Probability Concepts and ACTOR CRITIC ALGORITHM

DR SRIPARNA SAHA,
CSE DEPT, IIT PATNA

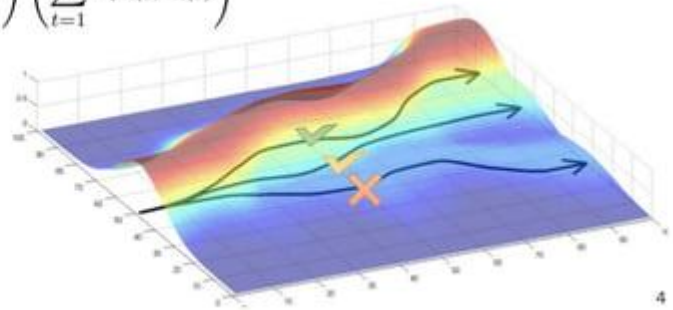
Outline

- Recap policy gradient
- Actor-Critic algorithm
- Recommended Papers

Recap: policy gradient

REINFORCE algorithm:

1. sample trajectory τ^i from $\pi_\theta(a_t|s_t)$
2. $$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)$$
3. $$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$$



Recap: policy gradient

REINFORCE algorithm:

1. sample trajectory τ^i from $\pi_\theta(a_t|s_t)$
2. $$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)$$
3. $$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$$

Issue: We need to sample whole trajectory to get this term (Monte Carlo)

→ Make policy gradient learn slowly.

Recap: policy gradient

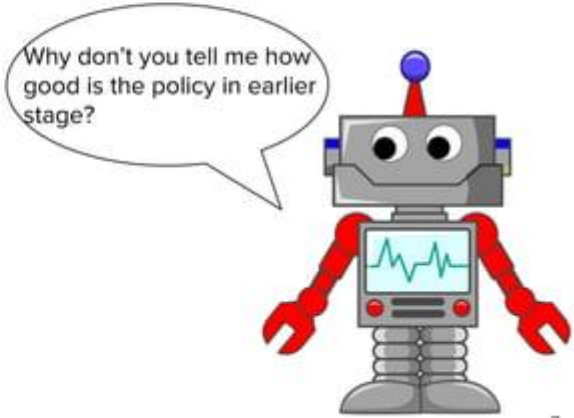
REINFORCE algorithm:

1. sample trajectory τ^i from $\pi_\theta(a_t|s_t)$
2. $$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)$$
3. $$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$$

Can we learn step by step?

Actor-Critic algorithm

In vanilla policy gradient, we only can evaluate our policy when we finish the whole episode.



Actor-Critic algorithm

In vanilla policy gradient, we only can evaluate our policy when we finish the whole episode.

Hello, critic is here. I'll give you score in every step.

Why don't you tell me how good is the policy in earlier stage?

Actor-Critic algorithm

Objective of vanilla policy gradient:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'})\right]$$

$$\widehat{Q}(s_{i,t}, a_{i,t})$$

The return in policy gradient with causality in step ***t*** could be replaced by expected action-value function.

If we could find the action-value function in each step, we can improve learning efficiency by TD learning.

Actor-Critic algorithm

Policy gradient with causality:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \widehat{Q}(s_{i,t}, a_{i,t}) \right]$$

Question: How do we get action-value function Q?

Actor-Critic algorithm

Policy gradient with causality:

Policy Network

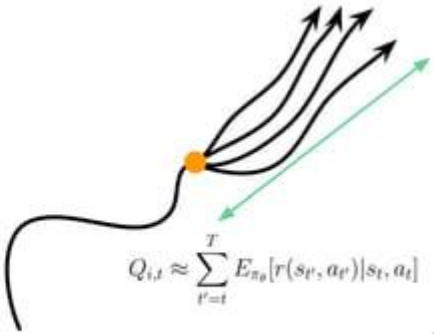
Critic Network

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\widehat{Q}(s_{i,t},a_{i,t})\right]$$

Question: How do we get action-value function Q?

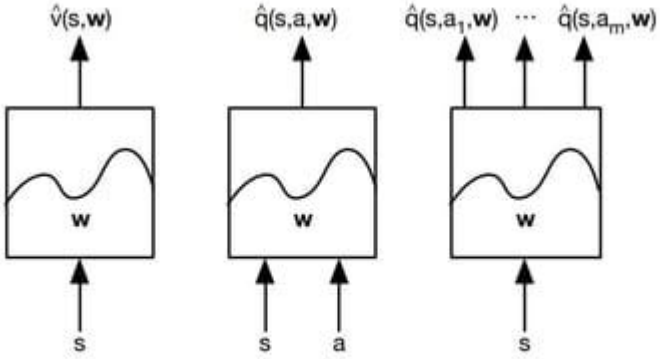
Using another neural network to approximate value function called Critic. This is so-called Actor-Critic.

By using Critic network, we can update the neural network step by step. However, it will also introduce bias.



Actor-Critic algorithm

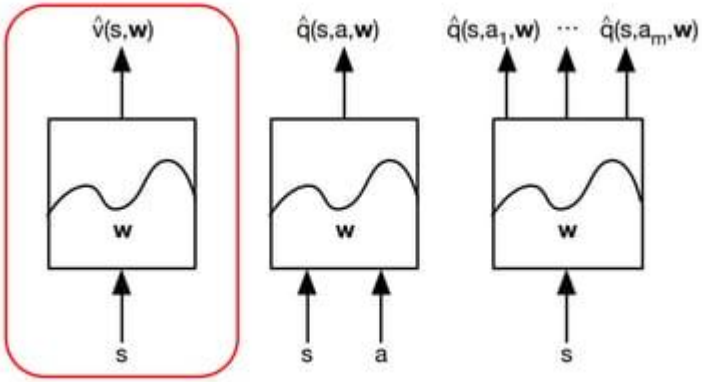
Which kind of format of neural network do we choose in Critic?



Actor-Critic algorithm

Which kind of format of neural network do we choose in Critic?

We usually fit the value function. I'll show you the reason soon. (Other choices are fine)



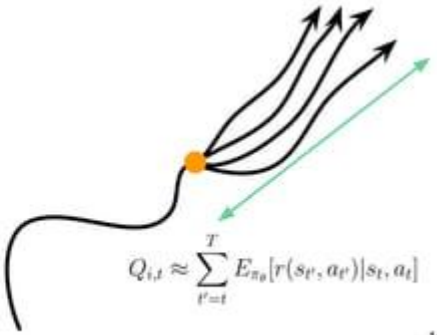
Actor-Critic algorithm

The objective of Actor-Critic

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \widehat{Q}(s_{i,t}, a_{i,t}) \right]$$

This objective function in this version have lower variance and higher bias than REINFORCE when we learning by TD learning.

Can we also subtract a baseline to reduce the variance?



Actor-Critic algorithm

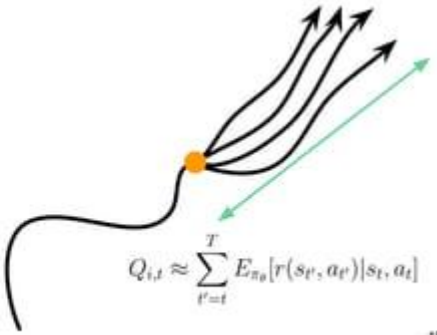
The objective of Actor-Critic:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \, \widehat{Q}(s_{i,t}, a_{i,t}) \right]$$

This objective function in this version have lower variance and higher bias than REINFORCE when we learning by TD learning.

Can we also subtract a baseline to reduce the variance? Yes! we could subtract this term:

$$V(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)} \left[Q(s_t, a_t) \right]$$



Actor-Critic algorithm

The objective of Actor-Critic with value function baseline:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\big(Q(s_{i,t},a_{i,t}) - V(s_{i,t})\big)$$

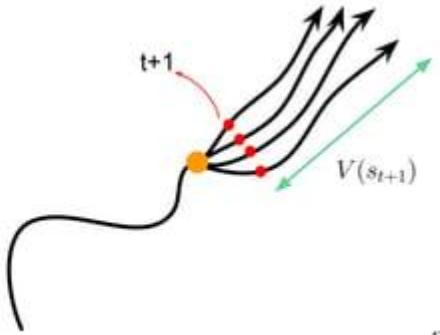
$Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'})|s_t, a_t]$: how good the action we take from current state.

$V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)]$: The average return when other agent face the same state.

$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$: We called this **advantage function**, which reflects how good the action we've taken compared to other candidates.

Actor-Critic algorithm

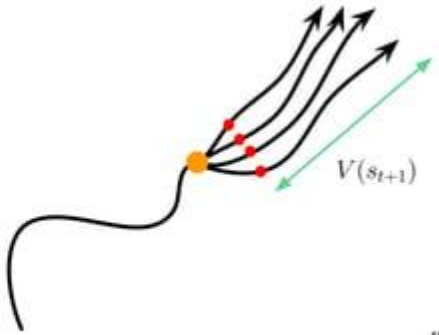
$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \underbrace{\sum_{t'=t+1}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'})|s_t, a_t]}_{V(s_{t+1})}$$



Actor-Critic algorithm

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \underbrace{\sum_{t'=t+1}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'})|s_t, a_t]}_{V(s_{t+1})}$$

$$Q^{\pi}(s_t, a_t) \approx r(s_t, a_t) + V^{\pi}(s_{t+1})$$

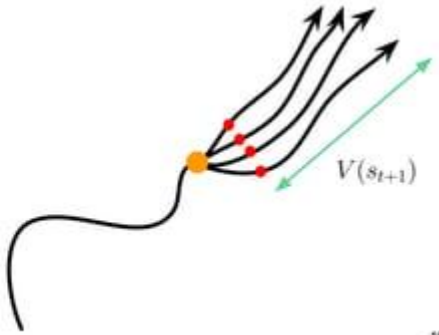


Actor-Critic algorithm

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \underbrace{\sum_{t'=t+1}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'})|s_t, a_t]}_{V(s_{t+1})}$$

$$Q^{\pi}(s_t, a_t) \approx r(s_t, a_t) + V^{\pi}(s_{t+1})$$

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



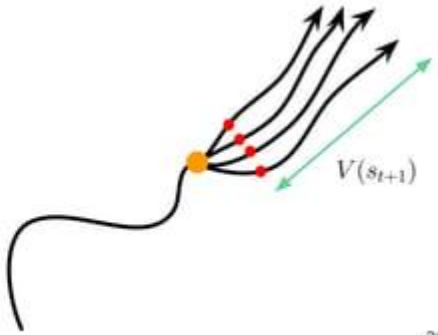
Actor-Critic algorithm

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \sum_{t'=t+1}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'})|s_t, a_t]$$

$$V(s_{t+1})$$

$$Q^{\pi}(s_t, a_t) \approx r(s_t, a_t) + V^{\pi}(s_{t+1})$$

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



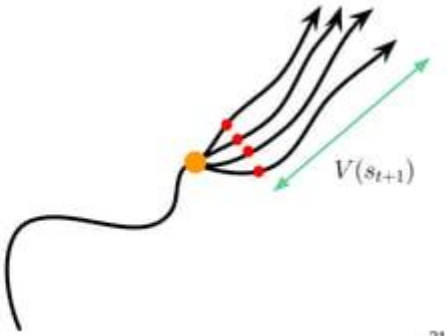
Actor-Critic algorithm

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \underbrace{\sum_{t'=t+1}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'})|s_t, a_t]}_{V(s_{t+1})}$$

$$Q^{\pi}(s_t, a_t) \approx r(s_t, a_t) + V^{\pi}(s_{t+1})$$

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

$$A^{\pi}(s_t, a_t) \approx r(s_t, a_t) + V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$

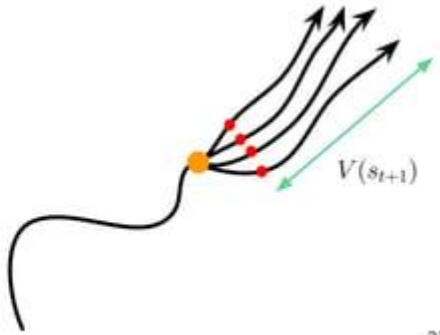


Actor-Critic algorithm

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \underbrace{\sum_{t'=t+1}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'})|s_t, a_t]}_{V(s_{t+1})}$$

We just fit the value function!

$$A^{\pi}(s_t, a_t) \approx r(s_t, a_t) + V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$



Actor-Critic algorithm: fit value function

Monte Carlo evaluation:

$$V^{\pi}(s_t) \approx \sum_{t'=t}^{T} r(s_{t'}, a_{t'})$$

we could sample multiple trajectories like this:

$$\Big(s_{i,t}, \underbrace{\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'})}_{y_{i,t}}\Big)$$

Then, compute the loss by supervised regression:

$$L(\phi) = \frac{1}{2} \sum_{i} \left\| \widehat{V}_{\phi}^{\pi}(s_i) - y_i \right\|^2$$

Actor-Critic algorithm: fit value function

TD evaluation:

$$y_{i,t} = \sum_{t'=t}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'}|s_{i,t})] \approx r(s_{i,t}, a_{i,t}) + V^\pi(s_{i,t+1}) \approx r(s_{i,t}, a_{i,t}) + \widehat{V}^\pi_\phi(s_{i,t+1})$$

training sample:

$$\left(s_{i,t}, \underbrace{r(s_{i,t}, a_{i,t}) + \widehat{V}^\pi_\phi(s_{i,t+1})}_{y_{i,t}}\right)$$

Then, compute the loss by supervised regression:

$$L(\phi) = \frac{1}{2}\sum_i \left\| \widehat{V}^\pi_\phi(s_i) - y_i \right\|^2$$

Actor-Critic algorithm

Online actor-critic algorithm:

1. Take action, get one-step experience (s, a, s', r)
2. Fit Value function

$$L(\phi) = \frac{1}{2} \sum_i \left\| \widehat{V}_\phi^\pi(s_i) - y_i \right\|^2$$

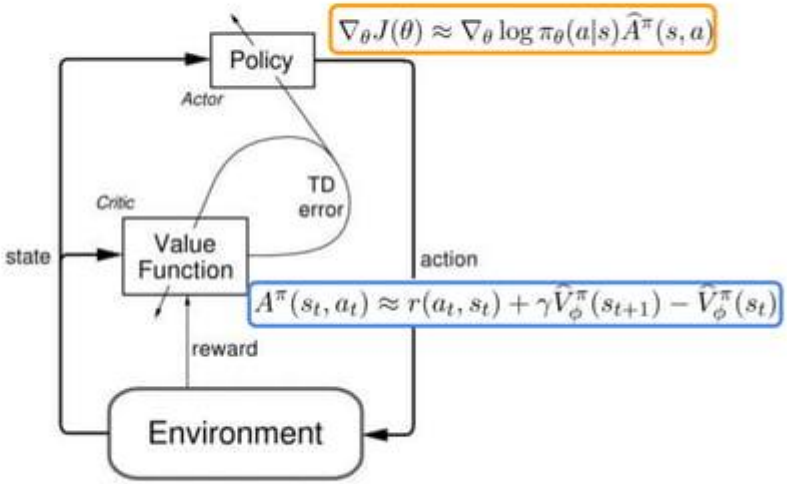
3. Evaluate advantage function

$$A^\pi(s_t, a_t) \approx r(a_t, s_t) + \gamma \widehat{V}_\phi^\pi(s_{t+1}) - \widehat{V}_\phi^\pi(s_t)$$

4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a|s) \widehat{A}^\pi(s, a)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Referenced from CS294 in UCB.

Actor-Critic algorithm



Network Architecture

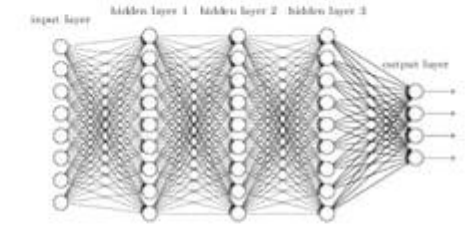
Neural architecture plays an important role in Deep Learning. In Actor-Critic algorithm, there are two kinds of network architecture:

- Separate policy network and critic network
- Two-head network

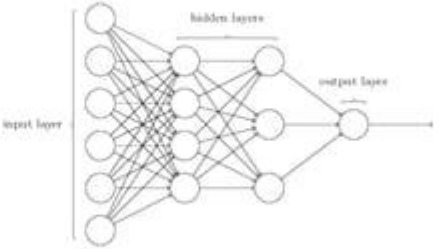
Network Architecture

Separate policy and critic network

- More parameters
- More stable in training



policy network

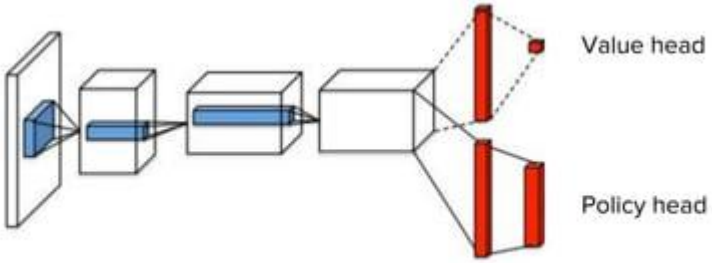


critic network

Network Architecture

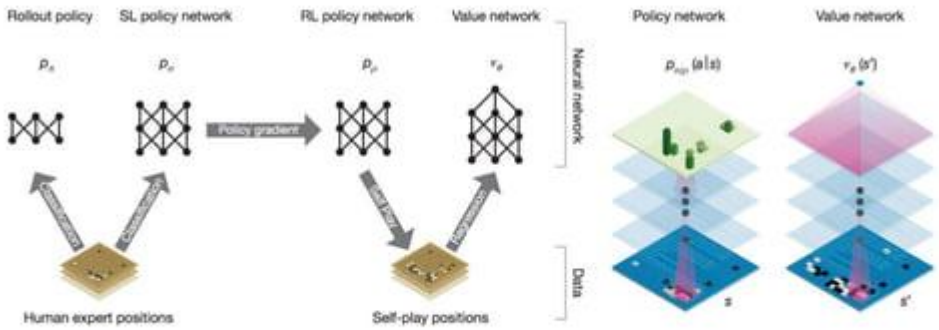
Two-head network

- Share features, less parameters
- Hard to find good coefficient to balance actor loss and critic loss



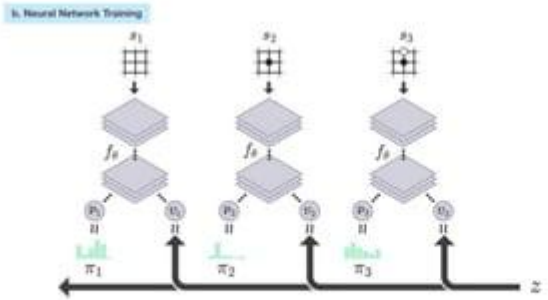
AlphaGO

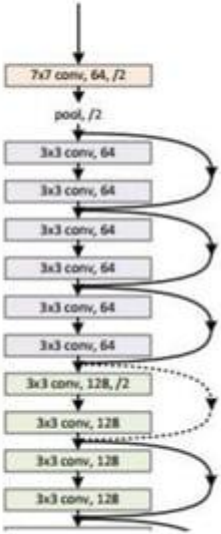
- MCTS, Actor-Critic algorithm
- Separate network architecture



AlphaGo Zero

- MCTS, Policy Iteration
- Shared ResNet





Correlation Issue

Online actor-critic algorithm:

1. Take action, get one-step experience (s, a, s', r)
2. Fit Value function

$$L(\phi) = \frac{1}{2} \sum_i \left\| \widehat{V}_\phi^\pi(s_i) - y_i \right\|^2$$

3. Evaluate advantage function

$$A^\pi(s_t, a_t) \approx r(a_t, s_t) + \gamma \widehat{V}_\phi^\pi(s_{t+1}) - \widehat{V}_\phi^\pi(s_t)$$

4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a|s) \widehat{A}^\pi(s, a)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

In online actor-critic algorithm, there still exist correlation problem.

Policy Gradient algorithm is **on-policy** algorithm so that we **cannot use replay buffer** to solve correlation problem.

Think about how to solve correlation problem in Actor-Critic.

Advance Actor-Critic algorithm

Currently, many state-of-the-art RL algorithms are developed on the basis of Actor-Critic algorithm:

- Asynchronous Advantage Actor-Critic (A3C)
- Synchronous Advantage Actor-Critic (A2C)
- Trust Region Policy Optimization (TRPO)
- Proximal Policy Optimization (PPO)
- Deep Deterministic Policy Gradient (DDPG)

Advance Actor-Critic algorithm

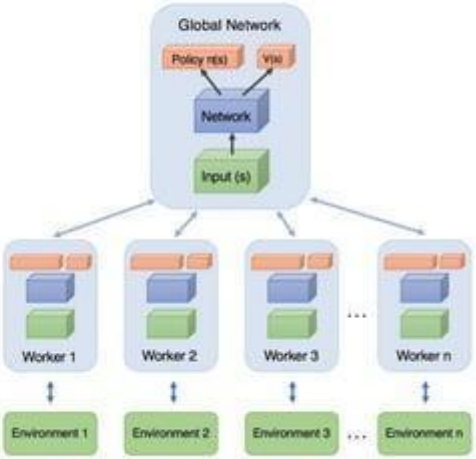
Asynchronous Advantage Actor-Critic (A3C)

V Mnih et al. (ICML 2016) proposed a parallel version of Actor-Critic algorithm which not only solves the correlation problem but also speeds up learning process. It's so-called **A3C. (Asynchronous Advantage Actor-Critic)**, A3C become the state-of-the-art baseline in 2016, 2017

Asynchronous Advantage Actor-Critic

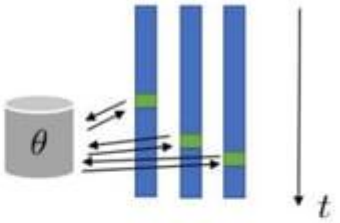
They use **multiple workers** to sample n-step experience. Each worker have **shared global network** and their **local network**.

Upon collecting enough experience, each worker computes the gradients of its local network, copies the gradients to shared global network and then does backpropagation to update global network asynchronously.



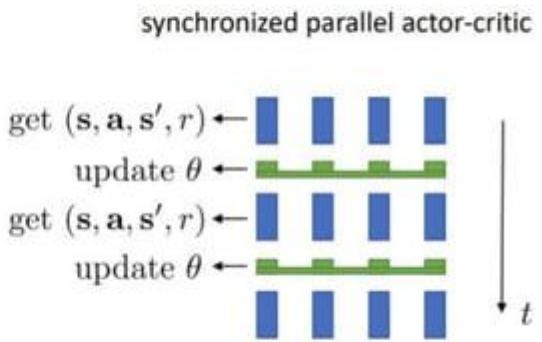
Asynchronous Advantage Actor-Critic

- Asynchronous Advantage Actor-Critic
  - proposed by DeepMind
  - use n-step bootstrapping
  - easier to implement (without lock)
  - some variability in exploration between workers
  - only use CPUs, poor GPU usage



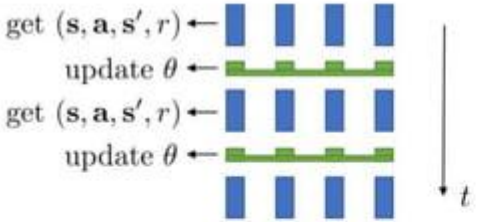
Synchronous Advantage Actor-Critic

- Synchronous Actor-Critic
  - proposed by OpenAI
  - synchronous workers
  - use n-step tricks
  - better GPU usage

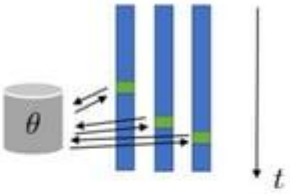


The difference between A3C and A2C

synchronized parallel actor-critic

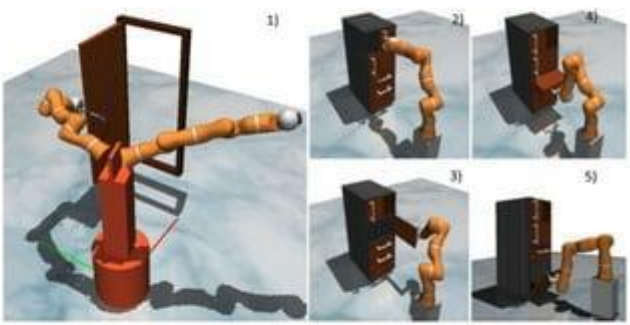


asynchronous parallel actor-critic



The usage of Actor-Critic algorithm

Gaming, model research

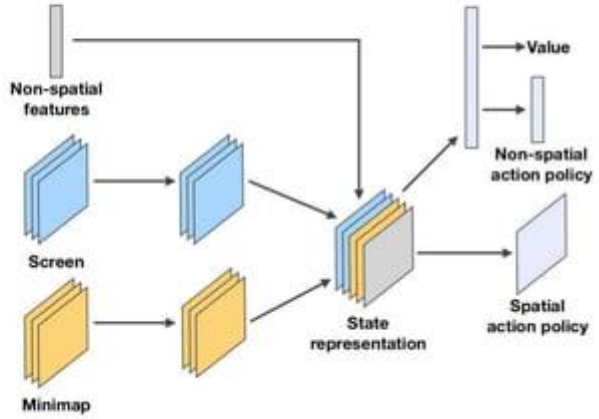


Robotics, continuous control

The devil is hidden in the details

When we do research on reinforcement learning, computation resources are needed to be considered.

A3C/A2C use much memory to cache the status for each worker in the forward step, especially in the **n-step** bootstrapping.



From StarCraft II: A New Challenge for Reinforcement Learning

Related Papers

1. Volodymyr Mnih et al. (ICML 2016): **A3C**, Asynchronous Methods for Deep Reinforcement Learning
2. John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, Pieter Abbeel. (ICML 2015): **TRPO**, Trust Region Policy Optimization
3. John Schulman et al. (2017): **PPO**, Proximal Policy Optimization Algorithms
4. Timothy P. Lillicrap et al. (ICLR 2016): **DDPG**, Continuous control with deep reinforcement learning
5. David Silver, Aja Huang et al. (Nature 2016): **AlphaGo**, Mastering the Game of Go with Deep Neural Networks and Tree Search
6. David Silver et al. (Nature 2017): **AlphaGo Zero**, Mastering the game of Go without human knowledge
7. Emilio Parisotto, Jimmy Lei Ba, Ruslan Salakhutdinov. (ICLR 2016): **Actor-Mimic**: Deep Multitask and Transfer Reinforcement Learning