# Winter School 2024
# Reinforcement Learning

*Control*

Dr. Lorenzo Brigato

Artificial Intelligence in Health and Nutrition (AIHN) Laboratory
ARTORG Center for Biomedical Engineering Research

Dr. Mykhailo Vladymyrov

Data Science Lab

University of Bern

# Recall

- Why RL?
- State Value function
- Action Value function
- MC vs TD

# Outlook

- Policy Iteration
- Monte Carlo Control
- Q-Learning

# How to Improve a Policy

- Given a policy $\pi$
  - Evaluate the policy $\pi$

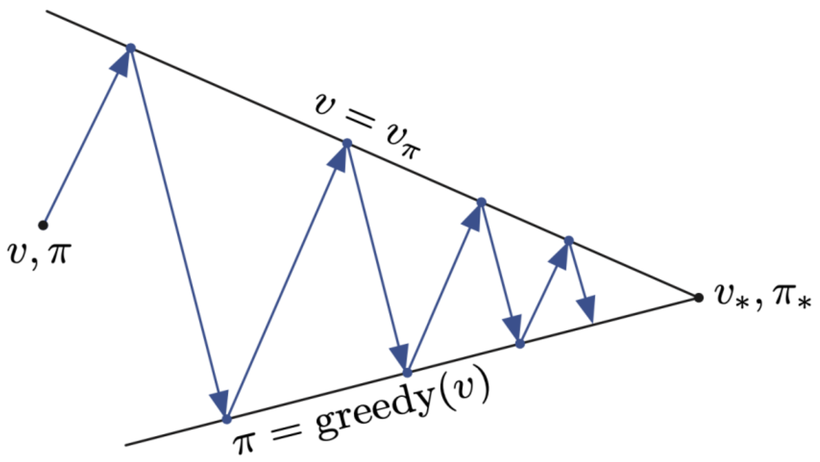
$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \middle|\ S_t = s\right]$$

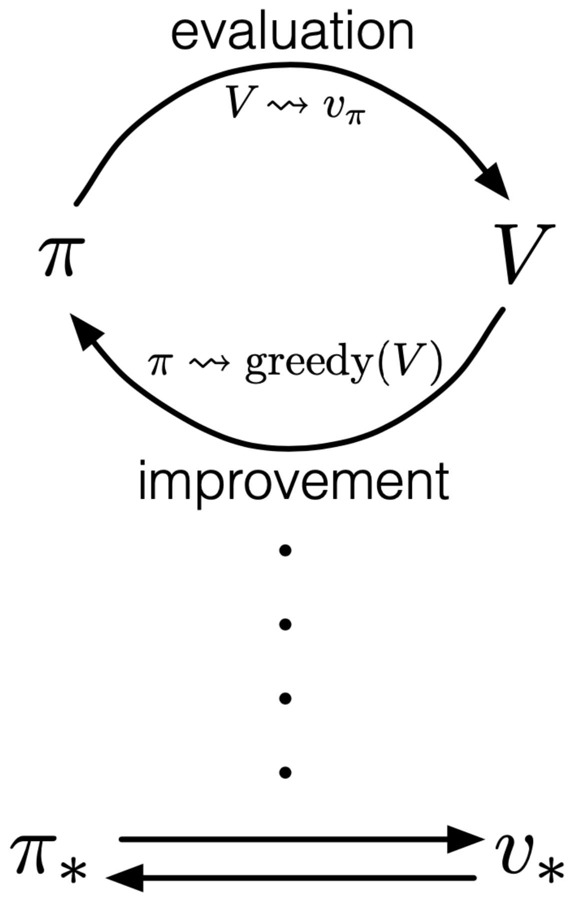
  - Improve the policy by acting greedily with respect to $v_\pi$

$$\pi' = greedy(v_\pi)$$

- In Small Gridworld improved policy was optimal, $\pi' = \pi^*$
- In general, need more iterations of improvement/evaluation
- But this process of policy iteration always converges to π∗ (deterministic environments)

# Policy Iteration




- Policy evaluation Estimate $v_\pi$
  - Iterative policy evaluation
- Policy improvement Generate $\pi' \geq \pi$
  - Greedy policy improvement

[An Introduction to Reinforcement Learning, Sutton and Barto]

# Policy Iteration

**Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$**

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$; $V(terminal) \doteq 0$

2. Policy Evaluation
   Loop:
   - $\Delta \leftarrow 0$
   - Loop for each $s \in \mathcal{S}$:
     - $v \leftarrow V(s)$
     - $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s))\big[r + \gamma V(s')\big]$
     - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   *policy-stable* $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   - *old-action* $\leftarrow \pi(s)$
   - $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
   - If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*

   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

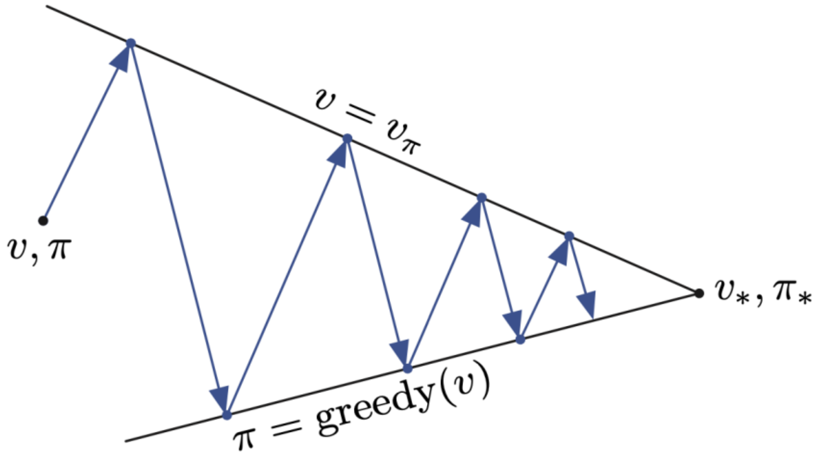
[An Introduction to Reinforcement Learning, Sutton and Barto]

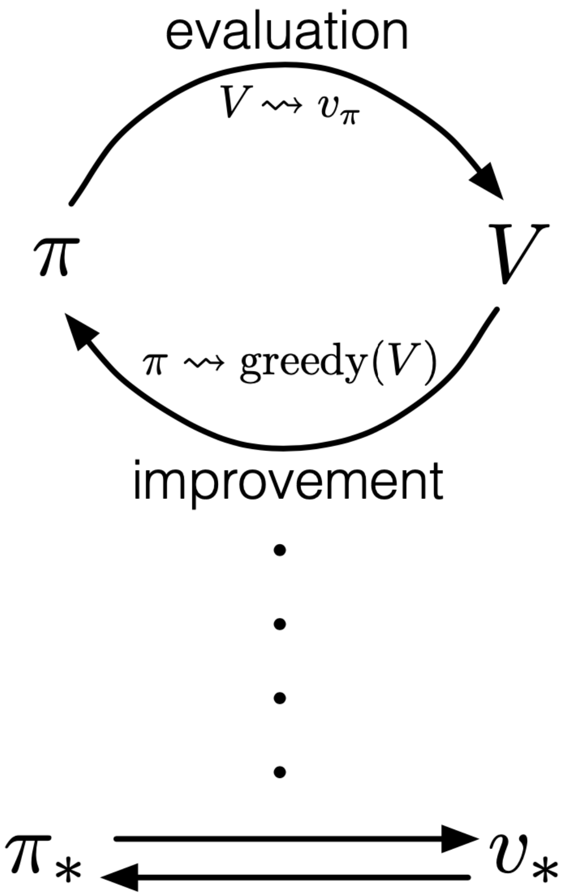
# Model-Free Control

- Some problems can’t be tackled with DP:
  - 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

# Generalized Policy Iteration for Monte-Carlo




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

[An Introduction to Reinforcement Learning, Sutton and Barto]

# Model-Free Policy Iteration Using Action-Value Function

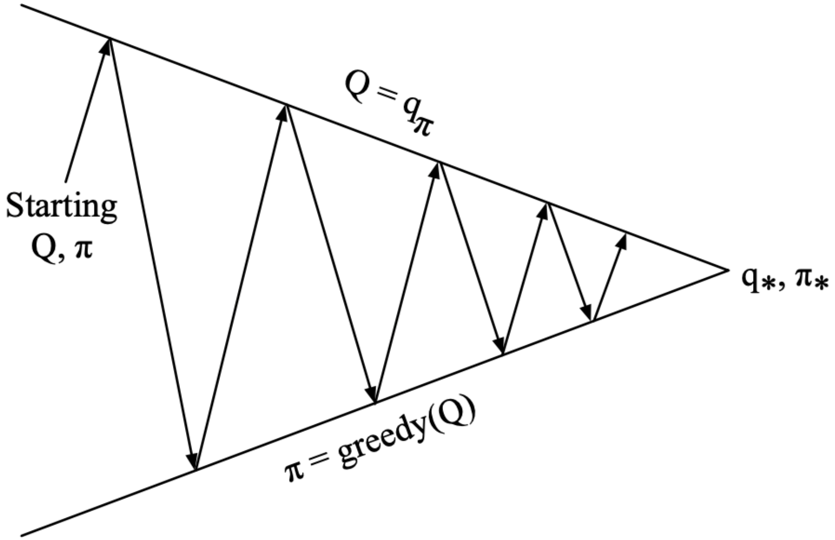
- Greedy policy improvement over $V(s)$ requires model of MDP

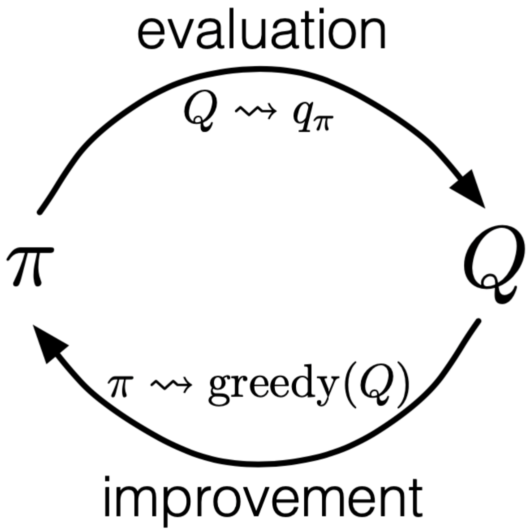
$$\pi'(s) = \underset{a\in\mathcal{A}}{\operatorname{argmax}}\left[\mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s')\right]$$

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

[David Silver, IRL, UCL 2015]

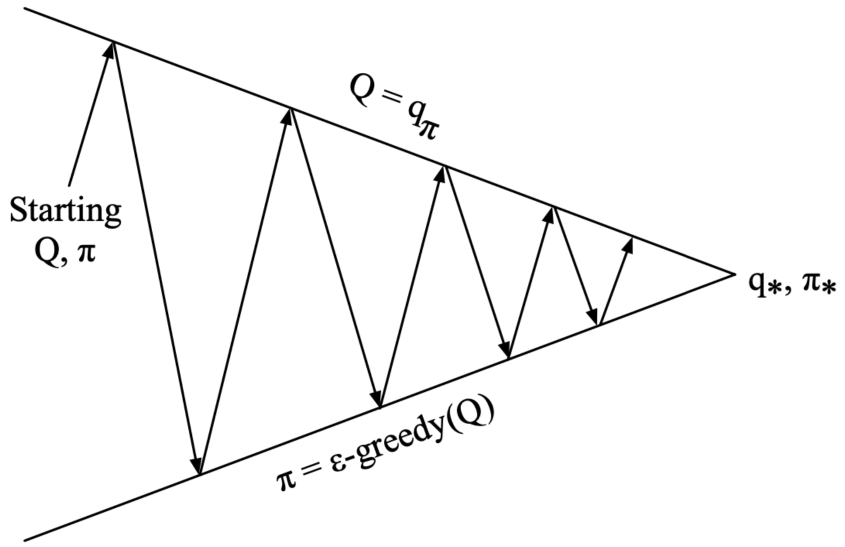
[An Introduction to Reinforcement Learning, Sutton and Barto]

# ε-Greedy Exploration

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

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

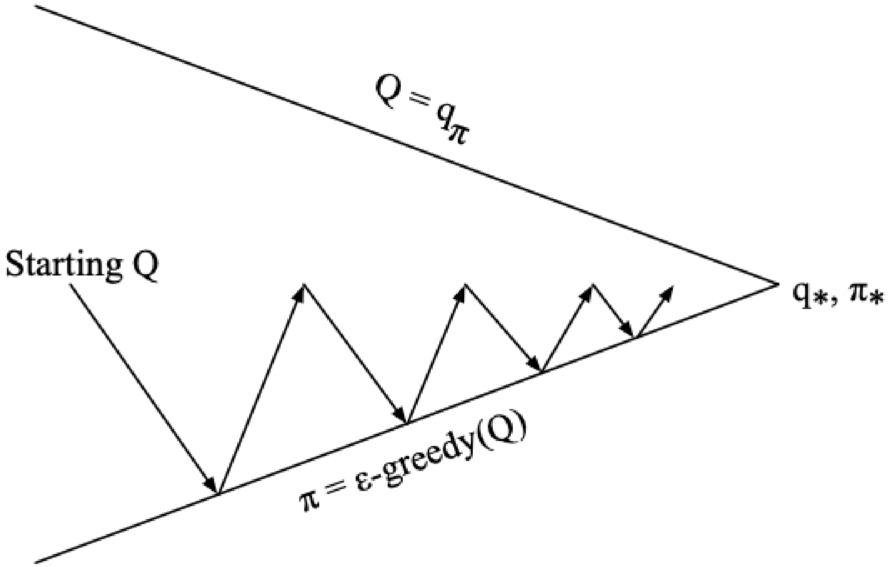
# Monte-Carlo Policy Iteration




- 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 Monte-Carlo Control

- Sample kth episode using $\pi$: $\{S_1, A_1, R_2, \dots, S_T\} \sim \pi$
- 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

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

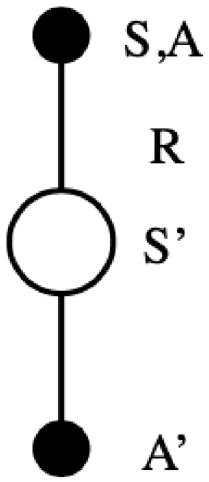
**Theorem**

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

# 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 and Off-Policy Learning

- On-policy learning
  - "Learn on the job"
  - Learn about policy $\pi$ from experience sampled from $\pi$
- Off-policy learning
  - "Look over someone's shoulder"
  - Learn about policy $\pi$ from experience sampled from $\mu$

# 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., $\varepsilon$-greedy)
  Loop for each step of episode:
    Take action $A$, observe $R$, $S'$
    Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\varepsilon$-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]

# Off-Policy Learning

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

$$\{S_1, A_1, R_2, \dots, S_T\} \sim \mu$$

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

# Off-Policy Control with Q-Learning

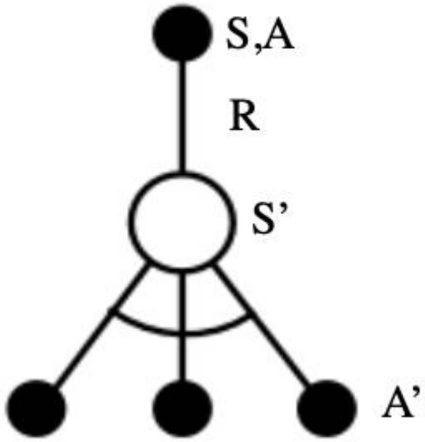
- We now allow both behavior and target policies to improve
- The target policy $\pi$ is greedy w.r.t. $Q(s, a)$

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

- The behavior policy $\mu$ 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., $\varepsilon$-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]

**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 $Q(\textit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
  Loop for each step of episode:
    Take action $A$, observe $R$, $S'$
    Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\varepsilon$-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

**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 $Q(\textit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-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