CSCE 636 Neural Networks (Deep Learning)

Lecture 15: Deep Reinforcement Learning (continued)

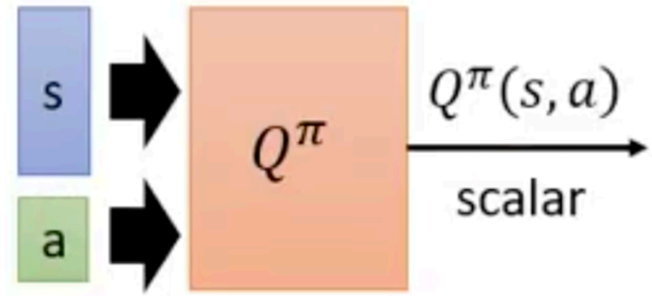
Anxiao (Andrew) Jiang

Based on the interesting lectures of Prof. Hung-yi Lee "Deep Reinforcement Learning"
https://www.youtube.com/watch?v=tnPVcec22cg&list=PLJV_el3uVTsODxQFgzMzPLa16h6B8kWM_&index=5
https://www.youtube.com/watch?v=j82QLgfhFiY&list=PLJV_el3uVTsODxQFgzMzPLa16h6B8kWM_&index=6

Outline

- Introduction of Q-Learning
- Tips of Q-Learning
- Q-Learning for Continuous Actions

Continuous Actions



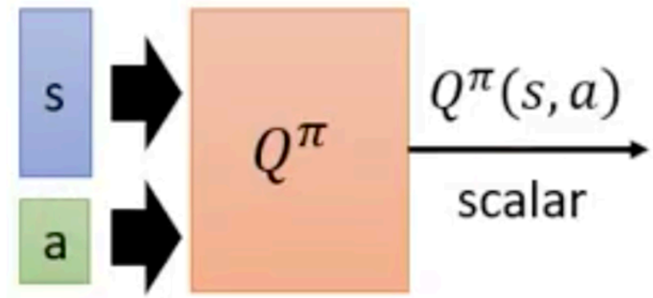
- Action a is a *continuous vector*

$$a = arg\max_a Q(s,a)$$

<u>Solution 1</u>

<u>Solution 2</u>

Continuous Actions



- Action a is a *continuous vector*

$$a = arg \max_a Q(s,a)$$

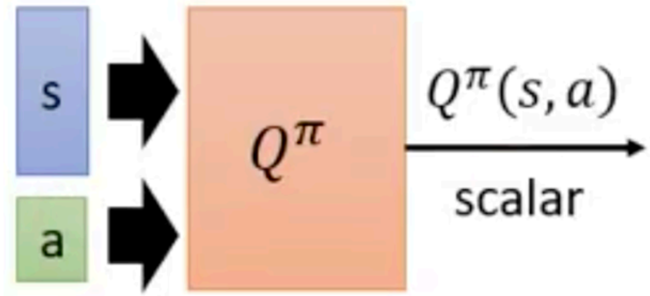
Solution 1

Sample a set of actions: $\{a_1, a_2, \cdots, a_N\}$

See which action can obtain the largest Q value

Solution 2

Continuous Actions



- Action a is a *continuous vector*

$$a = arg\max_{a} Q(s,a)$$

Solution 1

Sample a set of actions: $\{a_1, a_2, \cdots, a_N\}$

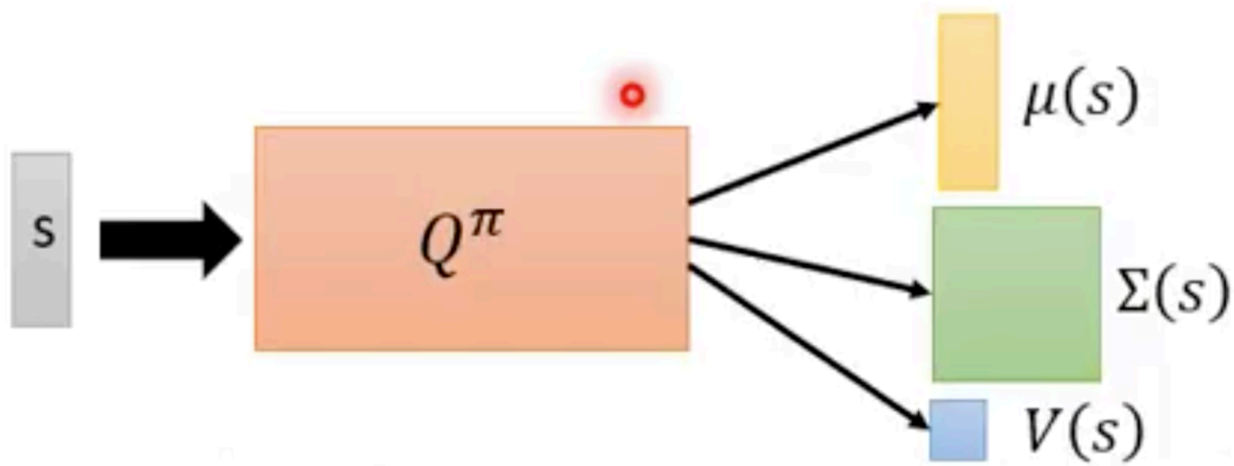
See which action can obtain the largest Q value

Solution 2

Using gradient ascent to solve the optimization problem.

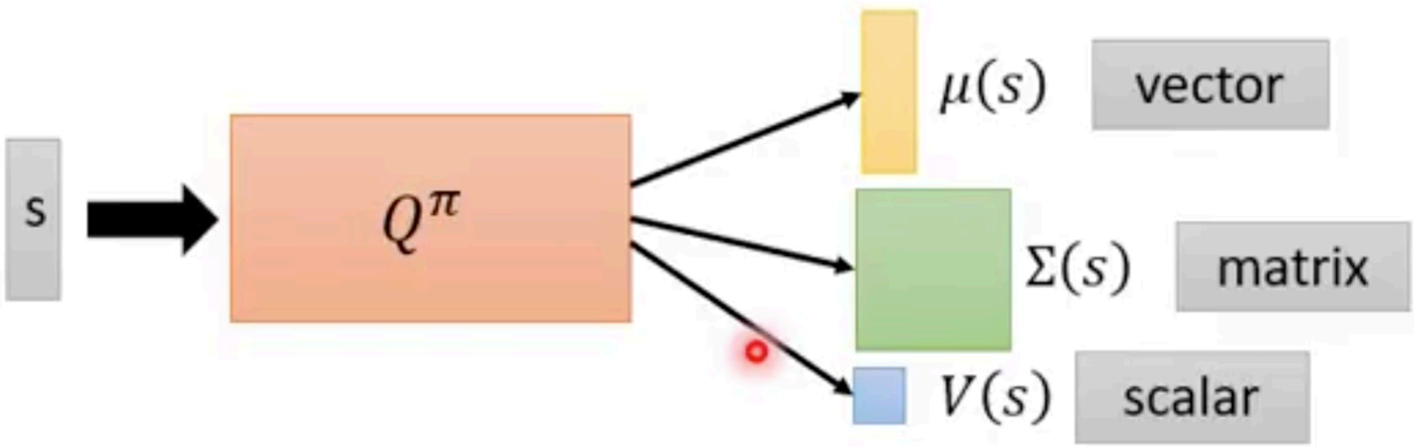
Continuous Actions

Solution 3 Design a network to make the optimization easy.



Continuous Actions

Solution 3 Design a network to make the optimization easy.

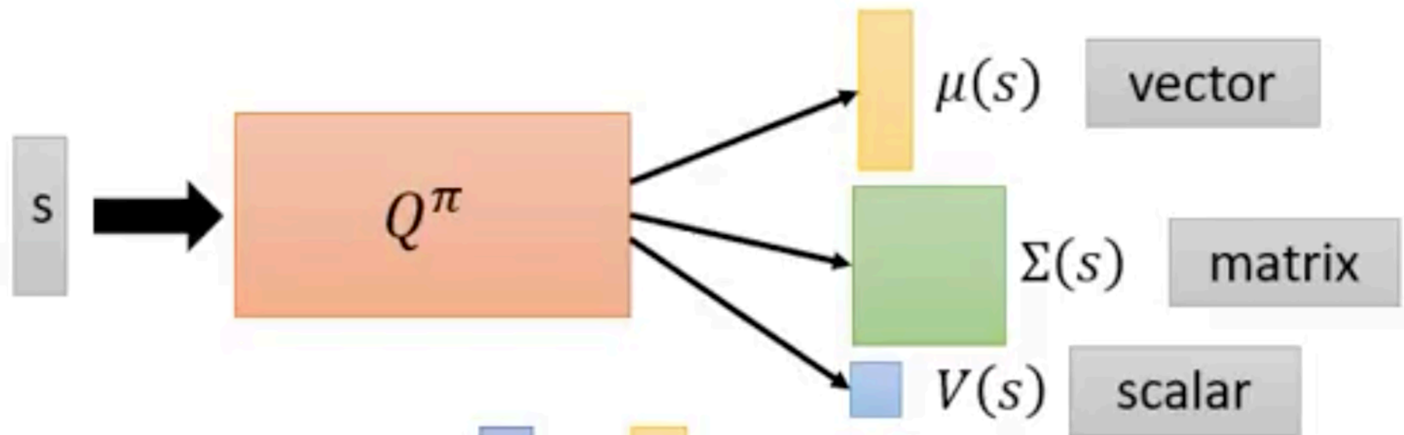


$$Q(s,a) = -\left(a-\mu(s)\right)^T \Sigma(s) \left(a-\mu(s)\right) + V(s)$$

Positive semi-definite matrix

Continuous Actions

Solution 3 Design a network to make the optimization easy.

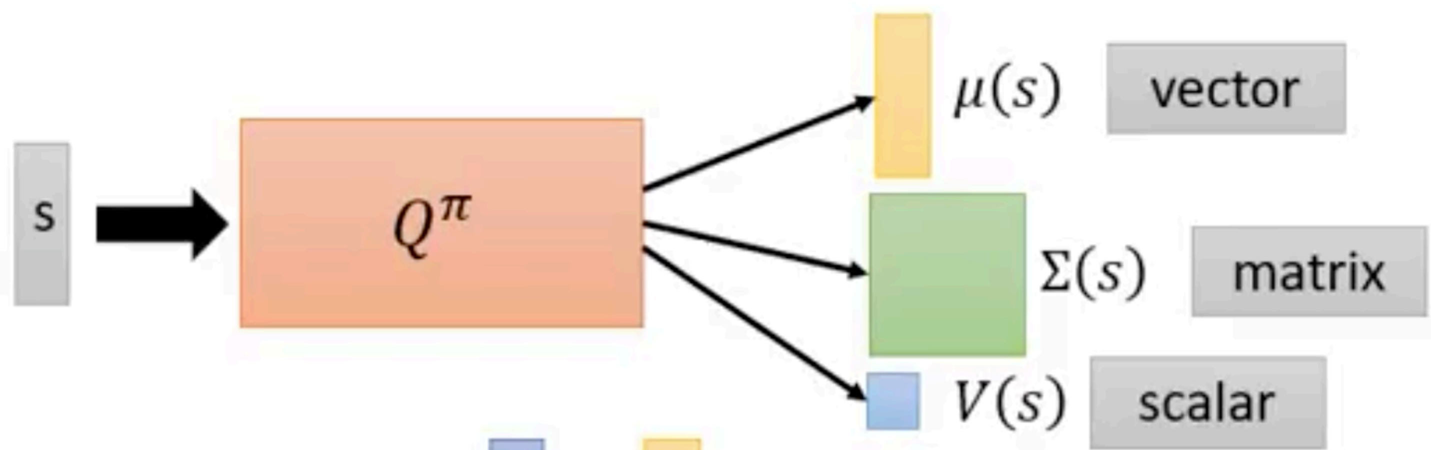


$$Q(s,a) = -\left(a - \mu(s)\right)^T \Sigma(s) \left(a - \mu(s)\right) + V(s)$$

$$= arg \max_a Q(s,a)$$

Continuous Actions

Solution 3 Design a network to make the optimization easy.



$$Q(s,a) = -(a-\mu(s))^T \Sigma(s)(a-\mu(s)) + V(s)$$

$$\mu(s) = arg\max_a Q(s,a)$$

Actor-Critic

Asynchronous Advantage Actor-Critic (A3C)

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, Koray Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning", ICML, 2016

Review – Policy Gradient

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \left(\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b \right) \nabla log p_\theta(a_t^n | s_t^n)$$

Review – Policy Gradient

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\Big(\underbrace{\sum\nolimits_{t'=t}^{T_n}\gamma^{t'-t} r_{t'}^{n}}_{G_t^n\text{ : obtained via interaction}} - \overset{\text{baseline}}{\underline{b}}\Big)\nabla log p_\theta(a_t^n|s_t^n)$$

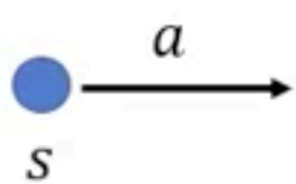
Review – Policy Gradient

baseline

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\underline{\sum_{t'=t}^{T_n}\gamma^{t'-t} r_{t'}^n} - \underline{b}\right)\nabla \log p_\theta(a_t^n|s_t^n)$$

G_t^n : obtained via interaction

Very unstable



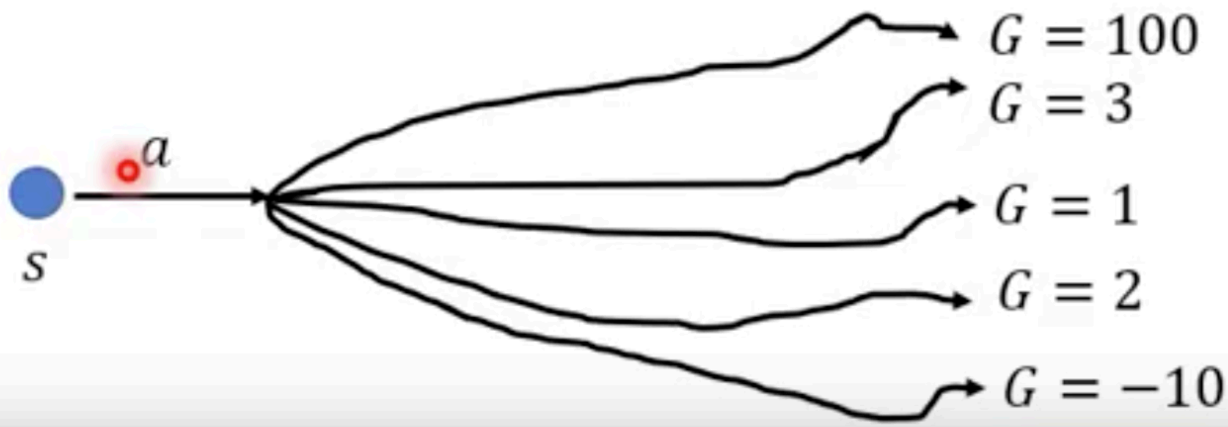
Review – Policy Gradient

baseline

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\underline{\sum_{t'=t}^{T_n}\gamma^{t'-t}r_{t'}^{n}} - \underline{b}\right)\nabla log p_\theta(a_t^n|s_t^n)$$

G_t^n : obtained via interaction

Very unstable



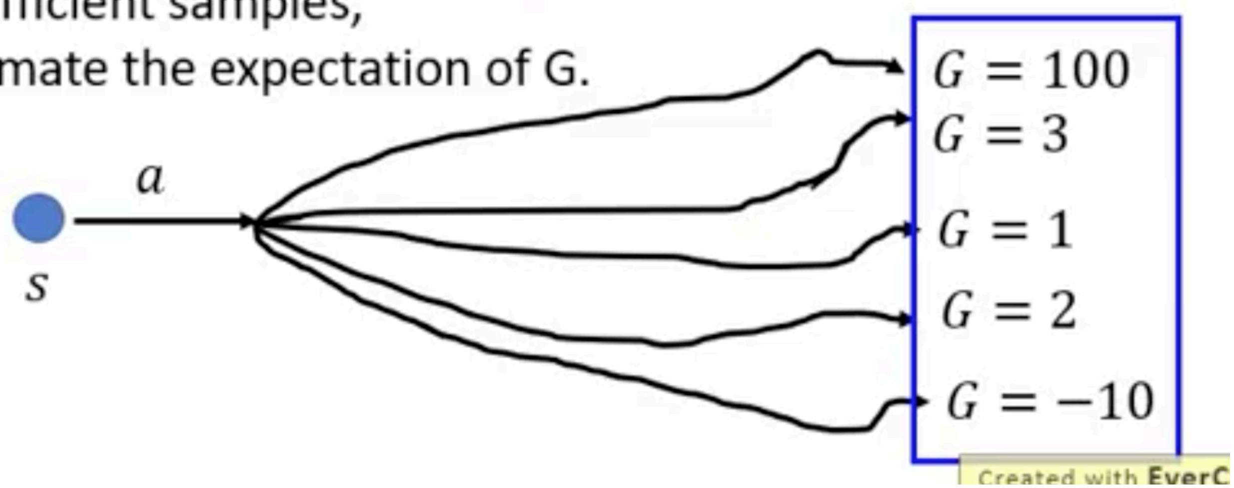
Review – Policy Gradient

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\Big(\underbrace{\sum_{t'=t}^{T_n}\gamma^{t'-t}r_{t'}^{n}}_{G_t^n} - \underbrace{b}_{\text{baseline}}\Big)\nabla log p_\theta(a_t^n|s_t^n)$$

G_t^n: obtained via interaction

Very unstable

With sufficient samples,
approximate the expectation of G.



Review – Policy Gradient

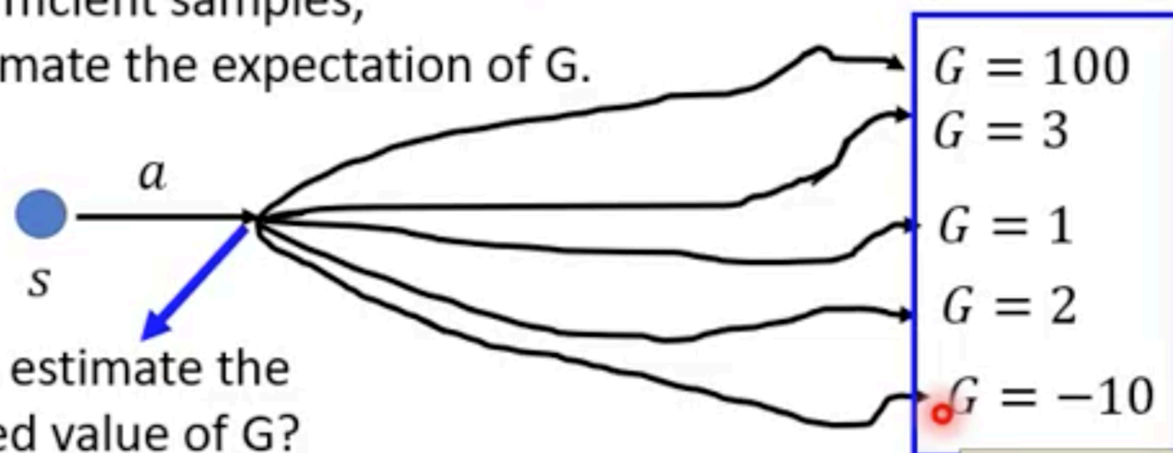
baseline

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\sum_{t'=t}^{T_n}\gamma^{t'-t}r_{t'}^{n} - b\right)\nabla log p_\theta(a_t^n|s_t^n)$$

G_t^n : obtained via interaction

Very unstable

With sufficient samples, approximate the expectation of G.



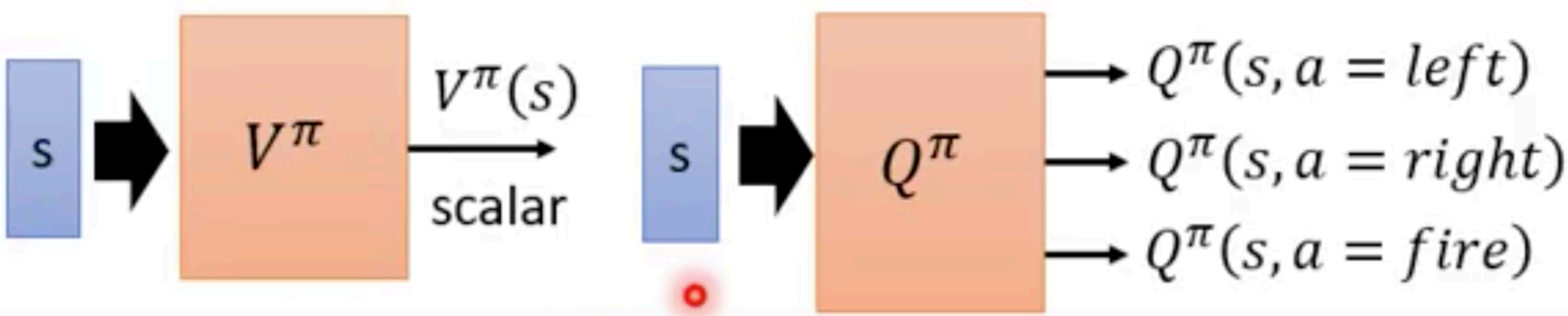
Can we estimate the expected value of G?

Review – Q-Learning

- State value function $V^{\pi}(s)$
  - When using actor π, the *cumulated* reward expects to be obtained after visiting state s
- State-action value function $Q^{\pi}(s, a)$
  - When using actor π, the *cumulated* reward expects to be obtained after taking a at state s

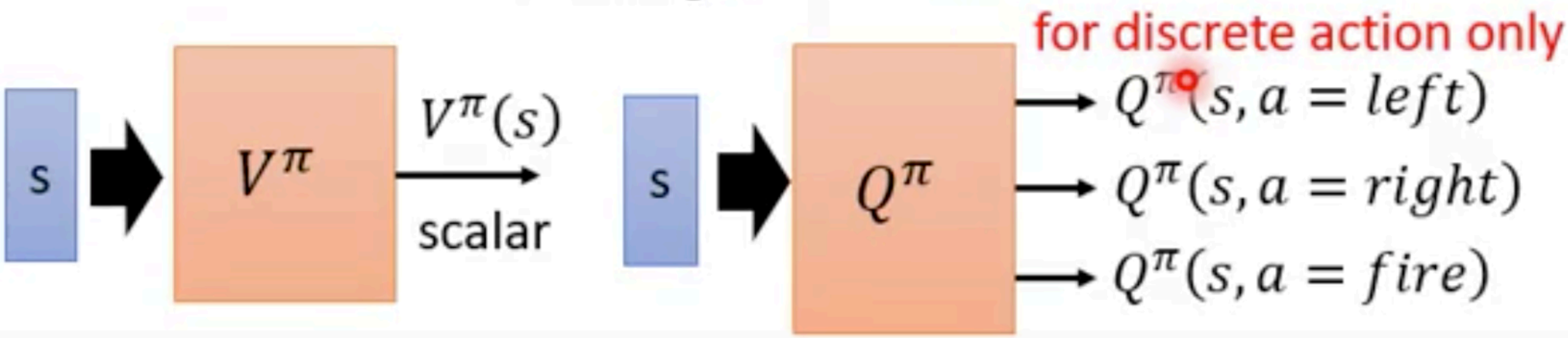
Review – Q-Learning

- State value function $V^{\pi}(s)$
  - When using actor π, the *cumulated* reward expects to be obtained after visiting state s
- State-action value function $Q^{\pi}(s, a)$
  - When using actor π, the *cumulated* reward expects to be obtained after taking a at state s



Review – Q-Learning

- State value function $V^\pi(s)$
  - When using actor π, the *cumulated* reward expects to be obtained after visiting state s
- State-action value function $Q^\pi(s,a)$
  - When using actor π, the *cumulated* reward expects to be obtained after taking a at state s



Estimated by TD or MC

Actor-Critic

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\Big(\underbrace{\sum_{t'=t}^{T_n}\gamma^{t'-t} r_{t'}^n}_{G_t^n \text{ : obtained via interaction}} - \overset{\text{baseline}}{\underline{b}}\Big)\nabla \log p_\theta(a_t^n|s_t^n)$$

$$E[G_t^n] =$$

Actor-Critic

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\underbrace{\sum\nolimits_{t'=t}^{T_n}\gamma^{t'-t} r_{t'}^n}_{G_t^n\text{ : obtained via interaction}} - \overset{\text{baseline}}{\underline{b}}\right)\nabla logp_\theta(a_t^n|s_t^n)$$

$$E[G_t^n] = Q^{\pi_\theta}(s_t^n, a_t^n)$$

Actor-Critic

$V^{\pi_\theta}(s_t^n)$

baseline

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\underbrace{\sum_{t'=t}^{T_n}\gamma^{t'-t}r_{t'}^n}_{G_t^n} - b\right)\nabla log p_\theta(a_t^n|s_t^n)$$

G_t^n : obtained via interaction

$$E[G_t^n] = Q^{\pi_\theta}(s_t^n, a_t^n)$$

Actor-Critic

$$Q^{\pi_\theta}(s_t^n, a_t^n) - V^{\pi_\theta}(s_t^n)$$

$$V^{\pi_\theta}(s_t^n)$$

baseline

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \boxed{\left(\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b \right)} \nabla log p_\theta(a_t^n | s_t^n)$$

G_t^n : obtained via interaction

$$E[G_t^n] = Q^{\pi_\theta}(s_t^n, a_t^n)$$

Advantage Actor-Critic

$$\boxed{Q^{\pi}(s_t^n, a_t^n) - V^{\pi}(s_t^n)}$$

Estimate two networks? We can only estimate one.

Advantage Actor-Critic

$$\boxed{Q^\pi(s_t^n, a_t^n) - V^\pi(s_t^n)}$$

Estimate two networks? We can only estimate one.

$$Q^\pi(s_t^n, a_t^n) = E[r_t^n + V^\pi(s_{t+1}^n)]$$

Advantage Actor-Critic

$\boxed{Q^{\pi}(s_t^n, a_t^n) - V^{\pi}(s_t^n)}$

Estimate two networks? We can only estimate one.

$$Q^{\pi}(s_t^n, a_t^n) = E[r_t^n + V^{\pi}(s_{t+1}^n)]$$

$$Q^{\pi}(s_t^n, a_t^n) = r_t^n + V^{\pi}(s_{t+1}^n)$$

Advantage Actor-Critic

$$Q^{\pi}(s_t^n, a_t^n) - V^{\pi}(s_t^n)$$

$$\Downarrow$$

$$r_t^n + V^{\pi}(s_{t+1}^n) - V^{\pi}(s_t^n)$$

Estimate two networks? We can only estimate one.

$$Q^{\pi}(s_t^n, a_t^n) = E[r_t^n + V^{\pi}(s_{t+1}^n)]$$

$$Q^{\pi}(s_t^n, a_t^n) = r_t^n + V^{\pi}(s_{t+1}^n)$$

Advantage Actor-Critic

$$Q^{\pi}(s_t^n, a_t^n) - V^{\pi}(s_t^n)$$

Estimate two networks? We can only estimate one.

$$\Downarrow$$

$$r_t^n + V^{\pi}(s_{t+1}^n) - V^{\pi}(s_t^n)$$

Only estimate state value

$$Q^{\pi}(s_t^n, a_t^n) = E[r_t^n + V^{\pi}(s_{t+1}^n)]$$

$$Q^{\pi}(s_t^n, a_t^n) = r_t^n + V^{\pi}(s_{t+1}^n)$$

Advantage Actor-Critic

$$Q^{\pi}(s_t^n, a_t^n) - V^{\pi}(s_t^n)$$

Estimate two networks? We can only estimate one.

$$r_t^n + V^{\pi}(s_{t+1}^n) - V^{\pi}(s_t^n)$$

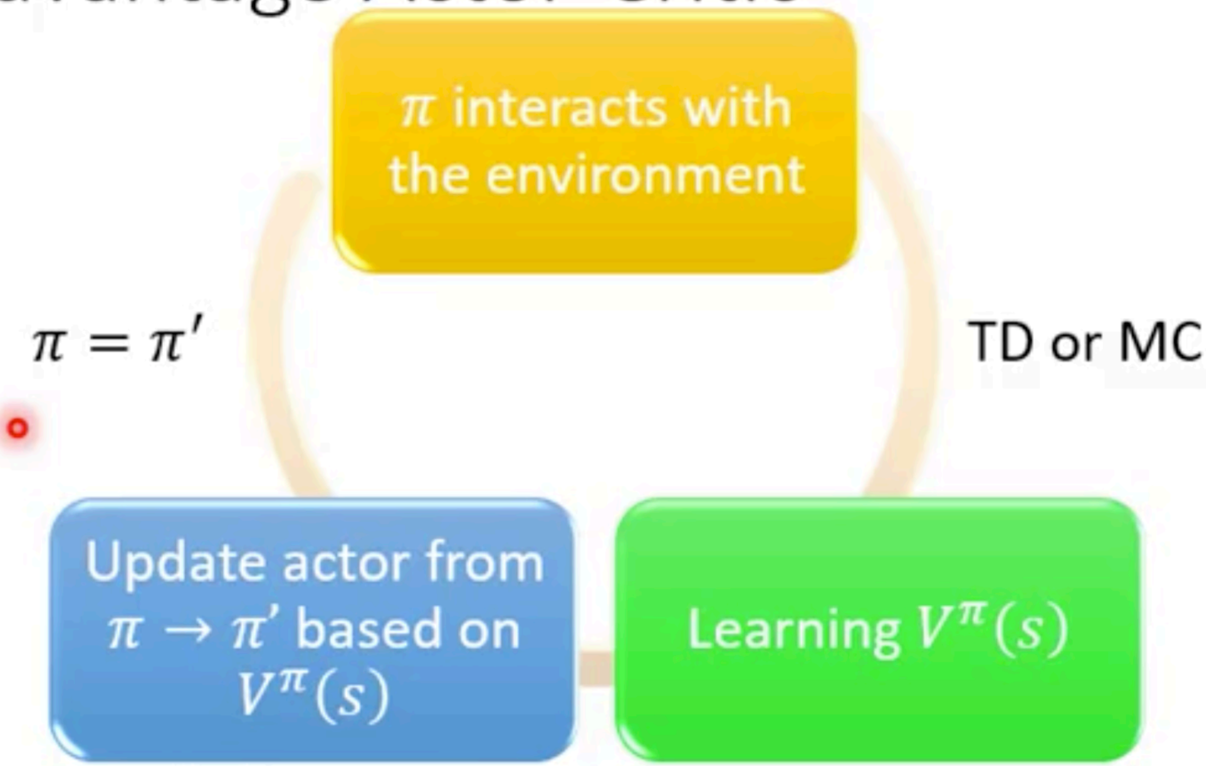
Only estimate state value

A little bit variance

$$Q^{\pi}(s_t^n, a_t^n) = E[r_t^n + V^{\pi}(s_{t+1}^n)]$$

$$Q^{\pi}(s_t^n, a_t^n) = r_t^n + V^{\pi}(s_{t+1}^n)$$

Advantage Actor-Critic



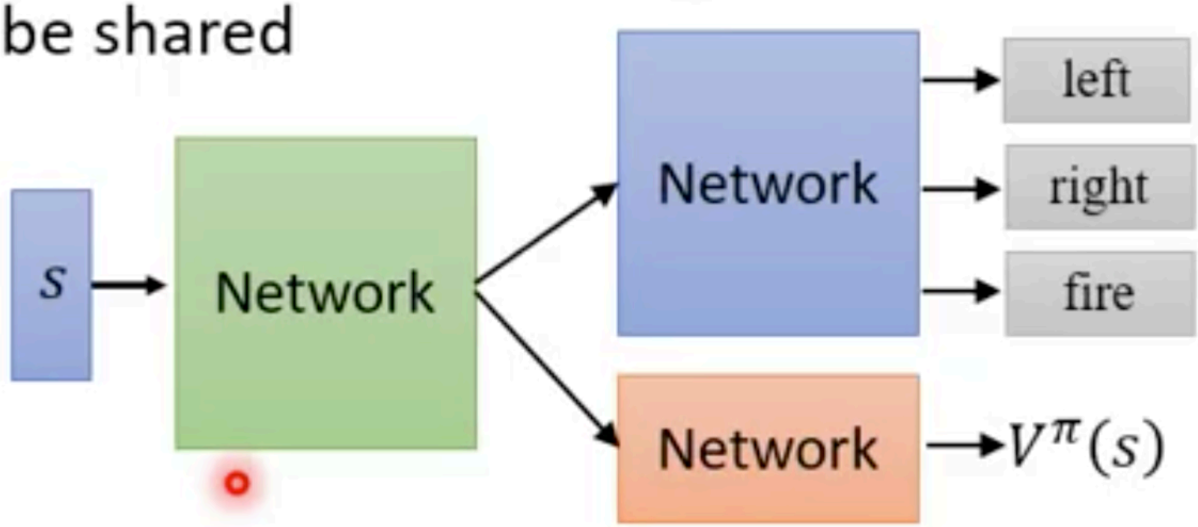
$$\nabla \bar{R}_{\theta} \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \left(r_t^n + V^{\pi}(s_{t+1}^n) - V^{\pi}(s_t^n)\right) \nabla log p_{\theta}(a_t^n | s_t^n)$$

Advantage Actor-Critic

- Tips
  - The parameters of actor $\pi(s)$ and critic $V^{\pi}(s)$ can be shared

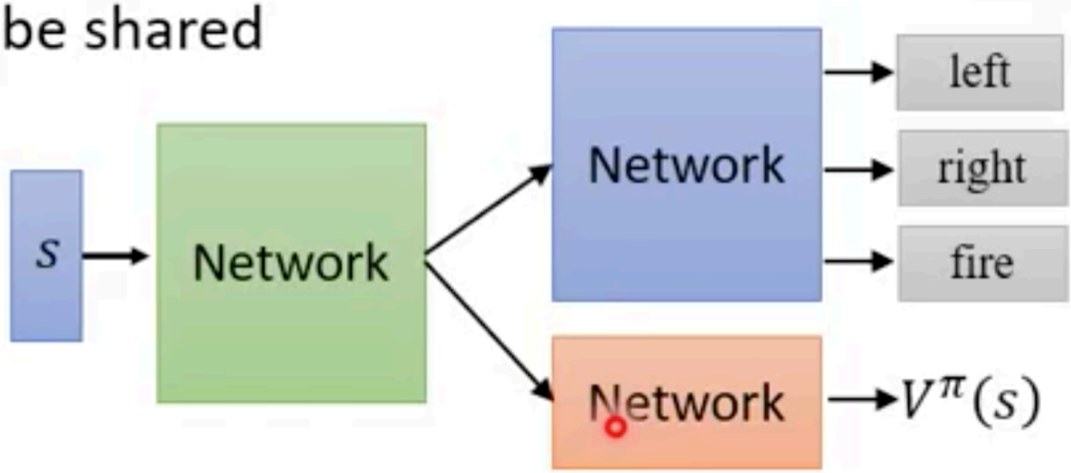
Advantage Actor-Critic

- Tips
  - The parameters of actor $\pi(s)$ and critic $V^{\pi}(s)$ can be shared



Advantage Actor-Critic

- Tips
  - The parameters of actor $\pi(s)$ and critic $V^{\pi}(s)$ can be shared

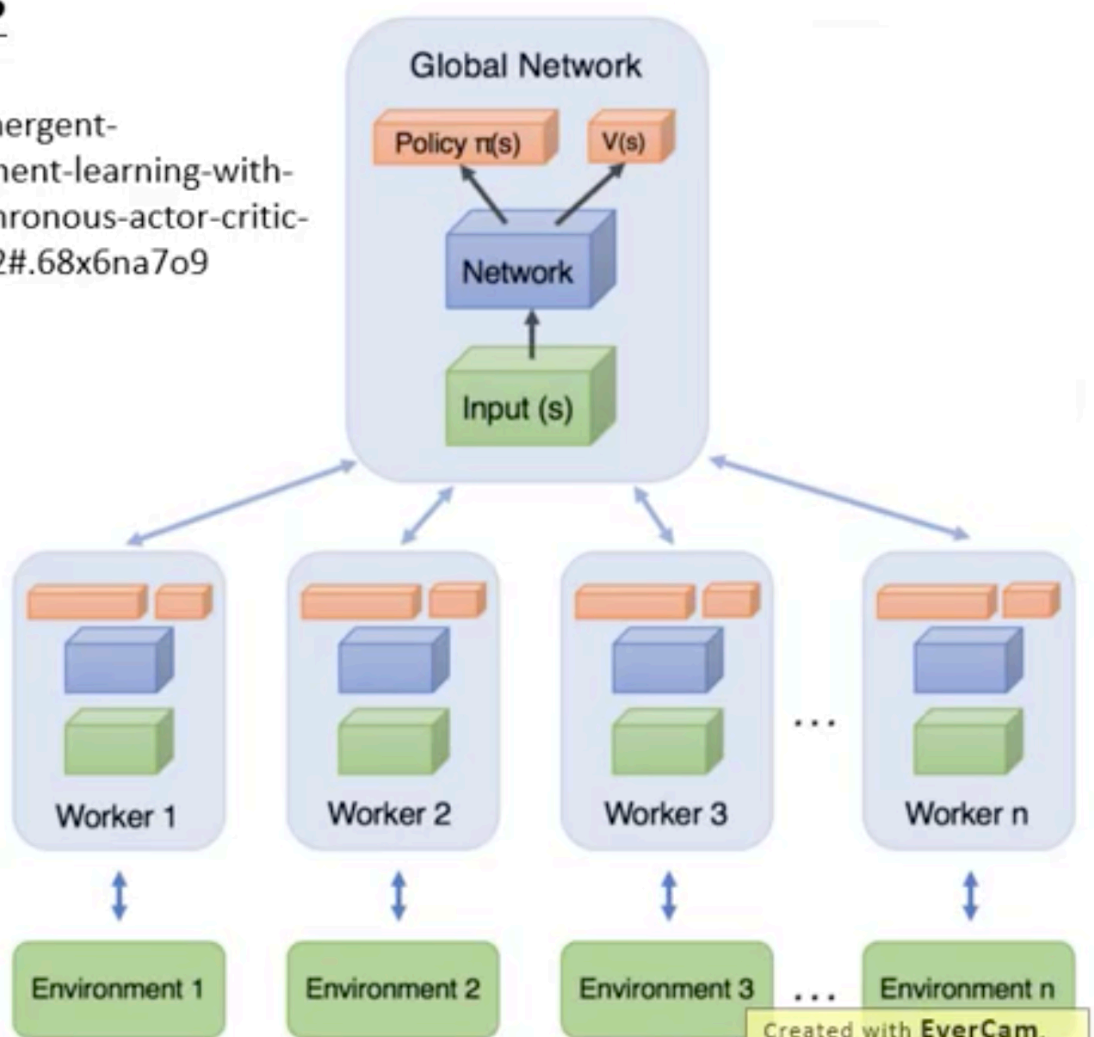


  - Use output entropy as regularization for $\pi(s)$
    - Larger entropy is preferred → exploration

Asynchronous Advantage Actor-Critic (A3C)

Asynchronous

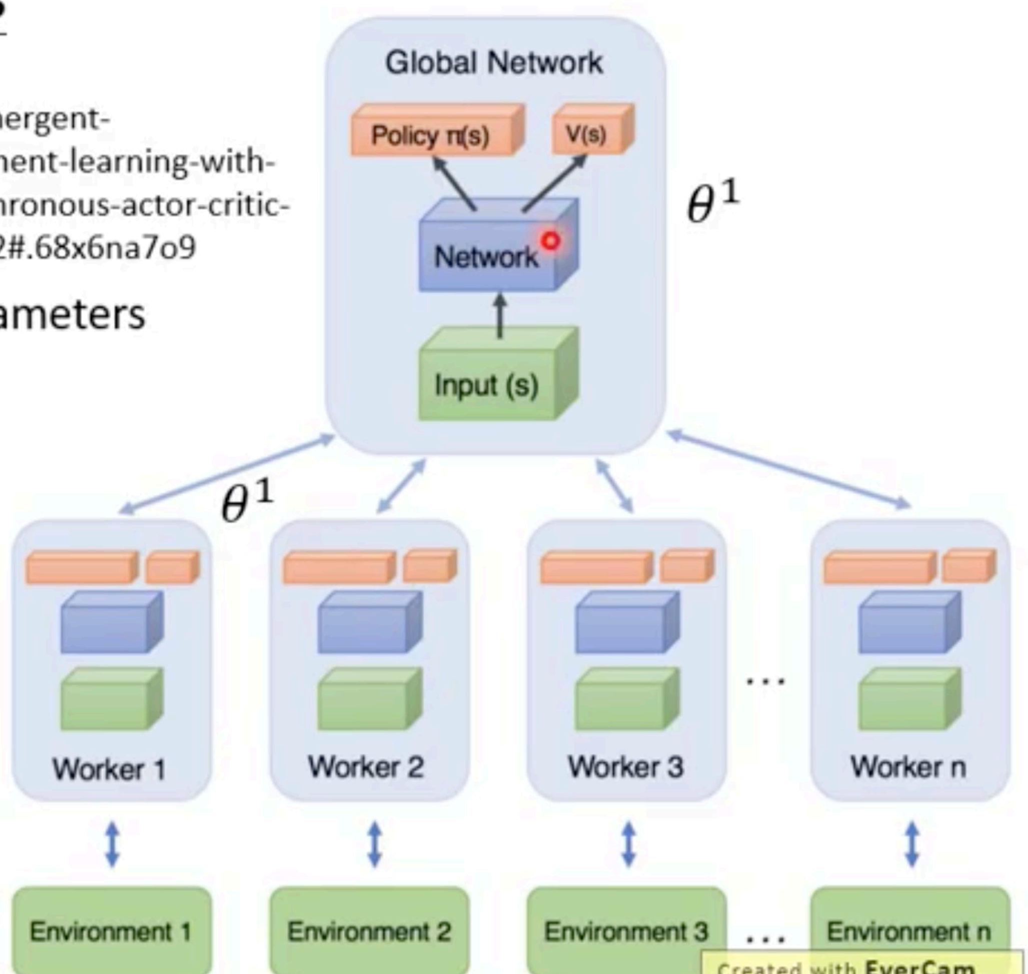
Source of image: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2#.68x6na7o9



Asynchronous

Source of image: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2#.68x6na7o9

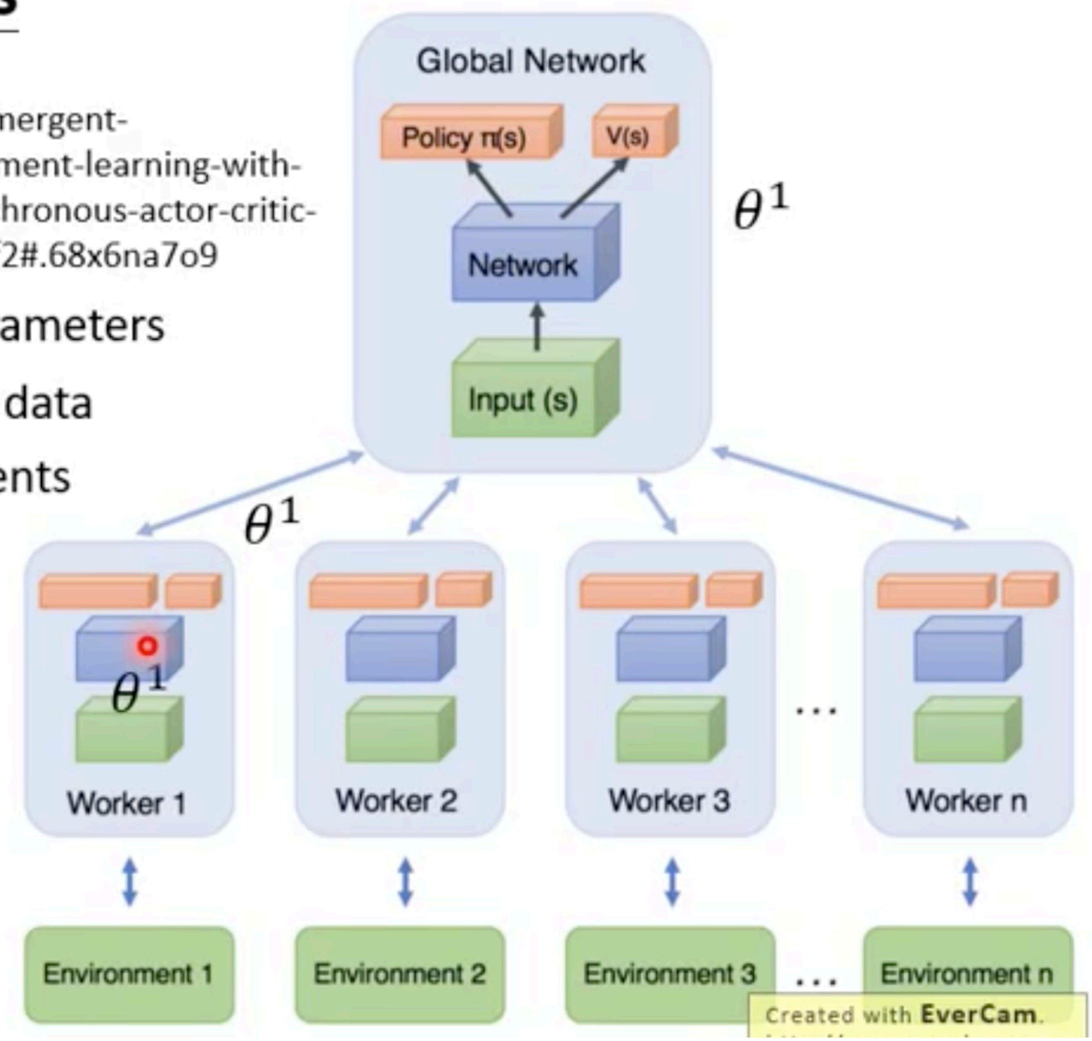
1. Copy global parameters



Asynchronous

Source of image: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2#.68x6na7o9

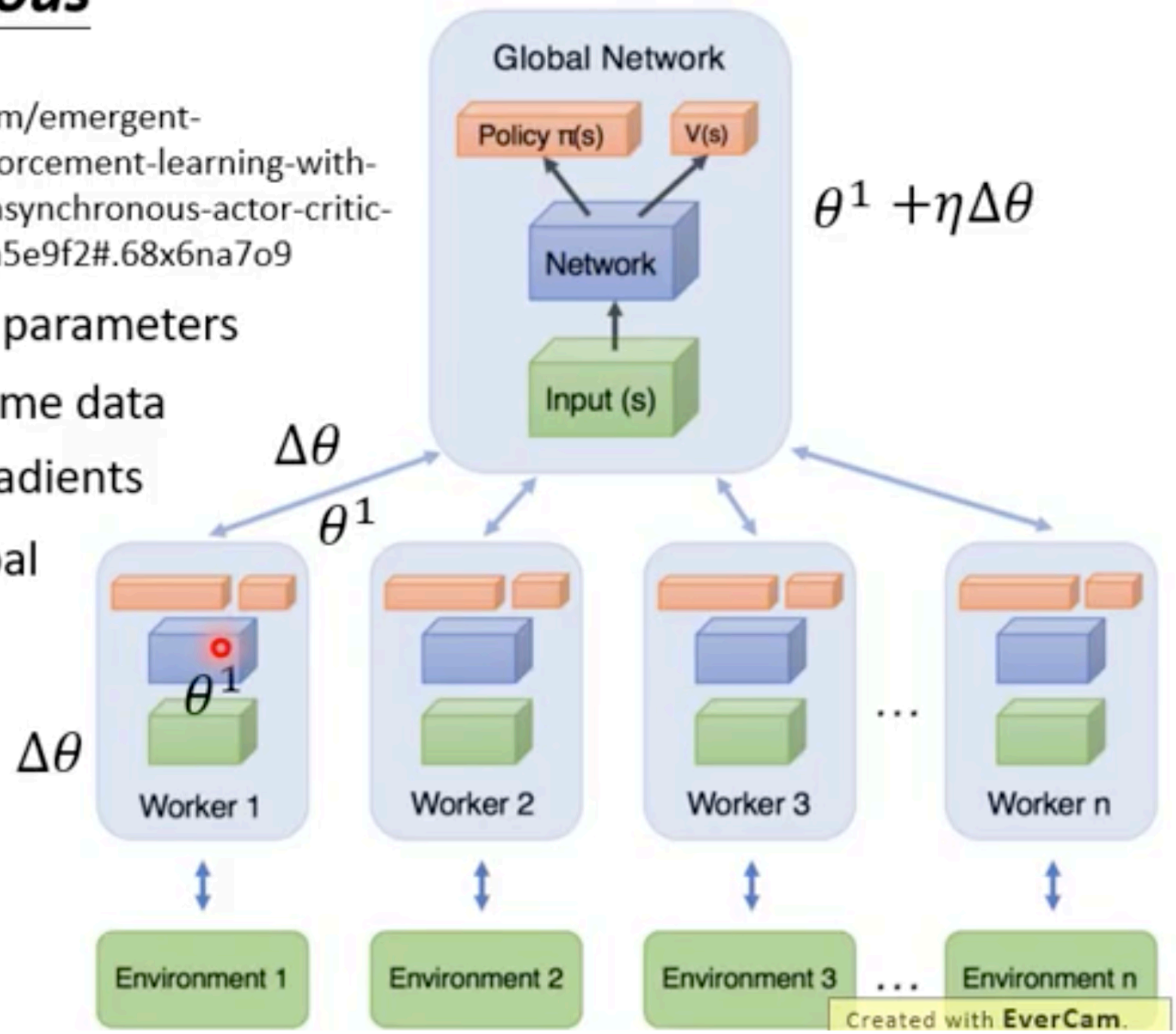
1. Copy global parameters
2. Sampling some data
3. Compute gradients



Asynchronous

Source of image: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2#.68x6na7o9

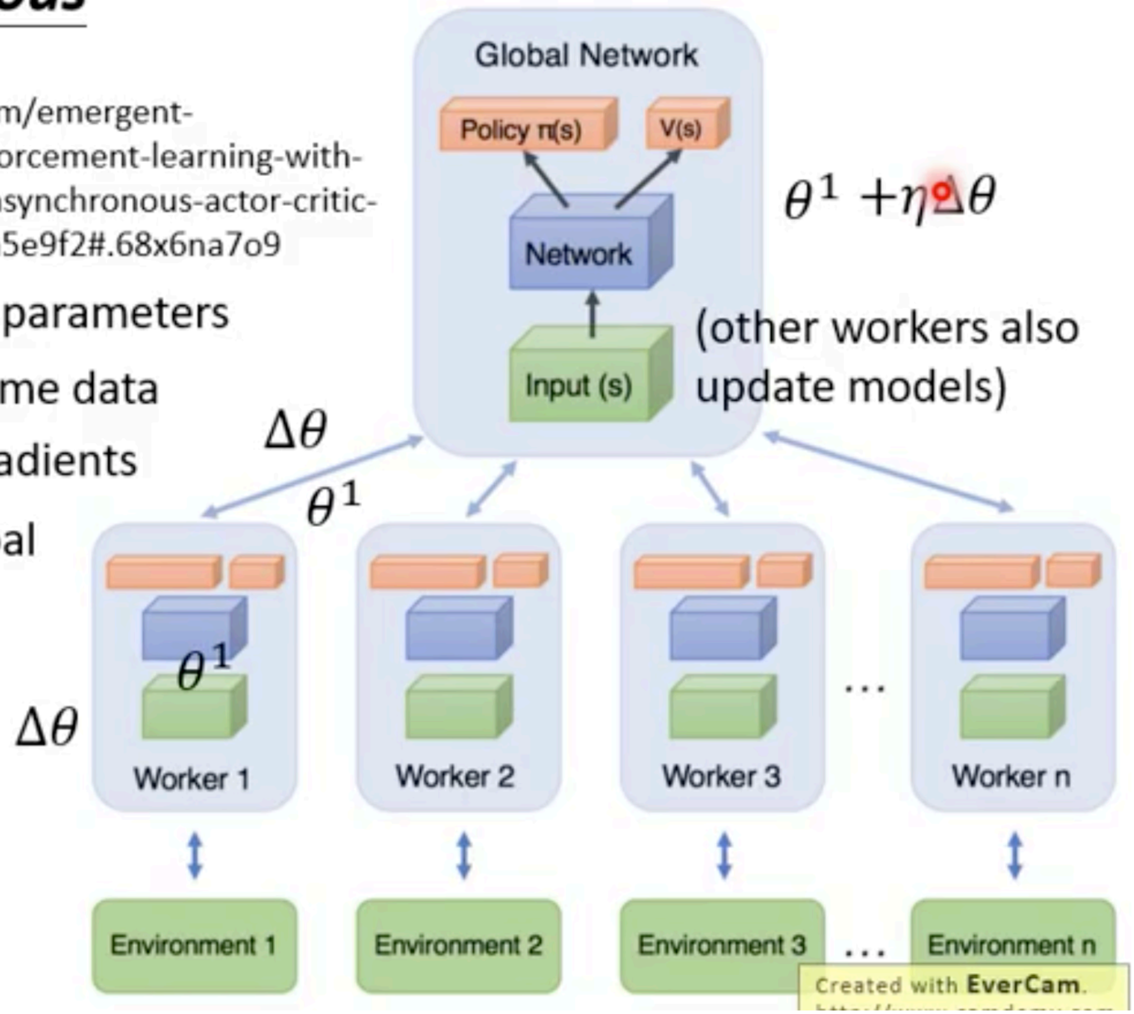
1. Copy global parameters
2. Sampling some data
3. Compute gradients
4. Update global models



Asynchronous

Source of image:
https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2#.68x6na7o9

1. Copy global parameters
2. Sampling some data
3. Compute gradients
4. Update global models



Asynchronous

Source of image: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2#.68x6na7o9

1. Copy global parameters
2. Sampling some data
3. Compute gradients
4. Update global models