COMP6247: Reinforcement and Online Learning

Mahesan Niranjan

School of Electronics and Computer Science
University of Southampton

Monte Carlo Methods
Sampled from Chapter Five, Sutton and Barto

Spring Semester 2020/21

Monte Carlo Methods

- Dynamic Programming:
  - We have knowledge of environment
  - We solve the task of finding optimal policies
- Next steps:
  - We learn to model the environment
  - And learn optimal policies along the way
  - ... approximating expectations
  - ... learning by successive approximation

Monte Carlo Methods

- We will consider episodic tasks; every episode (trial) terminates
- We will update values (of states / actions) when each episode ends
- Incremental in episode by episode sense, not at every step (as in online)
- *Monte Carlo* – techniques that have random searches and averaging
- For every state-action pair, we generate sample returns and average them to get updates
- In Dynamic programming, we *computed* the value function from *full knowledge* of the MDP; here, we *learn* the value function from *sample returns* of the MDP.

Monte Carlo Updates: $v_\pi(s)$

- *First-visit* and *every-visit* Monte Carlo averaging.
- Forms of update: difference in formal derivation of convergence.

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

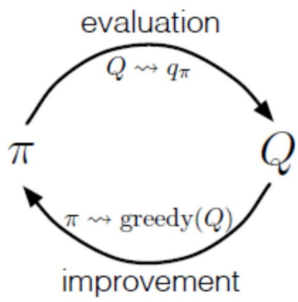
Monte Carlo Updates: $q_\pi(s, a)$

- Approach same as before
- Issue: Some actions may never be chosen if the policy is deterministic (*i.e.* when in any state, we will choose the same action
- *Exploring starts:* Episodes start in a state-action pair with every pair having a non-zero probability
- Infinitely many visits and exploring starts are necessary for formal convergence of MC.
- Practical algorithm: Can we improve on these assumptions? [Page 98, brief discussion]

Monte Carlo Control

Section 5.3, Sutton and Barto

- Control - Approximate optimal policy



- Policy evaluation (prediction) from Monte Carlo averages
- Policy improvement: greedy with respect to current value function

$$\pi(s) = \arg\max_a q(s, a)$$

Policy Estimation Monte Carlo Exploring Starts

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:
  $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
  $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
  $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):
  Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
  Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
      Append G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
      $\pi(S_t) \leftarrow \text{argmax}_a\, Q(S_t, a)$

efficient.

Monte Carlo without Exploring Starts

- *On policy* and *Off policy* methods
- Data generated by following the policy of interest or by applying a different policy
- On policy control methods, usually $\pi(a|s)\,\forall\, s$ and a, gradually shift towards a deterministic policy

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$
Initialize:
  $\pi \leftarrow$ an arbitrary ε-soft policy
  $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
  $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):
  Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
      Append G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
      $A^* \leftarrow \text{argmax}_a\, Q(S_t, a)$ (with ties broken arbitrarily)
      For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

Off Policy Predictions via Importance Sampling

- Policy of interest: *target policy*
- Generate data from: *behaviour policy* $b(A_t|S_t)$
- Draw parallel to importance sampling in Bayesian inference (and particle filters)
- Probability of state-action trajectory:

$$\begin{aligned}\Pr\{A_t, S_{t+1}, A_{t+1}, ..., S_T\} &= \pi(A_t|S_t)\, p(S_{t+1}|S_t, A_t)\, \pi(A_{t+1}|S_{t+1}) \dots p(S_T|S_{T-1}, A_t) \\ &= \prod_{k=t}^{T-1} \pi(A_k|S_k)\, p(S_{k+1}|S_k, A_k)\end{aligned}$$

- Importance ratio (Note state transition probabilities cancel):

$$\begin{aligned}\rho_{t:T-1} &= \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k) p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k) p(S_{k+1}|S_k, A_k)} \\ &= \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)}\end{aligned}$$

- Weighted sum as estimated value: $E\left[\rho_{t:T-1}\, G_t \,|S_t = s\right] = v_\pi(s)$

Incremental Implementation

Section 5.6, Sutton & Barto

- Suppose we have a sequence of returns: $G_1, G_2, ..., G_{n-1}$
- All returns starting from the same state
- Corresponding weights: W_i : $\rho_{t_i:T_{t_i}-1}$
- Weighted returns:

$$V_n = \frac{\sum_{k=1}^{n-1} W_k\, G_k}{\sum_{k=1}^{n-1} W_k}$$

- Can do this incrementally (Eqn 5.8, S & B):

$$\begin{aligned}V_{n+1} &= V_n + \frac{W_n}{C_n}\left[G_n - V_n\right] \\ C_{n+1} &= C_n + W_{n+1}\end{aligned}$$

Off Policy Monte Carlo Algorithm

Off-policy MC prediction (policy evaluation) for estimating $Q \approx q_\pi$

Input: an arbitrary target policy π
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \in \mathbb{R}$ (arbitrarily)
    $C(s,a) \leftarrow 0$

Loop forever (for each episode):
    $b \leftarrow$ any policy with coverage of π
    Generate an episode following b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$, while $W \neq 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t,A_t)} [G - Q(S_t, A_t)]$
        $W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$

Summary

- Dynamic programming was about
    - Known environment
    - Finding an optimal policy to maximize returns
    - Value and policy iterations
- Here: Learning about the environment
    - Simulate many times and average
    - Expected values (returns) as sample averages
    - On-policy and Off-policy approaches
- Next: Temporal Difference Algorithms
  Update estimates “on the fly” **without waiting for** end of episodes (true online learning)