limited hardware and there were numerous shooter, action-adventure, and puzzle games developed for the platform. Many popular arcade games, including Space Invaders, Pac-Man, and Donkey Kong were also ported to the Atari 2600. Our focus is on the Berzerk-v0 version of the shooting game Berzerk.

OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms. This is the gym open-source library, which gives you access to a standardized set of environments.

There are two basic concepts: the environment (namely, the outside world) and the agent (namely, the algorithm you are writing). The agent sends actions to the environment, and the environment replies with observations and rewards (that is, a score).

The core gym interface is Env, which is the unified environment interface. The following are the Env methods you should know:

1. reset(self): Reset the environment's state. Returns observation.
2. step(self, action): Step the environment by one timestep. Returns observation, reward, done, info.

   Observation is the 3D array representation of the RGB pixel values of the environment.

   Reward is the points achieved for each action. It is added to the score value for each action and the final score is returned after the game ends.

   Done is a boolean value which is used to check whether the game/episode is terminated or not. It is returned as True if the game/episode has terminated and False otherwise.

   Info is a dictionary containing auxiliary diagnostic information (helpful for debugging).

3. render(self, mode='human'): Render one frame of the environment. The default mode will do something human friendly, such as pop up a window.

Installation

You can perform a minimal install of gym with:

```
git clone https://github.com/openai/gym.git

cd gym

pip install -e .
```

If you prefer, you can do a minimal install of the packaged version directly from PyPI:

```
pip install gym
```

METHODS

Method for Agent1 (Shooting Agent):-

Our line of thinking while trying to maximize the score for the berzerk agent was very simple: detect the presence of enemy, keep shooting till all enemies are dead and then move randomly.

This approach showed a significant improvement in the score as compared to the score of the random agent. This improvement is measured and shown in box plot statistical analysis of the agent's scores. But one of problems of this approach was that we were scanning every pixel of every observation which made the process of shooting extremely slow and therefore we decided to keep the movements random because calculating what move to take next would take up a lot more time making the program painstakingly slow.

Pseudocode for agent1:-

```
    Function Act(observation,reward,done) : -
        returns an action
        For  w in observation
            For  pixel value for every w:
                If not(pixel_value = wall_color or pixel_value=agent_color or
                pixel_value=black or pixel_value = score/logo color):-
            then enemy must be present
        So while enemy_present == true:
        Keep shooting
```

We used this approach of elimination because the enemy color keeps changing at every level. So using this methods we eliminated the other possible values and if it still found a value it must mean that it found an enemy pixel.

Method for Agent2(Shooting and Moving Agent):-

In order to overcome the problems of agent 1 and further improve the maximum score, we decided to first optimize the shooting and then provide the logic for movement. As we were earlier comparing the color of every pixel in every frame of an observation, the process was very slow. So we decided the there was no need to check every pixel of the frame i.e we could break from the inner loop as soon as we found one enemy and then repeat this procedure for a new observation. This showed significant improvement in the shooting timings of Agent2 as compared to Agent1.

For the logic behind the movement decisions, we compared the distance of the agent to the upper wall and unless it was less than 10, we would keep moving up. As the distance dropped below 10, we would determine whether we are on the left or right of the exit based on coordinates as the upper exit had a fixed position. Then as we would reach the fixed exit coordinate we would go up to exit.

Pseudocode for agent2:-

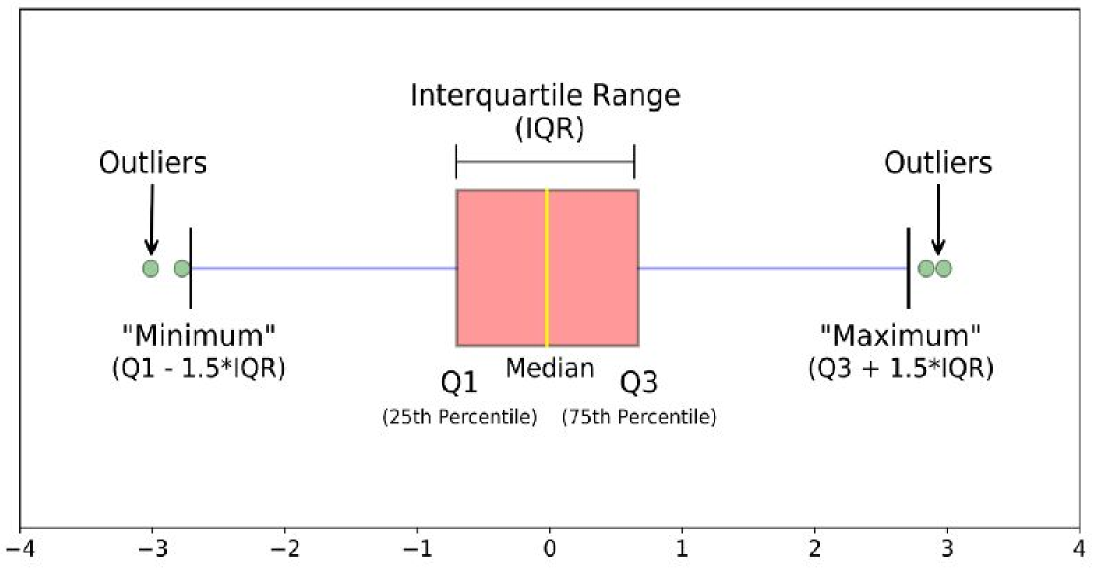
```
    Function Act(observation,reward,done) returns an action :-
     for w in observation :
         for  x,y,z in w:
             If not(pixel_value = wall_color or pixel_value=agent_color or
         pixel_value=black or pixel_value = score/logo color):-
             then enemy_present=true
                 Break
             Else:
                 Continue
             If enemy_present==true:
                 Break
             Else:
                 Continue
    if enemy_present==true:
      Increment shoot counter
      Return shoot_actions[shoot counter%len(shoot_actions)]
```

RESULTS

In order to compare the results of our agents , we did some statistical analysis using Boxplot and Anova methods. We documented the scores that our agents generated over a period of 100 different episodes. And using that information, we did ANOVA analysis of it and represented it using a Box-plot too.

BOXPLOT:-

Boxplot is a way of showcasing data based on minimum, first-quartile, median, third-quartile and maximum. Outliers are represented using circles.



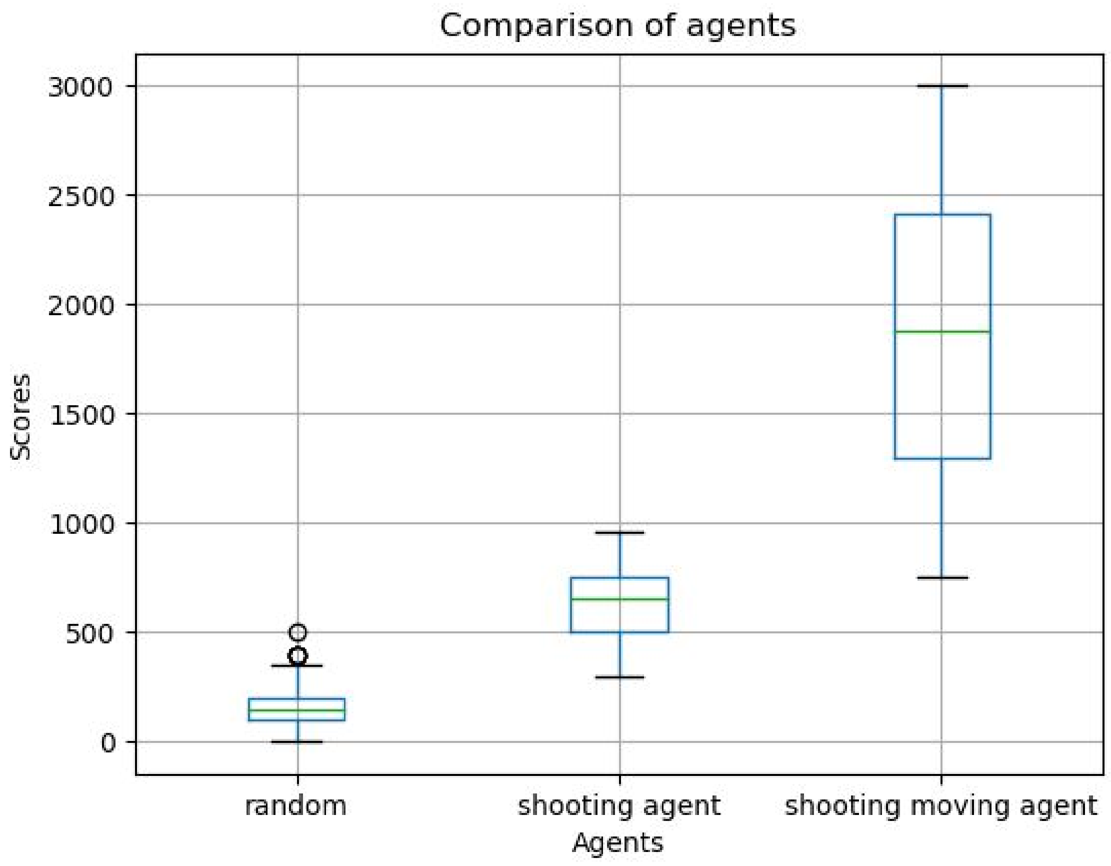
median (Q2/50th Percentile): the middle value of the dataset.

first quartile (Q1/25th Percentile): the middle number between the smallest number (not the “minimum”) and the median of the dataset.

third quartile (Q3/75th Percentile): the middle value between the median and the highest value (not the “maximum”) of the dataset.

interquartile range (IQR): 25th to the 75th percentile.

BOXPLOT ANALYSIS OF AGENT SCORES:-



As shown in boxplot, the agents keep on improving their performance as the boxplot of their scores becomes more indistinguishable and they have increasing maximum scores as well.

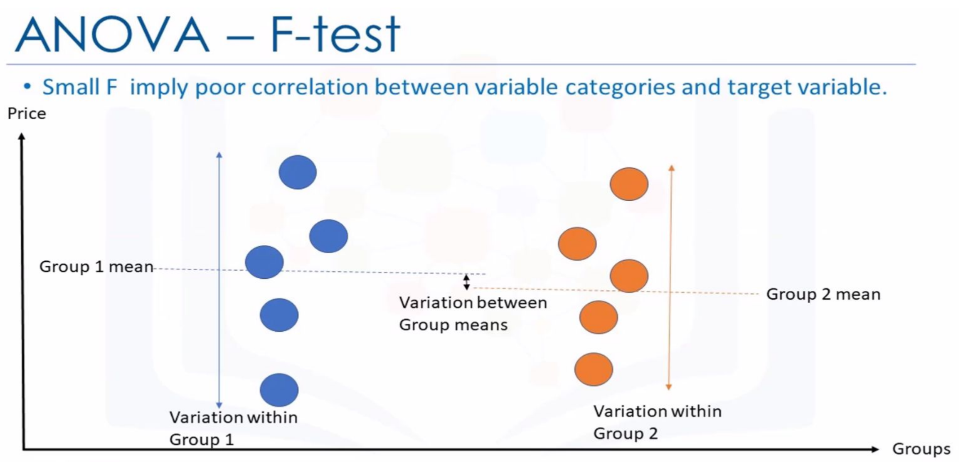
For the random agent, there are a few outliers present as one or two values for example 500 appear once in the set of 100 data points.

ANOVA :-

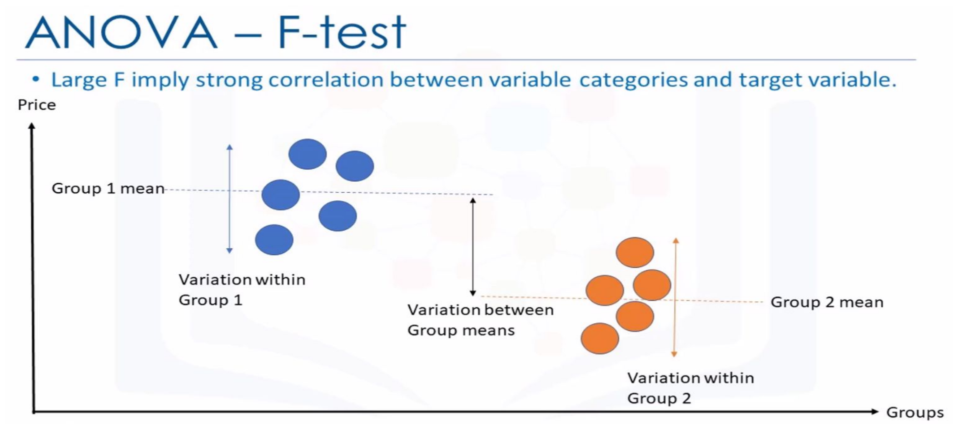
ANOVA is a statistical test that stands for Analysis of Variance. When we have a single variable, we use One-way ANOVA to test the difference between means of two datasets.

The ANOVA test returns two values:

F-test score: The ratio of variations over the groups' mean over the variation between the mean of each of the sample group.


ANOVA – F-test
• Small F imply poor correlation between variable categories and target variable.
Price
Group 1 mean
Variation between Group means
Group 2 mean
Variation within Group 1
Variation within Group 2
Groups

Weak correlation


ANOVA – F-test
• Large F imply strong correlation between variable categories and target variable.
Price
Group 1 mean
Variation within Group 1
Variation between Group means
Group 2 mean
Variation within Group 2
Groups

Strong correlation

Above images shows the dependency of correlation of data on F-score. Smaller F means poor correlation and larger F implies strong correlation

P-value: It tells whether the obtained result is statistically significant.

Range of p-value:

P-value < 0.001 Strong

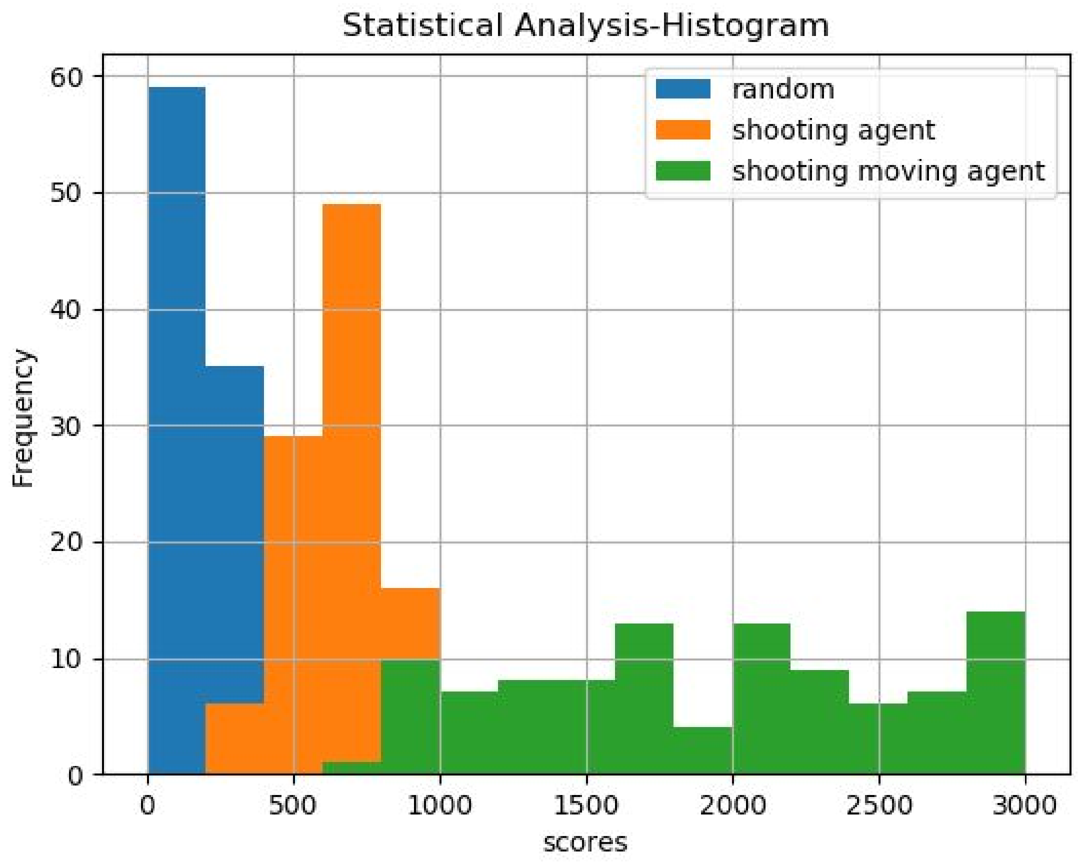
P- value < 0.05 Moderate

P- value < 0.1 Weak

Result obtained after running the One-way ANOVA:

F_onewayResult(statistic=array([505.33401974]), p-value=array([1.25094689e-96]))

Where F-score is 505.33401974 and p-value 1.25094689e-96



COMPARISON OF THE AGENTS' PERFORMANCE:-

Agent Name	Highest Score	Lowest Score	Average Score
Random Agent	500	0	163.9
Shooting_Agent	960	300	638.9
Shooting_Moving_Agent	2810	750	1895

CONCLUSION

So in conclusion , based on our statistical analysis, we can say that the agents that have more information about the environment at each frame tend to provide a higher score but they have a very high time complexity and are very slow to execute.