Reinforcement Learning
Lecture 4: Monte Carlo Methods

Lecturer: Prof. Dr. Mathias Niepert

Institute for Artificial Intelligence
Machine Learning and Simulation Lab

University of Stuttgart
Germany

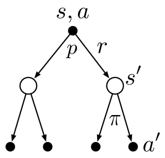
imprs-is

May 2, 2024

Outline

Recap: Recursive relationship for q_π

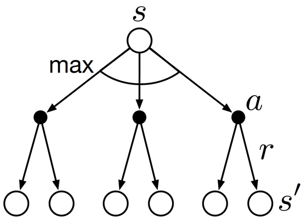
$$
\begin{aligned}
q_\pi(s,a) &= \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a\right] \\
&= \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i R_{t+i+1} \mid S_t = s, A_t = a\right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a\right] \\
&= \sum_{s',r} p(s',r \mid s,a)\Big[r + \gamma \mathbb{E}_\pi \left[G_{t+1} \mid S_{t+1} = s', A_{t+1} = a'\right]\Big] \\
&= \sum_{s',r} p(s',r \mid s,a)\Big[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a')\Big]
\end{aligned}
$$



Recap: Bellman optimality equation for v_*

- Value functions define an ordering over policies
- Value under optimal policy = expected return for **best** action from that state
- Optimal value function is a fixed point of the General Policy Iteration (GPI) algorithm

$$
\begin{aligned}
v_*(s) &= \max_a q_*(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}\left[G_t \mid S_t = s, A_t = a\right] \\
&= \max_a \mathbb{E}_{\pi_*}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a\right] \\
&= \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right] \\
&= \max_a \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_*(s')\Big]
\end{aligned}
$$

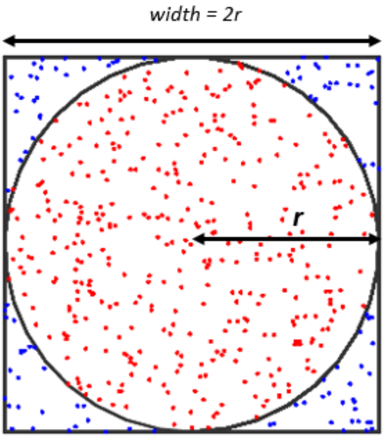


Recap: Value and Policy Iteration

- Why is a deterministic policy guaranteed to be optimal?
- In dynamic programming, we assume full access to the transition dynamics $p(s' \mid s, a)$ and the reward function of the MDP
- How do we compute value and action value functions (and optimal policies) when we can only "follow" policies and sample trajectories?

Monte Carlo Prediction

Monte Carlo Integration



Monte Carlo Integration

- **Estimate** integral

$$\mathbb{E}\left[f(x)\right] = \int f(x)p(x)dx$$

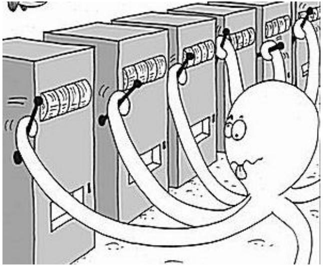
- **Draw samples** $x_i \overset{\text{i.i.d.}}{\sim}$ [1] $p(x)$ and approximate the integral as

$$\hat{f} \approx \frac{1}{L}\sum_{l=1}^{L} f(x_l)$$

- The **empirical mean estimator** $\hat{f}$ converges to the true mean $\mathbb{E}\left[f(x)\right]$ as the number of samples L increases (law of large numbers)

[1] i.i.d. means Independent and Identically Distributed: each random variable has the same probability distribution as the others and all are mutually independent

Example: k-armed bandit



- There are k actions (machines) and each machine returns a reward from a (stationary) probability distribution
- Objective is to maximize the expected total reward, aggregated over the first T choices of machines
- We have no access to the probability distribution over rewards so want to compute the expected value of the reward (**an integral**!)

Monte Carlo Integration

Estimation of the expected output of a black box function f w.r.t. some distribution over inputs

Assume we have access to

- Samples (i.i.d.) from prior distribution over states
- Black box function that encodes environment and returns sequence of experience

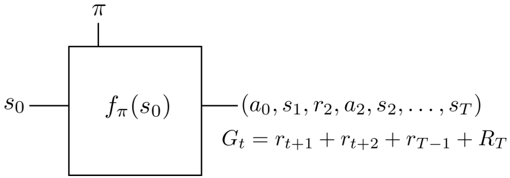


Figure: Black box view on RL

Use Monte Carlo methods to estimated the expected output or a function of it, e.g. the expected cumulative reward.

Monte Carlo methods in RL

- **Learn** value function from *experience*
- **Discover** optimal policies
- **Blackbox view** does not require knowledge of the environment: $p(s'|s,a)$ & $r(s,a,s')$
- *Experience:* sample sequences of states, actions, rewards: $S_1, A_1, R_2, S_2, A_2, \ldots$
  - real experience: interaction with the environment
  - simulated experience: interaction with a simulator
- Achieve optimal behavior

Monte Carlo principle

- Divide experience into episodes
- All episodes must terminate!
- Maintain estimates of value (action value) function
- Update estimate at end of each episode
- MC vs. DP:
  - update every episode vs. every step
  - We cannot use value function to derive improved policy (more later)

Returns

- Return = Expected cumulative future discounted reward
- Return for finite episode starting at time t:
  $G_t = R_{t+1} + \gamma R_{t+2} + \ldots \gamma^{T-2} R_{T-1} + \gamma^{T-1} R_T$
- Discounted sum of immediate rewards up to and including terminal state at $t = T$
- $v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$ is the expected cumulative discounted reward
- We cannot solve the Bellman equation for $v_\pi(s)$ explicitly since we don't have access to the dynamics
- *Idea:* value function for state s is approximated by average returns over many episodes starting from s
  - approximation of value function $v_\pi(s)$ for that state and policy π

First–visit vs. every–visit MC

- A state can occur more than once in one episode
- First–visit MC:
  - Estimate $v_\pi(s)$ as the average of returns following **first** visits to s
- Every–visit MC:
  - Estimate $v_\pi(s)$ as the average of returns following **every** visit to s
- Both strategies converge to $v_\pi(s)$ as the number of visits to s goes to infinity

Properties of MC

- Estimates of v for each state are independent
- Compute time is independent of $|\mathcal{S}|$
- If only a few states are relevant, we can generate episodes from those states and ignore the value of others
- No need to know the full model $p(s'|s,a)$ and $r(s,a,s')$
- Learning from real/simulated experience
- Often (i.e. in games) it is possible to generate transitions without actually having explicit access to p

Monte Carlo Prediction (Estimation of v_π)

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, $t = T-1, T-2, \ldots, 0$:

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

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

Append G to $Returns(S_t)$

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

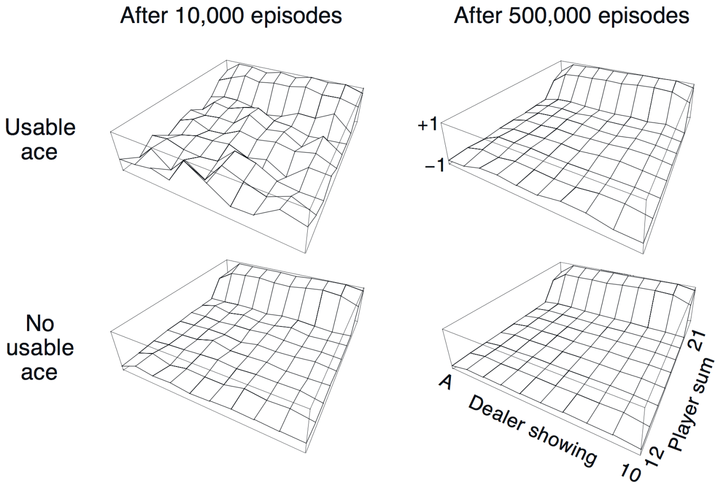
Backup diagram

- Entire episode included
- Only single choice considered at each state
- Thus, there will be an explore/exploit dilemma
- Value is estimated by mean return

Blackjack example

Wikipedia

- **Objective:** your card sum greater than the dealer's without exceeding 21
- Number cards count as their number, the jack, queen, and king count as 10, and aces count as either 1 or 11
- **Actions**
  - *stick* (no more cards)
  - *hit* (receive another card)
- **States**
  - current sum (12-21), we do not consider cases bellow 12 $\rightarrow$ always hit
  - dealer's showing card (A-2-3-...-9-10)
  - usable ace (can be counted as 11)?
- **Reward:** $+1$ for winning, 0 for a draw, -1 for losing
- **Policy:** stick if sum > 20, else hit


After 10,000 episodes
After 500,000 episodes
Usable ace
No usable ace
+1
−1
A
Dealer showing
10
12
Player sum
21

Monte Carlo Control

Recap: Policy Iteration

- Alternating policy **evaluation** and policy **improvement**
- **Policy evaluation:** estimate v_π for fixed π
- **Policy improvement:** determine greedy policy π' w.r.t. to v_π
- Iterate until optimal value function & policy is reached
- We can use *Monte Carlo* instead of DP for policy *evaluation* in policy *iteration*
- MC estimates the value function given a policy

Policy Improvement: Value vs. Action Value Functions

- In DP, when we have the full knowledge of the MDP dynamics $p(s', r \mid s, a)$, the best policy π' wrt current value function can be obtained

$$\pi'(s) = \arg\max_a \sum_{s', r} p(s', r \mid s, a)\Big[r + \gamma v_\pi(s')\Big]$$

- **Not** the case for the Monte Carlo setting where the assumption is that we have no direct access to $p(s', r \mid s, a)$
- Here we need estimates of the action values to extract optimal policy wrt the current MC estimates of the action value function

$$\pi'(s) = \arg\max_a q_\pi(s, a).$$

Estimating q–values

- Same principle as for v_π

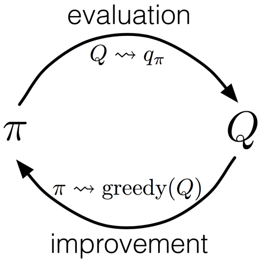
$$q_\pi(s, a) = \mathbb{E}_\pi\Big[G_t \mid S_t = s, A_t = a\Big]$$

- Update estimate $q_\pi(s, a)$ by averaging returns following first visit to that state–action pair (s, a)
- *Warning:* if the policy is deterministic, some (s, a) pairs may never be visited

Monte Carlo control

- **MC policy iteration step:** policy evaluation using MC methods
- **Policy improvement step:** greed w.r.t. to action–value

$$\pi_0 \xrightarrow{\mathsf{E}} q_{\pi_0} \xrightarrow{\mathsf{I}} \pi_1 \xrightarrow{\mathsf{E}} q_{\pi_1} \xrightarrow{\mathsf{I}} \pi_2 \xrightarrow{\mathsf{E}} \dots \xrightarrow{\mathsf{I}} \pi_* \xrightarrow{\mathsf{E}} q_*$$



Greedy policy

- For any action–value function q_π, the corresponding greedy policy is:

$$\pi'(s) = \arg\max_a q_\pi(s, a)$$

- *Policy improvement* is simply constructing each π_{k+1} as the greedy policy w.r.t. to q_{π_k}

$$\pi_{k+1}(s) = \arg\max_a q_{\pi_k}(s, a)$$

Convergence of MC control

$$
\begin{aligned}
v_{\pi_{k+1}}(s) &= q_{\pi_k}(s, \pi_{k+1}(s)) \\
&= q_{\pi_k}(s, \arg\max_a q_{\pi_k}(s,a)) \\
&= \max_a q_{\pi_k}(s,a) \\
&\geq q_{\pi_k}(s, \pi_k(s)) \\
&= v_{\pi_k}(s)
\end{aligned}
$$

- Thus π_{k+1} must be equal or better than π_k
- Assumes *exploring starts*, that is, non-zero probability that state-action pair is selected as start
- In the limit of an infinite number of episodes, this guarantees that every pair will be visited an infinite number of times

Monte Carlo ES (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 $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ 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, $t = T-1, T-2, \ldots, 0$:

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

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 $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

On & Off Policy

On–policy Monte Carlo control

- **On-policy:** learn about policy currently used to generate experience
- We must explore since we need to visit as many states as possible
- How do we avoid the assumption of exploring starts?
- E.g., using ϵ–greedy or softmax policies, i.e., $\pi(s, a) > 0$ for all (s, a)
- **Off-policy:** Evaluate and improve a policy that is different from the one used for generating episodes

On-policy Monte Carlo control

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 Monte Carlo control

Learn the value of the **target policy** π from experience generated using a **behavior policy** μ

- For example, π is the **greedy policy** (thus ultimately the optimal policy), while μ is an exploring (e.g. softmax) policy
- In general, we only require that μ generates behavior that *covers*/includes π

$$\pi(a \mid s) > 0 \Rightarrow \mu(a \mid s) > 0 \; \forall s, a$$

- *Idea:* Compute MC estimates from the trajectories generated by μ but make adjustments such that we obtain estimates compatible with π

Importance Sampling

The Problem with off-policy prediction

- Recall that we want to obtain (estimate) $\mathbb{E}_\pi [G_t \mid S_t = s] = v_\pi(s)$ (or the action value function, as seen above)
- If we use MC methods to estimate value and action value functions from episodes generated by a different policy μ we have biased (incorrect) estimates
- We need to find a way to obtain estimates that are unbiased with respect to π while looking at episodes generated by μ

Importance sampling

- **Target distribution** $p(x)$ from which it's complicated to draw samples
- **Proposal distribution** $q(x)$ from which it's easy to draw samples
- We need to be able to evaluate $p(x)$ numerically

$$\mathbb{E}_{p(x)}[f(x)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = \mathbb{E}_{q(x)}\left[f(x)\frac{p(x)}{q(x)}\right]$$

$$\approx \frac{1}{L}\sum_{l=1}^{L} f(x_l)\underbrace{\frac{p(x_l)}{q(x_l)}}_{w_l} \quad \text{,with samples } x_l \overset{i.i.d.}{\sim} q(x)$$

- The ratio w_l is called importance weight
- Choice of **proposal distribution** $q(x)$ is crucial for efficiency

Importance sampling for off policy prediction

Consider the trajectory $\psi = (a_t, s_{t+1}, a_{t+1}, \ldots, s_T)$

$$\rho_{t:T-1} = \frac{Pr\{\psi \mid \pi\}}{Pr\{\psi \mid \mu\}} = \frac{\prod_{k=t}^{T-1} \pi(a_k \mid s_k) p(s_{k+1} \mid s_k, a_k)}{\prod_{k=t}^{T-1} \mu(a_k \mid s_k) p(s_{k+1} \mid s_k, a_k)} = \prod_{k=t}^{T-1} \frac{\pi(a_k \mid s_k)}{\mu(a_k \mid s_k)}$$

Ordinary importance sampling

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}$$

Weighted importance sampling

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$

Notation: Time step numbering increases across episodes boundaries

- $\mathcal{T}(s)$ denotes the set of all time steps in which state s is visited
- $T(t)$ the first time of termination following time t

Summary

- Monte Carlo has several advantages over dynamic programming:
  - can learn directly from experience
  - no need for full models
  - less harmed by violating Markov property
- MC methods provide an alternative to policy evaluation
- MC requires sufficient exploration
- On–policy vs. off-policy methods
- Importance sampling for off–policy