핵심 머신러닝 - 3

생성자	㉦ 재환 김
태그	머신러닝

- **기본 알고리즘**: 이 장에서는 다섯 가지 핵심 머신 러닝 알고리즘을 소개합니다. 이들은 효율적일 뿐만 아니라 최신 알고리즘의 주요 구성 요소로도 널리 활용됩니다.
- **3.1 선형 회귀 (Linear Regression)**:
  - 선형 회귀는 가장 널리 사용되는 회귀 알고리즘 중 하나입니다.
  - 이 방법은 주로 입력 데이터의 특정 값들을 선형 결합(linear combination)하여 모델을 구축합니다.
- **3.1.1 문제 정의**:
  - 여기서 문제는 레이블이 달린 예제 집합 (x_i, y_i)가 있다고 가정합니다.
  - N은 이 집합의 크기(예제 수)를 나타냅니다.
  - 각 예제에 대해 D 차원 특성 벡터(feature vector) x_i가 주어지며, 벡터의 성분은 x_{ij}로 표현됩니다.
  - 목표(target)는 y_i로 정의됩니다.

선형 회귀 (Linear Regression)

1.1 정의

선형 회귀는 주어진 데이터를 바탕으로 예측 모델을 만드는 알고리즘 중 하나입니다.
입력된 데이터의 특정 값을 선택하여, 그 값들을 선형 결합(linear combination)으로 표현하는 모델을 생성합니다.
즉, 독립 변수
x들이 종속 변수 y와 어떻게 관계되는지를 선형 방정식을 통해 표현하는 알고리즘입니다.

- **선형 회귀 방정식**:

  $y = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$
  여기서
  $x_1, x_2, \ldots, x_n$는 입력 변수들이고, $w_1, w_2, \ldots, w_n$는 각 변수에 곱해지는 가중치(weight)입니다. b는 편향(bias) 또는 절편(intercept)이라고 부릅니다.

이 방정식에서 알고리즘은 데이터에서 w와 b 값을 찾아내는 것을 목표로 합니다. 이렇게 찾은 값들이 새로운 데이터에 대해 예측을 가능하게 합니다.

1.2 목적

선형 회귀의 목표는 주어진 입력 변수들과 출력 변수 간의 관계를 최적화된 직선 형태로 나타내는 것입니다. 이를 통해 예측 또는 분석할 수 있게 됩니다.

예를 들어, 특정 날의 기온을 예측한다고 할 때, 선형 회귀는 그 날의 과거 기온 데이터를 바탕으로 새로운 날의 기온을 예측하는 방식을 따릅니다.

2. 문제 정의 (3.1.1에서 설명한 부분)

2.1 데이터 구조

선형 회귀 문제에서 우리는 레이블이 달린 예제 집합을 가지고 있습니다. 이 집합은 다음과 같은 구조로 이루어집니다:

- **입력 데이터**: (x_i, y_i)예를 들어, 여러 날의 기온을 예측하는 모델을 만들 때, x_i는 그 날의 습도, 풍속 등의 독립 변수들이고, y_i는 기온이 될 수 있습니다.
  - x_i: **입력 벡터** 혹은 **특성 벡터**로, 각 x_i는 여러 독립 변수(특성)들로 이루어져 있습니다.
  - y_i: **레이블** 또는 **목표 값**으로, x_i에 대응되는 출력 값입니다.

2.2 예제

- N: 예제의 개수, 즉 데이터 셋에 포함된 레코드(행)의 수입니다.
- D: 입력 벡터 x_i의 차원 수, 즉 독립 변수(특성)의 개수입니다.
  예를 들어, 기온 예측 문제에서 습도, 풍속, 구름량 등의 데이터가 있을 때, 이 데이터의 특성 개수
  D는 3이 될 수 있습니다.

2.3 수학적 표현

각 예제에 대해,

- **특성 벡터** $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$
  - 이 벡터는 D차원으로 구성되어 있으며, 예를 들어 습도, 풍속, 기압과 같은 요소들을 나타냅니다.
- **목표 값** y_i: 예제의 레이블로, 예를 들어 특정 날의 실제 기온을 의미합니다.

3. 선형 회귀의 학습 과정

3.1 손실 함수 (Loss Function)

모델이 예측한 값과 실제 값 사이의 차이를 최소화하기 위해 사용하는 함수입니다. 일반적으로 선형 회귀에서 사용하는 손실 함수는 평균 제곱 오차(MSE, Mean Squared Error)입니다.

- **평균 제곱 오차**:

  $$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

  여기서 y_i는 실제 값이고, y^i는 예측된 값입니다.

모델은 이 오차를 최소화하는 방향으로 가중치 w와 절편 b를 학습하게 됩니다.

3.2 경사 하강법 (Gradient Descent)

가중치 w와 절편 b를 최적화하는 방법 중 가장 널리 사용되는 방법이 경사 하강법입니다. 경사 하강법은 손실 함수의 기울기를 따라 가중치를 업데이트하면서 오차를 줄이는 방식입니다.

4. 선형 회귀의 적용 사례

- **주택 가격 예측**: 여러 특성(집의 크기, 방의 개수 등)을 이용해 집의 가격을 예측하는 문제에서 자주 사용됩니다.
- **기온 예측**: 과거 기상 데이터를 바탕으로 미래 기온을 예측하는 데 사용될 수 있습니다.

5. 선형 회귀 방정식

5.1 선형 회귀 모델

이미지 상단의 내용에서, 선형 회귀 모델은 다음과 같이 표현됩니다:

$f_{\mathbf{w},b}(x) = \mathbf{w}x + b$

여기서:

- $\mathbf{w}$: 파라미터를 나타내는 벡터입니다. 각 입력 값 x에 대한 가중치입니다.
- b: 절편(bias)입니다. 모델의 예측치에 상수로 더해지는 값입니다.

이 식은 입력 x가 주어졌을 때, 모델이 그 입력에 대해 예측하는 값을 나타냅니다. 모델이 하는 일은 파라미터 $\mathbf{w}$와 b를 학습하여, 주어진 데이터를 가장 잘 설명하는 선형 관계를 찾는 것입니다.

5.2 그래프 설명 (그림 3.1)

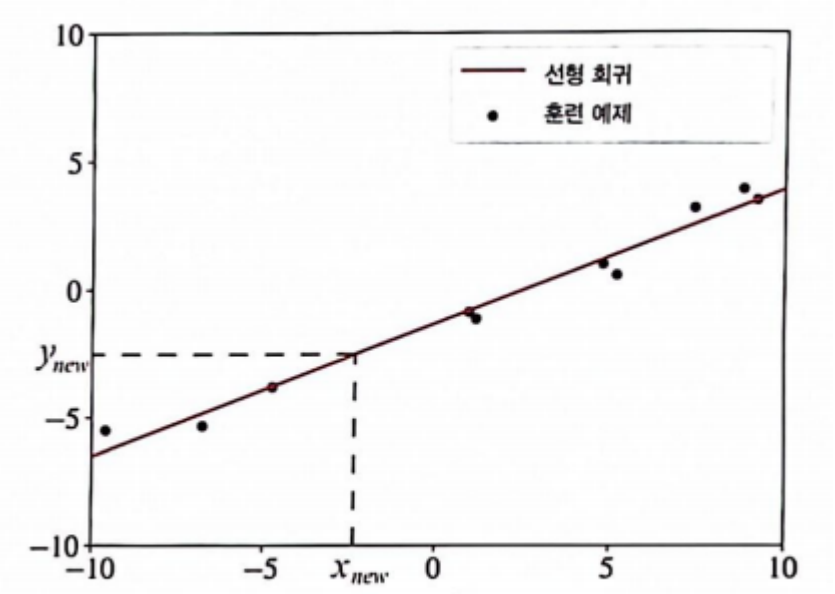

그림 3.1 1차원 예제에 대한 선형 회귀

오른쪽에 있는 그래프는 선형 회귀 모델의 예시를 보여줍니다.

- **빨간색 직선**은 학습된 선형 회귀 모델을 나타냅니다.
- **검은 점들**은 학습 데이터 포인트입니다.

이 직선은 데이터 포인트들의 전반적인 경향을 나타내며, 각 데이터 포인트에 대한 모델의 예측 정확도를 시각적으로 보여줍니다. 직선에 가까이 위치한 점들은 모델이 해당 데이터를 정확하게 예측하고 있음을 의미합니다.

6. 해결 방법 (3.1.2 해결 방법)

6.1 손실 함수 (Loss Function)

이미지의 오른쪽에 나오는 수식 (3.2)은 선형 회귀 모델에서 사용하는 손실 함수로, 평균 제곱 오차(MSE)를 나타냅니다.

$\frac{1}{N}\sum_{i=1}^{N}\left(f_{\mathbf{w},b}(x_i)-y_i\right)^2$

- N: 데이터 포인트의 개수.

- $f_{\mathbf{w},b}(x_i)$: 모델이 예측한 값.
- y_i: 실제 값.

이 수식은 예측값과 실제값의 차이를 제곱한 뒤 평균을 구하는 방식으로, 모델의 예측 정확도를 측정합니다. 손실 함수의 값이 작을수록 모델의 예측이 더 정확하다는 것을 의미합니다.

6.2 최적화 문제

선형 회귀의 목표는 위 손실 함수를 최소화하는 $\mathbf{w}$와 b를 찾는 것입니다. 이 과정에서 경사하강법(Gradient Descent)과 같은 최적화 기법을 사용하여 파라미터를 조정합니다.

손실 함수를 최소화한다는 것은 실제 데이터와 예측값 사이의 차이를 최소화하는 파라미터를 찾는 것을 의미합니다. 이때 손실 함수는 모델의 목표 함수(objective function)라고도 불리며, 이 값을 최소화하는 것이 모델 학습의 최종 목표가 됩니다.

7. 오버피팅(Overfitting)의 문제

오버피팅은 모델이 학습 데이터에 과도하게 적응하여 새로운 데이터에 대한 예측 성능이 저하되는 현상입니다. 이는 학습 데이터에 대한 정확도는 높지만, 일반화 능력이 부족해지는 것을 의미합니다.

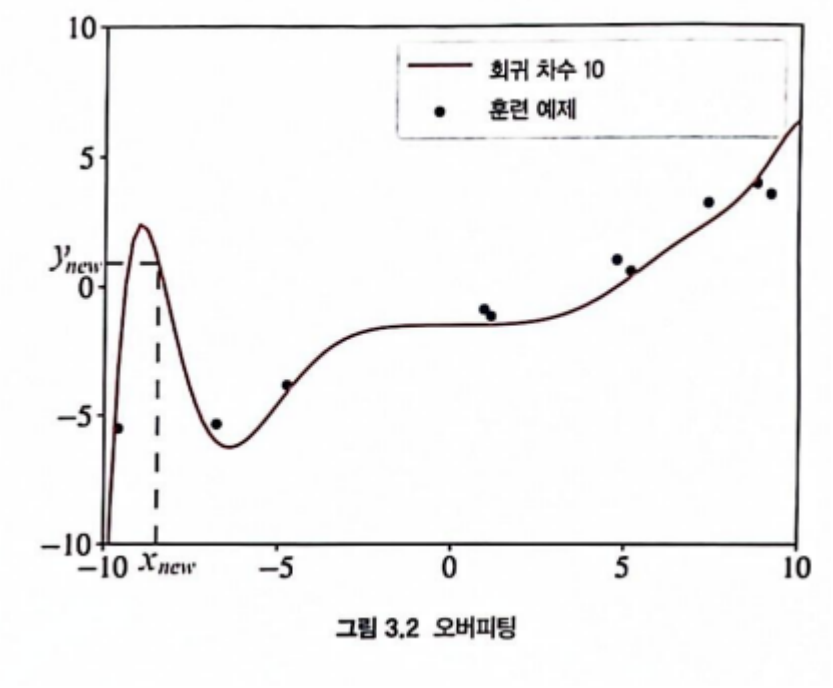

그림 3.2 오버피팅

- **그림 3.2**는 오버피팅의 예시를 보여줍니다.
- 학습 데이터에 지나치게 정확하게 맞춰진 곡선으로 인해 중간에 급격한 변화가 나타납니다.

- 모델이 학습 데이터 포인트 근처에서 매우 정교하게 적합되었지만, 이로 인해 새로운 데이터에 대해서는 예측이 불안정해질 가능성이 높습니다.

오버피팅의 대표적인 특징은 학습 데이터에 대한 성능은 우수하지만, 일반화된 새로운 데이터에 대한 성능은 저하되는 것입니다.

8. 손실 함수와 해결 방법

이미지에서는 손실 함수의 개념을 활용하여 모델의 학습 방식을 설명합니다. 여기서 중요한 개념은 **경험적 위험(empirical risk)**과 **제곱 손실(quadratic loss)**입니다.

8.1 손실 함수의 역할

- 모델에서 손실 함수를 사용하는 이유는 모델의 예측값과 실제값 간의 차이를 최소화하기 위함입니다.
- 손실 함수는 모델이 데이터에 얼마나 잘 적합되는지를 평가하는 기준입니다.
- 손실 함수 값이 작을수록 모델이 데이터를 더 잘 학습했다고 판단할 수 있습니다.

8.2 새로운 알고리즘의 필요성

기존 알고리즘보다 더 나은 모델을 만들기 위해 새로운 알고리즘을 개발하는 이유는 다음과 같습니다:

1. 실제 주어진 문제에 더 적합한 알고리즘 개발
2. 기존 모델의 한계를 극복하고, 새로운 데이터를 더 잘 예측할 수 있는 알고리즘 필요

9. 모델의 일반화와 매끄러운 함수

모델의 오버피팅을 방지하기 위해서는 매끄러운(smooth) 함수로 모델을 학습하는 것이 중요합니다. 이는 모델이 학습 데이터의 개별적인 변동성에 과도하게 반응하지 않도록 하는 것입니다. 예를 들어, **Adrien-Marie Legendre**가 제안한 방법처럼, 매끄러운 곡선을 사용하여 모델의 일반화 능력을 향상시키는 방법을 채택할 수 있습니다.

- **매끄러운 함수**: 데이터 포인트들 사이에서 급격한 변화를 방지하고, 곡선이 부드럽게 이어지도록 만드는 방법입니다.
- 매끄러운 함수를 사용하면, 모델이 학습 데이터의 변동성을 과도하게 반영하지 않고, 더 일반적인 패턴을 학습할 수 있습니다.

로지스틱 회귀 (Logistic Regression)

1.1 로지스틱 회귀란?

로지스틱 회귀는 선형 회귀와 달리 연속적인 수치를 예측하지 않고, **범주형 데이터를 분류**하는 알고리즘입니다. 주로 결과가 특정 클래스(예: 0 또는 1)에 속하는지를 예측하는 문제에 사용됩니다. 이 알고리즘은 선형 모델 대신 **로지스틱 함수(시그모이드 함수)**를 사용하여 예측값을 0과 1 사이의 확률로 변환합니다.

로지스틱 회귀는 **종속 변수**(결과)가 이진(binary)인 경우, 즉 종속 변수가 두 개의 클래스로 나뉘는 문제에서 특히 유용합니다.

1.2 시그모이드 함수 (Sigmoid Function)

로지스틱 회귀에서 사용하는 시그모이드 함수는 다음과 같이 정의됩니다:

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

- 이 함수는 **S자형 곡선**을 그리며, z값이 무한대일 때 함수의 값은 1에 가까워지고, z 값이 음의 무한대일 때 함수의 값은 0에 가까워집니다. 따라서, 이 함수는 모든 값을 0과 1 사이의 값으로 변환하는 역할을 합니다.

1.3 로지스틱 회귀의 예측 함수

로지스틱 회귀 모델은 아래와 같은 형태로 표현됩니다:

$$f_{\mathbf{w},b}(x) = \frac{1}{1+e^{-(\mathbf{w}x+b)}}$$

- 여기서 $\mathbf{w}x + b$는 선형 결합이며, 이 값을 시그모이드 함수에 넣어 0과 1 사이의 값으로 변환하게 됩니다.
- 이 값이 0.5 이상이면 1로, 0.5 미만이면 0으로 분류하게 됩니다. 즉, 해당 데이터 포인트가 특정 클래스에 속할 확률을 나타냅니다.

2. 문제 정의

로지스틱 회귀의 목표는 주어진 입력 x에 대해 결과 y가 특정 클래스에 속할 확률을 예측하는 것입니다. 이를 통해 모델은 x가 주어졌을 때, 해당 데이터 포인트가 클래스 1 또는 0에 속할 확률을 예측합니다.

2.1 로지스틱 회귀와 선형 회귀의 차이

로지스틱 회귀는 선형 회귀와 달리 예측값이 0과 1 사이의 확률로 제한됩니다. 이는 로지스틱 회귀가 이진 분류 문제에 적합하게 만듭니다. 선형 회귀가 연속적인 값을 예측하는 반면, 로지스틱 회귀는 특정 클래스에 속할 확률을 예측합니다.

- **선형 회귀**는 예측값이 실수 형태로 산출됩니다.

- **로지스틱 회귀**는 예측값이 0과 1 사이의 확률로 나오며, 이 확률을 기준으로 분류가 이루어집니다.

2.2 수학적 표현

로지스틱 회귀 모델의 수학적 형태는 다음과 같은 식으로 정의됩니다:

$f_{\mathbf{w},b}(x) = \frac{1}{1+e^{-(\mathbf{w}x+b)}}$

이 식에서:

- $\mathbf{w}x + b$ 는 선형 방정식이며,
- 이 값은 시그모이드 함수를 통해 0과 1 사이의 값으로 변환됩니다.

이를 통해 모델은 **확률 값**을 산출합니다. 이 확률이 0.5 이상이면 1로, 0.5 미만이면 0으로 예측합니다.

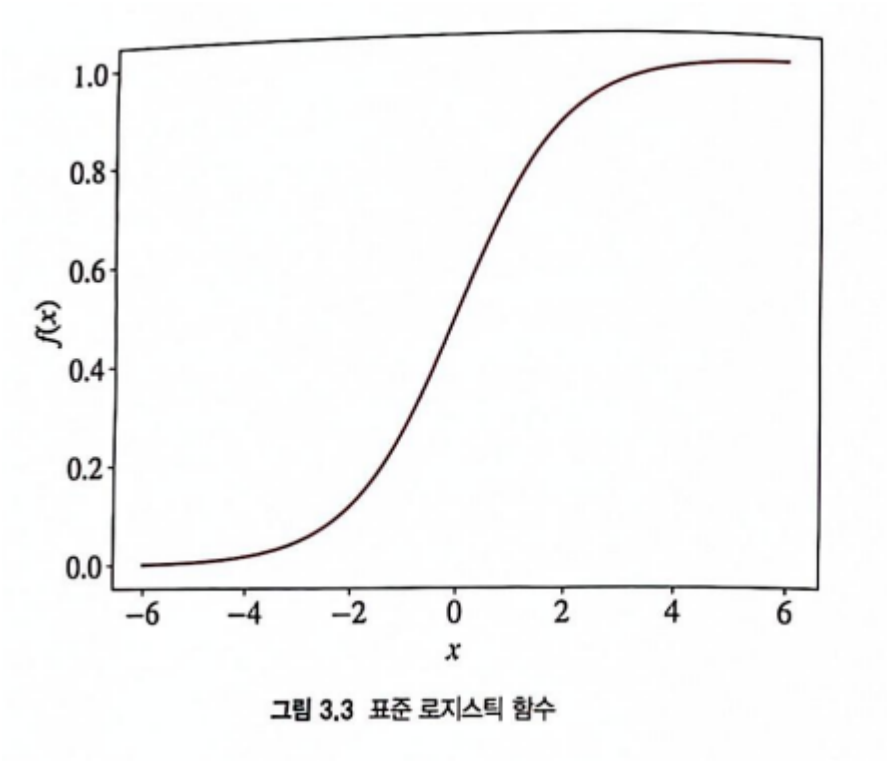

그림 3.3 표준 로지스틱 함수

3 로지스틱 회귀에서의 해결 방법

로지스틱 회귀는 선형 회귀와는 달리 평균 제곱 오차(MSE)를 최소화하는 것이 아니라, 가능도 함수(Likelihood Function)를 최대화하는 방식으로 파라미터 $\mathbf{w}$와 b를 최적화합니다.

3.1 가능도 함수 (Likelihood Function)

이미지에 나타난 로지스틱 회귀에서의 **가능도 함수**는 다음과 같은 수식으로 표현됩니다:

--

$$L_{\mathbf{w},b} = \prod_{i=1}^{N} f_{\mathbf{w},b}(x_i)^{y_i}(1 - f_{\mathbf{w},b}(x_i))^{1-y_i}$$

이 식에서:

- N은 데이터 포인트의 수.
- $f_{\mathbf{w},b}(x_i)$는 로지스틱 회귀 모델의 예측값입니다.
  - 이 값은 시그모이드 함수로 계산되며, 각 데이터 포인트가 특정 클래스에 속할 확률을 나타냅니다.
- y_i는 실제 레이블로, $y_i = 1$이면 해당 데이터 포인트가 클래스 1에 속하고, $y_i = 0$이면 클래스 0에 속합니다.

이 가능도 함수는 모델이 예측한 값과 실제 값 사이의 확률을 바탕으로 각 데이터 포인트에 대한 모델의 성능을 평가합니다. 목표는 이 가능도 함수를 최대화하도록 파라미터를 조정하는 것입니다.

3.2 로그 가능도 함수 (Log-Likelihood Function)

가능도 함수는 곱셈 형태로 되어 있어 계산이 복잡해질 수 있습니다. 이 문제를 해결하기 위해 **로그 가능도 함수(Log-Likelihood Function)**를 사용하여 최적화 과정을 간소화합니다. 로그 가능도 함수는 다음과 같은 수식으로 표현됩니다:

$$\log L_{\mathbf{w},b} = \sum_{i=1}^{N} \left(y_i \log f_{\mathbf{w},b}(x_i) + (1 - y_i) \log(1 - f_{\mathbf{w},b}(x_i)) \right)$$

로그 함수의 특성으로 인해 계산이 간소화되며, 로그 가능도 함수의 최대화가 최적화 문제의 핵심 목표가 됩니다.

4. 해결 과정

4.1 최대 가능도 추정법 (Maximum Likelihood Estimation, MLE)

로지스틱 회귀에서 파라미터 w와 b를 추정하기 위해 최대 가능도 추정법(MLE)을 사용합니다. 이 방법은 주어진 데이터에서 관찰된 결과를 가장 잘 설명할 수 있는 파라미터 값을 찾습니다.

최대 가능도 추정법을 통해 파라미터를 추정하려면 로그 가능도 함수를 최대화해야 합니다. 이를 위해 경사 하강법(Gradient Descent)과 같은 최적화 알고리즘으로 파라미터를 업데이트합니다.

4.2 최적화 목표

최종 목표는 로그 가능도 함수를 최대화하는 것입니다. 이 과정에서 파라미터 w와 절편 b를 최적화하여 모델이 주어진 데이터를 가장 잘 설명할 수 있도록 합니다.

로그 가능도 함수를 최대화함으로써 모델은 새로운 데이터에 대해 더 정확한 예측을 할 수 있게 됩니다. 이는 선형 회귀에서 손실 함수를 최소화하는 것과는 대조적으로, **가능도를 최대화**하는 접근 방식입니다.

결정 트리 학습

1.1 결정 트리란?

결정 트리(Decision Tree)는 데이터를 분류하는 데 사용되는 비순환 그래프(acyclic graph) 형태의 알고리즘입니다. 각 노드에서는 특정 **특성(feature)**을 기준으로 데이터를 여러 갈래로 분할합니다. 이러한 분할 과정을 거쳐 최종적으로 데이터를 하나의 **클래스(class)**로 분류합니다.

- 예를 들어, 특정 특성의 값이 임계값(threshold)을 기준으로 데이터를 둘로 나누며, 이후 노드에서도 같은 방식으로 분할이 계속됩니다.
- 이 과정에서 **중간 노드**는 데이터를 분할하는 기준이 되고, **말단 노드(leaf node)**에서 최종 분류가 이루어집니다.

1.2 결정 트리의 활용

결정 트리는 주로 **분류 문제**에 사용됩니다. 각 노드를 거치며 데이터를 점점 더 세분화하여 최종적으로 분류합니다. **비선형적인 데이터**에도 유연하게 적용할 수 있다는 점이 큰 장점입니다.

2. 문제 정의

결정 트리 알고리즘의 학습 과정은 데이터를 여러 분할 규칙에 따라 나누는 방식으로 이루어집니다. 이때, 각 데이터 포인트는 레이블이 달려 있으며, 예를 들어 $y = 0$ 또는 $y = 1$과 같은 이진 분류 문제가 있다고 가정합니다.

- **문제의 목표**: 데이터의 특성 벡터 x를 기반으로 해당 데이터를 적절한 클래스로 분류하는 것입니다.
- 각 노드에서는 데이터를 나누는 기준을 설정하고, 이 기준에 따라 데이터를 두 집합으로 분할합니다.

3. 해결 방법

결정 트리 알고리즘을 구현하는 방법은 다양하며, 그중 **ID3 알고리즘**이 널리 사용됩니다. 이 알고리즘의 작동 과정은 다음과 같습니다:

3.1 ID3 알고리즘

ID3 알고리즘은 **정보 이득(Information Gain)**을 기준으로 데이터를 분할합니다. 정보 이득은 데이터 분할의 효과성을 측정하는 척도로, 이를 통해 최적의 분할 기준을 선택합니다.

- **정보 이득**은 분할 후 데이터의 균일성(즉, 한쪽으로의 치우침 정도)을 측정합니다.
- 분할된 데이터가 더 균일할수록, 다시 말해 한 클래스에 속하는 데이터가 많을수록 정보 이득이 높아집니다.

ID3 알고리즘은 각 특성에 대해 가능한 모든 분할을 시도합니다. 그 후 정보 이득이 가장 높은 분할 기준을 선택하여 데이터를 나눕니다. 이 과정을 반복하여 최종적으로 **분류 트리**를 구축합니다.

3.2 수식 설명

아래 수식은 결정 트리에서의 평균 로그 가능도(Average Log-Likelihood)를 나타내며, 이를 최적화하는 과정이 설명되고 있습니다.

$$\frac{1}{N}\sum_{i=1}^{N}[y_i \log f_D(x_i) + (1-y_i)\log(1-f_D(x_i))]$$

이 수식은 모델의 성능을 평가하는 지표로, 분류 결과와 실제 값의 일치 정도를 측정합니다. 값이 클수록 모델이 데이터를 더 정확하게 분류하고 있음을 의미합니다.

4. 예시 설명 (그림 3.4)

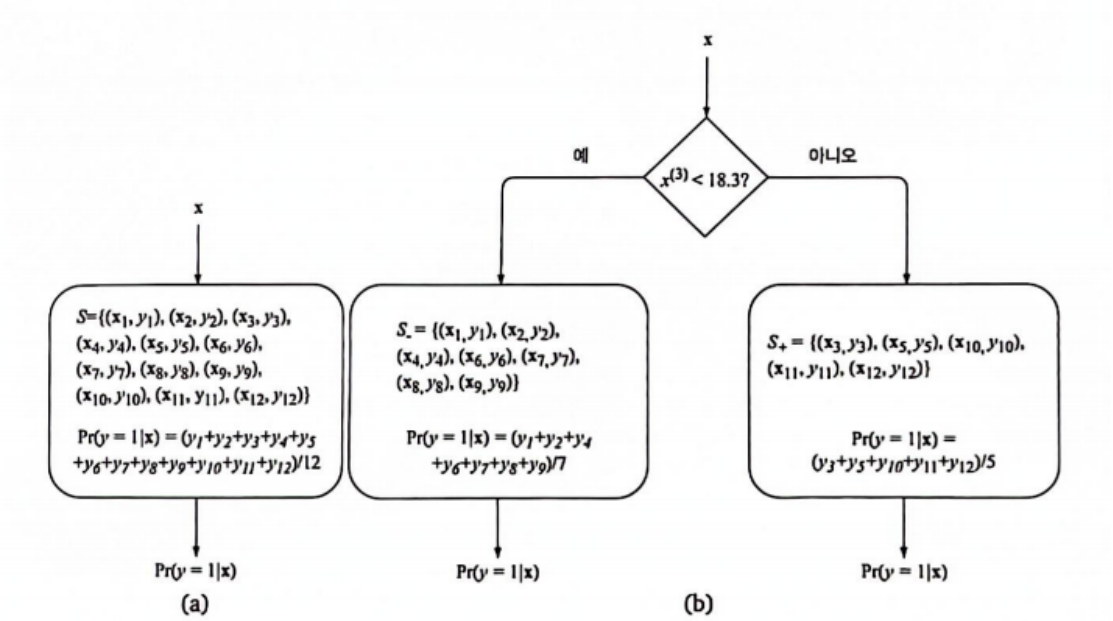

그림 3.4 결정 트리 구축 알고리즘을 표현한 순서도. 훈련 집합 S는 레이블이 달린 예제 12개로 구성된다. (a) 처음에는 결정 트리에 시작 노드만 있다. 그래서 모든 입력에 대해 예측값이 같다. (b) 첫 번째 분할 이후에 생성된 결정 트리로서 특징 3의 값이 18.3보다 작은지를 검사한 결과에 따라 두 말단 노드 중 하나로 예측한다.

위의 그림 3.4는 결정 트리의 분할 과정을 시각적으로 보여주고 있습니다. $x_1 > 18.37$ 라는 기준을 사용하여 데이터를 두 그룹으로 나누는 예시를 들고 있습니다.

- **왼쪽 노드**는 $x_1 \leq 18.37$인 데이터를 포함하고,
- **오른쪽 노드**는 $x_1 > 18.37$인 데이터를 포함합니다.

각 노드에서 데이터가 어떻게 분할되었는지를 시각적으로 확인할 수 있으며, 이 과정을 반복하여 트리가 확장되고, 최종적으로 모든 데이터가 특정 클래스에 속하게 됩니다.

5. 결정 트리 알고리즘의 분할 기준

결정 트리 알고리즘에서는 데이터를 여러 번 분할하면서 최종적으로 데이터를 특정 클래스에 할당합니다. 이때 중요한 것은 **각 분할이 얼마나 잘 이루어졌는가**를 평가하는 방법입니다.

5.1 분할 과정 설명

- 데이터를 분할할 때, 각 특성 x_i에 대해 임계값 t를 설정하고 그 값에 따라 데이터를 두 그룹 S_1과 S_2로 나누게 됩니다. 이때 나눈 두 그룹이 얼마나 잘 분할되었는지를 평가하는 척도가 필요합니다.

  예를 들어, 특성 x_i의 값이 특정 임계값 t보다 크거나 작으면, 그에 따라 데이터가 나뉘게 됩니다. **분할된 그룹**은 각각 **좌측 노드**와 **우측 노드**로 표현되며, 각 그룹에서 데이터가 얼마나 균일하게 분포되었는지를 평가합니다.

5.2 엔트로피 (Entropy) 기반 평가

- 엔트로피(Entropy)는 데이터의 불확실성을 측정하는 지표로, 분할된 데이터가 얼마나 균일하게 구성되어 있는지를 나타냅니다. 엔트로피 값이 낮을수록 데이터가 한 클래스에 집중되어 있음을 의미하며, 이는 좋은 분할을 의미합니다.

  엔트로피는 다음과 같이 정의됩니다:

  $$H(S_1, S_2) = -\sum_{i=1}^{2} p_i \log(p_i)$$

  여기서 p_i는 각 그룹 S_1과S_2에 속한 데이터의 비율입니다.

6. 엔트로피 기반 분할 기준

6.1 엔트로피 수식 설명

아래 수식 은 분할된 두 집합 S_1과 S_2의 엔트로피를 계산하는 과정을 나타냅니다.

$$f_{IDS} = \frac{1}{|S|} \sum_{(x_i, y_i) \in S} y$$

이는 분할된 데이터 집합에서 각 클래스에 속하는 데이터의 비율을 계산한 후, 이를 엔트로피로 변환하여 데이터의 불확실성을 측정합니다. 이 엔트로피 수치를 기반으로 결정 트리의

각 분할이 얼마나 효과적으로 이루어졌는지 평가할 수 있습니다.

6.2 분할된 데이터의 엔트로피 평가

아래 수식은 **엔트로피의 가중 평균**을 나타냅니다. 이는 분할된 데이터 그룹의 크기와 각 그룹의 엔트로피를 고려하여 전체 분할의 품질을 평가하는 방식입니다.

$$H(S_1, S_2) = \frac{|S_1|}{|S|} H(S_1) + \frac{|S_2|}{|S|} H(S_2)$$

이 수식은 각 그룹의 엔트로피를 해당 그룹의 크기에 비례하여 가중 평균한 것으로, 전체 분할의 엔트로피를 계산합니다. 이를 통해 분할의 품질을 평가할 수 있으며, 계산된 엔트로피가 낮을수록 더 효과적인 분할이 이루어졌음을 의미합니다.

7. 분할 후 평가 방법

결정 트리 알고리즘은 데이터 분할 시 각 노드의 엔트로피를 계산하여 최적의 분할을 찾습니다. **ID3 알고리즘**은 이 과정에서 엔트로피를 기준으로 데이터를 가장 효과적으로 나눌 수 있는 특성과 임계값을 식별합니다.

7.1 최적의 분할 기준

- ID3 알고리즘은 모든 특성에 대해 가능한 임계값을 평가하고, 엔트로피를 기준으로 최적의 분할을 선택합니다.
- 분할이 진행될수록 엔트로피는 감소하며, 데이터는 점진적으로 단일 클래스로 수렴합니다.

7.2 엔트로피 최소화와 정보 이득

엔트로피가 감소함에 따라 정보 이득(Information Gain)은 증가합니다. 높은 정보 이득은 효과적인 데이터 분할을 의미하며, 결정 트리 알고리즘은 이 정보 이득을 최대화하는 방향으로 분할을 수행합니다.

SVM (Support Vector Machine)

1. SVM의 기본 개념

1.1 SVM 정의

SVM은 **분류 문제**를 해결하기 위한 머신러닝 알고리즘입니다. 이 알고리즘은 **초평면(hyperplane)**을 사용하여 데이터를 두 그룹으로 분류합니다. SVM의 핵심은 **결정 경계(margin)**를 최대화하는 것입니다. 즉, 두 그룹 사이의 거리가 가장 넓은 초평면을 찾아 데이터를 분류합니다.

2. SVM의 특징적인 상황

SVM은 **노이즈가 있는 데이터**와 **비선형 데이터**에 대해 효과적으로 적용될 수 있습니다.

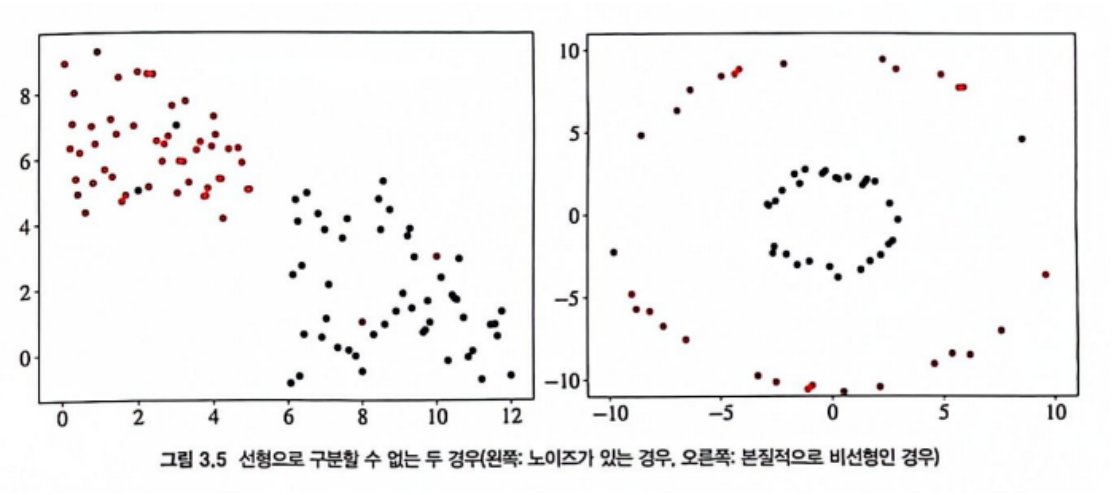

그림 3.5 선형으로 구분할 수 없는 두 경우(왼쪽: 노이즈가 있는 경우, 오른쪽: 본질적으로 비선형인 경우)

2.1 노이즈가 있는 경우

그림 3.5(a)는 노이즈가 있는 데이터를 보여줍니다. 이런 경우 데이터가 완벽하게 두 그룹으로 나뉘지 않으며, 일부 포인트가 다른 그룹에 속할 수 있습니다. SVM은 이러한 **노이즈(잡음)**를 고려하여 **최소한의 분할 오류**로 데이터를 분류합니다.

- 노이즈 데이터에서는 **완벽한 분리**가 불가능할 수 있습니다. SVM은 이런 상황에서 일부 오분류된 포인트를 허용하면서 **최적의 결정 경계**를 찾습니다.

2.2 비선형 데이터

그림 3.5(b)는 **비선형적**으로 분포된 데이터를 보여줍니다. 이런 데이터는 단순한 직선이나 평면으로 구분할 수 없습니다. SVM은 **커널 트릭(kernel trick)**을 사용하여 데이터를 **고차원 공간으로 변환**한 후, 그 공간에서 선형 초평면을 찾아 비선형 데이터를 분류합니다.

3. SVM의 수식적 표현

3.1 SVM의 제약 조건

SVM 알고리즘의 핵심은 데이터가 결정 경계(margin)에서 떨어져 있도록 만드는 것입니다. 이를 위해 **최적화 문제**로 표현됩니다.

SVM의 제약 조건은 다음과 같습니다:

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 \quad \text{for all} \quad i = 1, 2, \ldots, N$$

여기서:

- y_i는 각 데이터 포인트의 실제 레이블입니다 (즉, $y_i = 1$ 또는 $y_i = -1$).

- $\mathbf{w} \cdot \mathbf{x_i}$는 데이터 포인트 $\mathbf{x_i}$와 결정 경계 초평면 사이의 거리입니다.
- b는 초평면의 절편입니다.

이 조건을 만족시킬 때, SVM은 데이터 포인트들이 각 클래스에 속하도록 분류합니다.

3.2 목적 함수

SVM의 최적화 목적은 **결정 경계와 각 클래스 사이의 거리를 최대화**하는 것입니다. 이를 위해 SVM은 다음과 같은 목적 함수를 최소화합니다:

$$\min \frac{1}{2}\|\mathbf{w}\|^2$$

이 목적 함수는 **초평면의 가중치**를 최소화하여 결정 경계와 데이터 포인트들 사이의 거리가 최대가 되도록 합니다. 이때, 경사 하강법(gradient descent)과 같은 최적화 알고리즘을 사용하여 최적의 $\mathbf{w}$와 b 값을 찾습니다.

4. 노이즈를 다루는 방법

4.1 힌지 손실 함수 (Hinge Loss)

노이즈가 있는 데이터를 처리하기 위해 SVM은 힌지 손실 함수(hinge loss function)를 사용합니다. 이 함수는 다음과 같이 정의됩니다:

$$\max(0, 1 - y_i(\mathbf{w} \cdot \mathbf{x_i} - b))$$

힌지 손실 함수는 데이터 포인트가 결정 경계에서 잘못 분류되었을 때 페널티를 부과합니다. 즉, **잘못 분류된 데이터 포인트**일수록 더 큰 손실을 발생시키고, 이 손실을 최소화하는 방식으로 모델을 학습시킵니다.

4.2 비용 함수 (Cost Function)

노이즈 데이터를 처리하기 위해 SVM은 비용 함수(cost function)를 최소화하는 방식으로 최적화를 진행합니다. 이 비용 함수는 다음과 같습니다:

$$C\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w} \cdot \mathbf{x_i} - b))$$

이 수식에서:

- 첫 번째 항 $C\|\mathbf{w}\|^2$는 **결정 경계의 폭**을 최대화하는 역할을 합니다.
- 두 번째 항은 **힌지 손실 함수**로, 잘못 분류된 데이터 포인트에 대한 페널티를 나타냅니다.

이 두 항을 동시에 최소화함으로써, SVM은 노이즈 데이터에서도 최적의 결정 경계를 찾을 수 있습니다.

5. SVM의 분류 성능과 일반화 능력 균형

5.1 SVM의 분류 성능 최적화

SVM에서 **C 파라미터**는 분류 성능을 조정하는 핵심 역할을 합니다. **C 값**은 분류 오류와 **마진 크기** 사이의 균형을 결정합니다.

- **높은 C 값**: 분류 오류 최소화에 집중합니다. 이 경우, 모델이 학습 데이터에 과도하게 적합되어 **과적합(overfitting)**이 발생할 수 있습니다.
- **낮은 C 값**: 분류 마진을 크게 유지하며, 일부 분류 오류를 허용합니다. 이는 모델의 **일반화** 성능을 향상시키지만, 특정 상황에서는 부적절할 수 있습니다.

따라서 SVM은 **C 값**을 적절히 조정하여 학습 데이터에 대한 분류 성능과 새로운 데이터에 대한 **일반화 능력** 사이의 균형을 맞추는 것이 중요합니다.

6. 비선형 데이터 처리 방법

6.1 커널 트릭(Kernel Trick) 활용

SVM은 기본적으로 **선형 분류** 알고리즘이지만, **비선형 데이터**도 처리할 수 있습니다. 그 핵심은 **커널 트릭(Kernel Trick)**입니다. 커널 트릭은 데이터를 **고차원 공간으로 매핑**하여 선형 분리가 가능하도록 하는 기법입니다.

- **커널 함수**: 데이터를 고차원 공간으로 변환합니다. 이를 통해 SVM은 비선형 데이터도 **고차원 공간에서 선형적으로 분리**할 수 있습니다.
- **커널 함수의 예**: 대표적으로 **다항식 커널(Polynomial Kernel)**과 **가우시안 커널(Gaussian Kernel)** 등이 있습니다.

그림에서 언급된 수식은 **비선형 변환 함수** $\phi(x)$를 적용하여 데이터를 변환하는 과정을 나타냅니다. 이를 통해 비선형 데이터를 고차원 공간으로 매핑하고, 그 공간에서 선형 분리 초평면을 찾을 수 있습니다.

6.2 커널 적용 결과 (그림 3.6)

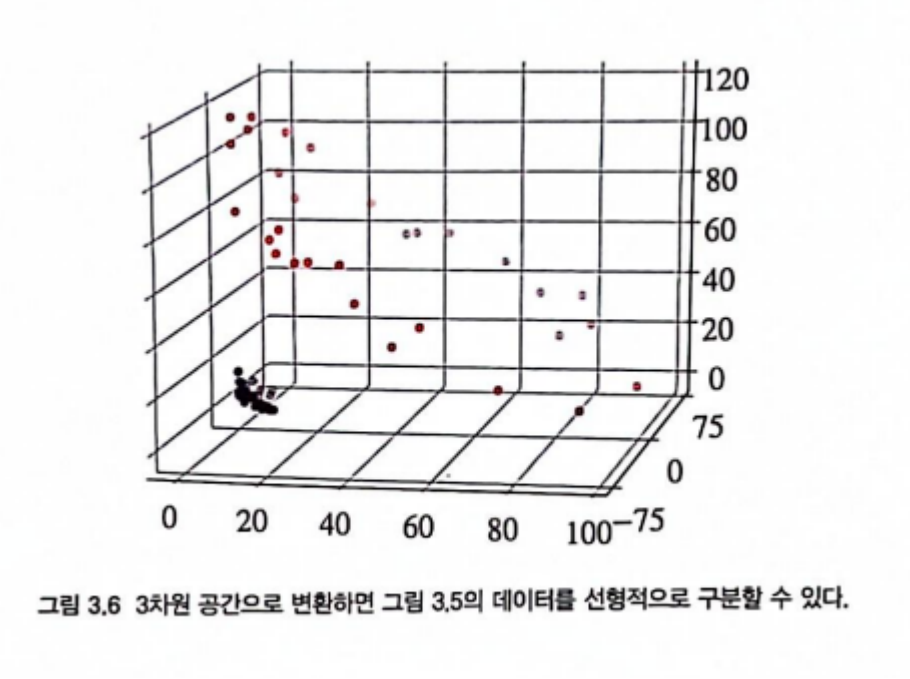

그림 3.6 3차원 공간으로 변환하면 그림 3.5의 데이터를 선형적으로 구분할 수 있다.

그림 3.6은 **커널 트릭**을 사용하여 3차원 공간에서 데이터를 분류하는 예시를 보여줍니다. 데이터를 3차원으로 변환하면 원래 비선형적으로 분포된 데이터가 **선형적으로 분리**될 수 있습니다. 이러한 커널 트릭을 통해 SVM은 비선형 데이터도 효과적으로 분류할 수 있습니다.

7. SVM의 최적화 과정

7.1 라그랑주 승수법(Lagrange Multiplier Method)

SVM의 최적화 문제는 **라그랑주 승수법**으로 해결됩니다. 이 방법은 **제약 조건**이 있는 최적화 문제를 푸는 데 사용되며, SVM에서도 이를 적용하여 **결정 경계**를 찾습니다.

- **최적화 문제**는 다음과 같이 표현됩니다:

$$\max_{\alpha_1 \ldots \alpha_n} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

  이 수식에서:

  - α_i는 **라그랑주 승수**입니다.
  - $\langle x_i, x_j \rangle$는 **커널 함수**로 변환된 데이터 간의 내적을 나타냅니다.

이 최적화 문제를 풀면 **결정 경계**가 도출되며, 이를 통해 데이터가 올바르게 분류됩니다.

7.2 라그랑주 승수의 역할

라그랑주 승수 α_i는 데이터 포인트들이 결정 경계에 얼마나 영향을 미치는지를 결정합니다. α_i 값이 큰 데이터 포인트는 서포트 벡터(support vector)가 되며, 결정 경계를 형성하는 데 중요한 역할을 합니다.

- 서포트 벡터는 결정 경계와 가장 가까운 데이터 포인트들로, SVM은 이 서포트 벡터들을 기준으로 데이터를 분류합니다.

8. 커널 트릭 (Kernel Trick)

커널 트릭은 비선형 데이터를 고차원 공간으로 변환하여 선형적으로 분리할 수 있게 하는 기법입니다. 이는 특히 **SVM**과 같은 알고리즘에서 비선형 데이터를 효과적으로 처리하는 데 활용됩니다.

8.1 커널 함수의 정의

커널 함수는 두 벡터 x와 x' 사이의 내적을 **고차원 공간에서 계산**하는 함수입니다. 이를 통해 고차원 공간에서의 연산을 효율적으로 수행할 수 있습니다. 주로 사용되는 커널 함수로는 **RBF 커널(Radial Basis Function)** 또는 **가우시안 커널**이 있습니다.

- **RBF 커널 함수**의 수식은 다음과 같습니다:

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

- 여기서 $\|x - x'\|$는 두 벡터 사이의 **유클리드 거리**를 나타내며, σ는 커널 함수의 폭을 결정하는 하이퍼파라미터입니다.

8.2 유클리드 거리 (Euclidean Distance)

유클리드 거리는 두 벡터 x와 x' 사이의 직선 거리를 나타내며, 수식으로는 다음과 같이 표현됩니다:

$$d(x, x') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + \cdots + (x_D - x'_D)^2}$$

이 거리는 고차원 공간에서 데이터를 비교하고 분류할 때 사용됩니다.

8.3 커널 트릭의 역할

커널 트릭은 데이터를 고차원 공간으로 변환하여 **비선형 데이터**도 선형적으로 분리할 수 있도록 만듭니다. RBF 커널은 데이터 간의 거리를 기반으로 고차원에서의 변환을 수행하며, 이를 통해 복잡한 결정 경계를 효과적으로 찾을 수 있습니다.

9. k-NN 알고리즘 (k-최근접 이웃)

- k-NN (k-Nearest Neighbors)은 **비모수 학습**에 속하는 대표적인 분류 알고리즘입니다. 이 알고리즘은 새로운 데이터 포인트가 주어졌을 때, 학습 데이터 중에서 가장 가까운 k개의 이웃을 찾아 그 이웃들의 레이블을 기반으로 새로운 데이터 포인트를 분류합니다.

9.1 기본 개념

k-NN은 **데이터 간의 거리**를 기반으로 분류를 수행합니다. 가장 가까운 이웃들의 다수결에 따라 새로운 데이터의 클래스를 결정하며, 이 과정에서 거리 측정 방법이 매우 중요한 역할을 합니다.

9.2 거리 측정 방법

k-NN에서 두 벡터 사이의 거리를 측정하는 방법으로는 여러 가지가 있습니다. 가장 일반적인 방법은 **유클리드 거리**를 사용하는 것이지만, 그 외에도 다양한 거리 함수가 존재합니다:

- **유클리드 거리**: $d(x_i, x_j) = \sqrt{\sum_{k=1}^{D} (x_i^{(k)} - x_j^{(k)})^2}$

  이는 두 점 사이의 직선 거리를 계산하는 방식입니다.

- **코사인 유사도(Cosine Similarity)**:

  $$s(x_i, x_j) = \frac{\sum_{k=1}^{D} x_i^{(k)} x_j^{(k)}}{\sqrt{\sum_{k=1}^{D} (x_i^{(k)})^2} \sqrt{\sum_{k=1}^{D} (x_j^{(k)})^2}}$$

  코사인 유사도는 두 벡터가 이루는 각도에 기반하여 유사도를 측정하는 방식입니다. 값이 1에 가까울수록 두 벡터가 같은 방향을 가지며, 0에 가까울수록 다른 방향을 가집니다.