Deep Reinforcement Learning Tutorials

Wilbert Santos Pumacay Huallpa

August 10, 2019

Outline for today

- Policy Evaluation and Control using Dynamic Programming (DP)
  - Bellman Equations
  - Policy Evaluation
  - Policy Improvement
  - Policy Iteration and Value Iteration
- Solving Gridworlds using Policy Iteration and Value Iteration.

Recap

Recap

- We defined a **Policy** π, as a mapping between states to actions. Our agents use policies to act in the environment.

$$\pi : \mathbb{S} \rightarrow \mathbb{A}; \text{ Deterministic policy}$$

$$\pi : \mathbb{S} \times \mathbb{A} \rightarrow [0, 1]; \text{ Stochastic policy}$$

Recap

- We defined a **Policy** π, as a mapping between states to actions. Our agents use policies to act in the environment.

$$\pi : \mathbb{S} \rightarrow \mathbb{A}; \text{ Deterministic policy}$$

$$\pi : \mathbb{S} \times \mathbb{A} \rightarrow [0, 1]; \text{ Stochastic policy}$$

- We defined the **State-value function** $V^{\pi}(s)$ as the expected return we would get from a given state s by following policy π.

$$V(s) = \mathbb{E}_{\pi}\{\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_t = s\}$$

Recap

- We defined a **Policy** π, as a mapping between states to actions. Our agents use policies to act in the environment.

$$\pi : \mathbb{S} \rightarrow \mathbb{A}; \text{ Deterministic policy}$$

$$\pi : \mathbb{S} \times \mathbb{A} \rightarrow [0,1]; \text{ Stochastic policy}$$

- We defined the **State-value function** $V^{\pi}(s)$ as the expected return we would get from a given state s by following policy π.

$$V(s) = \mathbb{E}_{\pi}\{\sum_{t=0}^{\infty} \gamma^{t} r_{t+1} | s_t = s\}$$

- We defined the **Action-value function** $Q^{\pi}(s,a)$ as the expected return we would get by starting at state s, taking a given action a (not necessarily from the policy π), and then following π.

$$Q(s,a) = \mathbb{E}_{\pi}\{\sum_{t=0}^{\infty} \gamma^{t} r_{t+1} | s_t = s, a_t = a\}$$

Recap

- We also defined the objective of our agent as finding a policy π^* that maximizes the expected return over all possible policies. We called it an **optimal policy**.

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\{\sum_{t=0}^{\infty} \gamma^t r_{t+1}\}$$

Recap

- We also defined the objective of our agent as finding a policy π^* that maximizes the expected return over all possible policies. We called it an **optimal policy**.

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\{\sum_{t=0}^{\infty} \gamma^t r_{t+1}\}$$

- Related to this policy we also defined the corresponding optimal versions of both State-value and Action-value functions as $V^*(s)$ and $Q^*(s, a)$.

$$V^* = V^{\pi^*}, V^*(s) \geq V^{\pi} \;\forall\, s, \pi$$
$$Q^* = Q^{\pi^*}, Q^*(s, a) \geq Q^{\pi}(s, a) \;\forall\, s, a, \pi$$

Recap

- We also defined the objective of our agent as finding a policy π^* that maximizes the expected return over all possible policies. We called it an **optimal policy**.

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\{\sum_{t=0}^{\infty} \gamma^t r_{t+1}\}$$

- Related to this policy we also defined the corresponding optimal versions of both State-value and Action-value functions as $V^*(s)$ and $Q^*(s, a)$.

$$V^* = V^{\pi^*}, V^*(s) \geq V^{\pi} \; \forall \, s, \pi$$
$$Q^* = Q^{\pi^*}, Q^*(s, a) \geq Q^{\pi}(s, a) \; \forall \, s, a, \pi$$

- So, today we will discuss a first set of methods for finding these functions using **Dynamic Programming**.

Dynamic Programming

You might want to check these resources

Below there are some lectures and resources that I think you should check in more depth after this tutorial. I'll try to give the required high-level overview such that you will feel comfortable with these awesome resources.

- David Silver - lecture 3 on Dynamic Programming
- Hado Van Hasselt - lecture 3 on Dynamic Programming
- Sutton and Barto RL book - chapter 4

Dynamic Programming (DP)

Dynamic Programming (DP)

- Fibonacci?, Floyd Warshall?, Top Coder?, ... and various other problems can be solved using this approach.

Dynamic Programming (DP)

- Fibonacci?, Floyd Warshall?, Top Coder?, ... and various other problems can be solved using this approach.
- Recall that in those the key component was memoization and recursive formulation.

Dynamic Programming (DP)

- Fibonacci?, Floyd Warshall?, Top Coder?, ... and various other problems can be solved using this approach.
- Recall that in those the key component was memoization and recursive formulation.
- These two properties will appear in various problems, and can be defined as follows:

Dynamic Programming (DP)

- Fibonacci?, Floyd Warshall?, Top Coder?, ... and various other problems can be solved using this approach.
- Recall that in those the key component was memoization and recursive formulation.
- These two properties will appear in various problems, and can be defined as follows:
  - Principle of Optimality (**That recursive formulation**): If a problem presents this property, then the optimal solution can be decomposed solutions to subproblems in a recursive way.

Dynamic Programming (DP)

- Fibonacci?, Floyd Warshall?, Top Coder?, ... and various other problems can be solved using this approach.
- Recall that in those the key component was memoization and recursive formulation.
- These two properties will appear in various problems, and can be defined as follows:
  - Principle of Optimality (**That recursive formulation**): If a problem presents this property, then the optimal solution can be decomposed solutions to subproblems in a recursive way.
  - Overlapping subproblems (**That memoization technique**): The solutions to the subproblems can be cached and reused.