역진자 제어

Team 강화도조약돌

목차

- 카트폴 문제 정의
- 2중/3중 카트폴의 **스윙업**
- 4중/5중 카트폴의 **밸런싱**
- 확장된 잠재 공간에서의 에이전트

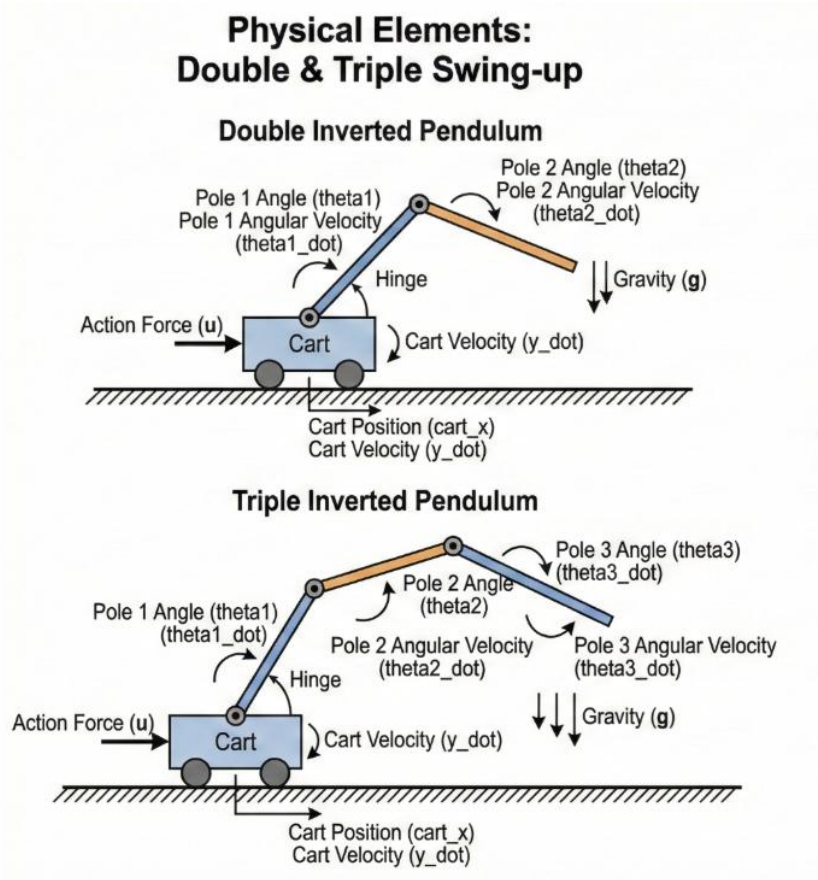
카트폴 문제 정의

N중 카트폴 문제 정의

- 카트의 위치, 위치 변화량(속도)
- 각 막대의 각도, 각도변화량(각속도)
- 막대가 N개 추가될 때 물리적인 요소도 2N개씩 추가됨.

스윙업: 아래로 늘어진 막대에 에너지를 가해 들어 올리는 것.

밸런싱: 이미 세워진 막대의 균형을 유지하는 것.



2중/3중 카트폴 **스윙업**

Actor Critic

Actor(정책 네트워크, $\boldsymbol{\pi\theta}$):

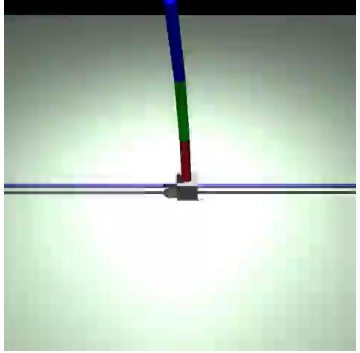
- 역할: 에이전트가 어떤 행동(Action)을 할지 결정하는 정책($\boldsymbol{\pi}$)를 담당.
- 입력: 현재 상태(s)(ex. 진자의 각도, 속도 등).
- 출력: 행동을 선택할 확률 $\boldsymbol{\pi\theta}$(a|s)

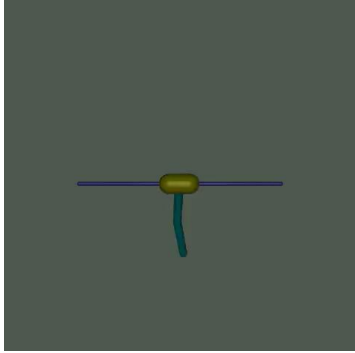
Critic(밸류 네트워크, Q or V):

- 역할: Actor가 선택한 행동이 얼마나 좋은지 평가하는 '가치 함수'를 담당.
- 입력: 현재 상태(s)와 행동(a).
- 출력: 그 행동의 가치(Q or V)

[넘어짐 → 보상 감소 → 예측 수정 → 행동 교정] 과정을 통해 에이전트가 제어 감각을 익힘.

Soft Actor Critic, SAC





일반 AC에 엔트로피라는 개념을 더함.

일반 Actor: 점수 높은 행동 하나만 추구하여 지역 최적해에 빠지기 쉬움.(실제로 막대기를 빙빙돌리며 점수를 얻음.)

SAC Actor: 점수 높은 행동을 하되, 가능한 다양한 시도를 해보자.

목표 함수 자체에 엔트로피가 포함되어 있어, 에이전트의 새로운 시도 자체를 보상으로 느낌.

→ 삼중 스윙업처럼 해답을 찾기 어려운 복잡한 물리 제어 문제에서, 에이전트가 일부러 불확실한 행동을 유지하려고 하므로, 더 나은 방법을 찾아냄.

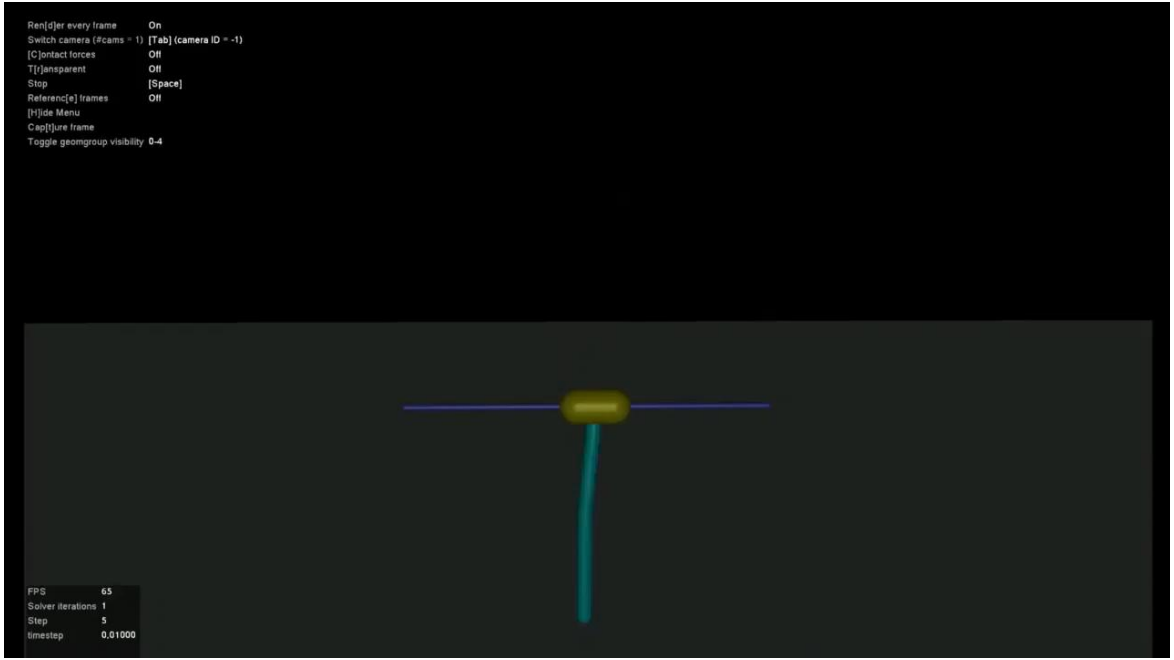
곱셈형 보상 함수

덧셈형 보상과 달리, 특정 조건을 무시하고 나머지만 잘해서 점수를 챙기는 꼼수(지역 최적해)를 차단.

보상 = 카트 위치 x 각도(수직 유지) x 각속도(제동) x 에너지(효율)

$$Reward = r_{pos} \times r_{angle} \times r_{vel} \times r_{action}$$

2중 스윙업

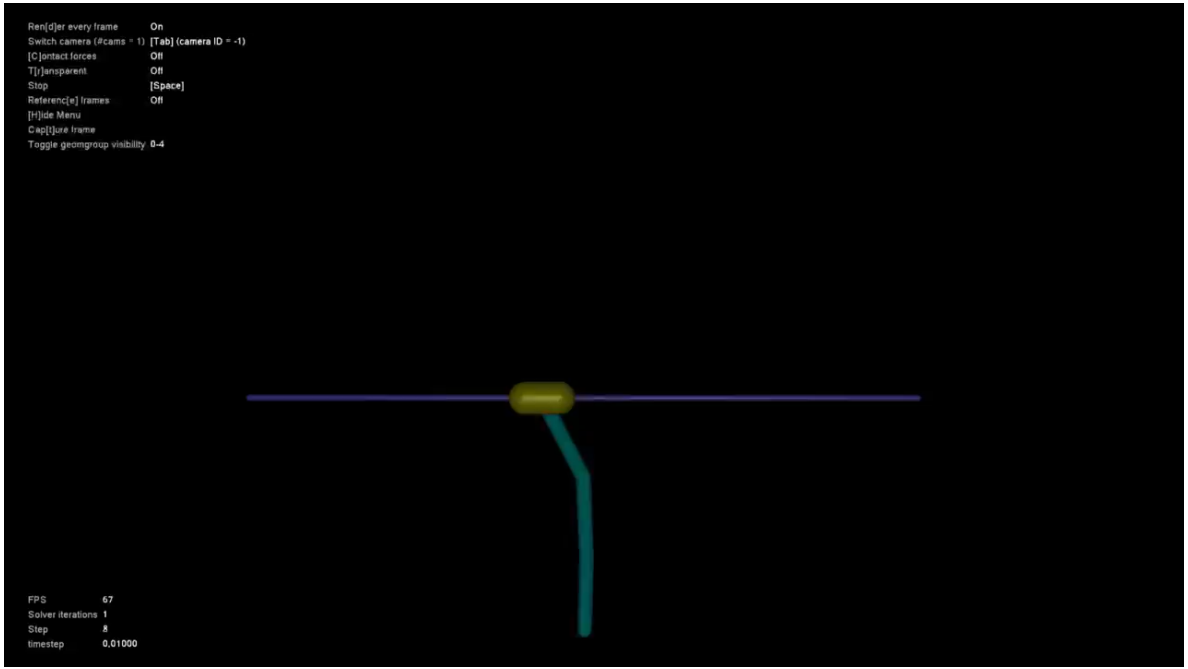

Ren[d]er every frame On
Switch camera (#cams = 1) [Tab] (camera ID = -1)
[C]ontact forces Off
T[r]ansparent Off
Stop [Space]
Referenc[e] frames Off
[H]ide Menu
Cap[t]ure frame
Toggle geomgroup visibility 0-4
FPS 65
Solver iterations 1
Step 5
timestep 0.01000

문제 확장(2중 →3중)

● 2중 스윙업	● **구분**	● 3중 스윙업
● Gymnasium 라이브러리에 내장된 환경 로드	● **환경**	● 내장 환경이 없기에 MuJoCo XML 파일을 직접 로드하여 새 환경 정의.
● [256,256]	● **신경망 구조**	● [512,512]
● 카트 위치 패널티(1.0:엄격), 성공 보상(+2.0)	● **보상함수**	● 카트 위치 패널티(0.5:완화, 카트를 더 멀리 움직여야 하므로), 3번째 막대 관련 항 추가. 성공 보상(+5.0)
● 50만 step(약 20만step에 수렴)	● **학습량 증가**	● 300만 step

1) **네트워크 용량 확장**
2) **활동 반경 보장**
3) **학습 시간 확보**

3중 스윙업


Ren[d]er every frame On
Switch camera (#cams = 1) [Tab] (camera ID = -1)
[C]ontact forces Off
T[r]ansparent Off
Stop [Space]
Referenc[e] frames Off
[H]ide Menu
Cap[t]ure frame
Toggle geomgroup visibility 0-4
FPS 67
Solver iterations 1
Step 8
timestep 0,01000

Actor-Critic과 GAN의 유사성

Actor와 Critic이 함께 학습하는 과정이 GAN(Generative Adversarial Nets)과 구조적으로 유사해보임.

차이: 경쟁 vs 협력

GAN의 생성자는 판별자를 속이는 방향으로 가짜를 진짜처럼 생성, 판별자는 생성자가 만든 가짜를 잡아내려고 함. 서로 **경쟁**하며 두 신경망이 학습됨.(위조지폐범과 경찰)

반면, Critic은 Actor의 행동에 대한 피드백을 주며 Actor가 더 나은 보상을 받을 수 있도록 가이드해주는 **협력**적인 관계임.(축구 선수와 코치)

4중/5중 카트폴 **밸런싱**

PPO(Proximal Policy Optimization)

비율 계산: 현재 정책이 과거 정책에 비해 특정 행동을 할 확률이 얼마나 변했는지 계산.

제한(Clipping): 이 변화율이 정해진 범위를 벗어나면, 그 이상의 업데이트는 무시.

효과: 학습 과정에서 에이전트가 너무 큰 모험을 하지 않게 되어, 학습이 안정적임.

왜 밸런싱은 왜 PPO를 사용했는가?

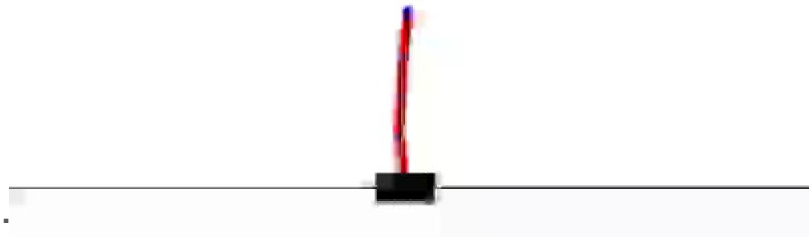
2중, 3중 스윙업에서는 에이전트가 넘어진 상태에서 일어나는 법을 적극적으로 탐험.(global optima를 찾기 어렵기 때문)

막대를 추가할수록 자유도가 높아져 스윙업의 전역 최적해 탐색이 어려움. → 문제를 스윙업에서 **밸런싱**으로 **재정의**.

밸런싱은 이미 서 있는 상태를 유지하는 것이라, 급격한 행동 변화보다는 정교한 제어가 중요

→ 현재 정책에서 크게 벗어나지 않으면서 안정적으로 미세 조정이 가능한 PPO 채택.

4중 밸런싱



슬로우 모션처럼 보이는 이유:

- 코드 내 time step τ를 일반 설정보다 작게 잡음.

막대가 한쪽으로 쓰러지는게 아닌 무너지는 것처럼 보이는 이유:

- 비용함수를 막대를 세우게 하는게 아닌 전체 시스템의 에너지 비용을 낮추게 목표를 설정함.
- 에이전트가 모든 막대를 1자로 세우는 어려운 방법보다 막대들이 서로 지그재그로 굽어지게 만들어 서로의 무게 중심을 상쇄시키는 전략을 취함.

문제 확장(4중 →5중)

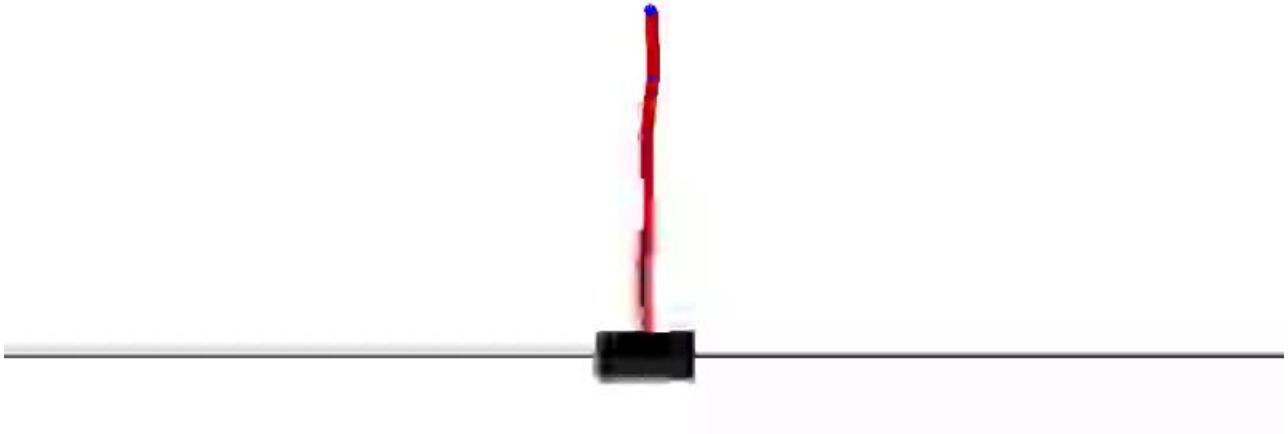
- 4중 밸런싱
- 카트, 4개 막대
- 카트 위치 패널티 엄격
- 300만 step

- **구분**
- **관측공간**
- **보상 함수**
- **학습량**

- 5중 밸런싱
- 카트, 5개 막대
- 카트 위치 패널티 완화
- 500만 step

1) **입력 차원 확장**
2) **더 넓은 카트 활동 반경**
3) **학습 시간 증가**

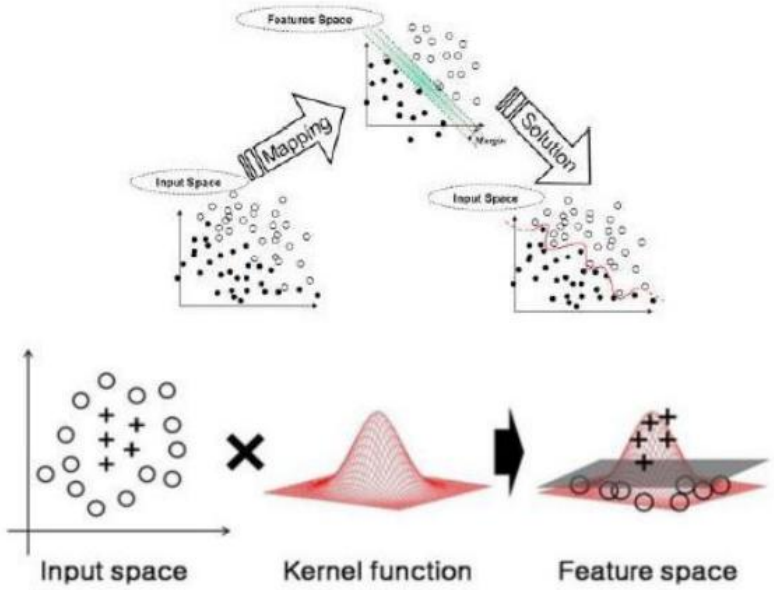
5중 밸런싱



“N중 역진자 문제처럼 변수가 많아지고 복잡한 비선형 문제를 어떻게 해결할까?”

커널 SVM

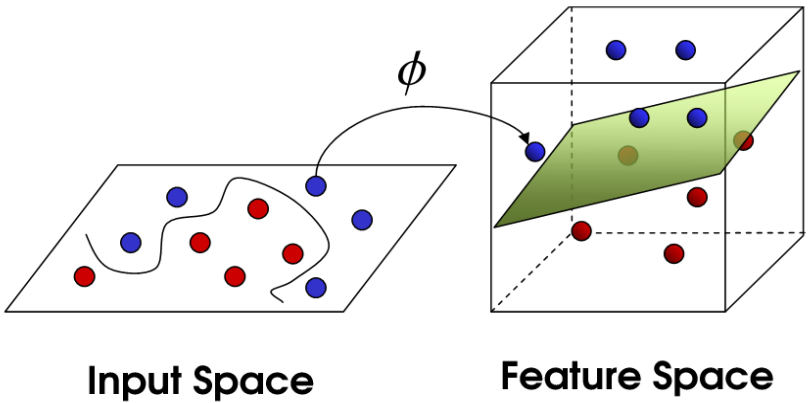
데이터가 저차원에서 선형적으로 분리가 되지 않는 경우 고차원으로 매핑하는 커널 함수를 적용시켜 직선(초평면)으로 분리할 수 있음.



커버의 정리(Cover's theorem)

정의: 선형적으로 분리할 수 없는 훈련 데이터 세트로 주어졌을 때, 비선형 변환을 통해 더 높은 차원의 공간으로 투영함으로써 높은 확률로 선형적으로 분리 가능한 훈련 세트로 변환할 수 있음.

다시 말해, 저차원 공간의 비선형적인 패턴을 고차원 공간으로 매핑하면 선형적으로 분리될 확률이 높아짐.



Overcomplete Representation

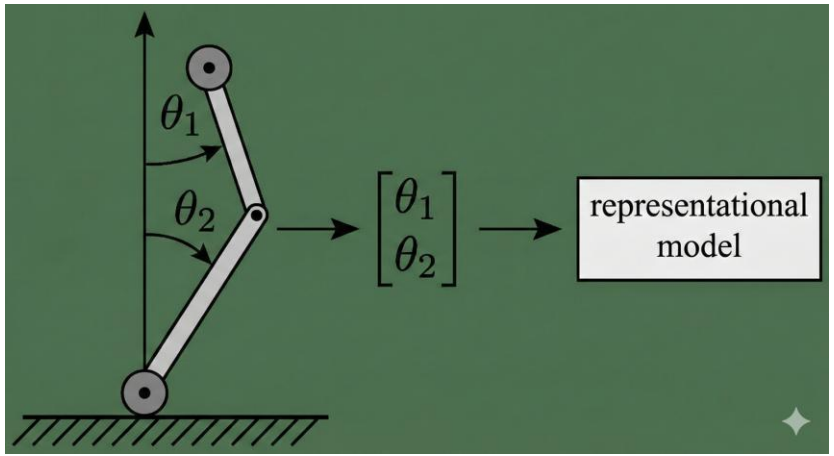
입력보다 더 큰 차원으로 정보를 확산시켜 특징(Feature)을 풍부하게 만듦.

→ “Actor가 더 쉽게 최적해를 찾도록 유도해보자”

이중 역진자에서의 overcomplete

이중 역진자의 상태가 두 막대의 각도 [θ1, θ2] 2개뿐이라고 가정.

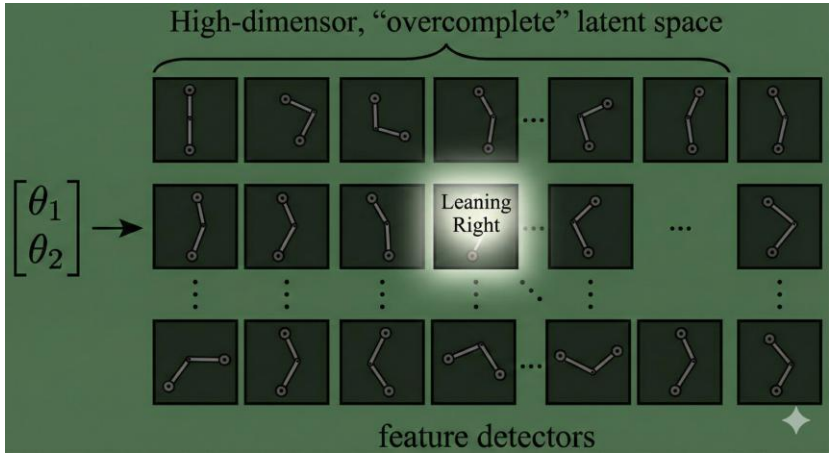
입력 공간(2D): 에이전트는 이 두개의 숫자만 보고 현재 상태를 파악한 후 행동을 해야 함.



이중 역진자에서의 overcomplete

두 각도값을 100차원의 Overcomplete 잠재 공간으로 보냈을때, 해당 공간에서의 에이전트는 수많은 상태 중 현재 상태에 대응되는 축만 바라봄.

에이전트는 복잡한 계산 없이 “아, ‘오른쪽 기울어짐’상태니까 왼쪽으로 밀어야겠다.”라고 매우 직관적으로 판단할 수 있음을 기대해봄직함.



차원을 확장할 때 고려해야 할 요소

1. 차원을 얼마나 확장시킬것인가?
2. 실제 공간의 미세한 변화가 확장된 잠재 공간에서도 미세한 변화로 이어져야 안정적인 제어가 가능함.(립시츠 연속성이 보장되어야 함.)

→ 잠재 공간을 연속적인 확률 분포로 강제하는 생성모델이 필요.

더 고민해보기.