Renju project

Kozhikhov Alexander, FCS, HSE
Group 162
May, 2018

Annotation.

In this paper I am going to describe my project in general and to focus on what important features were implemented and on the result.

About Renju.

Renju is a two-player board game. It has quite complicated rules, so we played its most simple variant, when a player should only put 5 black or white elements in a row or on the diagonal. The board had 15 rows and 15 columns.

Our goals and approaches.

Our goal was basically to write a program which can play Renju game on a descent level of complexity.

First of all, we studied the AlphaGo paper by DeepMind team. And our program was trying to implement the ideas of the paper.
We started with Supervised learning, and later used the trained model in the Monte-Carlo tree search algorithm. Details are described below.

Details.

We collected data from 2 million renju games and used it in supervised learning. The data was augmented by rotations and shifts of the board in order to learn to play not only in the center of the board, as the overwhelming amount of data had the first step in the exact center of the board. The model was trained to predict steps, done by professional players in any game situation. The model consisted of 8 convolutional layers, separated by rectifier activation functions. It had about 900 hundred trainable parameters.

The Monte-Carlo tree search algorithm was based on the pretrained model. It had 15 seconds time to traverse over the tree, creating new nodes and simulating the game. The main difference between our implementation of MCTS and the classic variant is that it wasn't necessary for our MCTS to finish the game during a simulation. The limit depth was 20 steps from the root. During the propagation in case of win or lose the reward, earned by all the nodes on the way from the root to the finish state was +1 or -1 correspondingly. If the game was not finished in 20 steps the reward was zero, however the new traverse was counted in all the visited nodes. Such approach helped us to increase the number of traverses, as the randomized algorithm requires big amount of simulation to have a positive effect.

Results.

With only supervised model, it still had quite good chances against a human. But with some training it could be easily defeated. However, the Monte-Carlo tree search algorithm improved the power of model, comparing to itself without MCTS and other supervised models. Experiments were held on the number of parameters for the pretrained model, on the effectiveness of hyperparameters in MCTS, such that depth of traversal and special heuristics, which helped to chose best steps.

After that we had a competition, comparing our models to each other. The model had some good draw games and wins but had problems in games versus the best models from the project. The probable reason for that is that the pretrained model had too many trainable parameters, so the computation of each prediction takes a very long time and the MCTS algorithm potential was not fully used.

Nevertheless, the model showed its worthy skills in playing renju, which is definitely a great achievement.

Talking about personal achievements, the project gave a great opportunity to study basic machine learning methods especially Reinforcement learning, to study programming with tensorflow library to make different models learn as well as using the models in the future for the algorithms such as MCTS.