Simple Reinforcement Learning with Tensorflow Part 0: Q-Learning with Tables and Neural Networks

Arthur Juliani Follow
Aug 25, 2016 · 6 min read

We'll be learning how to solve the OpenAI FrozenLake environment. Our version is a little less photo-realistic.

For this tutorial in my Reinforcement Learning series, we are going to be exploring a family of RL algorithms called Q-Learning algorithms. These are a little different than the policy-based algorithms that will be looked at in the the following tutorials (Parts 1–3). Instead of starting with a complex and unwieldy deep neural network, we will begin by implementing a simple lookup-table version of the algorithm, and then show how to implement a neural-network equivalent using Tensorflow. Given that we are going back to basics, it may be best to think of this as Part-0 of the series. It will hopefully give an intuition into what is really happening in Q-Learning that we can then build on going forward when we eventually combine the policy gradient and Q-learning approaches to build state-of-the-art RL agents (If you are more interested in Policy Networks, or already have a grasp on Q-Learning, feel free to start the tutorial series here instead).

Unlike policy gradient methods, which attempt to learn functions which directly map an observation to an action, Q-Learning attempts to learn the value of being in a given state, and taking a specific action there. While both approaches ultimately allow us to take intelligent actions given a situation, the means of getting to that action differ significantly. You may have heard about DeepQ-Networks which can play Atari Games. These are really just larger and more complex implementations of the Q-Learning algorithm we are going to discuss here.

Top highlight

Tabular Approaches for Tabular Environments

```
SFFF       (S: starting point, safe)
FHFH       (F: frozen surface, safe)
FFFH       (H: hole, fall to your doom)
```

If this post has been valuable to you, please consider *donating* to help support future tutorials, articles, and implementations. Any contribution is greatly appreciated!

If you'd like to follow my work on Deep Learning, AI, and Cognitive Science, follow me on Medium @Arthur Juliani, or on Twitter @awjliani.

. . .

More from my Simple Reinforcement Learning with Tensorflow series:

1. ***Part 0 — Q-Learning Agents***
2. *Part 1 — Two-Armed Bandit*
3. *Part 1.5 — Contextual Bandits*
4. *Part 2 — Policy-Based Agents*
5. *Part 3 — Model-Based RL*
6. *Part 4 — Deep Q-Networks and Beyond*
7. *Part 5 — Visualizing an Agent's Thoughts and Actions*
8. *Part 6 — Partial Observability and Deep Recurrent Q-Networks*
9. *Part 7 — Action-Selection Strategies for Exploration*
10. *Part 8 — Asynchronous Actor-Critic Agents (A3C)*

Machine Learning | Artificial Intelligence | Neural Networks | Deep Learning | Robotics