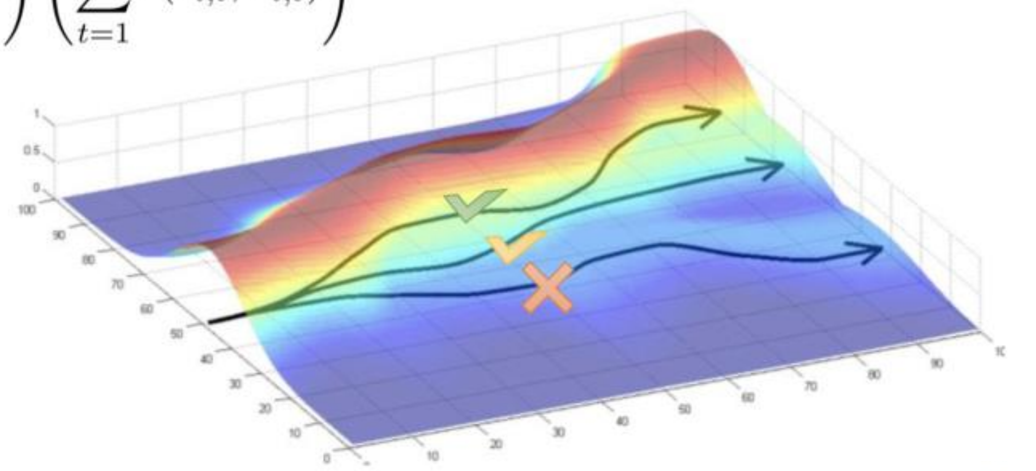
Actor-Critic Algorithm

* With some material from David Silver (UCL), Emma Brunskill (Standford), Jiehan Chen (NCKU)

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. $$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$



REINFORCE algorithm:

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

REINFORCE algorithm:

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

Objective of vanilla policy gradient:

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

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

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.

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?

Policy gradient with causality:

Policy Network

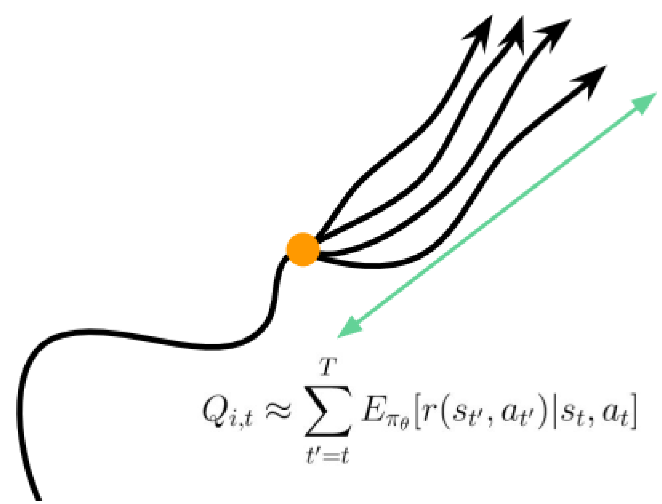
Critic Network

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

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

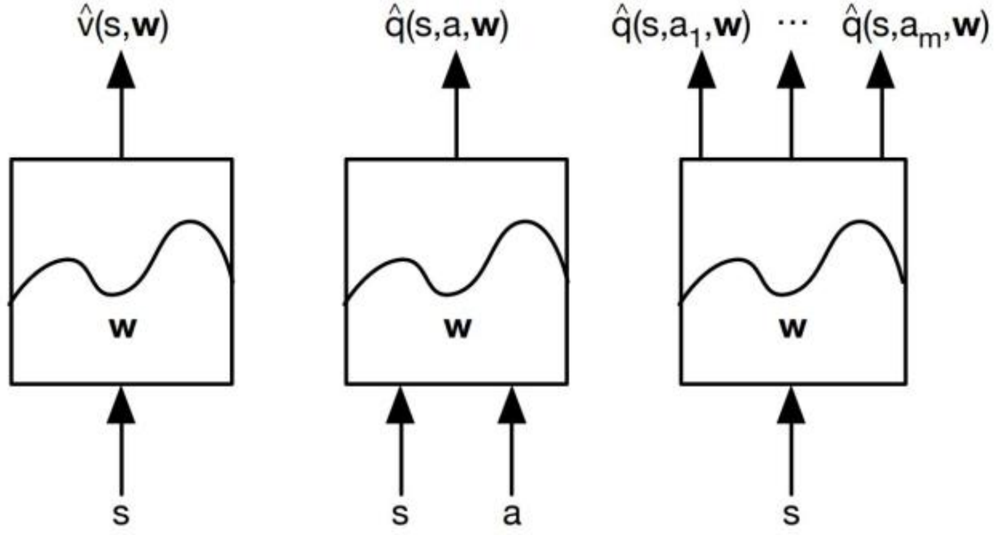
Using another neural network to approximate value function called Critic. This is so-called Actor-Critic.

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



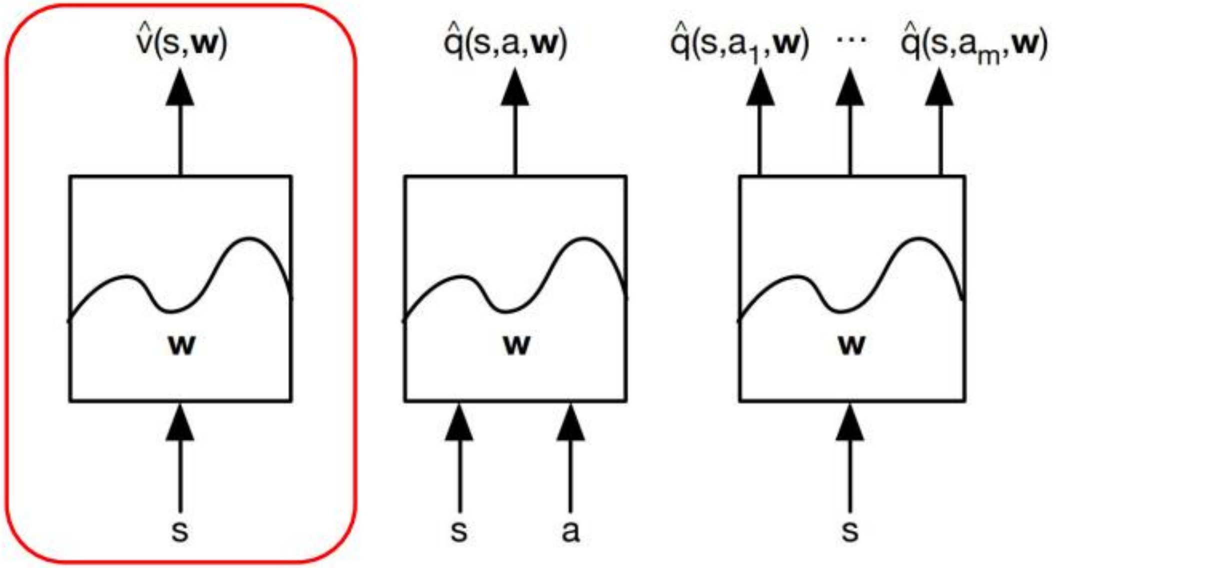
$$Q_{i,t} \approx \sum_{t'=t}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'}) | s_t, a_t]$$

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



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)

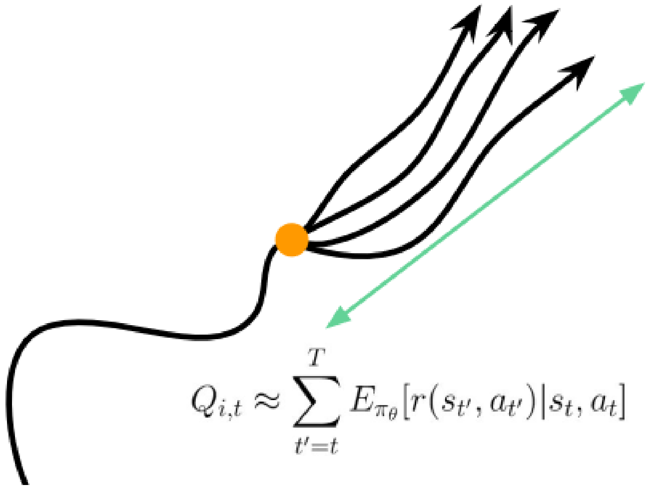


The objective of Actor-Critic

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

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?



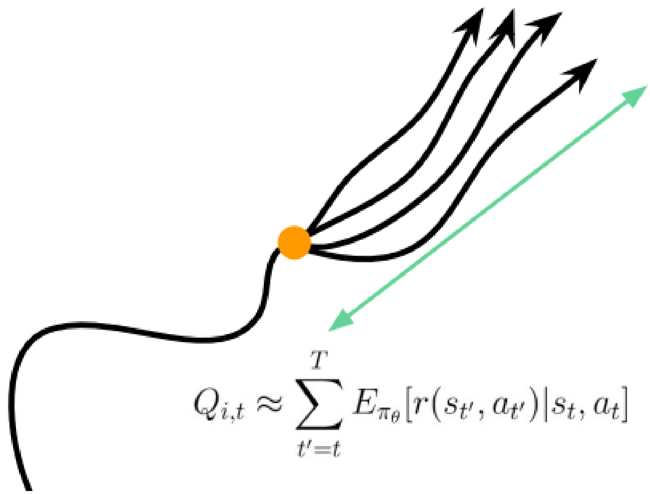
The objective of Actor-Critic:

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

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:

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



The objective of Actor-Critic with value function baseline:

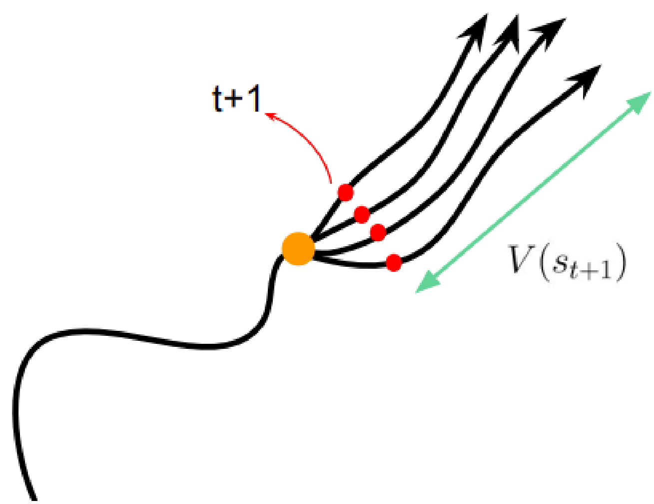
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \big(Q(s_{i,t}, a_{i,t}) - V(s_{i,t})\big)$$

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

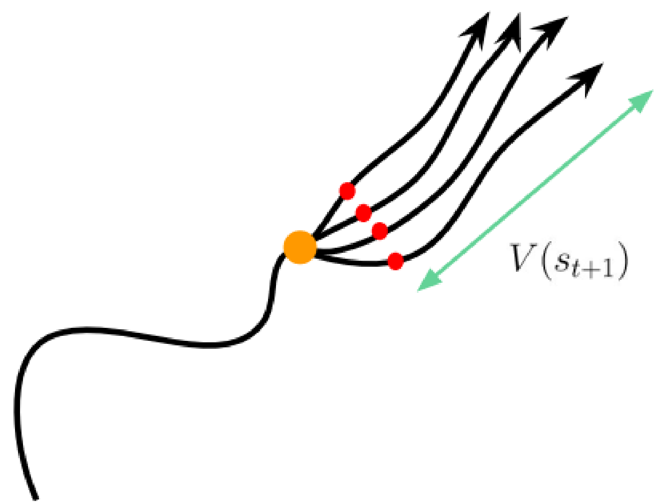
$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$$: We called this **advantage function**, which reflects how good the action we've taken compared to other candidates.

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



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

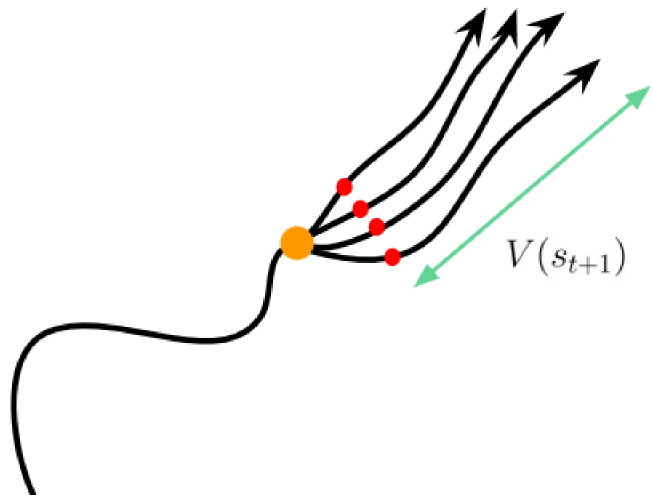
$$Q^{\pi}(s_t, a_t) \approx r(s_t, a_t) + V^{\pi}(s_{t+1})$$



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

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

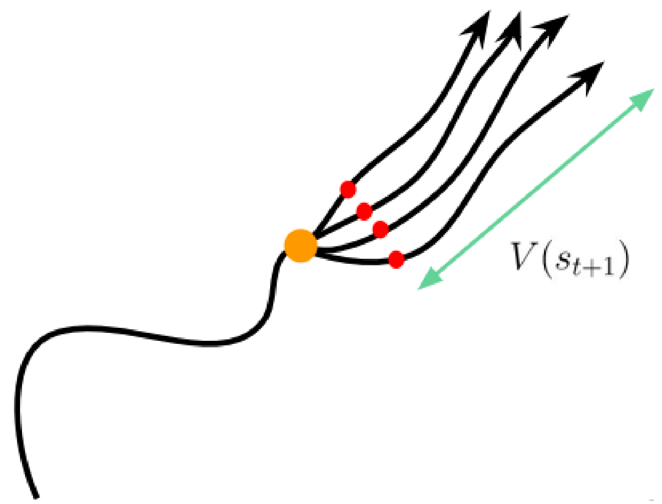
$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$$



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

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

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

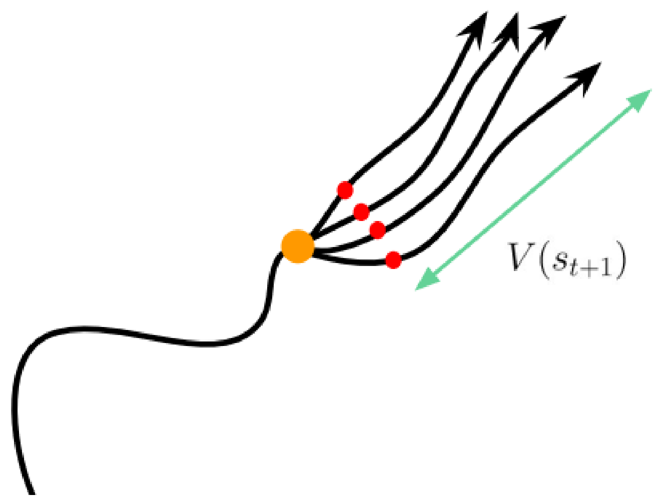


$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \underbrace{\sum_{t'=t+1}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'})|s_t, a_t]}_{V(s_{t+1})}$$

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

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

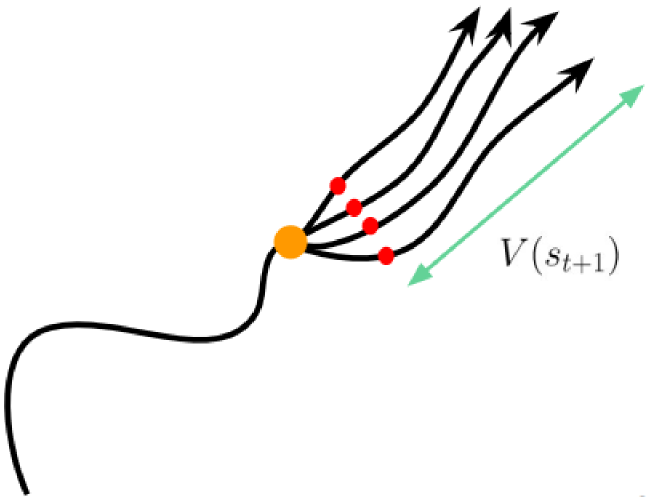
$$A^{\pi}(s_t, a_t) \approx r(s_t, a_t) + V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$



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

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:

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

Then, compute the loss by supervised regression:

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

TD evaluation:

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

training sample:

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

Then, compute the loss by supervised regression:

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

Actor-Critic algorithm

Online actor-critic algorithm:

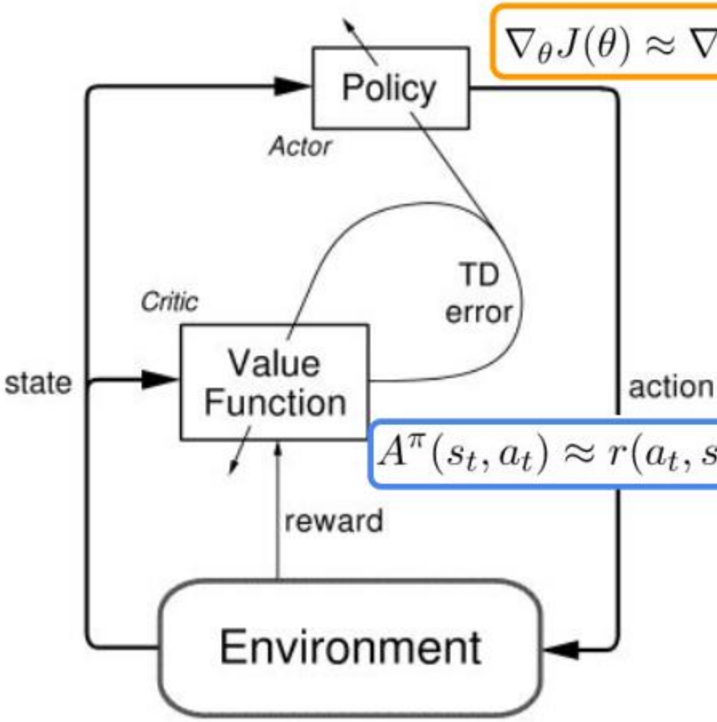
1. Take action, get one-step experience (s, a, s', r)
2. Fit Value function

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

3. Evaluate advantage function

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

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



$$\boxed{\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a|s)\widehat{A}^\pi(s,a)}$$

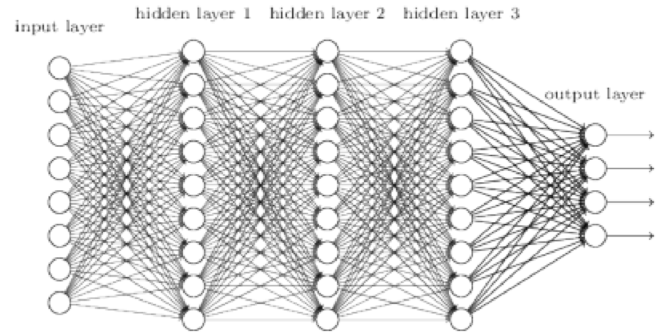
$$\boxed{A^\pi(s_t,a_t) \approx r(a_t,s_t) + \gamma \widehat{V}^\pi_\phi(s_{t+1}) - \widehat{V}^\pi_\phi(s_t)}$$

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

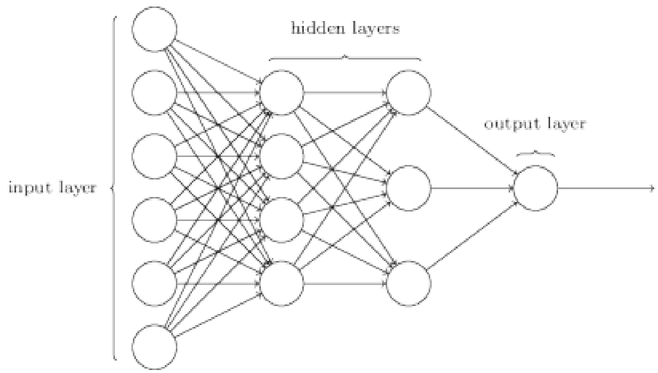
- Separate policy network and critic network
- Two-head network

Separate policy and critic network

- More parameters
- More stable in training



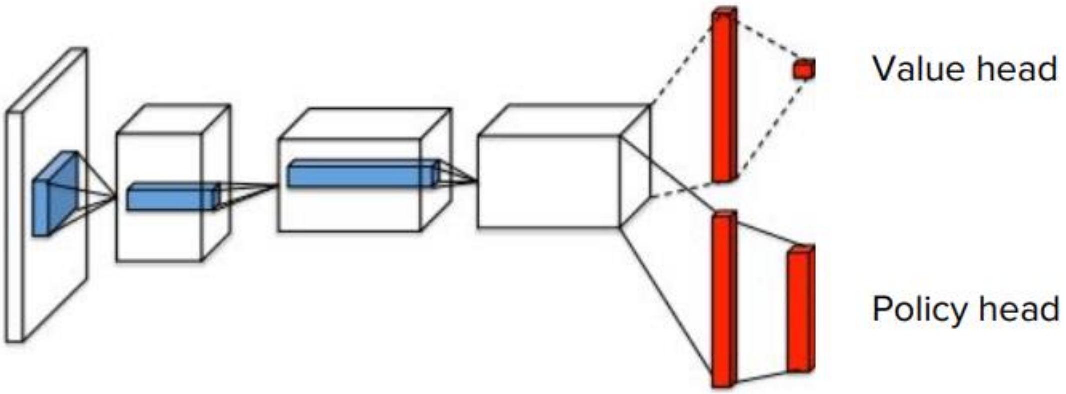
policy network



critic network

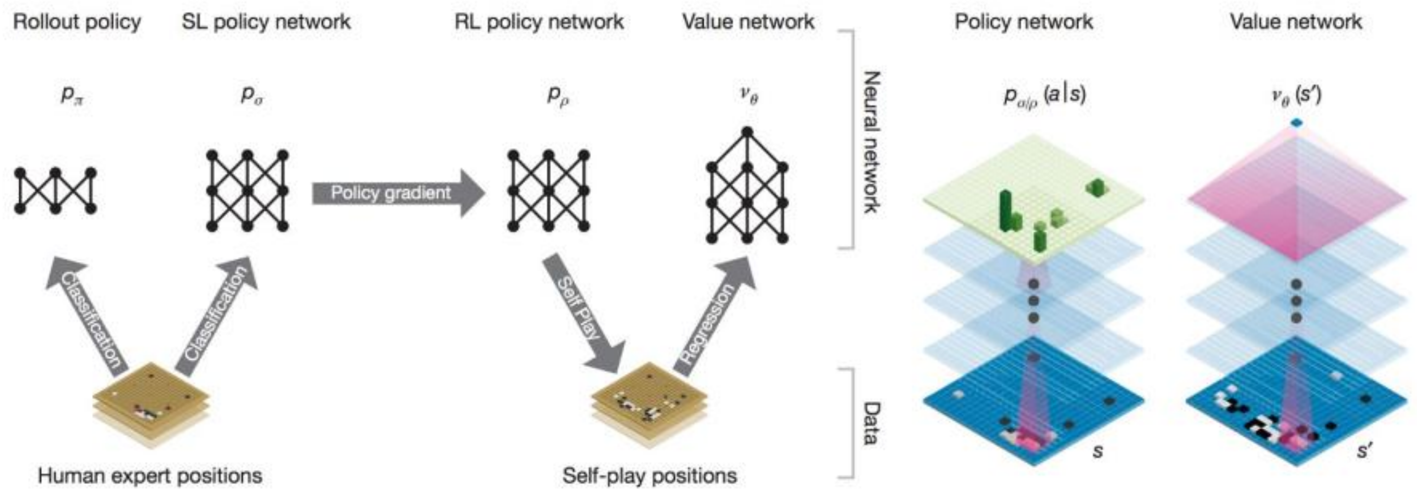
Two-head network

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



AlphaGO

- MCTS, Actor-Critic algorithm
- Separate network architecture



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)