Project 3: Decision Trees for Expert Iteration (Reinforcement learning)

February 7, 2020

1 Description

A combination of Neural Networks (NNs) and Monte Carlo Tree Search (MCTS) has resulted in really strong players for a multitude of games.

The basic idea is to create a learning loop with the following steps:

1. Multiple games are played using a standard search algorithm like MCTS.
2. The actions that the player took at each state are kept and used to train a neural network.
3. The neural network is used to enhance/guide the search algorithm.

For this project we are going to implement a simpler version of this algorithm (no neural network included!) using decision trees as the classifier.

2 Tasks

1. Using the code provided here, generate (and collect) sample data for at least one kind of game (e.g., OXO, Othello). Sample data that includes combinations of state values (keep your encoding as close to the game mechanics as possible) and the action that the agent took. For example, for OXOState/Game, the state is the board and who is playing; the action is the action the MCTS/UCT took.
2. Use a machine learning approach (e.g., a decision tree) to learn a fast classifier that is able to predict which action to take, by using supervised learning on the MCTS/UCT actions.
3. (OPTIONAL) If you want to improve performance further, instead of feeding the above classifier with data from the the board directly, use a neural network to learn to predict the next state given an action (this is termed the inverse model) and use the middle layer of the neural network as input to the decision tree. — To do this you might have to modify the way you collect your data.
4. Modify the following part of the MCTS function in the code provided so that, instead of doing a purely random search, it uses the learned classifier and finds out which action to take. Do the best action 90% of the time, while playing randomly 10%:

```
# Rollout - this can often be made quicker using a state.GetRandomMove() function
while state.GetMoves() != []: # while state is non-terminal
    state.DoMove(random.choice(state.GetMoves()))
```

5. Collect the data using this approach. You now have a new set of states and actions — repeat the above procedure for as many iterations as possible.
6. Every 10 iterations of the whole algorithm, have an agent play with all its past self 10 games and record the results — are the agents improving?

3 References

1. [Anthony, Thomas, Zheng Tian, and David Barber. "Thinking fast and slow with deep learning and tree search." In Advances in Neural Information Processing Systems, pp. 5360-5370. 2017.

2. Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert et al. "Mastering the game of Go without human knowledge." Nature 550, no. 7676 (2017): 354.