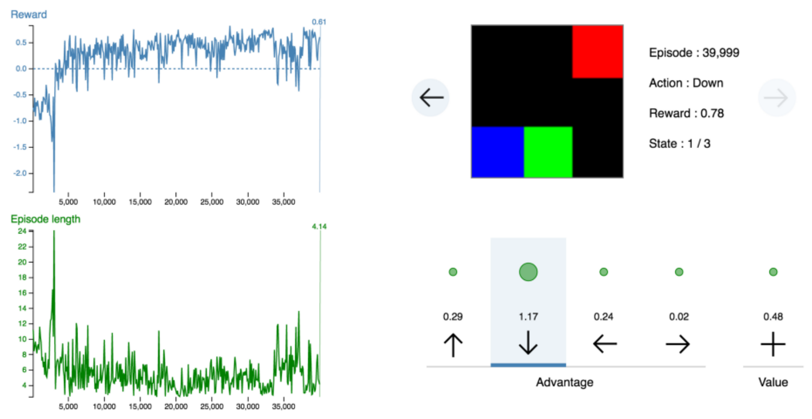
Arthur Juliani Follow
Deep Learning @Unity3D & Cognitive Neuroscience PhD student.
Sep 10, 2016 · 6 min read

Simple Reinforcement Learning with Tensorflow Part 5: Visualizing an Agent's Thoughts and Actions

In this post of my Reinforcement Learning series, I want to further explore the kinds of representations that our neural agent learns during the training process. While getting a high score, or accomplishing a specified task is what we often want our neural agents to be capable of, it is just as important to understand ***how,*** and even more critically, ***why*** that agent is behaving in a certain way. In order to make the learning process a little more transparent, I built a d3.js powered web interface that presents various kinds of information about our agent as it learns. I call it the *Reinforcement Learning Control Center.* In this post I will use it to provide a little more insight into the how and why of an RL agent.

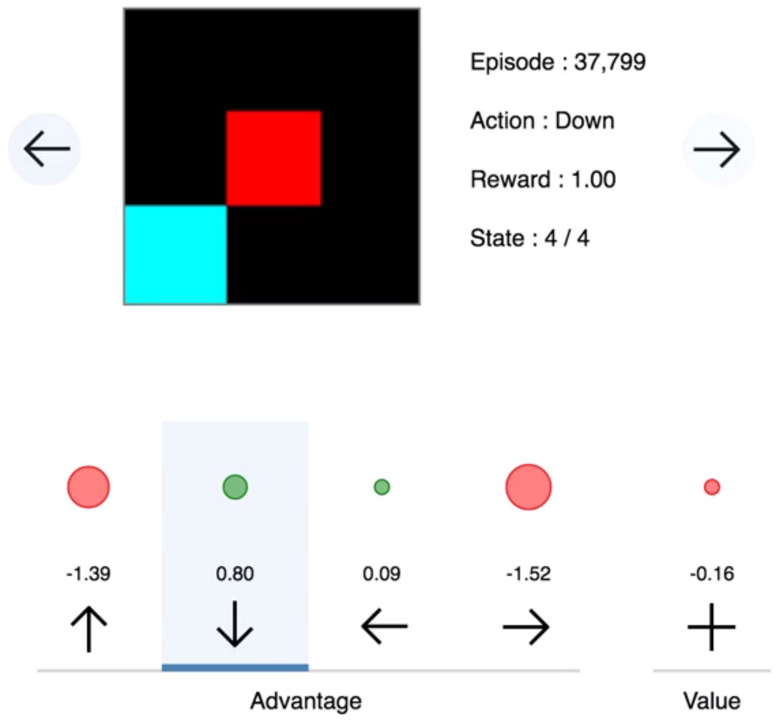
The Control Center Interface



The Reinforcement Learning Control Center interface.

The Control Center was designed with the purpose of allowing the user to track the performance of an agent in realtime as it learns to perform a task. On the left of the interface, The episode length and reward over time are tracked and updated dynamically. The right displays an animated gif of a sample training episode, along with the advantage and value functions being computed by the agent at every step of the episode.

The interface is currently specific to the neural network and task that I described in my last post. It is a Double-Dueling-DQN, and the environment is a simple gridworld. The rules of that gridworld are as follows: the agent controls the blue square and can move either up, down, left, or right. The goal is to get to the green square as quickly as possible, while avoiding the red square. The green square provides +1 reward, the red square -1 reward, and each step is -0.1 reward. At the beginning of each episode, the three squares are randomly placed on the grid.



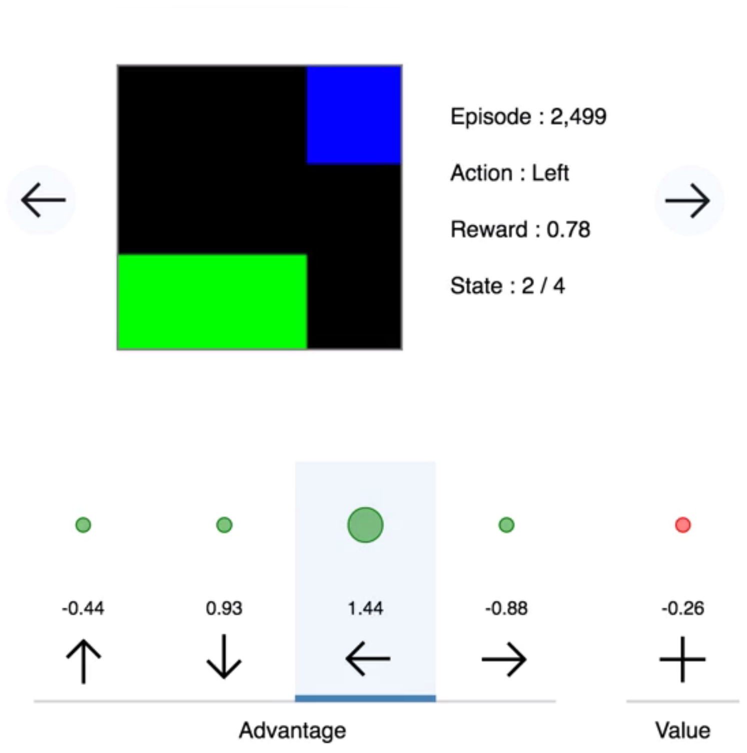
Display of a neural agent's actions and thoughts during an episode of training.

The DD-DQN neural agent processes two separate streams of information as it experiences the gridworld: an advantage stream and a value stream. The advantage stream represent how good the network thinks it is to take each action, given the current state it is in. The value stream represents how good the network thinks it is to be in a given state, regardless of possible action. With the Control Center, we can watch as the network learns to correctly predict the value of its state and actions over time. As the training process proceeds, it goes from seemingly random values to accurately interpreting certain actions as the most advantageous. We can think of this visualization as providing a portal into the "thought process" of our agent. Does it know that it is in a good position when it is in a good position? Does it know that going down was a good thing to do when it went down? This can give us the insights needed to understand why our agent might not be performing ideally as we train it under different circumstances in different environments.

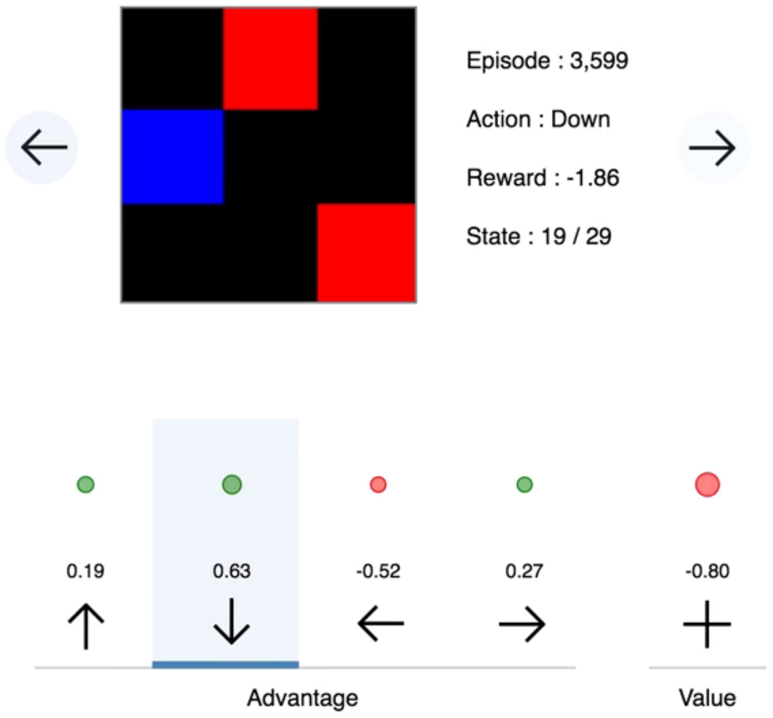
Getting Inside Our Agent's Head

Not only can we use the interface to explore how the agent does during training, we can also use it for testing and debugging our fully trained agents. For example, after training our agent to solve the simple 3x3 gridworld described above, we can provide it with some special test scenarios it had never encountered during the training process to evaluate whether it really is representing experience as we would expect it to.

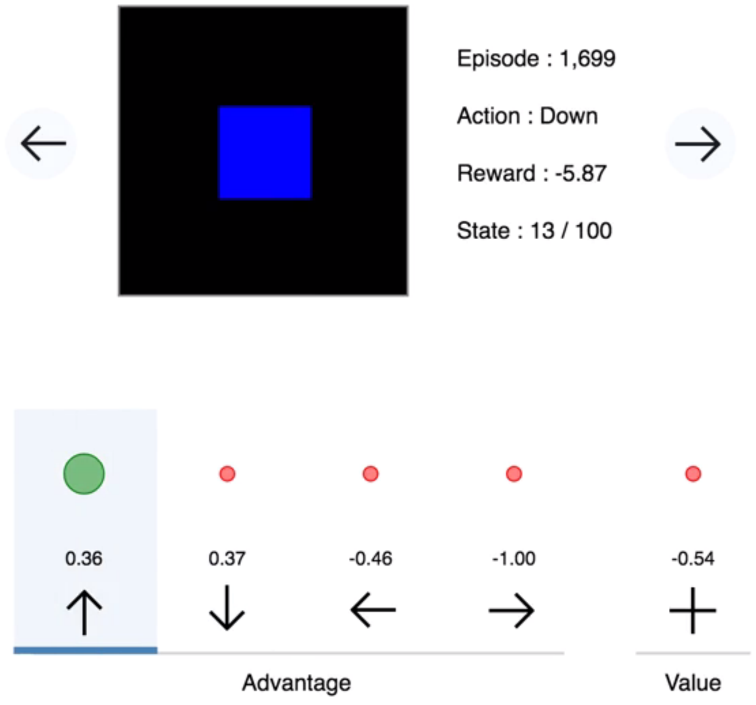
Below is an example of the agent performing a modified version of the task with only green squares. As you can see, as the agent gets closer to the green squares the value estimate increases just as we would expect. It also has high estimates of the advantage for taking actions that get it closer to the green goals.



For the next test we can invert the situation, giving the agent a world in which there were only two red squares. It didn’t like this very much. As you can see below, the agent attempts to stay away from either square, resulting in behavior where it goes back and forth for a long period of time. Notice how the value estimate decreases as the agent approaches the red squares.



Finally, I provided the agent with a bit of an existential challenge. Instead of augmenting the kind of goals present, I removed them all. In this scenario, the blue square is by itself in the environment, with no other objects. Without a goal to move towards, the agent moves around seemingly at random, and the value estimates are likewise seemingly meaningless. What would Camus say?



Taken together, these three experiments provide us with evidence that our agent is indeed responding to the environment as we would intuitively expect. These kinds of checks are essential to make when designing any reinforcement learning agent. If we aren't careful about the expectations we built into the agent itself and the reward structure of the environment, we can easily end up with situations where the agent doesn't properly learn the task, or at least doesn't learn it as we'd expect. In the gridworld for example, taking a step results in a -0.1 reward. This wasn't always the case though. Originally there was no such penalty, and the agent would learn to move to the green square, but do so after an average of about 50 steps! It had no "reason" to hurry, to the goal, so it didn't. By penalizing each step even a small amount, the agent is able to quickly learn the intuitive behavior of moving directly to the green goal. This reminds us of just how subconscious our own reward structures as humans often are. While we may explicitly only think of the green as being rewarding and the red as being punishing, we are subconsciously constraining our actions by a desire to finish quickly. When designing RL agents, we need ensure that we are making their reward structures as rich as ours.

Using the Control Center

If you want to play with a working version of the Control Center without training an agent yourself, just follow this link (currently requires Google Chrome). The agent's performance you will see was pretrained on the gridworld task for 40,000 episodes. You can click the timeline on the left to look at an example episode from any point in training. The earlier episodes clearly show the agent failing to properly interpret the task, but by the end of training the agent almost always goes straight to the goal.

The Control Center is a piece of software I plan to continue to develop as I work more with various Reinforcement Learning algorithms. It is currently hard-coded to certain specifics of the gridworld and DD-DQN described in Part 4, but if you are interested in using the interface for your own projects, feel free to fork it on Github, and adjust/adapt it to your particular needs as you see fit. Hopefully it can provide new insights into the internal life of your learning algorithms too!

. . .

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*