Stanford CS229 I Weighted Least Squares, Logistic regression, Newton's Method I 2022 I Lecture 3

---

1. 강의 소개

강의의 주요 흐름

Regression(회귀)에서 **Classification**(분류)으로의 전환, 이를 위한 **Logistic Regression**, 그리고 다양한 **Optimization 기법**을 다루며 이론적 배경과 실용적 고려사항을 심도 있게 설명함.

지난 복습

- Classification과 Regression 차이

회귀

연속적인 값을 예측하며, **선형 회귀**는 다음과 같이 잔차(residual)를 최소화:

$$\hat{\theta} = \arg\min_\theta \sum_{i=1}^{n} (y_i - \theta^\top x_i)^2$$

분류

데이터를 이산적인 카테고리로 나누는 문제.

예측값 y는 0 또는 1로 표현되며, 이를 확률적 관점에서 모델링:

$$h_\theta(x) = 1/1 + e^{-\theta^\top x}$$

데이터는 독립적이고 동일한 분포(IID)를 따른다는 가정 아래 최소 제곱 오차를 사용:

$$\hat{\theta} = \arg\min_\theta \sum_{i=1}^{n} (y_i - \theta^\top x_i)^2$$

- Classification의 개념과 필요성

분류 문제는 데이터가 연속적 값이 아닌 **카테고리 값**을 가질 때 등장. 예를 들어, 어떤 사진에서 고양이인지, 개인지, 돼지인지를 예측하려면 **분류 모델**이 필요함

Regression을 그대로 사용할 수도 있으나, 분류 문제에서는 다음과 같은 제약이 발생

- 예측값이 확률 범위(0~1)를 벗어날 수 있음.
- 데이터가 선형적으로 구분되지 않을 가능성이 높음.

- Machine Learning에서 Classification 문제의 사례
  - 이미지 속 동물 분류: 사진에서 고양이, 개, 말 등을 분류.
  - 스팸 필터링: 이메일이 스팸인지 아닌지 분류.
  - 의학 데이터 분석: 환자가 특정 질환에 걸릴 가능성 예측.
  - 음성 인식: 특정 단어를 말했는지 여부 분류.

---

2. 선형 회귀의 확률적 해석

선형 회귀를 확률적 관점에서 해석하는 이유

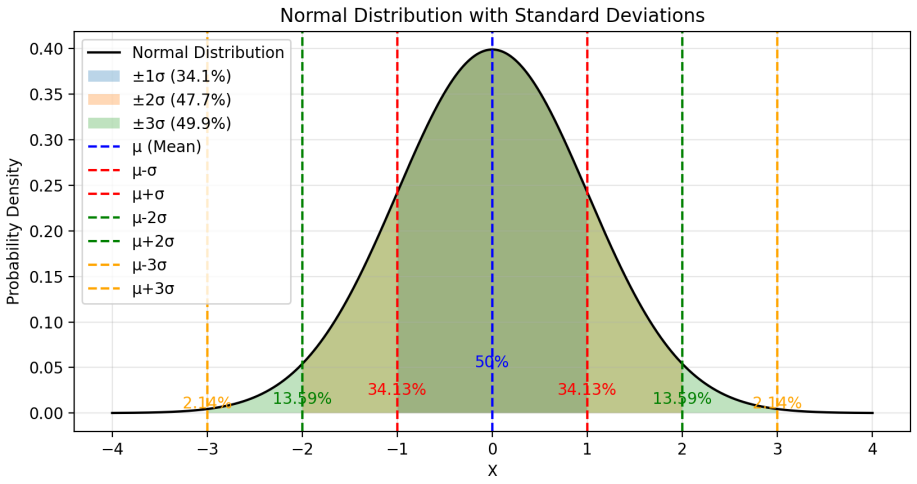
선형 회귀에서 데이터 y는 독립 변수 x와 모델 파라미터 θ의 선형 결합으로 표현됨:

$$y = \theta^\top x + \epsilon$$

여기서 ϵ은 오차 항(Noise)으로, 모델이 설명하지 못하는 데이터의 변동성을 나타냄.

이 오차 항 ϵ이 **가우시안 분포**를 따른다고 가정하여 선형 회귀를 확률론적으로 해석

$$\epsilon \sim N(0, \sigma^2)$$



▼ 가우시안 분포 그래프

```
import numpy as np
import matplotlib.pyplot as plt

# 데이터 생성
x = np.linspace(-10, 10, 500)  # 입력 값 범위
linear = x  # 선형 회귀의 출력
sigmoid = 1 / (1 + np.exp(-x))  # Sigmoid 함수의 출력

# 그래프 그리기
plt.figure(figsize=(10, 6))

# Linear Regression
plt.plot(x, linear, label="Linear Regression", linestyle="--", color="red"

# Sigmoid
plt.plot(x, sigmoid, label="Sigmoid Function (Logistic Regression)", color

# 그래프 꾸미기
plt.title("Linear Regression vs Sigmoid Function", fontsize=14)
plt.xlabel("Input (x)", fontsize=12)
plt.ylabel("Output", fontsize=12)
plt.axhline(0.5, color="gray", linestyle=":", label="Threshold (y=0.5)")
plt.axhline(0, color="black", linestyle="--", linewidth=0.7)
plt.axhline(1, color="black", linestyle="--", linewidth=0.7)
plt.legend(fontsize=12)
plt.grid(alpha=0.5)

# 그래프 출력
plt.show()
import matplotlib.pyplot as plt
import numpy as np

# Generate data for normal distribution
mu = 0  # Mean
sigma = 1  # Standard deviation
x = np.linspace(mu - 4*sigma, mu + 4*sigma, 1000)
y = (1 / (sigma * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((x - mu) / sigma)*

# Plot the normal distribution curve
plt.figure(figsize=(10, 6))
plt.plot(x, y, label="Normal Distribution", color='black')

# Shade regions and annotate
regions = {
    "1σ": (mu - sigma, mu + sigma, 0.3413),
    "2σ": (mu - 2*sigma, mu + 2*sigma, 0.4772),
    "3σ": (mu - 3*sigma, mu + 3*sigma, 0.4987)
```

```
}

for key, (left, right, value) in regions.items():
    plt.fill_between(x, y, where=(x >= left) & (x <= right), alpha=0.3, la

# Highlight μ and standard deviation lines
plt.axvline(mu, color='blue', linestyle='--', label="μ (Mean)")
plt.axvline(mu - sigma, color='red', linestyle='--', label="μ-σ")
plt.axvline(mu + sigma, color='red', linestyle='--', label="μ+σ")
plt.axvline(mu - 2*sigma, color='green', linestyle='--', label="μ-2σ")
plt.axvline(mu + 2*sigma, color='green', linestyle='--', label="μ+2σ")
plt.axvline(mu - 3*sigma, color='orange', linestyle='--', label="μ-3σ")
plt.axvline(mu + 3*sigma, color='orange', linestyle='--', label="μ+3σ")

# Annotate percentages
plt.text(mu, 0.05, "50%", horizontalalignment='center', color="blue")
plt.text(mu - sigma, 0.02, "34.13%", horizontalalignment='center', color="
plt.text(mu + sigma, 0.02, "34.13%", horizontalalignment='center', color="
plt.text(mu - 2*sigma, 0.01, "13.59%", horizontalalignment='center', color
plt.text(mu + 2*sigma, 0.01, "13.59%", horizontalalignment='center', color
plt.text(mu - 3*sigma, 0.005, "2.14%", horizontalalignment='center', color
plt.text(mu + 3*sigma, 0.005, "2.14%", horizontalalignment='center', color

# Title and labels
plt.title("Normal Distribution with Standard Deviations")
plt.xlabel("X")
plt.ylabel("Probability Density")
plt.legend(loc="upper left")
plt.grid(alpha=0.3)

plt.show()
```

이를 통해 회귀 문제를 최대 우도(Maximum Likelihood)의 문제로 변환할 수 있음. 최대 우도는 3번에서 마저 정리하겠음.

회귀식에서 오차를 최소화하는 과정

목적 함수 정의:

잔차(residual) $yi - \theta^\top x_i$의 제곱 합을 최소화함

$$\hat{\theta} = \arg\min_\theta \sum_{i=1}^{n} (y_i - \theta^\top x_i)^2$$

확률적 관점 도입:가우시안 분포를 기반으로 데이터 y의 조건부 확률을 정의

$$P(y|X,\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i-\theta^\top x_i)^2}{2\sigma^2}\right)$$

로그 가능도 함수 변환:로그를 취해 계산을 단순화하고, 최대화 문제를 최소화 문제로 변환:

$$\ell(\theta) = -\log P(y|X,\theta) = \sum_{i=1}^{n} \frac{(y_i-\theta^\top x_i)^2}{2\sigma^2} + \frac{n}{2}\log(2\pi\sigma^2)$$

가우시안 분포 도입 및 오정

Noise의 정의와 의미

잡음(Noise)란 데이터의 무작위적 변동으로, 모델이 설명하지 못하는 부분. 물리적 환경에서는 측정 오차나 외부 요인에 의해 발생.

가정 1. Noise의 평균이 0이라는 가정(편향 없음)

가정 2. 오차의 기댓값이 0으로 가정: $\mathbb{E}[\epsilon_i] = 0$

이는 모델이 편향되지 않고, 평균적으로 참값에 가까운 결과를 낼 것이라는 의미.

가정 3. Noise의 동일 분산 가정 (Variance Assumption, σ2)

오차가 모든 데이터에서 동일한 분산 σ^2을 가짐:

$$Var(\epsilon i) = \sigma^2$$

이는 데이터의 모든 샘플에서 오차의 크기가 비슷하다는 의미로, 계산의 단순화와 모델의 안정성을 제공.

가정 4. 독립성 및 IID(Independent and Identically Distributed) 가정

IID 가정은 데이터 샘플이 서로 독립적이며 동일한 확률 분포에서 생성되었다는 것을 의미.

- **Independent**: 데이터 샘플 간 독립성.
- **Identically Distributed**: 모든 샘플이 동일한 분포를 따름.

수학적 표현:

$P(y_1, y_2, \ldots, y_n|X,\theta) = \prod_{i=1}^{n} P(y_i|x_i,\theta)$

여기서$P(y_i \mid x_i, \theta)$는 동일한 분포에서 생성된 각 데이터 샘플의 조건부 확률.

독립성:

- 데이터 샘플 간에 상호작용이 없으므로, 한 샘플의 값이 다른 샘플의 결과에 영향을 미치지 않음.
- 예: 특정 이미지가 "고양이"로 분류되었다고 해서, 다른 이미지의 "고양이" 분류 결과가 변하지 않음.

동일 분포:

- 모든 데이터 샘플은 동일한 분포를 따르며, 이는 모델 학습의 일관성을 보장.
- 예: 동일한 카메라 환경에서 촬영된 이미지 데이터 세트는 같은 품질과 분포를 가질 가능성이 높음.

3. Likelihood와 Log-Likelihood

최대 우도 함수(Likelihood)의 정의

최대 우도 함수는 주어진 데이터 X와 y가 특정 모델 파라미터 θ 하에서 발생할 확률을 나타냄.

수학적으로,$P(y \mid X, \theta)$를 우도(Likelihood)라 하며 다음과 같이 정의:

$L(\theta) = P(y|X, \theta) = \prod_{i=1}^{n} P(y_i|x_i, \theta)$

여기서 $P(y_i \mid x_i, \theta)$는 각 데이터 샘플 i에 대한 조건부 확률.

최대 우도 추정(Maximum Likelihood Estimation, MLE)

모델이 관찰된 데이터를 가장 잘 설명하는 파라미터 θ를 찾는 것.

최적화 문제로 표현:

$\hat{\theta} = \arg\max_{\theta} L(\theta)$

우도 함수는 모델의 적합도(Fit)를 확률적으로 평가하는 도구로 사용됨.

L(θ)가 높을수록 데이터 (X,y)를 설명하는 데 있어 해당 모델이 더 적합하다는 의미.

예를 들어, 선형 회귀에서 P(y ㅣ X,θ)는 각 샘플 i의 오차가 가우시안 분포를 따른다고 가정:

$P(y_i \mid x_i, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \theta^\top x_i)^2}{2\sigma^2}\right)$

전체 데이터의 우도 함수는 다음과 같음:

$L(\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \theta^\top x_i)^2}{2\sigma^2}\right)$

Log-Likelihood로 변환하여 계산 간소화

우도 함수는 곱셈 연산이 포함되어 계산이 복잡하므로, 로그를 취해 로그 우도 함수로 변환

$\ell(\theta) = logL(\theta) = \sum_{i=1}^{n} logP(y_i \mid x_i, \theta)$

선형 회귀에서 로그 우도 함수는 다음과 같이 계산됨:

$\ell(\theta) = \sum_{i=1}^{n} \left[-\frac{1}{2}\log(2\pi\sigma^2) - \frac{(y_i - \theta^\top x_i)^2}{2\sigma^2}\right]$

로그 변환의 장점:

곱셈 연산을 덧셈으로 단순화.

계산적 안정성 증가(특히 우도 함수 값이 매우 작을 때 유용).

- 최소 제곱법(Least Squares)와 Log-Likelihood의 연결성
  - **최소 제곱법**:
    - 잔차의 제곱 합을 최소화하는 선형 회귀의 목적 함수:

      $J(\theta) = \sum_{i=1}^{n} (y_i - \theta^\top x_i)^2$
  - **로그 우도 함수와 최소 제곱법의 관계**:
    - 로그 우도 함수 ℓ(θ)를 최대화하는 것은 최소 제곱 오차를 최소화하는 것과 동일:

      $J(\theta) = -2\sigma^2 \cdot \ell(\theta) + C$
      여기서 C는 상수 항으로 최적화에 영향을 미치지 않음.
    - 결과적으로, 선형 회귀에서 최소 제곱법은 확률론적 관점에서 로그 우도 함수의 최대화를 기반으로 정당화될 수 있음.

---

4. Logistic Regression의 도입

- Logistic Regression의 역할과 중요성
  - **Logistic Regression**은 데이터를 이산적인 카테고리(예: 0 또는 1)로 분류하는 데 사용되는 기본적인 머신러닝 모델임.
    - 역할:
      - P(y=1 ∣ x)P(y = 1 | x)P(y=1 ∣ x)와 P(y=0 ∣ x)의 확률을 모델링하여 데이터를 분류.

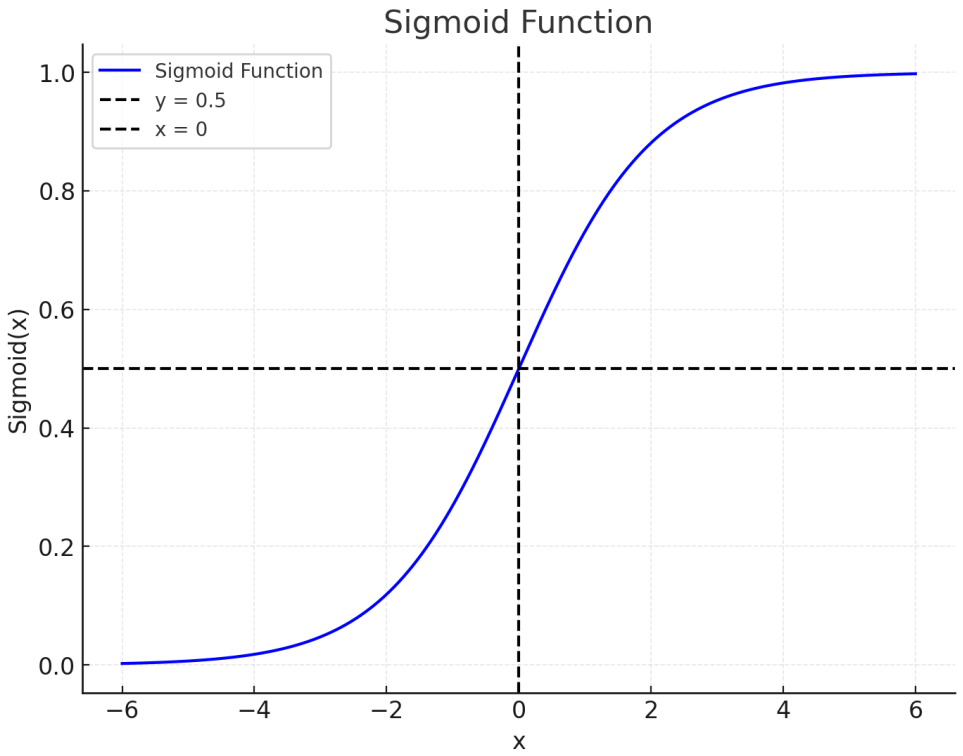
        P(y=0 ∣ x)P(y = 0 | x)
      - 예측값을 확률값으로 변환하므로, 모델 결과를 직관적으로 해석 가능.
    - **중요성**:
      - 선형 회귀와 달리 분류(Classification) 문제에 특화됨.
      - 시그모이드 함수(Sigmoid Function)를 통해 예측값을 0과 1 사이로 매핑하여 분류 문제에 적합한 확률적 해석 제공
- Sigmoid 함수: 정의, 특성 및 분류 문제에서의 활용
  1. **정의**

     Sigmoid 함수는 다음과 같이 정의됨:

     $\sigma(z) = \frac{1}{1+e^{-z}}$

     Logistic Regression에서는$z = \theta^\top x$로 대체되어 다음과 같이 표현:

     $h_\theta(x) = \frac{1}{1+e^{-\theta^\top x}}$
  2. **특성**:
     - **출력값 범위**: $\sigma(z) \in (0, 1)$, 즉 확률로 해석 가능.
     - **모양**:$z \to -\infty$일 때 σ(z)→0, z→+∞일 때 σ(z)→1.
     - **대칭성**: σ(−z)=1−σ(z).



▼ 코드

```
import matplotlib.pyplot as plt
import numpy as np

# Generate data for normal distribution
mu = 0  # Mean
sigma = 1  # Standard deviation
x = np.linspace(mu - 4*sigma, mu + 4*sigma, 1000)
y = (1 / (sigma * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((x - mu) / sigm

# Plot the normal distribution curve
plt.figure(figsize=(10, 6))
plt.plot(x, y, label="Normal Distribution", color='black')

# Shade regions and annotate
regions = {
    "1σ": (mu - sigma, mu + sigma, 0.3413),
    "2σ": (mu - 2*sigma, mu + 2*sigma, 0.4772),
    "3σ": (mu - 3*sigma, mu + 3*sigma, 0.4987)
}

for key, (left, right, value) in regions.items():
    plt.fill_between(x, y, where=(x >= left) & (x <= right), alpha=0.3,

# Highlight μ and standard deviation lines
plt.axvline(mu, color='blue', linestyle='--', label="μ (Mean)")
```

```
plt.axvline(mu - sigma, color='red', linestyle='--', label="μ-σ")
plt.axvline(mu + sigma, color='red', linestyle='--', label="μ+σ")
plt.axvline(mu - 2*sigma, color='green', linestyle='--', label="μ-2σ")
plt.axvline(mu + 2*sigma, color='green', linestyle='--', label="μ+2σ")
plt.axvline(mu - 3*sigma, color='orange', linestyle='--', label="μ-3σ")
plt.axvline(mu + 3*sigma, color='orange', linestyle='--', label="μ+3σ")

# Annotate percentages
plt.text(mu, 0.05, "50%", horizontalalignment='center', color="blue")
plt.text(mu - sigma, 0.02, "34.13%", horizontalalignment='center', colo
plt.text(mu + sigma, 0.02, "34.13%", horizontalalignment='center', colo
plt.text(mu - 2*sigma, 0.01, "13.59%", horizontalalignment='center', co
plt.text(mu + 2*sigma, 0.01, "13.59%", horizontalalignment='center', co
plt.text(mu - 3*sigma, 0.005, "2.14%", horizontalalignment='center', co
plt.text(mu + 3*sigma, 0.005, "2.14%", horizontalalignment='center', co

# Title and labels
plt.title("Normal Distribution with Standard Deviations")
plt.xlabel("X")
plt.ylabel("Probability Density")
plt.legend(loc="upper left")
plt.grid(alpha=0.3)

plt.show()
```

1. **분류 문제에서의 활용**:
   - $h_\theta(x)$는$P(y = 1 \mid x)$를 나타내며, 0.5를 기준으로 분류 수행:

$$\hat{y} = \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Logistic Regression과 Linear Regression의 차이

Linear Regression을 Classification에 사용하기 어려운 이유

: 선형 회귀에서는 출력값이 확률 범위 [0,1]를 벗어날 수 있음:

$y = \theta^\top x$
예:
$\theta^\top x > 1$또는 $\theta^\top x < 0$일 때 y가 확률로 해석되지 않음.

- 선형 회귀는 데이터가 선형적으로 구분되지 않을 때 분류 성능이 저하됨.
- 예측값이 극단적으로 커질 때 최적화 과정이 불안정해질 가능성이 있음.

Sigmoid 함수의 매끄러운 전환 특성 및 최적화 가능성

- Sigmoid 함수는 분류 경계를 매끄럽게 설정하며, 출력값을 확률로 변환:

$h_\theta(x) = \frac{1}{1+e^{-\theta^\top x}}$

- 경계의 매끄러운 특성:
  - σ(z)는 z=0에서 급격히 변동하며, z→±∞로 갈수록 안정화됨.
  - 이는 확률적 해석과 분류 경계 설정을 용이하게 함.

  Sigmoid 함수의 미분값이 연속적이고 적당히 작아, 경사 하강법(Gradient Descent)과 같은 최적화 기법에 적합

---

5. Optimization 기법 1: Gradient Descent

Gradient Descent의 기본 원리

목적 함수 J(θ)의 최소값을 찾기 위해, 매 반복 단계마다 기울기(gradient)를 따라 파라미터를 업데이트:

$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla J(\theta)$

η는 학습률(Learning Rate).

- Stochastic Gradient Descent (SGD)
- Batch Gradient Descent

특징	Stochastic Gradient Descent (SGD)	Batch Gradient Descent
데이터 처리 효율성	데이터를 한 번에 하나씩 처리, 대규모 데이터셋에 효율적	데이터 전체를 한 번에 처리
계산 효율성	실시간 업데이트에 빠르지만 더 많은 반복(iteration)이 필요할 수 있음	대규모 데이터셋에서는 계산 비용이 매우 큼
메모리 사용	단일 데이터 포인트만 처리하므로 메모리 사용량이 낮음	데이터 전체를 메모리에 적재해야 하므로 메모리 사용량이 큼
수렴 안정성	수렴 과정에서 진동이 발생할 수 있어 안정성이 낮음	결정론적 업데이트로 안정적으로 수렴
병렬 처리 적합성	독립적인 업데이트가 가능하므로 병렬 처리에 매우 적합	전체 데이터셋에 의존하므로 병렬 처리에 적합하지 않음

---

6. Optimization 기법 2: Newton's Method

- Newton's Method의 수학적 원리
  - 2차 미분을 사용한 최적화

    $\theta^{(t+1)} = \theta(t) - H^{-1}\nabla \ell(\theta)$

H는 목적 함수의 2차 미분 행렬:

$H = \nabla^2 J(\theta)$

Newton's Method: 장점	- 높은 정확도와 빠른 수렴 속도.- 각 반복에서 최적화 경로가 크게 줄어듦.

Newton's Method: 한계	- 계산 비용이 O(n^2)로 매우 큼. - 메모리 소모가 커서 대규모 문제에는 적합하지 않음.

Noise의 영향	- 모델의 오차 항(ϵ)은 데이터의 불확실성을 반영 노이즈 수준이 높을수록 (σ^2증가) 모델 신뢰도가 낮아지고 예측이 불안정해짐.

Noise에 따른 모델링 전략	- 낮은 노이즈(σ^2 작음): 모델이 정확한 패턴을 학습할 수 있음.- - 높은 노이즈(σ^2 큼): 규제를 적용하거나, 확률적 모델로 데이터의 불확실성을 반영해야 함.

.

1. **Classification과 Logistic Regression의 차별화**

   Linear Regression의 한계

   - 출력값이 확률의 범위를 벗어날 수 있음.
   - 선형 회귀는 데이터가 선형적으로 구분되지 않을 때 분류 성능 저하.

   Logistic Regression의 적합성

   - Sigmoid 함수를 사용하여 출력값을 [0,1]로 제한.
   - 출력값은 확률로 해석 가능:
     P(y=1 ∣ x)=hθ(x)
2. **Link Function의 개념 및 응용**
   - Link Function의 정의 및 Logistic Regression에서의 역할

     모델의 출력값을 확률 범위로 변환하는 함수. Logistic Regression에서는 Sigmoid 함수 사용:

$$g^{-1}(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

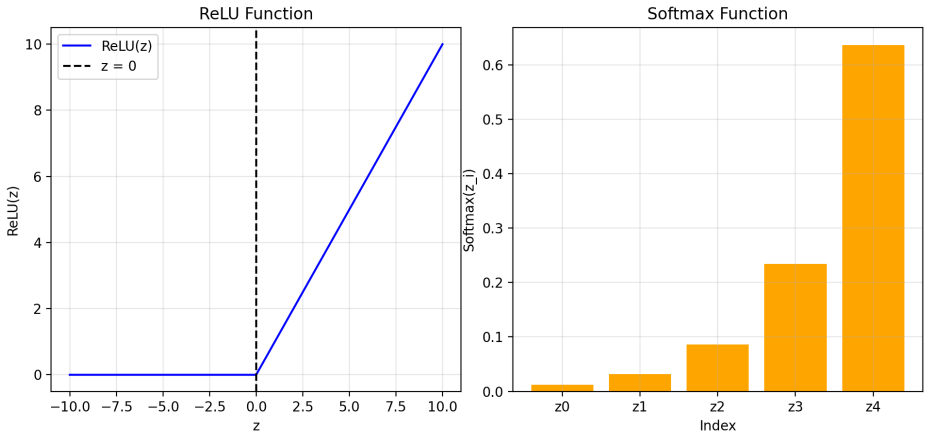
   Sigmoid 외의 다른 Link Functions

   - **ReLU (Rectified Linear Unit)**:

     $ReLU(z) = max(0, z)$
     비선형 문제에서 자주 사용됨.
   - **Softmax**:
     Softmax(zi)=∑jezjezi

다중 클래스 분류에 사용.

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$



▼ 코드

```
import numpy as np
import matplotlib.pyplot as plt

# Define ReLU function
def relu(z):
    return np.maximum(0, z)

# Define Softmax function
def softmax(z):
    exp_z = np.exp(z - np.max(z))  # for numerical stability
    return exp_z / np.sum(exp_z)

# Generate data for ReLU
z_relu = np.linspace(-10, 10, 500)
y_relu = relu(z_relu)

# Generate data for Softmax
z_softmax = np.array([1.0, 2.0, 3.0, 4.0, 5.0])  # example inputs
y_softmax = softmax(z_softmax)

# Plot ReLU function
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(z_relu, y_relu, color='blue', label="ReLU(z)")
plt.title("ReLU Function")
plt.xlabel("z")
plt.ylabel("ReLU(z)")
plt.grid(alpha=0.3)
plt.axvline(0, color='black', linestyle='--', label="z = 0")
```

```
plt.legend()

# Plot Softmax function
plt.subplot(1, 2, 2)
plt.bar(range(len(z_softmax)), y_softmax, color='orange', tick_label=[f"z{
plt.title("Softmax Function")
plt.xlabel("Index")
plt.ylabel("Softmax(z_i)")
plt.grid(alpha=0.3)

plt.tight_layout()
plt.show()
```

---

7. Optimization 기법의 비교

기법	장점	단점	사용 사례
Gradient Descent	높은 정확도	계산 비용이 큼	중소규모 데이터 세트
SGD	빠른 수렴, 메모리 효율적	최적값 근처에서 불안정	대규모 데이터, 딥러닝 모델
Newton's Method	빠른 수렴, 높은 정확도	고차원 데이터에서 계산 비용과 메모리 소모 큼	소규모 고정밀 작업

결론

- **Logistic Regression**:
  - Classification 문제에서 Linear Regression의 한계를 극복하고 확률적 접근을 통해 분류 성능 향상.
- **Optimization 기법**:
  - SGD는 대규모 데이터와 병렬 처리 환경에서 가장 실용적.
  - Newton's Method는 소규모 데이터에서 높은 정확도를 제공.
- 강의는 Regression, Classification, Optimization 간의 연결성을 설명하며, 확률적 모델과 머신러닝 최적화의 기본 원리를 제시함.