핵심 머신러닝 - 2

생성자	재 재환 김
태그	머신러닝

2.1 표기법

이 절에서는 학교에서 배웠지만 잊어버릴 수 있는 수학 정의와 표기법을 복습합니다.

2.1.1 데이터 구조

- **스칼라 값 (Scalar Value)**: 단일 숫자를 의미합니다. 예를 들어, 15나 -3.25와 같은 값이 스칼라 값입니다.
- **벡터 (Vector)**: 스칼라 값을 일정한 순서로 나열한 것입니다. 각 스칼라 값은 벡터의 성분(컴포넌트, component)이라고 합니다. 벡터는 공간에서 특정 방향을 가지며, 화살표로 표현할 수 있습니다.
    - 예시: 벡터 $a = [2, 3, 5]$또는 $b = [-2, 3, 5]$.

벡터와 성분 표현

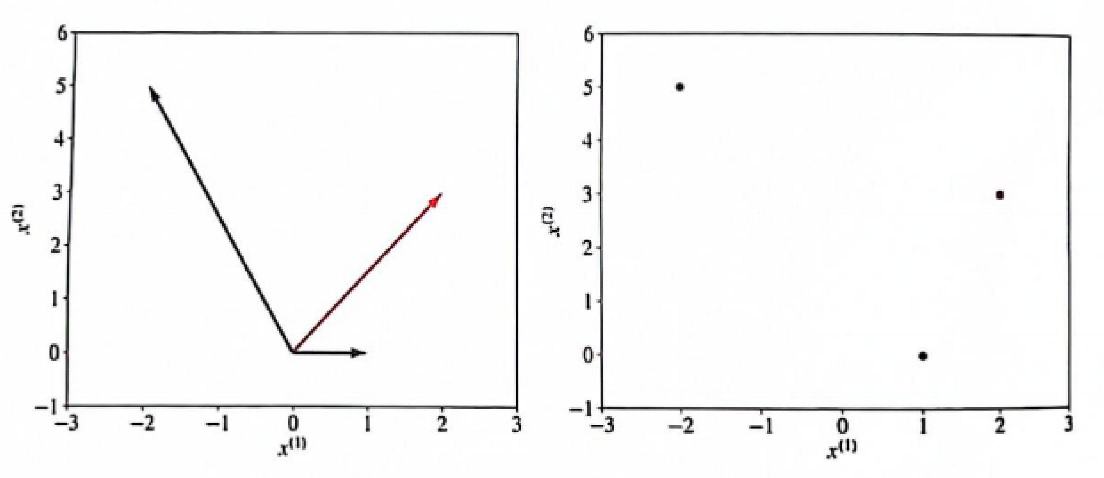

그림 2.1 세 벡터를 화살표로 표현한 예와 점으로 표현한 예

- **그림 2.1**에서는 벡터를 화살표로 시각적으로 표현한 예가 나옵니다.
    - 왼쪽 그래프에서는 두 벡터 $\mathbf{a}^{(1)} = [2, 3]$와 $\mathbf{a}^{(2)} = [3, 2]$가 서로 다른 방향으로 그려져 있으며, 이 벡터들의 성분을 통해 공간에서의 위치와 방향을 파악할 수 있습

니다.

  - 벡터는 공간 내에서 특정한 방향과 크기를 가지며, 그 방향성을 그림으로 표현한 것입니다.

벡터의 성분

- 각 벡터의 성분은 $\mathbf{a}_1^{(1)} = 2$, $\mathbf{a}_2^{(1)} = 3$와 같이 나타낼 수 있으며, 이는 벡터의 특정 차원에서의 위치를 나타냅니다.
- 예를 들어, 벡터 $\mathbf{a}^{(1)}$는 x-축에서 2, y-축에서 3을 차지하고 있습니다.

행렬의 구성

- **행렬 (Matrix)**: 여기서 행렬은 숫자들을 직사각형 형태로 나열한 것으로, 다수의 벡터를 모은 결과로도 해석할 수 있습니다.
  - 예시: 행렬 $A = \begin{bmatrix} 2 & 4 & -3 \\ 21 & -6 & -1 \end{bmatrix}$
  - 이 행렬은 두 개의 행으로 구성되며, 각 행에 있는 숫자들이 벡터의 성분처럼 다루어질 수 있습니다.
- 이 책에서는 행렬을 대문자 기호로 표기하고, 행과 열로 구분된 값들을 사용하여 수식을 나타냅니다.

집합과 관련 기호

- **집합**: 집합은 일련의 원소들을 모아놓은 수학적 개념으로, 숫자나 다른 수학적 객체들이 중복되지 않고 모여 있는 상태를 말합니다.
  - 예시: $S = \{1, 3, 5, 8\}$
  - 집합은 보통 중괄호 $\{\}$를 사용하여 표기하며, 집합 내 원소들이 나열됩니다.
- **교집합**: 두 집합 S_1과 S_2가 있을 때, 공통된 원소들만 포함하는 새로운 집합을 교집합이라고 합니다. 교집합은 기호 $\cap$으로 나타냅니다.
  - 예시: $S_1 = \{1, 3, 5\}$, $S_2 = \{3, 4, 5\}$일 때, $S_1 \cap S_2 = \{3, 5\}$
- **합집합**: 두 집합의 모든 원소를 포함한 새로운 집합을 합집합이라고 하며, 기호$\cup$로 나타냅니다.
  - 예시: $S_1 = \{1, 3, 5\}$, $S_2 = \{3, 4, 5\}$일 때, $S_1 \cup S_2 = \{1, 3, 4, 5\}$

2.1.2 대문자 시그마 기호 (∑)

- **대문자 시그마 기호 (Summation Sign)**: 여러 원소들의 합을 나타내는 기호로, 수학적으로 자주 사용됩니다.
  - 예시: 집합 $X = \{x_1, x_2, ..., x_n\}$에서 각 원소들의 합은 다음과 같이 대문자 시그마 기호를 사용하여 표현할 수 있습니다.

    $\sum_{i=1}^{n} x_i = x_1 + x_2 + \cdots + x_n$
- 대문자 시그마 기호는 주로 수열이나 벡터의 원소들을 더할 때 사용됩니다. 이는 여러 항을 하나로 묶어 간결하게 표현하는 수학적 기호입니다.

벡터의 성분 합

- 벡터의 성분을 모두 더하는 경우에도 대문자 시그마 기호를 사용하여 표현할 수 있습니다.
  - 예시: $\sum_{j=1}^{m} \mathbf{a}_j$는 벡터의 모든 성분 $\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_m$ 을 더하는 것을 의미합니다.

2.1.3 대문자 파이 기호

- **대문자 파이 기호**는 곱셈 연산을 표현하는데 사용됩니다.
- 시그마 기호(∑)가 덧셈을 표현하는 것과 마찬가지로, 파이 기호($\prod$)는 여러 항목을 곱할 때 사용됩니다.
  - 예시:

    $\prod_{i=1}^{n} x_i = x_1 \cdot x_2 \cdot \cdots \cdot x_n$
  - 이 기호는 x_1부터x_n까지 모든 값을 곱하는 연산을 나타냅니다.
  - 곱셈을 간략하게 표현하는 수학적 방법으로 사용됩니다.

2.1.4 집합 연산

- 새로운 집합을 만드는 방식 중 하나로, 주어진 집합 S에서 특정 조건을 만족하는 원소들로만 구성된 부분 집합을 생성할 수 있습니다.

  SS
  - 예시: $S' = \{x | x \in S, x > 3\}$
  - 이는 집합 S에서 $x > 3$인 원소들로만 이루어진 집합을 의미합니다.
- **집합의 크기 (Cardinality)**: 집합에 포함된 원소의 개수를 나타냅니다. 집합 S의 크기는 $\mid S \mid$로 표현하며, $\mid S \mid = 5$라면 집합 S에 5개의 원소가 포함되어 있다는 의미입니다.

2.1.5 벡터 연산

1. 벡터의 덧셈과 뺄셈

- 벡터는 성분별로 더하거나 뺄 수 있습니다.
  - 두 벡터 $\mathbf{x} = [x_1^{(1)}, x_2^{(1)}, \ldots, x_m^{(1)}]$와 $\mathbf{y} = [y_1^{(2)}, y_2^{(2)}, \ldots, y_m^{(2)}]$ 가 있을 때,
    - 벡터의 합: $\mathbf{x} + \mathbf{y} = [x_1^{(1)} + y_1^{(2)}, x_2^{(1)} + y_2^{(2)}, \ldots, x_m^{(1)} + y_m^{(2)}]$
    - 벡터의 차: $\mathbf{x} - \mathbf{y} = [x_1^{(1)} - y_1^{(2)}, x_2^{(1)} - y_2^{(2)}, \ldots, x_m^{(1)} - y_m^{(2)}]$

2. 스칼라곱

- 두 벡터의 스칼라곱(내적)은 각 성분의 곱을 모두 더한 값을 나타냅니다. 스칼라곱의 결과는 단일 값(스칼라)입니다.
  - 예시: $\mathbf{x} \cdot \mathbf{y} = x_1^{(1)} \cdot y_1^{(2)} + x_2^{(1)} \cdot y_2^{(2)} + \cdots + x_m^{(1)} \cdot y_m^{(2)}$

3. 행렬과 벡터의 곱

- 행렬 W와 벡터 $\mathbf{x}$의 곱은 새로운 벡터를 생성합니다.
  - 행렬 W는 여러 행과 열로 구성되며, 벡터의 각 성분을 행렬과 곱하여 새로운 벡터를 만들어냅니다.
  - 예시: $W = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ x^{(3)} \end{bmatrix}$

    행렬과 벡터의 곱은 다음과 같이 계산됩니다:

$$W\mathbf{x} = \begin{bmatrix} w_{1,1}x^{(1)} + w_{1,2}x^{(2)} + w_{1,3}x^{(3)} \\ w_{2,1}x^{(1)} + w_{2,2}x^{(2)} + w_{2,3}x^{(3)} \\ w_{3,1}x^{(1)} + w_{3,2}x^{(2)} + w_{3,3}x^{(3)} \end{bmatrix}$$

    - 이 결과는 새로운 벡터로 나옵니다.

행렬의 전치 (Transpose)

- 벡터를 전치한다는 것은 벡터의 차원을 전환하는 것을 의미합니다.
  - 열 벡터를 행 벡터로 바꾸거나, 그 반대의 경우에도 사용됩니다.
  - 예시: $\mathbf{x}^T = [x^{(1)}, x^{(2)}, x^{(3)}]$

2.1.6 함수

함수(function)는 집합 X의 원소를 다른 집합 Y의 원소와 연결하는 관계입니다. 함수는 입력 값(독립 변수)에 대해 출력 값을 반환하는 연산입니다. 함수는 일반적으로 다음과 같이 표현됩니다:

$$y = f(x)$$

- 여기서 x는 입력값(또는 인수, argument)이며, y는 함수의 결과인 출력값입니다. 함수 f는 주어진 x 값에 대해 특정한 y값을 할당하는 역할을 합니다.
- **예시**: 함수 $f(x)$는 특정 수학적 연산에 따라 결과를 생성하는 관계로, 그래프나 수식을 통해 시각화할 수 있습니다.

그래프 해석

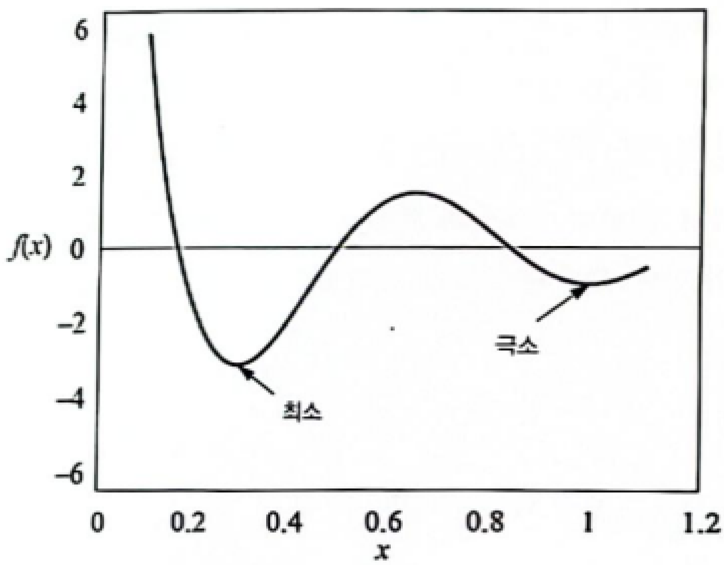

그림 2.2 함수의 극소와 최소

- **그림 2.2**에서는 함수의 그래프를 보여주고 있습니다. 이 그래프는 함수의 입력값 x에 따라 함수 $f(x)$의 값을 시각적으로 표현한 것입니다.
  - 그래프에서 특정 지점의 **최솟값 (minimum)**과 **최댓값 (maximum)**을 관찰할 수 있습니다.
  - 함수가 특정 지점에서 최솟값이나 최댓값을 가질 때, 그 지점을 함수의 극값 (extremum)이라고 합니다.

도메인과 공역

- 함수 f의 정의역 (domain)은 함수에 입력될 수 있는 값들의 집합입니다.

- 함수 f의 공역 (codomain)은 함수의 결과값이 속하는 집합입니다.

2.1.7 max와 min, arg max와 arg min 연산

- **최댓값 연산 (max)**: 집합 $A = \{a_1, a_2, \ldots, a_n\}$에서 함수 $f(a)$가 가장 큰 값을 가질 때, 그 값을 **최댓값**이라고 합니다.
    - 예시: $\max(f(a_1), f(a_2), \ldots, f(a_n))$
- **최솟값 연산 (min)**: 함수가 가장 작은 값을 가질 때 그 값을 **최솟값**이라고 부릅니다.
- **arg max와 arg min**: 이는 최댓값 또는 최솟값을 만드는 원소 x 자체를 구하는 연산입니다.
    - **arg max**는 함수의 최댓값을 만드는 x 값을 찾고, **arg min**은 최솟값을 만드는 x 값을 찾습니다.

2.1.8 대입 연산자

- **대입 연산자**: $a \leftarrow f(x)$는 함수 $f(x)$의 결과를 값 a에 대입하는 것을 의미합니다. 이 연산은 특정 변수에 함수를 적용한 값을 저장할 때 사용됩니다.

2.1.9 도함수와 기울기

- **도함수 (derivative)**: 도함수는 함수가 얼마나 빨리 변하는지를 나타내는 수학적 개념입니다. 도함수는 함수의 변화율(증가 또는 감소 속도)을 나타냅니다.
    - **표기법**: $f\prime(x)$ 또는 $\frac{df}{dx}$
    - 도함수의 값은 그래프에서 특정 지점에서의 기울기로 해석될 수 있습니다. 이는 함수의 변화 속도를 정량적으로 나타내는 중요한 값입니다.
- **기울기 (slope)**: 함수의 그래프에서 특정 구간에서의 증가율 또는 감소율을 나타냅니다. 도함수는 함수가 특정 지점에서 어떻게 변화하는지를 나타내는 중요한 도구입니다.

도함수의 계산 예시

- **기본 함수의 도함수**: 함수$f(x) = x^2$의 도함수는 $f'(x) = 2x$입니다. 즉, 이 함수는 각 지점에서의 기울기를 나타내며, 도함수의 값이 x의 변화에 따라 달라집니다.
    - 예시: $f(x) = x^2, f'(x) = 2x$
- **합성 함수의 도함수**: 만약 함수가 다른 함수의 조합으로 표현된다면, 연쇄 법칙 (chain rule)을 사용하여 도함수를 구할 수 있습니다.
    - 예시: 함수 $F(x) = g(f(x))$일 때, $F'(x) = g'(f(x)) \cdot f'(x)$로 계산할 수 있습니다.

- 예시로 $F(x) = 5x^2 + 10$인 경우, 이 함수의 도함수는 $F'(x) = 10x$입니다.

편미분 (Partial Derivative)

- 벡터 함수 또는 다변수 함수의 도함수를 구할 때, 각 변수에 대해 독립적으로 미분을 적용해야 합니다. 이를 **편미분**이라고 부릅니다.
  - 예시: 함수 $f(x_1, x_2) = ax + b$에 대해 x_1에 대한 편미분을 구하면, 다음과 같이 계산됩니다:

    $$\frac{\partial f}{\partial x_1} = a$$

  - 편미분은 여러 변수를 포함하는 함수에서 특정 변수에 대한 변화를 구할 때 사용됩니다.

밀도 함수

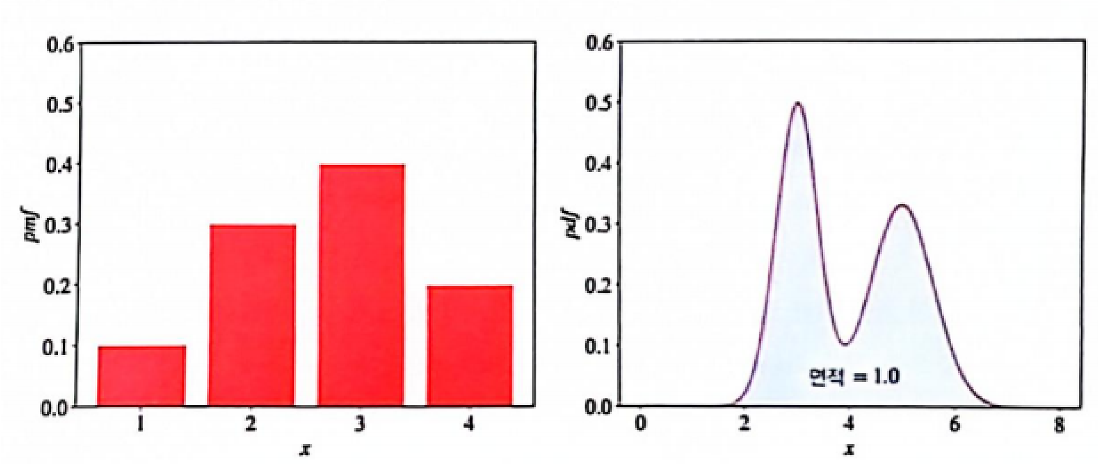

그림 2.3 확률 질량 함수(pmf)와 확률 밀도 함수(pdf)

- 오른쪽 그래프에서는 함수의 확률 밀도 함수 (Probability Density Function)를 보여주고 있습니다. 이는 특정 구간 내에서 값이 나올 확률을 나타냅니다.
  - 그림 2.3에서는 특정 구간 내에서의 확률 분포를 막대그래프와 곡선으로 표현하고 있습니다.
  - **밀도 함수**는 특정 구간에서의 면적을 통해 확률을 계산하며, 그 면적이 1이 되는 특성을 가집니다.

2.2 확률 변수

- **확률 변수 (Random Variable)**: 확률 변수는 무작위 현상에서 나오는 결과를 수치로 나타낸 변수입니다. 이 변수는 불확실성을 수반하며, 특정 확률 분포를 따릅니다.

- **예시**: 주사위를 굴렸을 때, 나올 수 있는 값(1, 2, 3, 4, 5, 6)은 확률 변수로 표현할 수 있습니다.
- 확률 변수는 일반적으로 대문자 X로 나타내며, 해당 변수의 값은 무작위로 선택됩니다.

- **확률 분포**: 확률 변수는 특정 확률 분포에 의해 결정되며, 각 값이 나올 확률이 확률 밀도 함수나 이산 확률 분포로 나타납니다. 이러한 분포를 통해 확률 변수가 나타낼 수 있는 값들의 범위와 그 확률을 알 수 있습니다.

이산 확률 변수

- 이산 확률 변수(discrete random variable)는 정수나 몇 개의 구분된 값들로만 나타낼 수 있는 변수를 의미합니다.
  - 예시: 주사위 던지기의 결과(1, 2, 3, 4, 5, 6) 또는 동전 던지기의 결과(앞면, 뒷면) 등이 이산 확률 변수입니다.
- 이산 확률 변수에 대한 **확률 분포**는 각 값이 나올 확률을 나타내는 함수로 설명됩니다. 이를 확률 질량 함수 (Probability Mass Function, PMF)라고 합니다.
  - 예시: $P(X = \text{앞면}) = 0.5, P(X = \text{뒷면}) = 0.5$
  - 모든 가능한 값의 확률의 합은 항상 1이 되어야 합니다.

연속 확률 변수

- 연속 확률 변수(Continuous Random Variable, CRV)는 연속적인 값들을 가질 수 있는 변수를 의미합니다. 예를 들어, 길이, 시간, 온도 등은 무한히 많은 실수 값으로 표현됩니다.
- 확률 밀도 함수 (Probability Density Function, PDF)는 연속 확률 변수에서 특정 구간에서 값이 나올 확률을 나타냅니다. 이 경우 특정 값 자체에 대한 확률은 0이지만, 구간에 대한 확률을 계산할 수 있습니다.
  - 예시: $P(a \leq X \leq b) = \int_a^b f_X(x)dx$
  - 확률 밀도 함수의 전체 영역 아래 면적은 항상 1이어야 합니다.

기댓값 (Expected Value)

- **기댓값**은 확률 변수의 평균적인 값을 의미하며, 확률 변수가 취할 수 있는 값들과 그 값들이 나올 확률의 곱으로 구해집니다. 이는 확률 변수의 장기적인 평균을 의미합니다.
  - 이산 확률 변수의 기댓값: $E[X] = \sum_{i=1}^{n} x_i \cdot P(X = x_i)$
  - 연속 확률 변수의 기댓값: $E[X] = \int_{-\infty}^{\infty} x \cdot f_X(x)dx$

분산 (Variance)

- **분산**은 확률 변수의 값들이 기댓값을 기준으로 얼마나 퍼져 있는지를 나타냅니다. 분산이 크면 값들이 평균에서 많이 떨어져 있고, 분산이 작으면 값들이 평균 근처에 모여 있다는 뜻입니다.
  - 분산의 공식은 다음과 같습니다: $\mathrm{Var}(X) = E[(X-\mu)^2]$

    여기서 μ는 기댓값 $E[X]$을 의미합니다.
  - 이산 확률 변수의 분산: $\mathrm{Var}(X) = \sum_{i=1}^{n}(x_i-\mu)^2 \cdot P(X=x_i)$
  - 연속 확률 변수의 분산: $\mathrm{Var}(X) = \int_{-\infty}^{\infty}(x-\mu)^2 \cdot f_X(x)dx$

표준편차 (Standard Deviation)

- **표준편차**는 분산의 제곱근으로, 값들이 기댓값에서 평균적으로 얼마나 떨어져 있는지를 나타냅니다. 이는 분산과 동일한 단위로 나타낼 수 있어 해석이 용이합니다.
  - 공식: $\sigma = \sqrt{\mathrm{Var}(X)}$

2.3 비편향 추정량

비편향 추정량(unbiased estimator)은 주어진 표본을 통해 모집단의 특성을 추정할 때 사용되는 통계적 방법입니다. 이 추정량은 모집단의 실제 값과 평균적으로 일치하는 값을 제공합니다. 즉, 표본으로부터 얻은 추정값이 평균적으로 모집단의 실제 값을 왜곡하지 않도록 하는 특성을 가집니다.

- **비편향 추정량의 정의**: 표본 $S_X = \{x_1, x_2, \ldots, x_n\}$에서 계산한 통계량 $\hat{\theta}(S_X)$의 기댓값이 모집단의 실제 파라미터 값θ와 같다면, 이를 비편향 추정량이라고 합니다.
  - 수식으로는 다음과 같이 표현됩니다: $E[\hat{\theta}(S_X)] = \theta$
- 이때, $\hat{\theta}(S_X)$는 표본으로부터 계산된 추정값이고, θ는 모집단의 실제 값입니다. 이러한 특성은 통계 추정 과정에서 중요한 역할을 합니다.

2.4 베이즈 규칙

- 베이즈 규칙(Bayes' Theorem) 은 조건부 확률을 계산하는데 사용되는 중요한 공식입니다. 이는 사건 Y가 발생했을 때, 다른 사건 X가 발생할 확률을 계산하는 방식입니다.
- 베이즈 규칙의 일반적인 형태는 다음과 같습니다:

  $$P(X=x|Y=y) = \frac{P(Y=y|X=x) \cdot P(X=x)}{P(Y=y)}$$

  여기서:

- $P(X = x|Y = y)$는 $Y = y$일 때 $X = x$일 확률 (사후 확률, Posterior Probability)
- $P(Y = y|X = x)$는$X = x$일 때 $Y = y$일 확률 (가능도, Likelihood)
- $P(X = x)$는 $X = x$일 확률 (사전 확률, Prior Probability)
- $P(Y = y)$는 $Y = y$일 확률 (정규화 상수, Normalizing Constant)

이 공식은 주로 의사결정 이론, 통계적 추론, 기계 학습 등에서 활용되며, 주어진 데이터를 기반으로 특정 사건이 발생할 확률을 계산하는 데 매우 유용합니다.

2.5 파라미터 추정

파라미터 추정(parameter estimation)은 주어진 데이터로부터 모집단의 파라미터를 추정하는 과정을 의미합니다. 이러한 추정은 데이터를 분석하고 모델을 구축할 때 매우 중요한 과정입니다.

- 예를 들어, 정규 분포(Gaussian distribution)를 가정한 경우, 정규 분포의 평균(μ)과 분산(σ^2)을 추정할 수 있습니다.
  - 정규 분포의 확률 밀도 함수(PDF)는 다음과 같이 주어집니다:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- 베이즈 추정을 사용하여 데이터로부터 파라미터를 추정할 수 있습니다. 베이즈 추정은 사전 확률(prior probability)과 가능도(likelihood)를 결합하여 사후 확률(posterior probability)을 계산하는 방법입니다. 이때 사후 확률이 최대가 되는 값을 추정치로 사용할 수 있습니다.
- 파라미터 추정의 일반적인 수식은 다음과 같습니다:

$$P(\theta = \hat{\theta}|X = x) = \frac{P(X=x|\theta=\hat{\theta}) \cdot P(\theta=\hat{\theta})}{\sum_{\theta} P(X=x|\theta) \cdot P(\theta)}$$

  이 식은 사전 확률과 가능도를 결합하여 사후 확률을 계산하는 방식입니다.

2.6 파라미터 vs. 하이퍼파라미터

- **파라미터 (Parameter)**: 모델이 데이터로부터 자동으로 학습하는 값입니다. 예를 들어, 선형 회귀에서의 기울기와 절편이 이에 해당합니다. 이 값들은 학습 과정에서 조정되어 모델이 데이터에 최적화된 예측을 할 수 있게 합니다.
- **하이퍼파라미터 (Hyperparameter)**: 학습 알고리즘의 특성을 결정하는 설정 값으로, 모델 학습 과정에서 변경되지 않습니다. 사용자가 사전에 설정하며, 예로는 학습률(learning rate), 신경망의 층 수, 정규화 계수 등이 있습니다. 하이퍼파라미터는 모델 성능에 큰 영향을 미치므로 적절한 설정이 중요합니다.

- 파라미터와 달리 직접 학습되지 않으며, 하이퍼파라미터 튜닝을 통해 최적값을 찾습니다.
- 일반적으로 **교차 검증(cross-validation)**을 통해 선택합니다.

2.7 분류 vs. 회귀

- **분류 (Classification)**: 주어진 데이터를 두 개 이상의 범주로 나누는 문제입니다. 데이터가 특정 범주에 속할 확률을 예측하는 것이 목표입니다.
  - **예시**: 스팸 메일 분류(스팸 여부), 암 진단(암 여부)
  - 클래스 레이블이 있으며, 예측값이 유한한 개수의 클래스 중 하나일 때 적용됩니다.
  - **이진 분류(Binary Classification)**: 클래스가 두 개인 경우. 예: 메일의 스팸 여부 예측.
  - **다중 클래스 분류(Multiclass Classification)**: 클래스가 세 개 이상인 경우. 예: 고양이, 강아지, 새 등 여러 카테고리 중 하나를 예측.
- **회귀 (Regression)**: 연속적인 값을 예측하는 문제입니다. 주어진 데이터에 대해 실수 범위의 값을 예측하는 것이 목표입니다.
  - **예시**: 집 가격 예측, 주식 가격 예측
  - 예측값이 실수 범위에 있을 때 적용됩니다. 예: 특정 날씨 조건에서의 온도 예측.

2.8 모델 기반 학습 vs. 사례 기반 학습

- **모델 기반 학습 (Model-Based Learning)**: 데이터를 학습하여 모델을 만든 후, 이 모델로 새로운 데이터를 예측하는 방식입니다. 특정 알고리즘을 사용해 모델을 학습하고, 이후 예측에 활용합니다.
  - **예시**: SVM, 선형 회귀 모델
  - 훈련 데이터로 일반화된 모델을 만들어 새로운 데이터에 대해 예측을 수행합니다.
- **사례 기반 학습 (Instance-Based Learning)**: 모델을 따로 학습하지 않고, 새로운 데이터가 주어졌을 때 과거 데이터를 직접 활용하여 예측하는 방식입니다.
  - **예시**: K-최근접 이웃 알고리즘(K-Nearest Neighbors, KNN)
  - 데이터를 직접 비교하여 예측하며, 새로운 데이터 입력 시마다 과거 데이터를 참조합니다.

2.9 표층 학습 vs. 심층 학습

- 표층 학습(Shallow Learning)과 심층 학습(Deep Learning)은 학습 알고리즘의 깊이에 따른 구분입니다.

1. 표층 학습

- 표층 학습 알고리즘(Shallow Learning Algorithm)은 비교적 간단한 모델 구조를 가지며, 별도의 특성 추출 과정 없이 주어진 데이터로부터 모델의 파라미터를 학습합니다.
- **지도 학습**(Supervised Learning) 알고리즘은 대부분 표층 학습 알고리즘에 해당합니다. 예를 들어, 선형 회귀와 SVM(Support Vector Machine)이 대표적인 표층 학습 알고리즘입니다.
- 이 방식의 학습은 데이터로부터 고차원적인 특징을 추출하지 않고, 주어진 입력 데이터를 그대로 모델이 학습합니다.

2. 심층 학습

- 심층 학습(Deep Learning)은 신경망(Neural Network)의 일종으로, 여러 개의 은닉층(Hidden Layer)을 통해 데이터를 처리하는 방식입니다. 심층 학습에서는 입력 데이터의 특징을 계층적으로 학습하여 복잡한 패턴을 인식할 수 있습니다.
  - 예를 들어, 심층 신경망(Deep Neural Network, DNN)이나 합성곱 신경망(Convolutional Neural Network, CNN)은 심층 학습의 대표적인 예입니다.
- **심층 학습의 특징**:
  - 심층 학습은 다층 구조로 이루어져 있어, 저차원 데이터에서 고차원 데이터로 변환되는 과정을 거치며 점점 더 복잡한 특징을 학습합니다.
  - 이 방식은 이미지 처리, 음성 인식 등과 같은 복잡한 문제를 해결하는 데 매우 효과적입니다.

3. 표층 학습 vs. 심층 학습 비교

- **표층 학습**은 단순한 모델을 사용하여 적은 양의 데이터를 학습하는 데 적합하며, 빠르게 훈련할 수 있습니다. 그러나 데이터의 특성을 직접적으로 추출하지 않기 때문에 복잡한 문제에서는 한계가 있습니다.
- **심층 학습**은 대량의 데이터를 필요로 하지만, 매우 복잡한 패턴을 학습할 수 있어 고차원 문제에 강력한 성능을 발휘합니다. 그러나 계산 자원이 많이 필요하고, 학습 시간이 오래 걸릴 수 있습니다.