AIL 722: Reinforcement Learning

Lecture 15: Temporal-Difference Prediction

Raunak Bhattacharyya

ScAI YARDI SCHOOL OF ARTIFICIAL INTELLIGENCE
INDIAN INSTITUTE OF TECHNOLOGY DELHI

Outline

- Incremental estimation of the value function
- Temporal difference prediction
- Example: Random walk

Monte Carlo Policy Evaluation

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

$$V^{\pi}(s^j) = \mathbb{E}_{p_\theta(\tau)}\left[\sum_{t'=t}^{T} \gamma^{t'-t} \cdot r(s_{t'}, a_{t'}) \middle| s_t = s^j\right]$$

Our goal: $\hat{V}^{\pi}(s^j)$

The **return** is defined as: $G_t = \sum_{t'=t}^{T} \gamma^{t'-t} \cdot r(s_{t'}, a_{t'})$

$$V^{\pi}(s^j) = \mathbb{E}_{p_\theta(\tau)}\left[G_t \middle| s_t = s^j\right]$$

Monte Carlo Policy Evaluation

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

Our goal: $\hat{V}^{\pi}(s^j)$

- Sample trajectories
- Store the obtained cumulative discounted reward
- Average

Monte Carlo Policy Evaluation

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

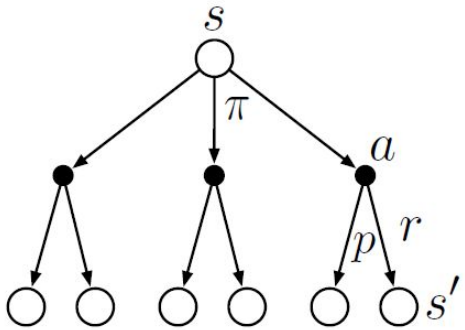
Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

Section 5.1, Reinforcement Learning: An Introduction, Sutton & Barto

About Monte Carlo



- Estimates for states are independent
- Estimate for one state does not build upon the estimate of any other state (this was the case in DP)
- Monte Carlo methods do not bootstrap

Computational expense of estimating the value of a single state is independent of the number of states

Section 5.1, Reinforcement Learning: An Introduction, Sutton & Barto

Incremental Approach

expectation of a single variable X from m samples:

$$\hat{x}_m = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$\hat{x}_m = \frac{1}{m}\left(x^{(m)} + \sum_{i=1}^{m-1} x^{(i)}\right)$$

$$= \frac{1}{m}\left(x^{(m)} + (m-1)\hat{x}_{m-1}\right)$$

$$= \hat{x}_{m-1} + \frac{1}{m}\left(x^{(m)} - \hat{x}_{m-1}\right)$$

Section 17.1, Algorithms for Decision Making, Kochenderfer

Incremental Approach

$$\hat{x}_m = \hat{x}_{m-1} + \alpha(m)\left(x^{(m)} - \hat{x}_{m-1}\right)$$

$$\sum_{m=1}^{\infty} \alpha(m) = \infty \qquad \sum_{m=1}^{\infty} \alpha^2(m) < \infty$$

$$\hat{x} \leftarrow \hat{x} + \alpha(x - \hat{x})$$

$$NewEstimate \leftarrow OldEstimate + StepSize\left[Target - OldEstimate\right]$$

Section 2.4, Reinforcement Learning: An Introduction, Sutton & Barto

Incremental Model-Free Policy Evaluation

mth sample

$$\hat{V}_m^\pi(s^j) \longleftarrow \hat{V}_{m-1}^\pi(s^j) + \alpha\left[G^{(m)} - \hat{V}_{m-1}^\pi(s^j)\right]$$

Estimate at m^{th} iteration

Estimate at $m\text{-}1^{th}$ iteration

Estimate at $m\text{-}1^{th}$ iteration

$$\hat{V}_m^\pi(s^j) \longleftarrow \hat{V}_{m-1}^\pi(s^j) + \alpha\left[G^{(m)} - \hat{V}_{m-1}^\pi(s^j)\right]$$

New estimate

Old estimate

Target

Monte Carlo vs. Dynamic Programming

$$V^{\pi}(s^j) = \mathbb{E}_{p_\theta(\tau)}\left[G_t \middle| s_t = s^j\right]$$

Target for Monte Carlo

$$V^{\pi}(s^j) = \mathbb{E}_{p_\theta(\tau)}\left[r_{t+1} + \gamma \cdot V^{\pi}(s_{t+1}) \middle| s_t = s^j\right]$$

Target for DP

Why do both approaches give estimates and not the actual value function until they converge?

Monte Carlo vs. Dynamic Programming

$$V^{\pi}(s^j) = \mathbb{E}_{p_\theta(\tau)}\left[G_t \middle| s_t = s^j\right]$$

Target for Monte Carlo

Uses a sample of the real thing

But not the expectation, just a single sample

$$V^{\pi}(s^j) = \mathbb{E}_{p_\theta(\tau)}\left[r_{t+1} + \gamma \cdot V^{\pi}(s_{t+1}) \middle| s_t = s^j\right]$$

Target for DP

Does compute expectations. Uses known model

Target not created using the real value function but rather the current estimate: Bootstrapping

Section 6.1, Reinforcement Learning: An Introduction, Sutton & Barto

Temporal Difference Policy Evaluation

$$\hat{V}_m^\pi(s^j) \longleftarrow \hat{V}_{m-1}^\pi(s^j) + \alpha \left[\left(r_{t+1} + \gamma \cdot \hat{V}_{m-1}^\pi(s_{t+1}) \right)^{(m)} - \hat{V}_{m-1}^\pi(s^j) \right]$$

No expectation, single sample

No reality, bootstrap

But it works!

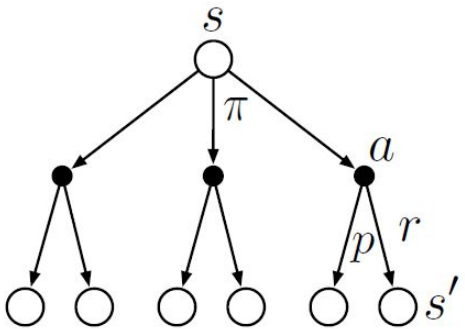
Algorithm: TD(0)

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 $V(terminal) = 0$

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'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

Backup Diagram Comparison



TD(0)

Pros of TD(0)

What's the advantage over Dyn Prog?

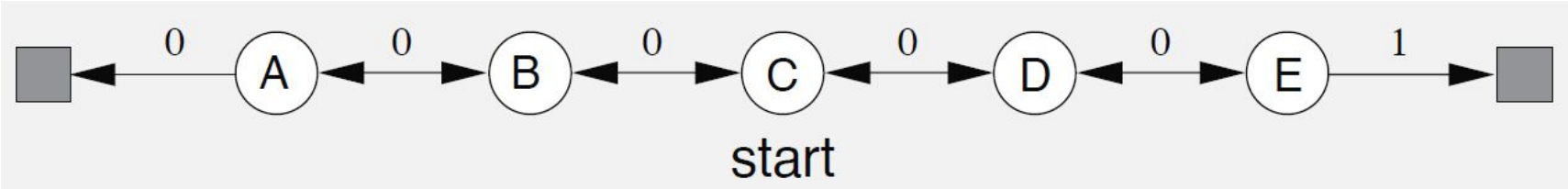
No model required

What's the advantage over Monte Carlo?

No need to wait until the end of the episode

Proven to converge to the true value function

Example: Random Walk



What is the value function?

Example 6.2, Reinforcement Learning: An Introduction, Sutton & Barto

Algorithm: TD(0)

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'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal