$J(\pi) - J(\mu) = \mathbb{E}_{\pi}[A^{\mu}]$ $\quad \mathbb{E}_{\mu}[A^{\pi}]$

Option 2 for using the perf. difference lemma

use states from μ, & compute $A^{\pi}(s,a)$

$Q^{\pi}(s,a)$ — can do this as we can learn Q-functions

can be estimated using data from any policy

for any policy π,

$\forall s,a: \quad Q^{\pi}(s,a) = r(s,a) + \gamma \mathbb{E}_{\substack{s' \sim p(\cdot|s,a) \\ a' \sim \pi(\cdot|s')}}[Q^{\pi}(s',a')]$

"train a Q-function"

We know how to implement this via ideas like value-based methods.

$Q^{\pi}_{\phi}(s,a) \Rightarrow$

→ use target nets

→ soft/hard target updates

→ everything else remains the same

value-based RL

$\pi_{\theta}(\cdot|s)$

$\max_{\theta} \mathbb{E}_{\mu}[Q^{\pi}_{\phi}(s,a)]$

Loss for training the Q-functions:

$r(s,a) + \gamma \max_{a'} Q(s',a')$

$$\mathcal{L}(\phi) = \mathbb{E}_{s,a,r,s' \sim RBL}\left[\left(Q_\phi(s,a) - y(s,a)\right)^2\right]$$

replay buffer

$$y(s,a) = r(s,a) + \gamma \mathbb{E}_{s',a' \sim \pi}\left[Q^{tgt}_{\bar{\phi}}(s',a')\right]$$

How do we compute this?

$$\mathbb{E}_{\substack{s' \sim p(\cdot|s,a) \\ a' \sim \pi(\cdot|s')}}\left[Q_{\bar{\phi}}(s',a')\right] \approx$$

1) sample $a'_0 \sim \pi_\theta(\cdot|s')$

2) query the tgt net. with it $Q_{\bar{\phi}}(s',a')$

$s' \in RB$

$a' \sim \pi_\theta(\cdot|s')$

How do we do the policy update?

"π_θ"

Policy update:

We ideally want to do:

$$\max_{\pi} \; \mathbb{E}_{\substack{s \sim d^{\mu}(s) \\ a \sim \mu(\cdot|s)}} \left[A^{\pi}(s,a) \right]$$

But there is a problem!!

How would you do gradient updates on π?

$$\theta' \leftarrow \theta + \eta \nabla_{\theta} \mathbb{E}_{s,a \sim \mu} \left[A^{\pi}(s,a) \right]$$

$$\downarrow$$

$$A^{\pi}_{\phi}(s,a)$$

(or $Q^{\pi}_{\phi}(s,a)$)

Another approximation

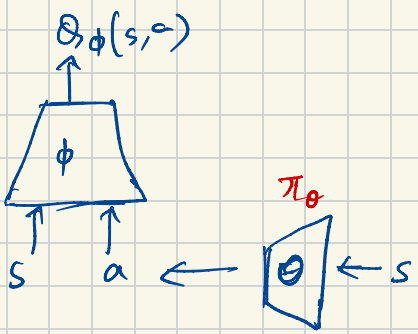
$$\theta' \leftarrow \theta + \eta \nabla_\theta \mathbb{E}_{s \sim d^\mu(s)} \left[Q^\pi_\phi(s, a) \right]$$

$$a \sim \pi_\theta(\cdot|s)$$

Back to something that looks like good, old PG!

But w/o the sum over time, which makes it easy!

How to compute? Just backprop through it!



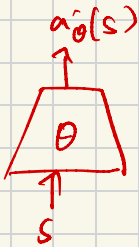
$$\nabla_\theta Q_\phi^\pi(s,a) = \nabla_\theta Q_\phi^\pi(s, a_\theta(s))$$

$$= \nabla_a Q_\phi^\pi(s,a)\Big|_{a=a_\theta(s)} \cdot \nabla_\theta a_\theta(s)$$

what is this exactly?

1) say the policy is deterministic!

DDPG



2) Or it is stochastic!

SAC

$$\pi_\theta(\cdot|s) = \mathcal{N}(\mu_\theta(s), \Sigma_\theta(s))$$

Then $a \sim \pi_\theta(\cdot|s)$ $\Leftrightarrow$ sample $\varepsilon \sim \mathcal{N}(0, I)$ & $a = \mu_\theta(s) + \varepsilon \cdot \sigma_\theta(s)$

How to implement DDPG or its updated variant TD3 correctly?

1) Two Q-networks, Two target Q-networks.

2) Deterministic policy: $\pi_\theta(s)$

Data collection: collect data using

$$a \sim \pi_\theta(s) + \varepsilon$$

↳ noise added to it

Training: sample $s, a, r, s' \sim$ Replay buffer.

$$y(s,a) = r(s,a) + \gamma \min\left(Q^{tgt}_{\phi,1}(s', a'), Q^{tgt}_{\phi,2}(s', a') \right)$$

$$a' \leftarrow \pi_\theta(s') + \varepsilon$$

Train $Q_{\phi,1}$ & $Q_{\phi,2}$. Train π_θ, update target networks.

once every d steps

Another algorithm — Soft Actor-Critic

different from standard RL parameterization

Standard RL:

$$\pi^*_{RL} = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[\sum_t \gamma^t r(s_t, a_t)\right]$$

Max-ent RL:

entropy of action dist.

$$\pi^*_{maxent} = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) + \mathcal{H}(\pi)\right)\right]$$

We can simplify the maxent RL objective!

$$H(\pi(\cdot|s_t)) = \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s_t)}\left[-\log \pi(\cdot|s_t)\right]$$

$$\pi^*_{\text{maxent}} = \max_{\pi} \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) + \alpha \mathop{\mathbb{E}}_{a_t\sim\pi}\left[-\log \pi\right]\right)\right]$$

Can move this here!!

$$\mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) - \alpha \log \pi(a_t|s_t)\right)\right]$$

"backup the entropy"

modified reward

$$\bar{r}_\alpha(s, a) = r(s, a) - \alpha \log \pi(a|s)$$

SAC := TD3 + Max-ent RL

(roughly, very roughly!!)

Q-function:

$$Q(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{a' \sim \pi}\left[Q(s',a')\right]$$

⇓

$$Q(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{a' \sim \pi}\left[Q(s',a') - \alpha \log \pi(a'|s')\right]$$

Policy extraction:

π_θ

$$\theta' \leftarrow \theta + \eta \nabla_\theta \mathbb{E}_{a \sim \pi_\theta(\cdot|s)} \left[Q_\phi(s,a) \right]$$

$$\Downarrow$$

$$\theta' \leftarrow \theta + \eta \nabla_\theta \mathbb{E}_{a \sim \pi_\theta(\cdot|s)} \left[Q_\phi(s,a) - \alpha \log \pi_\theta(a|s) \right]$$

Why use entropy in here?

Practical design choices

(1) Use two critic networks

$$\min(Q_1, Q_2)$$

(2) No noise during exploration Just sample from the stochastic policy

(3) α is dynamically tuned!

Why does adding the entropy help?

→ Stochastic policy is easier to optimize

→ More ways to succeed !!