Deep Learning Carom Billiards with Reinforced Neural Networks

Presented by Mrinal Sourav towards the Machine Learning course at Northeastern University, Seattle – Spring 2018

1. Introduction

Carom Billiards is a simple looking game on a pool/billiards/snooker format but without pockets. Players are assigned a ball and take turns to strike that ball. The goal is to bounce off the players ball with two other balls within the same strike to get a score. On scoring, the player continues to get the next strike until they fail to score. The game continues for a finite number of rounds and the player with the maximum score wins the game.

The seeming simplicity of the game hides the complex game dynamics wherein reflection from the boundaries of the table and other balls play a key role. There is also the aspect of trying to win the next round while at the same time making it difficult for the other player if there is a miss. The main objective of the project is inspired by the automation of Atari game by Google DeepMind's Deep Q-learning. I want to train a Deep Neural Network to play a simulation of carom billiards and then observe "superhuman" strategies acquired, if any. If time permits, I would also change parameters of the game or scoring strategies and then observe how well the deep learning model can adapt to new/changing circumstances. Finally, I would also provide fixed circular obstacles on the board to train the model and observe if the model learns to avoid the obstacles or use them!

I hereby present a simple simulation of the game coded in python using **pygame**. For simplicity, some basic rules have been tweaked so that players are assigned the same "q-ball" and take turns to bounce it off the other two balls.

A q-learning methodology will be implemented:

- The "state" would be defined by the position of the balls in the game display.
- The "action" will be defined in angles and discretized speeds of the q-ball.
- The "q-value", to be learned by the deep neural network model, would be represented by the rewards earned in each round (0 for no collision, 10 for collision with one ball and 100 for collision with both the other balls)

Although one popular approach to q-learning is to learn the probability distribution of the set of all possible actions, given a state, I would be using the alternate approach where a **state-action pair** is fed into the model as inputs and the **quality** associated with the action for the state will be learnt as the target for the model. This is done for the sole reason that the **action space for this problem is much bigger than most Atari games reference above**. Specifically, the action space includes each of the 360 angles for the q-ball and speeds ranging from 1 to 10; the total permutation of which is 3600. Most Atari games have an action space of "UP", "DOWN", "LEFT" and "RIGHT", which I believe is much easier for a model to converge to.