Reinforce Learning

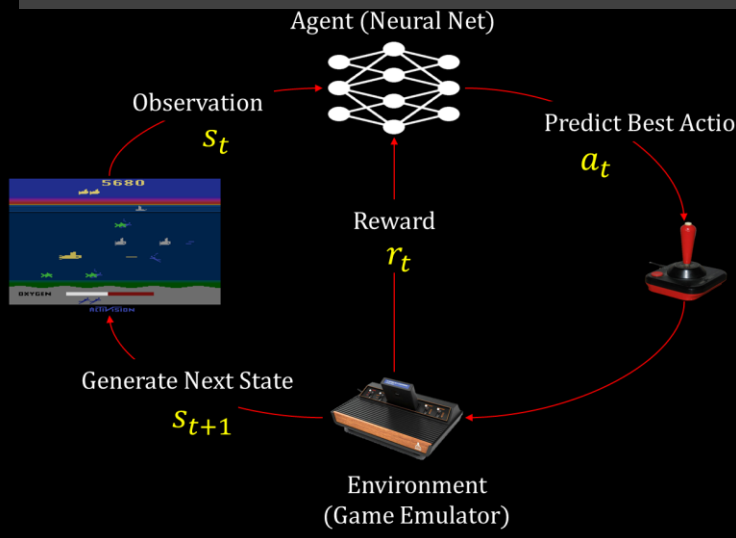
Dueling DQN

소프트웨어 꼰대 강의

노기섭 교수

(kafa46@hongik.ac.kr)

Recap: DQN Architecture



Choose Max Q-value Action

$Q(s_t, a_1)$ (Action 1)

$Q(s_t, a_2)$ (Action 2)

$Q(s_t, a_3)$ (Action 3)

$Q(s_t, a_4)$ (Action 4)

FC-4 (Q-values)

FC-256

32 4 ×4 Conv, Stride 2

16 8 × 8 Conv, Stride 4

Current state S_t: 84 × 84 × 4
stack of last 4 frames

Recap: Double DQN

Follow-up research: Double DQN

Almost same, but different tricks!

A dragon chases its own tail ^^

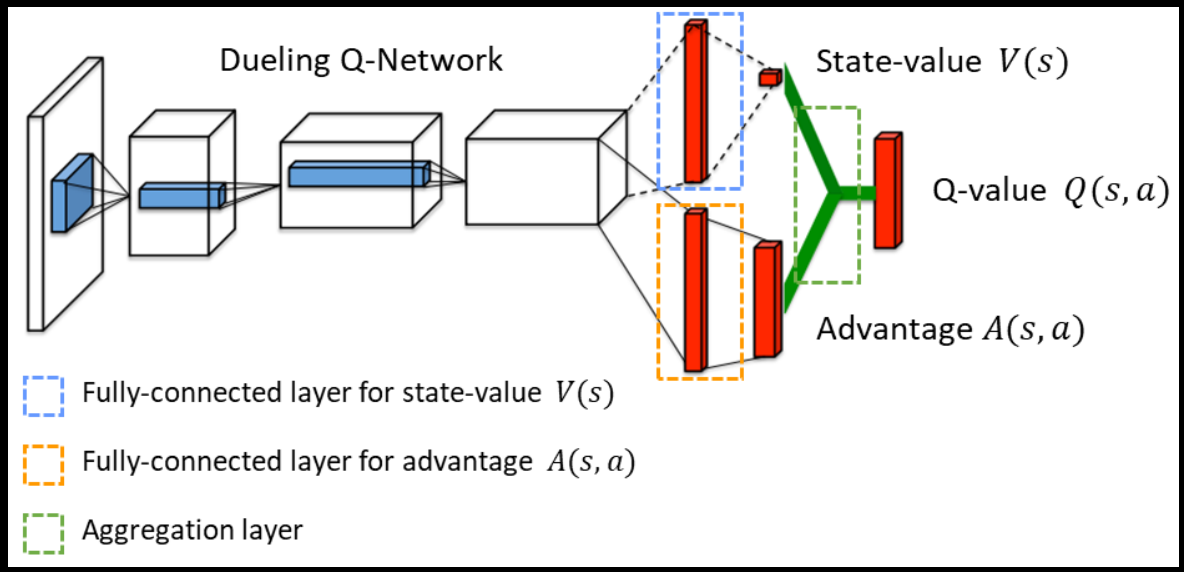
Two dragons fly for a goal ^^

Use different Q-networks for target & prediction

Paper: https://arxiv.org/pdf/1509.06461

Follow-up research

Follow-up research: Dueling DQN - Decompose $Q(s, a)$



Paper: https://arxiv.org/abs/1511.06581

Use 2 streams

- State value $V(s)$
- Advantage for each action $A(s, a)$

Prioritized Experience Replay (PER, 중요한 경험을 더 자주 학습)

Motivation

In many environments, the choice of action does not significantly affect the outcome in certain states.

Robot Arm Example

For example:

- When an agent is in a "waiting" or "idle" state,
- All actions may produce similar results,
- But the state itself still has an inherent value!

No move of robot arm

No matter how to move the arm
(No difference between actions)

But the arm was in danger

Action은 중요하지 않지만,
State는 중요할 수 있음!

Problem!

Single stream mixes two different signals:

- state quality
- action advantage

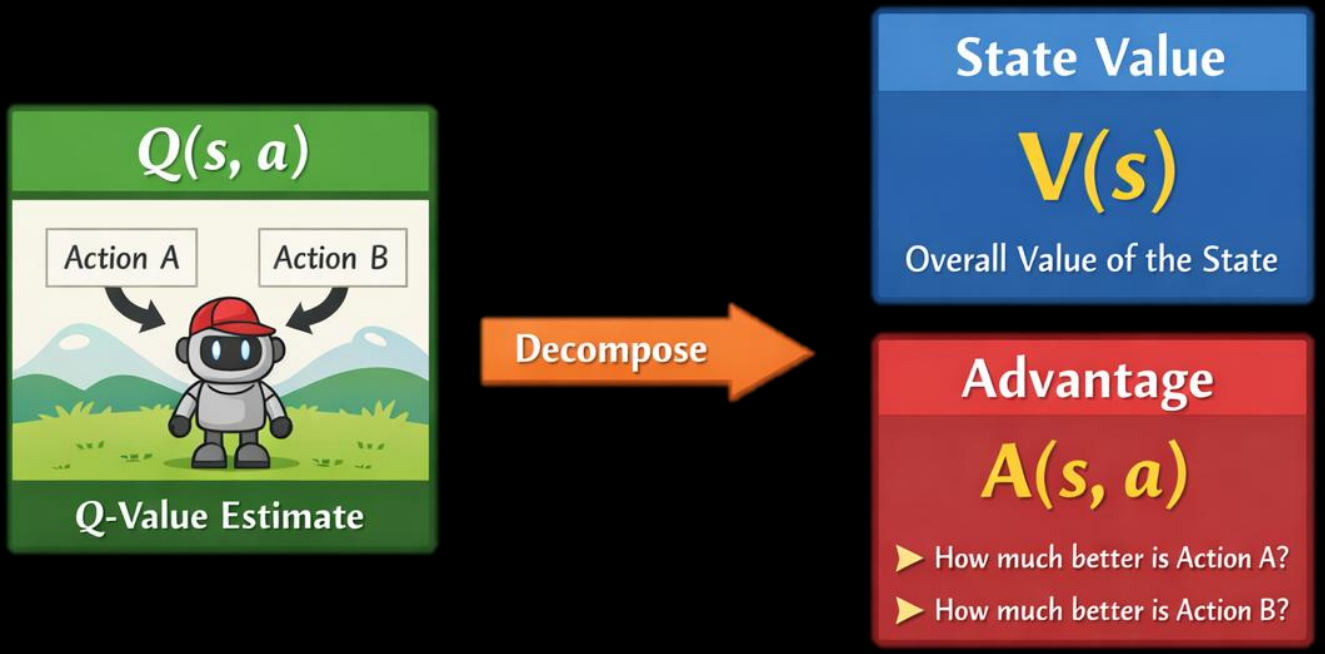
Key Idea

Decomposite network

$$Q(s,a) = V(s) + A(s,a)$$

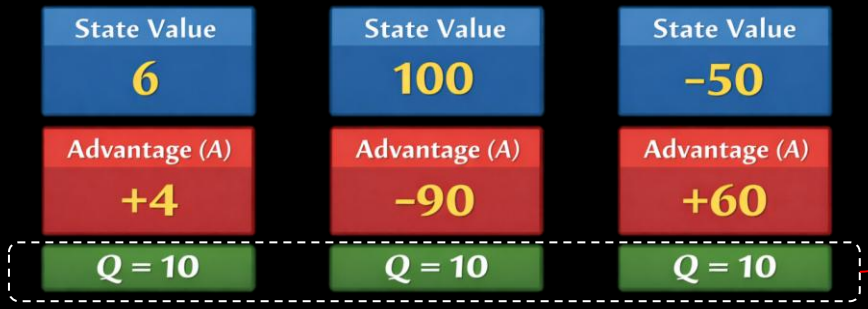
The state value

The advantage function representing how much better action a is compared to others



Identifiability Issue

The naive decomposition is not unique!



Identifiability problem

(식별 불가능 문제)

V + A always equals Q = 10!

The state value: Role of Baseline

Only represent difference

$$Q(s,a) = V(s) + [\, A(s,a) - mean(A) \,]$$

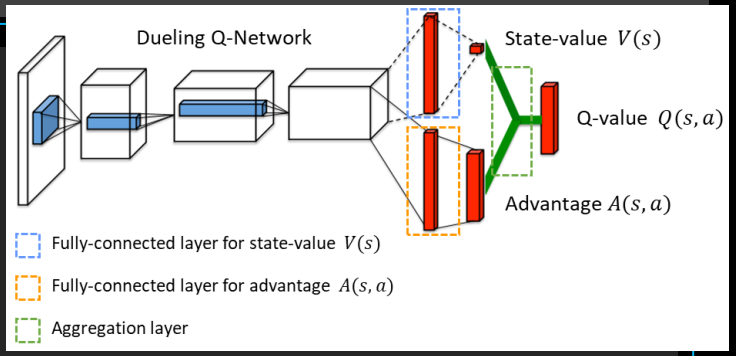
$$Q(s,a) = V(s) + \left[A(s,a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s,a') \right]$$

- Advantage는 "절댓값" 이 아니라
- "행동 간 상대적 차이" 만 표현해야 한다

➔ 평균을 빼서 기준점을 반영

Implementation Sketch

```python
class DuelingDQN(nn.Module):
    def __init__(self, state_dim, action_dim):
        super().__init__()
        self.feature = nn.Sequential(
            nn.Linear(state_dim, 128), nn.ReLU()
        )
        # Value stream
        self.value = nn.Sequential(
            nn.Linear(128, 128), nn.ReLU(), nn.Linear(128, 1)
        )
        # Advantage stream
        self.advantage = nn.Sequential(
            nn.Linear(128, 128), nn.ReLU(), nn.Linear(128, action_dim)
        )
    def forward(self, x):
        feat = self.feature(x)
        V = self.value(feat)
        A = self.advantage(feat)
        Q = V + (A - A.mean(dim=1, keepdim=True))
        return Q
```



State Value

V(s)

Overall Value of the State

Advantage

A(s, a)

- How much better is Action A?
- How much better is Action B?

Upgrading Codes

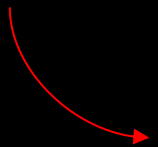
Add terminal argument (e.g., --model)

- train.py
- play_with_model.py

Create a model factory: e.g., def build_model( )

Modify model save functionality

- store_manager.py



Upgrade & differentiate

the path for DuelingDQN

수고하셨습니다 ..^^..