Indian Institute of Technology Indore

Discipline of Computer Science and Engineering

Minor Project in the course “Computational Intelligence”

Spring 2022-2023

Title: Learning To Play Connect-4 Using Reinforcement Learning

Abstract:

The goal of our project is to determine the best Connect Four playing method given a Connect Four board. The sequence of turn actions that maximizes a player's chances of winning a single Connect Four game is what we refer to as an optimal Connect Four playing strategy.Connect-4 game using reinforcement learning would involve designing and implementing a game engine and a reinforcement learning algorithm to train an AI agent. The project would include integrating the game engine with the algorithm, training the agent to play, and evaluating its performance. The final product would be a game engine and an AI agent that can play Connect-4 at a level comparable to or better than human players. The goal is to create an AI agent that can play optimally, as well as to understand the underlying techniques and methods used to achieve this.

On October 1, 1988, James Dow Allen and Victor Allis independently solved Connect 4. (October 16, 1988). Reference [1] delves deeply into Connect 4 and offers the best game plans for both players. A common framework that works well for issues where the environment can be simulated but heuristics on the best course of action are lacking is Monte Carlo Tree Search (MCTS) [2]. The AlphaGo [3] program, which was created by combining MCTS and neural networks, excelled in the game of Go. Minimax updates or heuristics are used into several MCTS variations [5] [4] to enhance the software's performance. Variants that spread confirmed wins/losses have also been developed and proved to be empirically superior in sudden-death games [6].

Approach for this project would include the following steps:

1. Designing and implementing the game environment.
2. Choosing an appropriate algorithm such as Q-learning, SARSA, DQN, PPO or A2C and implementing it to train the AI agent.
3. Training the AI agent by playing the game repeatedly with the agent and allow it to learn from its experiences and improve its strategy over time.

4. Test the AI agent against human players or other AI agents to evaluate its performance and identify areas for improvement

Problem Description

Our project attempts to find an optimal Connect Four playing strategy given a Connect Four board. We define an optimal Connect Four playing strategy as the sequence of turn actions that maximizes the probability of a player winning a single Connect Four game.

Reinforcement Learning:

Reinforcement learning is a type of machine learning that involves training an agent to take actions in an environment in order to maximize a cumulative reward signal. It involves exploring the environment, making decisions based on past experiences, and adjusting the decision-making process based on the outcomes of those decisions. The goal of reinforcement learning is to develop an optimal strategy for the agent to take actions that lead to the highest reward over time.

Goal:

Our goal to make AI learn the Connect Four game is to train an agent to play Connect Four at a high level of proficiency, such that it can play against human players or other pre-trained AI players. The agent should be able to understand the rules of the game, learn from past experiences, and use this knowledge to make optimal moves that lead to winning the game. The ultimate goal is to develop an AI player that can compete at a level comparable to human experts or better.

Algorithm:

DQN -

A basic distinction of reinforcement learning methods is that of ”on-policy” and ”off-policy” methods. On-policy methods attempt to evaluate or improve the policy that is used to make decisions, whereas off-policy methods evaluate or improve a policy different from that used to make decisions. Q-learning is an off-policy method. The reinforcement learning model consists of an agent, a set of states S, and a set of actions A of every state s, $s \in S$. In Q-learning, the agent can move to the next state s’ , $s' 0 \in S$ from state s after following action a, $a \in A$, denoted as $s \rightarrow s'$. After finishing the action a, the agent gets a reward, usually a

numerical score, which is to be maximized (highest reward). In order to achieve this goal, the agent must find the optimal action for each state. The reward of current state s by taking the action a, denoted as Q(s, a), is a weighted sum, calculated by the immediate reward R(s, a) of moving to the next state and the maximum expected reward of all future states' rewards:

$$Q(s, a) = R(s, a) + \gamma(\max_{a'} Q(s\ 0\ , a0\ ))$$

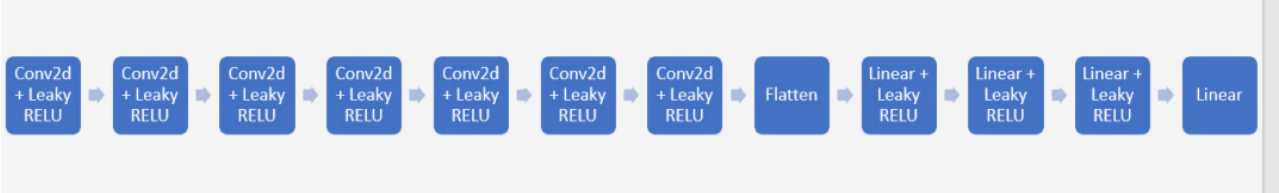
where a 0 ∈ A0 , A0 is the set of actions under state s 0 . γ is the discount factor of maxQ(s 0 , a0) for the next state s 0 . Q(s, a) can be updated by online interactions with the environment using the following rule:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(R(s, a) + \gamma(\max_{a'} Q(s\ 0\ , a0\ )))$$

where α ∈ [0, 1] is the learning rate. The Q-values are guaranteed to converge by some schemas, such as exploring every (s, a), which should be ensured by a suitable exploration and exploitation method (such as -greedy).

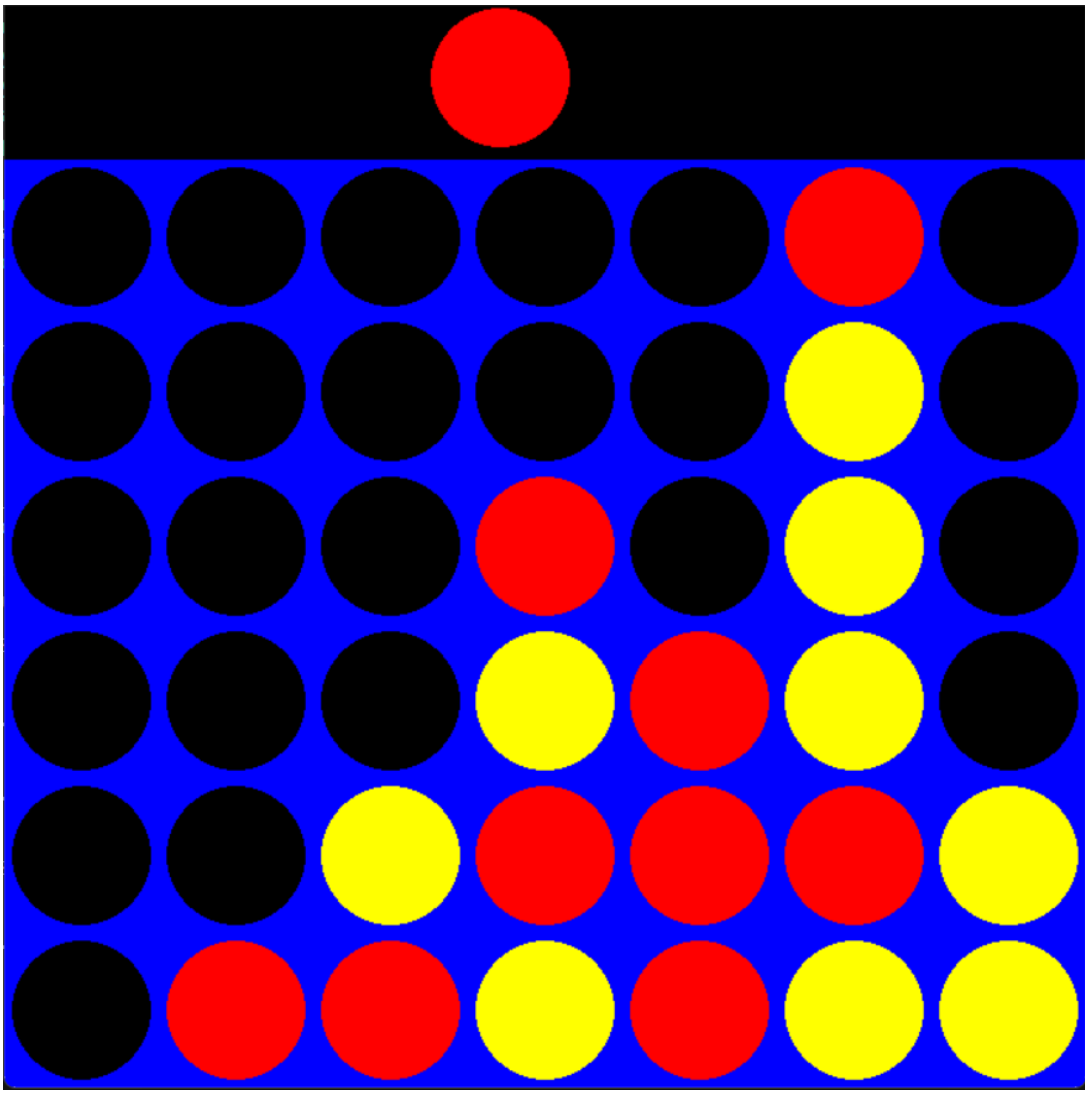
Implementation:

1. Class ***board*** has implementation details of the 6x7 connect4 board. This class maintains the number of rows and columns and also the state of board. It has some helper functions too which performs tasks like tracking the move made by the agent, checking if the move is valid or not and also if the game has ended.
2. Allow the player to play games against itself and generate episodes from each game. These episodes will be used to train the network. In the code, the class ***game*** has a function ***get_episode()*** which generates episodes from each game.
3. Each episode contains a state in which the agent was present, the action which the agent took while in that state and the reward it received by performing the action. Using this three element tuple, we train our neural network. The input to the network is a 2d array which represents the state in which the agent was present along with one row representing the action it took while in that state. The output of the neural network is Q-value which is calculated using reward using the above described equation. In the code, the class ***game*** has a function ***train()*** which trains the neural network after converting reward to Q-value.
4. The architecture of the neural network is as follows:



The network has ***7 convolution*** layers with ***Leaky RELU*** activation and finally ***4 linear*** layers. This network has been implemented using pytorch. Class ***architecture*** has implementation details of the network.

5. After training is complete, we can also test the model by running ***play.py.*** A Graphical User Interface has been implemented using pygame in which humans can play with the trained model and test its learning.



Github Link:

https://github.com/bhavyacontractor/connect4-rl.git

References:

[1] James Dow Allen. The Complete Book of Connect 4: History, strategy, puzzles. Sterling, 2010.

[2] Guillaume Chaslot et al. “Monte-Carlo Tree Search: A New Framework for Game AI.” In: Jan. 2008.

[3] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: Nature 529 (Jan. 2016), pp. 484–489. DOI: 10.1038/nature16961.

[4] Hendrik Baier and Mark Winands. “Monte-Carlo Tree Search and Minimax Hybrids with Heuristic Evaluation Functions”. In: Dec. 2014, pp. 45–63. ISBN: 978-3-319-14922-6. DOI: 10.1007/978-3-319-14923-3_4.

[5] Hendrik Baier and Mark Winands. “Monte-Carlo Tree Search and minimax hybrids”. In: Aug. 2013, pp. 1–8. ISBN: 978-1-4673-5308-3. DOI: 10.1109/CIG.2013.6633630.

[6] Mark Winands, Yngvi Björnsson, and Jahn-Takeshi Saito. “Monte-Carlo Tree Search Solver”. In: Sept. 2008, pp. 25–36. ISBN: 978-3-540-87607-6. DOI: 10.1007/978-3-540-87608-3_3.

Team Members:

Name1: Bhavya Contractor	Reg. No: 200001018	Sign.:
Name2: Priyansh Jaseja	Reg. No: 200001063	Sign.:

Under the Supervision of

Dr. Aruna Tiwari

Professor, CSE