AIL 722: Reinforcement Learning

Lecture 17: Monte Carlo Control

Raunak Bhattacharyya

YARDI SCHOOL OF ARTIFICIAL INTELLIGENCE
INDIAN INSTITUTE OF TECHNOLOGY DELHI

Outline

- From evaluation/prediction to control
- State-action values: Exploring starts
- Epsilon-soft policies
- On-policy and off-policy algorithms

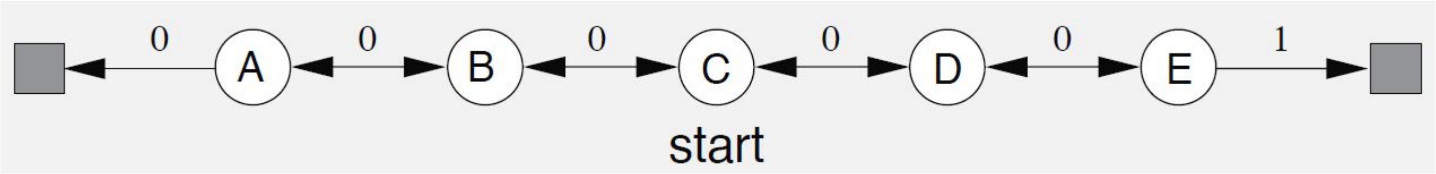
Recap

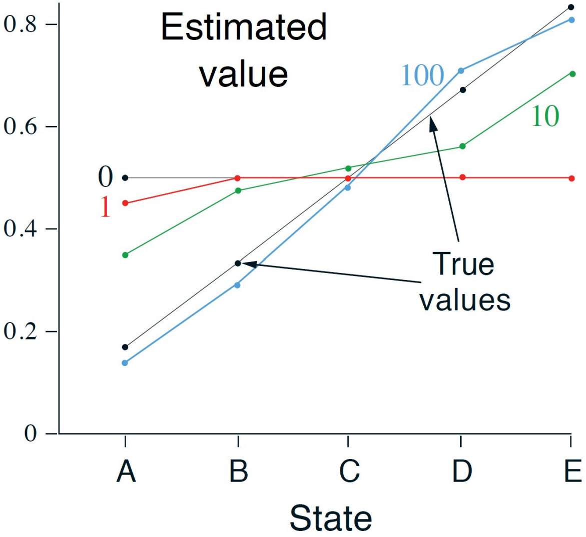
- Model free prediction

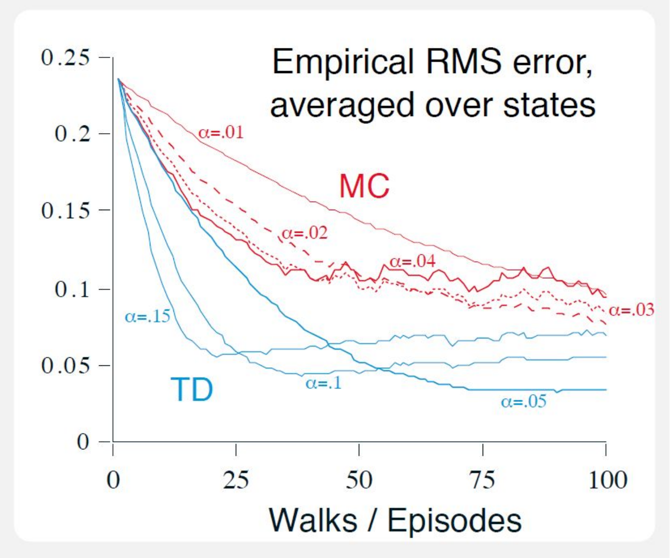
- Monte-Carlo, Temporal Difference. Important to develop MC ideas first and then repurpose for TD.

- Implementation of TD. Baselined against ground truth obtained through iterative policy evaluation

Example: Random Walk







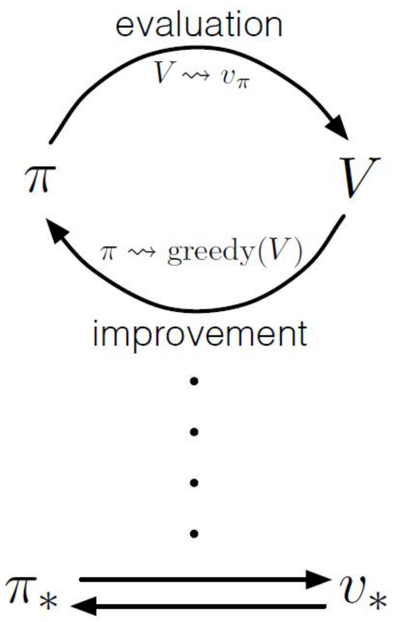
Example 6.2, [Reinforcement Learning: An Introduction](), Sutton & Barto

Clarification: Bootstrapping

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

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

Generalised Policy Iteration



- Two simultaneous, interacting processes
  - Make value fun consistent with current policy
  - Make policy greedy w.r.t. current value function
- In PI, these processes alternate, each completing before other begins
- In VI, single iteration of policy evaluation between each policy improvement

GPI: Evaluation and improvement processes interact, independent of granularity

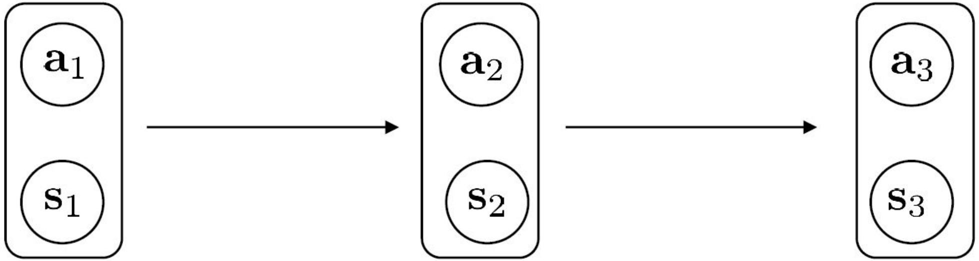
Model-free evaluation in GPI?

Section 4.6, [Reinforcement Learning: An Introduction](), Sutton & Barto

Monte-Carlo Estimation of Action-Values

Why is estimating Q-values helpful towards control?

$$p((\mathbf{s}_{t+1}, \mathbf{a}_{t+1})|(\mathbf{s}_t, \mathbf{a}_t)) =$$
$$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\pi_\theta(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})$$



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: $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

    $G \leftarrow 0$

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

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

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

            Append G to $Returns(S_t)$

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

What's the problem with this algorithm for estimating Q-values?

Section 5.1, [Reinforcement Learning: An Introduction](), Sutton & Barto

Monte Carlo: Exploring Starts and Control

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 $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

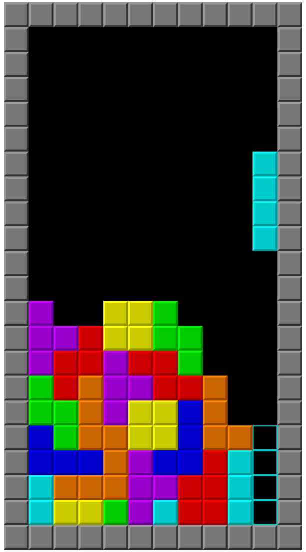
Loop forever (for each episode):
- Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
- Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1,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

What's the interleaving between evaluation and improvement processes?

After every episode

Section 5.3, Reinforcement Learning: An Introduction, Sutton & Barto

Monte Carlo: Without Exploring Starts



How do we do exploring starts in tetris?

In Monte-Carlo ES and control, the policy was set to greedy w.r.t. Q. End of exploration

For all actions to continue to be selected infinitely often, the policy has to continue to select them

$\epsilon - greedy$ policy:

For all actions to continue to be selected infinitely often, the policy has to continue to select them

$$\pi_{\text{new}} = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}|} & \text{if } a = \arg\max_a Q(s,a) \\ \frac{\epsilon}{|\mathcal{A}|} & \text{otherwise} \end{cases}$$

$$\pi(a|s) \geq \frac{\epsilon}{|A(s)|}$$

More general: epsilon-soft policy

Monte Carlo: Stochastic Exploration 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 $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
    - $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
    - For all $a \in \mathcal{A}(S_t)$:

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

Section 5.4, [Reinforcement Learning: An Introduction](), Sutton & Barto

An Issue

- Optimal policy may be deterministic

- But we need stochastic policies to visit state action pair infinitely often
    - To estimate the state-action value function

Get to an almost deterministic final policy (that still explores)

On-policy type of algorithms

Use one policy to explore. Using the exploration, update another (deterministic) policy, which eventually becomes the optimal policy

Off-policy type of algorithms

On-Policy and Off-Policy

Get to an almost deterministic final policy (that still explores)

On-policy type of algorithms

Use one policy to explore. Using the exploration, update another (deterministic) policy, which eventually becomes the optimal policy

Off-policy type of algorithms

- Monte Carlo exploring starts

- Monte Carlo epsilon-soft

On-policy/off-policy?

Summary & Announcements

- Summary:
  - Need to visit (s,a) pairs infinitely often
    - Exploring starts
  - Exploring starts unrealistic in real problems
    - Stochastic policies
  - But optimal policy may be deterministic
    - Importance sampling

Course webpage