1. AI, ML, DL의 정의

1) 데이터 구성요소(Feature/Label)

- 데이터의 중요성
  - 머신러닝은 규칙을 직접 코딩 X, 데이터에서 규칙을 학습 O
  - 데이터(Feature/Label)의 분포와 관계가 머신러닝의 학습 결과를 결정
- Feature(피처, 특성)
  - 모델이 예측에 사용하는 입력 정보
  - 예측, 판단의 근거/단서
- Label(라벨, 목표값)
  - 모델이 예측하려는 정답
  - 학습의 목표값

2) ML 실생활 예시

- 유튜브 추천
  - Feature : 각 영상들의 정보(장르, 크리에이터, 조회수, 좋아요 수 등), 사용자 정보(시청 이력, 구독 채널 등)
  - Label : 영상에 대한 사용자 피드백(시청 여부, 좋아요 클릭 여부)
- 스팸 메일 분류
  - Feature : 메일 제목, 발신자, 단어 빈도
  - Label : 스팸/정상

3) 단일 피쳐 기반 학습

(1) 1D 피쳐 기반 학습

- 1D 피쳐 기반 학습
  - 1D = 1차원
  - 정의

        - Feature가 하나일 때 머신 러닝이 학습하는 가장 단순한 형태
- $$Income_i = f^*(Yearsof Education_i) + \varepsilon_i$$
    - 데이터셋 D : 30명의 Years of Education(피쳐)와 Income(라벨) 쌍
    - $D = \{(\text{Years of Education}_i, \text{Income}_i)\}_{i=1}^{30}$
        - 미지의 함수(f^*)
            - Feature와 Label 사이의 실제 평균 관계
            - but 직접 관측 X
            - 오차가 포함된 데이터(점)만 관측 가능
        - 측정오차(ε)
            - 데이터에는 주로 측정 오차가 섞여 있음
            - 원인 : 측정 기기의 한계, 환경적 요인 등
            - 따라서, 데이터 = 참 함수 + 오차($f^* + \varepsilon$)
- 피쳐와 라벨의 관계를 잘 나타낸 함수 f
    - 데이터를 설명하는 여러 함수 후보가 존재
    - 어떤 함수가 가장 잘 맞는지 학습해야 함

(2) 모델과 가설 공간

- 학습(Learning)
    - 입력(Feature) -> 출력(Label) 관계를 찾는 과정
    - 평균 관계를 하나의 함수로 표현함
    - 하지만 관계를 표현할 수 있는 함수는 무수히 많음
- 가설 공간(Hypothesis Space)
    - 관계를 표현할 수 있는 모든 후보 함수들의 모음
    - 피쳐 공간과 라벨 공간 위에서 정의된 함수들의 집합
    - ex) 무수히 많은 선형 함수의 집합
- 모델(Model)
    - 가설 공간에 속한 특정 함수 f
    - 특정 선형 함수

(3) 학습이란

- 학습
    - 주어진 데이터와 성능척도를 바탕으로 가설공간의 후보들 중 최적의 모델을 선택하는 과정
    - 데이터 -> 가설공간 -> 선택된 모델

4) 복수 피쳐 기반 학습

(1) 2D 피쳐 기반 학습

- $Income_i = f^*(YearsofEducation_i) + \varepsilon_i$
  - f^* : 미지의 참 함수(입력과 출력을 이어주는 숨겨진 진짜 함수)
    - 관측 불가능
  - 학습 전 : 어떤 가설공간을 사용할까?
  - 학습 후 : 데이터를 활용하여 어떤 모델 f을 선택해야 할까?

(2) 일반적 용어 정리 및 모델 가정

- $\text{Income} = f^*(\text{Years of Education}, \text{Seniority}, \dots) + \epsilon \implies Y = f^*(\mathbf{X}) + \epsilon$
  - $Income$: 우리가 예측하려는 라벨(반응/목표) 변수 -> Y로 표기
  - $YearsofEducation$: 첫번째 피쳐(입력/예측) 변수 -> X_1로 표기
  - $Seniority$: 두번째 피쳐(입력/예측) 변수 -> X_2로 표기
  - 일반적인 p차원 피쳐(총 p개의 피쳐) 벡터 : $\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{bmatrix} \in \mathbb{R}^p$
  - 모델(함수형) : $f^* : \mathbb{R}^p \longrightarrow \mathbb{R}, \quad Y = f^*(\mathbf{X}) + \epsilon$
  - 측정오차 ε : ε는 피쳐 X와 독립 및 $\mathbb{E}[\epsilon] = 0$로 가정함

(3) 왜 $f(\cdot)$를 학습하는가?

- 예측
  - 잘 학습된 f가 있으면, 새로운 입력 $X = x$에서 반응/목표 Y를 예측할 수 있음
- 중요 특성 파악
  - 피쳐들 $X = (X1, X2, \dots, Xp)$의 어떤 특성이 Y를 설명하는데 중요하고, 어떤 것은 덜 중요(무관)한지 알 수 있음
  - ex) 근속연수(Seniority), 교육기간(Years of Education)은 소득(Income) 에 큰 영향을 줄 수 있지만, 혼인 여부(Marital Status) 는 영향이 거의 없을 것임
- 해석 가능성
  - f의 복잡도에 따라 각 구성요소 X_j 가 Y에 어떻게 영향을 미치는지(증가/감소 방향, 민감도 등) 이해할 수 있음

2. 지도학습(supervised learning)은 무엇인가?

1) 지도학습의 개념

(1) 지도학습이란

- 지도학습의 정의
  - 훈련 데이터가 아니라, 처음 보는 데이터에서의 예측 성능 향상
  - 입력 + 정답(레이블)을 가지고 예측 규칙을 배우는 방법
  - 이미 갖고 있는 데이터를 활용하여 학습하지만, 궁극적으로 새로운 데이터에서의 예측을 잘 하고자 하는데 초점
  - ex) 어제까지 고객 데이터로 "내일 이탈할 고객" 미리 알기, 기존 거래 사기(Fraud) 데이터로 새로운 사기 탐지
- 데이터
  - 입력(특성)과 정답(라벨)이 쌍으로 있는 데이터
- 목표
  - 새 입력이 들어오면 정답을 잘 맞추는 규칙을 학습
- 지도학습의 종류
  - 회귀: 예측값이 숫자(가격, 점수, 온도)
  - 분류: 예측값이 범주(스팸/정상, 질병 유/무)

(2) 지도학습 용어

- 특성(Feature, x)
  - 예측에 쓰는 설명 변수
  - ex) 집값 예측 {지역, 평수, 방수, 연식}
    이메일 스팸 필터링 {제목, 내용 텍스트, 송신인}
- 라벨(Label, y)
  - 맞춰야 하는 정답
  - ex) 집값, 스팸/정상이메일
- 예측값 $(\hat{y})$
  - 모델이 내놓은 결과(숫자 또는 범주)
- 오류(Error)

- 예측값$(\hat{y})$과 라벨(y)의 차이: $(\hat{y} - y)$

2) 회귀(Regression)

(1) 회귀 문제

- 입력으로부터 숫자를 얼마나 정확히 예측할까?
  - Feature: 면적·방수·연식 → Label: 집값(원 단위)
  - Feature: 매체별 광고비(TV/라디오/온라인) → Label: 매출액
- 라벨 및 예측 모델의 출력
  - 연속적인 수치

(2) 회귀 오류 : 평균제곱오차(MSE)

- 평균제곱오차(Mean Squared Error)
  - 각 데이터에서 정답(y_i)과 예측$(\hat{y}_i)$의 평균 제곱 차이값
  - $\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- 해석
  - 큰 오류를 더 크게 벌주므로, 전체 오류 수준을 한눈에 봄
- 참고
  - 데이터와 같은 단위를 쓰고 싶으면 RMSE(MSE의 제곱근)도 사용
    - RMSE에서 R은 root을 의미함
  - $\mathrm{RMSE} = \sqrt{\mathrm{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$

(3) 회귀 설명력 : R^2(결정계수)

- 결정계수
  - 라벨의 분산 중에서 특성으로 설명되는 비율
  - "평균만 쓰는 단순한 예측"보다 얼마나 더 잘 맞추는지를 0~1 사이로 나타낸 값
  - $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (\text{단}, \bar{y} = y_i\text{들의 평균값})$
- 해석
  - 1에 가까울수록 설명력이 높고, 낮을수록 설명력이 낮음
- 질문
  - R2가 음수가 나올 수 있을까?

- 답변
  → 예측값들이 평균값보다도 못한다면 나올 수 O !!!

3) 분류(Classification)

(1) 분류(Classfication) 문제

- 입력으로부터 범주는 얼마나 정확히 가려낼까?
  - Feature: 메일 내용·보낸이 이메일주소 → Label: 스팸/정상
  - Feature: 종양 반경, 면적 → Label: 악성/양성
- 라벨
  - 범주 라벨(이진/다중)

(2) 분류 정확도(Accuracy)

- 정확도
  - 전체 중 맞춘 비율
  - $\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(y_i = \hat{y}_i)$
    - $\mathbb{I}$는 지시(indicator) 함수이며, $\mathbb{I}(A) = 1 \text{ if } A \text{ true}$
- 정확도만 보면 발생하는 문제
  - 불균형 데이터(양성 1%, 음성 99%)에서는 전부 음성이라 해도 정확도가 99%로 보일 수 있음
- 결론
  - 정확도만 보지 말고 다른 지표도 함께 봐야 안전

(3) 혼동행렬(Confusion Matrix)

- 혼동행렬
  - 예측과 실제 값 사이의 관계를 행렬 형태로 표현
  - TP: 실제 양성, 예측도 양성
  - TN: 실제 음성, 예측도 음성
  - FP: 실제는 음성인데 양성이라 함(오탐)
  - FN: 실제는 양성인데 음성이라 함(누락)
- 정밀도(Precision)
  - "양성이라 판정한 것 중" 진짜 양성의 비율 = TP/(TP+FP)
- 재현율(Sensitivity or Recall)
  - "진짜 양성 가운데" 잡아낸 예측 양성 비율 = TP/(TP+FN)

- F1-score
  - 정밀도와 재현율의 조화평균
  - $\mathrm{F1} = 2 \times \frac{\text{정밀도} \times \text{재현율}}{\text{정밀도} + \text{재현율}}$

4) 학습의 목적

(1) 학습의 목적

- 학습의 목적은 테스트 예측(일반화)
  - 학습 모델의 성능 평가는 모델이 처음 보는(학습에 사용되지 않은) 데이터로 평가
    - 일반화(generalization) 오류의 최소화 지향
- 훈련 데이터에서 성능이 아무리 좋아도, 새로운 데이터에서 성능이 떨어지면 실전엔 사용할 수 없음

(2) 오버피팅(overfitting)이란?

- 오버피팅(overfitting)
  - 훈련 데이터의 우연한 패턴/잡음까지 외워버려서 훈련에서는 잘 맞지만 테스트에서는 성능이 나빠지는 현상
  - 현상: 훈련 오류 급격히 낮음, 테스트 오류 높음/요동
- 오버피팅이 왜 안 좋은가?
  - 표본(sample) 의존·불안정: 훈련 데이터는 모집단의 일부 표본이라 우연한 잡음이 섞임. 이 것에만 과하게 맞추어 학습하면 샘플 몇 개만 바뀌어도 예측이 크게 흔들림(분산↑).
  - 일반화 실패: 보지 못한 데이터(테스트) 오류가 커짐, 모(population)집단 성능과 격차가 벌어짐.

(3) 오버피팅에 대한 오해

- 오버피팅 ≠ 분포 변화(distribution shift)로 인한 에러 증가
  - 분포 변화로 인한 오류: 훈련 데이터 분포와 테스트 분포가 다름으로 (환경·계절·센서 변경 등) 성능이 떨어지는 현상
  - 분포 변화로 인한 에러 증가는 모델이 과적합하지 않아도 발생 가능

(4) 오버피팅 vs 언더피팅

- 오버피팅 vs 언더피팅 (균형 잡기)
  - 오버피팅: 모델이 너무 복잡

    → 잡음까지 학습(테스트 성능 나쁨)

- 언더피팅: 모델이 단순하거나 학습이 완료되지 않음

  → 중요한 패턴을 놓침(오류 큼)
- 해결 실마리
  - 더 많은 데이터, 테스트 데이터를 활용한 모델 선정. 교차 검증

AI & 기계학습 핵심 노트

1. 교차 검증 (Cross-Validation)

1.1 테스트 성능 평가 및 오류 유형

테스트 오류는 학습에 쓰지 않은 새로운 데이터에 대한 모델의 평균 예측 오류이며, 이는 곧 일반화 성능을 의미함

모델 복잡도가 증가함에 따라 훈련 오류는 계속 하강하지만, 테스트 오류는 U자형 곡선을 그림

언더피팅 (Underfitting): 모델이 너무 단순하여 중요한 패턴을 놓친 상태 (U자 곡선의 왼쪽, 복잡도 낮음)

오버피팅 (Overfitting): 모델이 지나치게 복잡하여 훈련 데이터의 노이즈까지 학습한 상태 (U자 곡선의 오른쪽, 복잡도 높음)

1.2 검증셋 (홀드아웃) 접근의 특징

가용 샘플을 훈련셋과 검증셋으로 단 한 번 무작위 분할하여 오류를 추정

주요 단점:

어떤 샘플이 검증에 들어가느냐에 따라 추정치가 매우 가변적
전체 데이터가 아닌 일부만 훈련에 사용되므로, 테스트 오류가 실제보다 과대 추정되는 경향이 있음

1.3 K-겹 교차 검증 (K-Fold CV)

데이터를 크기가 동일한 K개 폴드로 무작위 분할하여 사용

핵심 원리: K번의 반복 동안 각 폴드가 번갈아 가며 검증셋이 되고, 나머지 $K-1$개 폴드가 훈련셋이 됨
K개의 오류 값(MSE_k)을 평균하여 테스트 오류를 추정

LOOCV (Leave-One-Out CV): K의 값이 전체 관측치 수 n과 같을 때 ($K=n$)를 의미
관측치 하나만 검증셋으로 두고 $n-1$개로 훈련

2. 비지도 학습 (Unsupervised Learning)

2.1 정의 및 목표

비지도 학습은 레이블(정답) 없이 입력 데이터의 내재된 구조, 패턴, 잠재 서브그룹을 찾아내는 학습
-군집화(클러스터링), 차원 축소, 시각화, 밀도 추정/이상치 탐지 등이 있음

클러스터링의 목표:

데이터를 집단 내부는 서로 유사하고, 집단 간은 서로 상이하도록 하위 집단으로 분할하는 것

2.2 K-means

클러스터링클러스터의 수 K를 미리 지정

핵심 아이디어: 클러스터 내부 변동의 합이 최소가 되도록 분할을 찾음

알고리즘 특성: 반복 과정은 목표 함수 값을 감소시키지만, 전역 최솟값은 보장하지 못하며 초기값에 따라 지역 최솟값에 수렴할 수 있음

-> 서로 다른 초기값으로 여러 번 시도하는 것이 권장

2.3 계층적 군집

클러스터 수 K를 사전에 고정 X

결과를 **덴드로그램(Dendrogram)**으로 제공하여 전체 구조를 파악할 수 있음

링크 유형 (Single, Complete, Average 등) 선택에 따라 클러스터링 결과가 달라짐

2.4 클러스터링 주의점: 데이터 스케일링

클러스터링은 주로 **거리(유사도)**를 기반으로 계산되므로, 입력 변수 간의 단위 차이가 결과에 미치는 영향을 줄여야 함

-> 모든 변수를 동등하게 반영하기 위해 표준화 (스케일링, 평균 0 표준편차 1로 변환) 과정이 필수적으로 요구

선형회귀(Linear Regression)

입력 변수 X와 출력 변수 Y 사이 관계를 직선 형태로 근사하여 새로운 값을 예측하는 통계적 방법

$$Y = b_0 + b_1 X + \varepsilon$$

최소 제곱법(least squares)

실제 관측값과 예측값의 차이를 제곱해 합한 값을 최소화하는 방법(b0, b1을 추정)

$$\text{잔차}(residual) : e_i = y_i - \hat{y}_i$$

$$RSS : e_1^2 + e_2^2 + e_3^2 ... + e_n^2$$

단순 선형 회귀의 한계

- 단순 선형 회귀는 오직 하나의 독립 변수(X)와 종속 변수(Y) 관계만 모델링
- 변수 1개로는 종속 변수 변화를 충분히 설명하기 어려움

다중 선형 회귀(Multiple Linear Regression)

독립 변수(Feature)가 여러 개 존재할 때 사용하는 회귀 분석 기법

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 ... + \beta_p X_p + \varepsilon$$

선형 회귀 주의사항

- 훈련 데이터에서 성능(예측)이 좋다고 좋은 모델은 아니다.
- 테스트 성능 평가가 필요
- 여러 설명변수들이 서로 강하게 연관되면 계수 계산이 불안정해져 테스트 성능이 좋지 않을 수 있음.

로지스틱 회귀(Logistic Regression)

분류(Classification)

무언가를 미리 정해둔 카테고리(범주) 중 하나로 나누는 것

범주형 변수: 값 사이에 순서나 크기의 개념이 없는 변수(연속값이 아님)

f(X)를 학습하여, 입력 X가 속할 범주 C를 예측하는 것

분류 문제에 선형회귀 사용시 예측 확률이 0에서 1사이를 넘어 부적절함

로지스틱 회귀

함수의 출력 범위가 모든 입력에 대해 0~1사이의 범위를 가지는 함수를 사용해 예측

ex) sigmoid(0~1), tanh(-1~1), arctan(-1~1)

$$sigmoid : y = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}$$

$$odds : \frac{p(y=1|x)}{p(y=0|x} = \frac{p(y=1|x)}{1-p(y=1|x)} = \frac{\text{성공확률}}{\text{실패확률}}$$

$$logit(p) = log\frac{p(y=1|x)}{1-p(y=1|x)} = log\frac{\text{성공확률}}{\text{실패확률}}$$

MLE(Maximum Likelihood Estimation)

현재 확률 함수가 데이터를 얼마나 잘 설명하는지 나타내는 지표

NN(Neural Network)

Shallow 네트워크

hidden layer가 1개인 NN

$$y = f[x, \phi] = \phi_0 + \phi_1 a[\theta_{10} + \theta_{11}x] + \phi_2 a[\theta_{20} + \theta_{21}x] + \phi_3 a[\theta_{30} + \theta_{31}x]$$

Deep 네트워크

hidden layer가 2개 이상인 NN

Fitting

Loss Function

모델이 얼마나 잘못 예측하는지를 측정하는 함수

$$L[\phi, f[X_i, \phi], \{X_i, Y_i\}_{i=1}^{I}]$$

학습: 손실함수를 최소화하는 파라미터를 찾음

경사하강법(Gradient Descent)

손실 함수를 최소화하기 위해 파라미터를 반복적으로 갱신하는 알고리즘

미분값의 기호에 따라 파라미터의 크기와 방향을 조정

확률적 경사 하강법(Stochastic Gradient Descent)

Non-convex 문제에서 local 최소점에 빠지는 문제를 해결하기 위해

매 스텝마다 전체 데이터에 대한 미분값을 구하여 업데이트함

역전파(Backpropagation)

출력 오차를 기준으로 그래프를 거꾸로 따라가며 연쇄법칙으로 각 노드(파라미터 포함)의 미분값을 계산하는 절차

시험 정리

생성 일시	@2025년 10월 30일 오후 1:46

1. AI & 기계학습 방법론 1 선형회귀-69-150.pdf

Shallow 네트워크:

Hidden Unit

- 비선형성 도입: Hidden Unit은 활성화 함수를 사용해 입력값의 선형 변환 결과에 비선형 변환을 적용 -> 신경망이 단순한 선형 모델의 한계를 넘어 복잡하고 비선형적인 패턴을 학습
- 특징 변환/추출: 입력 데이터(x)로 부터 유의미한 새로운 특징(h)를 자동으로 학습, 추출

Hidden Unit에 비선형 활성화 함수를 적용하면 입력 공간이 여러 개의 영역으로 나누어짐, 각 영역은 선형 함수로 모델을 표현하게 됨
즉 shallow 네트워크는 전체 함수를 여러 개의 조각난 선형 함수들의 조합으로 표현하는 방식

활성화 함수가 비선형적이면 단 하나의 은닉층을 가진 신경망으로 충분히 많은 수의 Hidden Unit을 사용하면 임의의 연속 함수를 원하는 정확도로 근사할 수 있음

Deep 네트워크(다중 은닉층 모델)

Deep 네트워크는 여러 개의 Shallow 네트워크를 연속적으로 연결(합성)하여 층(Layer)을 깊게 쌓은 구조
구조: 은닉층이 2개 이상,
표현력: 층의 깊이에 따라 기하급수적으로 증가
복잡 함수 근사 효율: 복잡 함수 표현 시 더 적은 파라미터로 표현 가능

더 많은 비선형 영역을 만들어냄 복잡한 함수를 Shallow 네트워크0보다 훨씬 더 효율적으로 표현

$h^k = a(W^k * h^{(k-1)} + b^k$

1. AI & 기계학습 방법론 1 선형회귀-150-215.pdf

손실함수

- 학습 데이터셋: input/output
- **예측값$\hat{y}$**이 **실제 정답** y 얼마나 다르거나 잘못되었는지를 측정하는 함수입니다.
- 이 값이 작을수록 모델이 더 정확하게 학습되었다는 의미이며, 학습의 목표는 이 손실함수 값을 최소화하는 것입니다.

학습 (Learning)
• **정의: 손실함수를 최소화**하는 최적의 파라미터(모델의 가중치 $\mathbf{W}$와 편향 $\mathbf{b}$를 찾는 과정입니다.
• **목표:** 모델의 파라미터$\mathbf{\Theta} = \{\mathbf{W}, \mathbf{b}\}$를 업데이트하여 $J(\mathbf{\Theta})$가 최소가 되도록 합니다.

경사하강법(Gradient descent)

정의: 손실함수 $J(\mathbf{\Theta})$를 최소화하기 위해, 현재 위치에서 **기울기(경사, Gradient)**를 계산하고 기울기가 가리키는 방향의 **반대 방향**으로 파라미터를 조금씩 업데이트하며 이동하는 방법입니다.
• **수식:** $$\mathbf{\Theta}^{(t+1)} = \mathbf{\Theta}^{(t)} - \eta \frac{\partial J(\mathbf{\Theta}^{(t)})}{\partial \mathbf{\Theta}}$$
  ◦ $\mathbf{\Theta}^{(t)}$: 현재 시점 t의 파라미터 값
  ◦ η (에타): **학습률(Learning Rate)**. 기울기를 따라 이동하는 **보폭**을 결정합니다.
  ◦ $\frac{\partial J}{\partial \mathbf{\Theta}}$: 손실함수의 파라미터에 대한 편미분 (기울기)

경사 하강법 순서

1. 편미분(전체 손실에 대한 기울기) 구하기
2. 파라미터 업데이트(미분값의 방향을 반대 방향으로 이동(손실 최소화))

Convex vs Non-convex 최적화 문제

Convex: 곡선이 항상 U처럼 아래로 볼록, 그래프 위 임의 두 점을 잇는 직선이 그래프 위(또는 같은 위치)로 있음: 전역(global) 최소값이 유일함 → 최적화 쉬움

Non-convex: 봉우리 골짜기 오목한 구간이 섞인 모양, 두 점을 이은 직선이 그래프 아래로 내려가는 구간이 생김: 여러 개의 지역(local)최소값 또는 새들(saddle)점이 있음 → 최적화가 어려움

⇒ local 최소점에 빠지기 쉬움, 스텝별 계산양이 많음

해결법: 전체 데이터를 한번에 쓰는 대신, 무작위로 선택한 데이터 샘플

• **경사 하강법의 유형 :**
  ◦ **전체 경사 하강법 (Batch Gradient Descent): 전체 학습 데이터**를 사용하여 기울기를 계산하고 한 번 업데이트합니다. 계산량이 많지만 안정적으로 최솟값에 수렴합니다.

  ◦ **확률적 경사 하강법 (SGD, Stochastic Gradient Descent): 단 하나의 데이터 포인트**를 사용하여 기울기를 계산하고 업데이트합니다. 업데이트별 계산량이 적어 빠르지만, 불안정한 경향이 있습니다.

무작위 확률로 샘플된 일부 데이터(batch)만 사용해 기울기 계산

4. 확률적 경사 하강법 (Stochastic Gradient Descent, SGD)의 장점
1. **계산량 절감: 전체 데이터 대신 일부 데이터(미니 배치)만 사용**하여 업데이트별 **계산량 (Computation)**이 적습니다.
2. **Local 최소점 탈출 가능성:** 손실함수 표면에서 계산된 경사가 완벽하지 않아 약간의 진동이 발생하므로, **Local 최소점(지역 최솟값)**에 갇힐 확률이 적습니다.
3. 국소 최솟값: 일부 배치로 기울기를 계산하기 때문에 노이즈가 섞여 있음
4. 노이즈가 있지만 여전히 타당한 업데이트

5. 선형 회귀 예시를 통한 Gradient Descent 이해

• **목표:** 단순 선형 회귀 ($y = \phi_0 + \phi_1 x$)에서 ϕ_0와 ϕ_1를 업데이트하여 MSE $J(\phi_0, \phi_1)$를 최소화하는 것입니다.
• **편미분 (기울기 계산):**
  ◦ J를 ϕ_0에 대해 편미분: $\frac{\partial J}{\partial \phi_0} = -\frac{2}{I}\sum_{i=1}^{I}(y_i - \hat{y}_i)$
  ◦ J를 ϕ_1에 대해 편미분: $\frac{\partial J}{\partial \phi_1} = -\frac{2}{I}\sum_{i=1}^{I}(y_i - \hat{y}_i)x_i$
• **업데이트:** 계산된 기울기를 학습률 η와 곱하여 기존 ϕ 값에서 빼줌으로써 ϕ를 손실이 감소하는 방향으로 이동시킵니다.

- 손실 함수의 값이 줄어드는 방향으로 파라미터를 이동하는 과정

3-3. 역전파의 단계적 절차

역전파는 손실로부터 각 레이어별 파라미터의 미분을 거꾸로 구하는 과정입니다

1. **각 단계별 계산 분해:** 신경망의 계산을 $f_0, f_1, f_2, \ldots$ 와 같은 개별 함수로 분해하여 정의합니다
2. **순전파:** 각 Layer별 값(f_k)을 계산합니다
3. **오차 미분(역전파의 시작):** 각 Layer별 f_k에 대한 **출력 손실(l)**의 미분($\frac{\partial l}{\partial f_k}$)을 연쇄 법칙을 사용하여 거꾸로 계산해 나갑니다
   - 예: $\frac{\partial l}{\partial f_2} = \frac{\partial l}{\partial f_3}\frac{\partial f_3}{\partial f_2}$
4. **파라미터 미분:** 계산된 오차 미분을 사용하여, 손실에 대한 각 Layer의 **파라미터($\boldsymbol{\Omega}$) 미분**($\frac{\partial l}{\partial \boldsymbol{\Omega}}$)을 최종적으로 계산합니다