**COMPUTER SCIENCE AND INFORMATION SYSTEM DEPARTMENT**
**CS F407 - ARTIFICIAL INTELLIGENCE**
**ASSIGNMENT Review - 1**
**First Semester-2018**

## TEAM ID:

## PROJECT TITLE: ***The Breakout Game***


| Roll Number | Name | Department | Remarks |
|---|---|---|---|
| 2016A7PS0076H | Roshini Shetty | Computer Science | |
| 2016A3PS0272H | Utsav Kaushik | Electrical and Electronics | |
| 2016AAPS0159H | Sreekara Reddy | Electronics and Communication | |


## ABSTRACT :


This project aims at building an artificial intelligent system that links the theory of deep reinforcement to games for optimization of performance. It will be pointed out that reinforcement learning results in a series of games with non-decreasing scores, which finally approaches to an optimized approach for the game. In this kind of dynamics, a concepts of rewards and punishments is used, which allows the system to learn by itself without having to worry about fetching the “correct” moves.

# ANALYSIS DOCUMENT

## TABLE OF CONTENTS:

# 1. Introduction:

## A) The Game:

The idea of the game Breakout is that there is a moving platform, a ball and few blocks inside the box and the ball is bouncing off the surfaces. We need to move the platform in such a way that the ball does not fall on the ground until all the blocks are destroyed by the ball. The score is given according to the number of blocks the ball destroys.
For this project, we have only one level with sufficient obstacles to train the **Neural Network** to play the game. The game is hosted in Open AI environment.

### B) The Model:

This model is based on Deep Reinforcement Learning(DRL) and Q-learning intuition.
The model receives the contents of the current state of the environment and outputs an action.
(In this case, it can be ‘move left’, ‘move right’ or stay in the same position). We provide the model with **rewards and punishments** based on it’s every action.
The model finally learns to find the actions which lead to maximum reward.

## 2. Feasibility Report :

### 2.1 Feasibility

There are few pre-existing implementations of the model. The Model can be implemented by using Q-Learning Algorithms and Markov Principle and few other computational algorithms.

#### 2.1.1 Using Q-learning intuition :

In this game, you are given a state **S**, which consists of the locations of all the attributes of the game at certain point of time. The player has to take an action **A**, which in this case will be moving left or right or staying in the same position. As a result, there will be some reward **R** and a new state **S’**.
In Q-learning, we choose our action based on the highest expected future reward.

**The Q-Function is defined as Q(state, action)**

$$Q(S,A) = R + \gamma * \max Q(S',A')$$

The expected future reward is calculated as immediate reward **R** and the future expected reward. We assume that next action **A**’ is optimal and to describe the uncertainty about the future, we introduce a new factor $\gamma$.

### 2.2 The Training Process:

Firstly, we create a batch of experiences **< s, a, r, s' >**. For each possible action **a'**, predict the expected future reward **Q(s', a')**.
Chose the maximum value of all the predictions and calculate the target value for the neural net **[R + γ * max Q(S',A')].**

Now, finally train the neural net using a loss function, this function calculates the difference between the predicted value and the target value**.**

**The Loss Function is defined as:**

$$0.5 * (predicted\_Q(S,A) - target)^2$$

All the experiences are stored in replay memory which acts like a buffer of experiences, which can be further used in the next gameplay.

## 3. Modifications and Exploration:

Always choosing the best action means that the model can miss many things unexplored and may miss something that can give better rewards.
To avoid this and to explore more actions, we can sometimes choose a random action which can or cannot be the best option available.
We can implement the random actions in training method and may get some unexpected results.

## 4. Approach and thoughts:

There are many ways to train such models, we would like to explore more ways to train the model to reach the optimum score.
For example: Using TensorFlow to switch the algorithms and the whole training process can lead to different results.
The ultimate aim is to train the model in such a way that it leads to the maximum score and see if the same model can be applied for different games or other tasks**.**

## References


- https://goo.gl/H8CZNe
- https://en.wikipedia.org/wiki/Reinforcement_learning/
- https://en.wikipedia.org/wiki/Q-learning/