ENPM690

Homework #4

Name - Kartik Madhira

UID -116218795

1. INTRODUCTION

Connect Four is a two-player a two player connection game in which the players first choose a color and then take turns dropping one colored disc from the top into a seven-column, six-row vertically suspended grid. The pieces fall straight down, occupying the lowest available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs. The first player can always win by playing the right moves.[I]
The aim is to train a RL agent using **AlphaZero** algorithm developed by DeepMind, as more of a generalization of AlphaGo. This algorithm uses a **Deep convolutional residual neural network** that uses **Monte-Carlo Tree Search** (MCTS) to optimize the the search for future states.

2. ENVIRONMENT

In order to simulate the board game, a 6 rows and 7 column grid is created and player A chooses to put on ‘X’ on a particular location and the player be an ‘O’. Each location in the grid is a state and the image of the grid itself is used as input to the model. Since row traversal is not possible, the action space is limited to the number of columns which is 7.

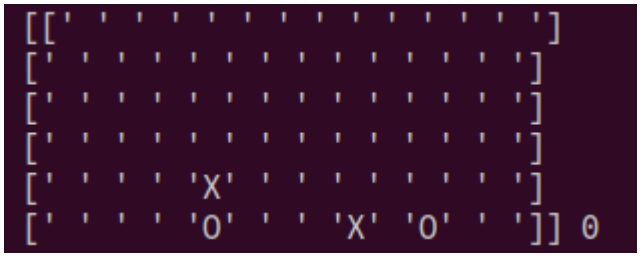

Fig.2 – Connect 4 grid setup

3. IMPLEMENTATION

We use the AlphaZero training code by Wee Tee Soh [II] to train the the agent along with MCTS search optimization. The overall approach consists of a deep convolutional residual neural network, implemented in PyTorch with the input being the connect4 board state. With the help of MCTS, which guides the algorithm to search for outputs in the future states guided by a policy. This policy is the probability of the next moves from the current state.
The neural network is trained on two outputs – sum of mean squared error and cross-entropy losses.
A game can be described as a tree in which the root is the board state and its branches are all the possible states that can result from it. As the game progresses, the number of possible states increases exponentially. MCTS algorithm helps here by searching in a smarter and efficient manner. MCTS uses Select, Expand and Evaluate and Backup as one search pipeline for each simulation from root node. Each simulation of MCTS from root node until end of game for that iteration is known as **MCTS self-play**.
We run about 800 such simulations from the root node and then direct a policy for the root node where it is defined to be proportional to the visit to direct child nodes. This policy then be used to select next action and this new board state will be treated as root node for MCTS simulations.

4. TRAINING RESULTS

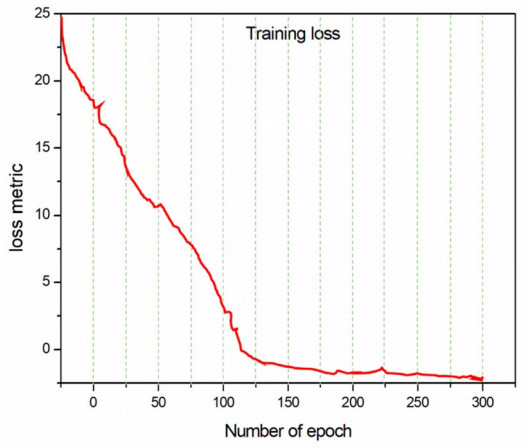

Fig.3 – Training loss vs Epoch for one of the iterations

Each iteration the neural net is trained using the play data and the previous version is used as the other connect4 player. The winner is chosen as the next iteration game candidate. The table consists of the training vs epochs iterated, which is fairly straightforward with negligible loss around the 290th epoch.

5. **REFERENCES**

I. https://en.wikipedia.org/wiki/Connect_Four
II. https://github.com/plkmo/AlphaZero_Connect4
III. https://towardsdatascience.com/from-scratch-implementation-of-alphazero-for-connect4-f73d4554002a