Model-Free Control

CMPS 4660/6660: Reinforcement Learning

Acknowledgement: slides adapted from David Silver's RL course

Model-Free Reinforcement Learning

- Model-free prediction
  - Estimate the value function of an unknown MDP
- Model-free control
  - Optimize the value function of an unknown MDP

Agenda

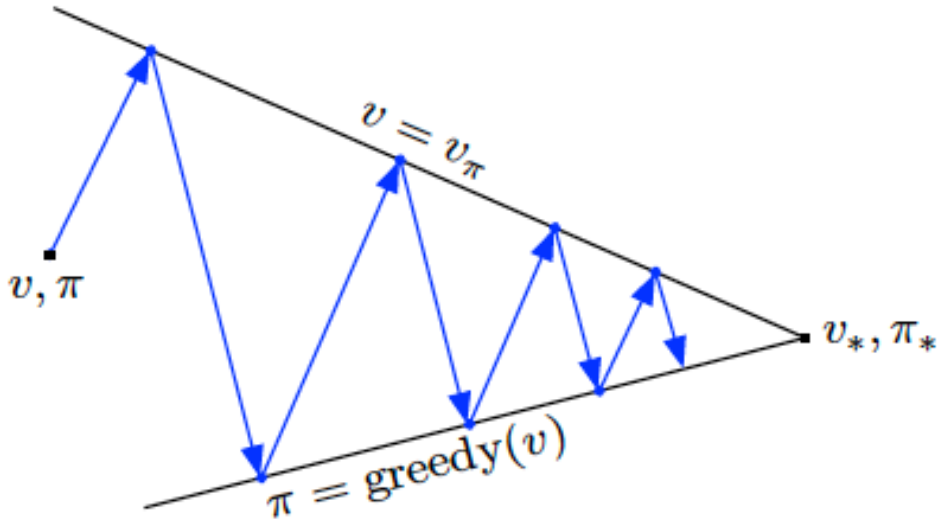
- On-Policy Monte-Carlo Control
- On-Policy Temporal-Difference Learning
- Off-Policy Learning

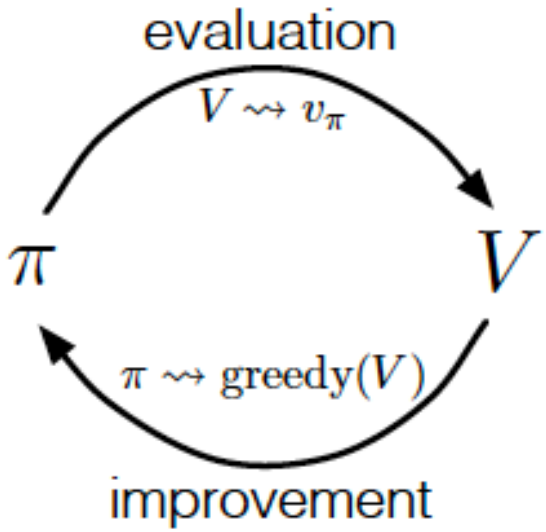
On and Off-Policy Learning

- On-policy learning
  - "Learn on the job"
  - Learn about policy π from experience sampled from π

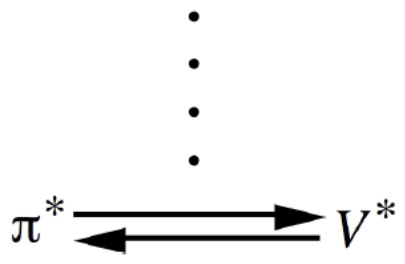
- Off-policy learning
  - “Look over someone's shoulder"
  - Learn about policy π from experience sampled from μ

Generalized Policy Iteration (Refresher)

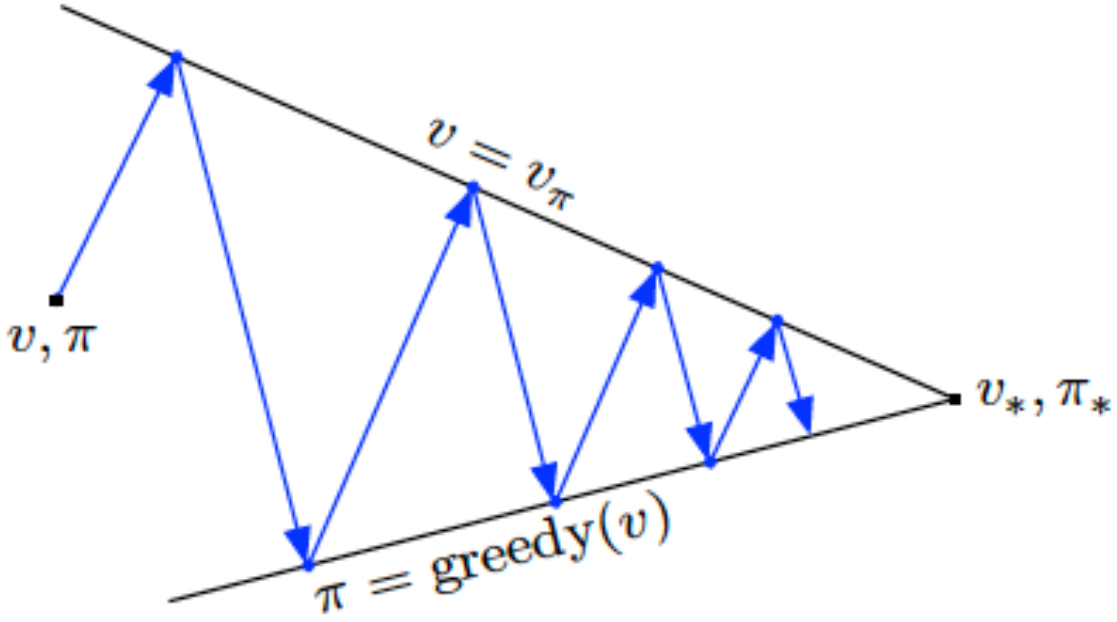




- Policy evaluation estimates v_π
  - e.g. Iterative policy evaluation
- Policy improvement generates π' s.t. $v_{\pi'} \geq v_\pi$
  - e.g. Greedy policy improvement



Generalized Policy Iteration With Monte-Carlo Evaluation



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

Policy improvement Greedy policy improvement?

Model-Free Policy Iteration Using Action-Value Function

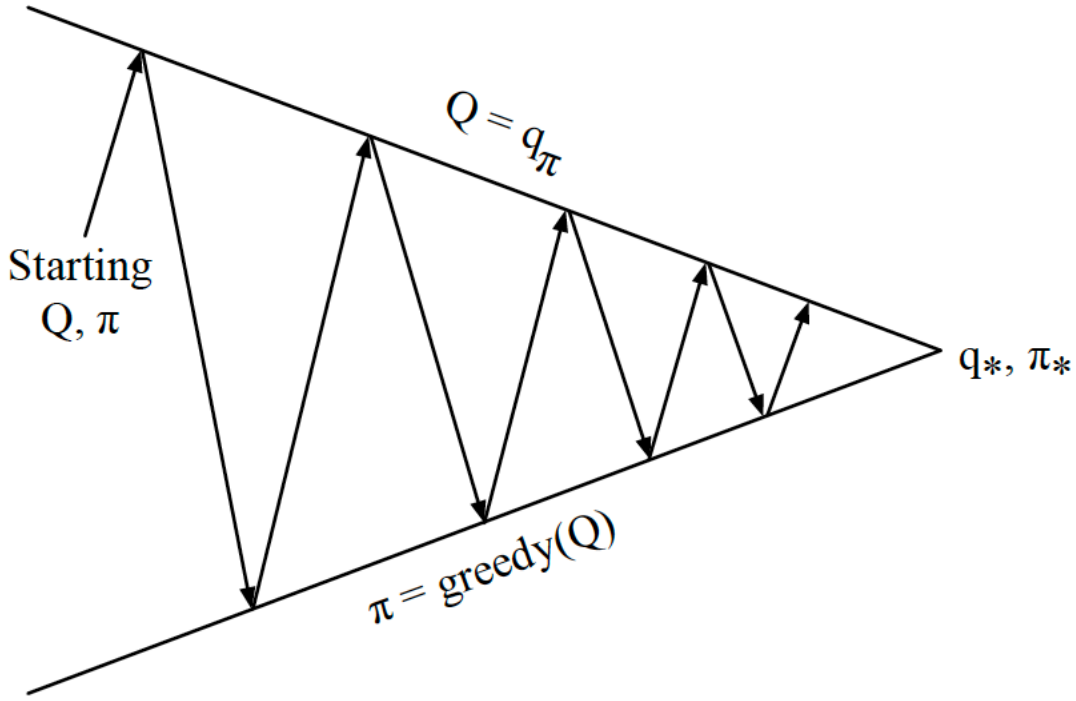
- Greedy policy improvement over $V_\pi(s)$ requires model of MDP

$$\pi'(s) = \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}} \left[r(s,a) + \gamma \sum_{s'} P_{ss'}(a) V_\pi(s') \right]$$

- Greedy policy improvement over $Q_\pi(s,a)$ is model-free

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

Generalized Policy Iteration with Action-Value Function



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

Policy improvement Greedy policy improvement?

Monte-Carlo Policy Evaluation

- Sample *k*th episode using policy π:

$$S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1,} R_T \sim \pi$$

- For each state S_t and action A_t in the episode:

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

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

Example of Greedy Action Selection

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

- 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) = +3
- You open the right door and get reward +2

  V(right) = +2
- ...
- Are you sure you've chosen the best door?

Exploration vs. Exploitation Tradeoff

- Exploration: Choose an action with more information
- Exploitation: Choose an action with more reward

- In MDP, the agent has complete information of what he is going to get for different actions from the Markov model.

- In RL, such information is inaccurate due to finite experience.

- One solution: ϵ-Greedy

ϵ-Greedy Exploration

- Simplest idea for ensuring continual exploration
- All $m = |\mathcal{A}(s)|$ actions are tried with non-zero probability
- With probability $1 - \epsilon$ choose the greedy action
- With probability ϵ choose an action at random

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

ϵ-Greedy Policy Improvement

Theorem

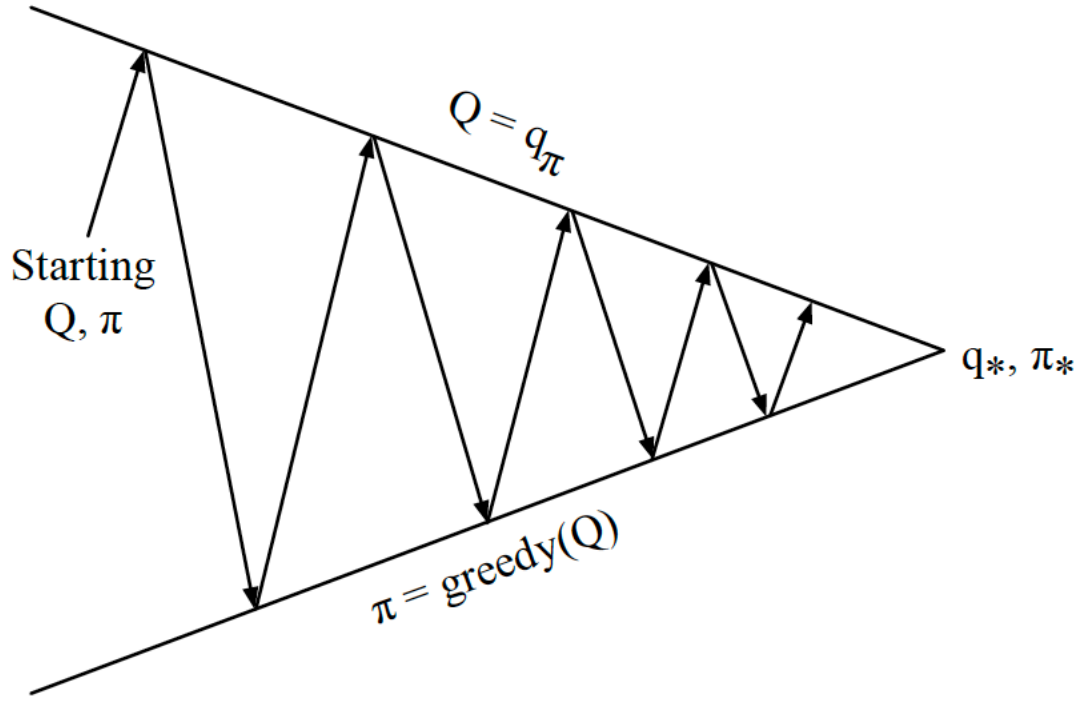
For any ϵ-greedy policy π, the ϵ-greedy policy π' with respect to q_π is an improvement, $v_{\pi'}(s) \geq v_\pi(s)$

$$
\begin{aligned}
q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) q_\pi(s, a) \\
&= \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \max_{a \in \mathcal{A}} q_\pi(s, a) \\
&\geq \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \epsilon/m}{1 - \epsilon} q_\pi(s, a) \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a) = v_\pi(s)
\end{aligned}
$$

$\pi(a|s) \geq \epsilon/m, \forall a$

Therefore from policy improvement theorem, $v_{\pi'}(s) \geq v_\pi(s)$

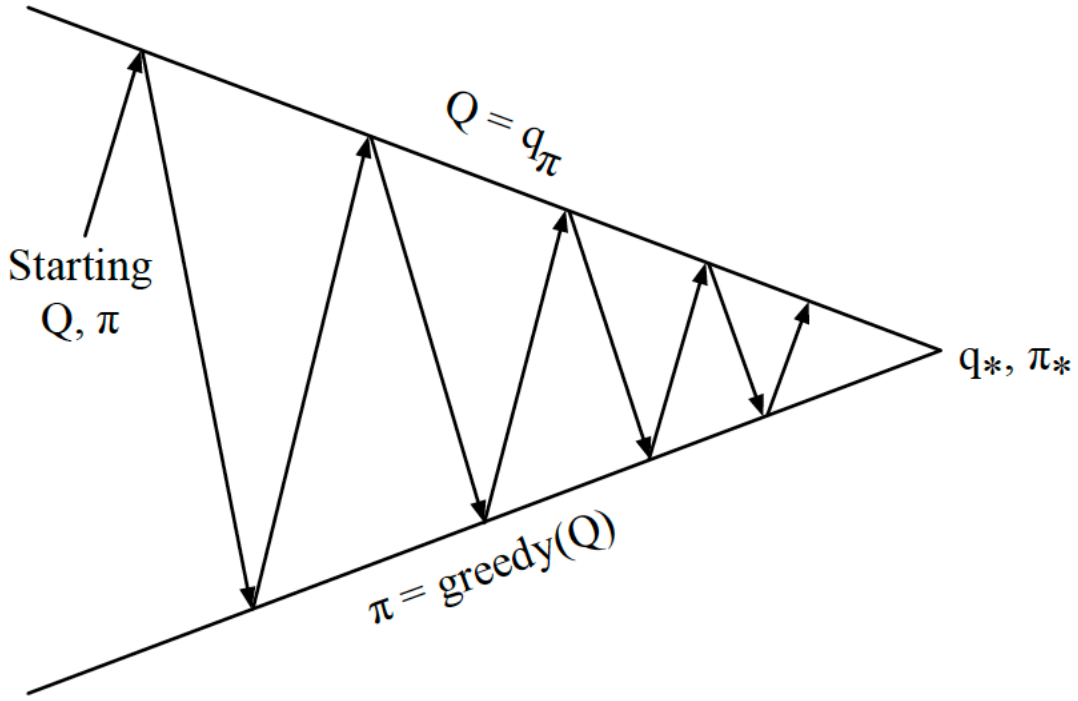
Monte-Carlo Policy Iteration



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

Policy improvement ϵ-greedy policy improvement

Monte-Carlo Policy Iteration



Every episode:

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

Policy improvement ϵ-greedy policy improvement

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 $Returns(S_t, A_t)$

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

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

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

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

GLIE

Definition

Greedy in the Limit with Infinite Exploration (GLIE)

- All state-action pairs are explored infinitely many times,

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

- The policy converges on a greedy policy,

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

- For example, ϵ-greedy is GLIE if ϵ reduces to zero at $\epsilon_k = \frac{1}{k}$

GLIE Monte-Carlo Control

- Sample *k*th episode using policy π:

$$S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T \sim \pi$$

- For each state S_t and action A_t in the episode:

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

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

- Improve policy based on new action-value function

$$\epsilon \leftarrow 1/k$$

$$\pi \leftarrow \epsilon\text{-greedy}\,(Q)$$

Theorem

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

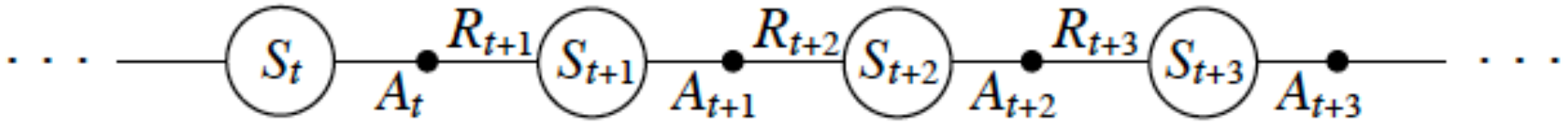
MC vs. TD Control

- 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

Agenda

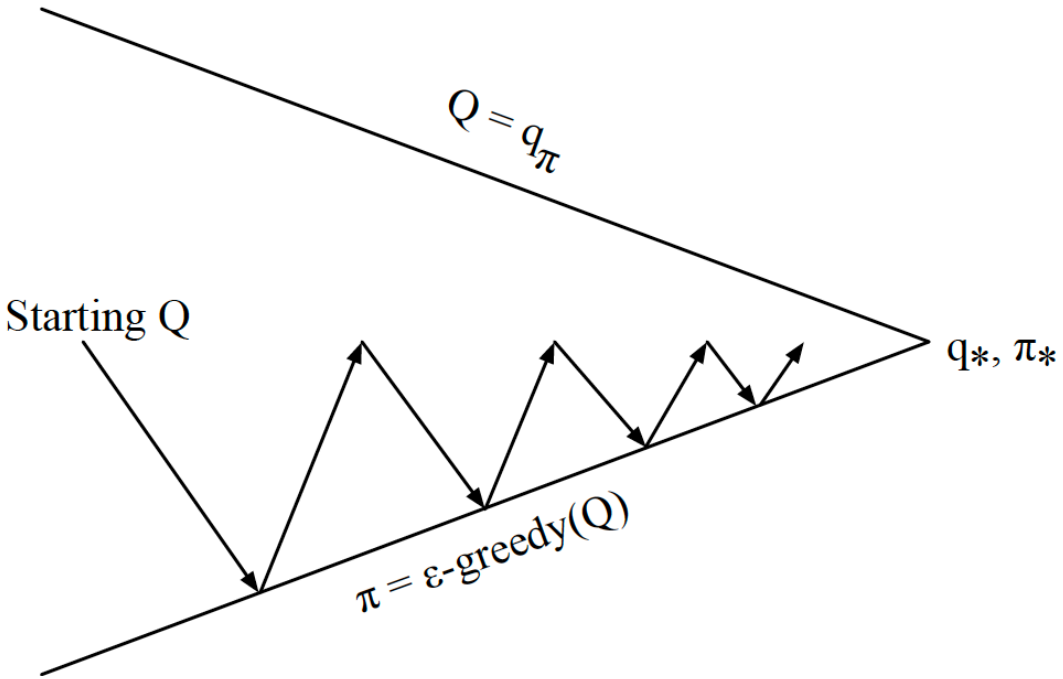
- On-Policy Monte-Carlo Control
- On-Policy Temporal-Difference Learning
- Off-Policy Learning

Updating Action-Value Functions with Sarsa



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

On-Policy Control With Sarsa



Every time-step:

- Policy evaluation Sarsa, $Q \approx q_\pi$
- 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 $Q(\mathit{terminal}, \cdot) = 0$

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

Convergence of Sarsa

Theorem

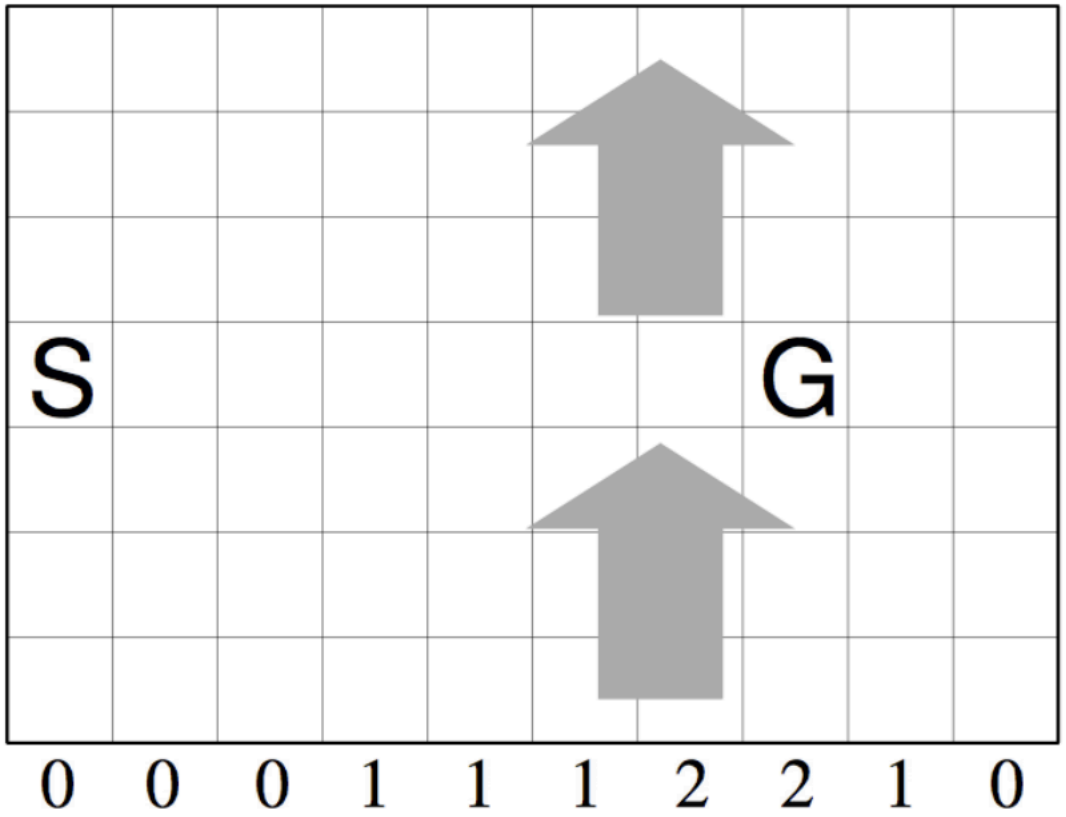
Sarsa converges to the optimal action-value function, $Q(s,a) \rightarrow q_*(s,a)$, under the following conditions:

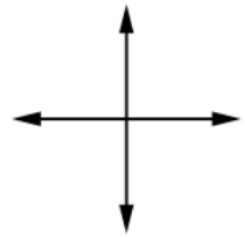
- *GLIE sequence of policies $\pi_t(a|s)$*
- *Robbins-Monro sequence of step-sizes α_t*

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

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

Windy Gridworld Example

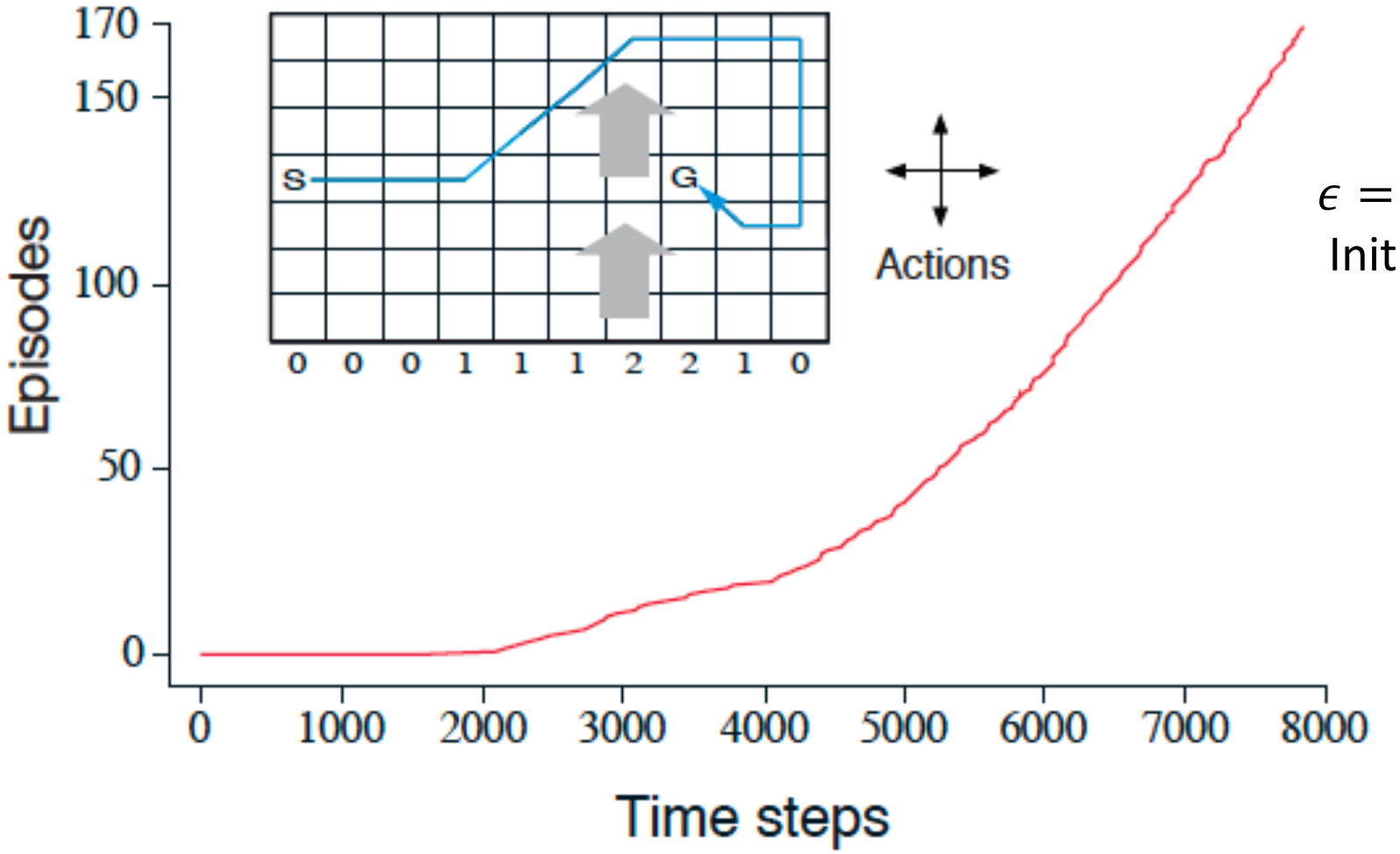




standard moves

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

Sarsa on the Windy Gridworld



$\epsilon = 0.1, \alpha = 0.5$

Initially $Q(s, a) = 0$

Expressions of Action-Value Function

- Conditional expectation of return:

$$q_\pi(s,a) = \mathbb{E}_\pi\left(\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_0 = s, A_0 = a\right)$$

- Bellman Equation:

$$q_\pi(s,a) = \mathbb{E}_\pi\left(R_{t+1} + \gamma\, q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a\right)$$
$$q_\pi(s,a) = \mathbb{E}_\pi\left(R_{t+1} + \gamma R_{t+2} + \gamma^2 q_\pi(S_{t+2}, A_{t+2}) | S_t = s, A_t = a\right)$$
$$q_\pi(s,a) = \mathbb{E}_\pi\left(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 q_\pi(S_{t+3}, A_{t+3}) | S_t = s, A_t = a\right)$$

…

n-Step Sarsa

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

$$
\begin{aligned}
n = 1 \quad (\textit{Sarsa}) \qquad & q_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) \\
n = 2 \qquad & q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}, A_{t+2}) \\
\vdots \qquad & \qquad \vdots \\
n = \infty \quad (\textit{MC}) \qquad & q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-t-1} R_T
\end{aligned}
$$

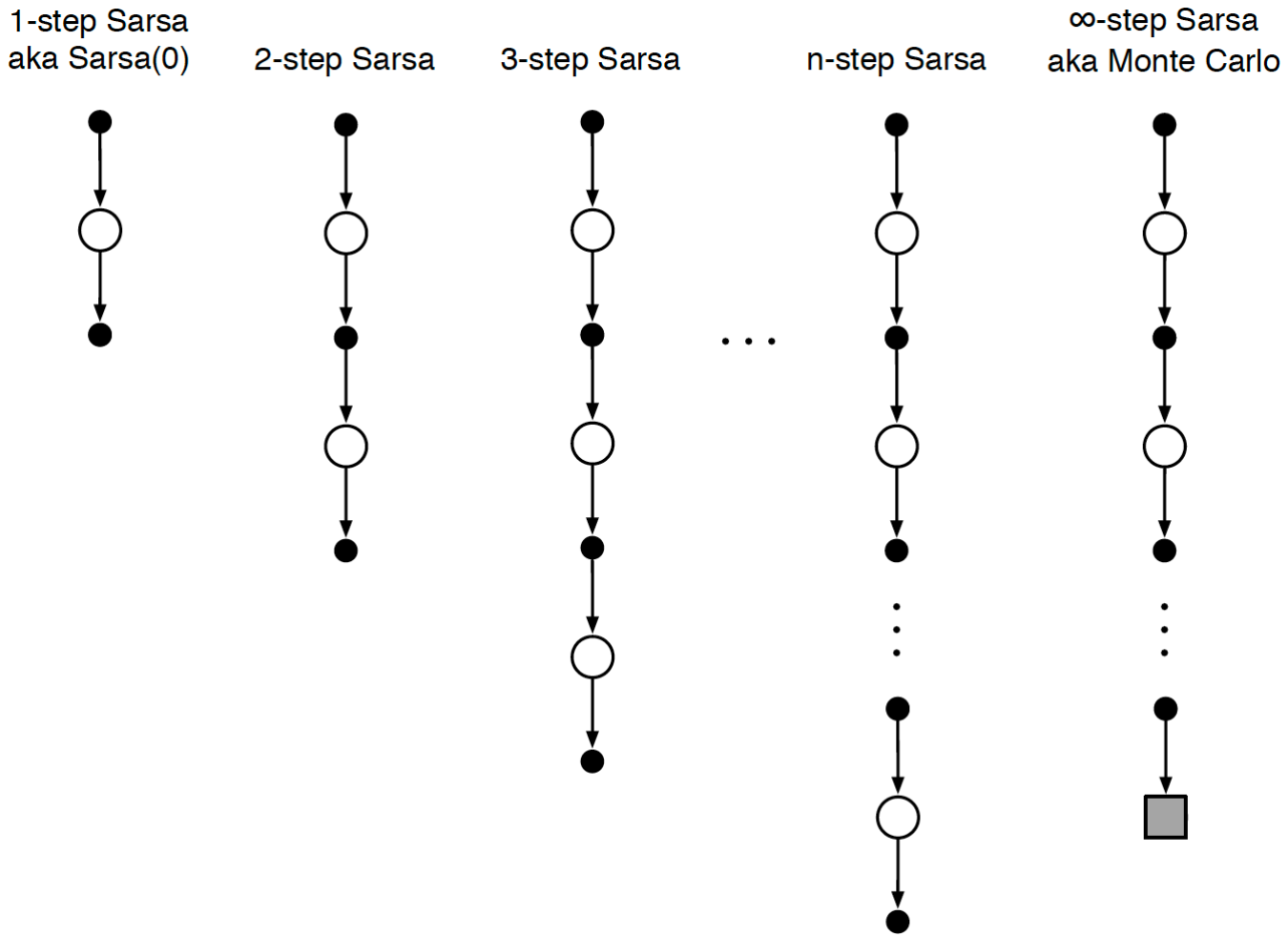
- Define the n-step Q-return

$$q_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n Q(S_{t+n}, A_{t+n})$$

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

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(q_t^{(n)} - Q(S_t, A_t) \right)$$

Sarsa(λ): Forward View



- The q^λ *return* combines all n-step Q-returns $q_t^{(n)}$
- Using weight $(1-\lambda)\lambda^{n-1}$

$$q_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}q_t^{(n)}$$

- Forward-view Sarsa(λ)

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha\left(q_t^\lambda - Q(S_t,A_t)\right)$$

Sarsa(λ): Backward View

- Just like TD(λ), we use eligibility traces in an online algorithm
- But Sarsa(λ) has one eligibility trace for each state-action pair

$$E_{-1}(s,a) = 0$$

$$E_t(s,a) = \gamma\lambda E_{t-1}(s,a) + \mathbf{1}(S_t = s, A_t = a)$$

- $Q(s,a)$ is updated for every state s and action a
- In proportion to TD-error δ_t and eligibility trace $E_t(s,a)$

$$\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$

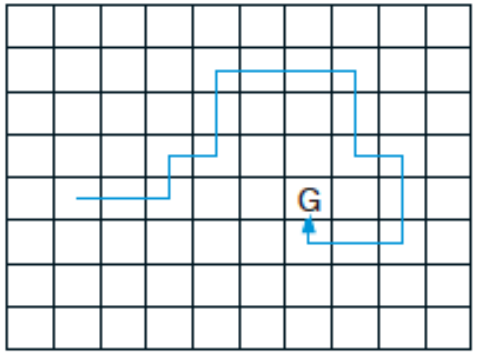
$$Q(s,a) \leftarrow Q(s,a) + \alpha\delta_t E_t(s,a), \forall s \in \mathcal{S}, a \in A$$

Sarsa(λ) Algorithm

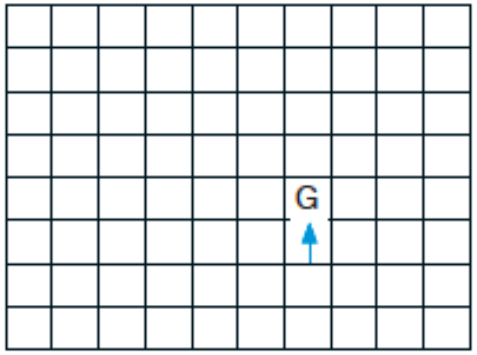
Initialize $Q(s, a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
Repeat (for each episode):
    $E(s, a) = 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
    Initialize S, A
    Repeat (for each step of episode):
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ε-greedy)
        $\delta \leftarrow R + \gamma Q(S', A') - Q(S, A)$
        $E(S, A) \leftarrow E(S, A) + 1$
        For all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:
            $Q(s, a) \leftarrow Q(s, a) + \alpha \delta E(s, a)$
            $E(s, a) \leftarrow \gamma \lambda E(s, a)$
        $S \leftarrow S'$; $A \leftarrow A'$
    until S is terminal

Sarsa(λ) Gridworld Example

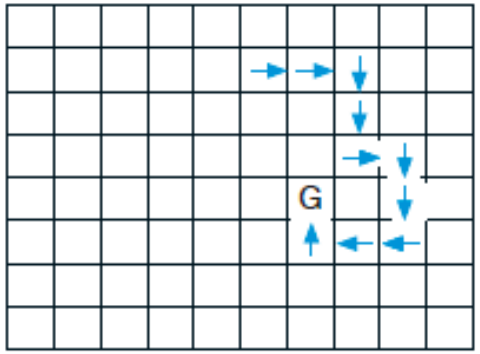
Path taken



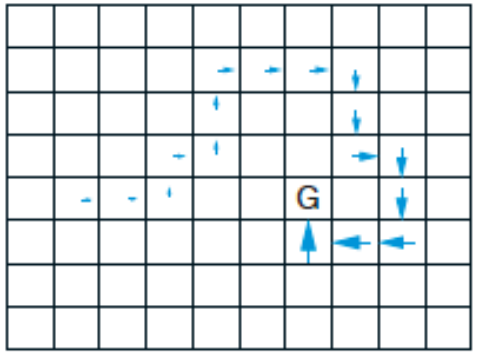
Action values increased by one-step Sarsa



Action values increased by 10-step Sarsa



Action values increased by Sarsa(λ) with λ=0.9



Agenda

- On-Policy Monte-Carlo Control
- On-Policy Temporal-Difference Learning
- Off-Policy Learning

Off-Policy Learning

- Evaluate target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s,a)$
- While following behavior policy $\mu(a|s)$

$$S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1,} R_T \sim \mu$$

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

Importance Sampling

- Estimate the expectation of a different distribution

$$E_{X\sim P}[f(X)] = \sum P(X) f(X)$$
$$= \sum Q(X) \frac{P(X)}{Q(X)} f(X)$$
$$= E_{X\sim Q}\left[\frac{P(X)}{Q(X)} f(X)\right]$$

Importance Sampling for Off-Policy Monte-Carlo

- Recall constant−α MC: $V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$
- Use returns generated from μ to evaluate π
- Weight return G_t according to similarity between policies

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T$$

$$v_\pi(s) = E_\pi\left(G_t | S_t = s\right)$$

$$= E_{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \sim \pi}[f(A_t, S_{t+1}, A_{t+1}, \ldots, S_T) | S_t = s]$$

$$= E_{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \sim \mu}[\rho_{t:T-1} f(A_t, S_{t+1}, A_{t+1}, \ldots, S_T) | S_t = s]$$

Importance Sampling for Off-Policy Monte-Carlo

$$\Pr_{\pi}(A_t, S_{t+1}, A_{t+1}, \dots, S_{T-1}, A_{T-1}, S_T)$$

$$= \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\,\pi(A_{t+1}|S_{t+1}) \dots p(S_T|S_{T-1}, A_{T-1})$$

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

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

- Update value towards corrected return
  - $V(S_t) = V(S_t) + \alpha(\rho_{t:T-1} G_t - V(S_t))$
- 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

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

- 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 behavior policy $A_{t+1} \sim \mu(\cdot \mid S_t)$
- 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

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

Q-Learning (Off-policy Control)

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

$$\pi(S_{t+1}) = \underset{a'}{\text{argmax}}\ Q(S_{t+1}, a')$$

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

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

Q-Learning (Off-policy Control)

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$

Repeat (for each episode):

  Initialize S

  Repeat (for each step of episode):

    Choose A from S using policy derived from Q (e.g., ε-greedy)

    Take action A, observe R, S'

    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$

    $S \leftarrow S'$;

  until S is terminal

or another behavior policy

greedy

Convergence of Q-Learning

Theorem:

If (i) all (state, action) pairs are generated infinitely often,

(ii) given (state, action), the next state is generated independently at each occurrence of the (state, action) pair,

(iii) the learning rate satisfies $\sum_{n\geq 0} \alpha_n = \infty$ and $\sum_{n\geq 0} \alpha_n^2 < \infty$,

(iv) a few other technical conditions are satisfied,

then $Q(s,a)$ converges to $q_*(s,a)$ with probability one.

[DB] p. 254

Tsitsiklis, Asynchronous Stochastic Approximation and Q-Learning, 1994

SARSA vs. Q-learning

- SARSA is on-policy: behavior and target policies are the same

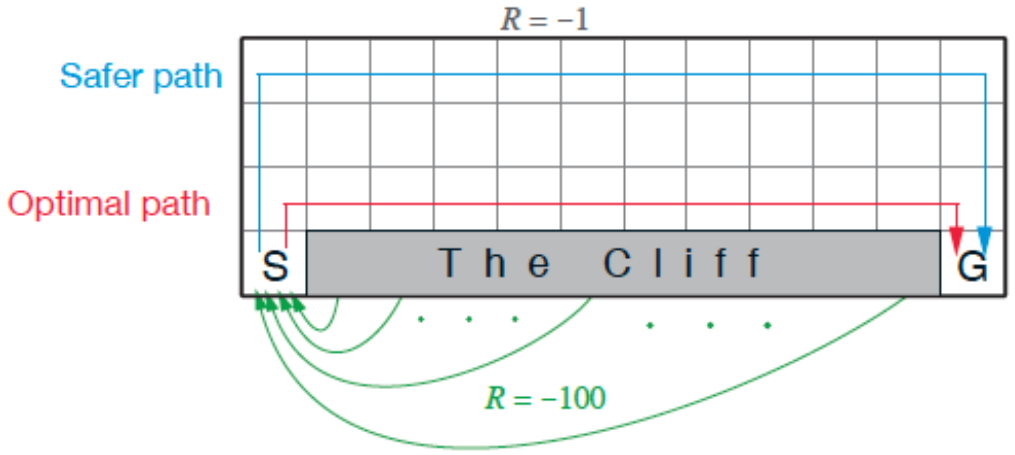
  Q-learning is off-policy: behavior and target policies are different
- In ϵ-greedy step, SARSA needs ϵ to decay for achieve optimality
- Q-learning directly learns the optimal policy and does not need ϵ to decay
- Q-learning has higher variance than SARSA

SARSA vs. Q-learning

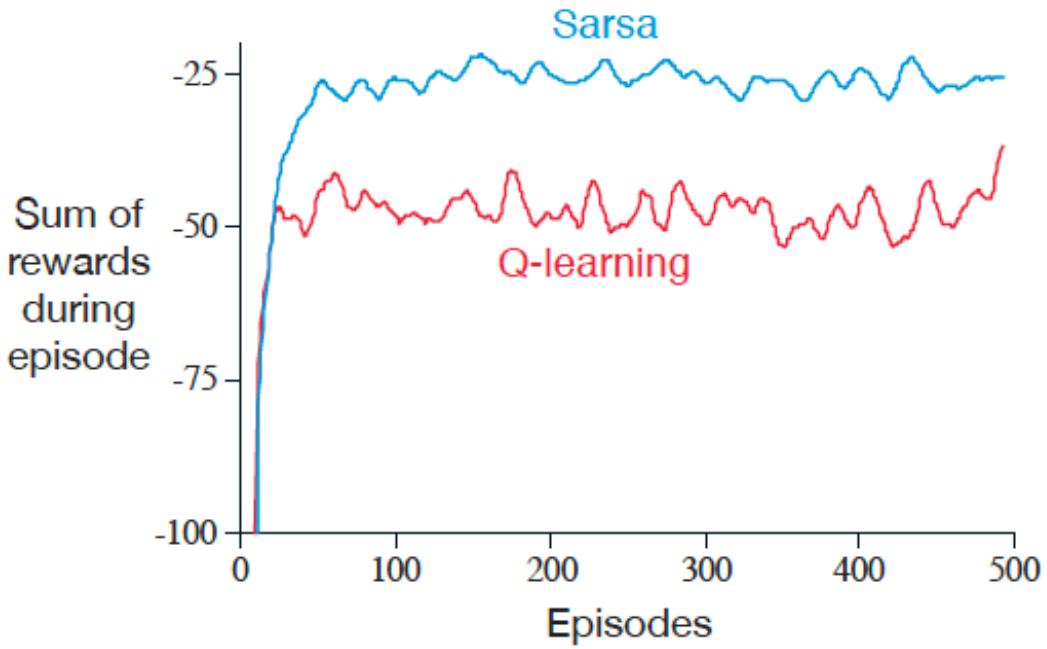
- “SARSA will approach convergence *allowing* for possible penalties from exploratory moves, whilst Q-learning will ignore them.
  - That makes SARSA more conservative - if there is risk of a large negative reward close to the optimal path, Q-learning will tend to trigger that reward whilst exploring, whilst SARSA will tend to avoid a dangerous optimal path and only slowly learn to use it when the exploration parameters are reduced. “

https://stats.stackexchange.com/questions/326788/when-to-choose-sarsa-vs-q-learning

Cliff Walking Example



$\epsilon = 0.1$



SARSA vs. Q-learning

- “In practice, the conservation of SARSA can make a big difference if mistakes are costly - e.g. you are training a robot not in simulation, but in the real world. You may prefer a more conservative learning algorithm that avoids high risk, if there was real time and money at stake if the robot was damaged.”
- “If your goal is to train an optimal agent in simulation, or in a low-cost and fast-iterating environment, then Q-learning is a good choice, as it learns the optimal policy directly. If your agent learns online, and you care about rewards gained *whilst learning*, then SARSA may be a better choice.”

https://stats.stackexchange.com/questions/326788/when-to-choose-sarsa-vs-q-learning

Relationship Between DP and TD

Full Backup (DP)	*Sample Backup (TD)*
Iterative Policy Evaluation	TD Learning
$V(s) \leftarrow \mathbb{E}\left[R + \gamma V(S') \mid s\right]$	$V(S) \overset{\alpha}{\leftarrow} R + \gamma V(S')$
Q-Policy Iteration	Sarsa
$Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma Q(S',A') \mid s,a\right]$	$Q(S,A) \overset{\alpha}{\leftarrow} R + \gamma Q(S',A')$
Q-Value Iteration	Q-Learning
$Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(S',a') \mid s,a\right]$	$Q(S,A) \overset{\alpha}{\leftarrow} R + \gamma \max_{a' \in \mathcal{A}} Q(S',a')$

where $x \overset{\alpha}{\leftarrow} y \equiv x \leftarrow x + \alpha(y - x)$

Expected Sarsa

- Similar to Q-learning except the update rule is replaced by

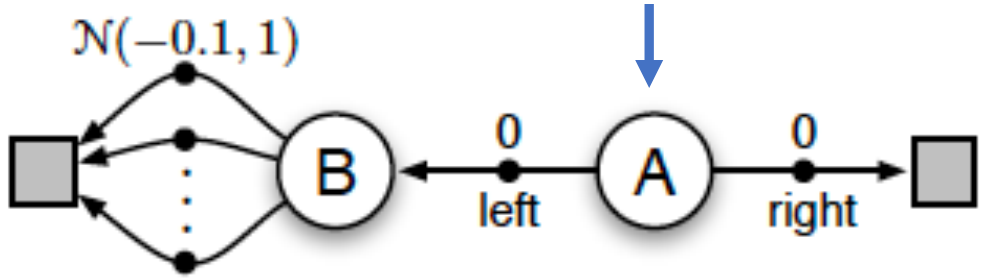
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \mathbb{E}_\pi[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)\Big]$$

$$\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \sum \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\Big],$$

- when π is the greedy policy with respect to $Q(\cdot,\cdot)$, it reduces to Q-learning
- when π is the policy used to generate the next action (e.g., ϵ-greedy), it moves deterministically in the same direction as Sarsa moves in expectation
- Generalizes Q-learning and reliably improves Sarsa (with small additional computational cost)

Maximization Bias

- Finding target policy involves taking maximization of estimated values
  - In Q-learning, the target policy is the greedy policy given the current action values
  - In Sarsa, the target policy is often ϵ-greedy
- In both methods, maximum over estimated values is used implicitly as an estimate of the maximum true value
- Maximization bias: $E_\pi(Q(s, \text{argmax}_a Q(s,a)) \neq q_\pi(s,a)$



Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
    Take action A, observe R, S'
    With 0.5 probabilility:

$$Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S', a)\big) - Q_1(S, A)\Big)$$

    else:

$$Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S', a)\big) - Q_2(S, A)\Big)$$

    $S \leftarrow S'$
  until S is terminal