UC San Diego
JACOBS SCHOOL OF ENGINEERING

ECE/SIOC 228 Machine Learning for Physical Applications
Lecture 18: Deep Reinforcement Learning III

Yuanyuan Shi

Assistant Professor, ECE

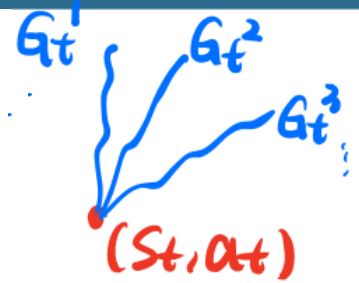
University of California, San Diego

Reference & Acknowledgement: David Silver's Course on RL https://www.davidsilver.uk/teaching/
Sergey Levine, Deep Reinforcement Learning https://rail.eecs.berkeley.edu/deeprlcourse/

REINFORCE Algorithm

$$\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t\right]$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) G_t^i$$



REINFORCE with Baseline

$$\nabla J(\theta) = E_{\tau \sim p_\theta(\tau)}\left[\sum_{t=0}^{T-1} \nabla \log \pi_\theta(a_t|s_t)\,(G_t - b(s_t))\right]$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\,(G_t^i - b(s_t^i))$$

- Substracting a baseline $b(s_t)$ is unbiased in expectation
- Setting $b(s_t) = V^{\pi_\theta}(s_t)$ leads to good variance reduction

Outline

UC San Diego

- State value function fitting $v_\pi(s)$
  - **Classic methods: discrete state space**
  - Value function approximation: continuous state space
- From value function to actor critic
- Actor-critic method variants

State Value Function $v_\pi(s)$

UC San Diego

The **state-value function $v_\pi(s)$ of a policy π** in an MDP is the expected return starting from state s, and then following policy π

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

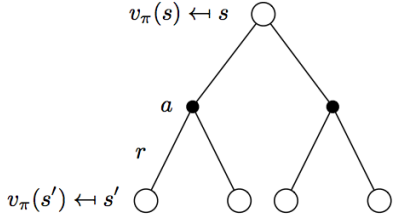
- **The return G_t** is the total discounted reward from time step t

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- Given a policy π, computing its value function $v_\pi(s)$ is called **policy evaluation**

State Value Function $v_\pi(s)$

UC San Diego



- For small MDP, we can solve a linear system to compute $v_\pi(s)$

$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$

- Iterative methods for large MDPs,

$$v_\pi^{(k+1)}(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi^{(k)}(s') \right)$$

$v_\pi^{(1)} \rightarrow v_\pi^{(2)} \rightarrow \cdots \rightarrow v_\pi$ (contraction mapping → convergence)

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

- Both methods need access to the true MDP model: $P_{ss'}^a$, $\mathcal{R}_s^a$
- **When we don't have the true MDP model, how to compute $v_\pi(s)$?** ← use samples!

First-Visit Monte Carlo

UC San Diego

Initialize $\boldsymbol{N(s) = 0, G(s) = 0, \forall s \in S}$; A policy π to be evaluated

For *i = 1, 2, …, N*

- Sample episode $i = s_0^i, a_0^i, r_1^i, s_1^i, a_1^i, r_2^i, \dots, s_{T-1}^i, a_{T-1}^i, r_T^i, s_T^i$
- Define $G_t^i = r(s_t^i, a_t^i) + \gamma\, r(s_{t+1}^i, a_{t+1}^i) + \gamma^2 r(s_{t+2}^i, a_{t+2}^i) + \cdots \gamma^{T-t-1} r(s_{T-1}^i, a_{T-1}^i) = \sum_{t'=t}^{T-1} \gamma^{t'-t} r(s_t^i, a_t^i)$ as return from time step *t* onwards in *i*th episode
- For each state ***s*** visited in episode i
  - **For first time t that state s is visited in episode *i***
  - Increment counter of total first visits: *N(s) = N(s) + 1*
  - Increment total return $G(s) = G(s) + G_t^i$
  - Update estimate $v_\pi(s) = \frac{G(s)}{N(s)}$

Every-Visit Monte Carlo

UC San Diego

Initialize $\boldsymbol{N(s) = 0, G(s) = 0, \forall s \in S}$; A policy π to be evaluated

For *i = 1, 2, …, N*

- Sample episode $i = s_0^i, a_0^i, r_1^i, s_1^i, a_1^i, r_2^i, \ldots, s_{T-1}^i, a_{T-1}^i, r_T^i, s_T^i$
- Define $G_t^i = r(s_t^i, a_t^i) + \gamma\, r(s_{t+1}^i, a_{t+1}^i) + \gamma^2 r(s_{t+2}^i, a_{t+2}^i) + \cdots \gamma^{T-t-1} r(s_{T-1}^i, a_{T-1}^i) = \sum_{t'=t}^{T-1} \gamma^{t'-t} r(s_t^i, a_t^i)$ as return from time step *t* onwards in *i*th episode
- For each state ***s*** visited in episode i
  - **For every time t that state s is visited in episode $\boldsymbol{i}$**
  - Increment counter of total visits: *N(s) = N(s) + 1*
  - Increment total return $G(s) = G(s) + G_t^i$
  - Update estimate $v_\pi(s) = \frac{G(s)}{N(s)}$

Incremental Mean

UC San Diego

The mean $\mu 1, \mu 2, \ldots$ of a sequence $x_1, x_2, \ldots$ can be computed incrementally,

$$\mu_k = \frac{1}{k}\sum_{j=1}^{k} x_j$$

$$= \frac{1}{k}(x_k + \sum_{j=1}^{k-1} x_j)$$

$$= \frac{1}{k}(x_k + (k-1)\mu_{k-1})$$

$$= \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})$$

previous estimate

mismatch between data x_k and previous estimate μ_{k-1}

Incremental (First or Every Visit) Monte Carlo

UC San Diego

Initialize $\boldsymbol{N(s) = 0, G(s) = 0, \forall s \in S}$; A policy π to be evaluated

For *i = 1, 2, …, N*

- Sample episode $i = s_0^i, a_0^i, r_1^i, s_1^i, a_1^i, r_2^i, \ldots, s_{T-1}^i, a_{T-1}^i, r_T^i, s_T^i$
- Define $G_t^i = r(s_t^i, a_t^i) + \gamma\, r(s_{t+1}^i, a_{t+1}^i) + \gamma^2 r(s_{t+2}^i, a_{t+2}^i) + \cdots \gamma^{T-t-1} r(s_{T-1}^i, a_{T-1}^i) = \sum_{t'=t}^{T-1} \gamma^{t'-t} r(s_t^i, a_t^i)$ as return from time step *t* onwards in *i*th episode
- **For each state *s* visited (or first visit) in episode *i***
  - Increment counter of total visits: *N(s) = N(s) + 1*
  - Update estimate

$$v_\pi(s) \leftarrow v_\pi(s) + \alpha\left(G_t^i - v_\pi(s)\right)$$

$\alpha = \frac{1}{N(s)}$: identical to every visit or first visit MC

$\alpha > \frac{1}{N(s)}$: more weights on new data, help for non-stationary domains

Temporal-Difference Learning

UC San Diego

Goal: learn $v_\pi(s)$ online from data for a given policy π

Incremental every-visit Monte-Carlo: Update value $v_\pi(s_t)$ towards actual return G_t

$$v_\pi(s_t) \leftarrow v_\pi(s_t) + \alpha\big(G_t - v_\pi(s_t)\big)$$

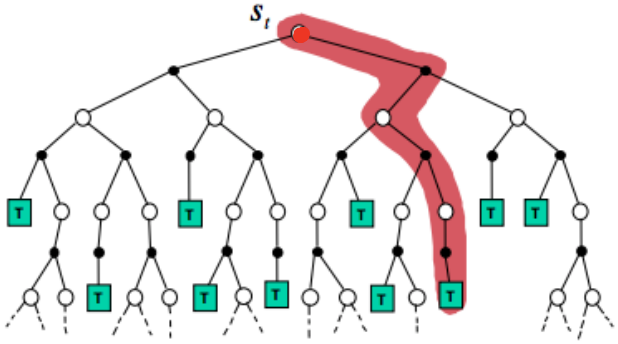
Temporal-difference learning algorithm: TD(0): Update value $v_\pi(s_t)$ towards the estimated return $r(s_t, a_t) + \gamma v_\pi(s_{t+1})$

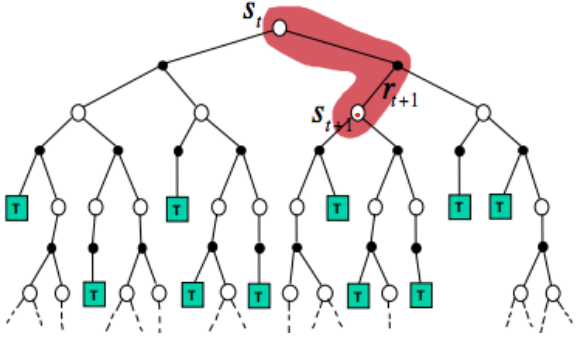
$$v_\pi(s_t) \leftarrow v_\pi(s_t) \ + \alpha\big(r(s_t, a_t) + \gamma v_\pi(s_{t+1}) - v_\pi(s_t)\big)$$

- $r(s_t, a_t) + \gamma v_\pi(s_{t+1})$ is called the **TD target**
- $\Delta_t = r(s_t, a_t) + \gamma v_\pi(s_{t+1}) - v_\pi(s_t)$ is called the **TD error**

Comparing Monte Carlo with TD

UC San Diego





MC does not exploit Markov property

- Must wait until the end of episode before return is known
- Can only learn from complete episode and episodic tasks
- Usually more effective in non-Markov environments

$$v_\pi(s_t) \leftarrow v_\pi(s_t) + \alpha\big(G_t - v_\pi(s_t)\big)$$

$$v_\pi(s_t) \leftarrow v_\pi(s_t) \; + \alpha\big(r(s_t, a_t) + \gamma v_\pi(s_{t+1}) - v_\pi(s_t)\big)$$

TD exploits Markov property:

- Can learn online after each step
- Can learn from incomplete episode and continuing tasks
- Usually more efficient in Markov environments

(Optional) TD(λ)

UC San Diego

- For a given policy π, consider the following *n*-step returns for $n = 1, 2, \ldots, \infty$:
  - n = 1 (TD) $G_t^{(1)} = r(s_t, a_t) + \gamma v_\pi(s_{t+1})$
  - n = 2 $G_t^{(2)} = r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 v_\pi(s_{t+2})$
  - ⋮ ⋮
  - n = ∞ or $T - t$ (MC) $G_t^{(\infty)} = r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \cdots + \gamma^{T-1-t} r(s_{T-1}, a_{T-1})$

- Define the *n-step* return

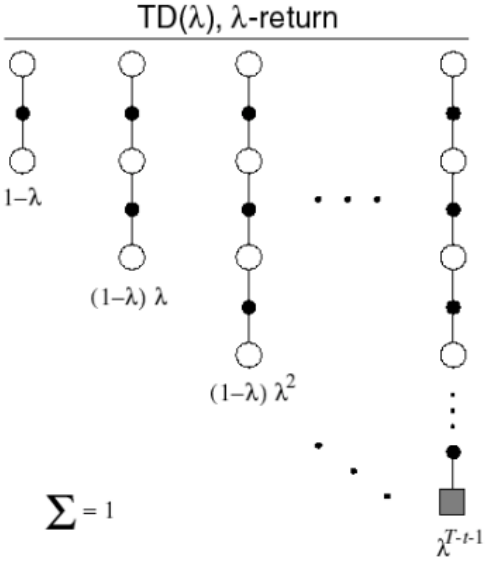
$$G_t^{(n)} = r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \cdots + \gamma^{n-1} r(s_{t+n-1}, a_{t+n-1}) + \gamma^n v_\pi(s_{t+n})$$

- n-step temporal-difference learning

$$v_\pi(s_t) \leftarrow v_\pi(s_t) + \alpha(G_t^{(n)} - v_\pi(s_t))$$

(Optional) TD(λ)

UC San Diego



Can we efficiently combine information from all time-steps?

- The λ-return G_t^λ combines all n-step returns $G_t^{(n)}$
- Using weight $(1-\lambda)\lambda^{n-1}$

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

- Forward-view TD(λ)

$$v_\pi(s_t) \leftarrow v_\pi(s_t) + \alpha(G_t^\lambda - v_\pi(s_t))$$

Summary

UC San Diego

Classic methods for discrete state space (s)

- For Monte Carlo (MC):

$$V(S_t) \leftarrow V(S_t) + \alpha (G_t - V(S_t))$$

- For Temporal-difference (TD(0))

$$V(S_t) \leftarrow V(S_t) + \alpha (r(S_t, a_t) + \gamma V(S_{t+1}) - V(S_t))$$

- For TD(λ)

$$V(S_t) \leftarrow V(S_t) + \alpha (G_t^{\lambda} - V(S_t)), \quad G_t^{\lambda} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

Current State S_t	State value $V(S)$
S_1	$V(S_1)$
S_2	$V(S_2)$
⋮	⋮
S_n	$V(S_n)$

$V(S)$ Lookup Table

Outline

UC San Diego

- State value function fitting $v_\pi(s)$
  - Classic methods: discrete state space
  - **Value function approximation: continuous state space**
- From value function to actor critic
- Actor-critic method variants

Value Function Approximation

UC San Diego

- So far we have represented value function by a lookup table
  - Every state s has an entry $v_\pi(s)$
- Problem with continuous state space (or large MDPs):
  - There are too many states to store in memory
  - It is too slow to learn the value of each state individually
- Solution for continuous state space (or large MDPs):
  - Estimate value function with function approximation

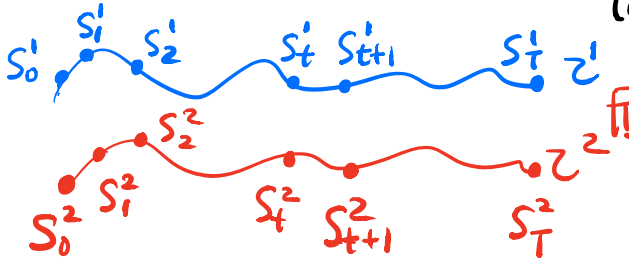
$$\hat{v}_\phi^\pi(s) \approx v_\pi(s)$$

move π to superscript; put function approximator parameter ϕ in the subscript

  - Generalize from seen states to unseen states
  - Update parameter ϕ using MC or TD learning

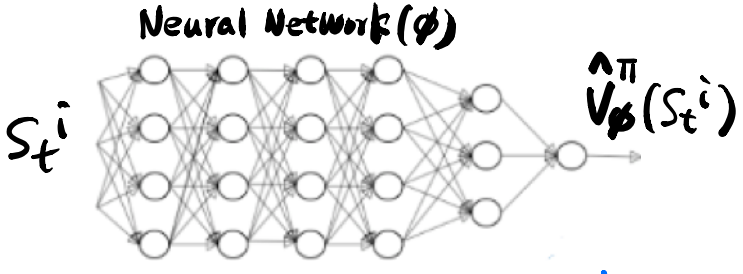
Value Function Approximation: Monte-Carlo

UC San Diego



(change episode index from 0 from last slide)

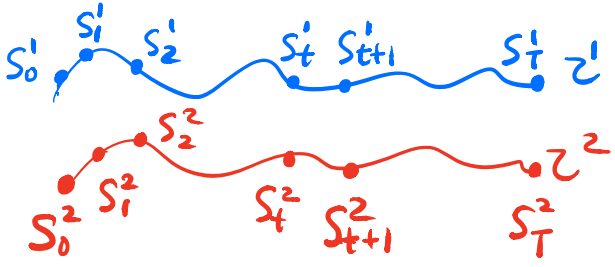
$T \times N$



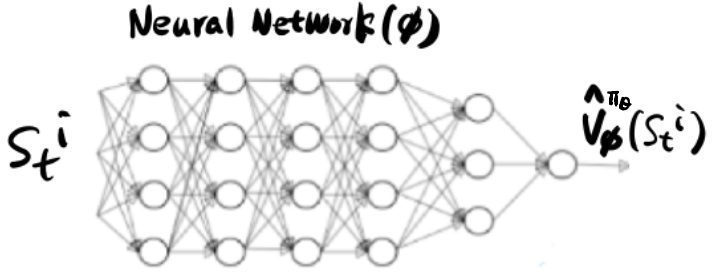
N episodes with length T

Training data: $\{(S_t^i, \sum_{t'=t}^{T-1} \gamma^{t'-t} r(S_{t'}^i, a_{t'}^i))\}_{t, G_t^i}$ (i: trajectory i's return at time t)

Supervised learning $L(\phi) = \frac{1}{2} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left(\hat{V}_\phi^\pi(S_t^i) - G_t^i \right)^2$



N episodes with length T



directly using previously fitted value function

Training data: $\{(s_t^i, \underbrace{r(s_t^i, a_t^i) + \gamma \hat{V}_\phi^\pi(s_{t+1}^i)}_{y_t^i})\}$

Supervised learning $L(\phi) = \frac{1}{2}\sum_{i=1}^{N}\sum_{t=0}^{T-1}\left(\hat{V}_\phi^\pi(s_t^i) - y_t^i\right)^2$

- Goal: find parameter ϕ minimizing mean-squared error between the approximate value $\hat{V}^{\pi}_{\phi}(S_t^i)$ and the "target value" y_t^i, either equal to G_t^i or $r(S_t^i, a_t^i) + \gamma \hat{V}^{\pi}_{\phi}(S_{t+1}^i)$

$$\min_{\phi} L(\phi) = \frac{1}{2} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left(\hat{V}^{\pi}_{\phi}(S_t^i) - y_t^i \right)^2$$

$$\nabla_{\phi} L(\phi) = \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left(\hat{V}^{\pi}_{\phi}(S_t^i) - y_t^i \right) \nabla_{\phi} \hat{V}^{\pi}_{\phi}(S_t^i)$$

Gradient descent:

$$\phi \leftarrow \phi - \alpha \nabla_\phi L(\phi)$$

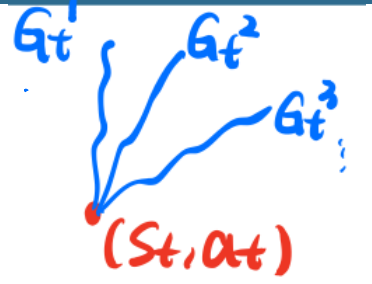
- MC: $\nabla_\phi L(\phi) = \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left(\hat{V}_\phi^\pi(s_t^i) - G_t^i \right) \nabla_\phi \hat{V}_\phi^\pi(s_t^i)$
- TD(0): $\nabla_\phi L(\phi) = \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left(\hat{V}_\phi^\pi(s_t^i) - (r(s_t^i, a_t^i) + \gamma \hat{V}_\phi^\pi(s_{t+1}^i)) \right) \nabla_\phi \hat{V}_\phi^\pi(s_t^i)$
- TD(λ): $\nabla_\phi L(\phi) = \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left(\hat{V}_\phi(s_t^i) - G_t^{i(\lambda)} \right) \nabla_\phi \hat{V}_\phi^\pi(s_t^i)$

Outline

UC San Diego

- State value function fitting $v_\pi(s)$
  - Classic methods: discrete state space
  - Value function approximation: continuous state space
- **From policy gradient to actor critic**
- Actor-critic method variants

Monte-Carlo policy gradient still has high variance



REINFORCE Algorithm

$$\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t\right]$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) G_t^i$$

"reward to go" if we take action a_t^i, at state s_t^i

$$Q^{\pi_\theta}(s_t, a_t) = E_{\pi_\theta}[G_t | s_t = s_t^i, A_t = a_t^i]$$: true expected reward-to-go

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) Q^{\pi_\theta}(s_t^i, a_t^i)$$

Reducing Variance Using a Critic

UC San Diego

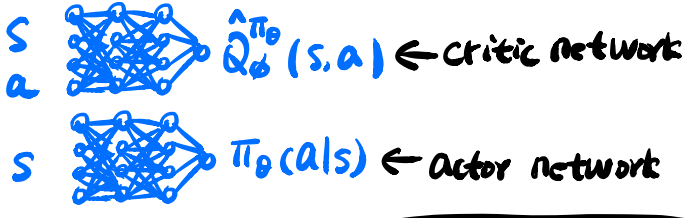
- Actor-critic algorithms maintain two sets of parameters.

Critic: value function parameter ϕ $\hat{Q}_\phi^{\pi_\theta}(s_t, a_t)$

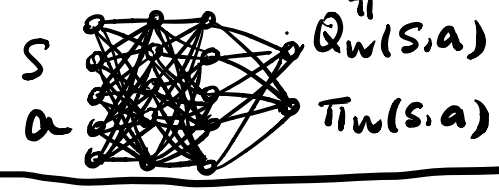
Actor: policy parameter θ $\pi_\theta(a_t|s_t)$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \hat{Q}_\phi^{\pi_\theta}(s_t^i, a_t^i)$$

Two networks design let's consider this!



Some methods use one network shared by actor & critic



How to fit state-action value function $\hat{Q}_{\phi}^{\pi_\theta}(s_t, a_t)$:

① Monte Carlo

② Temporal-difference learning TD(0)

③ TD(λ) ..

① Monte Carlo Method: $\{((s_t^i, a_t^i), \sum_{t'=t}^{T-1} \gamma^{t'-t} r(s_{t'}^i, a_{t'}^i))\}$, $\sum_{t'=t}^{T-1} \gamma^{t'-t} r(s_{t'}^i, a_{t'}^i) \rightarrow G_t^i$

Loss func: $L(\phi) = \frac{1}{2} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left(\hat{Q}_{\phi}^{\pi_\theta}(s_t^i, a_t^i) - G_t^i \right)^2$

② Temporal-difference learning (TD(0))

$$\left\{\left((s_t^i, a_t^i), r(s_t^i, a_t^i) + \gamma \hat{Q}_{\phi}^{\pi_\theta}(s_{t+1}^i, a_{t+1}^i)\right)\right\} \quad y_t^i$$

$$a_{t+1}^i \sim \pi_\theta(\cdot | s_{t+1}^i)$$

Loss func: $L(\phi) = \frac{1}{2}\sum_{i=1}^{N}\sum_{t=0}^{T-1}\left(\hat{Q}_{\phi}^{\pi_\theta}(s_t^i, a_t^i) - y_t^i\right)^2$

③ TD(λ): change target value, $y_t^i = G_t^{i(\lambda)}$

Actor-Critic Method with $Q^{\pi}(s,a)$

UC San Diego

- Using a neural network for function approximation $\hat{Q}_{\phi}^{\pi}(s,a)$
  - Critic Updates ϕ by temporal difference learning (*TD(0)*)
  - Actor Updates θ by policy gradient

Q − Actor Critic

Initialize policy θ; sample initial state $s \sim P_0$

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

for each step **do**

Obtain reward $r(s,a)$; sample transition $s' \sim \mathrm{P}_{ss'}^{a}$

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

$\Delta_{TD} = r(s,a) + \gamma \hat{Q}_{\phi}^{\pi_\theta}(s',a') - \hat{Q}_{\phi}^{\pi_\theta}(s,a)$ [TD(0)]

$\theta \leftarrow \theta + \alpha \nabla_\theta log\, \pi_\theta(s,a) \hat{Q}_{\phi}^{\pi_\theta}(s,a)$ ← actor update

$\phi \leftarrow \phi + \beta \Delta_{TD} \nabla_\phi \hat{Q}_{\phi}^{\pi_\theta}(s,a)$ ← critic update

$a \leftarrow a', s \leftarrow s'$

Note: This is an online version that only use current sample (s,a,s',r). We can also use batch Q actor-critic algorithm with state-action pairs collected from sampled episodes

Actor-Critic Method with $A^{\pi}(s,a)$

UC San Diego

- In standard REINFORCE algorithm, we can subtract a baseline $b(s)$ from the return G_t to reduce variance while remain unbiased → REINFORCE with Baseline
- Similarly, in actor-critic algorithms, we can subtract a baseline $b(s)$ from the state-action value function $\mathrm{Q}^{\pi}(\mathrm{s},\mathrm{a})$
- A good baseline is the state value function $b(s) = v^{\pi}(s)$
- Thus, we derive the *Actor-Critic Algorithm with Baseline* (or so called advantage function)

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\left[Q^{\pi_\theta}(s_t^i,a_t^i) - V^{\pi_\theta}(s_t^i)\right]$$

Substract baseline

Advantage function: $A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$

how much better a is, at state s.

Estimating the Advantage Function $A^{\pi}(s,a)$

UC San Diego

Two ways to estimate the advantage function $A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - v^{\pi_\theta}(s)$

- **Using two function approximators for estimating both $v^{\pi_\theta}(s)$ and $Q^{\pi_\theta}(s,a)$,**

$$\hat{v}_{\phi}^{\pi_\theta}(s) \approx v^{\pi_\theta}(s)$$

$$\hat{Q}_{\phi'}^{\pi_\theta}(s,a) \approx Q^{\pi_\theta}(s,a)$$

$$\hat{A}^{\pi_\theta}(s,a) = \hat{Q}_{\phi'}^{\pi_\theta}(s,a) - \hat{v}_{\phi}^{\pi_\theta}(s)$$

And updating both value functions by e.g., TD learning

- **Another way only learns function approximator for $v^{\pi_\theta}(s)$.** Define the TD error $\Delta_{TD} = r(s,a) + \gamma v^{\pi_\theta}(s') - v^{\pi_\theta}(s)$, $a \sim \pi_\theta(\cdot|s)$, since the TD error is an unbiased estimate of $A^{\pi_\theta}(s,a)$,

$$E_{\pi_\theta}[\Delta_{TD}|s,a] = E_{\pi_\theta}[r(s,a) + \gamma v^{\pi_\theta}(s')|s,a] - v^{\pi_\theta}(s) = Q^{\pi_\theta}(s,a) - v^{\pi_\theta}(s)$$

This approach only requires one set of critic parameters ϕ

$$\hat{A}^{\pi_\theta}(s,a) = r(s,a) + \gamma \hat{v}_{\phi}^{\pi_\theta}(s') - \hat{v}_{\phi}^{\pi_\theta}(s)$$

Actor-Critic Method with $A^{\pi}(s,a)$

UC San Diego

- Using a neural network for function approximation $\hat{v}_{\phi}^{\pi_\theta}(s)$
  - Critic Updates ϕ by temporal difference learning (*TD(0)*)
  - Actor Updates θ by policy gradient

Advantage −Actor Critic

Initialize policy θ; sample initial state $s \sim P_0$

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

for each step **do**

Obtain reward $r(s,a)$; sample transition $s' \sim P_{ss'}^{a}$

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

Evaluate $\hat{A}^{\pi_\theta}(s,a) = r(s,a) + \gamma \hat{v}_{\phi}^{\pi_\theta}(s') - \hat{v}_{\phi}^{\pi_\theta}(s)$ ← actor update

Update actor: $\theta \leftarrow \theta + \alpha \nabla_\theta log\, \pi_\theta(s,a) \hat{A}^{\pi_\theta}(s,a)$ ← critic update

Update critic: $\phi \leftarrow \phi + \beta \Delta_{TD} \nabla_\phi \hat{v}_{\phi}^{\pi_\theta}(s), \Delta_{TD} = r(s,a) + \gamma \hat{v}_{\phi}^{\pi_\theta}(s') - \hat{v}_{\phi}^{\pi_\theta}(s)$

$a \leftarrow a', s \leftarrow s'$

Note: This is an online version that only use current sample (s, a, s', r). We can also use batch Advantage Actor Critic

Summary of Policy Gradient Algorithms

$$\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t\right]$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) G_t^i \quad \leftarrow \text{REINFORCE}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\left(G_t^i - \underbrace{b(s_t^i)}_{V^{\pi_\theta}(s_t^i)}\right) \quad \leftarrow \text{REINFORCE with baseline}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \hat{Q}_\phi^{\pi_\theta}(s_t^i, a_t^i) \quad \leftarrow \text{Q Actor-critic}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \hat{A}_\phi^{\pi_\theta}(s_t^i, a_t^i) \quad \leftarrow \text{Advantage Actor-Critic}$$