Reinforce Learning

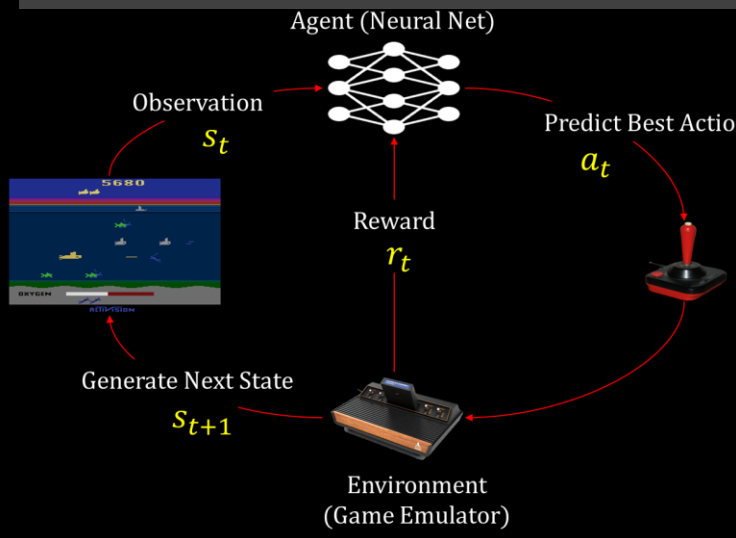
Double Deep Q Learning (Double DQN)

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 Idea

Follow-up research: Double DQN

Same structure,
but different roles!

→ Decouple action selection
and evaluation

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

Problem in Vanilla DQN

Vanilla DQN

Training Procedure

Sample random minibatch of transition $(\phi_j,\ a_j,\ r_j,\ \phi_{j+1})$ from D

Set $y_j = \begin{cases} r_j & \text{for teminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'\ ; \theta^-) & \text{for non-teminal } \phi_{j+1} \end{cases}$

Perform a gradient descent step on $\left(y_i - Q(\phi_j,\ a_j\ ; \theta)\right)^2$ according to Eq. 3

Target (Bellman)

Prediction (output) from NN

Same Q-value estimates used for both

- action selection (argmax)
- value evaluation

leads to overestimation bias!

Max operator bias!

next state s'

↓

Target Network Q_target $(s', a_1\ ; \theta)$
Target Network Q_target $(s', a_2\ ; \theta)$
Target Network Q_target $(s', a_3\ ; \theta)$
Target Network Q_target $(s', a_4\ ; \theta)$

↓

$\arg\max_a Q(s', a)$ # selection

↓

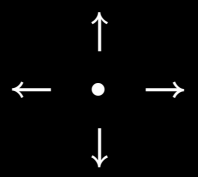
$Q(s', a^*)$ # evaluation

↓

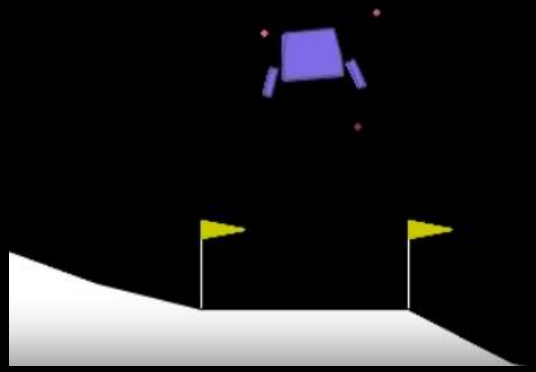
$y = r + \gamma Q(s', a^*)$

Overestimation Intuition

$$y_i = r_j + \gamma \max_{a'} Q(\phi_{j+1}, a' ; \theta^-)$$



Action	True Q	Estimated Q ($\hat{Q}$)
UP	5	4.9
DOWN	5	5.3
LEFT	5	6.8
RIGHT	5	5.1



DQN Target: $\max_a Q(s, a) = 6.8$

True value: 5

Overestimation: +1.8

$$\hat{Q} = Q + error\ (noise)$$

$$\mathbb{E}[\max \hat{Q}] \geq \max Q$$

Max over same noisy estimates!

Core Idea of Double DQN

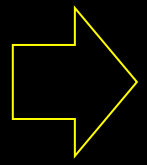
Instead of

$$\max_{a'} Q$$

Use

Online Network → Select action

Target Network → Evaluate value



Decouple selection and evaluation to reduce overestimation bias

$$y_i = r_j + \gamma Q\left(\phi_{j+1}, \arg\max_{a'} Q(s', a'; \theta) ; \theta^-\right)$$

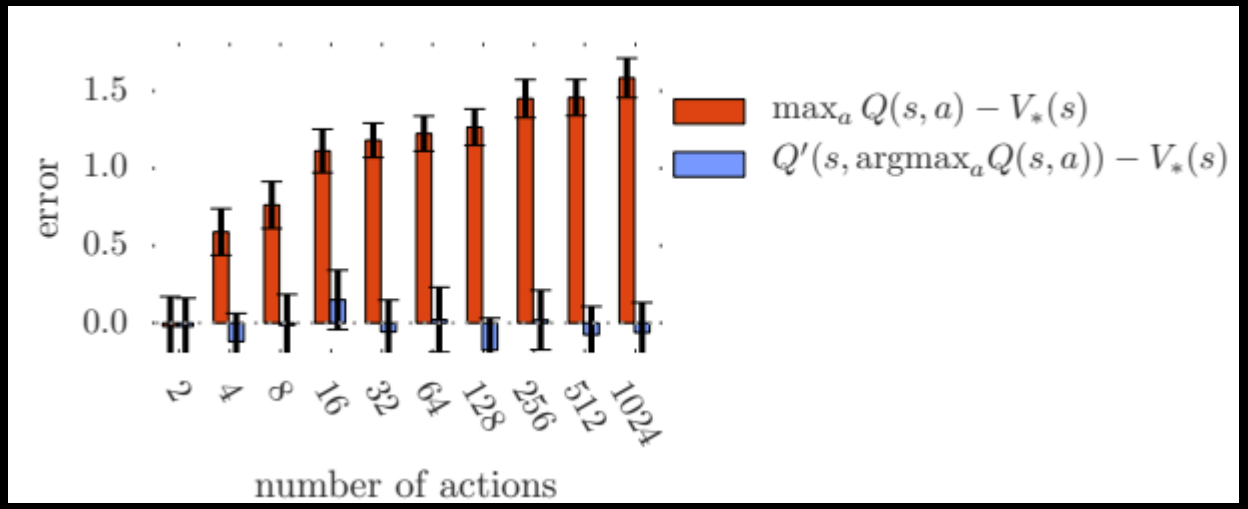
$$a^* = \arg\max_{a'} Q(s', a'; \theta) \quad \text{\# online network (Q-net)}$$

$$y_i = r_j + \gamma Q(s', a^*; \theta^-) \quad \text{\# target network}$$

, where θ^- is the target network.

Algorithm Enhancement

Feature	DQN	Double DQN
Action selection	target network	online network
Value evaluation	target network	target network
Bias	high	reduced
Stability	moderate	higher
Policy quality	lower	better



All bars are the average of 100 repetitions.

Implementation Sketch

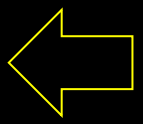
```python
'''dqn_algorithms.py'''
import torch
from torch import Tensor, nn

@torch.no_grad()
def compute_bootstrap_target(
    qnet: nn.Module,
    target: nn.Module,
    next_state: Tensor,
    reward: Tensor,
    done: Tensor,
    gamma: float,
    use_double_q: bool,
) -> Tensor:
    if use_double_q:
        # Double DQN:
        # 1) select action with online network
        # 2) evaluate selected action with target network
        next_actions = qnet(next_state).argmax(dim=1,
keepdim=True)
        next_q = target(next_state).gather(1, next_actions)
    else:
        # Vanilla DQN:
        # max over target network action-values
        next_q = target(next_state).max(dim=1,
keepdim=True).values

    return reward + gamma * next_q * (1.0 - done)
```

```python
'''agent.py'''
from dqn_algorithms import compute_bootstrap_target

# DQN only
# with torch.no_grad():
#     max_next_q = self.target(
#    next_state).max(dim=1, keepdim=True).values
#     y = reward + self.gamma * max_next_q * (1.0 - done)

# Double DQN + DQN
y = compute_bootstrap_target(
    qnet=self.qnet,
    target=self.target,
    next_state=next_state,
    reward=reward,
    done=done,
    gamma=self.gamma,
    use_double_q=self.use_double_q,
)
```

Double DQN Usage in our code

Double DQN Usage

```
python3 train.py \
   --env LunarLander-v3 \
   --episodes 1000 \
   --render_mode none \
   --double_q \
   --save_best \
   --avg_window 20
```



Train Double DQN

Use Stored Model

```
python3 play_with_model.py \
   --checkpoint \
     ./results/trained_models/best_LunarLander-v3_ep-1000_avg-window-20.pth \
   --episodes 5 \
   --render_mode human
```

수고하셨습니다 ..^^..