Q-Learning

CS 224R

Course reminders

- Today
  - **Homework 1** due, **homework 2** out
  - Sending out $100 AWS credits + form for GPU quota.
  - Extra Q-learning section (1:30-2:30 pm)

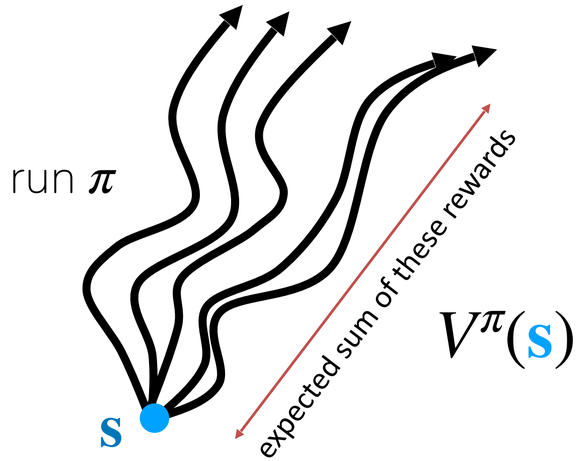
Very important if you want GPUs, especially for default project

Recap: Some Useful Objects

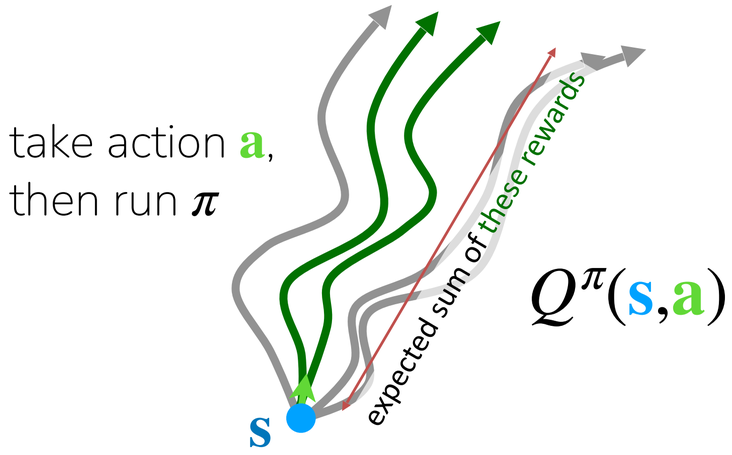
value function $\boldsymbol{V^{\pi}(\mathbf{s})}$ - future expected rewards starting at $\mathbf{s}$ and following $\boldsymbol{\pi}$

Q-function $\boldsymbol{Q^{\pi}(\mathbf{s}, \mathbf{a})}$ - future expected rewards starting at $\mathbf{s}$, taking $\mathbf{a}$, then following $\boldsymbol{\pi}$

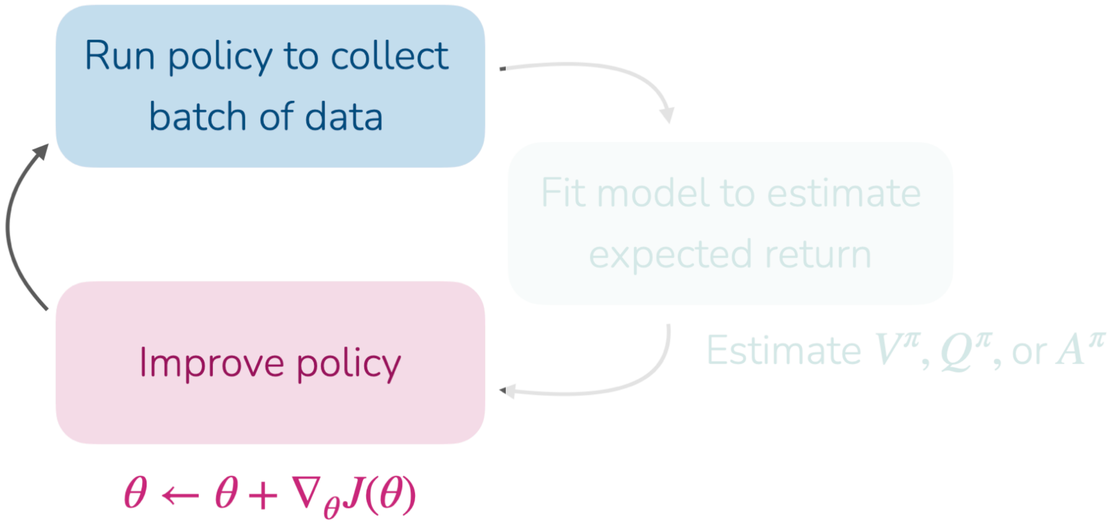
$$V^{\pi}(\mathbf{s}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t\right]$$



$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t, \mathbf{a}_t\right]$$



Recap: Methods



Online RL with policy gradients

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \left(\left(\sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right) - b \right)$$

samples from policy

policy log likelihood

reward to go

baseline

Do more of the above average stuff, less of the below average stuff.

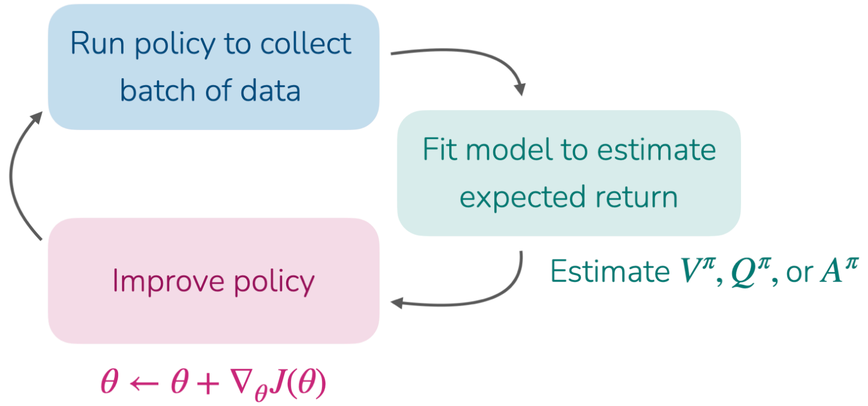
Online RL with actor-critic

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) A^\pi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

Estimate what is good and bad, then do more of the good stuff.

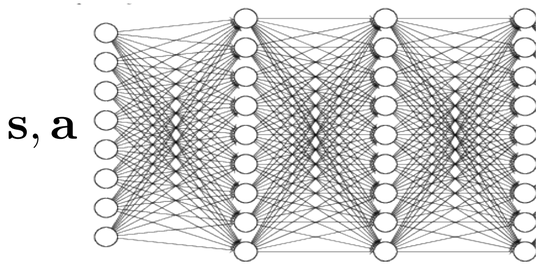
Recap: Off-Policy Policy Evaluation

Online RL with actor-critic



Estimate what is good and bad, then do more of the good stuff.

For off-policy version: can we estimate Q-function using past policy data?



For *any* $(\mathbf{s}, \mathbf{a})$:

$$Q^{\pi_\theta}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\cdot|\mathbf{s},\mathbf{a}), \bar{\mathbf{a}}' \sim \pi_\theta(\cdot|\mathbf{s}')} \left[Q^{\pi_\theta}(\mathbf{s}', \bar{\mathbf{a}}') \right]$$

1. Sample $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)$ from the buffer
2. Sample $\bar{\mathbf{a}}'_i \sim \pi_\theta(\cdot|\mathbf{s}'_i)$ from *current* policy

$$Q^{\pi_\theta}(\mathbf{s}_i, \mathbf{a}_i) \approx \underbrace{r(\mathbf{s}_i, \mathbf{a}_i) + \gamma Q^{\pi_\theta}(\mathbf{s}'_i, \bar{\mathbf{a}}'_i)}_{y_i}$$

$$\mathcal{L}(\phi) = \frac{1}{N} \sum_i \left\| \hat{Q}^\pi_\phi(\mathbf{s}_i, \mathbf{a}_i) - y_i \right\|^2$$

Note: To be accurate, targets y_i need sufficient action coverage

The plan for today

Value-based RL methods

1. Q-learning RL method
    a. Policy iteration
    b. Bellman optimality equation
    c. How to collect data for Q-learning methods
2. Q-learning in practice
    d. Target networks
    e. Double DQN
    f. N-step returns

Part of homework 2!

Key learning goals:

- How Q-functions relate to policies
- How to do RL without learning an explicit policy
- How to stabilize Q-learning in practice

A thought exercise

Recall: *Definition of Q-values*

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_\pi\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t\right]$$

"total reward we get if we take $\mathbf{a}_t$ in $\mathbf{s}_t$...
... and then follow the policy π"

For some policy π, say you have an accurate estimate $\hat{Q}^\pi(\mathbf{s}, \mathbf{a})$ for all $\mathbf{s}, \mathbf{a}$

Define a new policy $\pi_{\text{new}}(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg\max_{\mathbf{a}} Q_\phi(\mathbf{s}_t, \mathbf{a}) \\ 0 & \text{otherwise} \end{cases}$

2D navigation example

Initial state distribution $p(\mathbf{s}_1)$

Reward is 1 at ★, 0 elsewhere

Current policy π goes right in all states

takes 1 timestep, 2 timesteps to travel these distances

Questions: Is the new policy better, worse, or the same as the original policy?
Is it the optimal policy? Why or why not?

Can we omit policy gradient completely?

$Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$: expected reward from taking $\mathbf{a}_t$ and subsequently following π

$\arg\max_{\mathbf{a}_t} Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$: best action from $\mathbf{s}_t$, if we then follow π afterwards

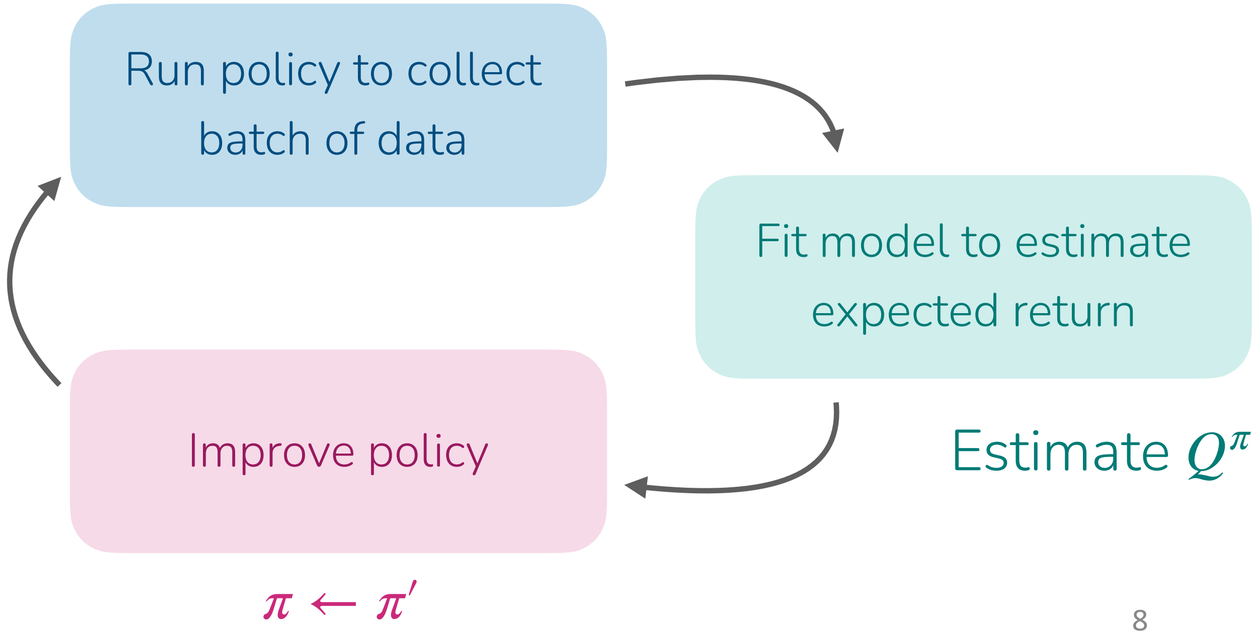
at *least* as good as any $\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$

regardless of what $\pi(\mathbf{a}_t|\mathbf{s}_t)$ is!

forget policies, let's just do this!

$$\pi'(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}} Q^\pi(\mathbf{s}_t, \mathbf{a}) \\ 0 \text{ otherwise} \end{cases}$$

as good as π
(probably better)



Slide adapted from Sergey Levine

From actor-critic to critic only

~~Off-policy actor critic with replay buffer~~ Q-learning

1. take action $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{Q}^\pi_\phi$ using targets $y_i = r_i + \gamma \hat{Q}^\pi_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$ where $\mathbf{a}'_i \sim \pi(\cdot|\mathbf{s}'_i)$
4. ~~$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}^\pi_i|\mathbf{s}_i) \hat{Q}^\pi(\mathbf{s}_i, \mathbf{a}^\pi_i)$ where $\mathbf{a}^\pi_i \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i)$~~
5. ~~$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$~~
4. define new policy $\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg\max_{\mathbf{a}} \hat{Q}_\phi(\mathbf{s}_t, \mathbf{a}) \\ 0 & \text{otherwise} \end{cases}$

From actor-critic to critic only

~~Off-policy actor critic with replay buffer~~

1. take action $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{Q}^\pi_\phi$ using targets $y_i = r_i + \gamma \hat{Q}^\pi_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$ where $\mathbf{a}'_i \sim \pi(\cdot|\mathbf{s}'_i)$ <- policy *evaluation*
4. ~~$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}^\pi_i|\mathbf{s}_i) \hat{Q}^\pi(\mathbf{s}_i, \mathbf{a}^\pi_i)$ where $\mathbf{a}^\pi_i \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i)$~~ (**Note**: can do multiple gradient steps here)
5. ~~$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$~~
4. define new policy $\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg\max_{\mathbf{a}} \hat{Q}_\phi(\mathbf{s}_t, \mathbf{a}) \\ 0 & \text{otherwise} \end{cases}$ <- policy *improvement*

"Policy iteration"

Can we improve the policy in the Q-function update?

From actor-critic to critic only

~~Off-policy actor critic with replay buffer~~ Q-learning

1. take action $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{Q}^{\pi}_{\phi}$ using targets ~~$y_i = r_i + \gamma \hat{Q}^{\pi}_{\phi}(\mathbf{s}'_i, \mathbf{a}'_i)$ where $\mathbf{a}'_i \sim \pi(\cdot|\mathbf{s}'_i)$~~ $y_i = r_i + \gamma \max_{\mathbf{a}'} \hat{Q}_{\phi}(\mathbf{s}'_i, \mathbf{a}')$
4. ~~$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}^{\pi}_i|\mathbf{s}_i) \hat{Q}^{\pi}(\mathbf{s}_i, \mathbf{a}^{\pi}_i)$ where $\mathbf{a}^{\pi}_i \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i)$~~
5. ~~$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$~~
4. define new policy $\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg\max_{\mathbf{a}} \hat{Q}_{\phi}(\mathbf{s}_t, \mathbf{a}) \\ 0 & \text{otherwise} \end{cases}$

Q-values for new policy!

From actor-critic to critic only

Q-learning

1. take action $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{Q}_\phi$ using targets $y_i = r_i + \gamma \max_{\mathbf{a}'} \hat{Q}_\phi(\mathbf{s}'_i, \mathbf{a}')$
4. define new policy $\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg\max_{\mathbf{a}} \hat{Q}_\phi(\mathbf{s}_t, \mathbf{a}) \\ 0 & \text{otherwise} \end{cases}$

Terminology

Bellman equation

Bellman optimality equation

Why does make sense?

Recall: $Q^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\cdot|\mathbf{s},\mathbf{a}), \bar{\mathbf{a}}' \sim \pi(\cdot|\mathbf{s}')} [Q^\pi(\mathbf{s}', \bar{\mathbf{a}}')]$ for *all* $(\mathbf{s}, \mathbf{a})$

-> This holds for any policy π (including the optimal policy π^*)

If π^* is the optimal policy, we also get:

$$Q^{\pi^*}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\cdot|\mathbf{s},\mathbf{a})} \left[\max_{\bar{\mathbf{a}}'} Q^{\pi^*}(\mathbf{s}', \bar{\mathbf{a}}') \right] \text{ for } \textit{all } (\mathbf{s}, \mathbf{a})$$

the action π^* would take in state $\mathbf{s}'$

When we optimize step 3, we are trying to make this equation hold!

From actor-critic to critic only

Q-learning

1. take action $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{Q}_\phi$ using targets $y_i = r_i + \gamma \max_{\mathbf{a}'} \hat{Q}_\phi(\mathbf{s}'_i, \mathbf{a}')$
4. define new policy $\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg\max_{\mathbf{a}} \hat{Q}_\phi(\mathbf{s}_t, \mathbf{a}) \\ 0 & \text{otherwise} \end{cases}$

Note: Q-learning is off-policy

Why does make sense?

If π^* is the optimal policy, we also get:

$$Q^{\pi^*}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim p(\cdot|\mathbf{s},\mathbf{a})} \left[\max_{\bar{\mathbf{a}}'} Q^{\pi^*}(\mathbf{s}', \bar{\mathbf{a}}') \right] \text{ for } \textit{all } (\mathbf{s}, \mathbf{a})$$

When we optimize step 3, we are trying to make this equation hold!

Will this algorithm converge to optimal Q^{π^*}?

Yes, if you maintain a table of Q-values for every state and action. More generally, no. 😢

Can construct scenarios where it diverges, even with linear Q. *But*, it can be made to work well

What kind of data should we collect?

Q-learning

from some exploration policy

1. take action $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{Q}_\phi$ using targets $y_i = r_i + \gamma \max_{\mathbf{a}'} \hat{Q}_\phi(\mathbf{s}'_i, \mathbf{a}')$
4. define new policy $\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg\max_{\mathbf{a}} \hat{Q}_\phi(\mathbf{s}_t, \mathbf{a}) \\ 0 & \text{otherwise} \end{cases}$

Note: Q-learning is off-policy

want coverage for many actions $\mathbf{a}$

What are good choices for data collection?

1. With probability ϵ, take uniformly random action

$$\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 - \epsilon \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q_\phi(\mathbf{s}_t, \mathbf{a}_t) \\ \epsilon/(|\mathcal{A}| - 1) \text{ otherwise} \end{cases}$$

"epsilon-greedy"

Often start with larger ϵ, decrease over training.

2. Take actions with probability proportional to their Q-value.

$$\pi(\mathbf{a}_t|\mathbf{s}_t) \propto \exp(Q_\phi(\mathbf{s}_t, \mathbf{a}_t))$$

"Boltzmann exploration"

Putting it together

full Q-learning with replay buffer:

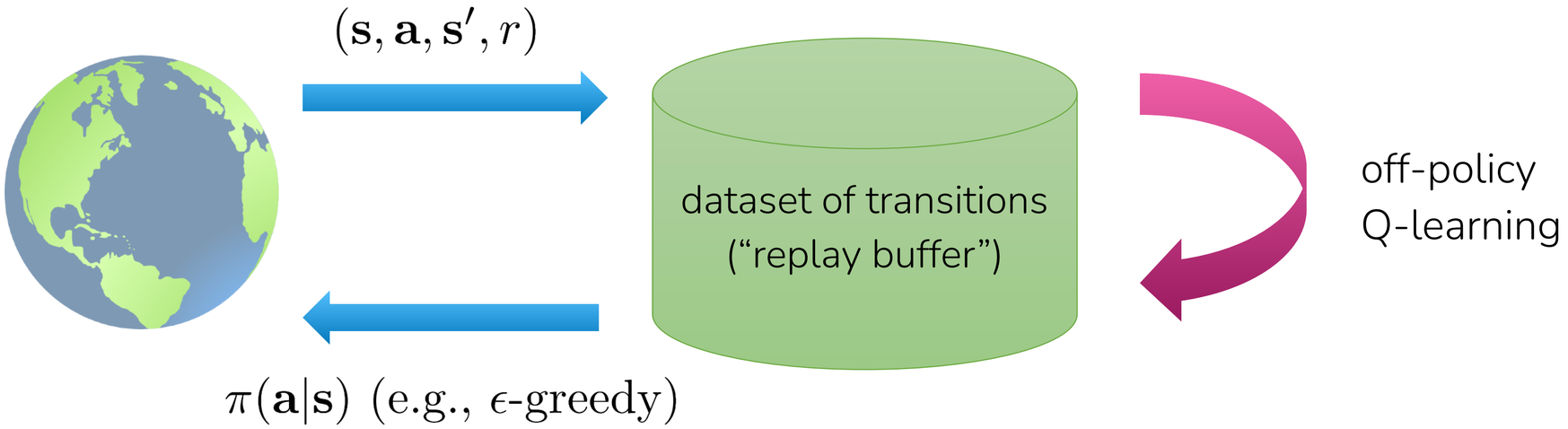
1. collect data $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy, add it to $\mathcal{R}$

$K\times$

2. sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ from $\mathcal{R}$
3. $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}')])$

K = 1 is common, though larger K more efficient

result: Q_ϕ final policy $\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg\max_{\mathbf{a}} Q_\phi(\mathbf{s}_t, \mathbf{a}) \\ 0 & \text{otherwise} \end{cases}$



Slide adapted from Sergey Levine

The plan for today

Value-based RL methods

1. Q-learning RL method
    a. Policy iteration
    b. Bellman optimality equation
    c. How to collect data for Q-learning methods
2. **Q-learning in practice**
    d. Target networks
    e. Double DQN
    f. N-step returns

How to make Q-learning stable?

full Q-learning with replay buffer:

1. collect data $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy, add it to $\mathcal{R}$

$K\times$

2. sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ from $\mathcal{R}$
3. $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}')])$

this is a moving target

can lead to unstable optimization

Can we change the target Q-values more slowly?

💡Simple idea: freeze parameters used for the target Q-values, update periodically

Slide adapted from Sergey Levine

How to make Q-learning stable?

Q-learning with replay buffer and target network:

0. save target network parameters: $\phi' \leftarrow \phi$

$N\times$

1. collect data $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy, add it to $\mathcal{R}$

$K\times$

2. sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ from $\mathcal{R}$
3. $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_{\phi'}(\mathbf{s}'_i, \mathbf{a}')])$

Inner loop is doing supervised learning

frozen target network

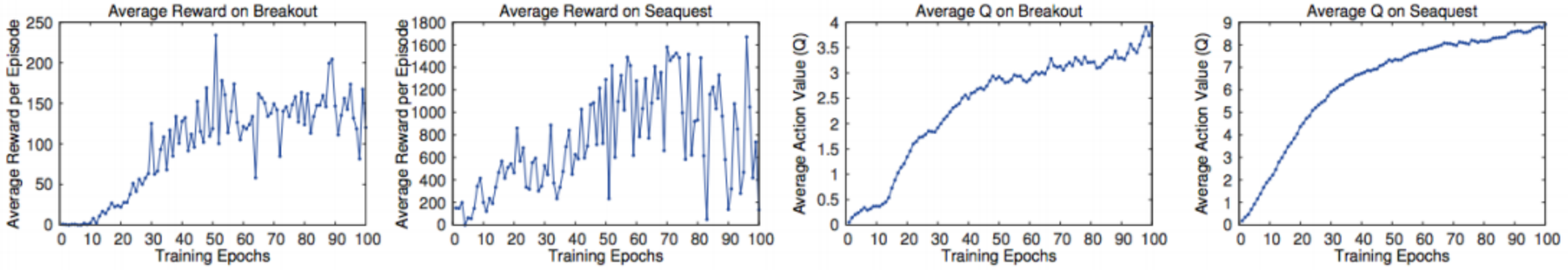
labels don't change in inner loop

Corresponds to DQN algorithm ("deep Q network")

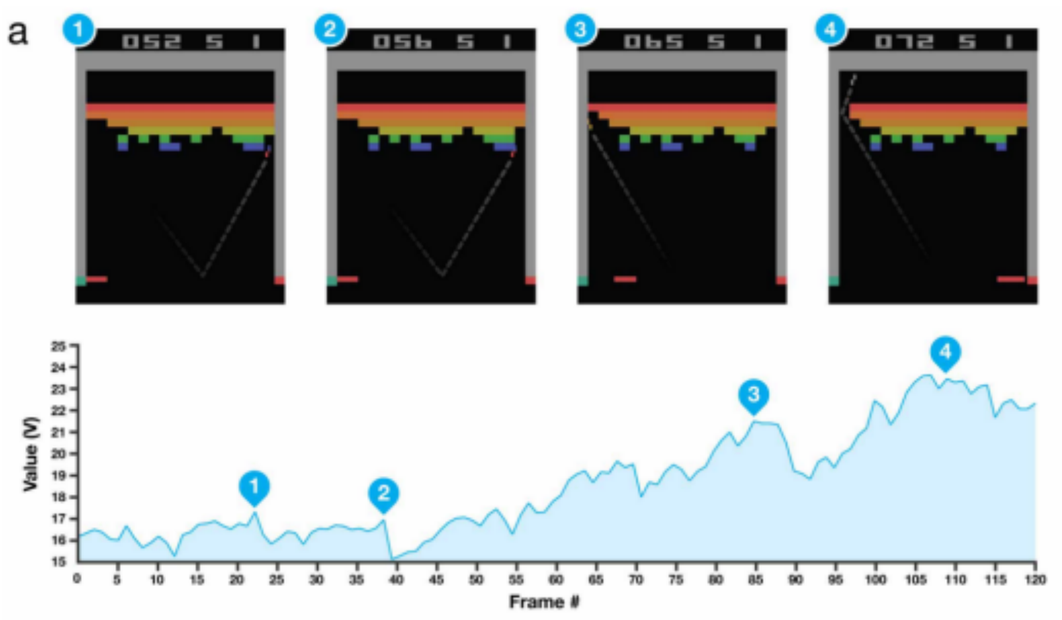
Mnih et al. Playing Atari with Deep Reinforcement Learning. 2013.

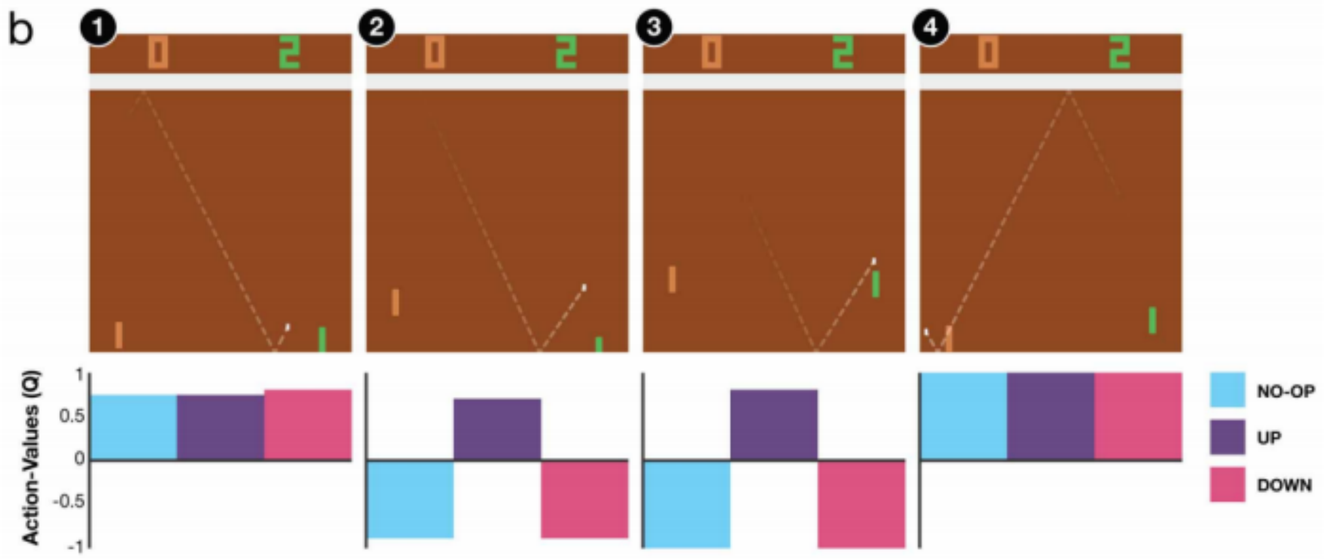
Along with an actor-critic method, you'll implement this algorithm in HW 2!

Are the Q-values accurate?



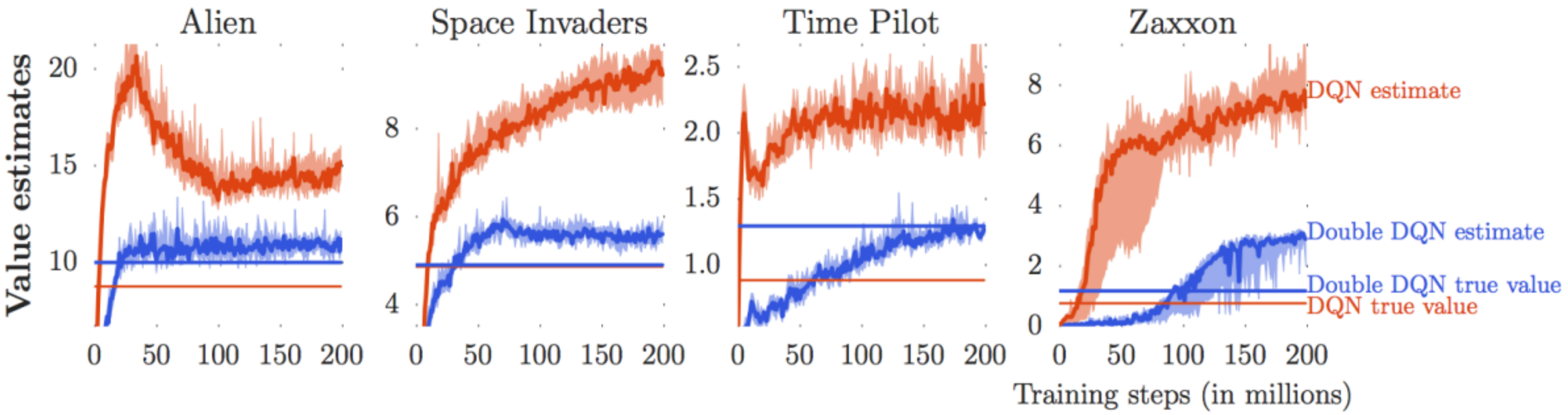
As predicted Q increases, so does the return





Slide adapted from Sergey Levine

Are the Q-values accurate?



Slide adapted from Sergey Levine

Overestimation in Q-learning

target value $y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$

this last term is the problem

$Q_{\phi'}(\mathbf{s}', \mathbf{a}')$ is not perfect – it looks "noisy"

hence $\max_{\mathbf{a}'} Q_{\phi'}(\mathbf{s}', \mathbf{a}')$ *overestimates* the next value!

note that $\max_{\mathbf{a}'} Q_{\phi'}(\mathbf{s}', \mathbf{a}') = \underline{Q_{\phi'}}(\mathbf{s}', \underline{\arg\max_{\mathbf{a}'} Q_{\phi'}(\mathbf{s}', \mathbf{a}')})$

value *also* comes from $Q_{\phi'}$ action selected according to $Q_{\phi'}$

Slide adapted from Sergey Levine

Double Q-learning

note that $\max_{\mathbf{a}'} Q_{\phi'}(\mathbf{s}', \mathbf{a}') = \underline{Q_{\phi'}}(\mathbf{s}', \underline{\arg\max_{\mathbf{a}'} Q_{\phi'}(\mathbf{s}', \mathbf{a}')})$

value *also* comes from $Q_{\phi'}$ action selected according to $Q_{\phi'}$

if the noise in these is decorrelated, the problem goes away!

idea: don't use the same network to choose the action and evaluate value!

"double" Q-learning: use two networks:

$$Q_{\phi_A}(\mathbf{s}, \mathbf{a}) \leftarrow r + \gamma Q_{\phi_B}(\mathbf{s}', \arg\max_{\mathbf{a}'} Q_{\phi_A}(\mathbf{s}', \mathbf{a}'))$$

$$Q_{\phi_B}(\mathbf{s}, \mathbf{a}) \leftarrow r + \gamma Q_{\phi_A}(\mathbf{s}', \arg\max_{\mathbf{a}'} Q_{\phi_B}(\mathbf{s}', \mathbf{a}'))$$

if the two Q's are noisy in *different* ways, there is no problem

Slide adapted from Sergey Levine

Double Q-learning in practice

where to get two Q-functions?

just use the current and target networks!

standard Q-learning: $y = r + \gamma Q_{\phi'}(\mathbf{s}', \arg\max_{\mathbf{a}'} Q_{\phi'}(\mathbf{s}', \mathbf{a}'))$

double Q-learning: $y = r + \gamma Q_{\phi'}(\mathbf{s}', \arg\max_{\mathbf{a}'} Q_{\phi}(\mathbf{s}', \mathbf{a}'))$

just use current network (not target network) to evaluate action

still use target network to evaluate value!

Slide adapted from Sergey Levine

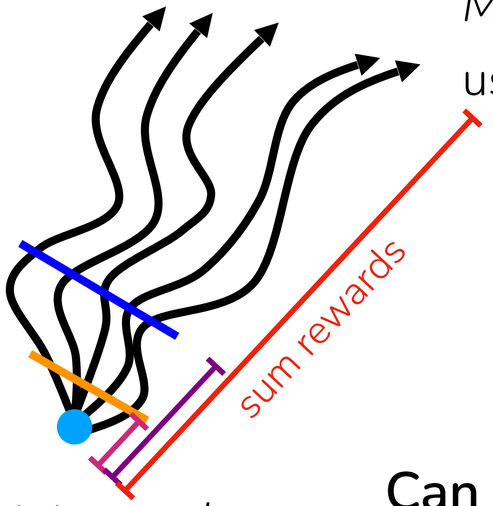
N-step returns?

Q-learning target: $y_{j,t} = r_{j,t} + \gamma \max_{\mathbf{a}_{j,t+1}} Q_{\phi'}(\mathbf{s}_{j,t+1}, \mathbf{a}_{j,t+1})$

these are the only values that matter if $Q_{\phi'}$ is bad!

these values are important if $Q_{\phi'}$ is good

Recall: fitting value functions with rewards + bootstrapped V



Monte Carlo:
using $\sum_t^T r$

Bootstrapped:
using r+V

How about
$\sum_t^{t+n-1} r + V_{t+n}$

-> less variance than MC
-> lower bias than 1-step bootstrap

Can we construct these multi-step targets for Q-learning?

$$y_{j,t} = \sum_{t'=t}^{t+N-1} \gamma^{t-t'} r_{j,t'} + \gamma^N \max_{\mathbf{a}_{j,t+N}} Q_{\phi'}(\mathbf{s}_{j,t+N}, \mathbf{a}_{j,t+N})$$

N-step returns

Q-learning target: $y_{j,t} = r_{j,t} + \gamma \max_{\mathbf{a}_{j,t+1}} Q_{\phi'}(\mathbf{s}_{j,t+1}, \mathbf{a}_{j,t+1})$

these are the only values that matter if $Q_{\phi'}$ is bad!

these values are important if $Q_{\phi'}$ is good

Can we construct these multi-step targets for Q-learning?

N-step target: $y_{j,t} = \sum_{t'=t}^{t+N-1} \gamma^{t-t'} r_{j,t'} + \gamma^N \max_{\mathbf{a}_{j,t+N}} Q_{\phi'}(\mathbf{s}_{j,t+N}, \mathbf{a}_{j,t+N})$

these are the rewards for policy that collected the data

+ (far) less biased target values when Q-values are inaccurate

+ typically faster learning, especially early on

- only actually correct when learning on-policy

(not an issue when N=1!)

Ways to fix? Most commonly: ignore the problem & still use N > 1

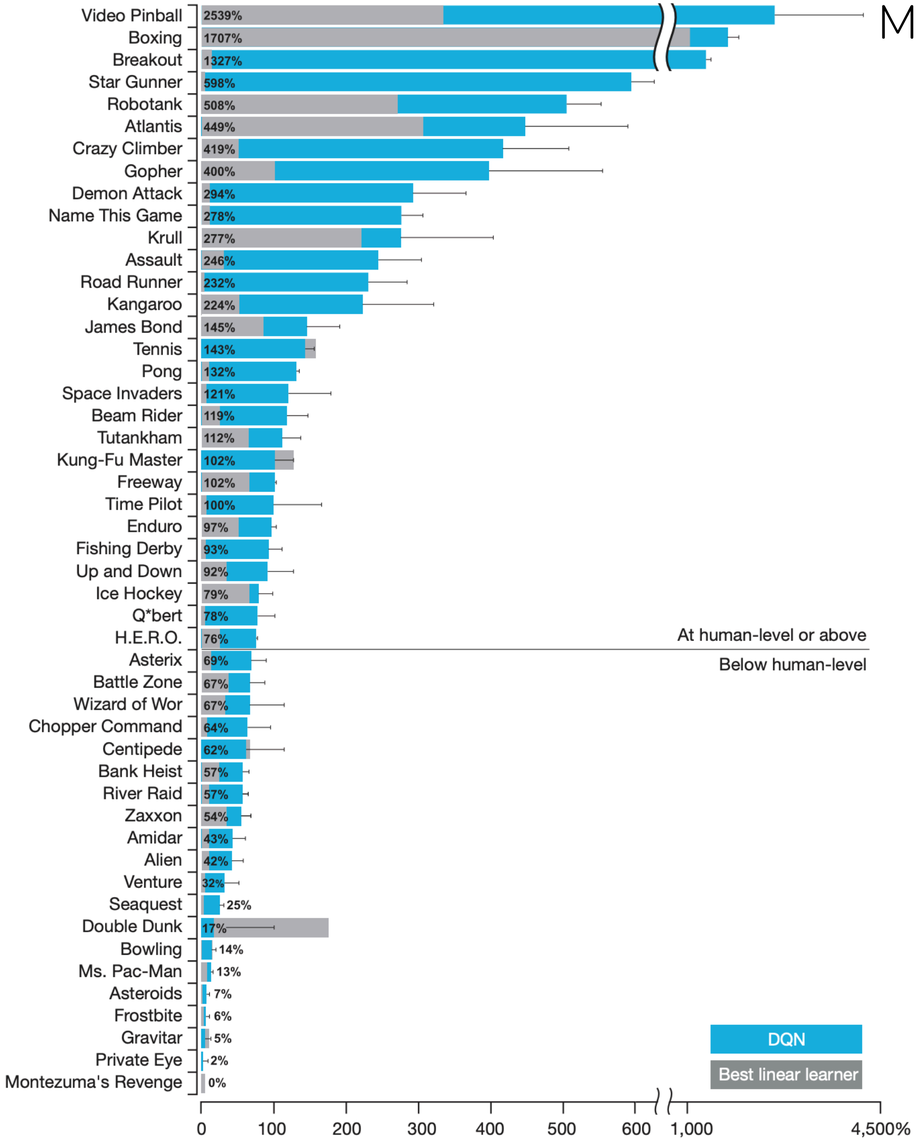
Can also:

- dynamically choose N to only use data that follows current policy (if data mostly on-policy, action space is small)
- use importance sampling

Q-learning in practice?

Q-learning learns to outperform people on most Atari games

Mnih et al. 2015



Q-learning trains robot grasping system

Kalashnikov et al. 2018

Method	Dataset	Test
QT-Opt (ours)	580k off-policy + 28k on-policy	**96%**
Levine et al. [27]	900k grasps from Levine et al. [27]	78%
QT-Opt (ours)	580k off-policy grasps only	87%
Levine et al. [27]	400k grasps from our dataset	67%

When to use one online RL algorithm vs. another?

Chelsea's advice

PPO & variants	When you care about stability, ease-of-use
	When you don't care about data efficiency
DQN & variants	When you have discrete actions or low-dimensional continuous actions
SAC & variants	When you care most about data efficiency
	When you are okay with tuning hyperparameters, less stability

The plan for today

Value-based RL methods

1. Q-learning RL method
    a. Policy iteration
    b. Bellman optimality equation
    c. How to collect data for Q-learning methods
2. Q-learning in practice
    d. Target networks
    e. Double DQN
    f. N-step returns

Key learning goals:

- How Q-functions relate to policies
- How to do RL without learning an explicit policy
- How to stabilize Q-learning in practice

Next time

Done with online RL methods!

Next week: Offline RL (can we learn behavior from existing datasets?)

Reward learning + start of RL for LLMs

Course reminders

- Today
  - **Homework 1** due, **homework 2** out
  - Sending out $100 AWS credits + form for GPU quota.
  - Extra Q-learning section (1:30-2:30 pm)

Very important if you want GPUs, especially for default project