AlphaZero Algorithm - Implementations

03.05.2021

Vincent Manier

Take-aways from last week:

- Continue to read about implementations of AlphaZero
- Think of computational costs of the AlphaZero algorithm
- Note the points I don't understand to help Jonathan better understand areas he can support

A. Approach

1. Work on a simple implementation of TicTacToe in Python

   See https://github.com/VinBots/MyAlphaZero

2. I played with different parameters to generate a perfect tic-tac-toe player
3. Many improvements / features remain to be implemented:
   a. Training is done after each self-play based on the last self-play experience. I'll obtain better results by using a replay buffer and train based on random batches of experiences with a slowly-growing training window (Oracle part VI) with position deduplication / averaging
   b. Competition between 2 AIs is not integrated in the algorithm - I assume that every iteration generates a better model, which is wrong (the model tends to break if too much training)
   c. Dirichlet distribution to be added
   d. training against z and q (Oracle series idea)

B. Computational Costs of AlphaZero

1. I started reading about sequential and parallel algorithms (https://sites.google.com/view/algorithms-book/home)

2. I run simple experiments with TicTacToe trying to understand time consumed by each part of the algorithm (see alphazero_test jupyter notebook). I kept a very simple neural network on purpose but it would be interesting to understand the costs of inferences during MCTS.

   Total time is mainly a function of:

   - number of self-play (episods)
   - Time per self-play

   In my algorithm, a self-play includes both the simulation of a game and the neural network training.

   Time per self-play is mainly a function of:

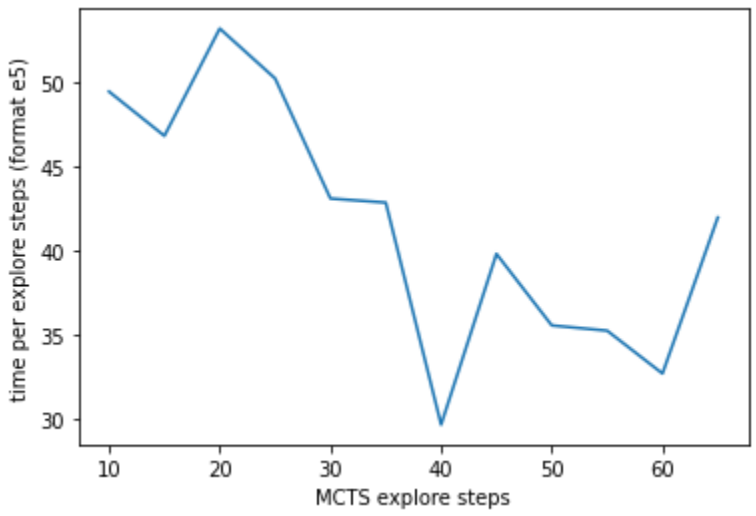
   - time for MCTS exploration per step
   - Average steps during a game
   - Neural network training time

   Time for MCTS exploration per step is mainly a function of :

   - Number of exploration during one MCTS
   - Time for each exploration during one MCTS

3. Results

number of self-play (A)	500
Time per self-play (B)	0.1111s
Total Time (A x B + others)	57s
time for MCTS exploration per step (C)	0.0136
Average steps during a game (D)	8.35
Neural Network training time (E)	0.0028
Time per self-play (B) = C x D + E	0.1111s
Number of exploration during one MCTS	See below
Time for each exploration during one MCTS	Between 30e-5 and 53e-5



C. Areas to explore further

Areas	Paper / Blog	Description
Parallelism	**Alpha Go Zero**	**Implementation of a parallel algorithm**
Asynchronous MCTS	Alpha Go Zero / Surag Nair / Oracle	
Virtual loss	Oracle	Ensures each thread evaluates different nodes (parallel implementation)
Neural network		**Implementation and tuning hyperparameters**
ResNet	Alpha Go Zero	/
Search parameters	Alpha Go Zero	selected by Gaussian process optimisation
L2 weight regularization	Alpha Go Zero	Encourages small weights
Parameter c	Alpha Go Zero	hyperparameter controlling the intensity of L2 penalty to avoid overfitting. How to measure in RL?
Learning rate / Momentum	Alpha Go Zero	Cyclical learning rate
Deployment / Distribution / Optimization	Part III and Part V - Oracle series	Model deployment process? Distributed inference in practice? Inference bucket size? calibration? int8 quantization/inference optimization tools
Implementation	/	Design, code quality, testing and running experiences
Running experiences	/	How to run experiences in a more systematic way?