CS 295: Optimal Control and Reinforcement Learning

Winter 2020

Lecture 5: Temporal-Difference Methods

Roy Fox
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

Today's lecture

- Monte Carlo vs. Temporal-Difference
- On-policy vs. off-policy
- Policy evaluation and policy improvement
- Function representation in table-form vs. differentiable

Policy evaluation

- Distribution over trajectories:

$$p_\pi(\xi) = p(s_0) \prod_t \pi(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

- Expected return: $\mathbb{E}_{\xi \sim p_\pi}[R]$
- State value function: $V_\pi(s) = \mathbb{E}_{\xi \sim p_\pi}[R|s_0 = s]$
- **Dynamic Programming**: compute this recursively

$$V_\pi(s) = \mathbb{E}_{a|s \sim \pi}[r(s,a) + \gamma\, \mathbb{E}_{s'|s,a \sim p}[V_\pi(s')]]$$

Model-free policy evaluation

- Monte Carlo (MC) evaluation:

$$\xi_i | s \sim p_\pi \qquad V(s) = \frac{1}{N} \sum_i R_i$$

- Temporal-Difference (TD) evaluation:

$$\text{for each } (\mathrm{s_i, a_i, r_i, s_i'}) : \quad \Delta V(s_i) \leftarrow \alpha(r_i + \gamma V(s_i') - V(s_i))$$

  - Only works on-policy $a_i | s_i \sim \pi$

- Off-policy version:

$$Q_\pi(s, a) = \mathbb{E}_{\xi \sim p_\pi}[R | s_0 = s, a_0 = a]$$

$$\text{for each } (\mathrm{s_i, a_i, r_i, s_i'}) : \quad \Delta Q(s_i, a_i) \leftarrow \alpha(r_i + \gamma\, \mathbb{E}_{a' | s_i' \sim \pi}[Q(s_i', a')] - Q(s_i, a_i))$$

Deep MC policy evaluation

- Monte Carlo (MC) evaluation:

$$\xi_i | s \sim p_\pi \qquad V(s) = \frac{1}{N} \sum_i R_i$$

- What if the state space is large?

$$\mathcal{L}_\theta(\xi) = (V_\theta(s_0) - R)^2$$

- With proper parametrization, this can yield generalization over state space

- But still very data inefficient

Deep TD policy evaluation

- On-policy Temporal-Difference (TD) evaluation:

$$\text{for each } (\mathrm{s_i}, \mathrm{a_i}, \mathrm{r_i}, \mathrm{s'_i}) : \quad \Delta V(s_i) \leftarrow \alpha(r_i + \gamma V(s'_i) - V(s_i))$$

- Lends itself nicely to SGD:

$$\mathcal{L}_\theta(s, a, r, s') = (r + \gamma V_\theta(s') - V_\theta(s))^2$$

- Using both current-state $V_\theta(s)$ and next-state $V_\theta(s')$ may be unstable
  - Heuristic: use **target network** $V_{\bar{\theta}}(s')$, update it periodically with $\bar{\theta} \leftarrow \theta$

Policy improvement

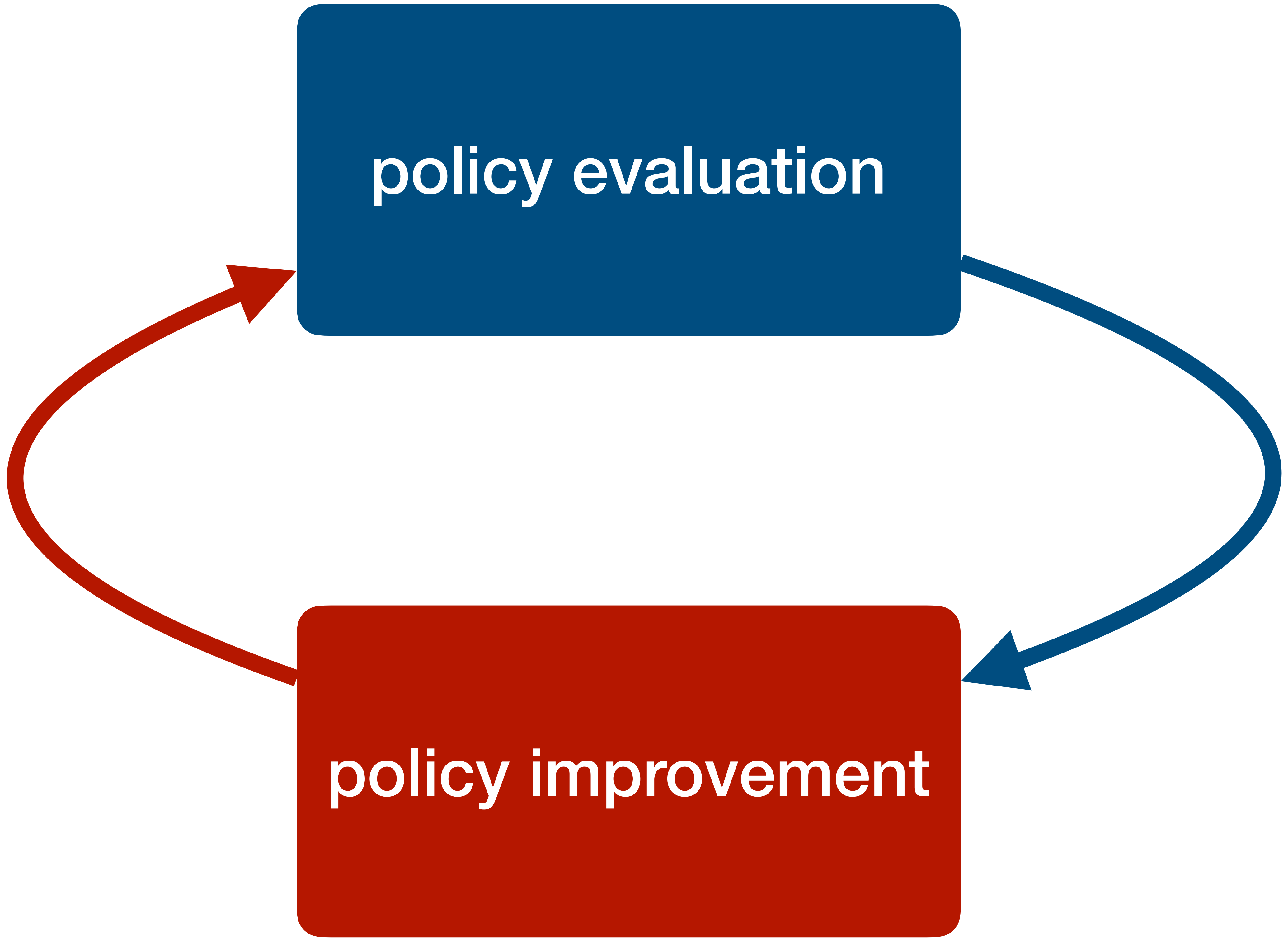
- A value function suggests the greedy policy:

$$\pi(s) = \operatorname*{argmax}_a Q(s,a) = \operatorname*{argmax}_a (r(s,a) + \gamma\, \mathbb{E}_{s'|s,a\sim p}[V(s')])$$

- Proposition: the greedy policy for Q_π is never worse than π
  - Generally: the greedy policy for $\max(Q_{\pi_1}, Q_{\pi_2})$ is never worse than π_1 or π_2
- Corollary 1: any optimal policy π^* is greedy for $Q^* = Q_{\pi^*}$
- Corollary 2: all fixed points of $\pi(s) = \operatorname*{argmax}_a Q_\pi(s,a)$ have $Q_\pi = Q^*$

Bellman optimality

The RL scheme



Policy Iteration

- Evaluate the policy $Q_\pi(s,a) = \mathbb{E}_{\xi \sim p_\pi}[R | s_0 = s, a_0 = a]$
- Update to the greedy policy $\pi(s) = \operatorname*{argmax}_a Q_\pi(s,a)$
- Repeat

- When loop converges, $Q_\pi = Q^*$

Value Iteration

- Repeat:

$$V(s_i) \leftarrow \max_a (r(s_i, a) + \gamma \, \mathbb{E}_{s'|s_i,a \sim p}[V(s')])$$

  - Must update each state repeatedly until convergence

Generalized Policy Iteration

- Alternate by some schedule:

$$V(s_i) \leftarrow \mathbb{E}_{a|s_i \sim \pi}[r(s_i, a) + \gamma\, \mathbb{E}_{s'|s_i, a \sim p}[V(s')]]$$

$$\pi(s_i) \leftarrow \operatorname*{argmax}_a (r(s_i, a) + \gamma\, \mathbb{E}_{s'|s_i, a \sim p}[V(s')])$$

Model-free reinforcement learning

- MC:

$$\xi_i | s, a \sim p_\pi \qquad Q(s,a) \leftarrow \frac{1}{N} \sum_i R_i$$

$$\pi \leftarrow \operatorname{argmax} Q$$

- Q-learning (TD):

$$\Delta Q(s_i, a_i) \leftarrow \alpha(r_i + \gamma \max_{a'} Q(s'_i, a') - Q(s_i, a_i))$$

Deep MC reinforcement learning

- A variant of Monte Carlo Tree Search (MCTS):

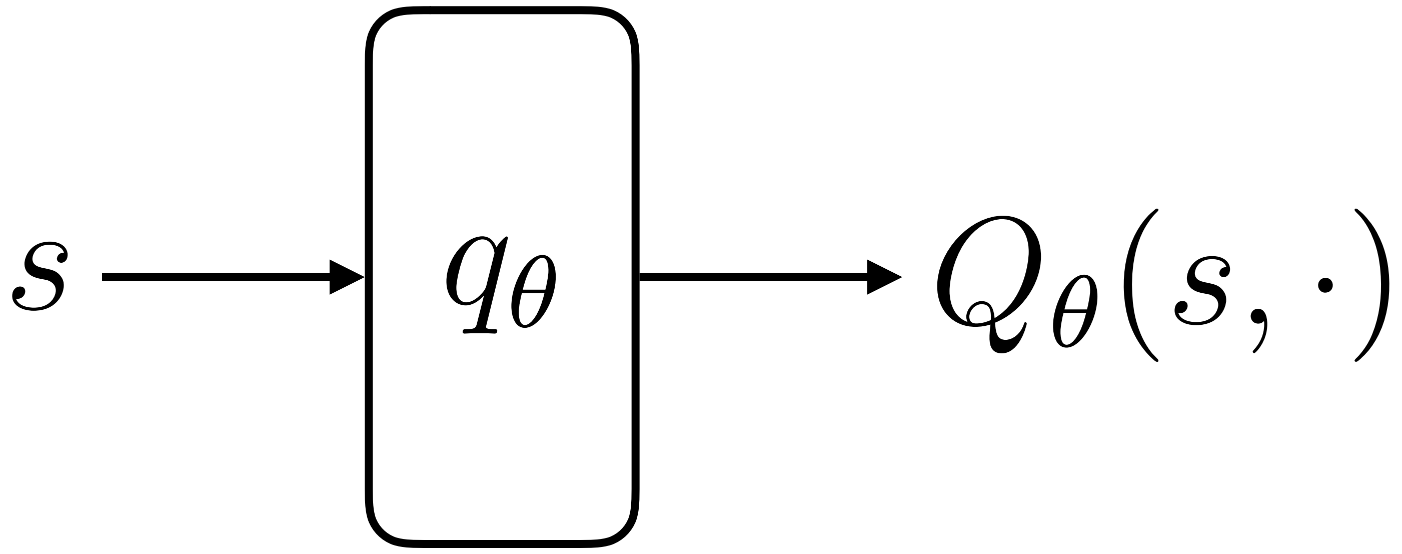
$$\xi \sim p_{\pi_{\bar{\theta}}} \qquad \mathcal{L}_\theta(\xi) = (Q_\theta(s_0, a_0) - R)^2$$

  - With $\pi_{\bar{\theta}}$ greedy for a snapshot of Q_θ

- We need a representation of Q_θ that allows computing

$$\pi_\theta(s) = \operatorname*{argmax}_a Q_\theta(s, a)$$

- For a small action space: Deep Q Network



$$(q_\theta(s))_a = Q_\theta(s, a)$$

- π_θ is not differentiable, but we don't need it to be

Deep TD reinforcement learning

- Deep Q Learning (historically called DQN):

$$\mathcal{L}_\theta(s,a,r,s') = (r + \gamma \max_{a'} Q_{\boxed{\bar{\theta}}}(s',a') - Q_\theta(s,a))^2$$

- This algorithm should work off-policy, so we can keep **replay buffer**
- Variants differ on
  - How to add experience to the buffer
  - How to sample from the buffer

Interaction policy

- In model-free RL, we often get data by interaction with the environment
  - How should we interact?
- On-policy methods (e.g. MC): must use current policy
- Off-policy methods: can use different policy — but not too different!
  - Otherwise may have train–test distribution mismatch (with Deep RL)
- In either case, must make sure interaction policy **explores** well enough

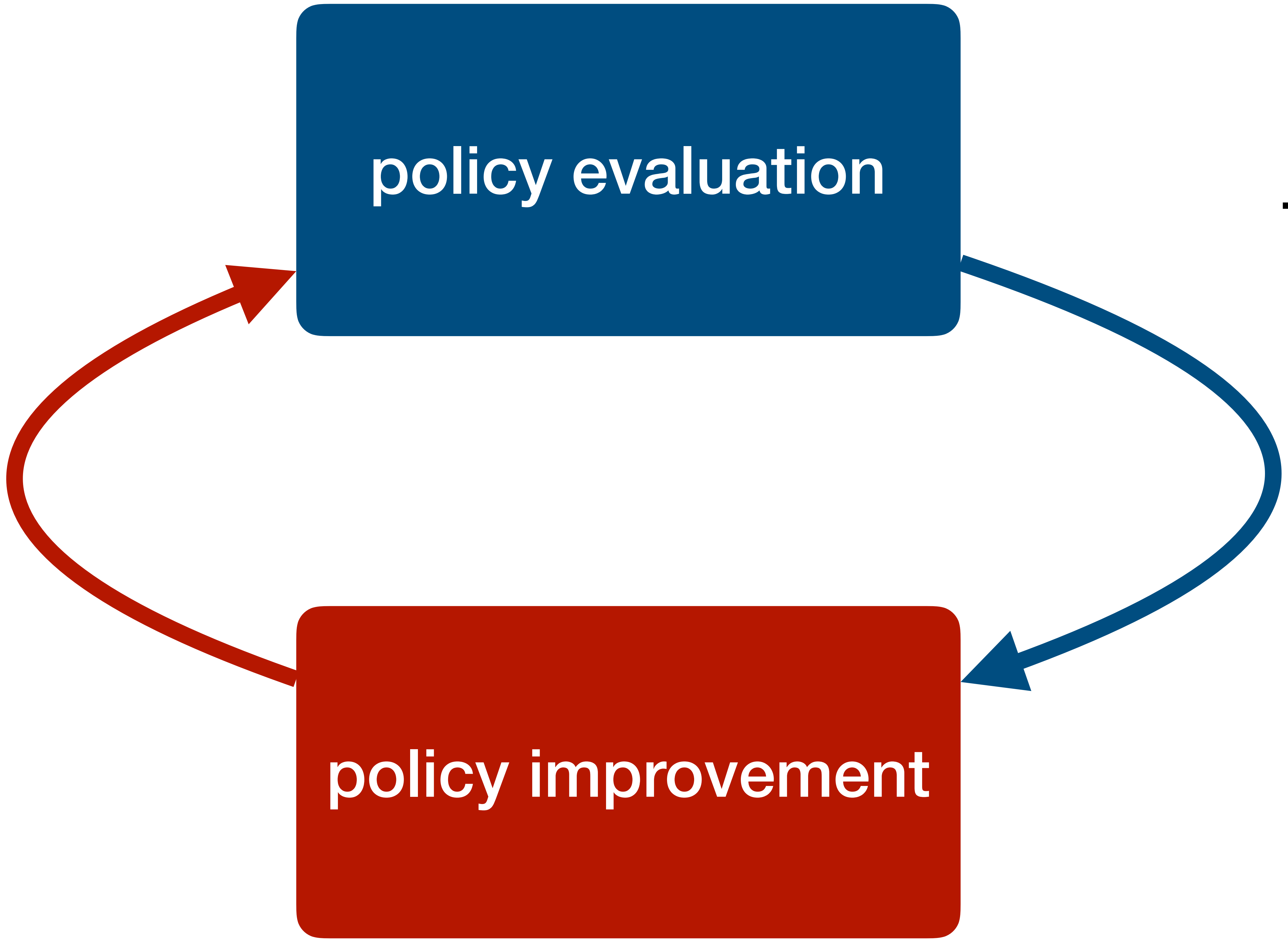
Exploration policies

- ε-greedy exploration: select uniform action w.p. ε, otherwise greedy
- Boltzmann exploration:

$$\pi(a|s) = \underset{a}{\mathrm{sm}}(Q(s,a);\beta) = \frac{\exp(\beta Q(s,a))}{\sum_{a'} \exp(\beta Q(s,a'))}$$

  - Becomes uniform as $\beta \to 0$, greedy as $\beta \to \infty$

Putting it all together: DQN



exploration

differentiable value function approximation

greedy policy

Recap

- RL is a policy evaluation ↔ policy improvement loop
- Temporal-Difference methods exploit the dynamical-programming structure
- Off-policy methods don't need to throw out data as often when policy changes
- Many approaches can be made differentiable for Deep RL