Reinforcement Learning

Monte Carlo Methods

Heiko Zimmermann

15.05.2017

Monte Carlo

- Monte Carlo policy evaluation
- First–visit policy evaluation
- Estimating q–values
- On–policy methods
- Off–policy methods
- Importance sampling

Monte Carlo Integration

- **Estimate** integral

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

- **Draw samples** $x_i \overset{i.i.d.}{\sim} p(x)$ 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 $\mathbb{E}[f(x)]$ as the number of samples L increases (law of large numbers)

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

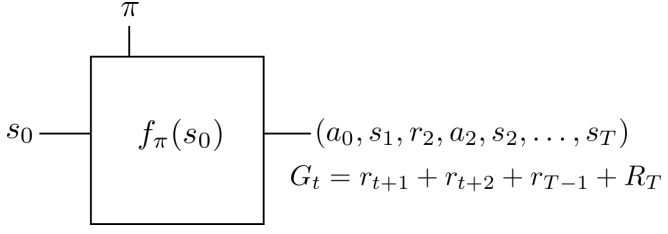


Figure 1: Black box view on RL

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!
- Keep estimates of value function / policy
- Update estimate at end of each episode
- MC vs. DP:
  - update every episode vs. every step
  - average total return vs. bootstrapping (average 1-step lookahead)

Returns

- Return at time t: $G_t = R_{t+1} + R_{t+2} + \dots R_{T-1} + R_T$
- Total sum of immediate rewards up to and including terminal transition at $t = T$
- *Idea:* average returns over many episodes starting from same state s
  - gives the value function $v_\pi(s)$ for that state and policy π
  - $v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$ is the expected cumulative future discounted reward
  - $\gamma = 1$ as every episode terminates and we update at end of epsiode

Monte Carlo learning of v_π

- Learn v_π from experience
- Generate many *plays*/episodes starting from state s
- Observe total reward of each *play*
- Average over many *plays*

1. **Initialize:**

   $\pi \leftarrow$ policy to be evaluated
   $V \leftarrow$ arbitrary state-value function
   $Returns(s) \leftarrow$ empty list for all $s \in S$

2. **Repeat forever:**

   Generate episode using π
   for each state s in episode **do**
      $G \leftarrow$ return following the first occurrence of s
      Append G to $Returns(s)$
      $V(s) \leftarrow average(Returns(s))$
   end for

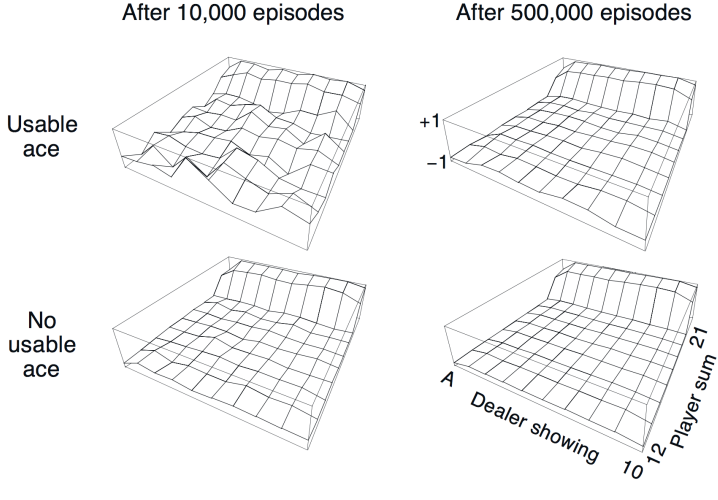
Backup diagram

- Entire episode included
- Only single choice considered at each state (non-averaging like DP)
- Thus, there will be an explore/exploit dilemma
- No bootstrap from successor states' values
- Value is estimated by mean return

Figure 2:

Blackjack example

- **Objective:** your card sum greater than the dealer's without exceeding 21
- **States:**
  - current sum (12-21)
  - dealer's showing card (A-10)
  - useable ace?
- **Reward:** $+1$ for winning, 0 for a draw, -1 for losing
- **Actions:** *stick* (no more cards), *hit* (receive another card)
- **Policy:** stick if sum ≥ 20, else hit
- No discounting ($\gamma = 1$)s


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

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 policy *evaluation* in policy *iteration*
- MC estimates the value function given a policy

First–visit vs. every–visit MC

- Fist–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
  - no bootstrapping!
  - thus no need to learn about all states
- 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 from p without actually having to express p explicitly
- Less harmed by violating Markov property

Estimating q–values

- Same principle as for v_π

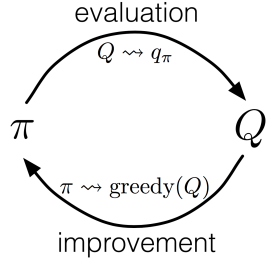
$$q_\pi(s, a) = \mathbb{E}_\pi \Big[G_t \mid S_t = s, A_t = a \Big]$$

- 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

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



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}

Convergence of MC control

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

- Thus π_{k+1} must be equal or better than π_k
- Assumes *exploring starts* and *infinite number of episodes* for MC policy evaluation

Monte Carlo ES (exploring starts)

1. **Initialize** for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$

   $Q(s,a) \leftarrow$ arbitrary
   $\pi(s) \leftarrow$ arbitrary
   $Returns(s,a) \leftarrow$ empty list

2. **Repeat forever:**

   Choose $S_0 \in \mathcal{S}, A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
   Generate episode starting from (S_0, A_0) following π
   for each pair (s,a) in episode **do**
      $G \leftarrow$ return following the first occurrence of (s,a)
      Append G to $Returns(s,a)$
      $Q(s,a) \leftarrow average(Returns(s,a))$
   end for
   for each s in episode **do**
      $\pi(s) \leftarrow \arg\max_a Q(s,a)$
   end for

On–policy Monte Carlo control

- **On-policy:** learn about policy currently used to generate experience
- We must explore
- 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)

On-policy Monte Carlo control

1. **Initialize** for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$

   $Q(s, a) \leftarrow$ arbitrary

   $Returns(s, a) \leftarrow$ empty list

   $\pi(a \mid s) \leftarrow$ arbitrary ϵ-soft policy

2. **Repeat forever:**

   Generate episode using π

   for each pair (s, a) in episode **do**

   $G \leftarrow$ return following the first occurrence of (s, a)

   Append G to $Returns(s, a)$

   $Q(s, a) \leftarrow average(Returns(s, a))$

   end for

   for each s in episode **do**

   $a^* \leftarrow \arg\max_a Q(s, a)$

   for each a in $\mathcal{A}(s)$ **do**

$$\pi(a \mid s) \leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|} & \text{if } a = a^* \\ \frac{\epsilon}{|\mathcal{A}(s)|} & \text{if } a \neq a^* \end{cases}$$

   end for

   end for

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:* **importance sampling**
  - weight each return by the ratio of the probabilities of the trajectory under the two policies

Importance sampling

- **Target distribution** $p(x)$ from which it's complicated to fraw 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 $\psi = (a_t, s_{t+1}, a_{t+1}, \ldots, s_T)$

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

Ordinary importance sampling

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} p_t^{T(t)} G_t}{|\mathcal{T}(s)|}$$

Weighted importance sampling

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} p_t^{T(t)} G_t}{\sum_{t \in \mathcal{T}(s)} p_t^{T(t)}}$$

Notation: Time step numbering increases across episodes boundaries

- $\tau(s)$ 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