CSCI 5622 : Machine Learning
Connect Four Bot using Reinforcement Learning

Team Members:
Anupama
Harshini Muthukrishnan
Prashil Bhimani
Santosh ALN

Motivation

Every child play a lot of games, be it board games, atari games, outdoor sports or video games. These games are considered as a very important part of a child's mental development as there is not a fixed strategy followed in these games but more on a reward based system where every move gives certain rewards in terms of points, scores or just more similarity to the end goal. More and more they play the game, the better they become at it. This type of learning is basically known as reinforcement learning.

With the advancement in AI field where scientists are trying to mimic a human brain it was obvious that reinforcement learning has gained traction. Reinforcement learning is learning to analyze a current state and take an action that maximizes a future reward, through continuous interaction. That is, the model takes different actions and awards itself when it does something well. In this type of learning , the model tries to improve by retro-feeding itself i.e, model is trained by self play. The goal is to maximize the reward over the lifetime of the task of the agent or model.

Connect Four is one of the games that is widely played by children and hence we are planning to train a bot to play Connect Four game using reinforced learning.

Why use reinforcement learning

At every move in a game, the task is to decide what action to take, given a current state. If we had training data, say, 1000 'states' and their corresponding 'most optimal action', then we're done! We can train any supervised learning model to tell us what to do given a state. But for a game like Connect Four, the solution space is gigantic. To implement the model using supervised learning, we will need billions of training data which was collected based on the games played by the human. But we don't have that data! That's where the reinforcement part comes into picture. We are going to initially take arbitrary decisions and as we move forward, we are going to either encourage or discourage those past actions based on the reward. If we try to use a fixed strategy approach by using Min-Max we have a total of 4,531,985,219,092 different games that can be played. Video games suits best to this type of learning as scores in the game serves the role of reward perfectly. The model objective is to maximize the reward .

Alternate of the dataset

For reinforcement learning there is no need for a train or a test data. Instead the model will play against different agents to test its effectiveness once it is trained. The agents can be actual humans or a random play or a strategy based agent. For our model we are planning 3 kinds of player to play against our bot.

1. Random agent - Will play a random move
2. Strategy agent - Will play a game move based on fixed strategy.
3. Iterative Deepening Min-Max agent Will generate a d depth min max tree and evaluate the best move.

The main goal of the reinforcement learning is to deal with agents that have to learn sequence of actions. There is some sequence of decisions which needs to be made and there is some dependencies between the decisions. So we assume that information from the environment is captured in the form of a state.

Possible approaches

To decide a policy that the agent will learn we are planning to use Neural Networks. We are planning to use temporal difference learning (TDL), a well-known variant of the reinforcement learning approach, in combination with n-tuple networks to the game Connect-4.

The TDL algorithm is used to learn a value function by setting up an agent,who plays a sequence of games against itself. It learns from the environment, which gives a reward $r \in \{-1.0, 0.0, 1.0\}$ for { Yellow-win, Draw, Red-win } at the end of each game.

Each n-tuple value is a sequence of n different board locations where each value in a sequence represents a specific cell of the board.Each board location has a state which tells about the next move of the agent.To create an n-tuple of length K, we start at a random cell, which is the first cell of the n-tuple. Afterwards, we visit one of its 8 neighbors and add it to the n-tuple (if not already in there). We continue to one of its neighbors, and so on, until we have K cells in the n-tuple.

The n-tuple network induces a mighty feature space and the agent learns to select the right features.The n-tuple network is an important ingredient for the overall success and identify several aspects that are relevant for achieving high-quality results.