COMP6247: Reinforcement and Online Learning

Mahesan Niranjan

School of Electronics and Computer Science
University of Southampton

Temporal Difference Methods
Sampled from Chapter Six, Sutton and Barto

Spring Semester 2020/21

Overview

- We have seen dynamic programming (given knowledge of environment find optimal policy)
- We can learn about the environment by Monte Carlo simulations
- We now see more practical algorithms
  - Q-Learning
  - SARSA
- The algorithms we learn will be combinations of ideas from Dynamic Programming and Monte Carlo methods:
  - Incremental improvements towards optimal returns
  - Stochasticity in search and sample averages
- Tabular (discrete) formulations now, function approximations later

MC & TD

- Monte Carlo Methods update after running an episode to finish and observing return G_t:

$$V(S_t) \leftarrow V(S_t) + \alpha \left[G_t - V(S_t) \right]$$

- TD Methods use observed reward: $V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right]$

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\textit{terminal}) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\left[R + \gamma V(S') - V(S)\right]$
    $S \leftarrow S'$
  until S is terminal

- This is $TD(0)$; other variants $TD(\lambda)$, n-step TD.... (later)

DP, MC, TD: Know Your Targets

$$\begin{aligned} v_\pi(s) &= E_\pi \left[G_t \mid S_t = s \right] \\ &= E_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s \right] \\ &= E_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s \right] \end{aligned}$$

- Monte Carlo methods use estimates of $E_\pi \left[G_t \mid S_t = s \right]$ as target.
- DP methods use estimates of $E_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s \right]$ as target.
- Estimates(!) in both cases.
- TD target $R_{t+1} + \gamma V(S_{t+1})$ borrows from both
- This is an important insight; (see page 120, S & B).

Temporal Difference and Monte Carlo Errors

- TD Error

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

- Expanding Montr Carlo Error:

$$\begin{aligned}
G_t - V(S_t) &= R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1}) \\
&= \delta_t + \gamma(G_{t+1} - V(S_{t+1)}) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2(G_{t+2} - V(S_{t+2)}) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+1} + \ldots + \gamma^{T-t-1}\delta_{T-1}(G_T - V(S_T)) \\
&= \sum_{k=t}^{T-1} \gamma^{k-t}\,\delta_k
\end{aligned}$$

- Monte Carlo error is sum of Temporal Difference errors.
- (Recall similar structure in stochastic gradient descent / perceptron algorithms)

SARSA: On Policy Temporal Difference Control

Section 6.4, Sutton & Barto

- SARSA: Transitions from state-action pair to state-action pair

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\,[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

- If S_{t+1} is terminal state, $Q(S_{t+1}, A_{t+1}) = 0$

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop 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: Off Policy Temporal Difference Control

- Q Learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

- off policy because the target being set is not from the policy being followed (as in SARSA).

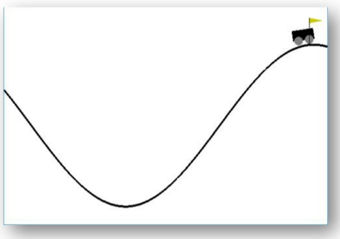
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Summary

- Temporal difference algorithms combine ideas of Dynamic Programming and Monte Carlo simulations.
- Two important algorithms: SARSA, Q-Learning
- Next:
  - Tabular methods fail at high dimensions
  - Function approximations
- Lab Work:
  - Mountain Car problem



  - Simulation environment `gym`

```
import numpy as np
import gym
```