Generals Exam: Enhancing AlphaZero to Accomodate a Larger Policy Space and Applications

Zachary Hervieux-Moore

18/05/18

Overview

MDPs

Dynamic Programming

Multi-armed Bandit Problems

UCB

EXP3

MCTS

Bandit Algorithms on MCTS

Dynamic Programming Formulation of MCTS

History of AlphaZero

Overview of AlphaZero

Policy and Value Learning

Reinforcement Through Self Play

Evaluating

Putting it All Together

Mathematical Formulation of AlphaZero

Motivation: AlphaZero and Action Space

Motivation: Example

Motivation: Scrabble

Motivation: Potential Solution

Framework Overview

Framework Progress

Game Class

MCTS Class

Parallel Architecture

Technical Difficulties: Distributive Computing

Talk about GIL in Python

Technical Difficulties: Differences with AlphaZero

Technical Difficulties: Hyperparameters

Proof of Concept: Pawns

Demo

Next Steps: Low Hanging Fruit

- Make code more performant in the MCTS loop
  - AlphaZero 0.4s vs. 4s
  - AlphaZero 800 rollouts vs. 100
- Optimize the parallelization
- Add more complicated games and validate
- Refactor code to make more modular

Next Steps: Compute Cluster

- Write slurm wrapper for the framework
- Need to refactor optimization code to allow splitting of batches across multiple GPUs
- Predict that I will be able to get the framework doing 500 games per second on 1,000 CPUs which matches the production of AlpheZero

Next Steps: ELO Evaluation

- Standard in all of the AlphaZero papers
- Two ways ELO calculation can be done:
  1. Using the formula:
  2. Using techniques that take into account cross-play

Next Steps: Scrabble

- The ultimate goal of the project
- Generating moves presents a much harder challenge than the other games
- Pre existing libraries written in C++

Future Directions: Robotics

Future Directions: Multiplayer Games

Future Directions: SmartDriving Cars

Future Directions: Real World Applications