清華大學
Tsinghua University

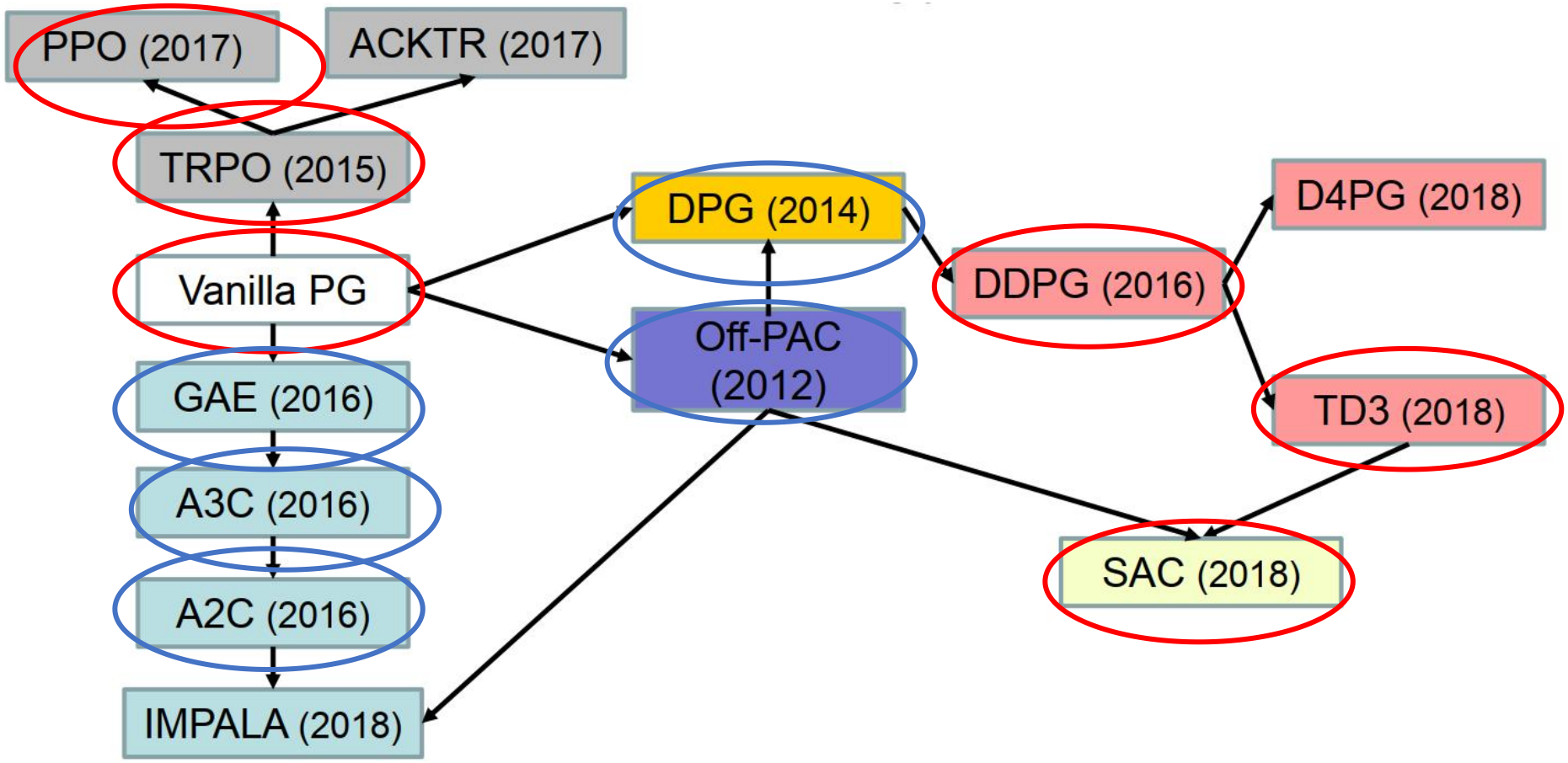
Deep Reinforcement Learning

Lecture 5: Actor-Critic

Huazhe Xu

Tsinghua University

AI This Week


PPO (2017)
ACKTR (2017)
TRPO (2015)
Vanilla PG
GAE (2016)
A3C (2016)
A2C (2016)
IMPALA (2018)
DPG (2014)
Off-PAC (2012)
DDPG (2016)
D4PG (2018)
TD3 (2018)
SAC (2018)

Alert: In these 2–3 lectures

- Toward practical algorithms
- Less intuitive, more mathy
- Less funny pictures, more equations
- Be prepared ☺

In Lec5

1. Policy Gradient Ctd
2. Off—Policy Policy Gradient
3. Actor—Critic Algorithms

In Lec5

1. Policy Gradient Ctd
2. Off-Policy Policy Gradient
3. Actor-Critic Algorithms

Baseline with REINFORCE

1. We all know MC causes high _____(variance/bias).
   - Why variance is reduced?
2. What if $R(\tau) = 0$ but actually it is the greatest trajectory?

We noticed a great property:

$$\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s, a) B(s)] = 0$$

$$\nabla_\theta \mathbb{E}_\tau[R(\tau)] = \mathbb{E}_\tau \left[(R(\tau) - B) \nabla_\theta \sum_{t=0}^{T-1} \log \pi(a_t \mid s_t, \theta) \right]$$

If you do not know the math, here is the proof:

Recall that all probability distributions are normalized:

$$\int_x P_\theta(x) = 1$$

Take the gradient of both sides of the normalization condition:

$$\nabla_\theta \int_x P_\theta(x) = \nabla_\theta 1 = 0$$

Use the log derivative trick to get:

$$\begin{aligned} 0 &= \nabla_\theta \int_x P_\theta(x) \\ &= \int_x \nabla_\theta P_\theta(x) \\ &= \int_x P_\theta(x) \nabla_\theta \log P_\theta(x) \end{aligned}$$

$$\mathbb{E}[\nabla \log \pi(a \mid s) \cdot b(s)] = b(s) \cdot \mathbb{E}[\nabla \log \pi(a \mid s)] = 0,$$

By subtracting $V(s)$, the updates focus on the relative benefit of actions rather than absolute returns, which are often noisy. This reduces the magnitude of the updates and centers them around zero, lowering variance.

In Lec5

1. Policy Gradient Ctd
2. Off-Policy Policy Gradient
3. Actor-Critic

We want to use some else'experience.

Including the past of yourself

Importance Sampling

- **Importance sampling** is a **Monte Carlo method** for evaluating properties of a particular **distribution**, while only having samples generated from a different distribution than the distribution of interest. （Wikipedia）
- $f(x)$, $where\ x \sim p(x)$:

$$E[f(x)] = \int f(x)p(x)dx \approx \frac{1}{n}\sum_i f(x_i)$$

$$E[f(x)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx \approx \frac{1}{n}\sum_i f(x_i)\frac{p(x_i)}{q(x_i)}$$

Off-Policy PG Objective

- Use samples from another policy $\bar{p}(\tau)$

$$J(\theta) = E_{\tau \sim \bar{p}(\tau)} \left[\frac{p_\theta(\tau)}{\bar{p}(\tau)} r(\tau) \right]$$

$$\frac{p_\theta(\tau)}{\bar{p}(\tau)} = \frac{p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}{p(\mathbf{s}_1) \prod_{t=1}^{T} \bar{\pi}(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)} = \frac{\prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\prod_{t=1}^{T} \bar{\pi}(\mathbf{a}_t|\mathbf{s}_t)}$$

Off—Policy PG

$$\theta^\star = \arg\max_\theta J(\theta) \quad J(\theta) = E_{\tau\sim p_\theta(\tau)}[r(\tau)]$$

$$\begin{aligned}
\nabla_{\theta'} J(\theta') &= E_{\tau\sim p_\theta(\tau)}\left[\frac{p_{\theta'}(\tau)}{p_\theta(\tau)}\nabla_{\theta'}\log\pi_{\theta'}(\tau)r(\tau)\right] \\
&= E_{\tau\sim p_\theta(\tau)}\left[\left(\prod_{t=1}^{T}\frac{\pi_{\theta'}(\mathbf{a}_t\mid\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t\mid\mathbf{s}_t)}\right)\left(\sum_{t=1}^{T}\nabla_{\theta'}\log\pi_{\theta'}(\mathbf{a}_t\mid\mathbf{s}_t)\right)\left(\sum_{t=1}^{T}r(\mathbf{s}_t,\mathbf{a}_t)\right)\right] \\
&= E_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T}\nabla_{\theta'}\log\pi_{\theta'}(\mathbf{a}_t\mid\mathbf{s}_t)\left(\prod_{t'=1}^{t}\frac{\pi_{\theta'}(\mathbf{a}_{t'}\mid\mathbf{s}_{t'})}{\pi_\theta(\mathbf{a}_{t'}\mid\mathbf{s}_{t'})}\right)\left(\sum_{t'=t}^{T}r(\mathbf{s}_{t'},\mathbf{a}_{t'})\right)\right]
\end{aligned}$$

Future actions do not affect current weight

In Lec5

1. Policy Gradient Ctd
2. Off-Policy Policy Gradient
3. Actor-Critic Algorithms

Actor Critic

Someone evaluates.

Someone acts.

Policy Gradient

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

PG with Baseline

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} \mid \mathbf{s}_{i,t})(Q(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) - V(\mathbf{s}_{i,t}))$$

$$V^\pi(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)]$$

$$A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)$$

This baseline can be the critic!

How to get the value?

- Policy evaluation
  - MC-based, need to reset
- TD learning

Batch Actor–Critic Algorithm

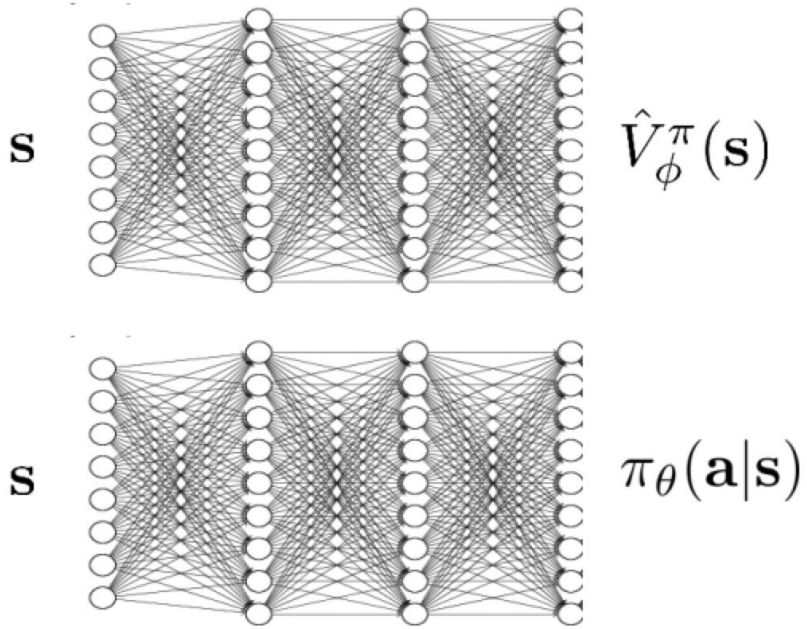
1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a} \mid \mathbf{s})$ (run it on the robot)
2. fit $\hat{V}^\pi_\phi(\mathbf{s})$ to sampled reward sums
3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i \mid \mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

What if we need it to be online and with discount? (use incomplete sequences to update)

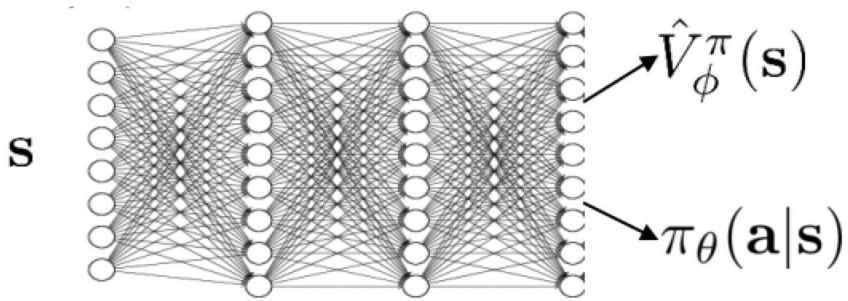
1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a} \mid \mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
2. update $\hat{V}_\phi^\pi$ using target $r + \gamma \hat{V}_\phi^\pi(\mathbf{s}')$
3. evaluate $\hat{A}^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}') - \hat{V}_\phi^\pi(\mathbf{s})$
4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(\mathbf{a} \mid \mathbf{s}) \hat{A}^\pi(\mathbf{s}, \mathbf{a})$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Design Choices in Actor Critic

two network design

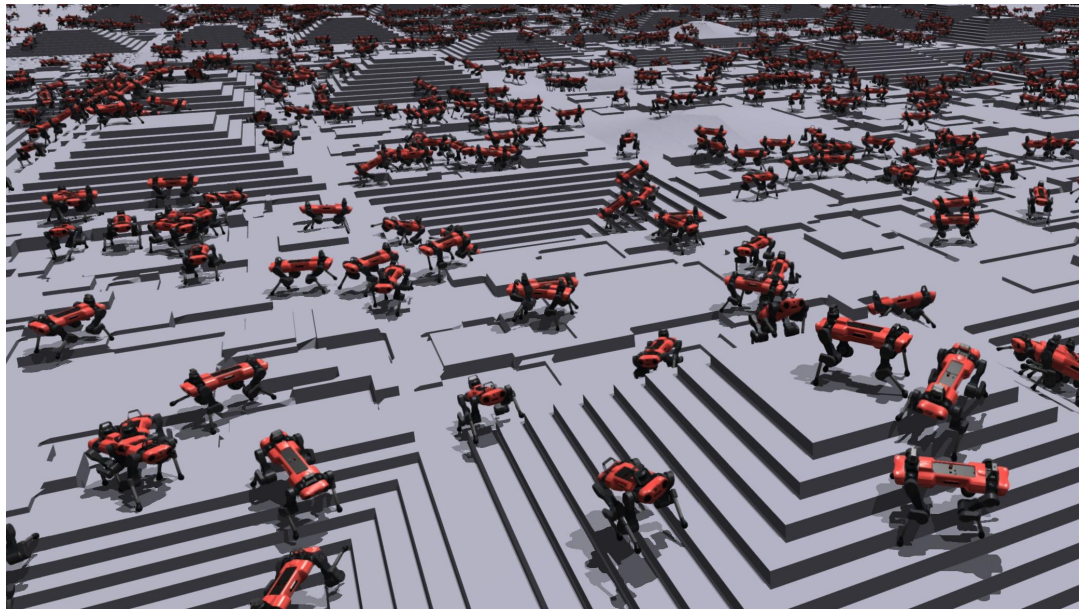


shared network design



For PG, using one sample to perform GD is usually not a good idea

- Can you have multiple samples in some way?



Asynchronous Actor Critic

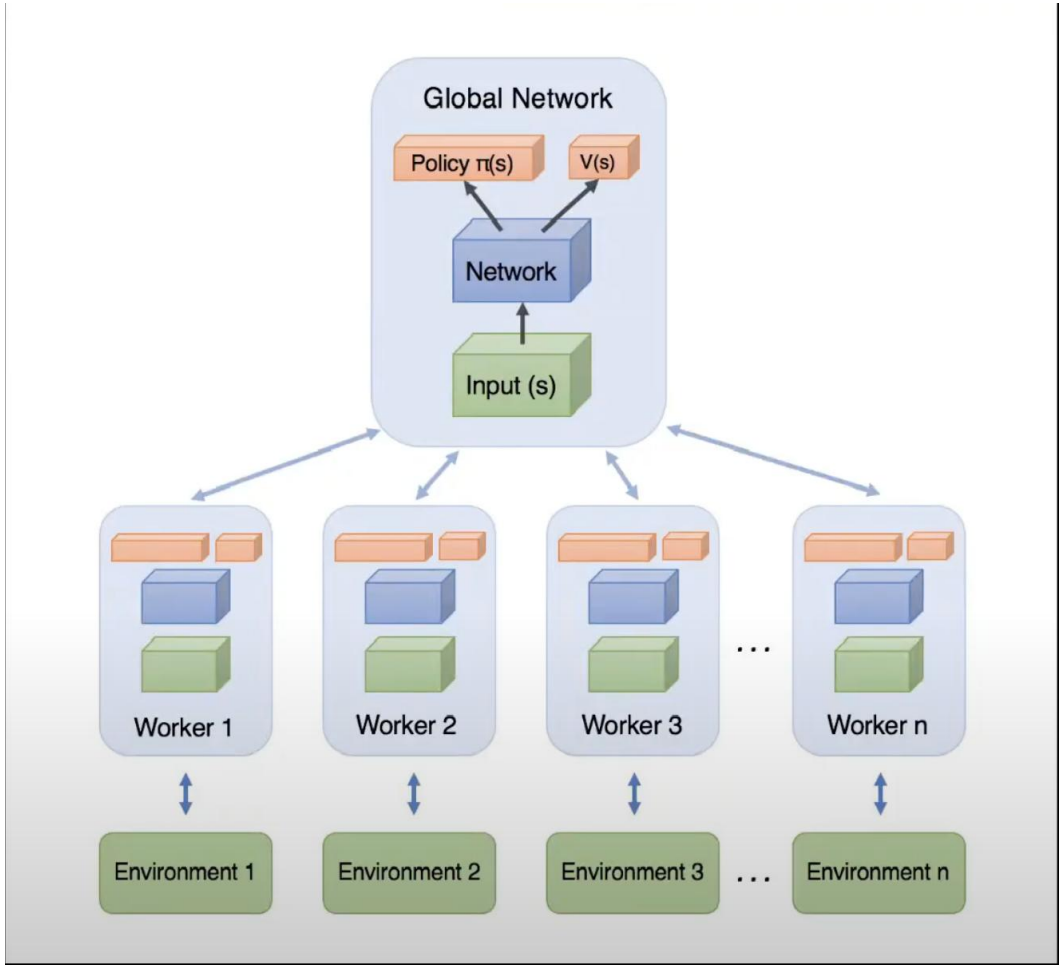


Image Source: Python Lessons

Reference: Asynchronous Methods for Deep Reinforcement Learning

Algorithm Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ and θ_v and global shared counter $T = 0$

// Assume thread-specific parameter vectors θ' and θ'_v

Initialize thread step counter $t \leftarrow 1$

repeat

Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.

Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$

$t_{\text{start}} = t$

Get state s_t

repeat

Perform a_t according to policy $\pi\left(a_t \mid s_t; \theta'\right)$

Receive reward r_t and new state s_{t+1}

$t \leftarrow t + 1$

$T \leftarrow T + 1$

until terminal s_t or $t - t_{\text{start}} == t_{\max}$

$$R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$$

for $i \in \{t-1, \ldots, t_{\text{start}}\}$ do

$R \leftarrow r_i + \gamma R$

Accumulate gradients wrt θ' : $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi\left(a_i \mid s_i; \theta'\right)\left(R - V\left(s_i; \theta'_v\right)\right)$

Accumulate gradients wrt θ'_v : $d\theta_v \leftarrow d\theta_v + \partial\left(R - V\left(s_i; \theta'_v\right)\right)^2 / \partial \theta'_v$

end for

Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.

until $T > T_{\max}$

Instead of multiple workers, can we use our own history?

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a} \mid \mathbf{s})$, 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 $\hat{V}^\pi_\phi$ using targets $y_i = r_i + \gamma \hat{V}^\pi_\phi(\mathbf{s}'_i)$ for each $\mathbf{s}_i$
4. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
5. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i \mid \mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
6. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

There are non-on-policy data!

How to deal with V?

3. update $\hat{V}_\phi^\pi$ using targets $y_i = r_i + \gamma \hat{V}_\phi^\pi(\mathbf{s}_i')$ for each $\mathbf{s}_i$

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

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

No assumption

3. update $\hat{Q}_\phi^\pi$ using targets $y_i = r_i + \gamma \hat{V}_\phi^\pi(\mathbf{s}_i')$ for each $\mathbf{s}_i, \mathbf{a}_i$

$$= r_i + \gamma \hat{Q}_\phi^\pi(\mathbf{s}_i', \mathbf{a}_i')$$

Action from the current policy, not the buffer

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

How to deal with π?

$$5.\ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i \mid \mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$$

$$\text{sample } \mathbf{a}_i^\pi \sim \pi_\theta(\mathbf{a} \mid \mathbf{s}_i)$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i^\pi \mid \mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i^\pi)$$

$$\text{in practice: } \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i^\pi \mid \mathbf{s}_i) \hat{Q}^\pi(\mathbf{s}_i, \mathbf{a}_i^\pi)$$

High-variance!

What is still wrong?

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a} \mid \mathbf{s})$, 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 $y_i = r_i + \gamma \hat{Q}^\pi_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$ for each $\mathbf{s}_i, \mathbf{a}_i$
4. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}^\pi_i \mid \mathbf{s}_i) \hat{Q}^\pi(\mathbf{s}_i, \mathbf{a}^\pi_i)$ where $\mathbf{a}^\pi_i \sim \pi_\theta(\mathbf{a} \mid \mathbf{s}_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

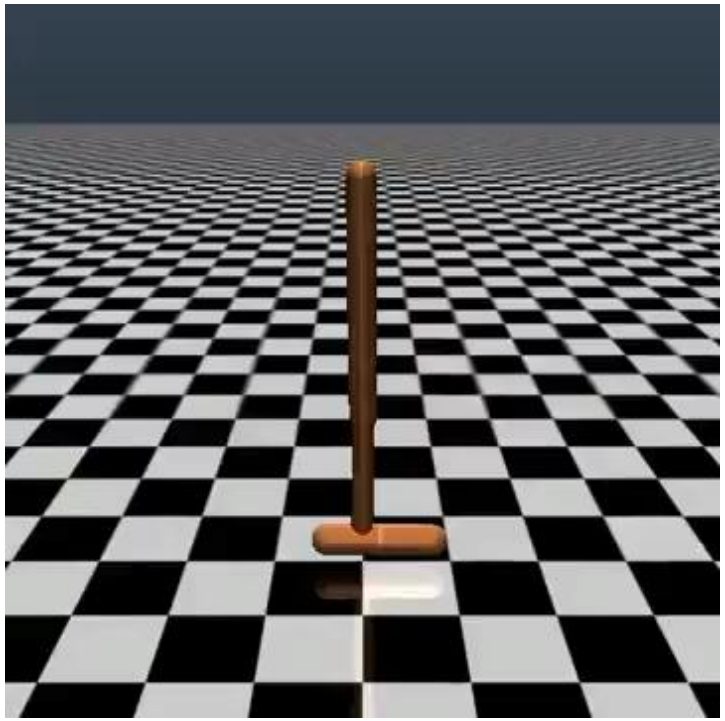
The state distribution! But it is fine.

An example paper that builds upon this.

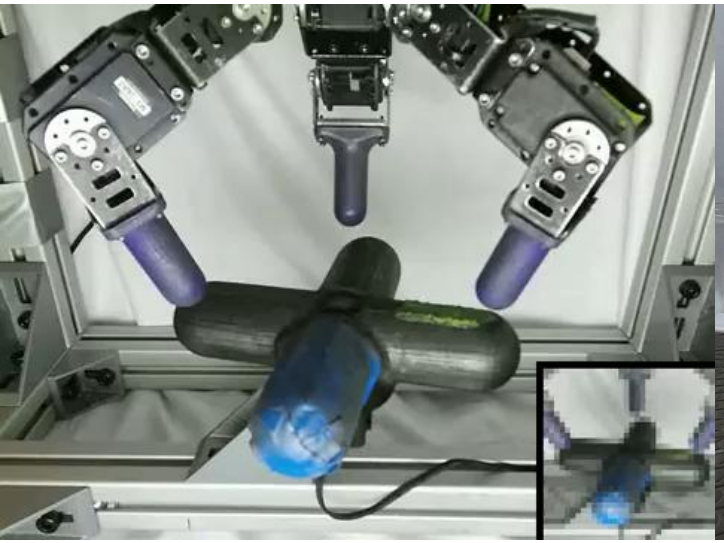
Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor — 4934 — 2018

T Haarnoja, A Zhou, P Abbeel, S Levine

International conference on machine learning, 1861-1870



Many projects are based on SAC.



Recap

- We learn the second category of RL algorithm: PG.
- We have to calculate gradient of the return against policy parameters. (A bunch of math tricks.)
- It has high variance. We need to somehow reduce it: baseline and causality.
- A detour: Off-policy PG with importance sampling
- Using the learned value function as your baseline: actor-critic

What can be still wrong (improved) about current algorithms?

- Unstable
  - Large step → bad policy
  - Next batch is generated from current bad policy → collect bad samples
  - Bad samples → worse policy
  - (compare to supervised learning: the correct label and data in the following batches may correct it)
  - If step size is too small: the learning process is slow
- We may not always want stochastic policies.
- Distributed training

What we have learned in RL?

- Value—based: Deep Q Learning
- Policy—based: Policy Gradient
- Hybrid: Actor—Critic
- We even invented off—policy versions of them!
- These methods are great, but we can be finicky!

Hey DQN, can we have continuous action?

$$\pi(\mathbf{a}_t \mid \mathbf{s}_t) = \begin{cases} 1 \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q_\phi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 \text{ otherwise} \end{cases}$$

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

- It is hard to deal with continuous action!

Solution 1: Sample and Optimize

- Sample N actions from some distribution

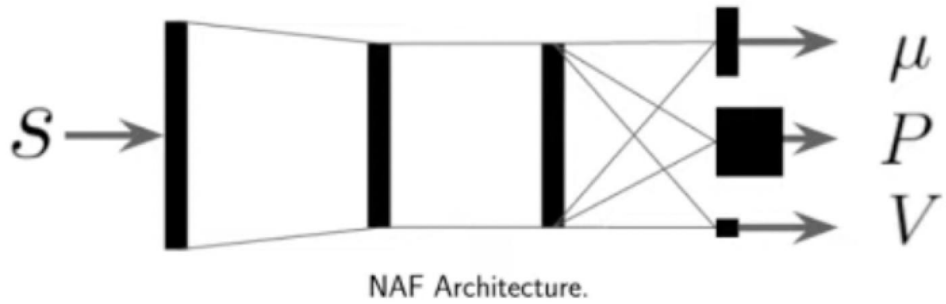
$$\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a}) \approx \max\{Q(\mathbf{s}, \mathbf{a}_1), \ldots, Q(\mathbf{s}, \mathbf{a}_N)\}$$

  - Simple, efficient
  - I say it is bad. Why?

- Cross-entropy method (CEM)
  - Iteratively optimize
- CMA-ES
  - https://en.wikipedia.org/wiki/CMA-ES

Solution 2: Better Q function structure

$$Q_\phi(\mathbf{s}, \mathbf{a}) = -\frac{1}{2}(\mathbf{a} - \mu_\phi(\mathbf{s}))^T P_\phi(\mathbf{s})(\mathbf{a} - \mu_\phi(\mathbf{s})) + V_\phi(\mathbf{s})$$



NAF Architecture.

$$\arg\max_{\mathbf{a}} Q_\phi(\mathbf{s}, \mathbf{a}) = \mu_\phi(\mathbf{s}) \quad \max_{\mathbf{a}} Q_\phi(\mathbf{s}, \mathbf{a}) = V_\phi(\mathbf{s})$$

Less expressive!

Gu et al. ICML 2016

Solution 3: Gradient Descent (Ascent)

- We perform GD/GA for every step.
- It is bad. Why?
  - It is inefficient.

Solution 4: Learn an Approximate Maximizer

$$\max_{\mathbf{a}} Q_\phi(\mathbf{s}, \mathbf{a}) = Q_\phi\left(\mathbf{s}, \arg\max_{\mathbf{a}} Q_\phi(\mathbf{s}, \mathbf{a})\right)$$

- Big idea: Train a neural network $\mu_\theta(s)$ such that $\mu_\theta(s) = arg\max_a Q(s, a)$

$$\frac{dQ_\phi}{d\theta} = \frac{d\mathbf{a}}{d\theta}\frac{dQ_\phi}{d\mathbf{a}}$$

$$y_j = r_j + \gamma Q_{\phi'}\left(\mathbf{s}'_j, \mu_\theta\left(\mathbf{s}'_j\right)\right)$$

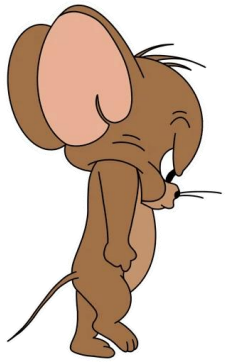
Deep Deterministic Policy Gradient (DDPG)

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$, add it to $\mathcal{B}$
2. sample mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from $\mathcal{B}$ uniformly
3. compute $y_j = r_j + \gamma Q_{\phi'}\left(\mathbf{s}'_j, \mu_{\theta'}\left(\mathbf{s}'_j\right)\right)$ using target nets $Q_{\phi'}$ and $\mu_{\theta'}$
4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$
5. $\theta \leftarrow \theta + \beta \sum_j \frac{d\mu}{d\theta}(\mathbf{s}_j) \frac{dQ_\phi}{d\mathbf{a}}(\mathbf{s}_j, \mu(\mathbf{s}_j))$
6. update ϕ' and θ' (e.g., Polyak averaging)

Lillicrap et al, 2016

DDPG can be improved

- It suffers similar problems as in DQN
  - Overestimation
- The critics might be unstable.



- The critic weirdly prefers some action but not their neighbor actions. Strange landscape.

Twin Delayed DDPG (TD3)

- Solution 1: Clipped Double Q Learning
  - Toward addressing overestimation
  - We have two Qs. And we calculate min of them as my Q.

$$y(r, s', d) = r + \gamma(1-d) \min_{i=1,2} Q_{\phi_{i,\text{targ}}}\left(s', a_{\text{TD3}}(s')\right)$$

  - Why don't we use original double Q?

Fujimoto et al., ICML 2018

Twin Delayed DDPG (TD3)

- Solution 2: Delayed Policy Updates
  - Toward addressing unstable critics
  - Lower the frequency of actor updates.

Fujimoto et al, ICML 2018

Twin Delayed DDPG (TD3)

- Solution 3: Target Policy Smoothing
  - Toward addressing strange landscape
  - Add noise to the actions to smooth the value

$$a_{\mathrm{TD3}}(s') = \mathbf{clip}\left(\mu_{\theta,\mathrm{targ}}(s') + \mathbf{clip}(\epsilon, -c, c), a_{\mathrm{low}}, a_{\mathrm{high}}\right)$$

Fujimoto et al., ICML 2018

Algorithm 1 Twin Delayed DDPG

Input: initial policy parameters θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\theta_{\text{targ}} \leftarrow \theta$, $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$
repeat
  Observe state s and select action $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{Low}, a_{High})$, where $\epsilon \sim \mathcal{N}$
  Execute a in the environment
  Observe next state s', reward r, and done signal d to indicate whether s' is terminal
  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
  If s' is terminal, reset environment state.
  if it's time to update **then**
    for j in range(however many updates) **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute target actions

$$a'(s') = \text{clip}\left(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{Low}, a_{High}\right), \qquad \epsilon \sim \mathcal{N}(0, \sigma)$$

      Compute targets

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', a'(s'))$$

Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_{\phi_i}(s,a) - y(r,s',d) \right)^2 \qquad \text{for } i = 1,2$$

if j mod `policy_delay` $= 0$ **then**

Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_{\phi_1}(s, \mu_\theta(s))$$

Update target networks with

$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1-\rho)\phi_i \qquad \text{for } i = 1,2$$
$$\theta_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1-\rho)\theta$$

end if

end for

end if

until convergence

In next lecture, you will

- learn more advanced & practical RL algorithms
- More information about midterm quiz time (around week 8–10)

Thank you!