Monte Carlo Methods

Machine Learning

Daniele Loiacono

POLITECNICO MILANO 1863

Outline and References

- Outline
  - Policy Evaluation
  - Policy Iteration
  - ϵ-Soft Policy Iteration
  - Off-Policy Learning

- References
  - Reinforcement Learning: An Introduction [RL Chapter 5]
  - Sample-based Learning Methods (Coursera)

Why do we need sample-based methods?

- With Dynamic Programming we are able to find the optimal value function and the corresponding optimal policy
- For example, we can use Value Iteration:

$$V_{k+1}(s) \leftarrow \max_a \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V_k(s') \right], \forall s \in \mathcal{S}$$

- Which is the major limitation of this approach?
- We generally do not know the problem dyanmics!
- We want methods to learn the optimal policy directly from data!

we generally don't know the expected reward for each action and we don't know the probability of the next states

→ we want to learn from data

What is "data" in this context? A direct interaction with our environment (problem) (:= EXPERIENCE)

)

sequence of actions, states and rewards, that we observe by a direct interaction with the environment

Overview of Monte Carlo Methods

- Monte Carlo methods relies only on the experience (**data**) to learn value functions and policy
- Monte Carlo methods can be used in two ways:
  - **model-free:** no model necessary and still attains optimality
  - **simulated:** needs only a **simulation**, not a **full** model
- Monte Carlo methods learn from **complete** sample returns
- Only defined for episodic tasks

because we need complete sample returns

we don't have the full knowledge of the problem (we may not know the equations of the one step dynamics), but we can simulate the problem

Policy Evaluation

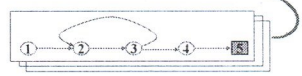
Monte Carlo Policy Evaluation

- Goal: learn $V_\pi(s)$
- Given: some number of episodes under π which contain s
- Idea: Average returns observed after visits to s

$V_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s]$ ⟹ $V_\pi(s) \approx average[G_t|S_t = s]$

expected return from state **s** following the policy **π**

$\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s$



What if we have:

and we want $V_\pi(2)$?
We have two different approaches:

- Every-Visit MC: average returns for **every** time s is visited in an episode
- First-visit MC: average returns only for **first** time s is visited in an episode
- Both converge asymptotically

First-visit 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 $Returns(S_t)$
      $V(S_t) \leftarrow$ average$(Returns(S_t))$

states, actions and rewards all selected according to the policy π

going backward is easier to update the returns

First-visit 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 $s \in$
Loop forever (for each episode):
  Generate an episode following π: S_0, A_0
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1,$
    $G \leftarrow \gamma G + R_{t+1}$
    Unless S_t appears in $S_0, S_1, \ldots, S_{t-}$
      Append G to $Returns(S_t)$
      $V(S_t) \leftarrow$ average$(Returns(S_t))$

Incremental Updates

$$N(s_t) \leftarrow N(s_t) + 1$$
$$V(s_t) \leftarrow V(s_t) + \frac{1}{N(s_t)}(G - V(s_t))$$

or

$$V(s_t) \leftarrow V(s_t) + \alpha(G_t - V(s_t))$$

incremental version of the empirical mean

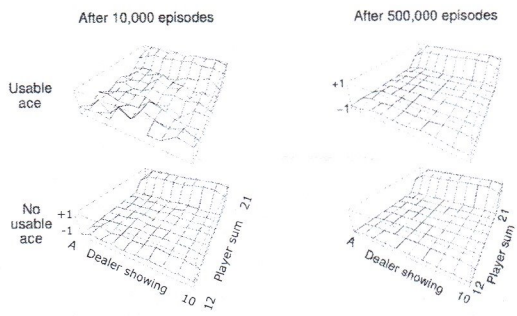
running mean (the greater is α the faster we forget previous values)

Blackjack example

- Goal: Have your card sum be greater than the dealer's without exceeding 21
- State space (200 states):
  - current sum (12-21)
  - dealer's showing card (ace-10)
  - do I have a useable ace? (yes/no)
- Reward: +1 for winning, 0 for a draw, -1 for losing
- Actions: stick (stop receiving cards), hit (receive another card)
- Policy: Stick if my sum is 20 or 21, else hit
- No discounting ($\gamma = 1$)

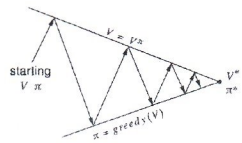
stop collecting if the sum is 20/21, keep collecting if not

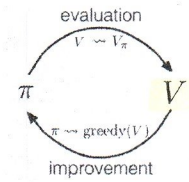
Blackjack example (2)



Policy Iteration

Let's go back to Generalized Policy Iteration





- Evaluation
  - Monte Carlo Policy Evaluation
- Improvement

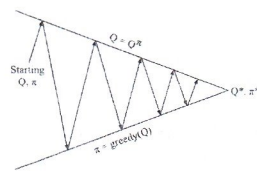
$$\pi'(s) = \arg\max_a \left\{ \boxed{r(s,a)} + \gamma \sum_{s' \in \mathcal{S}} \boxed{p(s'|s,a)} V_\pi(s') \right\}$$

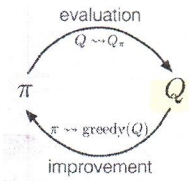
(? ?)

we cannot use it with Monte Carlo (we don't no either $r(s,a)$ or $p(s'|s,a)$)

Let's go back to Generalized Policy Iteration

Instead of doing the evaluation V_π (Useless for the improvement) we do the evaluation of the action values Q_π. From here the improvement is trivial (since it doesn't depend on the one-step dynamics)





- Evaluation
  - Monte Carlo Action Values Evaluation
- Improvement

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

Monte Carlo Action Values Evaluation

- We average return starting from state *s* and action *a* following π:

$$Q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \quad \Longrightarrow \quad Q_\pi(s,a) \approx average[G_t | S_t = s, A_t = a]$$

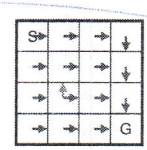
↑ we need to be in state s and also perform action a

- Converges asymptotically **if every** state-action pair is visited:

Exploration

- Exploring Starts

$s_0, a_0, s_1, a_1, s_2, a_2, \ldots$
Random From π and p



! This is something new! In the previous case we only needed some episodes of experiences following the policy π. Here the situation is different.

Let's assume we have a policy π that in state s_1 is:

$$\pi(s_1|a) = \begin{cases} A_1 & \text{with } P = 1 \\ A_2 & P = 0 \\ A_3 & P = 0 \end{cases}$$

if we generate our experiences using this policy π we will never be able to estimate $Q_\pi(s_1, A_2)$ or $Q_\pi(s_1, A_3)$ because we'll never experience these.

→ We have to do an EXPLORATION := we have to take actions that are not necessarily suggested by the policy π.

The easiest approach to solve this is to use the "EXPLORING STARTS" (↑)

keep following a policy π to generate the data, but at least in the first step move completely randomly: we start from a state randomly and we choose a random action. If we do it consistently we obtain that minimum amount of exploration that allows to observe all the possible state–actions pairs.

Monte Carlo Action Values Evaluation

- We average return starting from state *s* and action *a* following π:

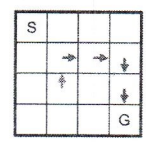
$$Q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \quad \Longrightarrow \quad Q_\pi(s,a) \approx average[G_t | S_t = s, A_t = a]$$

- Converges asymptotically **if every** state-action pair is visited:

Exploration

- Exploring Starts

$s_0, a_0, s_1, a_1, s_2, a_2, \ldots$
Random From π and p



Monte Carlo Policy Iteration with Exploring Starts

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

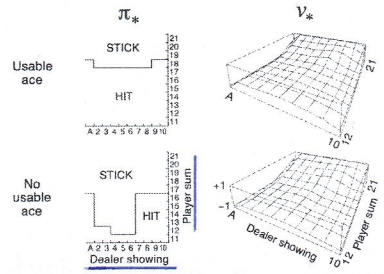
Initialize:
$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

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 $Returns(S_t, A_t)$
$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

(again) going backward is easier to update the returns

Blackjack example

- We start from the same policy previously described
- Here is the policy we found with MC and exploring starts



In the previous case we were evaluating a policy, now we're computing a policy (optimal)

ε-Soft Monte Carlo Policy Iteration

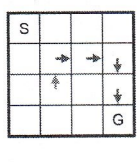
we need to introduce another form of exploration

Why ε-Soft Policies?

- Exploring starts is a simple idea but it is not always possible:

we might not have a full control on the environment in order to start from a random state making a random action. The control of the environment is required to start from a random state (and this is typically something that we don't have).

$S_0, a_0, s_1, a_1, s_2, a_2, \ldots$
Random From π and p



- But, we need to keep exploring during the learning process!
- This leads to a key problem in RL: the **Exploration-Exploitation Dilemma**

if we want to learn an optimal policy sampling from data, we cannot behave optimally during the learning process (we have to go through non-optimal policies to state that one is optimal) *

* However, also if it is exploring every possible state and policy may not be convenient! If we have to train autonomous cars we may not want to crash the car just "for the sake of exploring". In the case of stock market, to learn how to earn money we should also lose them!

→ EXPLORATION vs. EXPLOITATION

(Exploration means that we search over the whole sample space, exploitation means that we exploit the promising areas)

Exploration: learning new things, avoid repeating, exposing to different actions

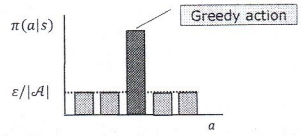
Exploitation: focus on something that is already good and minimize the new actions

ε-Greedy Exploration

- It is the simplest solution to the exploration-exploitation dilemma
- Instead of searching the optimal deterministic policy we search the optimal **ε-soft policy**, i.e., a policy that selects each action with a proability that is at least $\varepsilon/|\mathcal{A}|$ $(0 < \varepsilon \leq 1)$
- In particular we use **ε-greedy** policy:

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{|\mathcal{A}(s)|} + 1 - \epsilon & \text{if } a^* = arg\max_{a \in \mathcal{A}} Q(s,a) \\ \frac{\epsilon}{|\mathcal{A}(s)|} & \text{otherwise} \end{cases}$$

the idea is: we are focusing on the (current) optimal action, however we'll reserve a small chance also to the other actions



ε-Soft Monte Carlo Policy Iteration

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 $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow$ average$(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

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

before here we had the updating of a deterministic policy (deterministic greedy policy) (we updated the policy each time by selecting always the optimal action for each state). In this case instead we update the policy using the ε-greedy approach.

ε-Greedy Policy Improvement

- Thorem

  Any ε-greedy policy π' with respect to Q_π is an improvement over any ε-soft policy π

- Proof

$$\begin{aligned} V_\pi(s) &= Q_\pi(s, \pi'(s)) \\ &= \sum_{a \in \mathcal{A}} \pi'(a|s) Q_\pi(s,a) \\ &= \frac{\epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} Q_\pi(s,a) + (1-\epsilon) \max_{a \in \mathcal{A}} Q^\pi(s,a) \\ &\geq \frac{\epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} Q_\pi(s,a) + (1-\epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \frac{\epsilon}{|\mathcal{A}|}}{1-\epsilon} Q_\pi(s,a) \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) Q_\pi(s,a) = V_\pi(s) \end{aligned}$$

if we update the policy by making it greedy w.r.t. the current action value function, we'll obtain a new value function that is better or equal to the previous policy's (equal if the policy was already optimal)

→ we expect the algorithm to converge to the optimal ε-soft policy possible

Off-policy Learning

So far we always assumed that the policy that we are following is the policy that we are learning. Can we follow a policy different from the one we are learning? Can we learn the action value function of a policy different from the policy that we're following to generate the episodes?

→ Off-policy learning

Is it okay to learn a non-deterministic (ultimate) policy? Is it a good idea to learn something that has a probability >0 (even if bounded) to behave sub-optimally?

Yes! We determine a deterministic policy from the optimal ε-greedy policy ($\varepsilon \to 0$). (and we obtain a policy which is deterministic and which is **better** or equal that the ε-greedy one)

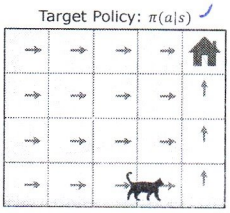
On-Policy vs Off-Policy Learning

- On-Policy Learning
  - The agent learns the value functions of the **same** policy used to select actions
  - Exploration/Exploitation Dilemma: we cannot easily learn an optimal deterministic policy
- Off-Policy Learning
  - The agent select action using a **behavior policy** $b(a|s)$
  - These data is used to learn the value functions of a **different target policy** $\pi(a|s)$
  - While $b(a|s)$ can be an explorative policy, the agent can learn an optimal deterministic policy $\pi^*(a|s)$

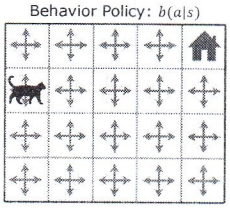
Target and Behavior Policy

the one we want to learn

Target Policy: $\pi(a|s)$ ✓



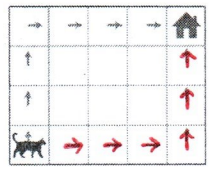
Behavior Policy: $b(a|s)$



Target and Behavior Policy (2)

- We can only learn a target policy that is covered by behavior policy:

$$\pi(a|s) > 0 \text{ where } b(a|s) > 0$$



- How is this possible? **Importance Sampling**

we cannot learn a policy that chooses an action with probability > 0 if we never experienced taking that action in that state

→ if we have no assumptions (and so any pair [action, state] can be the optimal one), the behavior policy needs to be at least ϵ-soft.

if we select the red policy as the behavior policy then we'll never be able to learn the grey one

Importance Sampling

- Importante sampling allow to estimate expectation of a distribution **different** w.r.t. the distribution used to draw the samples

$$\mathbb{E}_p[x] = \sum_{x \in X} x p(x) = \sum_{x \in X} x \frac{p(x)}{q(x)} q(x) = \sum_{x \in X} x \rho(x) q(x) = \mathbb{E}_q[x\rho(x)] \qquad \rho(x) = \frac{p(x)}{q(x)}$$

- Accordingly, we can use this for sample-based estimate:

$$\mathbb{E}_p[x] \approx \frac{1}{N} \sum_{i=1}^{N} x_i \text{ if } x_i \sim p(x) \quad \Rightarrow \quad \mathbb{E}_p[x] \approx \frac{1}{N} \sum_{i=1}^{N} x_i \rho(x_i) \text{ if } x_i \sim q(x)$$

suppose that sampling from $p(\cdot)$ is difficult but $\exists q(\cdot)$ from which sampling is easy.
sampling $x \sim p(\cdot)$ and compute $\mathbb{E}_p[x] = \sum x p(x)$ is equivalent to sampling $x \sim q(\cdot)$ and compute $\mathbb{E}_q\left[x \cdot \frac{p(x)}{q(x)}\right] = \sum x \frac{p(x)}{q(x)} q(x)$.

(**Notice**: $p(\cdot)$ is difficult in sampling, not necessarily in evaluation)

Importance Sampling for Policy Evaluation

- When following policy π we computed the state value function as:

$$V_\pi(s) \approx average(Returns[0], Returns[1], Returns[2], \dots)$$

returns generated following a policy π

- But when following policy b, the value function becomes:

$$V_\pi(s) \approx average(\rho_0 Returns[0], \rho_1 Returns[1], \rho_2 Returns[2], \dots)$$

- Where ρ_i is the probability of performing the trajectory observed in episode i while following policy π over the probability of observing the same trajectory while following policy b

$$\rho = \frac{Prob[\text{trajectory under } \pi]}{Prob[\text{trajectory under } b]}$$

Importance Sampling for Policy Evaluation (2)

for example:
suppose that we follow the behavior policy **b** and we observe:
$S_0, A_0, S_1, A_1, S_2, A_2, S_T$
and we observe the following returns: G_0 (in S_0), G_1 (in S_1) and G_2 (in S_2).
We need to correct all the returns to be able to evaluate V_π.
We correct the returns with ρ's:

$$\rho_0 = \frac{\pi(A_0|S_0)\,\pi(A_1|S_1)\,\pi(A_2|S_2)}{b(A_0|S_0)\,b(A_1|S_1)\,b(A_2|S_2)}$$

→ we correct G_0 with ρ_0: we add $\rho_0 \cdot G_0$ to the list of expected returns following policy π

Then we correct G_1 with:

$$\rho_1 = \frac{\pi(A_1|S_1)\,\pi(A_2|S_2)}{b(A_1|S_1)\,b(A_2|S_2)}$$

and so on.

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

probability of following some trajectory according to a given policy π

$\pi(A_k|S_k)$ = probability of choosing the action A_k in the state S_k, according to π

$p(S_{k+1}|S_k, A_k)$ = probability of going to S_{k+1} after choosing the action A_k in the state S_k

→ we don't know this, but we're lucky since it simplifies

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

the ratio (ρ) is not affected by the one step dynamics

Ordinary Sampling vs Weighted Sampling

- Ordinary sampling

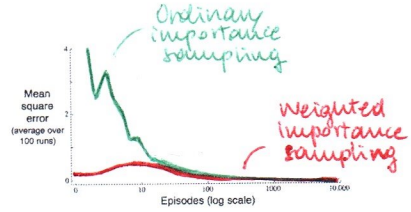
$$V_\pi(s) \approx \frac{\sum_i \rho[i] \cdot Return[i]}{N(s)}$$

  - Unbiased
  - Higher Variance

- Weighted sampling

$$V_\pi(s) \approx \frac{\sum_i \rho[i] \cdot Return[i]}{\sum_i \rho[i]}$$

  - Biased (bias converges to zero)
  - Lower Variance



we can observe a trajectory using policy **b** that is extremely unlikely to achieve with policy π (and the opposite)
→ it requires a lot of data to be sure that $V_\pi(s) \approx \frac{1}{N(s)} \sum \rho[i]\, Return[i]$

if we initially observe some trajectories that are very unlikely according to the target policy π, we won't weight this trajectory a lot (we weight more the trajectories that are more likely to π)

Off-Policy every visit MC prediction (ordinary sampling)

Input: a policy π to be evaluated
Initialize:
$V(s) \in \mathbb{R}$, **arbitrarily, for all** $s \in S$
$Returns(s) \leftarrow$ **an empty list, for all** $s \in S$
Loop forever (for each episode):
Generate an episode following $b: S_0, A_0, R_1, S_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0 \quad W \leftarrow 1$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$
$G \leftarrow \gamma W G + R_{t+1}$
Append G **to** $Returns(S_t)$
$V(S_t) \leftarrow$ **average**$(Returns(S_t))$
$W \leftarrow W \frac{\pi(A_t \mid S_t)}{b(A_t \mid S_t)}$