1장. 머신러닝과 딥러닝

1.1 인공지능, 머신러닝과 딥러닝

인공지능: 인간의 지능을 모방하여 사람이 하는 일을 기계가 할 수 있도록 하는 기술

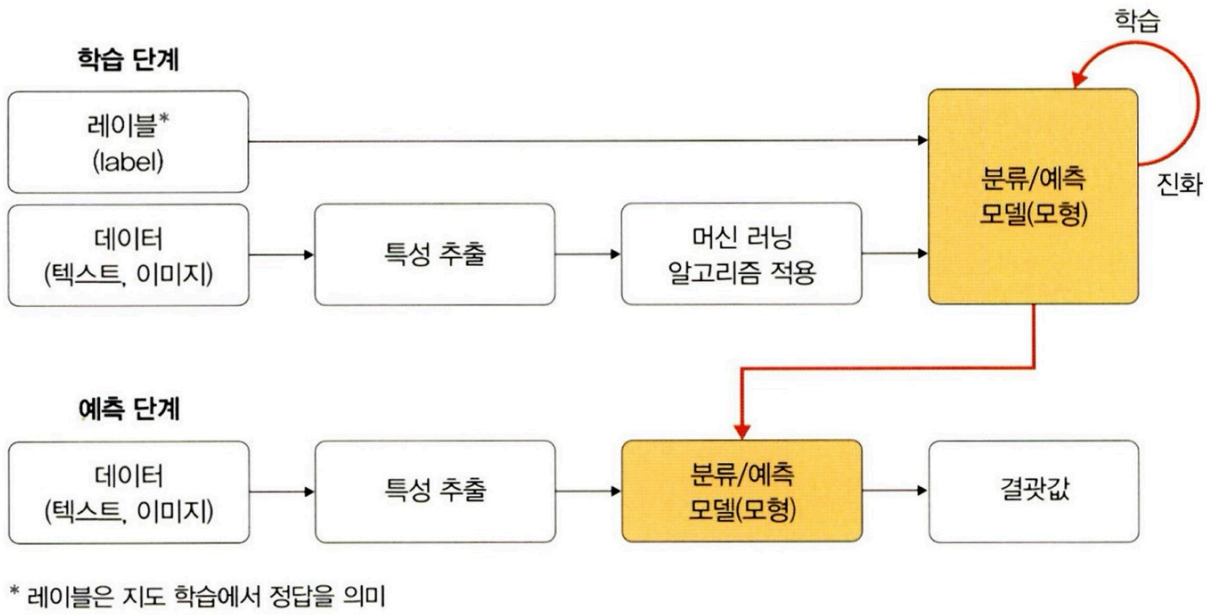
- 인공지능 > 머신러닝 > 딥러닝
- 머신러닝과 딥러닝 모두 학습 모델을 제거하여 데이터를 분류할 수 있는 기술
    - **머신러닝**은 주어진 데이터를 인간이 먼저 처리, 데이터의 특징 스스로 추출 불가
    - **딥러닝**은 인간이 하던 작업을 생략, 컴퓨터가 스스로 분석

구분	머신 러닝	딥러닝
동작 원리	입력 데이터에 알고리즘을 적용하여 예측을 수행한다.	정보를 전달하는 신경망을 사용하여 데이터 특징 및 관계를 해석한다.
재사용	입력 데이터를 분석하기 위해 다양한 알고리즘을 사용하며, 동일한 유형의 데이터 분석을 위한 재사용은 불가능하다.	구현된 알고리즘은 동일한 유형의 데이터를 분석하는 데 재사용된다.
데이터	일반적으로 수천 개의 데이터가 필요하다.	수백만 개 이상의 데이터가 필요하다.
훈련 시간	단시간	장시간
결과	일반적으로 점수 또는 분류 등 숫자 값	출력은 점수, 텍스트, 소리 등 어떤 것이든 가능

1.2 머신러닝이란

머신러닝: 컴퓨터 스스로 대용량 데이터에서 지식이나 패턴을 찾아 학습 및 예측 수행

- 머신러닝 학습 과정



- 주요 구성 요소: 데이터, 모델(모형)
  - **데이터**: 학습 모델을 만드는 데 사용하는 것
    - 편향되지 않는 훈련 데이터를 확보하는 것이 중요
    - 훈련 데이터셋과 테스트 데이터셋으로 분리해서 사용
      - 훈련 데이터셋을 훈련 데이터셋과 검증 데이터셋으로 분리해서 사용하기도 함
  - **모델**: 학습 단계에서 얻은 최종 결과물(가설)
    - 학습 절차: 모델 선택 -> 모델 학습 및 평가 -> 평가를 바탕으로 모델 업데이트

- 머신러닝 학습 알고리즘
  - **지도 학습**: 정답이 무엇인지 컴퓨터에 알려 주고 학습시키는 방법
  - **비지도 학습**: 정답을 알려 주지 않고 특징이 비슷한 데이터를 클러스터링하여 예측하는 학습 방법
  - **강화 학습**: 자신의 행동에 대한 보상을 받으며 학습을 진행

구분	유형	알고리즘
지도 학습 (supervised learning)	분류(classification)	• K-최근접 이웃(K-Nearest Neighbor, KNN) • 서포트 벡터 머신(Support Vector Machine, SVM) • 결정 트리(decision tree) • 로지스틱 회귀(logistic regression)
	회귀(regression)	선형 회귀(linear regression)
비지도 학습 (unsupervised learning)	군집(clustering)	• K-평균 군집화(K-means clustering) • 밀도 기반 군집 분석(DBSCAN)
	차원 축소 (dimensionality reduction)	주성분 분석 (Principal Component Analysis, PCA)
강화 학습 (reinforcement learning)	-	마르코프 결정 과정 (Markov Decision Process, MDP)

1.3 딥러닝이란

딥러닝: 인간의 신경망 원리를 모방한 심층 신경망 이론을 기반으로 고안된 머신 러닝 방법의 일종

- 딥러닝 학습 과정: 신경망과 역전파가 핵심 구성 요소
    - 데이터 준비: 파이토치나 케라스에서 제공하는 데이터셋 사용 / 캐글 같은 곳에 공개된 데이터 사용
    - 모델 정의: 신경망 생성, 은닉층 개수가 많을수록 성능이 좋아지지만 과적합 발생할 수 있음 주의!
    - 모델 컴파일: 활성화 함수, 손실 함수, 옵티마이저 선택
        - 훈련 데이터셋 형태가 연속형이면 평균 제곱 오차 사용
        - 훈련 데이터셋 형태가 이진 분류이면 크로스 엔트로피 선택
    - 모델 훈련: 한 번에 처리할 데이터양 지정, 데이터양이 많아지면 학습 속도가 느려지고 메모리 부족 문제를 야기할 수 있음
        - 전체 훈련 데이터셋에서 일정한 묶음으로 나누어 처리할 수 있는 배치, 훈련의 횟수인 에포크 선택이 중요
    - 모델 예측: 검증 데이터셋을 생성한 모델에 적용하여 실제로 예측을 진행해 보는 단계

- 딥러닝 학습 알고리즘
    - 지도 학습
        - 합성곱 신경망: 목적에 따라 이미지 분류, 이미지 인식, 이미지 분할로 분류
        - 순환 신경망: 시계열 데이터를 분류할 때 사용

        - LSTM - 망각 게이트, 입력 게이트, 출력 게이트를 도입하여 기울기 소멸 문제 해결
    - 비지도 학습
        - 워드 임베딩: 단어를 벡터로 표현, 워드투벡터와 글로브를 가장 많이 사용
        - 군집: 아무런 정보가 없는 상태에서 데이터를 분류하는 방법
    - 전이 학습: 사전에 학습이 완료된 모델을 가지고 우리가 원하는 학습에 미세 조정 기법을 이용하여 학습시키는 방법
        - 사전 학습 모델: 풀고자 하는 문제와 비슷하면서 많은 데이터로 이미 학습이 되어 있는 모델

구분	유형	알고리즘
지도 학습(supervised learning)	이미지 분류	• CNN • AlexNet • ResNet
	시계열 데이터 분석	• RNN • LSTM
비지도 학습 (unsupervised learning)	군집 (clustering)	• 가우시안 혼합 모델(Gaussian Mixture Model, GMM) • 자기 조직화 지도(Self-Organizing Map, SOM)
	차원 축소	• 오토인코더(AutoEncoder) • 주성분 분석(PCA)
전이 학습(transfer learning)	전이 학습	• 버트(BERT) • MobileNetV2
강화 학습(reinforcement learning)	–	마르코프 결정 과정(MDP)

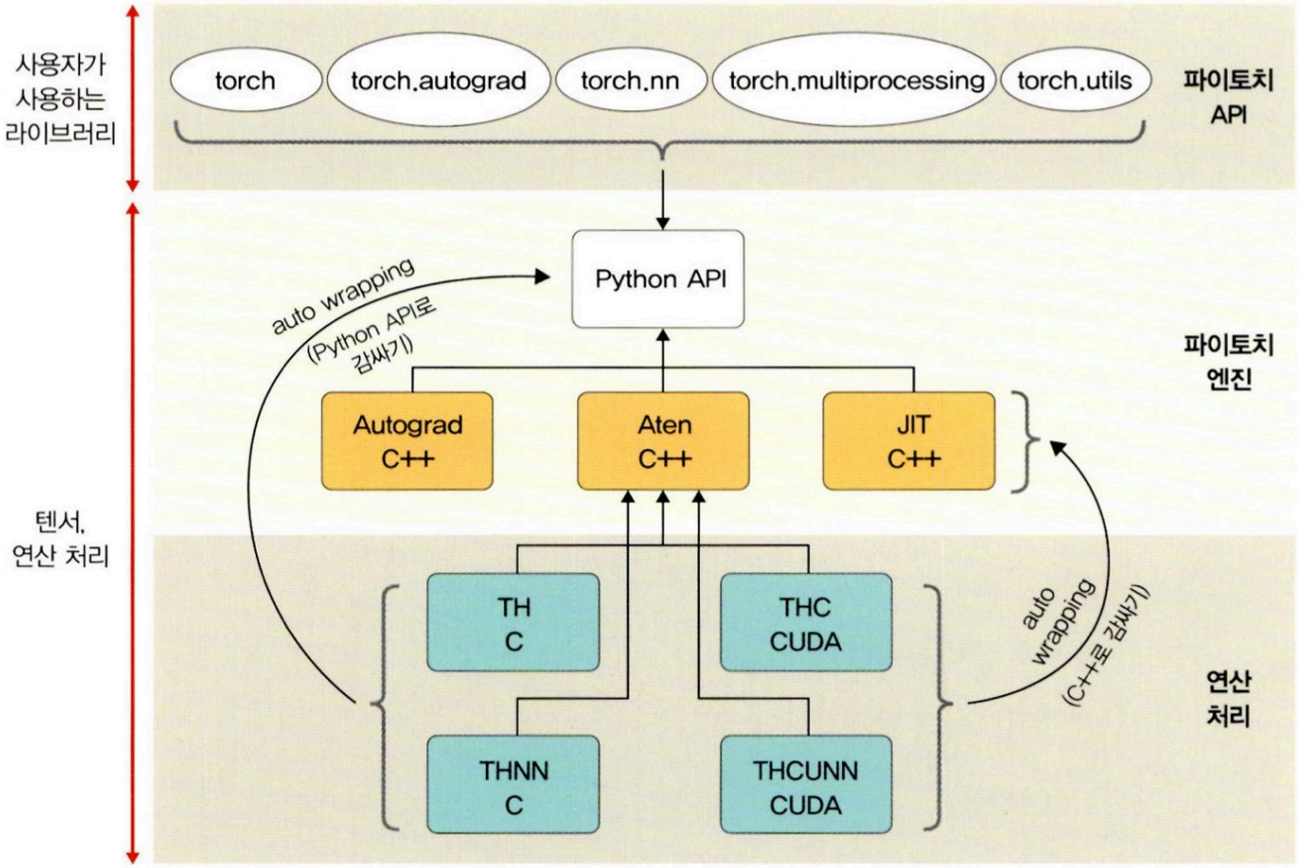
2장. 실습 환경 설정과 파이토치 기초

2.1 파이토치 개요

파이토치: 딥러닝 프레임워크, 루아 언어로 개발되었던 토치를 파이썬 버전으로 내놓은 것

- 파이토치 특징: GPU에서 텐서 조작 및 동적 신경망 구축이 가능한 프레임워크
    - GPU: 연산 속도를 빠르게 하는 역할
    - 텐서: 파이토치의 데이터 형태, 단일 데이터 형식으로 된 자료들의 다차원 행렬
    - 동적 신경망: 훈련을 반복할 때마다 네트워크 변경이 가능한 신경망
        - 연산 그래프를 정의하는 것과 동시에 값도 초기화되는 'Define by Run' 방식 사용

- 파이토치 장점: 효율적인 계산, 낮은 CPU 활용, 직관적인 인터페이스, 낮은 진입 장벽
- 파이토치의 아키텍쳐
  - 파이토치 API: 사용자가 이해하기 쉬운 API를 제공하여 텐서에 대한 처리와 신경망을 구축하고 훈련할 수 있도록 도움
    - 지원 패키지: torch(GPU 지원), torch.autograd(자동 미분), torch.nn(신경망 구축 및 훈련), torch.multiprocessing(파이썬 멀티프로세싱), torch.utils(DataLoader 및 기타 유틸리티를 제공)
  - 파이토치 엔진: Autograd C++, Aten C++, JIT C++, Python API로 구성
  - 연산 처리: 상위의 API에서 할당된 거의 모든 계산을 수행



2.2 파이토치 기초 문법

- 텐서 다루기
  - 텐서는 넘파이의 ndarray와 비슷하며 GPU에서의 연산도 가능
  - 텐서의 인덱스 조작: 인덱스 바로 지정, 슬라이드 등 사용 가능
  - 텐서의 자료형: torch.FloatTensor, torch.DoubleTensor, torch.LongTensor
  - view로 차원 변경, stack/cat으로 결합, t/transpose로 차원 교환

- 데이터 준비
    - 단순하게 파일을 불러와서 사용: 판다스 라이브러리를 이용하여 JSON, PDF, CSV 등의 파일을 불러옴
    - 커스텀 데이터셋을 만들어서 사용: 데이터를 한 번에 다 부르지 않고 조금씩 나누어 불러서 사용
    - 파이토치에서 제공하는 데이터셋 사용: 토치비전은 파이토치에서 제공하는 데이터셋들이 모여 있는 패키지

- 모델 정의: 모듈을 상속한 클래스를 사용
    - 계층: 모듈 또는 모듈을 구성하는 한 개의 계층
    - 모듈: 한 개 이상의 계층이 모여서 구성된 것
    - 모델: 최종적으로 원하는 네트워크

- 모델의 파라미터 정의
    - 손실 함수: 학습하는 동안 출력과 실제 값 사이의 오차를 측정
    - 옵티마이저: 데이터와 손실 함수를 바탕으로 모델의 업데이트 방법을 결정
    - 학습률 스케줄러: 미리 지정한 횟수의 에포크를 지날 때마다 학습률을 감소
    - 지표: 훈련과 테스트 단계를 모니터링

- 모델 학습
    - y=wx+b라는 함수에서 w와 b의 적절한 값을 찾는 것
    - 기울기 초기화, 오차 계산, 기울기 업데이트

- 모델 평가: 함수와 모듈을 이용하는 두 가지 방법이 있음

- 훈련 과정 모니터링
    - 텐서보드를 이용하면 학습에 사용되는 각종 파라미터 값이 어떻게 변화하는지 손쉽게 시각화하여 살펴볼 수 있

2.3 실습 환경 설정

- 아나콘다 설치
- 가상 환경 생성
    - conda create -n torch_book python=3.9.0
    - conda env list: 생성된 가상 환경 확인
    - activate torch_book: 가상 환경 활성화
    - conda env remove -n torch_book: 가상 환경 삭제

    - conda install ipykernel: 커널 설치
    - ipython kernel install --name tf2_book --user: 커널 연결
    - jupyter notebook: 주피터 노트북 접속
- 파이토치 설치
    - conda install pytorch==1.9.0 torchvision==0.10.0 torchaudio==0.9.0 -c pytorch

2.4 파이토치 코드 맛보기

- 범주형 데이터를 텐서로 변환
    - 범주형 데이터 -> dataset[category] -> 넘파이 배열 -> 텐서
- np.stack: 두 개 이상의 넘파이 객체를 합칠 때 사용
- get_dummies: 가변수로 만들어 주는 함수
- 워드 임베딩: 유사한 단어끼리 유사하게 인코딩되도록 표현하는 방법
- 모델의 네트워크 생성
    - 클래스 형태로 구현되는 모델은 nn.Module을 상속받음
    - __ init __(): 모델에서 사용될 파라미터와 신경망을 초기화하기 위한 용도로 사용
    - super().**init**(): 부모 클래스에 접근할 때 사용
    - 모델의 네트워크를 구축하기 위해 for 문을 사용하여 각 계층을 all_layers 목록에 추가
    - forward(): 학습 데이터를 입력받아 연산을 진행
- 딥러닝 분류 모델 성능 평가 지표
    - 정확도: 전체 예측 건수에서 정답을 맞힌 건수의 비율