2.2 수식

2.2.1 로지스틱 회귀 (Logistic Regression)

로지스틱 회귀는 이진 분류 문제를 풀기 위한 대표적인 지도 학습 모델입니다. 입력 벡터 $\mathbf{x} \in \mathbb{R}^d$ 가 주어졌을 때, 클래스 $y \in \{0, 1\}$ 중 하나에 속할 확률을 예측합니다.

2.2.1.1 선형 결합과 시그모이드 함수

입력 벡터 $\mathbf{x}$에 대해 먼저 선형 결합을 계산합니다.

$$z = \mathbf{w}^T \mathbf{x} + b$$

이 값을 시그모이드 함수에 통과시켜 확률로 변환합니다.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad \Rightarrow \quad \hat{y} = \sigma(\mathbf{w}^T \mathbf{x} + b)$$

2.2.1.2 확률 분포 기반 모델링

출력 클래스 $y \in \{0, 1\}$는 확률값 $\hat{y}$를 따르는 베르누이 분포(Bernoulli Distribution)로 가정합니다.

> 베르누이 시행: 결과가 두 가지 중 하나로만 나오는 실험이나 시행(동전 던지기)을 말합니다.
> 베르누이 확률 변수: 베르누이 시행의 결과를 실수 0 또는 1로 바꾼 것을 말합니다.
> 베르누이 확률 분포: 베르누이 확률변수의 분포를 베르누이 확률분포 혹은 베루누이 분포라고 합니다.

$$P(y \mid \mathbf{x}) = \hat{y}^y (1 - \hat{y})^{1 - y}$$

위 수식은 **y가 1일 확률은 $\hat{y}$, 0일 확률은 $1 - \hat{y}$** 를 **하나의 공식으로 통합** 한 것입니다.

상황	수식 계산	의미
$y = 1$	$\hat{y}^1 (1 - \hat{y})^0 = \hat{y}$	양성 클래스의 확률
$y = 0$	$\hat{y}^0 (1 - \hat{y})^1 = 1 - \hat{y}$	음성 클래스의 확률

2.2.1.3 우도 함수 정의

훈련 데이터셋 $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n$에 대해 전체 우도 함수는 다음과 같습니다.

> 우도(likelihood)란, 현재 주어진 모델 파라미터 (w, b)에 대해 우리가 관측한 데이터가 나올 확률이 얼마나 높은지를 나타내는 값입니다.
> 즉, 파라미터를 고정시키고 관측 데이터가 나올 확률을 계산해보고 그 확률을 최대화 시켜보자는 것입니다.
> → 파라미터를 바꾸어서 말입니다.

$$\mathcal{L}(\mathbf{w}, b) = \prod_{i=1}^{n} \left[\hat{y}^{(i)}\right]^{y^{(i)}} \left[1 - \hat{y}^{(i)}\right]^{1 - y^{(i)}}$$

$$\hat{y}^{(i)} = \sigma(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \quad \text{where} \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

2.2.1.4 로그우도 함수 도출

계산을 용이하게 하기 위해 우도에 로그를 취하여 **로그우도 함수**로 변환합니다.

> 곱셈이 덧셈으로 바뀝니다.
> 미분도 간단해집니다.(곱셈의 미분보다 말입니다)
> 음수를 취해서 최소화 문제로 바꾸어도 상관없습니다.

$$J(\mathbf{w}, b) = -\ell(\mathbf{w}, b) = -\sum_{i=1}^{n} \left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

또는 평균 손실로 표현하는 것도 가능합니다. 평균을 구하기 위해서 전체 데이터 수로 나누어 줍니다.

$$J(\mathbf{w}, b) = -\frac{1}{n} \sum_{i=1}^{n} \left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

2.2.1.5 경사 하강법을 위한 미분

이 손실 함수를 최소화하기 위해, 각 파라미터에 대해 기울기를 계산합니다.

(1) 가중치 $\mathbf{w}$에 대한 편미분:

$$\frac{\partial J}{\partial \mathbf{w}} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$$

(2) 편향 b 에 대한 편미분:

$$\frac{\partial J}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}^{(i)} - y^{(i)})$$

2.2.2 서포트 벡터 머신 (Support Vector Machine)

SVM은 데이터를 분류하기 위해 결정 경계(선/면)를 학습합니다.

그 과정에서 **클래스 간 마진을 최대화**함으로써 일반화 성능을 높이는 것이 핵심 아이디어입니다.

2.2.2.1 결정 경계와 선형 분리

SVM은 먼저 선형 결정 경계를 정의합니다. 이 경계는 다음과 같은 형태입니다.

$$\mathbf{w}^T \mathbf{x} + b = 0$$

- $\mathbf{x} \in \mathbb{R}^d$: 입력 벡터(예측하고자 하는 데이터 포인트)
- $\mathbf{w}$: 경계면의 방향을 나타내는 법선 벡터(어떤 면에 수직인 방향을 가리키는 벡터입니다.)
- b: 절편

분류는 다음 규칙에 따릅니다:

- $y = \begin{cases} +1 & \text{if } \mathbf{w}^T \mathbf{x} + b \geq 0 \\ -1 & \text{if } \mathbf{w}^T \mathbf{x} + b < 0 \end{cases}$

2.2.2.2 마진의 정의

마진(margin)이란, 두 클래스 데이터 사이에서 결정 경계로부터 가장 가까운 점까지의 거리입니다.

SVM은 이 마진을 가능한 넓게 설정하려고 합니다. 마진의 수학적 표현은 다음과 같습니다.

$$\text{Margin} = \frac{2}{\|\mathbf{w}\|}$$

여기서$\|\mathbf{w}\|$는 벡터 $\mathbf{w}$ 의 유클리디안 노름입니다.

> $\|\mathbf{w}\| = \sqrt{w_1^2 + w_2^2 + \cdots + w_d^2}$

2.2.2.3 하드 마진 최적화 문제

모든 데이터가 완벽하게 선형 분리 가능하다고 가정할 때, SVM은 다음과 같은 최적화 문제를 풉니다.

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

이때, 제약 조건은 다음과 같습니다.

$$y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 \quad \forall i$$

이 구조는:

- $\frac{1}{2}\|\mathbf{w}\|^2$: 마진 최대화를 위한 최소화 대상
- 제약 조건: 모든 샘플이 경계에서 마진 이상 떨어지도록 강제

2.2.2.4 소프트 마진과 패널티 C

현실에서는 데이터가 완벽하게 분리되지 않는 경우가 많습니다. 이럴 때는 슬랙 변수 ξ_i를 도입해 **오류를 허용**하는 소프트 마진 모델을 사용합니다. 최적화 문제는 다음과 같이 바뀝니다.

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

제약 조건 역시 다음과 같이 바뀝니다.

$$y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

- $C > 0$: 마진 위반에 대한 패널티 계수(마진을 위반한 샘플에 얼마나 강하게 벌점을 줄지를 정합니다.)
- ξ_i: 각 데이터 포인트가 마진을 위반한 정도

Orange UI 항목	수식에서의 변수	설명
Cost	C	소프트 마진 SVM에서 **마진 위반에 대한 패널티 계수**입니다.

2.2.2.5 커널 함수 개념

SVM은 선형 분리 기준을 가지지만, 현실의 데이터는 종종 **선형적으로 분리할 수 없습니다.**

이런 경우, 입력 데이터를 **고차원 공간으로 매핑**해 선형 분리가 가능하도록 만드는데, 이 과정을 효율적으로 수행하기 위해 커널 함수(Kernel function)를 사용합니다.

커널 함수는 입력 벡터를 직접 고차원으로 올리지 않고도, 고차원에서의 내적 결과를 계산해주는 함수입니다.

가장 대표적인 커널:

- **RBF 커널 (Radial Basis Function kernel)**:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2)$$

항목	설명
$\|\mathbf{x} - \mathbf{x}'\|^2$	두 벡터 사이의 거리 (유클리디안 거리 제곱)
$\exp(-\gamma \cdot 거리)$	거리가 작으면 1에 가까워지고, 크면 0에 가까워짐
γ	얼마나 민감하게 거리 차이를 반영할지 조절하는 하이퍼파라미터

Polynomial 커널: 입력 벡터를 고차 다항식 특성 공간으로 확장합니다.

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + r)^d$$

2.2.3 KNN (K-Nearest Neighbors)

KNN은 **사전 학습이 없는** 알고리즘으로, 새로운 데이터가 들어올 때마다 **기존 학습 데이터와의 거리**를 계산하여 가장 가까운 K개의 이웃을 찾고, **다수결 혹은 평균**으로 결과를 결정합니다.

분류 문제에서는 **다수결**, 회귀 문제에서는 **평균값**을 기반으로 예측합니다.

2.2.3.1 거리 측정 방식 (Distance Metrics)

입력 벡터 두 개를 각각 $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$, $\boldsymbol{x'} = (x'_1, x'_2, \ldots, x'_d)$라고 할 때, 다양한 거리 측정 방법은 다음과 같습니다.

(1) 유클리디안 거리 (Euclidean Distance)

가장 흔히 쓰이는 "직선 거리"입니다.

$$d(\boldsymbol{x}, \boldsymbol{x'}) = \sqrt{\sum_{i=1}^{d} (x_i - x'_i)^2}$$

(2) 맨해튼 거리 (Manhattan Distance)

$$d(\boldsymbol{x}, \boldsymbol{x'}) = \sum_{i=1}^{d} |x_i - x'_i|$$

(3) 체비쇼프 거리 (Chebyshev Distance, Maximal Distance)

가장 큰 차이 하나를 기준으로 계산합니다.

$$d(\boldsymbol{x}, \boldsymbol{x'}) = \max_i |x_i - x'_i|$$

(4) 마할라노비스 거리 (Mahalanobis Distance)

공분산 행렬 $\boldsymbol{S}$를 고려한 거리로, 데이터의 분포를 반영합니다.

$$d(\boldsymbol{x}, \boldsymbol{x'}) = \sqrt{(\boldsymbol{x} - \boldsymbol{x'})^\top \boldsymbol{S}^{-1} (\boldsymbol{x} - \boldsymbol{x'})}$$

2.2.3.2 가중치 방식 (Weights)

K개의 이웃에 대해 결과를 집계할 때 가중치를 줄 수도 있습니다.

- **Uniform**: 모든 이웃에 동일한 가중치 → 단순 다수결
- **Distance-based**: 가까운 이웃에게 더 큰 가중치를 부여

거리 기반 가중치의 예:

$$w_i = \frac{1}{d(\boldsymbol{x}, \boldsymbol{x}_i) + \varepsilon}$$

2.2.3.3 예측 수식

분류 (Classification)

가장 많이 등장한 클래스를 선택합니다.

$$\hat{y} = \arg\max_{c \in \mathcal{C}} \sum_{i \in \mathcal{N}_k(\boldsymbol{x})} w_i \cdot \mathbf{1}[y_i = c]$$

기호	의미
$\hat{y}$	최종 예측 클래스
$\mathcal{C}$	가능한 클래스 집합 (예: {0, 1})
$\mathcal{N}_k(\boldsymbol{x})$	입력 $\boldsymbol{x}$의 주변 K개의 최근접 이웃의 인덱스 집합
y_i	이웃 i의 실제 클래스 레이블
w_i	이웃 i의 가중치 (예: 거리의 역수 등)
$\mathbf{1}[y_i = c]$	y_i가 클래스 c일 경우 1, 아니면 0인 **Indicator Function**

이웃 번호	레이블 y_i	거리 d_i	가중치 $w_i = \frac{1}{d_i}$
1	0	0.5	2.0
2	1	0.4	2.5
3	1	0.3	3.33

이웃 번호	레이블 y_i	거리 d_i	가중치 $w_i = \frac{1}{d_i}$
4	0	0.6	1.67
5	1	0.2	5.0

클래스 0:

- 이웃 1: $y_1 = 0, \quad w_1 = 2.0$
- 이웃 4: $y_4 = 0, \quad w_4 = 1.67$

합계: 3.67

클래스 1:

- 이웃 2: $y_2 = 1, \quad w_2 = 2.5$
- 이웃 3: $y_3 = 1, \quad w_3 = 3.33$
- 이웃 5: $y_5 = 1, \quad w_5 = 5.0$

합계: 10.83

2.2.4 결정 트리 (Decision Tree)

결정 트리는 입력 특성에 대해 **조건 분기를 반복**하여 데이터를 분류하는 트리 기반 모델입니다.

루트 노드부터 시작해 **특정 속성의 조건**에 따라 데이터를 분할하며,

각 리프 노드는 최종 분류 결과 또는 예측값을 나타냅니다.

- 분류(Classification) 문제에서는 클래스 레이블이 리프 노드에 배정됩니다.
- 회귀(Regression) 문제에서는 리프 노드에 평균값이 배정됩니다

2.2.4.2 노드 분할 기준: 불순도 (Impurity)

결정 트리는 각 노드에서 불순도(impurity)가 가장 많이 줄어드는 방향으로 데이터를 분할합니다.

(1) 지니 불순도 (Gini Impurity)

$$G(t) = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2$$

여기서 p_k는 노드 t에서 클래스 k가 등장할 확률입니다.

클래스	샘플 수	비율 p_k
0	4	$\frac{4}{10} = 0.4$
1	6	$\frac{6}{10} = 0.6$

$$G(t) = \sum_{k=1}^{K} p_k(1 - p_k) \quad = 0.4(1 - 0.4) + 0.6(1 - 0.6) \quad = 0.4 \cdot 0.6 + 0.6 \cdot 0.4 \quad = 0.24 + 0.24 \quad = 0.48$$

$$G(t) = 1 - \sum_{k=1}^{K} p_k^2 \quad = 1 - (0.4^2 + 0.6^2) \quad = 1 - (0.16 + 0.36) \quad = 1 - 0.52 \quad = 0.48$$

(2) 엔트로피 (Entropy)

$$H(t) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

정보 이론에서 유래된 개념으로, 분포의 불확실성을 측정합니다.

2.2.4.3 정보 이득 (Information Gain)

정보 이득은 **부모 노드의 불순도**에서 **자식 노드들의 가중 평균 불순도**를 뺀 값입니다.

즉, 이 분할로 얼마나 정돈됐는가를 수치화합니다.

$$IG = I(\text{부모}) - \left(\frac{N_L}{N} I(L) + \frac{N_R}{N} I(R) \right)$$

- $I(\cdot)$: 지니 또는 엔트로피
- N_L, N_R: 좌/우 자식 노드의 샘플 수
- N: 전체 샘플 수
- 정보 이득이 가장 높은 속성과 임계값을 찾아 분할(split)을 수행합니다.

example:

샘플	나이	클래스
A	22	0
B	25	0
C	28	1
D	30	0
E	32	1
F	35	1
G	38	1
H	40	1
I	42	1
J	45	0

$$p_0 = \frac{4}{10} = 0.4$$
$$p_1 = \frac{6}{10} = 0.6$$

전체 불순도

$$G(t) = 1 - \left(p_0^2 + p_1^2\right) \quad = 1 - (0.4^2 + 0.6^2) \quad = 1 - (0.16 + 0.36) \quad = 1 - 0.52 \quad = 0.48$$

분할 시도

나이 < 30

왼쪽 그룹 (나이 < 30): A, B, C → 클래스 0, 0, 1

→ 클래스 비율은

$$p_0 = \frac{2}{3}, \quad p_1 = \frac{1}{3}$$

$$G_L = 1 - \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2 \right) = 1 - \left(\frac{4}{9} + \frac{1}{9} \right) = 1 - \frac{5}{9} = \frac{4}{9} \approx 0.444$$

오른쪽 그룹 (나이 ≥ 30): D ~ J → 클래스 0, 1, 1, 1, 1, 1, 0

→ 클래스 비율은

$$p_0 = \frac{2}{7}, \quad p_1 = \frac{5}{7}$$

$$G_R = 1 - \left(\left(\frac{2}{7}\right)^2 + \left(\frac{5}{7}\right)^2 \right) = 1 - \left(\frac{4}{49} + \frac{25}{49} \right) = 1 - \frac{29}{49} = \frac{20}{49} \approx 0.408$$

2.2.4.5 과적합과 가지치기 (Overfitting & Pruning)

- 결정 트리는 과적합에 취약합니다. 모든 노드를 다 분할하면 훈련 데이터에 너무 잘 맞지만 일반화 성능이 낮아질 수 있습니다.

- 이를 방지하기 위해 다음과 같은 기법을 사용합니다.

	설명
사전 가지치기 (Pre-pruning)	트리 깊이 제한, 최소 샘플 수 제한 등 조건 미리 설정
사후 가지치기 (Post-pruning)	완성된 트리에서 덜 중요한 노드를 제거