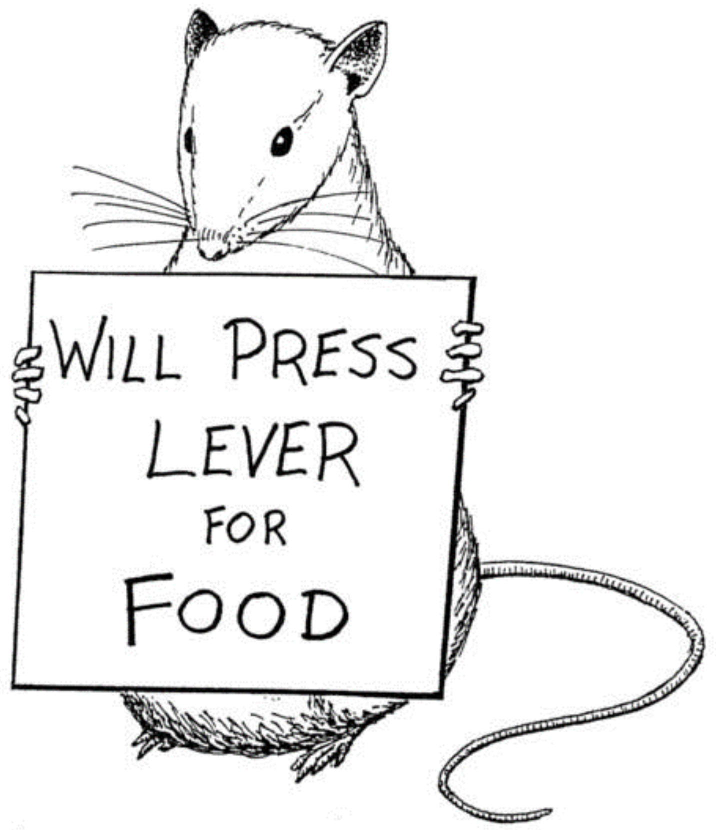
UCI University of California, Irvine

# CS 277: Control and Reinforcement Learning

## Winter 2021

# Lecture 3: Temporal-Difference Methods

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



# Logistics

assignments

- Assignments 1 will be published in 2 parts
  - Math part today
  - Programming part Thursday
  - Due next Thursday

resources

- Lots of resources on the website
- Will be updated with papers relevant to each lecture

# Today's lecture

Policy evaluation and improvement

Monte Carlo and Temporal-Difference

Differentiable Representation

# Policy evaluation

- Distribution over trajectories:

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

- Expected return: $\mathbb{E}_{\xi \sim p_\pi}[R]$

- State value function: $V_\pi(s) = \mathbb{E}_{\xi \sim p_\pi}[R \mid s_0 = s]$

- Dynamic Programming: compute $V_\pi$ recursively

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

# Model-free policy evaluation

- Monte Carlo (MC) evaluation:

$$\text{sample } \xi_i \,|\, s_0 = s \sim p_\pi \qquad V(s) = \frac{1}{N} \sum_i R(\xi_i)$$

- Temporal-Difference (TD) evaluation:

$$\text{for each } (s_i, a_i, r_i, s_i')\text{: } \Delta V(s_i) \leftarrow \alpha(\overbrace{r_i + \gamma V(s_i') - V(s_i)}^{\text{should be 0 in expectation, update towards that}})$$

  - Only works on-policy = data comes from the evaluated policy $a_i \,|\, s_i \sim \pi$

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

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

# Deep MC policy evaluation

- (reminder) Monte Carlo (MC) evaluation:

$$\text{sample } \xi_i \,|\, s_0 = s \sim p_\pi \qquad V(s) = \frac{1}{N} \sum_i R(\xi_i)$$

- What if the state space is large?

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

- With proper parametrization, this can yield generalization over state space
    - But still very data inefficient

# Deep TD policy evaluation

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

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

- Lends itself nicely to Stochastic Gradient Descent (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 $\bar{\theta} \leftarrow \theta$

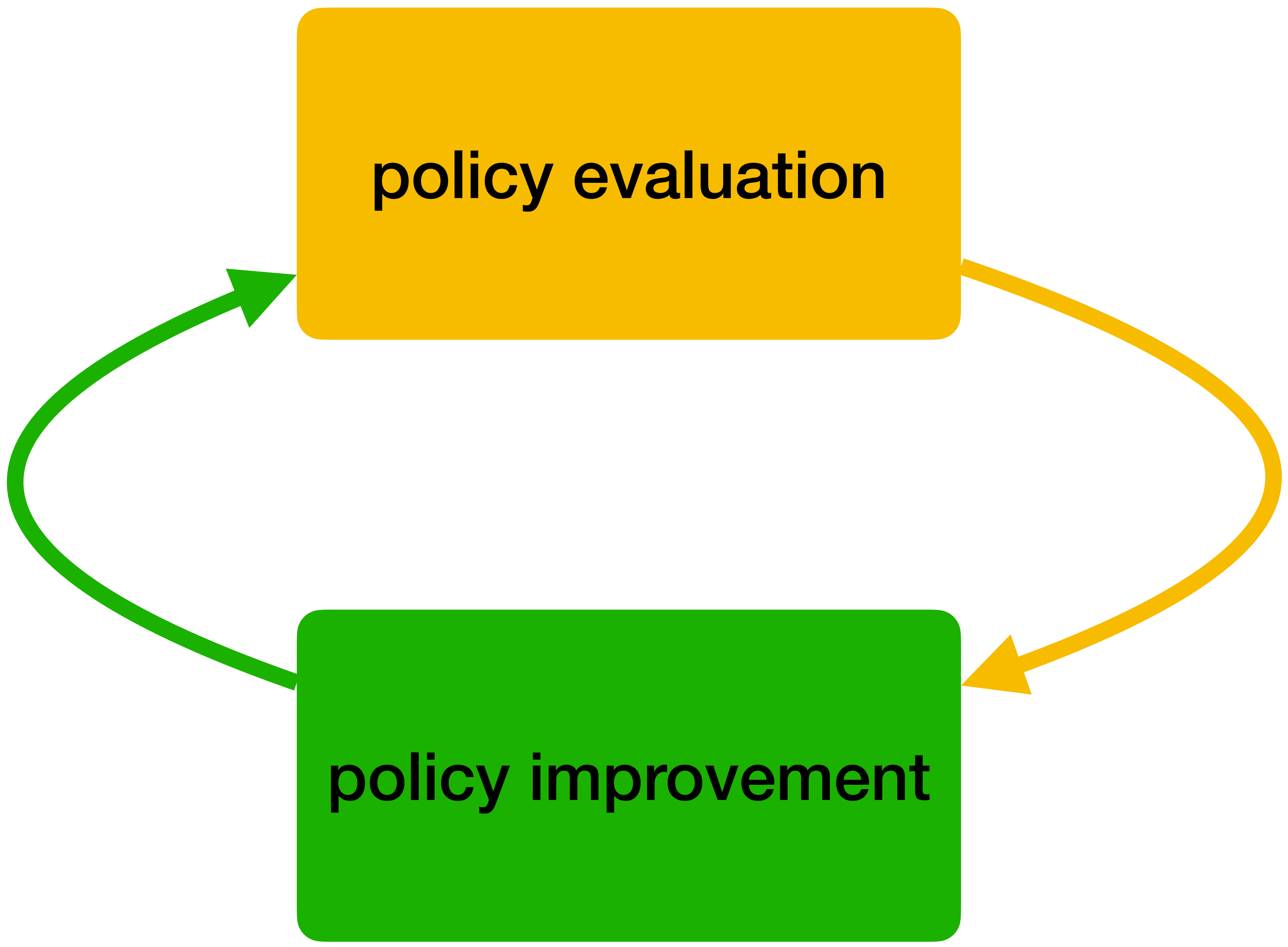
# Policy improvement

- A value function suggests the greedy policy:

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

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

- Repeat

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

# Value Iteration

- Repeat:
  - For each $s_i$:

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

  - Must update each state repeatedly until convergence

# Generalized Policy Iteration

- Alternate by some schedule:

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

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

# Model-free reinforcement learning

- Repeatedly perform MC estimation with the greedy policy:

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

$$\pi \leftarrow \arg\max Q$$

- Q-learning (TD): on experience $(s_i, a_i, r_i, s_i')$

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

# Deep MC reinforcement learning

- Repeatedly perform MC estimation with the greedy policy for $Q_\theta$:

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

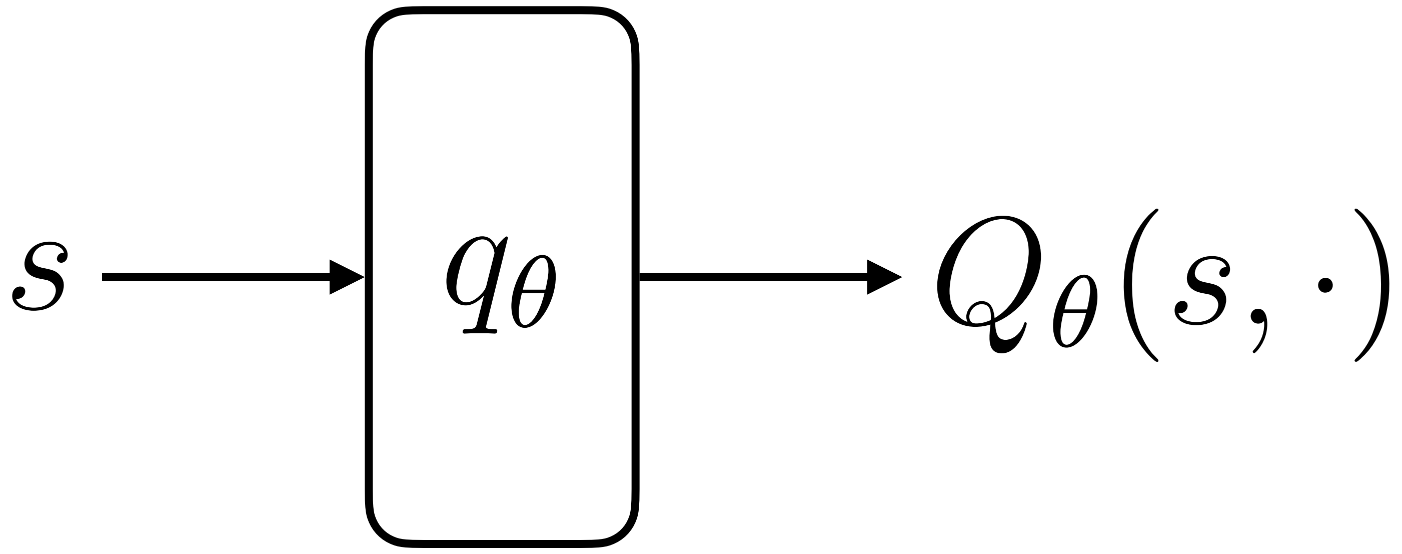
  - With $\pi_{\bar{\theta}}$ greedy for a snapshot of $Q_\theta$

- We need a representation of $Q_\theta$ that allows computing

$$\pi_\theta(s) = \arg\max_a Q_\theta(s, a)$$

- For a small action space: Deep Q Network

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




- $\pi_\theta$, unlike $Q_\theta$, 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 past experience
  - Replay buffer = data set of recent past experience of learner policy at that time
  - 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?
  - Must we use current learner policy (on-policy data) or another (off-policy data)
- 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
- In either case, must make sure interaction policy explores well enough

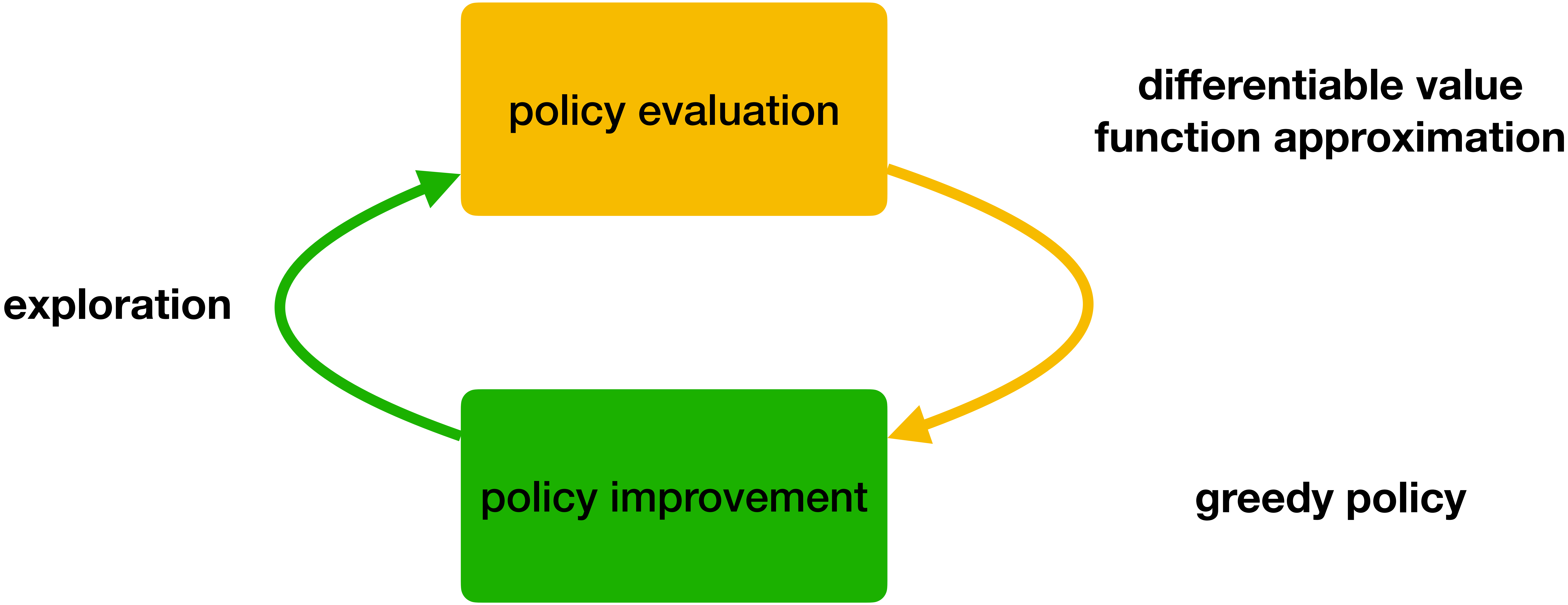
# Exploration policies

- $\epsilon$-greedy exploration: select uniform action w.p. $\epsilon$, otherwise greedy
- Boltzmann exploration:

$$\pi(a \mid s) = \mathrm{sm}_a(Q(s,a);\beta) = \frac{\exp(\beta Q(s,a))}{\sum_{\bar{a}} \exp(\beta Q(s,\bar{a}))}$$

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

# Putting it all together: DQN



# Recap

- RL is a (policy evaluation ↔ policy improvement) loop
- Temporal-Difference methods exploit the dynamical-programming structure
- Off-policy methods throw out data much less often when policy changes
- Many approaches can be made differentiable for Deep RL

# Logistics

assignments

- Assignments 1 will be published in 2 parts
  - Math part today
  - Programming part Thursday
  - Due next Thursday

resources

- Lots of resources on the website
- Will be updated with papers relevant to each lecture