# 2025.01.14.수

| | |
|---|---|
| 생성일 | @2026년 1월 14일 오전 9:18 |
| 태그 | |

## 목차



---

## 인공지능 기초

- 인공지능 > 머신러닝 > 딥러닝
  - 인공지능은 가장 포괄적인 개념으로, 인간의 지능을 모방하는 모든 시스템을 의미합니다. 여기에는 규칙 기반 전문가 시스템(Expert System), 탐색 알고리즘, 논리 추론 등이 포함됩니다.
  - 머신러닝은 인공지능의 부분집합으로, 명시적 프로그래밍 없이 데이터로부터 패턴을 학습하는 방법론입니다. 통계적 학습 이론에 기반하며, 경험(데이터)을 통해 성능을 개선하는 알고리즘을 다룹니다. 전통적인 ML에는 결정 트리, SVM, 랜덤 포레스트, k-NN 등이 있습니다.
  - 딥러닝은 머신러닝의 부분집합으로, 다층 신경망(Deep Neural Networks)을 사용한 학습 방법입니다. 특징 추출(feature extraction)을 자동화하며, 계층적 표현 학습(hierarchical representation learning)을 통해 raw data로부터 고수준 추상화를 학습합니다. CNN, RNN, Transformer 등이 대표적입니다.
- 판단 모델 vs 생성모델
  - *판별 모델(Discriminative Model)은 조건부 확률 $P(Y|X)$를 직접 학습합니다. 입력 X가 주어졌을 때 레이블 Y를 예측하는 결정 경계(decision boundary)를 학습하며, 분류나 회귀 작업에 최적화되어 있습니다. 예시로는 로지스틱 회귀, SVM, 전통적인 CNN 분류기 등이 있습니다. 계산 효율이 높고 예측 성능이 우수하지만, 데이터 생성이나 불확실성 추정에는 제한적입니다.
  - 생성 모델(Generative Model)은 결합 확률 $P(X,Y)$ 또는 $P(X)$를 학습합니다. 데이터의 분포 자체를 모델링하여 새로운 샘플을 생성할 수 있습니다. Naive Bayes, HMM, VAE, GAN, Diffusion Model, GPT 같은 autoregressive 모델이 대표적입니다. 베이즈 정리를 통해 $P(Y|X) = P(X|Y)P(Y)/P(X)$로 판별도 가능하며, missing data 처리, 이상치 탐지, 데이터 증강 등 다양한 응용이 가능합니다.
- 지도 학습 vs 비지도 학습
  - 레이블된 데이터를 활용한 분류와 회귀 모델이 주류입니다.
  - 레이블 없는 데이터를 활용하여 내재된 구조나 패턴을 발견하는 것이 목표
- 강화학습
  - 강화학습은 에이전트가 환경과 상호작용하며 보상을 최대화하는 정책(policy)을 학습하는 패러다임입니다. Markov Decision Process (MDP) 프레임워크를 기반으로 하며, 상태(state) S, 행동(action) A, 보상(reward) R, 전이 확률 $P(s'|s,a)$, 할인 인자 $\gamma$로 정의됩니다.
  - 목표는 기대 누적 보상 $E[\Sigma_t \gamma^t R_t]$를 최대화하는 최적 정책 $\pi^*$를 찾는 것입니다. 가치 기반 방법(Q-learning, DQN)은 action-value function $Q(s,a)$를 학습하고, 정책 기반 방법(Policy Gradient, PPO)은 정책을 직접 최적화합니다. Actor-Critic은 두 접근을 결합합니다.
  - 탐색-활용 딜레마(exploration-exploitation trade-off)가 핵심 과제이며, ε-greedy, UCB, Thompson sampling 등의 전략을 사용합니다. 게임 AI, 로보틱스, 자율주행, 추천 시스템 등에 응용됩니다.
- AI 에이전트 아키텍처
  - 프롬프트 체이닝

    복잡한 태스크를 여러 단계의 프롬프트로 분해하여 순차적으로 실행하는 기법입니다. 각 단계의 출력이 다음 단계의 입력이 되며, Chain-of-Thought (CoT)의 확장 개념입니다. 예를 들어, "분석 → 요약 → 번역 → 검증" 파이프라인을 구성할 수 있습니다. 중간 결과를 명시적으로 관리하여 디버깅이 용이하고, 각 단계를 독립적으로 최적화할 수 있습니다.
  - 라우팅

    입력의 특성에 따라 적절한 모델이나 전문가 시스템으로 요청을 분배하는 메커니즘입니다. 분류기(classifier)가 쿼리의 도메인, 난이도, 언어 등을 판단하여 최적의 처리 경로를 선택합니다. Mixture of Experts (MoE)에서 gating network가 이 역할을 수행하며, 계산 효율성과 전문화를 동시에 달성합니다.
  - 병렬화

    독립적인 서브태스크를 동시에 실행하여 처리 속도를 향상시킵니다. MapReduce 패턴처럼 작업을 분할(map)하고 결과를 집계(reduce)합니다. 예를 들어, 문서의 각 섹션을 병렬로 요약한 후 통합 요약을 생성할 수 있습니다. GPU의 데이터 병렬성(data parallelism)이나 모델 병렬성(model parallelism)도 이 개념의 확장입니다.

    - 중앙조율자 (Orchestrator)

      전체 워크플로우를 관리하고 조정하는 메타 컨트롤러입니다. 서브에이전트들의 실행을 계획하고, 리소스를 할당하며, 에러를 처리합니다. 태스크 의존성 그래프(DAG)를 관리하고, 동적으로 실행 계획을 수정할 수 있습니다. LangChain의 AgentExecutor나 AutoGPT의 플래닝 모듈이 이에 해당합니다.

    - 평가자, 최적화자

      평가자는 생성된 출력의 품질을 평가하고, 최적화자는 피드백을 바탕으로 개선합니다. Constitutional AI의 self-critique 메커니즘이나 ReAct (Reasoning + Acting) 패턴에서 사용됩니다. 평가 지표는 도메인별로 다르며 (정확성, 일관성, 안전성 등), 강화학습의 reward model과도 연결됩니다.

- Sematic vs Syntactic
    - 의미론(Semantic)은 기호가 지시하는 의미와 개념을 다룹니다. 참/거짓 판단, 의미적 유사성, 함의 관계 등을 포함합니다. "bank"가 "강둑"인지 "은행"인지는 맥락(context)에 따라 결정됩니다. Word2Vec, BERT 같은 임베딩 모델은 단어의 의미적 표현을 학습하며, distributional semantics ("You shall know a word by the company it keeps")에 기반합니다.
    - 구문론(Syntactic)은 기호의 형식적 구조와 규칙을 다룹니다. 문법적 올바름, 토큰 순서, 파싱 트리 등이 관심사입니다. 컴파일러나 formal language theory가 대표적입니다. 기계는 구문 규칙을 정확히 따르지만, "의미"를 이해하지는 못합니다.
- 키워드 기반 검색 vs 의미 유사도 검색
    - 키워드 기반 검색은 정확한 단어 일치를 찾습니다. 빠르고 해석 가능하지만 동의어, 다의어, 의역을 처리하지 못합니다.
    - 의미 유사도 검색은 임베딩 공간에서 벡터 유사도를 계산합니다. 쿼리와 문서를 인코딩하고 순위를 매깁니다. 의미적으로 유사한 단어가 다른 문서도 검색합니다.
- 임베딩
    - 임베딩은 이산적이고 고차원인 데이터(단어, 문장, 이미지)를 연속적인 저차원 벡터 공간으로 매핑하는 것입니다.
    - **목적**: 의미적으로 유사한 항목들이 벡터 공간에서 가까이 위치하도록 합니다 (geometric property). 벡터 연산으로 의미 관계를 표현할 수 있습니다 (예: king - man + woman ≈ queen).
    - **학습 방법**: Word2Vec (Skip-gram, CBOW)은 sliding window 내 단어 동시 출현을 예측하며, negative sampling으로 효율화합니다. GloVe는 전역 단어 동시 출현 행렬을 분해합니다. BERT는 masked language modeling으로 양방향 문맥을 학습하며, Sentence Transformers는 siamese network로 문장 임베딩을 학습합니다.
    - **특성**: 차원 d는 보통 128-1024이며, 압축과 일반화의 균형을 맞춥니다. 임베딩은 전이 학습(transfer learning)의 핵심이며, 다운스트림 태스크의 입력 표현으로 사용됩니다.
- 백터 데이터 베이스
    - 벡터 DB는 고차원 임베딩 벡터의 효율적인 저장과 유사도 검색을 위해 최적화된 데이터베이스입니다. 전통적 DB가 정확한 일치(exact match)를 위한 B-tree를 사용한다면, 벡터 DB는 근사 최근접 이웃 탐색(ANN, Approximate Nearest Neighbor)을 위한 인덱스를 사용합니다.
- RAG
    - RAG는 외부 지식 베이스에서 관련 정보를 검색(retrieve)하여 생성 모델의 입력에 추가함으로써 응답을 증강(augment)하는 아키텍처입니다.
- LLM 만드는 과정
    - 데이터 수집
    - 전처리 / 정제 (토큰화Tokenization)
    - 패턴 학습 (Pretraining → base model)
    - 파라미터 최적화 (RLHF → instruct model)
    - 파인튜닝
        - Full Fine-tuning: 모든 파라미터 업데이트, 고비용이지만 최고 성능
        - PEFT (Parameter-Efficient Fine-Tuning):
            - LoRA (Low-Rank Adaptation): 가중치 업데이트를 저랭크 행렬로 근사, $\Delta W = AB$ ($A \in \mathbb{R}^{d \times r}$, $B \in \mathbb{R}^{r \times k}$, $r \ll d$)
            - Prefix Tuning: 입력에 학습 가능한 prefix 추가
            - Adapter Layers: 작은 병목 레이어 삽입
- 자기회귀 모델
    - 자기회귀 모델은 시퀀스의 다음 요소를 이전 요소들의 조건부로 예측합니다

---

## 실습

https://www.kaggle.com/datasets/tongpython/cat-and-dog?resource=download

https://teachablemachine.withgoogle.com/train/image