

Reinforcement Learning Basic

Week6. Actor Critic

박준영

Hanyang University
Department of Computer Science

지난 시간

- 정책을 근사하는 방법(Policy Gradient)
- REINFORCE 알고리즘

1 REINFORCE Issues

2 Actor Critic

REINFORCE 문제점: Full episode required

$$\mathcal{L} = G_t \log \pi_{\theta}(a|s)$$

- 손실 함수의 계산에 G_t 가 필요하다.
- G_t 는 에피소드가 끝나야만 계산할 수 있다.
- 에피소드가 길어지면 G_t 의 계산이 무거워진다.

Solution

- 가치 함수를 만들어 G_t 대신 사용한다. ← 오늘의 주제
- 벨만 방정식을 이용해 계산을 중간에 끝낸다.

REINFORCE 문제점: High gradient variance

$$\mathcal{L} = G_t \log \pi_{\theta}(a|s)$$

- G_t 의 값은 환경에 따라 매우 천차만별이다.
- G_t 의 값이 매우 커지면 gradient가 매우 커진다.
- 학습이 불안정해진다.

Solution

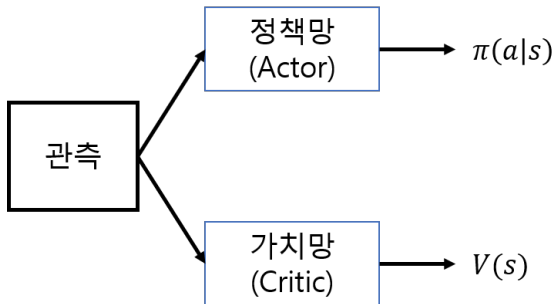
다음과 같이 **기준값(baseline)**을 손실 함수에 도입한다.

$$\mathcal{L} = (q_{\pi}(s, a) - b(s)) \pi_{\theta}(a|s)$$

Baseline이 될 수 있는 값들:

- 표준화 $\left(Z = \frac{X - \mu}{\sigma} \right)$
- 이동 평균(Moving Mean)
- $V(s) \leftarrow$ 오늘의 주제

Actor Critic



DP 관점에서의 Actor Critic

Actor Critic은 정책 반복의 구조를 사용해 학습한다.

$$\pi_0 \rightarrow v_{\pi_0} \rightarrow \pi_1 \rightarrow v_{\pi_1} \rightarrow \pi_2 \rightarrow \cdots \rightarrow \pi_* \rightarrow v_*$$

- 정책 발전 \rightarrow 정책망의 업데이트.
- 정책 평가 \rightarrow 가치망을 이용해 정책을 평가.

Actor Critic 알고리즘

- 1 인공 신경망을 랜덤하게 초기화한다.
- 2 환경과 상호작용 하여 학습 데이터를 만들.
- 3 예상 반환값을 계산.
- 4 Actor와 Critic의 손실 함수를 계산.
- 5 SGD로 각 인공 신경망을 갱신함.
- 6 수렴할 때까지 2로 돌아감.

Actor Loss Function

REINFORCE 알고리즘에서의 G_t 를 큐함수로 대체한다.

$$\mathcal{L} = q_{\pi}(s, a) \log \pi_{\theta}(a|s)$$

- 그런데, REINFORCE의 문제점 중 높은 분산은 해결되지 않았다.
→ baseline으로 상태 가치 함수 $v(s)$ 를 사용한다.

$$\mathcal{L} = (q_{\pi}(s, a) - v(s)) \log \pi_{\theta}(a|s)$$

Actor Loss Function

- 하지만 큐함수와 상태 가치 함수를 모두 알고 있어야 하는 번거로움이 있다.
- 큐함수는 상태 가치 함수로 표현할 수 있다.

$$q_{\pi}(s, a) = R + \gamma v_{\pi}(s')$$

따라서 최종적인 정책망의 손실 함수는

$$\mathcal{L}_{\text{actor}} = (R + \gamma v(s') - v(s)) \log \pi_{\theta}(a|s)$$

Critic Loss Function

시간차(TD) 학습을 이용해 가치망을 갱신한다.

$$\mathcal{L}_{\text{critic}} = [R + \gamma v(s') - v(s)]^2$$

구현하기

```
class ActorNet(nn.Module):  
    def __init__(self):  
        super(ActorNet, self).__init__()  
  
        self.net = nn.Sequential(  
            nn.Linear(4, 64),  
            nn.ReLU(inplace=True),  
            nn.Linear(64, 2),  
            nn.Softmax(dim=0)  
        )  
  
    def forward(self, x):  
        return self.net(x)
```

구현하기

```
class CriticNet(nn.Module):  
    def __init__(self):  
        super(CriticNet, self).__init__()  
  
        self.net = nn.Sequential(  
            nn.Linear(4, 64),  
            nn.ReLU(inplace=True),  
            nn.Linear(64, 1)  
        )  
  
    def forward(self, x):  
        return self.net(x)
```

구현하기



```
class Agent:
    def __init__(self, device):
        self.device = device

        self.actor = ActorNet().to(device)
        self.actor_opt = optim.Adam(self.actor.parameters(), lr=LEARNING_RATE)

        self.critic = CriticNet().to(device)
        self.critic_opt = optim.Adam(self.critic.parameters(), lr=LEARNING_RATE)
```


구현하기

```
def get_action(self, state):  
    policy = self.actor(state.to(self.device))  
  
    m = Categorical(policy)  
    action = m.sample()  
  
    return action.item(), m.log_prob(action)
```

구현하기

```
def train(self, state, log_prob, next_state, reward, done):
    state, next_state = state.to(self.device), next_state.to(self.device)

    value, next_value = self.critic(state), self.critic(next_state)

    if done:
        target = torch.tensor(reward).to(device)
    else:
        target = reward + DISCOUNT_FACTOR * next_value

    adv = target - value

    self.actor_opt.zero_grad()
    actor_loss = -log_prob * adv.detach()
    actor_loss.backward()
    self.actor_opt.step()

    self.critic_opt.zero_grad()
    critic_loss = (target.detach() - value) ** 2
    critic_loss.backward()
    self.critic_opt.step()
```

구현하기



```
while True:
    obs = env.reset()
    obs = torch.FloatTensor(obs)

    total_reward = 0

    while True:
        action, log_prob = agent.get_action(obs)
        next_obs, reward, done, _ = env.step(action)

        total_reward += reward

        next_obs = torch.FloatTensor(next_obs)

        agent.train(obs, log_prob, next_obs, reward, done)

        if done:
            break

    obs = next_obs
```