Bit formatting + alpha beta search:

https://markusthill.github.io/programming/connect-4-introduction-and-tree-search-algorithms/

Board games and AI

https://openreview.net/pdf?id=SklPFVXUsN

ML AGENTS:

Required software (20/07/2020)

- Unity 2018.4 or Later
- Unity MLAgents package
- Python 3.6/3.7 (64 bit)

Creating a Python virtual environment

- Create a folder for your virtual environments

  C:\python-envs

- Create a new virtual environment

  python -m venv C:\python-envs\mlagents-env

Using a the virtual environment

- Activate using

  C:\python-envs\mlagents-env\Scripts\activate

- Deactivate using

  deactivate

Prepare using mlagents

- Install mlagents

  pip3 install mlagents

https://github.com/Unity-Technologies/ml-agents/blob/release_4_docs/docs/Getting-Started.md

https://docs.unity3d.com/Packages/com.unity.ml-agents@1.0/api/Unity.MLAgents.Agent.html#Unity_MLAgents_Agent_CollectObservations_Unity_MLAgents_Sensors_VectorSensor_

https://github.com/gzrjzcx/ML-agents/blob/master/docs/Training-PPO.md

PPO vs SAC

https://blogs.unity3d.com/2019/11/11/training-your-agents-7-times-faster-with-ml-agents/

Self play

https://blogs.unity3d.com/2020/02/28/training-intelligent-adversaries-using-self-play-with-ml-agents/

Improvements

https://www.codeproject.com/Articles/5160398/A-Tic-Tac-Toe-AI-with-Neural-Networks-and-Machine

Symetrical games

Reducing action space by using translations, rotations and mirroring on the original board state so

idea's:

minmax with ml agents evalution

training data

TEST RUNS:

TEST1 AI0 vs AI1: 500.000 matches = dumb AI (no preventing or finishing 4 in a row, no prioritizing middle row)

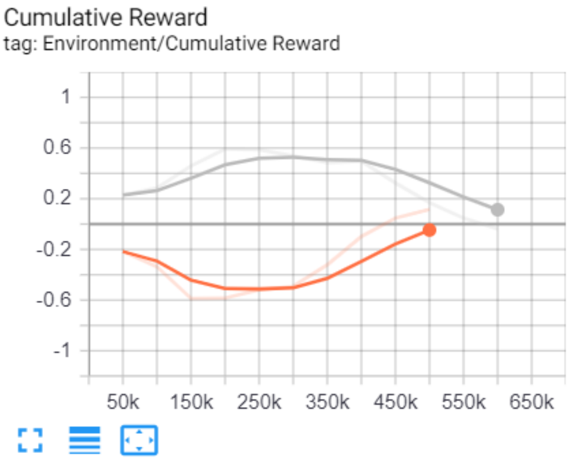
TEST 2 AI vs RND: 500.000 matches = dumb AI (no preventing or finishing 4 in a row, no prioritizing middle row)

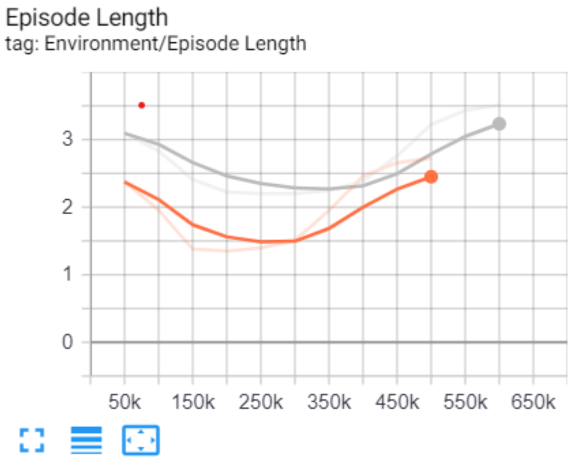
TEST 3 AI0 vs AI1: 4x4 connect3 -> starting player makes vertical 3 in a rows, second player stops them, no further tactics visible (still dumb AI)

TEST 3 AI0 vs AI1: 4x4 connect3 -> starting player makes vertical 3 in a rows, second player stops them, no further tactics visible (still dumb AI)

=>> used basic yaml file which only was one layer deep so obviously the results were bad

Tic Tac Toe:





Behaviour A trained against Behaviour B with Behaviour A always being the starting player.

1000 < Games	Beginner wins	Draw	Beginner loses
Perfect vs Perfect	100.0%	0.0%	0.0%
Random vs Random	58.2%	11.7%	30.2%
Behaviour A vs Behaviour A	97.5%	1.7%	0.8%
Behaviour B vs Behaviour B	80.4%	10.4%	9.2%
Behaviour A vs Random	96.0%	2.2%	1.8%
Behaviour B vs Random	81.0%	10.6%	8.4%
Random vs Behaviour A	21.4%	5.0%	73.6%
Random vs Behaviour B	49.9%	16.5%	33.6%
Behaviour A vs Behaviour B	21.0%	49.4%	29.6%
Behaviour B vs Behaviour A	67.9%	17.1%	14.9%

These are some really weird results, at least it is visible the training actually worked. Some remarks:

- You would expect Behaviour vs Behaviour matches to have only one outcome, but they seem to be creative. As for now we are clueless why this is happening.

- Although Behaviour B was trained by always responding to the moves of Behaviour A it also performs okay in the starting position.

- Behaviour B became an expert in drawing/winning to Behaviour A (79% of the matches) but has more trouble drawing/winning against random moves (50% of the matches) which is clearly a case of overfitting.

- Although these results look good these behaviours still miss finishing/blocking every three in a row. The behaviours do not understand the game of TicTacToe, they just have an incomplete statistical idea of which moves are good. Many previous tests are not reported because the results looked worthless while probably they would have given a winrate just above Random vs Random.

cd C:\Github\BoardGameAI\Four\Assets\ML-Agents

mlagents-learn C:\Github\BoardGameAI\Four\Assets\ML-Agents\Basic.yaml --run-id=XXX

```
tensorboard --logdir=results
```

TENNIS SAC:

behaviors:

  Tennis:

    trainer_type: sac

    hyperparameters:

      batch_size: 128

      buffer_size: 50000

      buffer_init_steps: 0

      init_entcoef: 1.0

      learning_rate: 0.0003

      learning_rate_schedule: constant

      save_replay_buffer: false

      steps_per_update: 10.0

      tau: 0.005

      reward_signal_steps_per_update: 10.0

    network_settings:

      normalize: true

      hidden_units: 256

      num_layers: 2

      vis_encode_type: simple

    reward_signals:

```
      extrinsic:
        gamma: 0.99
        strength: 1.0
    keep_checkpoints: 5
    max_steps: 20000000
    time_horizon: 64
    summary_freq: 10000
    threaded: true
    self_play:
      save_steps: 50000
      team_change: 250000
      swap_steps: 50000
      window: 10
      play_against_latest_model_ratio: 0.5
      initial_elo: 1200.0
```


TENNIS PPO


```
behaviors:
  Tennis:
    trainer_type: ppo
    hyperparameters:
      batch_size: 2048
      buffer_size: 20480
      learning_rate: 0.0003
      beta: 0.005
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: constant
    network_settings:
      normalize: true
```

```yaml
  hidden_units: 256
  num_layers: 2
  vis_encode_type: simple
reward_signals:
  extrinsic:
    gamma: 0.99
    strength: 1.0
keep_checkpoints: 5
max_steps: 50000000
time_horizon: 1000
summary_freq: 10000
threaded: true
self_play:
  save_steps: 50000
  team_change: 100000
  swap_steps: 2000
  window: 10
  play_against_latest_model_ratio: 0.5
  initial_elo: 1200.0
```