Reinforcement Learning III
Model-Free Control and Model-Based RL Approach

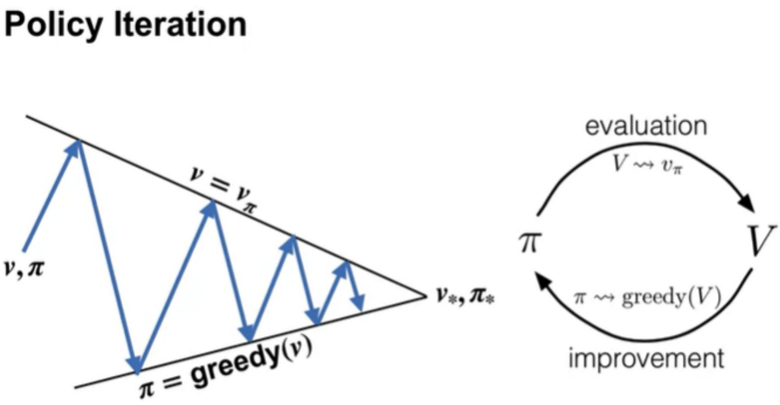
Antoine SYLVAIN

EPITA

2022-2023

Contents

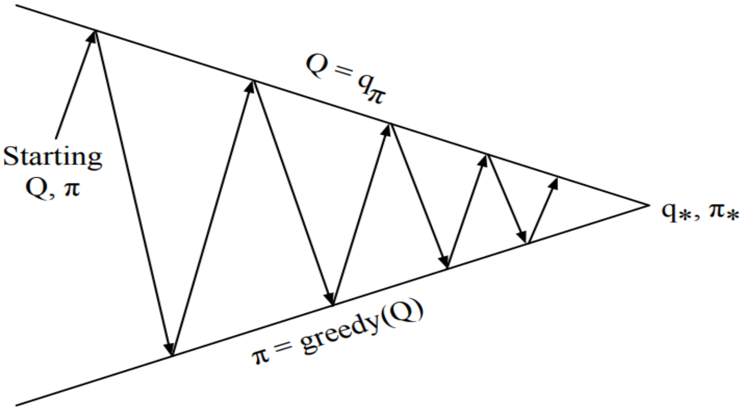
Policy Iteration (reminder)



Model-Free Policy Iteration using Action-Values

- Greedy policy improvement over $V(s)$ requires a model of the MDP
  $\pi' = argmax_{a \in \mathcal{A}} \mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s')$
- Greedy policy improvement over $Q(s,a)$ is model-free:
  $\pi'(s) = argmax_{a \in \mathcal{A}} Q(s,a)$

Policy Iteration with action-value function



Policy Iteration with action-value function

Monte Carlo Generalized Policy Iteration

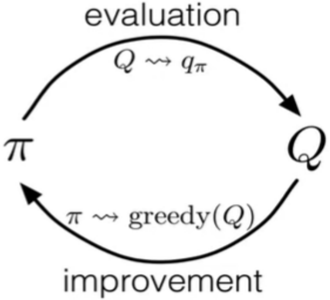
$$\pi_0 \rightarrow \pi_1 \rightarrow \pi_2 \rightarrow \cdots$$

Improvement:

$$\pi_{k+1}(s) \doteq \underset{a}{\operatorname{argmax}}\, q_{\pi_k}(s,a)$$

Evaluation:

Monte Carlo Prediction



Monte-Carlo ES Algorithm

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

Example of blackjack

Example of blackjack

S,A	Returns(S,A)	Q(S,A)		π(S)
		Hit	Stick	

Example of blackjack

S,A	Returns(S,A)	Q(S,A)		π(S)
		Hit	Stick	

Example of blackjack

S,A	Returns(S,A)	Q(S,A)		π(S)
		Hit	Stick	
X,Stick	Returns(X,Stick) = [1]	0	1	Stick

X = (NoAce, 20, 8)

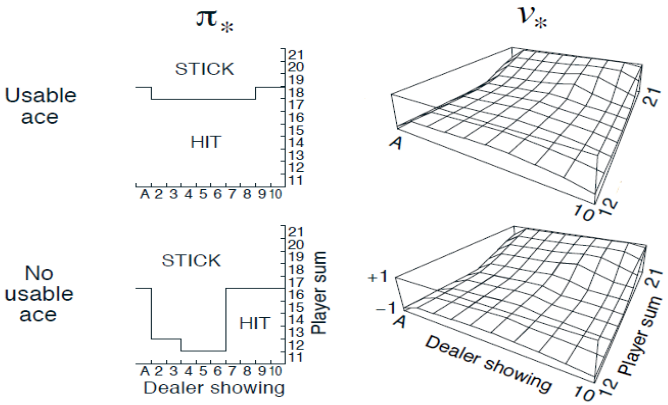
Example of blackjack

S,A	Returns(S,A)	Q(S,A)		π(S)
		Hit	Stick	
X,Stick	Returns(X,Stick) = [1]	0	1	Stick
Y,Hit	Returns(Y,Hit) = [1]	1	0	

X = (NoAce, 20, 8)

Y = (NoAce, 13, 8)

Example of blackjack



Limits of Exploring-Starts strategies

- We can not always chose the start
- We don't always to chose a new start
- How can we ensure exploration ?

Monte-Carlo ϵ-soft Algorithm

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

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\quad \pi \leftarrow$ an arbitrary ε-soft policy

$\quad Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$\quad Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

$\quad$ Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$\quad G \leftarrow 0$

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

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

$\quad\quad$ Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

$\quad\quad\quad$ Append G to $Returns(S_t, A_t)$

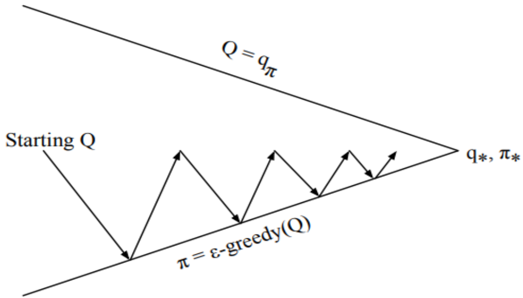
$\quad\quad\quad Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

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

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

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

MC Control



Every episode:

Policy evaluation Monte-Carlo policy evaluation, $Q \approx q_\pi$

Policy improvement ϵ-greedy policy improvement

Contents

Control with TD

- Natural idea: use TD instead of MC in our control loop
- Apply TD to $Q(S, A)$
- Use ϵ-greedy policy improvement
- Update every time-step

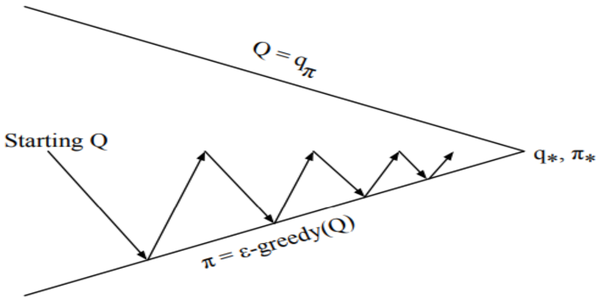
SARSA Algorithm

- SARSA
- $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$
- The agent chooses an action in the initial state to create the first state-action pair
- It observes the reward and next state
- Then it needs to know the next state-action pair to update its value estimates
- Since our agent is learning action values for a specific policy, it uses that policy to sample the next action:
  $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1} \sim \pi$

SARSA Algorithm

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

SARSA Improvement



Every time-step:

Policy evaluation Sarsa, $Q \approx q_\pi$

Policy improvement ϵ-greedy policy improvement

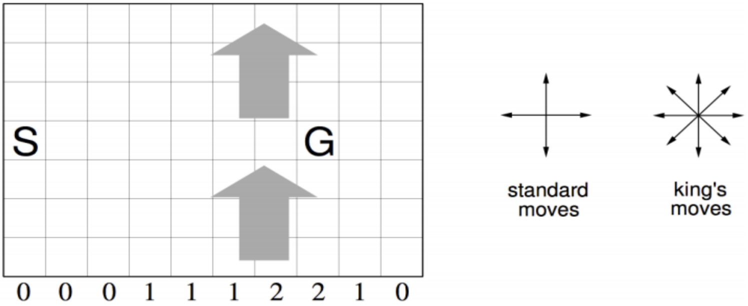
SARSA Algorithm

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

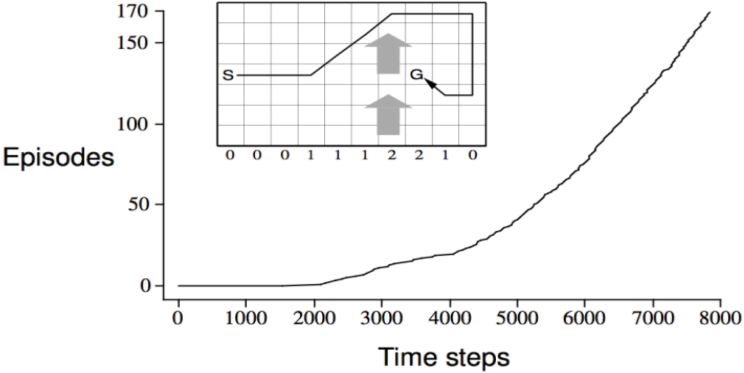
Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Windy Gridworld Example



- Reward = -1 per time-step until reaching goal
- Undiscounted

Windy Gridworld Example



Contents

On-Policy, Off-Policy

- **On-Policy**: improve and evaluate the policy being used to select actions
- **Off-Policy**: improve and evaluate a different policy from the one used to select actions

Off-Policy

- Learning Off-Policy, we use 2 policies:
  - **The target policy**: the policy we are learning, usually noted $\pi(a|s)$
  - **The behavior policy**: the one we use to select actions, usually noted $\mu(a|s)$
- Learn from observing other agents
- Re-use experience generated from old policies $\pi_1, \pi_2, ..., \pi t - 1$
- Learn about optimal policy while following exploratory policy
- Learn about multiple policies while following one policy

Importance Sampling

- Sample : $x \sim \mu$
- Estimate : $\mathbb{E}_{\pi}[x]$

Derivation of Importance Sampling

$$\mathbb{E}_\pi[X] \doteq \sum_{x \in X} x\pi(x)$$
$$\mathbb{E}_\pi[X] = \sum_{x \in X} x\pi(x)\frac{\mu(x)}{\mu(x)}$$
$$\mathbb{E}_\pi[X] = \sum_{x \in X} x\frac{\pi(x)}{\mu(x)}\mu(x)$$
$$\mathbb{E}_\pi[X] = \sum_{x \in X} x\rho(x)\mu(x)$$

$\rho(x) = \frac{\pi(x)}{\mu(x)}$ is called the importance sampling ratio

Derivation of Importance Sampling

$$\mathbb{E}_\pi[X] = \sum_{x \in X} x\rho(x)\mu(x)$$
$$\mathbb{E}_\pi[X] = \mathbb{E}_\mu[X\rho(X)]$$

$$\mathbb{E}_\mu[X\rho(X)] = \sum_{x \in X} x\rho(x)\mu(x)$$

Using a weighted sample average $\mathbb{E} \approx \frac{1}{n}\sum_{i=1}^{n} x_i)$

$$\mathbb{E}_\mu[X\rho(X)] \approx \sum_{i=1}^{n} x_i\rho(x_i) \approx \mathbb{E}_\pi[X]$$

$$x \sim \mu$$

Importance Sampling for Off-Policy MC

- Use returns generated from μ to evaluate π
- Weight return G_t according to similarity between policies
- Multiply importance sampling corrections along whole episode
  $G_t^{\pi/\mu} = \frac{\pi(A_t,S_t)}{\mu(A_t,S_t)} \frac{\pi(A_{t+1},S_{t+1})}{\mu(A_{t+1},S_{t+1})} \cdots \frac{\pi(A_\tau,S_\tau)}{\mu(A_\tau,S_\tau)}$
- Update value towards corrected return
  $V(S_t) \leftarrow V(S_t) + \alpha(G_t^{\pi/\mu} - V(S_t))$
- Cannot use if μ is zero when π is non-zero
- Importance sampling can dramatically increase variance

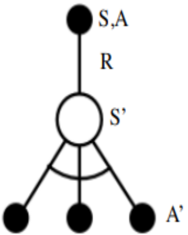
Importance Sampling for Off-Policy TD

- Use TD targets generated from μ to evaluate π
- Weight TD target $R + \gamma V(S')$ by importance sampling
- Only need a single importance sampling correction
  $V(S_t) \leftarrow V(S_t) + \alpha(\frac{\pi(A_t,S_t)}{\mu(A_t,S_t)}(R_{t+1} + \gamma V(S_{t+1})) - V(S_t))$
- Much lower variance than Monte-Carlo importance sampling
- Policies only need to be similar over a single step

Q-Learning

- We now consider off-policy learning of action-values $Q(s, a)$
- No importance sampling is required
- Next action is chosen using behaviour policy $A_{t+1} \sim \mu(.|S_t)$
- But we consider alternative successor action $A' \sim \pi(.|S_t)$
- And update $Q(S_t, A_t)$ towards value of alternative action
  $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t))$

Q-Learning Control



$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma \max_{a'} Q(S', a') - Q(S,A) \right)$$

Q-Learning Algorithm

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Q-Learning Algorithm

Initialized

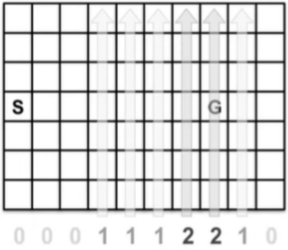
Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	327	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	499	0	0	0	0	0	0

Training

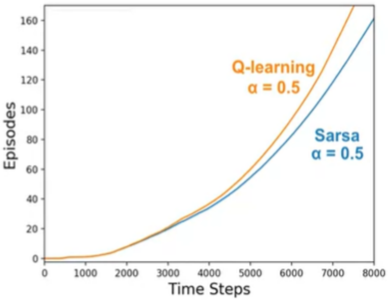
Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

Q-Learning in Windy Gridworld Example

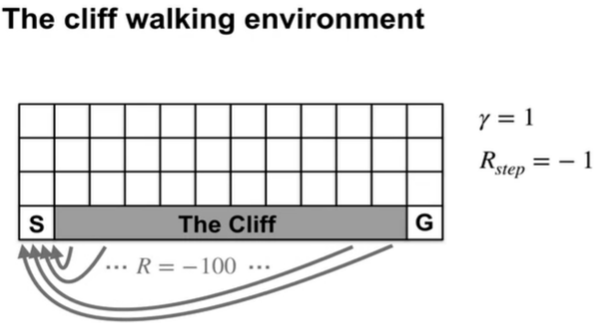
The Windy Gridworld



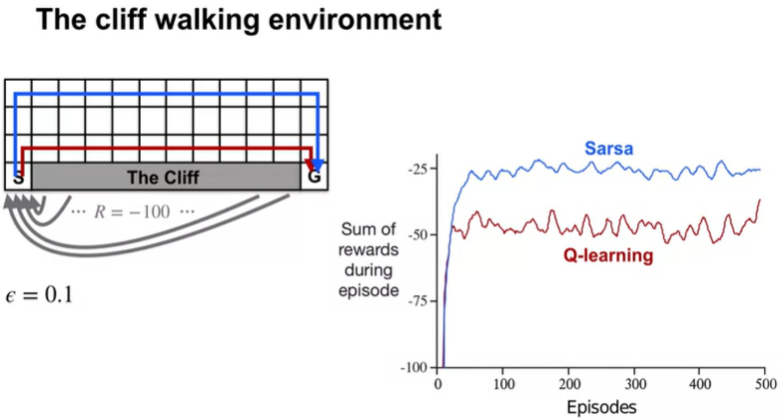
$\epsilon = 0.1$



A specific example: the Cliff environment



A specific example: the Cliff environment



Thank you for your attention