기술 백서: Continuous Control with Deep Reinforcement Learning (DDPG)

〔Paper Title / Core Technology〕

Continuous Control with Deep Reinforcement Learning (Deep Deterministic Policy Gradient - DDPG)

1. 설계 철학 및 문제 정의 (Architectural Philosophy)

기존 기술의 임계점 (Legacy Bottleneck):

• 근본 원인 (Root Cause): 당시 SOTA였던 DQN (Deep Q-Network)은 행동 공간이 이산적(Discrete)일 때만 작동하도록 설계됨. 연속적인 행동 공간(Continuous Action Space)에서 최적 행동 $a^* = \max_a Q(s,a)$를 찾으려면, 매 스텝마다 복잡한 반복 최적화(Iterative Optimization)를 수행해야 함.

• 치명적 한계 (Critical Failure): 로봇 제어와 같은 고차원 자유도(DoF)를 가진 시스템에서 행동을 이산화(Discretization)하면, 차원의 저주(Curse of Dimensionality)로 인해 행동 공간이 기하급수적으로 폭발하거나($3^7=2187$), 세밀한 제어가 불가능해짐.

• 패러다임 전환 (Paradigm Shift): 본 논문은 DQN의 성공 요인(Replay Buffer, Target Network)을 Deterministic Policy Gradient (DPG) 알고리즘과 결합하여, 연속 행동 공간에서도 안정적으로 작동하는 Model-free, Off-policy Actor-Critic 아키텍처(DDPG)를 제안함.

개념 시각화 (Conceptual Analogy):

〉〔Analogy〕 DQN이 객관식 시험(이산적 선택지)에서 정답을 고르는 법을 배우는 학생이라면, DDPG는 골프 스윙의 각도와 힘(연속적 값)을 미세하게 조정하며 배우는 프로 선수와 같다. 코치(Critic)는 선수의 자세를 보고 "점수(Q-value)"를 매겨주기만 하면, 선수(Actor)는 그 점수를 높이는 방향(Gradient)으로 자세를 조금씩 수정한다.

2. 수학적 원리 및 분류 (Mathematical Formalism)

시스템 분류 (System Taxonomy):

• 아키텍처 유형: Actor-Critic Architecture

• 학습 방식: Model-free, Off-policy

• 정책 유형: Deterministic Policy ($\mu : S \rightarrow A$)

핵심 수식 및 상세 해설 (Core Formulation & Breakdown):

1. Deterministic Policy Gradient (Actor Update):

$\nabla_{\theta^{\mu}} J \approx (1/N) \Sigma_i \nabla_a Q(s, a \mid \theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s \mid \theta^{\mu})|_{s_i}$

- Variable Definition (변수 정의):
- θ^{μ} : Actor 네트워크 파라미터 / θ^Q: Critic 네트워크 파라미터
- $\nabla_a Q$: 행동 a에 대한 Q 값의 변화율 (기울기)
- $\nabla_{\theta^{\mu}} \mu$: 파라미터 변화에 따른 행동의 변화율
- Physical Meaning (수식의 물리적 의미):
- Chain Rule을 적용한 것이다. "행동을 어떻게 바꿔야 Q값이 오르는가($\nabla_a Q$)?"와 "파라미터를 어떻게 바꿔야 행동이 그렇게 변하는가($\nabla_{\theta^{\mu}} \mu$)?"를 곱하여, 최종적으로 Q값을 높이는 방향으로 Actor 파라미터를 업데이트한다. 이는 확률 분포를 학습하는 Stochastic Policy Gradient보다 데이터 효율성이 높다.

2. Critic Loss Function (Bellman Equation):

$L = (1/N) \Sigma_i (y_i - Q(s_i, a_i \mid \theta^Q))^2$

$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} \mid \theta^{\mu'}) \mid \theta^{Q'})$

- Variable Definition:
- y_i: Target Value (정답지 역할)
- Q', μ': Target Networks (안정적인 학습을 위해 천천히 업데이트되는 복사본)
- Physical Meaning:
- 현재 상태-행동의 가치(Q)가 실제 보상(r)과 다음 상태의 예상 가치(Q')의 합과 같아지도록 학습한다(TD Learning). Target Network를 사용하여 y_i가 진동하는 것을 막아 학습 안정성을 확보한다.

3. Soft Target Update:

$\theta' \leftarrow \tau \theta + (1 - \tau) \theta' \quad (\text{with } \tau \ll 1)$

- Physical Meaning:
- Target Network의 파라미터를 한 번에 덮어쓰지 않고, 매 스텝 아주 조금씩(예: 0.1%)만 가져온다. 이는 급격한 가치 추정 변화(Oscillation)를 억제하여 전반적인 학습 과정을 부드럽게(Smooth) 만든다.

3. 실행 파이프라인 및 데이터 흐름 (Execution Pipeline)

입력 명세 (Trace Spec):

- Input Context: State Vector `[Batch, 3]` (e.g., Pendulum: θ, θ, θ)
- Output: Action Vector `[Batch, 1]` (e.g., Torque $\in [-2.0, 2.0]$)

순전파 로직 (Forward Propagation Logic):

1. Interaction & Exploration:

- Action Selection: $a_t = \mu(s_t | \theta^\mu) + N_t$
- Mechanism: Deterministic 행동에 Ornstein-Uhlenbeck Noise N을 더해 탐색(Exploration) 수행.
- Objective: 물리 시스템의 관성(Inertia)을 고려한 시간적 상관관계가 있는 탐색 노이즈 생성.
- Storage: `Replay Buffer` ≤ftarrow `(s_t, a_t, r_t, s_{t+1})`

2. Training Batch Sampling:

- Input: Random Minibatch `N=64`
- Data: `batch_s`, `batch_a`, `batch_r`, `batch_s_next`

3. Target Calculation (Forward):

- Next Action: $a'_{next} = \mu'(s_{next})$ (Target Actor)
- Target Q: $y = r + \gamma \ Q'(s_{next}, a'_{next})$ (Target Critic)
- Shape: `〔64, 1〕`

4. Critic Update (Forward/Backward):

- Prediction: $Q_{pred} = Q(s, a)$
- Loss: MSE $(y - Q_{pred})^2$
- Update: θ^Q ≤ftarrow $\theta^Q - \alpha \ \nabla L$

5. Actor Update (Forward/Backward):

- Action Pred: $a_{pred} = \mu(s)$
- Q Evaluation: $Q_{val} = Q(s, a_{pred})$
- Gradient: Maximizing Mean Q Value (Gradient Ascent)
- Update: θ^μ ≤ftarrow $\theta^\mu + \beta \ \nabla J$

4. 학습 메커니즘 및 최적화 (Optimization Dynamics)

역전파 역학 (Backpropagation Dynamics):

- Step 1: Critic Optimization (가치 판단 학습)
- Principle: TD(Temporal Difference) Error Minimization.
- Purpose: 현재 정책 μ 가 선택한 행동의 가치를 정확하게 평가하도록 함.
- Data Example: "왼쪽으로 가라"는 행동에 대해 실제 보상이 -10이었다면, Q함수는 해당 상태-행동의 가치를 낮추도록 파라미터를 수정함.
- Step 2: Actor Optimization (행동 교정)
- Principle: Deterministic Policy Gradient Theorem.

• Purpose: Critic이 "가치가 높다"고 판단하는 방향으로 행동 출력(μ (s))을 이동시킴.

• Data Example: Critic 네트워크를 통해 역전파된 기울기(∇_a Q)가 +방향(오른쪽)을 가리킨다면, Actor는 출력값 a를 증가시키도록 가중치를 수정함.

• Step 3: Soft Target Update (안정화)

• Principle: Moving Average (Polyak Averaging).

• Purpose: 학습 대상(Critic/Actor)이 Target을 쫓아가면 Target이 도망가는(Moving Target) 문제를 완화.

알고리즘 구현 (Pseudocode Strategy on Pythonic Logic):

```
def train_ddpg():
    # 1. Initialization
    actor, critic = Actor(), Critic()
    target_actor, target_critic = copy(actor), copy(critic)
    buffer = ReplayBuffer()
    noise = OUNoise()

    for episode in range(Max_Ep):
        s = env.reset()
        for t in range(Max_Step):
            # 2. Action with Noise
            a = actor(s) + noise.sample()
            s_next, r, done, _ = env.step(a)
            buffer.add(s, a, r, s_next)

            # 3. Update (if buffer enough)
            if len(buffer) > Batch_Size:
                batch = buffer.sample()

                # Critic Update
                target_q = batch.r + gamma * target_critic(batch.s_next, target_actor(batch.s_next))
                current_q = critic(batch.s, batch.a)
                critic_loss = MSE(target_q, current_q)
                critic_optim.step(critic_loss)

                # Actor Update
                actor_loss = -mean(critic(batch.s, actor(batch.s)))
                actor_optim.step(actor_loss)

                # Soft Update
                soft_update(target_actor, actor, tau)
                soft_update(target_critic, critic, tau)

            s = s_next
```

5. 구현 상세 및 제약 사항 (Details & Constraints)

안정화 기법 (Stabilization Techniques):

• Batch Normalization: 상태 값(State)의 스케일이 서로 다른 경우(예: 위치 vs 속도) 학습이 불안정해짐. 입력을 정규화(Whitening)하여 다양한 환경에서 동일한 하이퍼파라미터로 학습 가능하게 함.

• Ornstein-Uhlenbeck Process: 물리적 제어 문제에서는 이전 시간의 노이즈와 상관관계가 있는 노이즈(Correlated Noise)가 탐색 효율을 높여줌.

시스템 한계 (System Limitations):

• Hyperparameter Sensitivity: Learning Rate, , Noise Parameter 등에 매우 민감하여 튜닝이 까다로움.

• Overestimation Bias: Q-learning 기반이므로 Q값이 실제보다 높게 추정되는 경향이 있음 (이후 TD3 등에서 개선됨).

6. 산업 적용 전략 (Industrial Application)

비즈니스 가치 (Business Value Proposition):

• Operational Efficiency: 로봇 팔 제어, 자율 주행 차량의 조향/가속 등 연속적인 정밀 제어가 필요한 분야에 직접 적용 가능.

• Use Case:

• 화학 공정 제어: 밸브 개폐량, 온도 설정값 등의 미세 조절.

• 데이터센터 냉각: 팬 속도 및 온도를 연속적으로 조절하여 전력 효율 최적화.

7. 검증 및 누락 점검 (Validation Agent - Self-Correction)

Missing Information Check:

• 〔x〕 Core Equations: Actor Update, Critic Loss, Soft Update 모두 LaTeX로 명시됨.

• 〔x〕 Key Algorithms: Replay Buffer, Noise Process, Target Network 통합됨.

• 〔x〕 Physical Meaning: 수식별로 물리적/직관적 해석 포함됨.

Final Polish:

• 시스템 명세서(System Specification) 톤을 유지하였고, "Legacy Bottleneck"과 "Paradigm Shift"를 명확히 대비시킴.