CH1

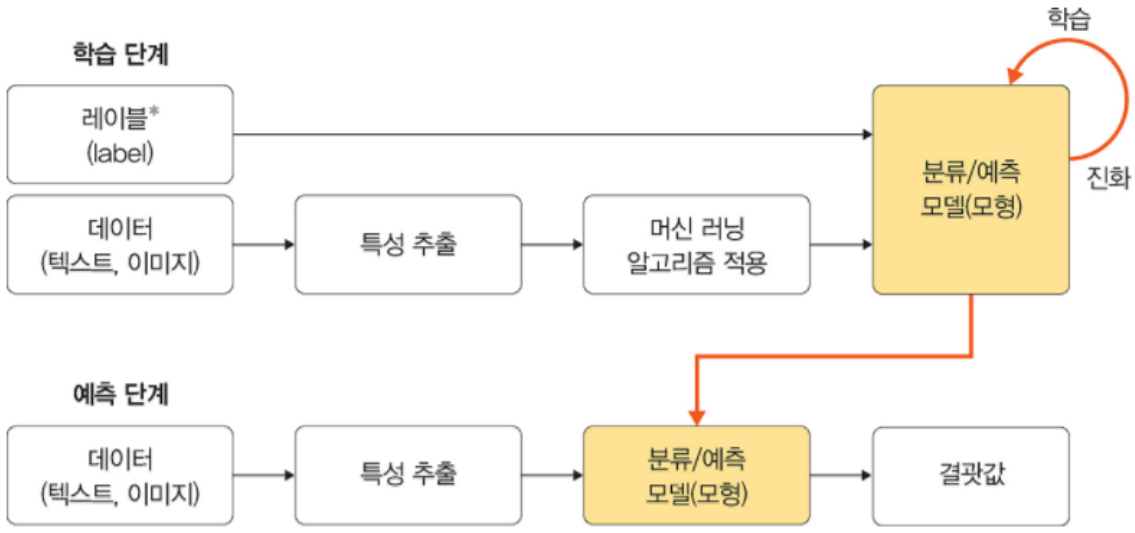
1.1 인공지능, 머신러닝과 딥러닝

1. 인공지능 : 인간의 지능을 모방하여 사람이 하는 일을 컴퓨터가 할 수 있도록 하는 기술. 구현 방식으로 머신러닝과 딥러닝이 있음
2. 머신러닝 : 주어진 데이터를 인간이 먼저 처리한 후, 데이터 특성을 컴퓨터에 인식시키고 학습시켜 문제 해결
3. 딥러닝 : 인간의 작업이 생략되어 데이터를 신경망에 적용하면 컴퓨터가 스스로 분석 후 해결

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

1. 학습 단계 : 훈련 데이터를 알고리즘에 적용하여 학습, 모형 생성
2. 예측 단계 : 새로운 데이터 적용하여 결과 예측



* 레이블은 지도 학습에서 정답을 의미

▲ 그림 1-3 머신 러닝 학습 과정

- 모델 선택 - 모델 학습&평가 - 모델 업데이트 → 세 단계를 반복하며 best 모델 찾기
- 검증 데이터셋 사용하는 경우도 있는데 이는 모델 성능을 평가를 위해(데이터 양 충분할 때만)

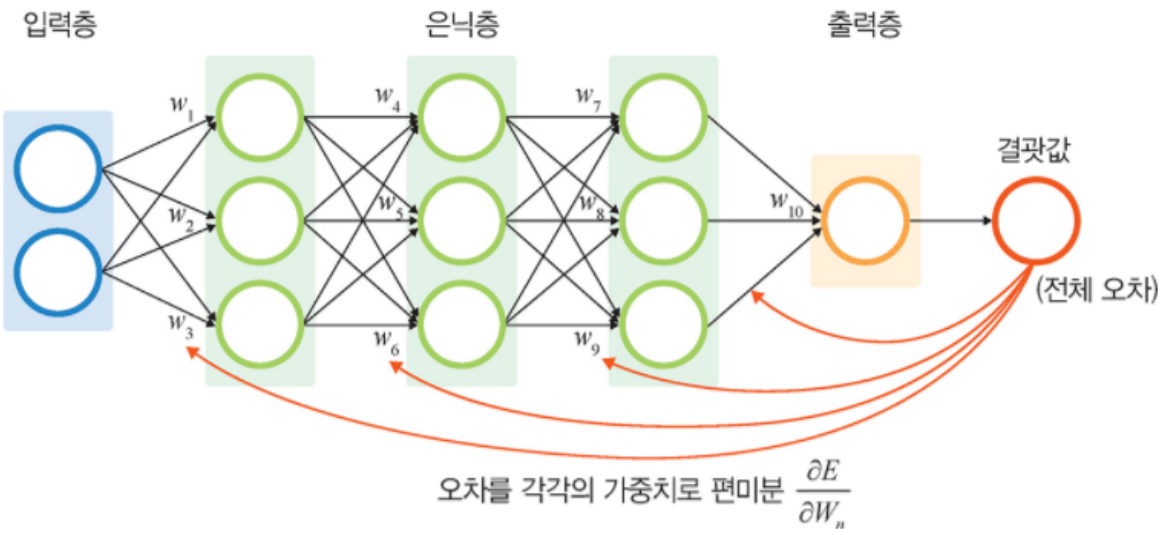
1. 지도학습 : 정답을 알려주고 학습시킴
2. 비지도학습 : 정답을 알려주지 않고, **특징이 비슷한 데이터 클러스터링**을 통해 예측
3. 강화학습 : 행동에 대한 보상을 받으며 학습 진행! 게임처럼!

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

- **인간의 신경망을 모방**한 심층 신경망 이론을 기반으로 고안된 머신러닝 기법
- 뇌의 뉴런과 시냅스 개념을 적용

단계	세부설명
데이터 준비	
모델 정의	신경망 생성, 은닉층 개수가 많을수록 성능 좋아지나 오버피팅 가능성도 높아짐.
모델 컴파일	활성화 함수 손실함수 옵티마이저 선택 ex) MSE, 크로스 엔트로피
모델 훈련	메모리 부족 문제로 인해 적당한 데이터 양을 한 번에 처리해야 함 → 배치와 에포크 선택! 파라미터와 하이퍼파라미터 최적 값 업데이트
모델 예측	



- 심층 신경망 → 역전파 계산하여 가중치 값 업데이트

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)

1. 지도학습
    a. 합성곱 신경망 CNN

        이미지 분류/인식/분할

    b. 순환 신경망 RNN

        시계열 데이터. 기울기 소멸 문제를 해결한 LSTM
3. 비지도학습
    a. 워드 임베딩

        단어 의미 벡터화 EX) Word2Vec, GloVe

    d. 군집

        머신러닝과 함께 사용하면 좋다~
5. 전이 학습

사전 학습 모델을 가지고 원하는 학습에 **미세 조정 기법**을 이용하여 학습시키는 방법. EX) VGG, 인셉션, MobileNet

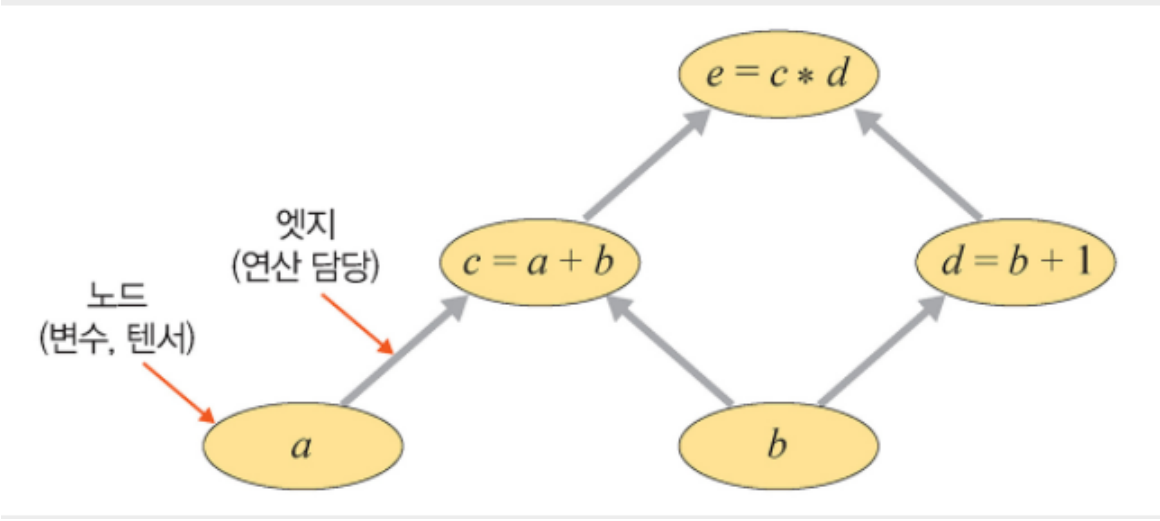
사전 학습 모델 : 풀고자 하는 문제와 비슷하면서 많은 데이터로 이미 학습되어 있는 모델

CH2

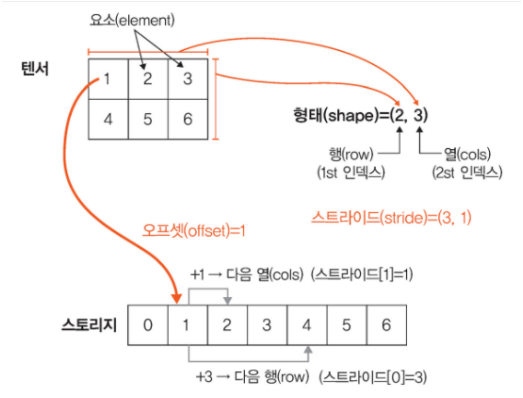
2.1 파이토치 개요

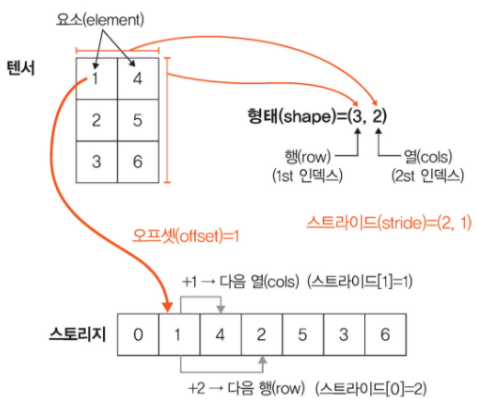
2017 공개된 딥러닝 프레임워크. 루아로 개발된 토치를 파이썬 버전으로 발매.

- 간결하고 구현이 빠름.
- **GPU에서 텐서 조작 및 동적 신경망 구축이 가능한 프레임워크**
- 텐서 : 파이토치 데이터 형태로 단일 데이터 형식으로 된 자료들의 다차원 행렬. 3차원 이상의 배열 형태 → torch.tensor() 꼴



→ 네트워크가 학습될 때 손실 함수의 기울기가 가중치와 바이어스를 기반으로 계산되며, 이후 경사 하강법을 사용하여 가중치가 업데이트된다.





- 오프셋 : 텐서에서 첫 요소가 스토리지에 저장된 인덱스

- 스트라이드 : 다음 요소를 얻기 위해 건너뛰기가 필요한 스토리지 요소 개수. 행 중심으로는 항상 1

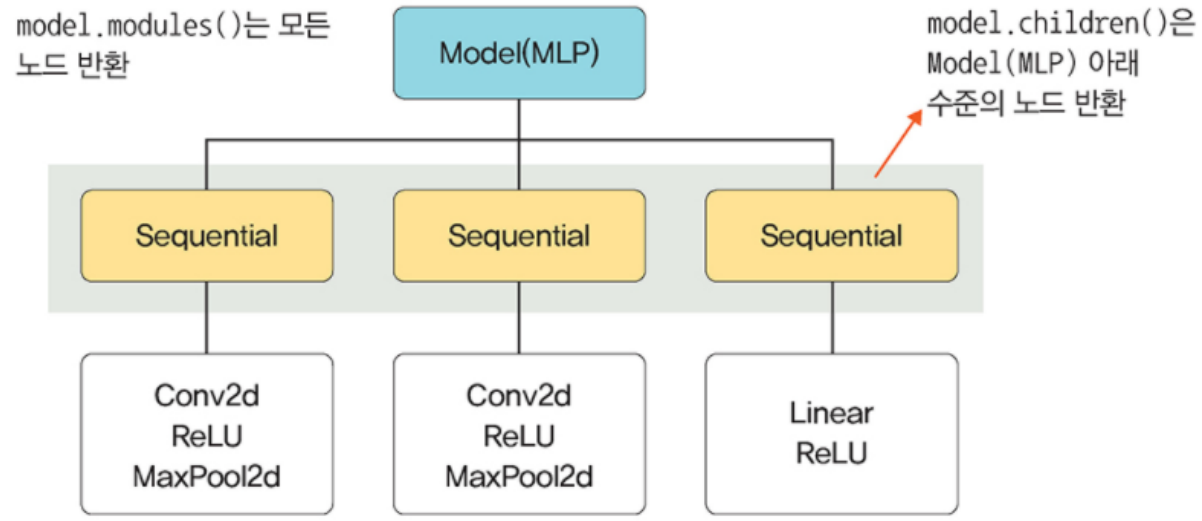
2.2 파이토치 기초 문법

```
import torch
temp= torch.tensor([[1,2], [3,4]], dtype=torch.float64)
temp.numpy() #ndarray로 변환

#인덱싱, 슬라이싱, 사칙연산 모두 가능
temp[0:3]
torch.tensor([1,2]) + torch.tensor([3,4])

#차원 조작
temp.view(4,1) #dim 4x1
temp.view(-1) #dim 1
temp.view(1,-1) #정해진 차원값으로 다른 값 유추

#데이터 준비
x= torch.from_numpy(data['x'].values).unsqueeze(dim=1).float()

#커스텀 데이터셋 구현 코드는 교재 참고
#파이토치 제공 데이터셋
pip install requests
from torchvision.datasets import MNIST
import requests
download_root='../MNIST'
train= MNIST(download_root, transform=mnist_transform, train=True, download=True)
```

- 계층 : 모듈/모듈 구성하는 한 개의 계층
- 모듈 : 한 개 이상의 계층이 모여서 구성
- 모델 : 최종 네트워크

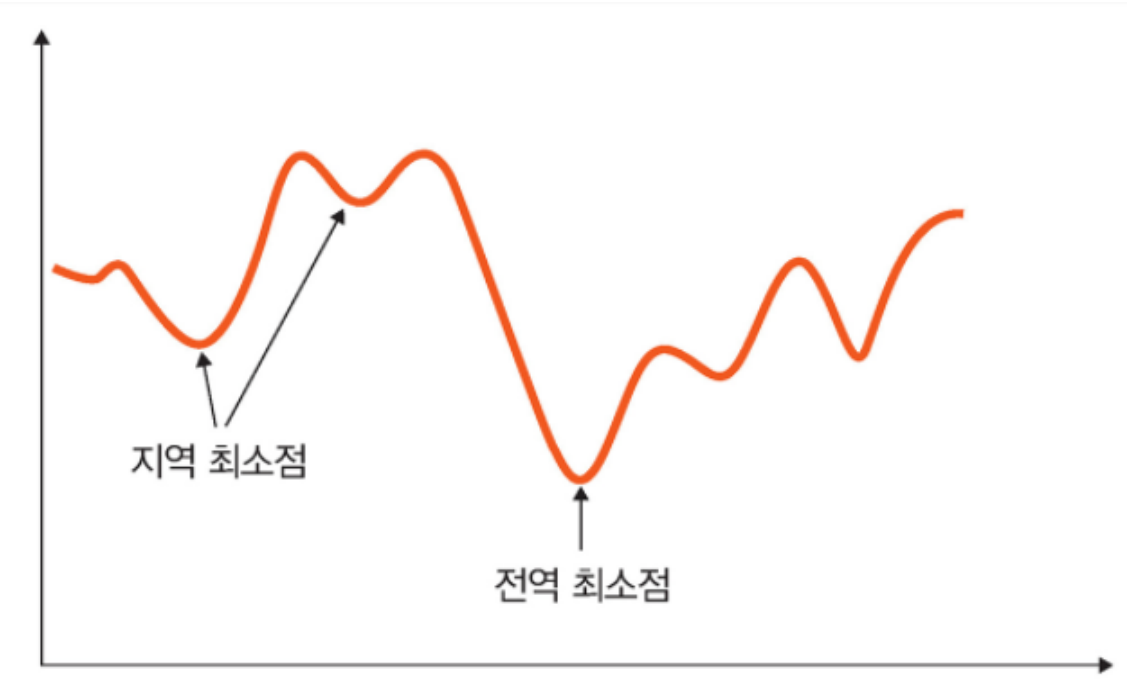
```
#단순 신경망
model = nn.Linear(in_features=1, out_features=1, bias=True)
class MLP(Module):
  def __init__(self, inputs):
    super(MLP, self).__init__()
  self.layer = Linear(inputs, 1) ------ 계층 정의
  self.activation = Sigmoid() ------ 활성화 함수 정의

  def forward(self, X):
    X = self.layer(X)
    X = self.activation(X)
    return X

#sequential 신경망
import torch.nn as nn
class MLP(nn.Module):
  def __init__(self):
    super(MLP, self).__init__()
    self.layer1 = nn.Sequential(
    nn.Conv2d(id_channels=3, out_channels=64, kernel_size=5),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(2))

    self.layer2 = nn.Sequential(
```

```
    nn.Conv2d(id_channels=64, out_channels=30, kernel_size=5),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(2))

    self.layer3 = nn.Sequential(
    nn.Linear(in_features=30*5*5, out_features=10, bias=True),
    nn.ReLU(inplace=True))

  def forward(self, X):
    X = self.layer1(X); X = self.layer2(X)
    X = X.view(X.shape[0],-1); X = self.layer3(X)
    return X

model=MLP()
```



1. 파라미터 정의

- 손실함수 : BCELoss, CrossEntropyLoss, MSELoss
- 옵티마이저 : step() 메서드를 통해 업데이트 optim.Adam/ASGD,,,



- 학습률 스케줄러 : 지정한 횟수의 에포크를 지날 때마다 학습률 감소, 전역 최소점 근처에 다다르면 학습률을 줄여 최적점을 찾도록! optim.lr_scheduler.LambdaLR/StepLR,MultiStepLR,,,

2. 모델 훈련

딥러닝 학습 절차	파이토치 학습 절차
모델, 손실 함수, 옵티마이저 정의	모델, 손실 함수, 옵티마이저 정의
	optimizer.zero_grad(): 전방향 학습, 기울기 초기화
전방향 학습(입력 → 출력 계산)	output = model(input): 출력 계산
손실 함수로 출력과 정답의 차이(오차) 계산	loss = loss_fn(output, target): 오차 계산
역전파 학습(기울기 계산)	loss.backward(): 역전파 학습
기울기 업데이트	optimizer.step(): 기울기 업데이트

- optimizer.zero_grad() : 기울기 초기화
- loss.backward() : 기울기 자동 계산

3. 모델 평가

```
import torchmetrics

preds=torch.randn(10,5).softmax(dim=-1)
```

```
target= torch.randint(5, (10,))
acc= torchmetrics.functional.accuracy(preds, target)

metric= torchmetrics.Accuracy()
```

2.3 실습 환경 설정

2.4 파이토치 코드 맛보기