CS 747 - Foundations of Intelligent and Learning Agents

Final Project Report

Submitted By :-
Team Name: Dark_Horse
Team Members:
Mustafa Filmwala (163050005)
Rushang Dhanesha (163050024)
Parth Lathiya (163050095)

Carrom

Carrom is a "strike and pocket" table game.

The objective of play is to use a striker disk with a flick of the finger to make contact with and move lighter object disks called carrom men, which are thus propelled into one of four corner pockets.
The aim of the game is to pot (or pocket) one's nine carrom men and the Queen before your opponent.

For the final project, we will develop intelligent agent to win the carrom game.

We have used 3 strategies to solve the problem at hand.

1. SARSA
2. Deep Neural Network
3. Deterministic (Laws of Physics)

At the end results of all methods is compared.

Defining State and Actions for the game of Carrom

The reward for hitting a coin into pocket is kept to be 5. Other rewards (4, 3, 2, 1, 0) are given based on the nearness of the coin to the target pocket.

To create binary features we divided the given carrom board into small boxes of coin’s size. Stride length to create such boxes is kept one-fifth the length of coin.

If the coin is present at that position the feature is marked as 1 else it is 0.

We provided this state as a set of feature to the input of a neural network. The output of the neural network will be the position of the box to be hit, striker position and force.

For 2 player game:

We extended this model by providing negative reward(-5) for pocketing opponents coin. The state of each box also changed - it can be blank or have opponent’s coin or our coin.

SARSA

SARSA stands for State-Action-Reward-State-Action. In SARSA, the agent starts in state 1, performs action 1, and gets a reward (reward 1). Now, it’s in state 2 and performs another action (action 2) and gets the reward from this state (reward 2) before it goes back and updates the value of action 1 performed in state 1.

Let Q(s, a) denote the reward in the next step for taking action a at state s and Z_t(s, a) denote the trace for state–action pair s, a.

Learning rate (α):

The learning rate determines to what extent the newly acquired information will override the old information. A factor of 0 will make the agent not learn anything, while a factor of 1 would make the agent consider only the most recent information.

Discount factor ($\boldsymbol{\gamma}$):

The discount factor determines the importance of future rewards. A factor of 0 will make the agent "opportunistic" by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward. If the discount factor meets or exceeds 1, the Q values may diverge.

The State-Action values for next state is calculated as:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t Z_t(s,a), \qquad \text{for all } s, a$$

where

$$\delta_t = R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1}) - Q_t(S_t, A_t)$$

and

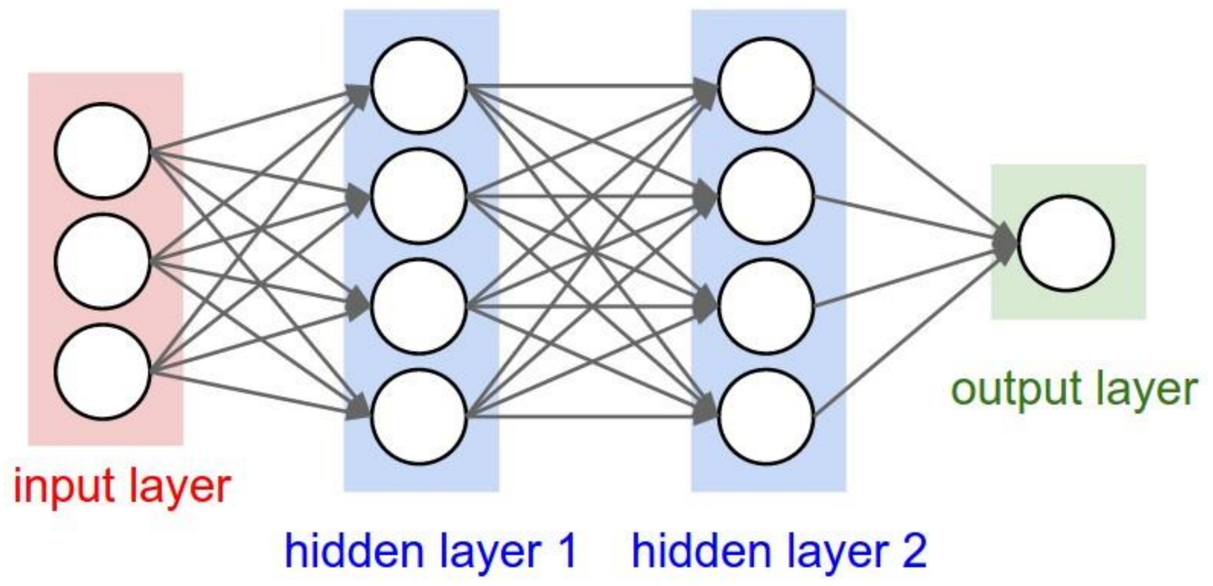
$$Z_t(s,a) = \begin{cases} \gamma\lambda Z_{t-1}(s,a) + 1 & \text{if } s = S_t \text{ and } a = A_t; \\ \gamma\lambda Z_{t-1}(s,a) & \text{otherwise.} \end{cases} \qquad \text{for all } s, a$$

The algorithm to implement SARSA is shown below.

```
Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
Repeat (for each episode):
    $Z(s,a) = 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
    Initialize $S$, $A$
    Repeat (for each step of episode):
        Take action $A$, observe $R$, $S'$
        Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        $\delta \leftarrow R + \gamma Q(S',A') - Q(S,A)$
        $Z(S,A) \leftarrow Z(S,A) + 1$
        For all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:
            $Q(s,a) \leftarrow Q(s,a) + \alpha\delta Z(s,a)$
            $Z(s,a) \leftarrow \gamma\lambda Z(s,a)$
        $S \leftarrow S'$; $A \leftarrow A'$
    until $S$ is terminal
```

When this agent was tested for 1 player mode the least number of turns required to clear the carrom board came out to be 143.

Deep Neural Networks



- The neural network contains 3 hidden layers.
- Each layer contains 83 neurones.
- Learning rate of the network is 1e-3.

This output will point to the best rewarding coin on the carrom. We trained this neural network by deciding what could have been the best shot by deterministic algorithm. For training, the INITIAL_STATE was kept with small number of coins on the board.

After training it, it was able to clear board in 107 moves at best in 1 player mode.

Deterministic(Laws of Physics)

We have tried to hit the coin with 5 different types of shots:

1. Direct Shot:
   Shot when striker, coin and pocket are collinear).
2. Trick Shot:
   Shot which hits coins stuck in the left or right portion of board where direct shot is not possible.
3. Upside Trick Shot:
   Shot which hits coins stuck on top of board and are not possible to hit using direct or trick shot.
4. Downside shot:
   Shot which hits coins below the strike line
5. Touch shot:
   Shot which hits the coin slowly to make it accessible for other shots. This is last possible shot.

Conclusion

- For 1 player mode:
  SARSA: 143 moves
  Deep Neural Network: 107 moves
  Deterministic(Laws of Physics): 27 moves

Therefore, we can conclude that for 1 player mode by applying laws of physics we can get clear the board in least number of turns.

- For 2 player mode:

We played the 2 player game by running the agents against each other. The output in this case was also same.

Deterministic Agent won against both SARSA and DNN Agent

So we can conclude that Deterministic Agent is best among these 3 agents.