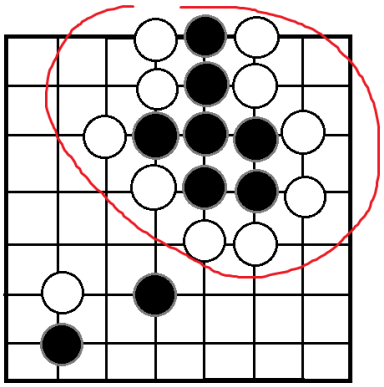
Reinforcement Learning Project Proposal Severian Adams: Capture Go

---

I would like to apply existing RL algorithms to a new problem. The problem I would like to approach is designing a reinforcement learning algorithm that can learn a good strategy for the game of **Capture Go** [1].



In this example, black's group has no more liberties, in other words no empty intersections adjacent to its group. This results in a capture, and in Capture Go this would result in a victory for white.

Capture Go is similar to the game of Go, if you are familiar with it. In Capture Go, two players take turns placing different colored stones on the intersections of a square grid. Where Capture Go is different from Go however, is that in Capture Go the first player to capture any stone or group of stones wins the game. A stone is captured just like in Go, when its group has no more liberties (a liberty in Go is an empty intersection adjacent to a group of stones).

I chose to model Capture Go instead of standard Go for several reasons.

1. Capture Go is a bit simpler than Go, especially from a design standpoint. In Capture Go I would not need to design algorithms for complicated things, especially calculating territory, which even modern algorithms can have trouble with.
2. Unlike standard Go, Capture Go is guaranteed to end in a finite number of moves (on a finite board space).
3. Capture Go games tend to finish a lot quicker, which means training a reinforcement learning algorithm on it will be a lot faster.
4. The goal of Capture Go is a lot more direct than standard Go.
5. Despite being simpler than standard Go, Capture Go is still somewhat complicated and interesting, which I think will make for a good reinforcement learning exercise.

Because of the sheer size of the state space, even on a very small board, our tabular methods will be incredibly impractical for designing an algorithm to play Capture Go, which means I will almost certainly use a function approximation reinforcement learning algorithm.

It is my hope that I can train the algorithm on a smaller board size, such as 9x9, 7x7, or even 5x5, and then use that same algorithm on much larger board sizes. That way an algorithm could be trained quickly, but still be able to apply its learned strategies to play on larger boards that it has not seen before. When teaching new players Go, it is actually common to use smaller board sizes and start with capture go, which is what we are modeling here.

I also plan to have the agent learn by continually playing games against itself. Not only does this mean that I don't have to source some other Go algorithm, since the algorithm is playing both sides it has double the opportunities to learn good moves. This is one of the methods that Google's AlphaGo reinforcement algorithm used [2], although I will not be introducing my algorithm to example games like Google did for AlphaGo.

References

[1] https://www.usgo-archive.org/sites/default/files/pdf/rules_trifold.pdf
[2] https://deepmind.google/research/projects/alphago/