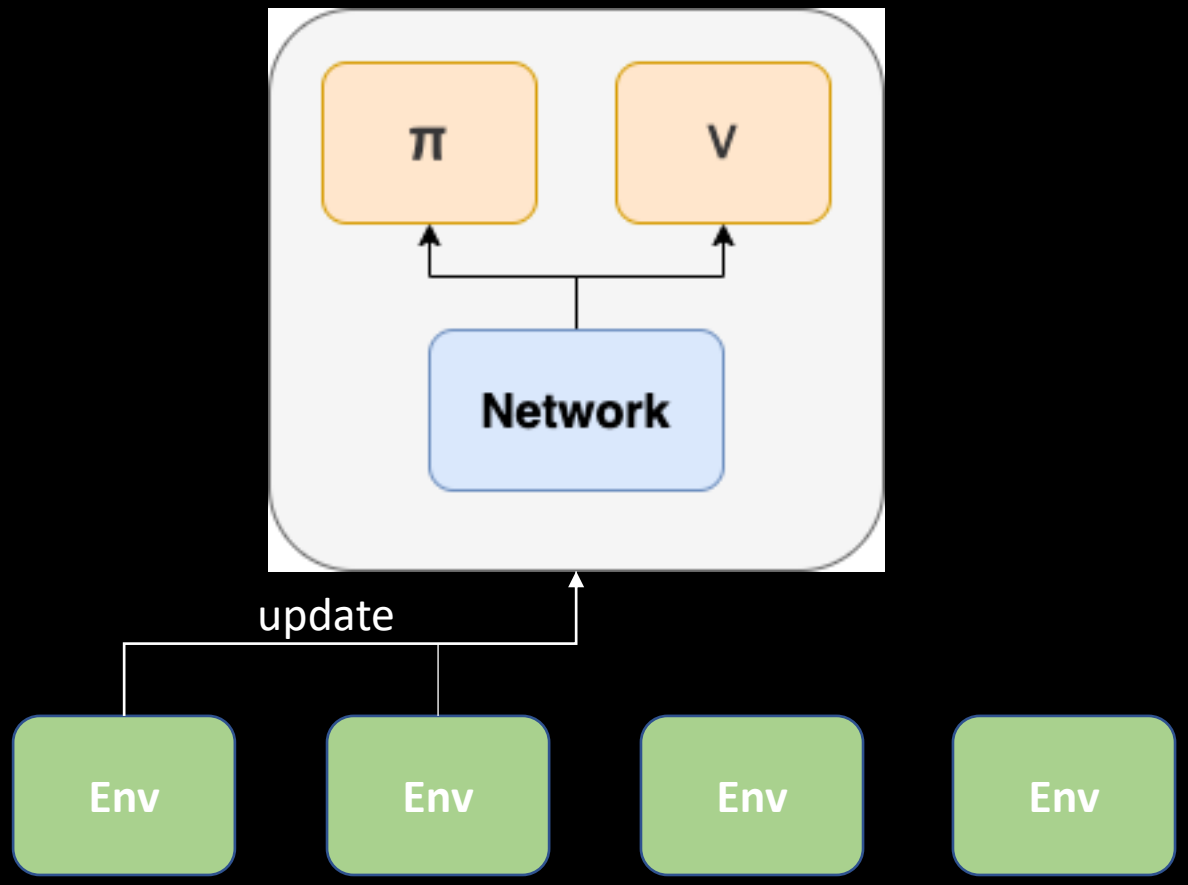
A3C

Asynchronous
Advantage
Actor-Critic

신승윤

Asynchronous



Why asynchronous?

+ agent 간의 experience가 각각 독립
+ train이 더 diverse해짐

Asynchronous update

```
#lnet : agent
#gnet : global network

for lp, gp in zip(lnet.parameters(), gnet.parameters()):
        gp._grad = lp.grad

lnet.load_state_dict(gnet.state_dict())
```

Advantage

Advantage : $A = Q(s,a) - V(s)$

Actor-Critic

$$J(\theta) = \nabla_\theta E[\sum_{t=0}^{T-1} r_{t+1} | \pi_\theta]$$

$$= \sum_{t=0}^{T-1} \nabla_\theta P(s_t, a_t | \tau) r_{t+1}$$

$$= \sum_{t=0}^{T-1} P(s_t, a_t | \tau) \frac{\nabla_\theta P(s_t, a_t | \tau)}{P(s_t, a_t | \tau)} r_{t+1}$$

Actor-Critic

기존 policy gradient 는 offline

$$\nabla_\theta log\pi_\theta(a_t|s_t)\, Q_{\pi\theta}(s_t, a_t)$$

Q를 알 수 있으면 online

Actor-Critic

$$\nabla_{\theta'} \log \pi(a_t|s_t;\theta')A(s_t, a_t;\theta,\theta_v)$$

Actor

Critic

Actor-Critic loss

$$\nabla_{\theta'} \log \pi(a_t|s_t;\theta')A(s_t,a_t;\theta,\theta_v)$$

```python
def loss_func(self, s, a, v_t):
        self.train()

        logits, values = self.forward(s)
        td = v_t - values #advantage
        c_loss = td.pow(2) #critic

        probs = F.softmax(logits, dim=1)
        m = self.distribution(probs)
        exp_v = m.log_prob(a) * td.detach().squeeze()
        a_loss = -exp_v #actor
        total_loss = (c_loss + a_loss).mean()
        return total_loss
```

Atari

https://github.com/SeungyounShin/pytorch-A3C