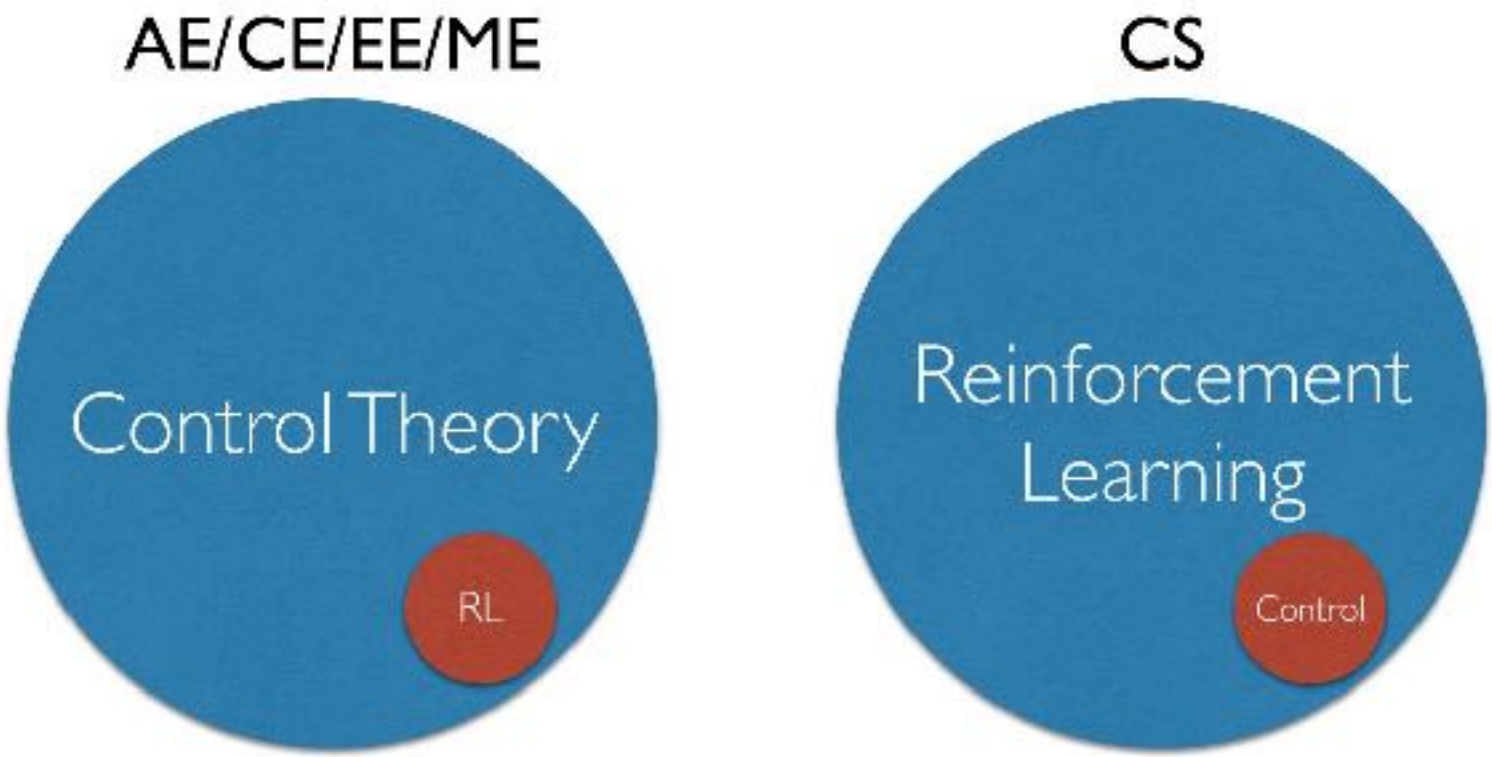
AI + 제어

2019. 01. 23

김성필 I ㈜제이마플

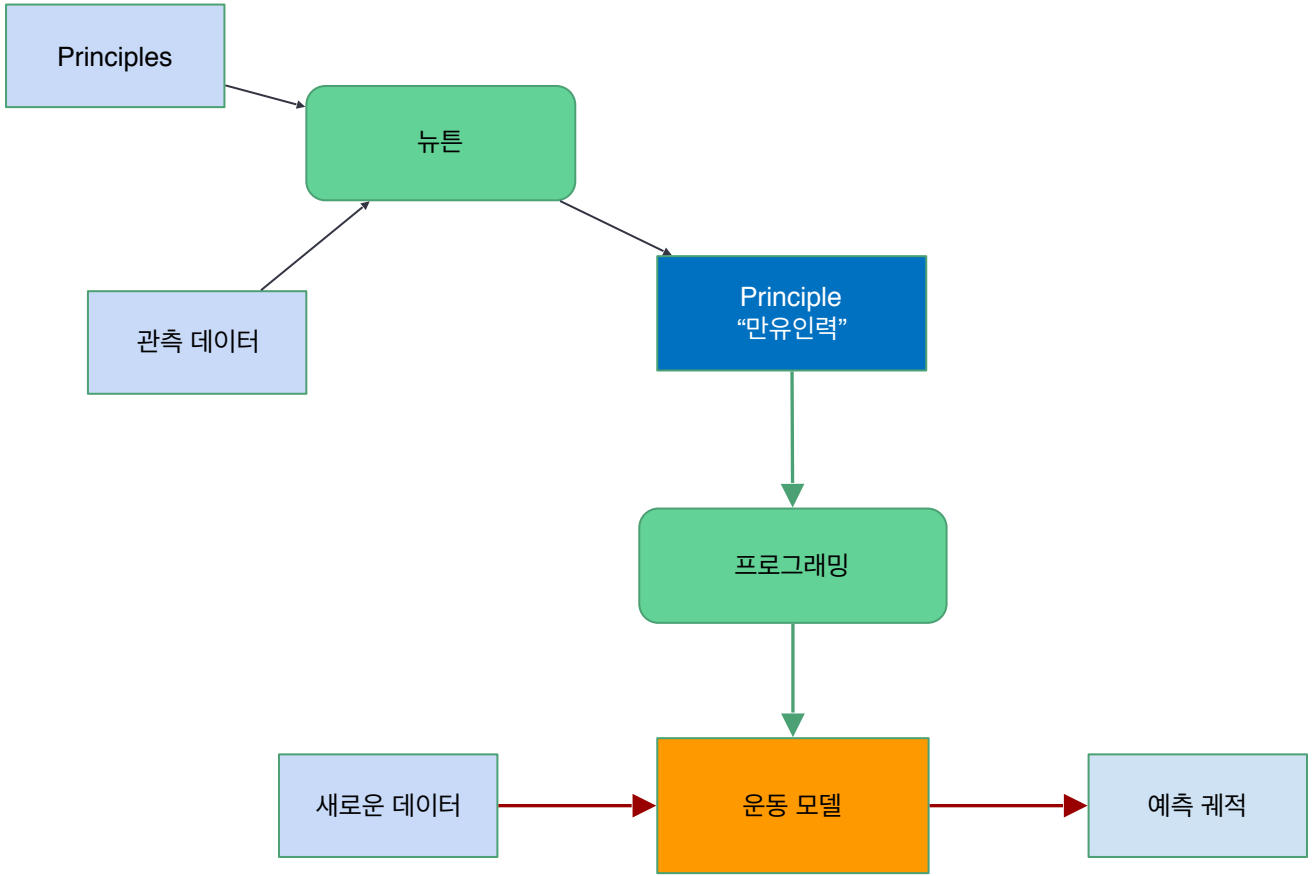

AE/CE/EE/ME
Control Theory
RL
CS
Reinforcement Learning
Control

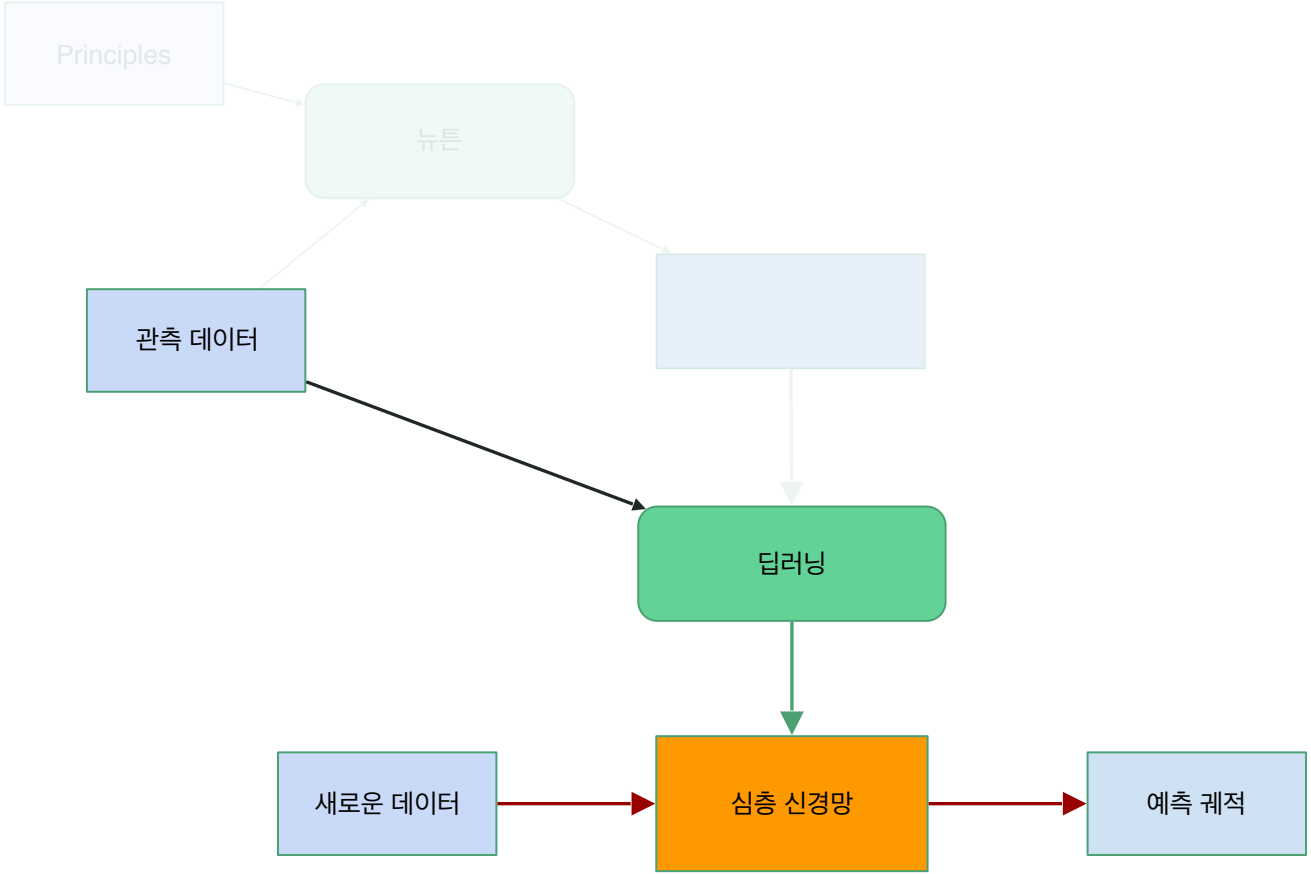
차례

딥러닝을 바라보는 관점

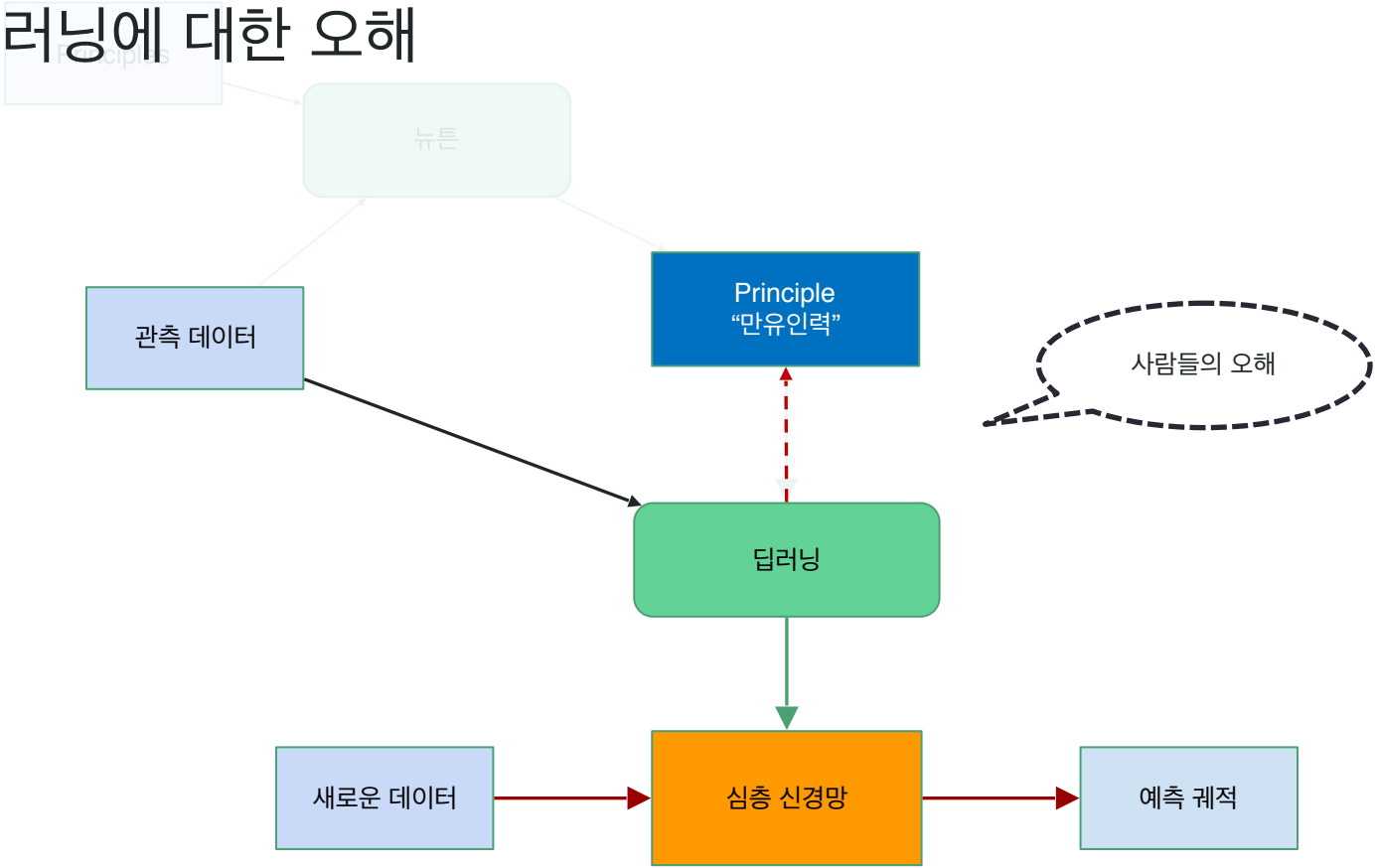
연역적 vs 귀납적 개발

연역적 개발	귀납적 개발
- 원리를 기반으로 개발자가 직접 구현 - 컴퓨터와 역사를 같이 해 익숙함 - 거의 SW 개발 전 분야에 적용 - 구조적 프로그래밍, OOP 등	- 데이터를 기반으로 간접적으로 구현 - 최근에 등장해 익숙하지 않음 - 연역적 개발이 불가능한 분야에서 각광 - 머신러닝, 딥러닝 등


Principles
뉴튼
관측 데이터
Principle
“만유인력”
프로그래밍
새로운 데이터
운동 모델
예측 궤적


Principles
뉴튼
관측 데이터
딥러닝
새로운 데이터
심층 신경망
예측 궤적

딥러닝에 대한 오해



딥러닝에 대한 올바른 인식

귀납적 개발의 일종

- ‘지능’ 아님. 신경망의 구조 등 핵심은 사람이 설계
- 전자 계산기, 컴파일러처럼 곧 익숙해질 것

연역적 개발과 상호보완

- 서로 상충하지 않고, 연역적 개발로 성과를 내지 못한 다양한 분야로 확산될 것
- 그렇다고 연역적 개발을 대체하는 것은 아님. 적용 분야가 서로 다름

기능	연역적 개발	귀납적 개발
더하기 함수	function add(x, y) return x+y end	?
얼굴인식 함수	?	딥러닝

딥러닝은

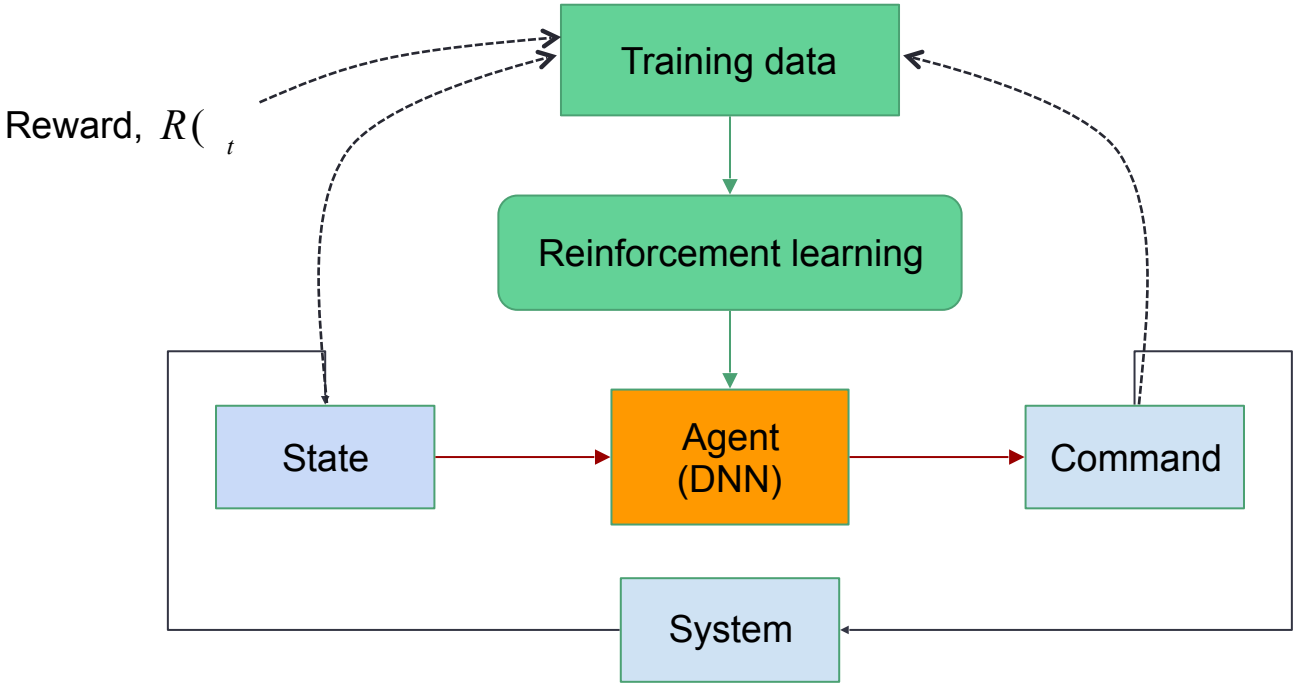
인공지능이라 쓰고

귀납적 개발이라 읽는다

RL을 바라보는 관점

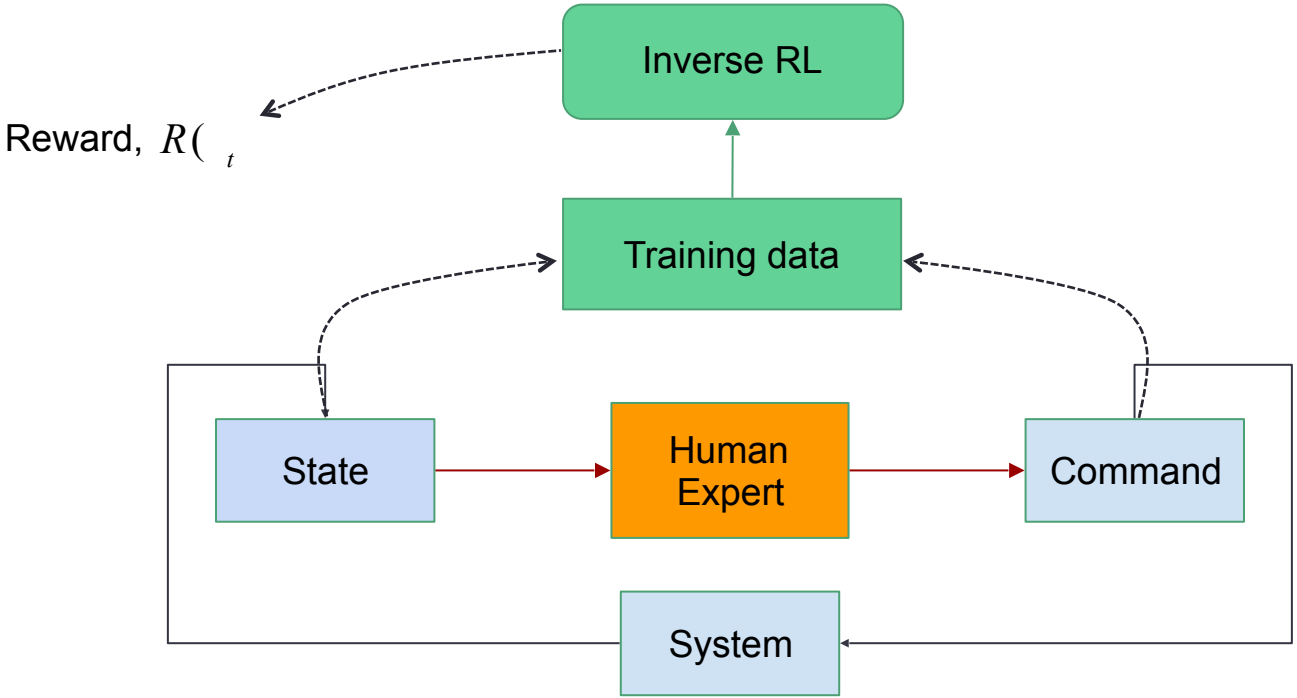
Reinforcement Learning

$$x_1, u_1, r_1, x_2, u_2, r_2, \cdots, x_t, u_t, r_t$$



Inverse Reinforcement Learning

목표: Reward function



강화학습은 Reward로 정답을 만들어내는 기법

Agent 모델은 Supervised Learning으로 학습

설계자의 의도는 Reward를 통해 반영

제어기 개발의 진짜 문제

제어기 설계 ~ 시스템 모델의 정확성

제어기 구현 ~ 센서의 정확성

모델링의 정확성

모델링 오차

- 제어기 설계 대상(수학 모델)과 적용 대상(실제 시스템)의 차이

모델링 오차의 영향

- 적용(시험) → 제어 성능 저하 → 제어기 튜닝 → 적용 반복: 많은 시간과 비용 소모

불일치

모델링 오차 대처 방안

정밀 모델 개발

- 해석적 방법, 축소 모델 시험 등 모든 방법을 동원
- (돈 많은 사람들)

제어 기법

- Robust control: 모델에 오차를 명시적으로 고려해서 제어기 설계
- Adaptive control: 모델링 오차에 따라 제어기를 적절히 바꿔주는 알고리즘 추가
- (머리 좋은 사람들)

시행착오법

- 실제 시스템의 반응을 보고 제어기를 튜닝하거나 재설계
- (가난한 사람들)

현실은…

정밀 모델 개발

- 그래도 잘 안 맞음 ➔ 시험을 통한 시행착오법 병행

제어 기법

- 실제 현장에서는 전공자들도 안 씀(손재기)

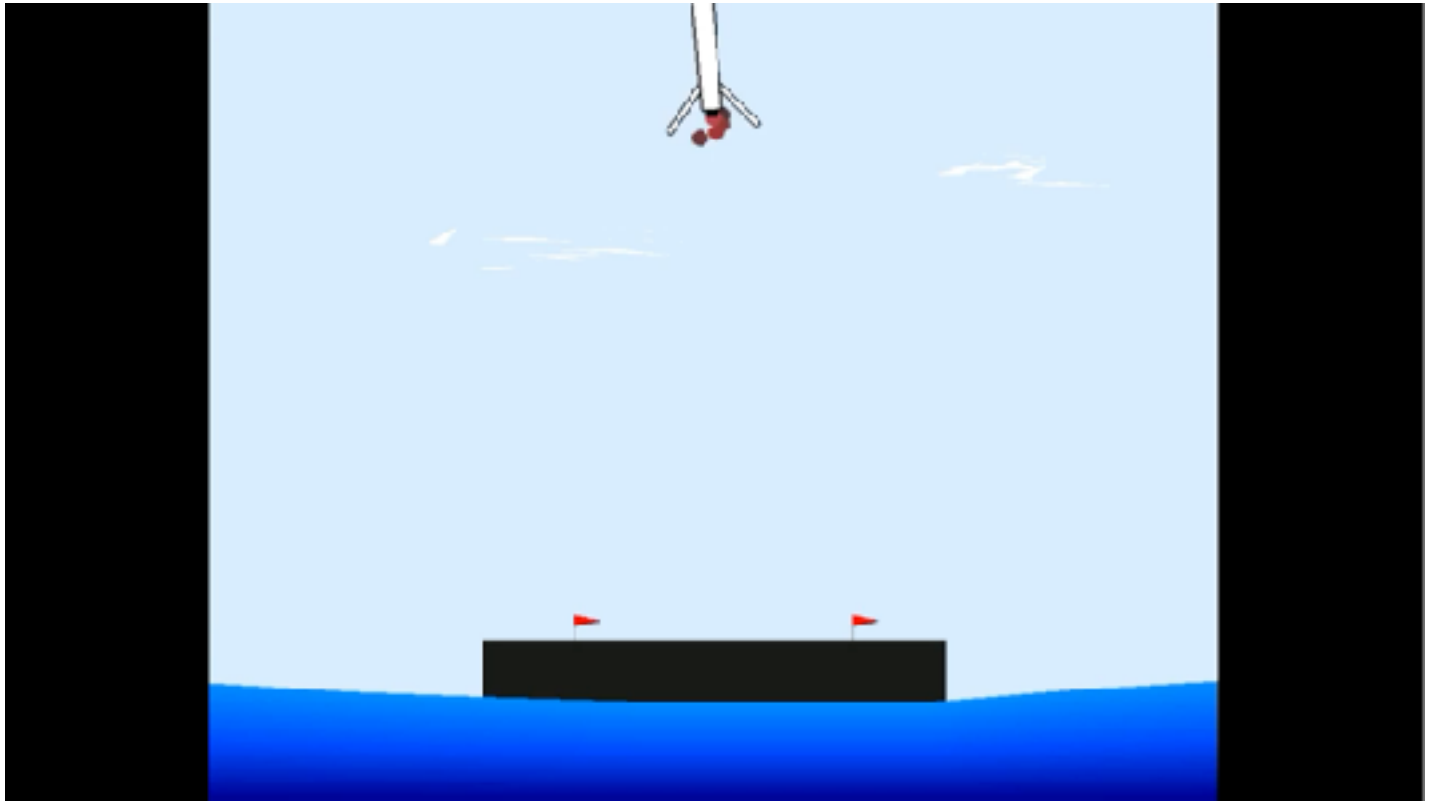
결국 시행착오법이 유일한 방법

- 실제 시스템의 반복 시험이 가능하고 싸야 함. 참고/ 바둑 vs 항공기
- 제어기의 튜닝이 쉽고 간단해야 함

☞ 모델링 오차에 효과적으로 대처하려면, 튜닝이 쉬운 제어기 구조 필요!

귀납적 개발의 효과적 활용

실제 시스템에 적용한다면?



RL 제어기

대부분 Model-free

- 게임/시뮬레이션 문제 ➔ 수학적 모델링 불가능, 시스템 = 시뮬레이션, 무한 반복 시험 가능
- 실제 시스템의 데이터를 모아 귀납적 구현 ➔ “로켓 10,000개만 만들어주세요”

설계자 의도는 Reward로만 반영 가능

- 다양한 성능 요구조건을 반영하려면? ➔ 착지 가속도, 측속도, 착륙 반경 등
- 성능 지표 변경에 따른 재설계가 간편한가?

평가 및 인증

- 신경망으로 구현된 제어기의 분석/평가 방법? ➔ 물리적 의미가 없는 파라미터들

☞ 기존 제어기의 핵심 문제는 해결 못함. 기존 제어기의 대안X

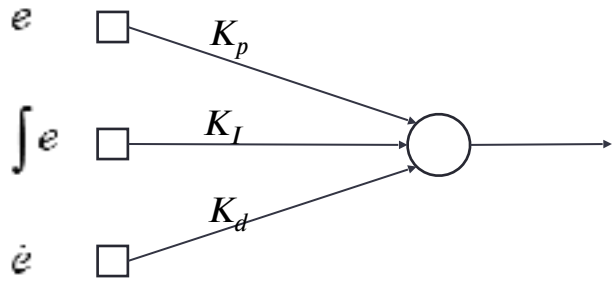
PID 제어기

사실상의 산업 표준

- 선형/비선형 시스템에 모두 적용 가능
- 모델/비모델 모두 설계 가능

단층 신경망과 유사

- 입력 = 3
- 출력 = 1



핵심 장점

- 물리적 의미가 있는 3개의 파라미터로 구성 ➔ 시행착오 튜닝에 적합
- 계층적 구성이 가능해 복잡한 제어 구조 가능 ➔ 설계자의 전문지식 반영 용이

☞ RL 보고 있나?

상보적 활용 방안

시스템을 잘 알고 경험이 많을수록 제어기 튜닝을 잘 함

숙련된 조종사는 새로운 항공기도 빠르게 적응해 비행을 잘 함

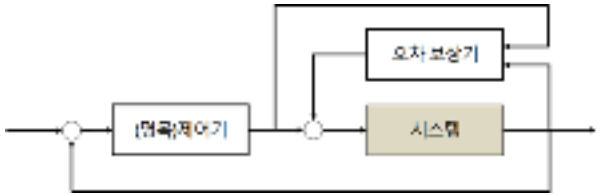
- 결국 사람은 '기본 원리 + 귀납적 모델링 오차 보상'을 혼용?

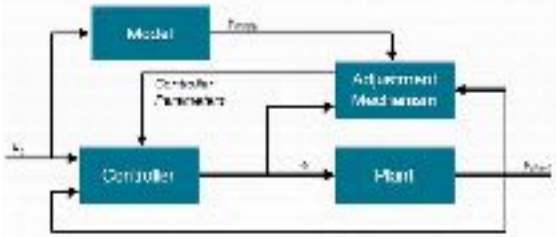
그렇다면, PID + 귀납적 기법은 어떨까?

- #1: PID 제어기의 출력 + 귀납 제어기의 보정 출력
- #2: PID 제어기의 파라미터를 귀납적 기법으로 튜닝
- 두 구조 모두 연역적 기법은 이미 존재

  ➔ 귀납적 기법이 더 적합하지 않을까?

  (Robust/Adaptive 제어의 실패는 연역적 접근 탓?)



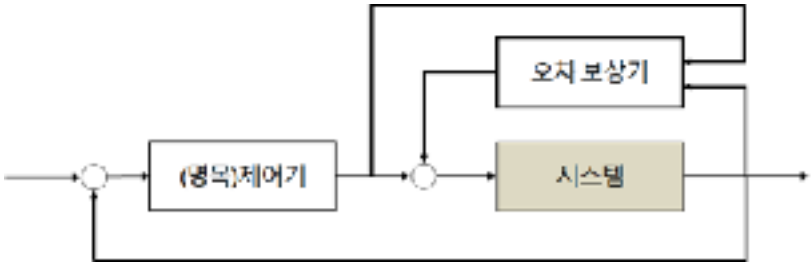


☞ 사람이 하던 귀납적 역할을 AI로 구현하자!

J.MARPLE

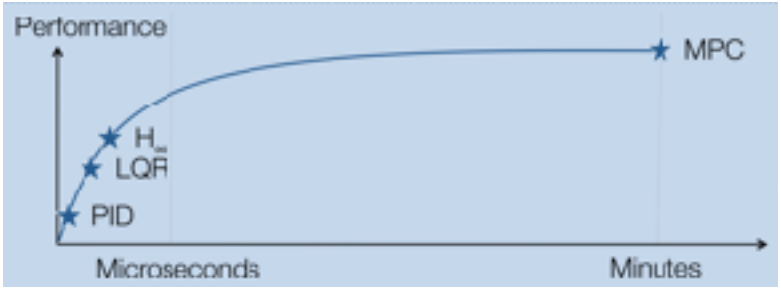
지능 제어

"귀납적 기법을 적용한 제어기"



#1. 모델링 오차와 외란에 대처

- 기존 제어기 + (연역적)오차 보상기
- 기존 제어기 + (귀납적)오차 보상기



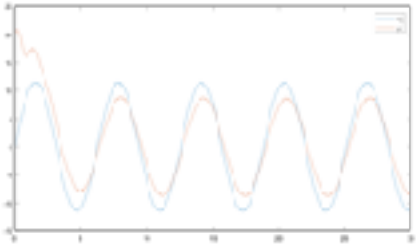
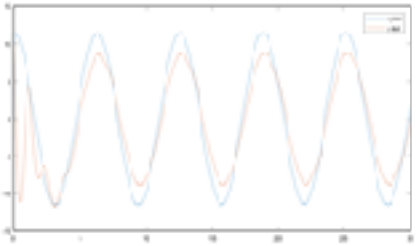
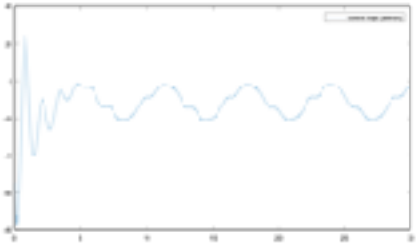
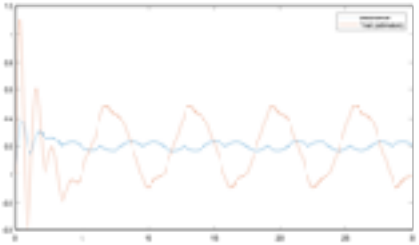
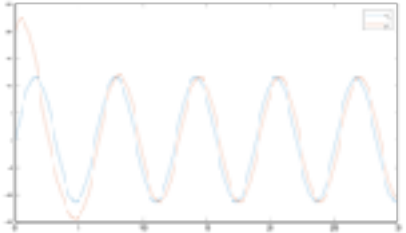
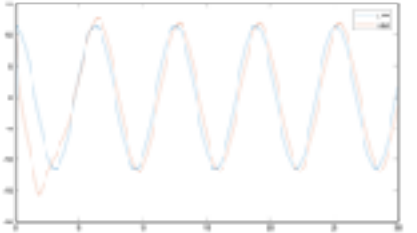
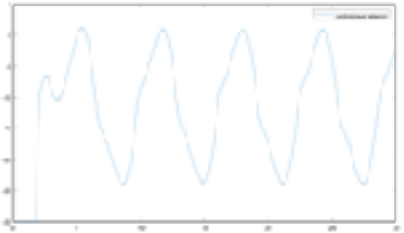
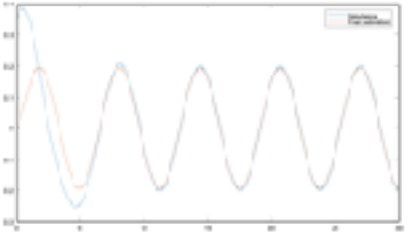
#2. 귀납적 MPC ➔ PID를 극복해보자!

- PID와 MPC는 현장에서 먼저 개발, 나중에 학계에서 연구 ➔ 잘되는 구조라는 의미!
- MPC의 핵심 전제 = 정밀 모델 ➔ 비선형 모델이면 실시간 구현 어려움
- 귀납적 모델 기반 실시간 MPC ➔ 비선형 MPC의 한계를 극복
- MPC + 오차 보상기 ➔ 오차에 능동적 대처

내풍 제어(연역적 오차 보상기)

Auto Landing on Moving Target

MPC + 오차 보상기

소개

- 약력
  - 서울대학교 항공우주공학과 졸업
  - 서울대학교 항공우주공학과 대학원(박사)
  - 한국항공우주연구원 선임연구원
  - 보건복지부 연구관(국립재활원)
  - ㈜제이마플 대표

- 저서
  - 칼만필터의 이해(2010)
  - Rigid Body Dynamics for Beginners(2013)
  - 딥러닝 첫걸음(2016)