Introduction to Reinforcement Learning

A primer on the concepts and algorithms

Krishnan Srinivasan

March 2, 2017

Yale University

Table of contents

Introduction

Reinforcement Learning

Reinforcement learning's popularity has spiked as a result of AlphaGo, which used a combination of Deep RL, CNNs, and Monte Carlo Search to beat the world's best Go player, Lee Sedol. Since then, it has evolved to become an even better player after competing in online Go competitions.

Reinforcement Learning: Definitions

- **Reinforcement learning problem:** An **agent** must learn to choose actions that allow it to transition from state to state in an environment, in order to get more reward.
- **Agent:** A Go-playing/helicopter-flying/etc. program
- **Environment:** A set of states and transitions (modeled as an MDP)
- **Policy:** $\pi : \mathcal{S} \to \mathcal{A}$
- **Total Discounted Reward:**

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- **Value:**
  1. action-value function $\quad q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
  2. state-value function $\quad v_\pi : \mathcal{S} \to \mathbb{R}$

Deep Reinforcement Learning

The main goal of reinforcement learning is to learn some optimal state-value/ action-value function and (as a consequence) an optimal policy. One of the key advancements in RL has been the use of neural networks as function approximators to evaluate states (first done by Tesauro in TD-Gammon).

What is Deep Reinforcement Learning?

Using neural networks to approximate

- value functions
- policy functions
- dynamic models (predicting future states/rewards)

Helicopter Apprenticeship Learning

Example of apprenticeship learning:

https://www.youtube.com/watch?v=M-QUkgk3HyE

Temporal Difference Learning

Temporal Difference Learning

The basis of the Q-Learning algorithm comes from the concept of **Temporal Difference Learning**.

Types of policy learning strategies:

- Off-policy Learning: Evaluate a target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$. Learning by observation (with a guiding target policy). Apprenticeship learning is an example of this.
- On-policy Learning: Learn policy π from experience gained from sampling π.

Temporal Difference Learning

TD Learning is the act of updating the values of your states while enacting the policy, to have a lower variance estimate of the actual value of a state. This is an example of "online" learning.
An alternative to TD learning is known as **MC Learning**, which requires waiting until the end of an episode (policy evaluation until the terminal state), before updating the value estimates.

Temporal Difference Learning

Generic TD learning update Step:

$$v'(s_t) \leftarrow v(s_t) + \alpha[R_{t+1} + \gamma v(s_{t+1}) - v(s_t)]$$

SARSA and Q-Learning Algorithms

SARSA

SARSA stands for State, Action, Reward, State, Action, and comes from the parameters used in the SARSA algorithm. Both SARSA and Q-Learning use a temporal-difference prediction, which updates the value of a current state using an estimate of the next state's value plus the reward at the next state.

SARSA

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Q-Learning

The algorithm for **Q-Learning** is almost identical to the SARSA algorithm shown in the previous slide, with the exception of the way the next state is chosen. In SARSA, the estimated value of a sampled state action pair for the next state is chosen to calculate the value of the current state.

In Q-Learning, the next state always results from choosing the action that maximizes the Q function, known as the action-value function.

Q-Learning is an example of an off-policy approach towards learning the optimal action-value function.

Q-Learning

- At each time step t, the action to change states is sampled from the current policy $A_{t+1} \sim \pi_t$.
- To measure the action-value of the current state-action pair (S_t, A_t), the temporal difference update in Q-learning samples a different action A' which is the action that maximizes the value of the next state S_{t+1}.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \arg\max_{A'} \gamma Q(S_{t+1}, A') - Q(S_t, A_t))$$

Q-Learning

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Repeat (for each step of episode):
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ε-greedy)
        $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

Code

Code Tutorial

1. Setup Python (install Homebrew if on MacOS)
2. **Juypter Notebooks**
3. **OpenAI Gym**

Questions?