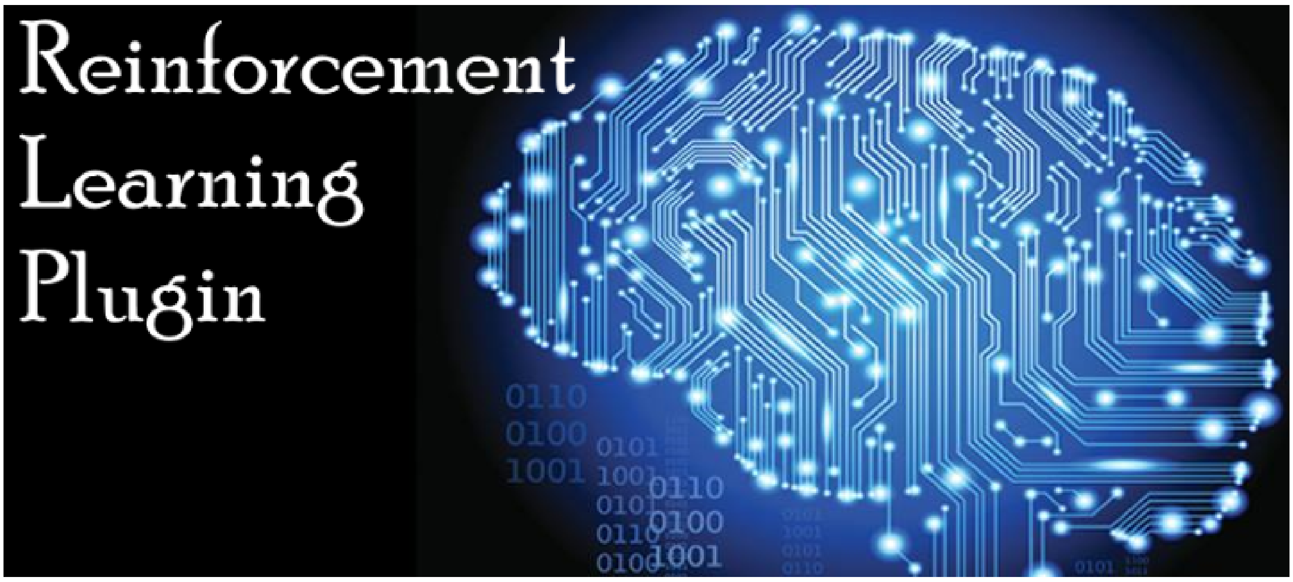
Project in Autonomous Agents
Catch the ball Game
Q-Learning



@Author :Logothetis Fragkoulis
AM:2013030016
Electrical and Computer Engineer
Technical University of Crete
Email : flogothetis95@gmail.com

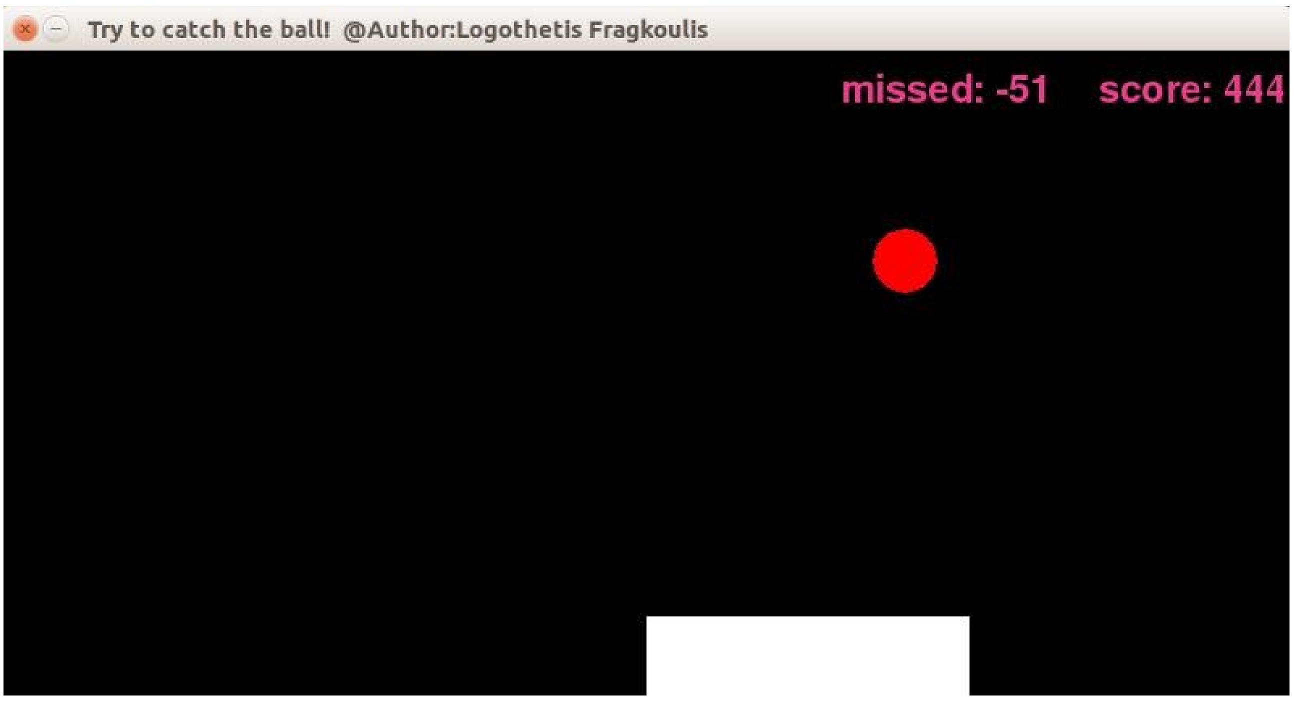
Abstract

The main purpose was to build an Agent ,who could play the Catch-ball Game like an ideal Player .So I used the pygame plug-in of python to built the environment of the Game . I designed a ball and a simple catcher of the ball .In addition , a clever Q-Learner agent was built to force the Agent to play rationaly to maximize its score .

1.The environment of the Game

The canvas of the games contains:
a) Ball ,it is a simple cycle (pygame plugin)
b) Catcher of the ball ,it is a simple rectangular (pygame)

Each pixel size of the above components are demonstrated in the following descriptions.
The ball changes pose at each frame .I set the frame-rate equal to 40 but if anybody wants to see the game in fast mode ,it is easy to modify this parameter.



2.**Analyze Source Code**

2.1 Classes.py

In this class I have defined the Circle instance as part of coordinates and the State (s ,s') which contains the rectangular and cycle coordinates .

2.2 Utils.py

- def new_state_after_action(s, act) : Returns new (s') after (s,a)
- *def new_rect_after_action(rect, act)*: Move rectangular after an action
- *def circle_falling(crclradius):* defines where the starting x position of the circle should be.
- *def calculate_score(rect, circle)*: calculate the score based on the relative position of the circle and the rectangle.
- *def state_to_number(s)*: numpy library array can't work with custom objects as indexes. That's why I created an integer representation of the states. The position of the rectangle and circle combined should give us a unique identifier. We are storing the value in another dictionary which would hold the unique indexes.
- *def get_best_action(s)*: Return the best action

2.2 initializers.py

In this part of code I have specified the circle and the rectangular properties (pixels ,weight,height) .Furthermore ,it was determined the appropriate size of the Q-Array .

```
Number of states = (windowWidth / 8) * (windowHeight / circleYStep) * (windowWidth / rectWidth)
```

According to these : Q= [5000,3]

- 5000 different states
- 3 different actions in each state

2.3 Main.py

Then, at each time the agent selects an action and observes a reward and a new state that may depend on both the previous state and the selected action, is updated. The core of the algorithm is a simple value iteration update, using the weighted average of the old value and the new information:

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

2.4 Reward

The game rewards the Agent at each phase with value +1 if the catcher got the ball . ,value -1 if the ball missed the ball else +0 during the falling time of the ball . We can be informed for the reward -score of the game by looking the score ,which it is presented on the window of the game

2. Influence of variables on the algorithm

This issue was one of the most difficult parts of the project .It can be easily understood the importance of the rates,factors and conditions of Learning .For instance , there are a lot of available techniques to control the learning-ratio(sigmoid ,linear functions ,static number) . I implemented all the above method and the result turned out to be positive for static approach of learning rate .That conclusion is quite logic due to the fact that each iterate of a game has the same importance .Additionally ,Discount-factor forces the Agent trying to learn quicker but in this game after a plenty of tests the static form of that factor gives the best pay-off .Furthermore,all rewards are positive so I chose to set all the Q-array table to zero .

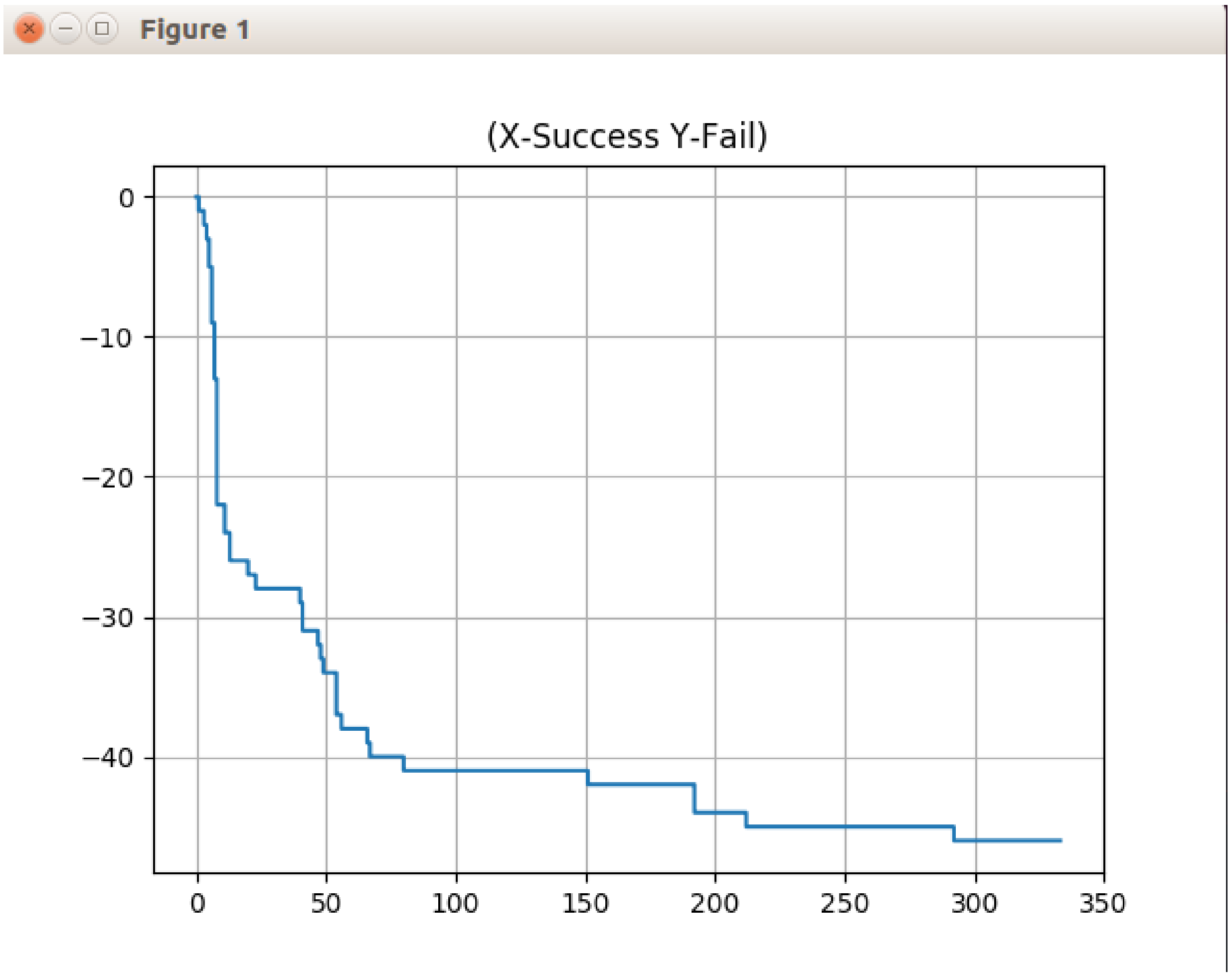
Learning rate = 0.85 static

Discount factor = 0.99 static

Initial conditions (Q0) =Zeros

3.Evaluation of Results

In order to present some figures of my job , I used a math-figures plugin of python. When a user exits from the game or after 1,000 cycles then a math figure will be opened in order to inform us about the Learning phase !



1. X-axis is how many balls are caught as the time passes and the Y-axis is how many balls are not caught
2. It could be observed that the Learning Algorithm getting closer to the Ideal-Player after 50 missing balls

Future Work

-It is argued that is a good idea to learn this Game with Deep Q-Learning Algorithms and Neural Networks .

-If the windows was bigger then a really big amount of memory will be used to keep all the different states . Under those conditions a base function Q-Learning approach will could be a really efficient method .

References

1. https://en.wikipedia.org/wiki/Q-learning
2. http://mnemstudio.org/path-finding-q-learning-tutorial.htm
3. http://people.revoledu.com/kardi/tutorial/ReinforcementLearning/Q-Learning-Algorithm.htm