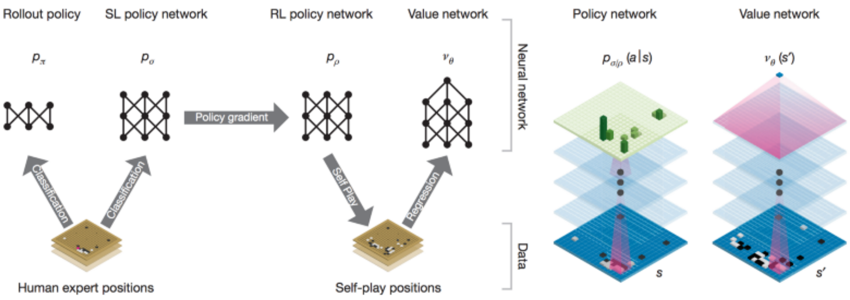
Andrej Karpathy Follow
Research Scientist at OpenAI. Previously CS PhD student at Stanford. I like to train Deep Neural Nets on...
May 31 · 5 min read

AlphaGo, in context

I had a chance to talk to several people about the recent AlphaGo matches with Ke Jie and others. In particular, most of the coverage was a mix of popular science + PR so the most common questions I've seen were along the lines of "to what extent is AlphaGo a breakthrough?", "How do researchers in AI see its victories?" and "what implications do the wins have?". I thought I might as well serialize some of my thoughts into a post.

The cool parts

AlphaGo is made up of a number of relatively standard techniques: behavior cloning (supervised learning on human demonstration data), reinforcement learning (REINFORCE), value functions, and Monte Carlo Tree Search (MCTS). However, the way these components are combined is novel and not exactly standard. In particular, AlphaGo uses a SL (supervised learning) policy to initialize the learning of an RL (reinforcement learning) policy that gets perfected with self-play, which they then estimate a value function from, which then plugs into MCTS that (somewhat surprisingly) uses the (worse!, but more diverse) SL policy to sample rollouts. In addition, the policy/value nets are deep neural networks, so getting everything to work properly presents its own unique challenges (e.g. value function is trained in a tricky way to prevent overfitting). On all of these aspects, DeepMind has executed very well. That being said, AlphaGo does not by itself use any fundamental algorithmic *breakthroughs* in how we approach RL problems.



On narrowness

Zooming out, it is also still the case that AlphaGo is a narrow AI system that can play Go and that's it. The ATARI-playing agents from DeepMind do not use the approach taken with AlphaGo. The Neural Turing Machine has little to do with AlphaGo. The Google datacenter improvements definitely do not use AlphaGo. The Google Search engine is not going to use AlphaGo. Therefore, AlphaGo does not generalize to any problem outside of Go, but the people and the underlying neural network components do, and do so much more effectively than in the days of old AI where each demonstration needed repositories of specialized, explicit code.

Convenient properties of Go

I wanted to expand on the narrowness of AlphaGo by explicitly trying to list some of the specific properties that Go has, which AlphaGo benefits a lot from. This can help us think about what settings AlphaGo does or does not generalize to. Go is:

1. fully **deterministic.** There is no noise in the rules of the game; if the two players take the same sequence of actions, the states along the way will always be the same.

2. fully **observed.** Each player has complete information and there are no hidden variables. For example, Texas hold'em does not satisfy this property because you cannot see the cards of the other player.

3. the action space is **discrete.** a number of unique moves are available. In contrast, in robotics you might want to instead emit continuous-valued torques at each joint.

4. we have access to a perfect **simulator** (the game itself), so the effects of any action are known exactly. This is a strong

assumption that AlphaGo relies on quite strongly, but is also quite rare in other real-world problems.

5. each episode/game is relatively **short,** of approximately 200 actions. This is a relatively short time horizon compared to other RL settings which may involve thousands (or more) of actions per episode.

6. the **evaluation** is clear, fast and allows a lot of **trial-and-error** experience. In other words, the agent can experience winning/losing millions of times, which allows is to learn, slowly but surely, as is common with deep neural network optimization.

7. there are huge **datasets of human play** game data available to bootstrap the learning, so AlphaGo doesn't have to start from scratch.

Example: AlphaGo applied to robotics?

© Geoff Robinson

Having enumerated some of the appealing properties of Go, let's look at a robotics problem and see how well we could apply AlphaGo to, for example, an Amazon Picking Challenge robot. It's a little comical to even think about.

- First, your (high-dimensional, continuous) actions are awkwardly /noisily executed by the robot's motors (1,3 are violated).
- The robot might have to look around for the items that are to be moved, so it doesn't always sense all the relevant information and has to sometimes collect it on demand. (2 is violated)
- We might have a physics simulator, but these are quite imperfect (especially for simulating things like contact forces); this brings its own set of non-trivial challenges (4 is violated).
- Depending on how abstract your action space is (raw torques -> positions of the gripper), a successful episode can be much longer than 200 actions (i.e. 5 depends on the setting). Longer

episodes add to the *credit assignment problem*, where it is difficult for the learning algorithm to distribute blame among the actions for any outcome.

- It would be much harder for a robot to practice (succeed/fail) at something millions of times, because we're operating in the real world. One approach might be to parallelize robots, but that can be quite expensive. Also, a robot failing might involve the robot actually damaging itself. Another approach would be to use a simulator and then transfer to the real world, but this brings its own set of new, non-trivial challenges in the domain transfer. (i.e. 6 is violated).
- Finally, there is rarely a human data source with millions of demonstrations (so 7 is violated).

In short, basically every single assumption that Go satisfies and that AlphaGo takes advantage of are violated, and any successful approach would look extremely different. More generally, some of Go's properties above are not insurmountable with current algorithms (e.g. 1,2,3), some are somewhat problematic (5,7), but some are quite critical to how AlphaGo is trained but are rarely present in other real-world applications (4,6).

In conclusion

While AlphaGo does not introduce fundamental breakthroughs in AI algorithmically, and while it is still an example of narrow AI, AlphaGo does symbolize Alphabet's AI power: in both the quantity/quality of the talent present in the company, the computational resources at their disposal, and the all in focus on AI from the very top.

Alphabet is making a large bet on AI, and it is a safe one. But I'm biased :)

EDIT: the goal of this post is, as someone on reddit mentioned, "quelling the ever resilient beliefs of the public that AGI is right down the road", and the target audience are people outside of AI who were watching AlphaGo and would like a more technical commentary.