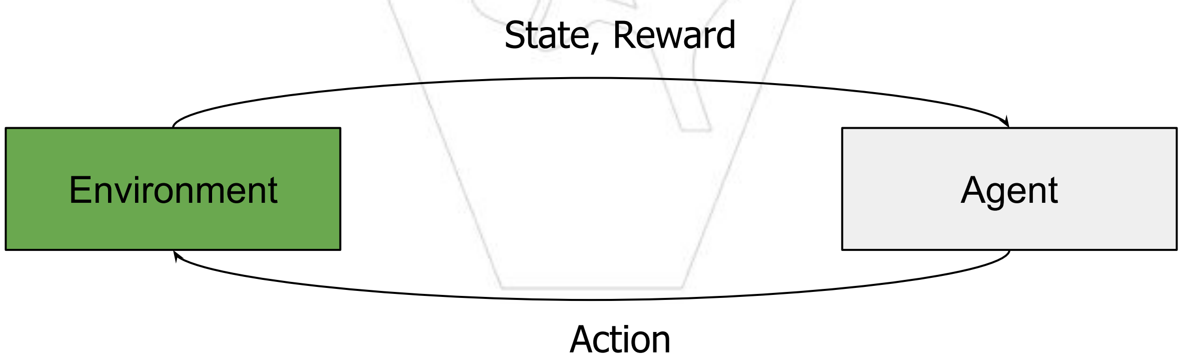
Reinforcement Learning
and Related Topics

Goal: Present a brief overview of reinforcement learning and related topics along with resources to make it easier to dig deeper.

Reinforcement Learning

Basics

- There is an agent interacting in an environment and trying to maximize the reward received from the environment



Terminology

- *State* is the numerical representation of the current state of the environment.
- An *episode* is one sequence of interactions (ie, trajectory) until termination. In games, this would mean playing until game over.
- *Reward* is a scalar returned after each interaction.
- *Return* is the cumulative reward over the course of a single episode
- An *agent* is the entity trying to maximize return.
- A *policy* defines the behavior for an agent.
- *State space* defines all possible values for the state.
- *Action space* defines all possible values for actions.

Markov Decision Process

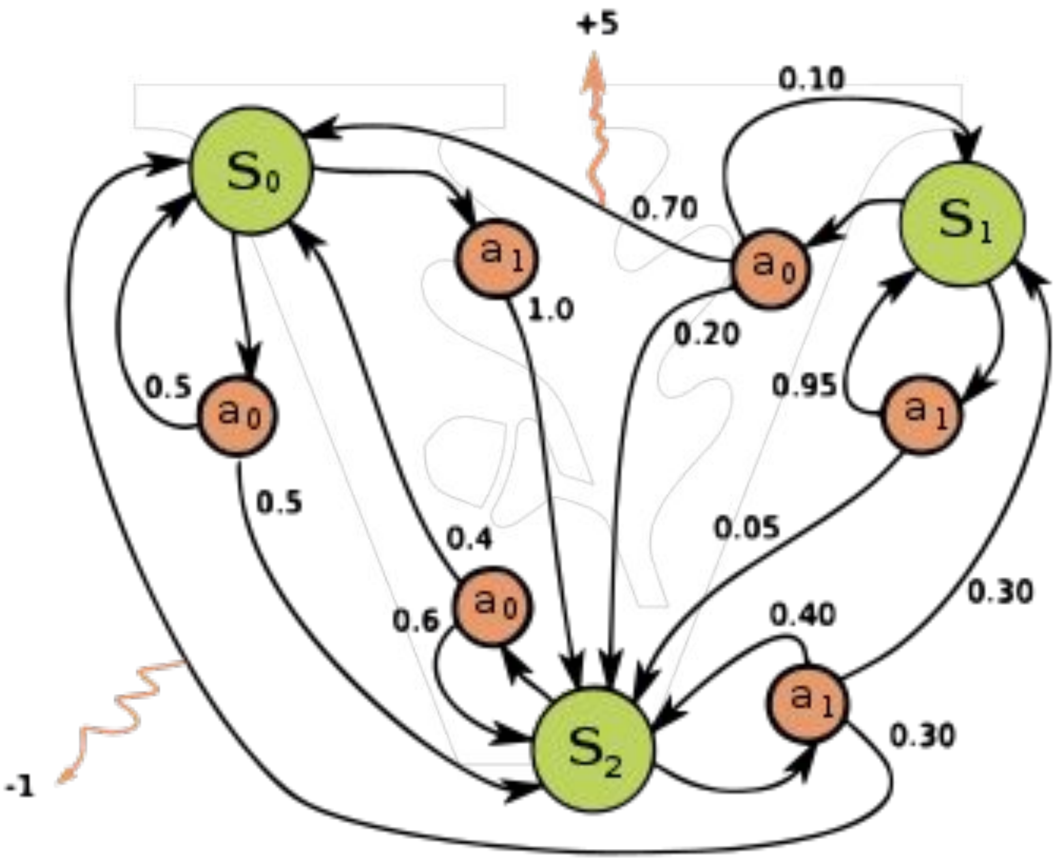


Image from https://en.wikipedia.org/wiki/Markov_decision_process

Value and Q Functions

- Q function: Given a state and action, return the expected return.
- Value function: Given a state, return the expected reward from the state.
- More Info...

$$Q(s_t, a_t) = R$$

$$V(s_t) = R$$

A Few Algorithms

Before starting the example...

- OpenAI Gym has basically become the standard interface for RL environments.
- It makes environments interchangeable as they each define:
  - A state space
  - An action space
  - Reward function
  - Methods for applying actions and resetting the environment

CartPole in PyTorch

Related Topics

- What if interactions with the environment are expensive? Could we just learn the transition dynamics? (ie, replace the environment with a neural network)
  - Yep, this is ***model-based reinforcement learning***
- **Deterministic vs stochastic environments**. If the environment is deterministic, you *could* memorize the best action sequence and simply replay it each time.
- What if the optimal policy is no longer optimal? This could mean that the environment is **non-stationary** making the problem much harder!
- **Exploration vs exploitation trade-off**. Some algorithms use approaches like epsilon-greedy exploration; others have exploration built into the algorithm.
- **On-policy vs off-policy.** Do we need data collected from our policy to improve (on-policy)? Or could we use data collected from other policies/agents/users (off-policy)?

More Related Topics

- What if actions in my environment do not impact subsequent decisions?
  - Then a **multi-armed bandit** is more suited for your task!
- What if I have human demonstrations available? Can I use them to help with the RL problem?
  - Yeah, you should be able to “warm start” the model using imitation learning initially then switch to reinforcement learning.
- What if I have a policy which is non-differentiable? Can I still optimize it?
  - There are optimization techniques such as CMAES which can be used to perform direct policy search.
- What if I want the policy to learn to be human-like?
  - You are probably looking for **imitation learning**

Demos and Resources

- DQN Tutorial in PyTorch
- Spinning Up in Deep RL
- Bandit Algorithms Book
- RLlib (framework with many implemented algorithms) (paper)
- Imitation Learning using Case-based Reasoning in NetsBlox