# CS 747: Programming Assignment-3
# Optimal Cue-stick Control

Palle Bhavana

November 2023

## 1 Approaches

### 1.1 Approach-1

Used **Expected Sarsa** with **genetic algorithm**
To get finite states, partitioned the table into 32 parts, angles into 18 partitions. The Q table is of dimension (32, 32, 18), where the first dimension denotes state of the white ball, second dimension denotes state of the **target ball** and the third dimension denotes the angle.
Chosen the target Ball that is near to the hole.

$$\text{Reward} = \begin{cases} 10 & \text{balls in next state} < \text{balls in present state} \\ -0.02 & \text{no change in state} \\ 0.00001 * (d1 - d2) & \text{if a ball moves closer to the hole so that the movement happens} \end{cases} \tag{1}$$

From this approach, used **Reward** and **target ball** choosing strategies.
**Problem in the Approach:**
Because the number of simulations are small, this is not able to generalise well in the latter levels, indeed giving the wrong predictions in the simple cases also. Tried to increase number of states but nothing worked.

### 1.2 Approach-2

To Remove finite number of states, thought to use **tile coding** with **linear function approximation and Sarsa**. But this seems very complicated and I did not havetime as well.

### 1.3 Approach-3

Using physics with some heuristics and It worked well, even though it is not having pervious learnings.
**Strategy 1:** The white ball as to collide the ball and make sure that it goes towards hole. For every ball I have stored the minimum distance hole, that along

with white ball subtends **abtuse angle** at the target ball. The abtuse angle is important because if it acute angle then even though the white ball reaches the position of line joining hole and target ball will not execute the force.

**Strategy 2:** Rather than focusing on target ball do 4 iterations on each of the

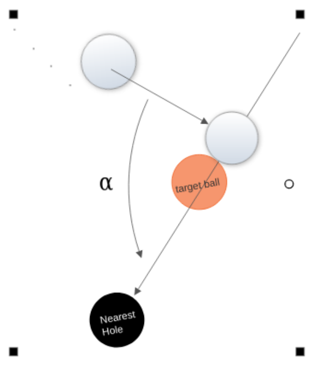

Figure 1: Collision

target ball but the iteration is in the order of minimum distance balls. Based on the best score, store the angle and the force.

## 2 Final Strategy

### 2.1 Abtuse Angle with minimum distance from holes

As mentioned in Approach-3 strategy-1, used this as main strategy for targetting the balls.
Functions used for Implementation:
1) **abtuse_angle**(**point, point1, point2**)
To check whether the white ball and the hole subtends abtuse angle or not.
2) **find_point_on_line**(**value1, value2, k**)
To find the point where the white ball has to hit on the line joining target ball and the hole.
3) **angle_calculate**(**position1, position2**)
Calculate angles with respect to the negative y axis in anticlockwise direction.

### 2.2 Scores:

As mentioned in Approach-1, Same Reward function worked well by tuning the values accordingly.
Functions used for Implementation:
1) **evaluation(self, ball_pos, next_state)**
Based on the present state and next state gives the score for the action

### 2.3 Iterations:

As mentioned in Approach-3, strategy-2, Iterate through every ball not only the target ball.
Functions used for Implementation:
1) **ball_choice(self, ball_pos)**
returns a dictionary, with keys as the ball numbers and the value as the distance and the hole coordinates.

## 3 Possible Improvements:

This approach is completely memoryless, adding reinforcement learning is needed but could not find an approach that could effectively solve this.