COMS 527 Proposal

Justin Stanley

October 15, 2021

1 Summary

For the class project I would like to train a neural network to play the game Connect 4. The model training will be distributed across multiple computers using multithreading and MPI. Nodes with GPUs will use CUDA to accelerate model performance. The model will train with reinforcement learning only through self-play, without access to any human training data. A terminal game position is illustrated on page 2.

Goals

- Final model should consistently win against random moves
- Model training speed should scale with node count (or GPU count)
- Model skill progression should be measurable

Limitations

- Non-GPU nodes may not be able to contribute much to the training process

2 Implementation

2.1 Libraries

The project will be written in C++, using OpenMP for threading and MPI for message passing. Torch will be used for neural network training and inference. The source code will be documented with inline Doxygen.

2.2 Architecture

The agent will use a directed Monte-Carlo tree search to make decisions. The search will be led by policy and value outputs from the neural network. Training data will be built in parallel across all nodes, however once the training data is ready the training step can take place on a single node.

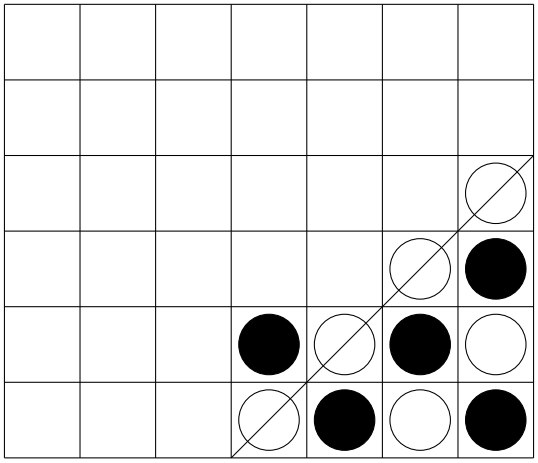

Figure 1: Connect-4 game, winning state for white

2.3 Evaluation

The agent will be evaluated by playing n games against a random policy network (effectively plain MCTS). Let a loss count as 0 points, a draw count as $\frac{1}{2}$ points, and a win count as 1 point. We let s be the total score after n games. The model performance function is then

$$p(s, n) = \frac{s}{n}.$$

The initial model is expected to have a performance rating close to 0.5, but as the model progresses the performance rating should tend towards 1. Performance evaluations will be run every 25 iterations of the network.