Convolutional Reinforcement Learning: Teaching a Neural Net to play Connect 4

Kyle Schutter

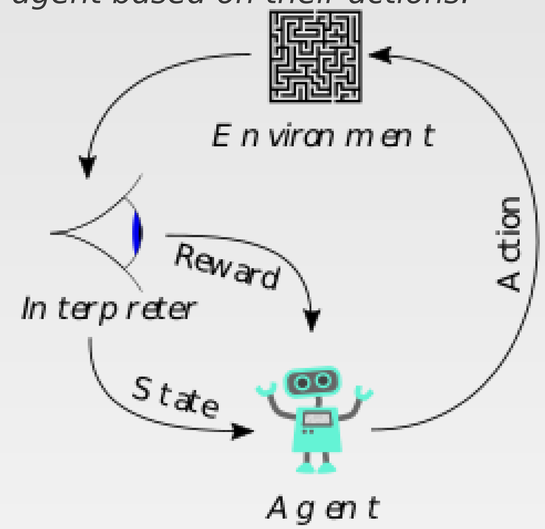
/github.com/kschutter

/in/kylemschutter

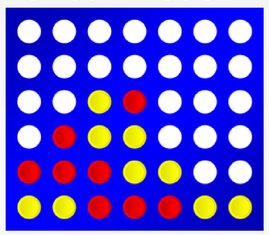
kmosswork+captsone@gmail.com

Background

Reinforcement learning is the machine learning paradigm of allowing our "agent" to explore the environment, and to maximize some cumulative "reward" that we feed back to agent based on their actions.



Connect 4 is a board game that has a discrete environment (or "state") that can be easily encoded into an array of integers for our model.

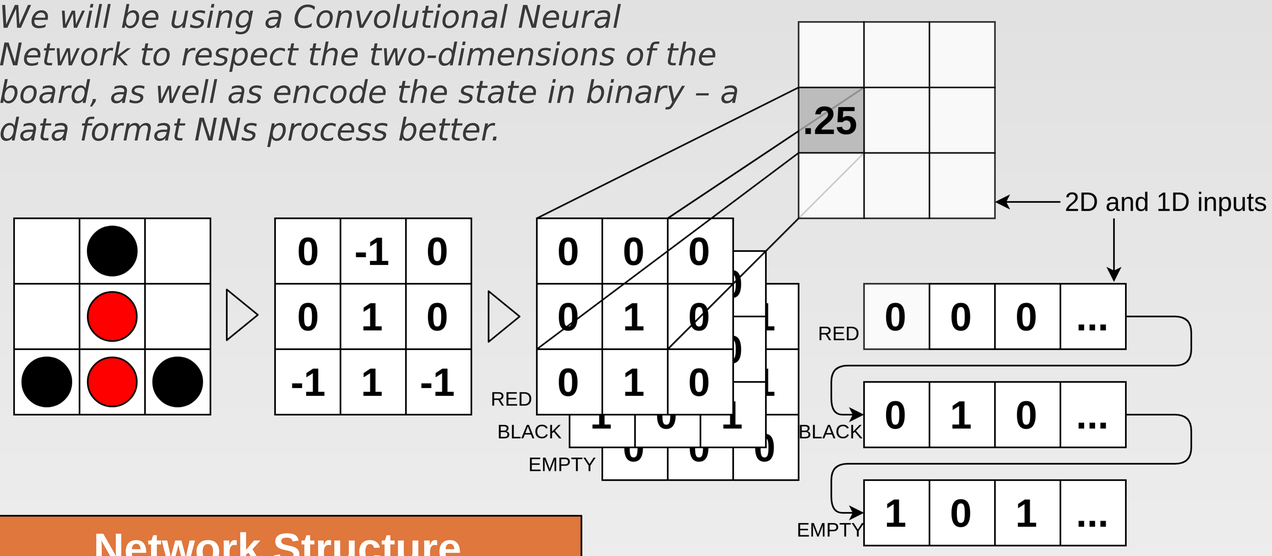


0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	-1	0	0	0
0	-1	1	1	0	0	0
-1	-1	-1	1	1	0	0
1	1	-1	-1	-1	1	1

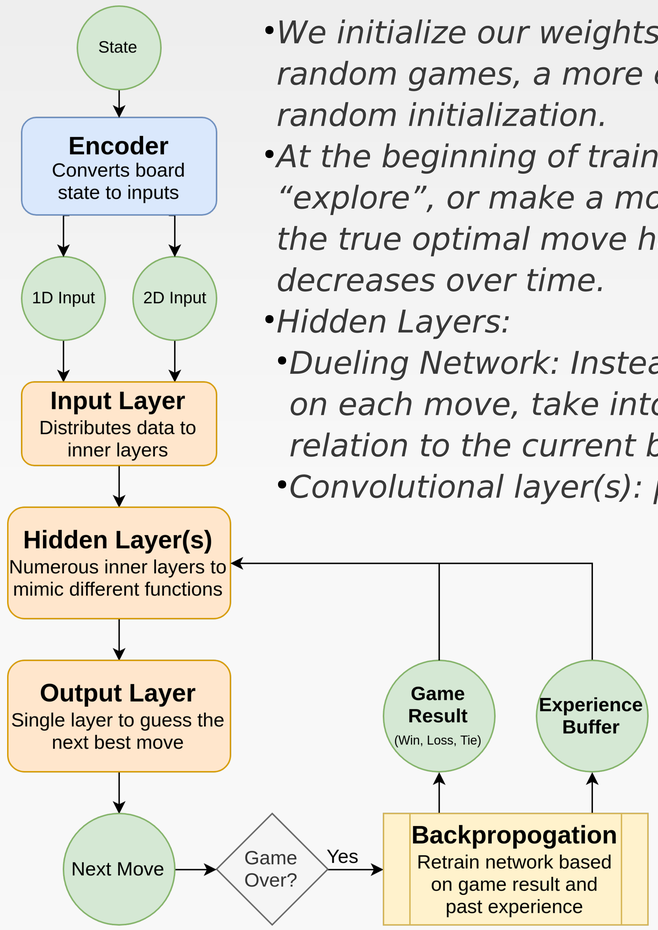
Connect 4 is a "solved" game: there exists a "perfect player" that will 100% of the time if going first. This is ideal, as we are trying to emulate that player through playing the game, without the mathematical mathematical proof or by calculating every possible game.

Data Preparation

We will be using a Convolutional Neural Network to respect the two-dimensions of the board, as well as encode the state in binary – a data format NNs process better.



Network Structure

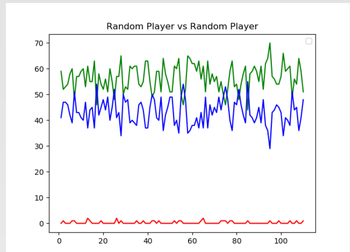


- *We initialize our weights according to a set of pre-played random games, a more effective approach than a completely random initialization.*
- *At the beginning of training, there is a very high chance to "explore", or make a move we do not believe is optimal, in case the true optimal move has not been tried yet. This chance decreases over time.*
- *Hidden Layers:*
  - *Dueling Network: Instead of putting an absolute value (reward) on each move, take into account the value of the move in relation to the current board state.*
  - *Convolutional layer(s): process the board in a 2D state.*
- *Experience Buffer: Take into account the last 3000 games when backpropogating weights.*

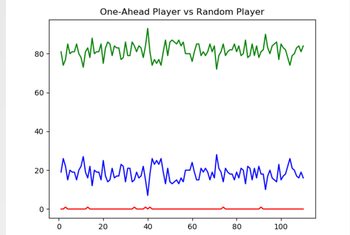
Results

As a baseline, we have simulated a completely random player, and one who looks one move ahead.

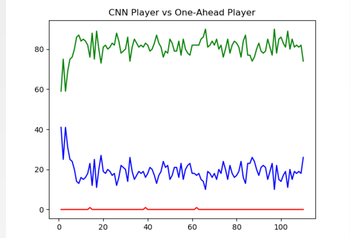
The random player who goes first will have a 54% win rate.



Even after changing colors each game, the "one-ahead" player wins 80% of the time.



We can see our player learning against the "one-ahead" player – to great effect, finishing with 80% win rate!



References

Huge thanks to Carsten Friedrich and his article detailing how he trained a NN to play tic-tac-toe. (link in github readme).

1. *https://en.wikipedia.org/wiki/Reinforcement_learning*
2. *https://en.wikipedia.org/wiki/Convolutional_neural_network*