Reinforcement Learning

Lecture 5
Model Free Control

Stergios Christodoulidis

MICS Laboratory
CentraleSupélec
Université Paris-Saclay

https://stergioc.github.io/

Last Lecture

MC and TD

- Goal: learn v_π from episodes of experience under policy π
- Incremental Monte-Carlo
  - Update value $V(S_t)$ toward actual return G_t

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

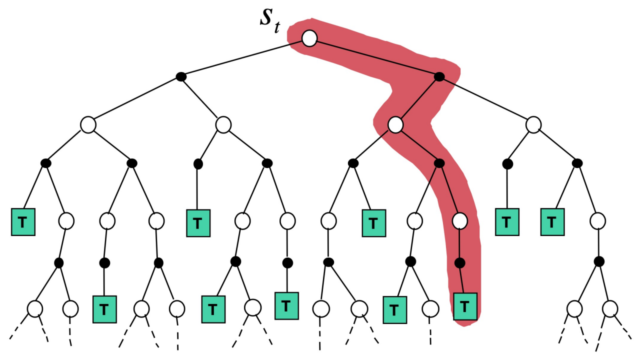
- Simplest temporal-difference learning algorithm: TD(0)
  - Update value $V(S_t)$ toward estimated return $R_{t+1} + \gamma V(S_{t+1})$

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

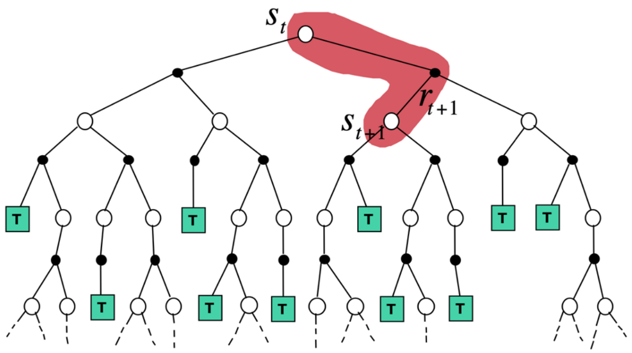
  - $R_{t+1} + \gamma V(S_{t+1})$ is called TD target
  - $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the TD error

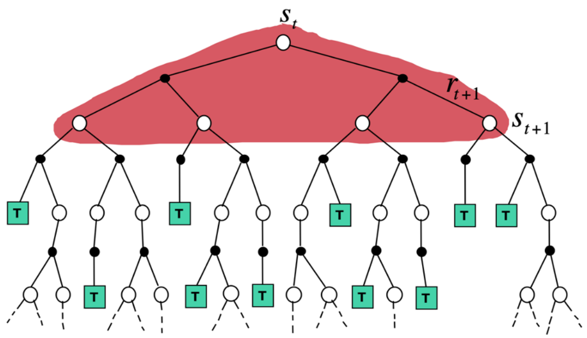
Monte-Carlo Update

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



$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

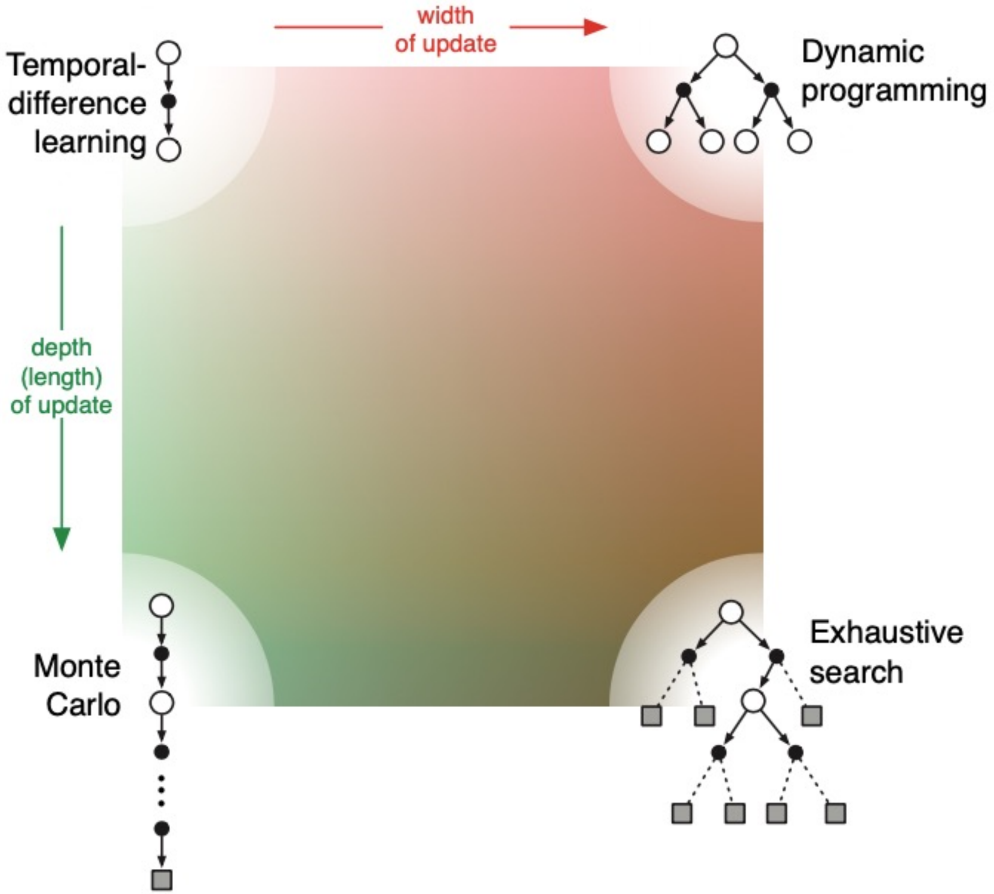




$$V(S_t) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V(S_{t+1})]$$

[David Silver, IRL, UCL 2015]

Unified View of Reinforcement Learning



[An Introduction to Reinforcement Learning, Sutton and Barto]

n-Step Return

- Consider the following n-step returns for n=1,2,…,∞
  - $n = 1$ (TD)
  $$G_{t:t+1} = R_{t+1} + \gamma V(S_{t+1})$$
  - $n = 2$
  $$G_{t:t+2} = R_{t+1} + \gamma R_{t+1} + \gamma^2 V(S_{t+2})$$
  - $n = \infty$ (MC)
  $$G_{t:t+\infty} = R_{t+1} + \gamma R_{t+1} + \ \ldots$$
- We can define the n-step return

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+1} + \cdots + \gamma^n V(S_{t+n})$$

- n-step temporal-difference learning

$$V(S_t) \leftarrow V(S_t) + \alpha(G_{t:t+n} - V(S_t))$$

Model-Free RL

- Last lecture:
  - Model-free prediction
  - Estimate the value function of an unknown MDP
- This lecture:
  - Model-free control
  - Optimize the value function of an unknown MDP

Today’s Lecture

Today's Lecture

- On-Policy vs Off-Policy
- On-Policy Monte-Carlo Control
- On-Policy Temporal Difference Control
  - Sarsa
  - Q-Learning
- Off-Policy Learning

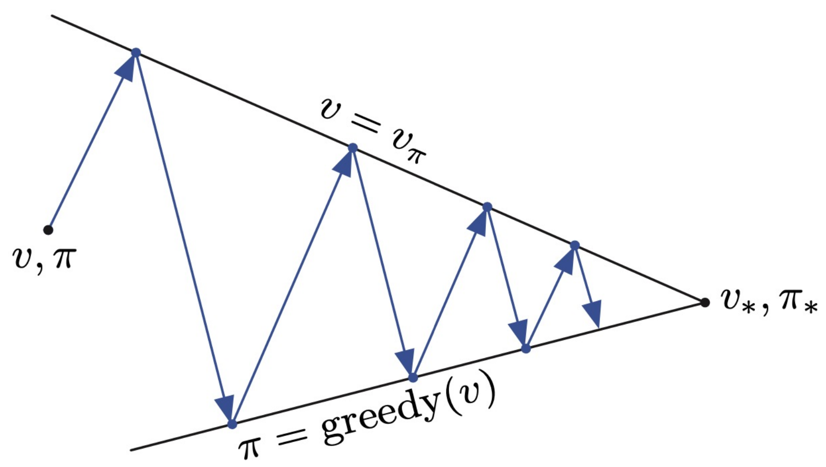
Uses of Model-Free Control

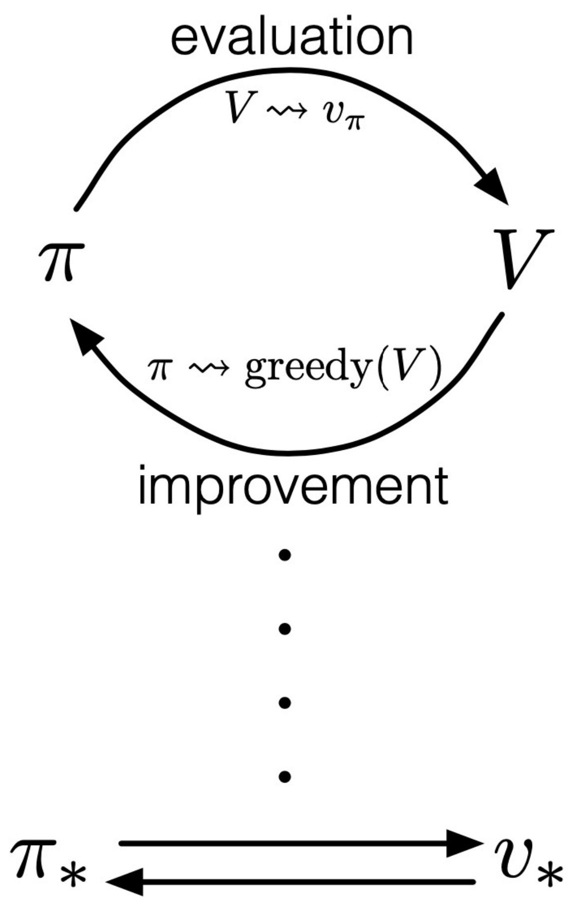
- Some example problems that can be modelled as MDPs
  - Ship Steering
  - Bioreactor
  - Helicopter
  - Portfolio management
  - Protein Folding Robot walking
  - Game of Go
- For most of these problems, either:
  - MDP model is unknown, but experience can be sampled
  - MDP model is known, but is too big to use, except by samples
- Model-free control can solve these problems

On and Off-Policy Learning

- On-policy learning
  - “Learn on the job”
  - Learn about policy π from experience sampled from π
- Off-policy learning
  - “Look over someone’s shoulder”
  - Learn about policy π from experience sampled from μ

Generalized Policy Iteration



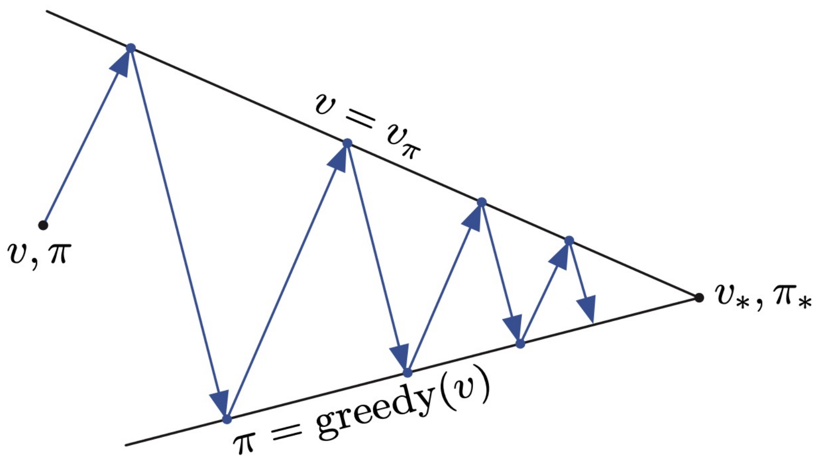


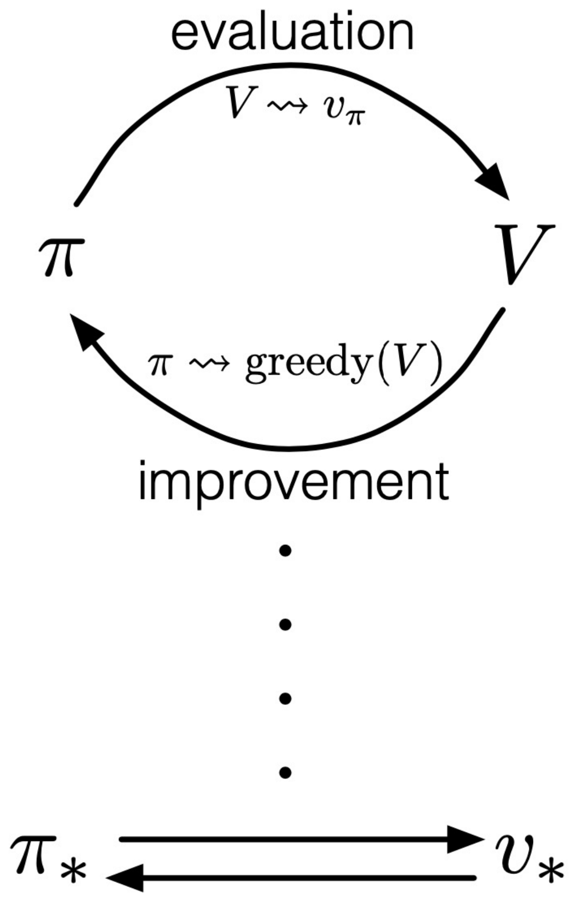
- Policy evaluation Estimate v_π
  - Any policy evaluation algorithm
- Policy improvement Generate $\pi' \geq \pi$
  - Any policy improvement algorithm

[An Introduction to Reinforcement Learning, Sutton and Barto]

Monte-Carlo Control

Generalized Policy Iteration for Monte-Carlo





- Policy evaluation
  - Monte-Carlo policy evaluation, $V = v_\pi$?
- Policy improvement
  - Greedy policy improvement ?

[An Introduction to Reinforcement Learning, Sutton and Barto]

Model-Free Policy Iteration Using Action-Value Function

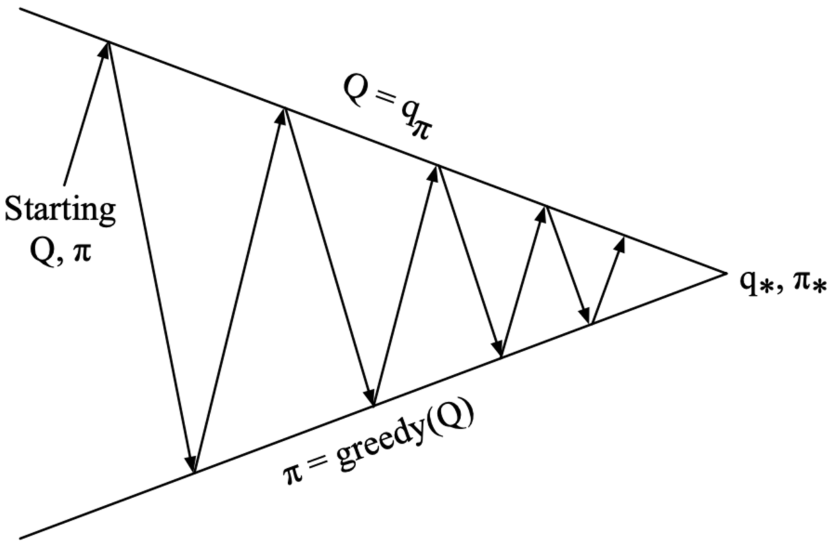
- Greedy policy improvement over $V(s)$ requires model of MDP

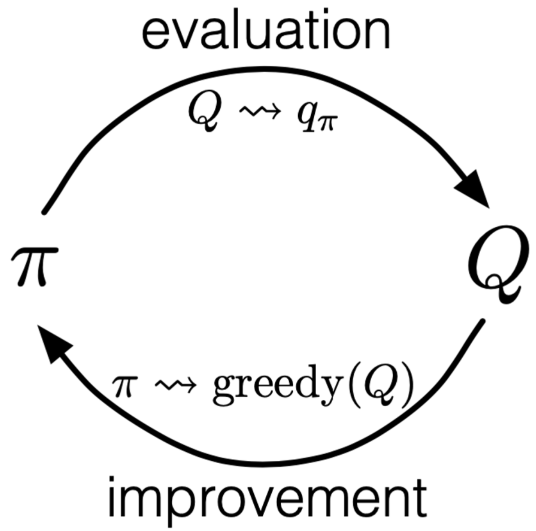
$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left[\mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s') \right]$$

- Greedy policy improvement over $Q(s, a)$ is model-free

$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q(s, a)$$

Generalized Policy Iteration with Action-Value Function





- Policy evaluation
  - <u>Monte-Carlo</u> policy evaluation, $Q = q_\pi$
- Policy improvement
  - <u>Greedy</u> policy improvement ?

[David Silver, IRL, UCL 2015]

[An Introduction to Reinforcement Learning, Sutton and Barto]

Example of Greedy Action Selection

- There are two doors in front of you.
- You open the left door and get reward 0
  - V(left) = 0
- You open the right door and get reward +1
  - V (right) = +1
- You open the right door and get reward +3
  - V (right) = +2
- You open the right door and get reward +2
  - V (right) = +2
- (…)
- Are you sure you've chosen the best door?

"Behind one door is tenure - behind the other is flipping burgers at McDonald's."

ε-Greedy Exploration

- Simplest idea for ensuring continual exploration
- All m actions are tried with non-zero probability
- With probability $1 \; - \; \varepsilon$ choose the greedy action
- With probability ε choose an action at random

$$\pi(\,a \mid s\,) = \begin{cases} {}^{\varepsilon}/_{m} + 1 - \varepsilon & , if \;\; a^{*} = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q(s, a) \\ {}^{\varepsilon}/_{m} & , otherwise \end{cases}$$

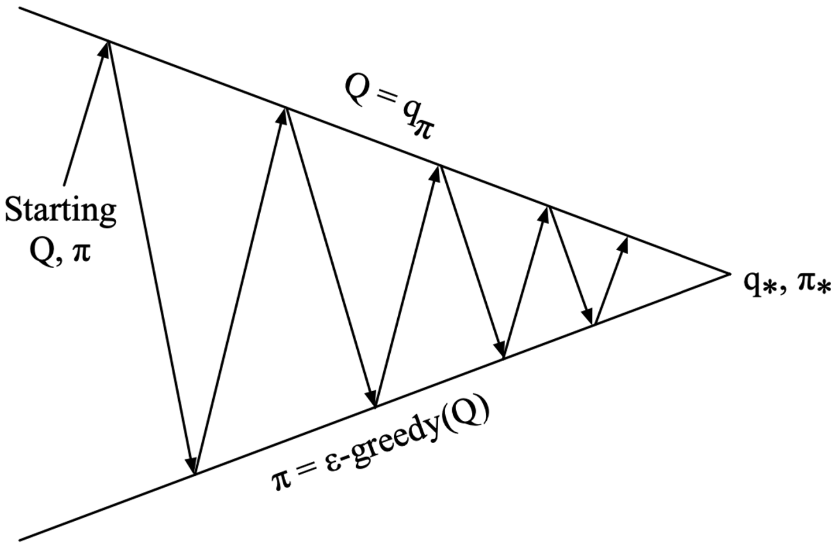
ε-Greedy Policy Improvement

Theorem

For any ε-greedy policy π, the ε-greedy policy π' with respect to q_π is an improvement, $\mathrm{v}_{\pi'}(s) \geq \mathrm{v}_\pi(s)$

$$
\begin{aligned}
q_\pi\big(s, \pi'(s)\big) &= \sum_{a \in \mathcal{A}} \pi'(a|s) q_\pi(s,a) \\
&= {}^{\varepsilon}/_{m} \sum_{a \in \mathcal{A}} q_\pi(s,a) + (1-\varepsilon) \max_{a \in \mathcal{A}} q_\pi(s,a) \\
&\geq {}^{\varepsilon}/_{m} \sum_{a \in \mathcal{A}} q_\pi(s,a) + (1-\varepsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - {}^{\varepsilon}/_{m}}{1-\varepsilon} q_\pi(s,a) \\
&= \sum_{a \in \mathcal{A}} \pi\,(a|s) q_\pi(s,a) = \mathrm{q}_\pi(s, \pi(s))
\end{aligned}
$$

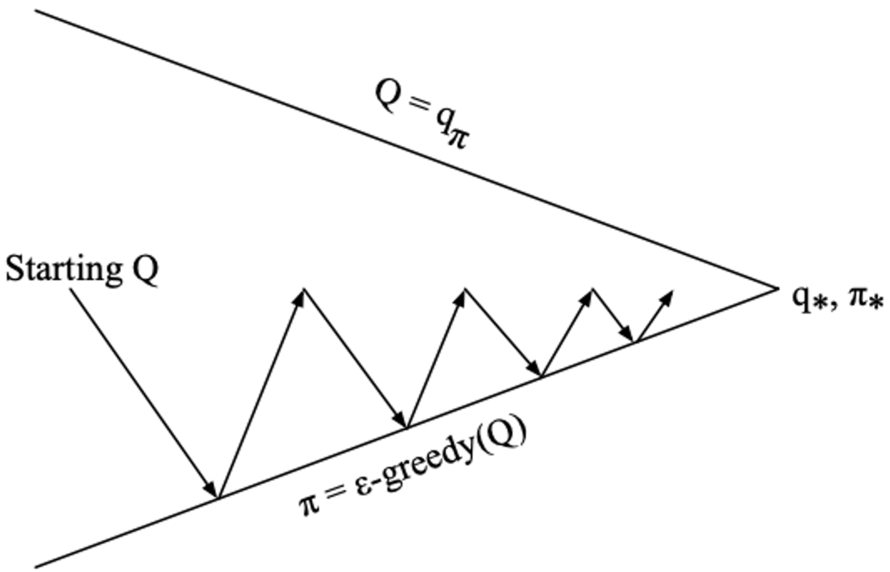
Monte-Carlo Policy Iteration



- Policy evaluation
  - Monte-Carlo policy evaluation, $Q = q_\pi$
- Policy improvement
  - ε-Greedy policy improvement

[David Silver, IRL, UCL 2015]

Monte-Carlo Control



Every episode:

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

GLIE

Definition

Greedy in the Limit with Infinite Exploration (GLIE)

- *All state-actions pairs are explored infinitely many times,*

$$\lim_{k\to\infty} N_k(s,a) = \infty$$

- The Policy converges on a greedy policy,

$$\lim_{k\to\infty} \pi_k(a|s) = \mathbf{1}(a = \underset{a'\in\mathcal{A}}{argmax}\, Q_k(s,a'))$$

- For example, ε-greedy is GLIE if ε reduces to zero, e.g., $\varepsilon_k = \frac{1}{k}$

GLIE Monte-Carlo Control

- Sample kth episode using π: $\{S_1, A_1, R_2, \dots, S_T\} \sim \pi$
- For each state S_t and action A_t in the episode,

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$$

- Improve policy based on new action-value function

$$\varepsilon \leftarrow {}^{1}/_{k}$$
$$\pi \leftarrow \varepsilon - greedy(Q)$$

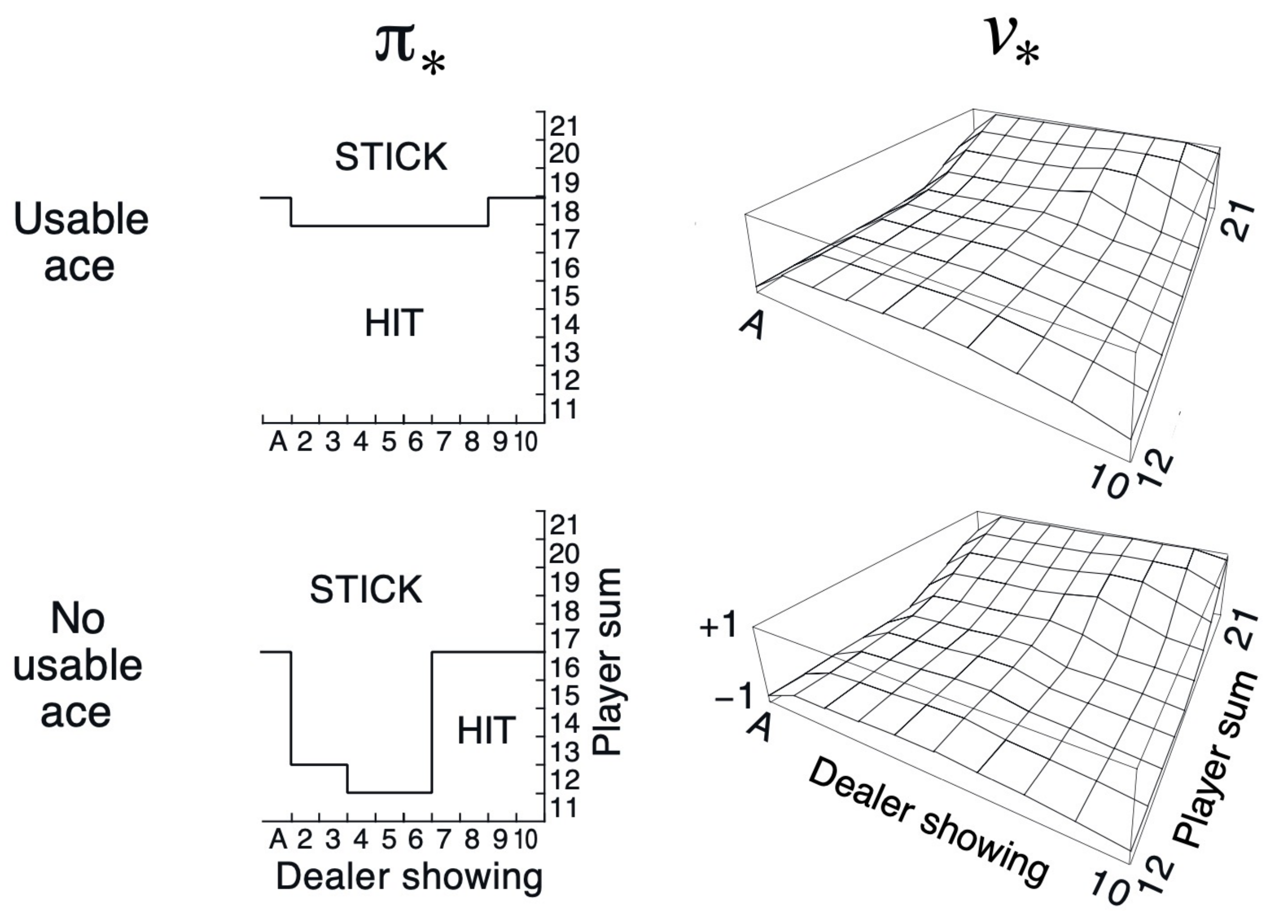
Theorem

GLIE Monte-Carlo control converges to the optimal action-value function, $Q(s, a) \rightarrow q_*(s, a)$

Example: Blackjack

- States (200 in total):
  - Current sum (12-21)
  - Dealer's showing card (ace or 2-10)
  - Do I have a "useable" ace? (yes-no)
- Actions
  - *hit*: Take another card (no replacement)
  - *stick*: Stop receiving cards (and terminate)
- Rewards
  - for stick:
    - +1 if sum of cards > sum of dealer cards
    - 0 if sum of cards = sum of dealer cards
    - -1 if sum of cards < sum of dealer cards
  - for hit:
    - -1 if sum of cards > 21 (and terminate)
    - 0 otherwise
- Policy: stick if sum of cards $\geq$ 20, otherwise hit

Example: Blackjack



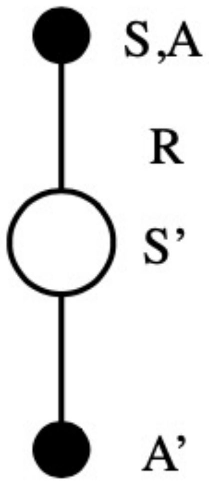
[An Introduction to Reinforcement Learning, Sutton and Barto]

Temporal-Difference Control

MC vs. TD Control

- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
  - Lower variance
  - Online
  - Incomplete sequences
- 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

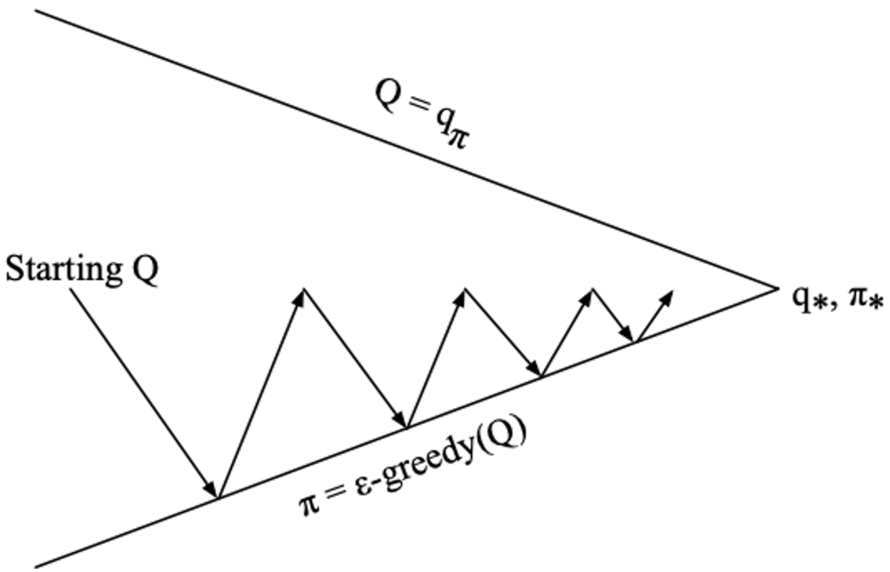
Updating Action-Value Functions with Sarsa



$$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A))$$

[David Silver, IRL, UCL 2015]

On-Policy Control with Sarsa



Every time-step:

- Policy evaluation
  - Sarsa policy evaluation, $Q \approx q_\pi$
- Policy improvement
  - ε-Greedy policy improvement

Sarsa Algorithm for On-Policy Control

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 $Q(terminal, \cdot) = 0$

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

[An Introduction to Reinforcement Learning, Sutton and Barto]

Convergence of Sarsa

Theorem

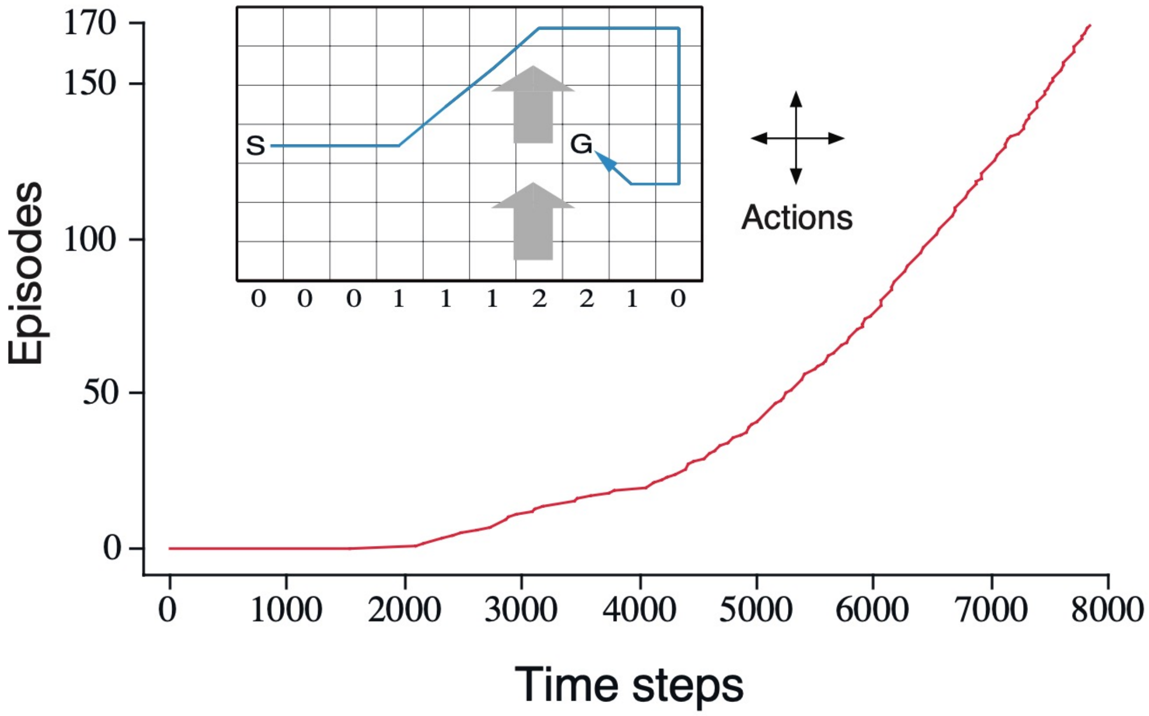
Sarsa converges to the optimal action-value function, $Q(s, a) \rightarrow q_{\pi}(s, a)$*, under the following conditions*

- *GLIE sequence of policies* $\pi_t(a|s)$*, i.e., make sure that the policy explores everything and converges to the greedy policy.*
- *Robbins-Monro sequence of step-sizes* α_t

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

$$\sum_{t=1}^{\infty} a_t < \infty$$

Example: Windy Gridworld



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

n-Step Sarsa

- Consider the following n-step returns for $n = 1,2, \dots, \infty$:
  - $n = 1$ (Sarsa)

$$G_{t:t+1} = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$$

  - $n = 2$

$$G_{t:t+1} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}, A_{t+2})$$

  - (...)
  - $n = \infty$ (MC)

$$G_{t:t+\infty} = R_{t+1} + \gamma R_{t+1} + \dots$$

- We can define the n-step Q-return

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+1} + \cdots + \gamma^n Q(S_{t+n}, A_{t+n})$$

- n-step Sarsa updates $Q(s, a)$ towards the n-step Q-Return

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(G_{t:t+n} - Q(S_t, A_t))$$

Off-Policy Learning

Off-Policy Learning

- Evaluate target policy $\pi(a|s)$ to compute $\mathrm{v}_{\pi}(s)$ or $\mathrm{q}_{\pi}(s,a)$
- While following behavior policy $\mu(a|s)$

$$\{\mathrm{S}_1, A_1, R_2, \ldots, S_T\} \sim \mu$$

- Why is this important?
  - Learn from observing humans or other agents
  - Re-use experience generated from old policies $\pi_1, \pi_2, \ldots, \pi_{t-1}$
  - Learn about optimal policy while following exploratory policy
  - Learn about multiple policies while following one policy

Importance Sampling

- A general technique to estimate the expectation of a distribution given a different distribution.
- i.e., We are weighting returns according to the relative probability of their trajectories occuring under the target and behavior policies.

$$
\begin{aligned}
\mathbb{E}_{X\sim P}[f(X)] &= \sum P(X)f(X) \\
&= \sum Q(X)\frac{P(X)}{Q(X)}f(X) \\
&= \mathbb{E}_{X\sim Q}\left[\frac{P(X)}{Q(X)}f(X)\right]
\end{aligned}
$$

Importance Sampling for Off-Policy Monte-Carlo Prediction

- Use returns generated from μ to evaluate π
- Weight return G_t according to similarity between policies
- Multiply importance sampling correction along whole episode

$$G_t^{\pi/\mu} = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \frac{\pi(A_{t+1}|S_{t+1})}{\mu(A_{t+1}|S_{t+1})} \dots \frac{\pi(A_T|S_T)}{\mu(A_T|S_T)} \mathrm{G}_t$$

- Update value towards corrected return

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t^{\pi/\mu} - V(S_t))$$

- Cannot use uf μ is zero when π is non-zero
- Importance sampling can dramatically increase variance

Importance Sampling for Off-Policy TD Prediction

- 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

$$V(S_t) \leftarrow V(S_t) + \alpha\left(\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}(R_{t+1} + \gamma V(S_{t+1})) - V(S_t)\right)$$

- Much lower variance than Monte-Carlo importance sampling
- Policies only need to be similar over a single step

Q-Learning for Action-Value Prediction

- We now consider off-policy learning of action-values $Q(s, a)$
- No importance sampling is required
- Next action is chosen using behavior policy $A_{t+1} \sim \mu(\cdot \mid S_t)$ a reward R_{t+1} is also observed.
- But we consider alternative successor action $A' \sim \pi(\cdot \mid S_t)$
- And update $Q(S_t, A_t)$ towards value of alternative action

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t))$$

Off-Policy Control with Q-Learning

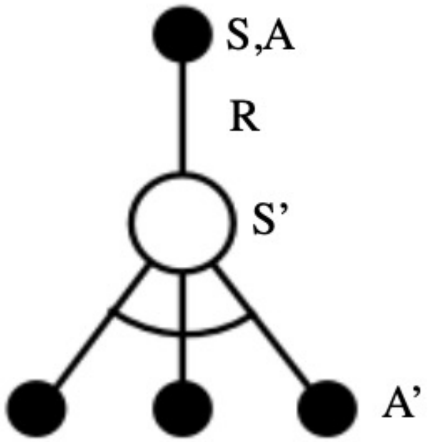
- We now allow both behavior and target policies to improve
- The target policy π is greedy w.r.t. $Q(s, a)$

$$\pi(\mathrm{S}_{t+1}) = \underset{a'}{\operatorname{argmax}}\, Q(S_{t+1}, a')$$

- The behavior policy μ is e.g., ε-greedy w.r.t. $Q(s, a)$
- The Q-learning target then simplifies:

$$\begin{aligned} & R_{t+1} + \gamma Q(S_{t+1}, A') \\ &= R_{t+1} + \gamma Q\left(S_{t+1}, \underset{a'}{\operatorname{argmax}}\, Q(S_{t+1}, a')\right) \\ &= R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') \end{aligned}$$

Q-Learning Control Algorithm



$$Q(S, A) \leftarrow Q(S, A) + \alpha\left(R_{t+1} + \gamma \max_{a'} Q(S', a') - Q(S, A)\right)$$

[David Silver, IRL, UCL 2015]

Q-Learning Algorithm for Off-Policy Control

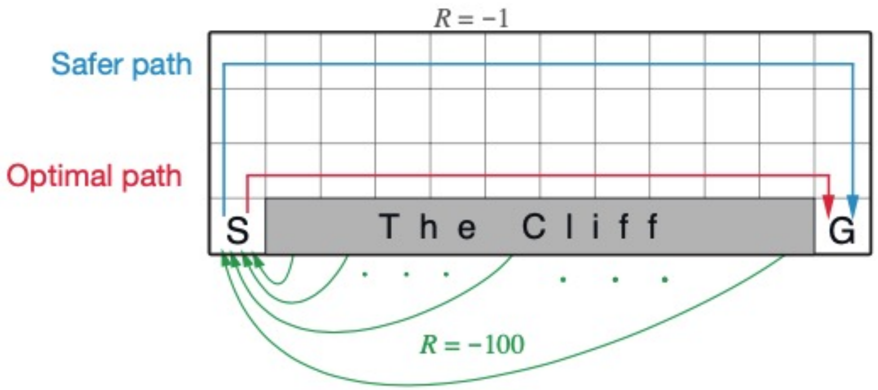
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

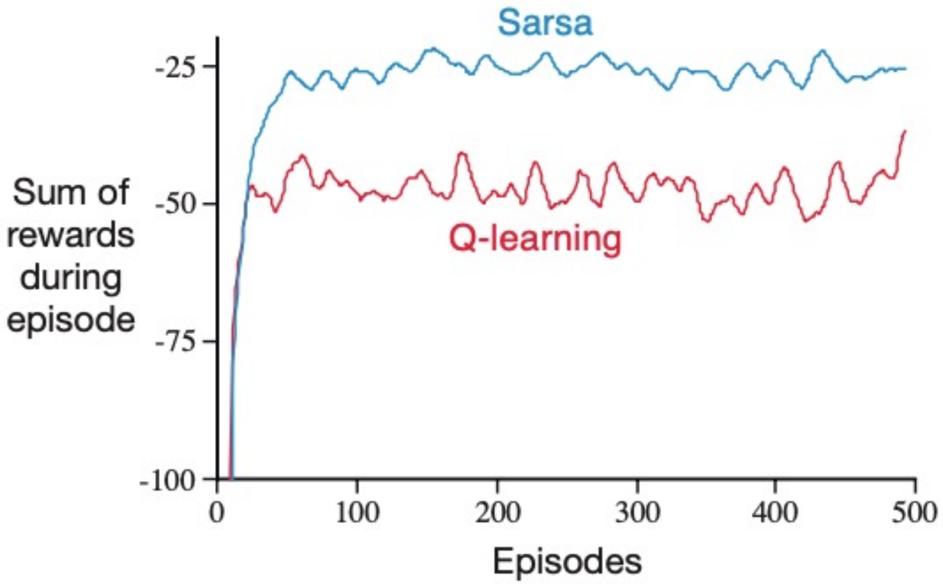
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

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

[An Introduction to Reinforcement Learning, Sutton and Barto]

Example: Cliff Walking





Summary

- We discussed methods that can be used for Model-Free control.
- We used as a basis the generalised policy iteration idea and we tried different methods for policy evaluation and policy improvement.
- We discussed about MC Control, Sarsa (TD).
- We introduced Off-Policy Q-Learning Algorithm.