선형 회귀 & 로지스틱 회귀

- Bitamin 8주차 세션 -

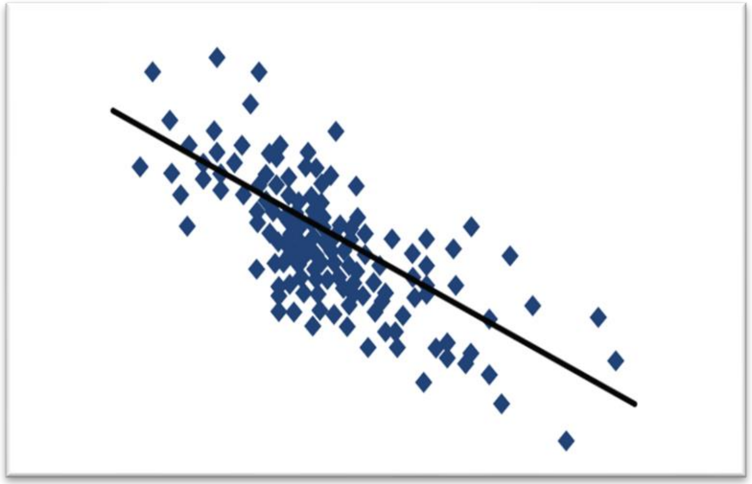
6조

권동구, 김은비, 조성우, 조형주

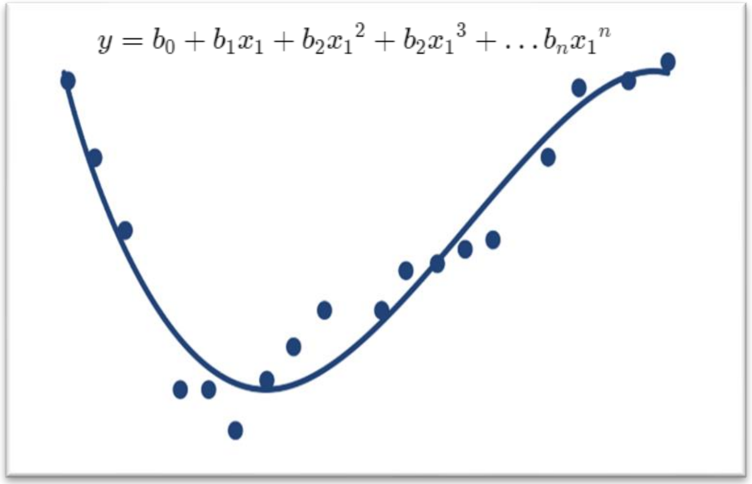
선형 회귀

1. 선형 회귀 종류
2. 선형 회귀 모델
3. 비용 함수
4. 최적화
5. 정규화, 규제

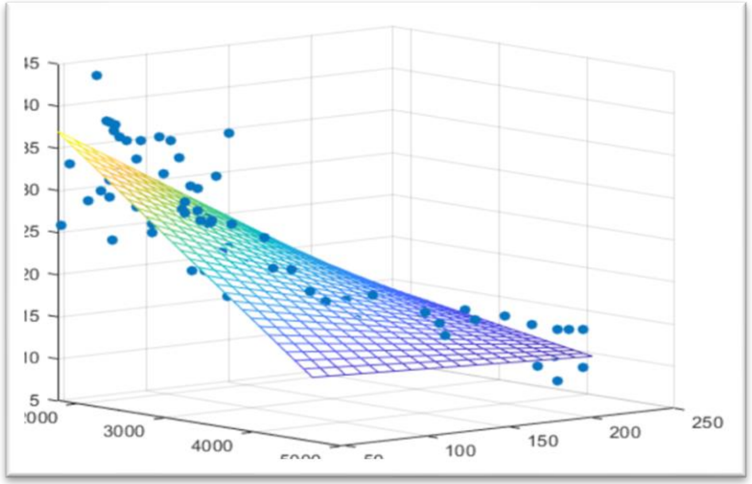
1. 선형 회귀 종류



① 단순 선형 회귀



② 다항 회귀



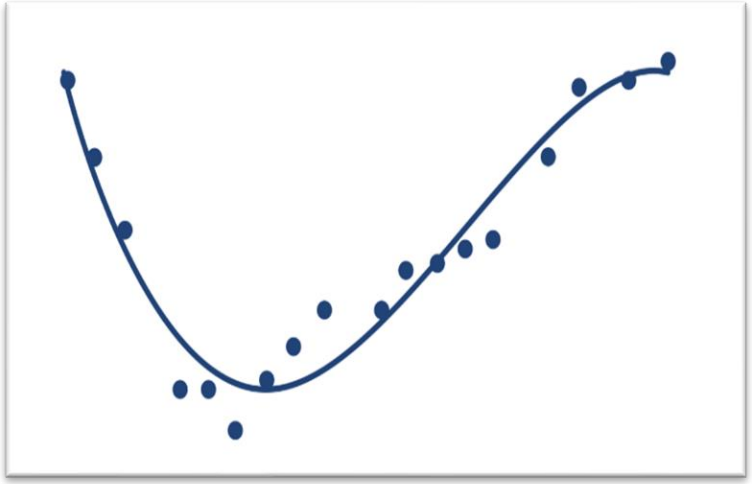
③ 다중 선형 회귀

> 특성과 타깃 사이의 관계를 가장 잘 나타내는 **선형 방정식**을 찾는 알고리즘

1. 선형 회귀 종류

변수 치환

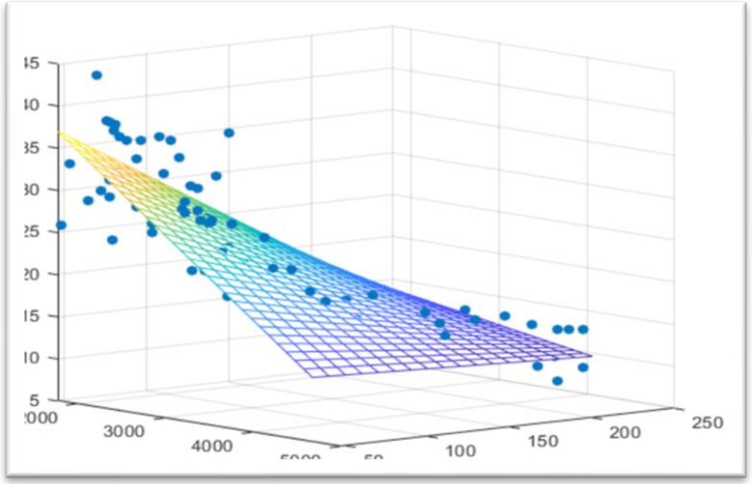
다항 회귀



$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 ..... + \theta_n x_1^n$$

$$x_1^2 \Rightarrow x_2$$
$$x_1^3 \Rightarrow x_3$$
$$\vdots$$
$$x_1^n \Rightarrow x_n$$

다중 선형 회귀



$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 ..... + \theta_n x_n$$

> 다항 회귀는 **변수 치환**을 통해서 선형 회귀로 바꿔 풀 수 있다

2. 선형 회귀 모델 (다중 회귀)

n개의 특징, m개의 dataset을 가진 다중 회귀 모델

- 특징 벡터 : $\mathbf{x} = [x_1, \dots, x_n]^T$
- 학습 dataset : $\mathbb{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m} = \{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(m)}, y^{(m)})\}$
- 선형 모델 : $h_\theta(x) = h_\theta(x_1, \dots. x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \dots. + \theta_n x_n$

① 한 개의 훈련 모델

$$\boxed{h_\theta(\underset{\sim}{x})} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

$$= [\theta_0 \quad \theta_1 \quad \dots \quad \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \boxed{= \underset{\sim}{\theta}^T \underset{\sim}{x}}$$

② 모든 훈련 모델

$$\boxed{\mathbf{h}_\theta(\mathbf{X}) = \begin{bmatrix} \boldsymbol{\theta}^T \mathbf{x}^{(1)} \\ \boldsymbol{\theta}^T \mathbf{x}^{(2)} \\ \vdots \\ \boldsymbol{\theta}^T \mathbf{x}^{(m)} \end{bmatrix} = \mathbf{X}\boldsymbol{\theta}}$$

$\mathbf{x} \triangleq [x_0, x_1, \dots, x_n]^T$

$\boldsymbol{\theta} \triangleq [\theta_0, \theta_1, \dots, \theta_n]^T$

2. 선형 회귀 모델 (다중 회귀)

n개의 특징, m개의 dataset을 가진 다중 회귀 모델

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)T} \\ \mathbf{x}^{(2)T} \\ \vdots \\ \mathbf{x}^{(m)T} \end{bmatrix} \longrightarrow \boxed{\mathbf{h}_{\boldsymbol{\theta}}} \longrightarrow$$

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$$\mathbf{h}_{\boldsymbol{\theta}}(\mathbf{X}) = \begin{bmatrix} h_{\boldsymbol{\theta}}(\mathbf{x}^{(1)}) \\ h_{\boldsymbol{\theta}}(\mathbf{x}^{(2)}) \\ \vdots \\ h_{\boldsymbol{\theta}}(\mathbf{x}^{(2)}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}^T\mathbf{x}^{(1)} \\ \boldsymbol{\theta}^T\mathbf{x}^{(2)} \\ \vdots \\ \boldsymbol{\theta}^T\mathbf{x}^{(m)} \end{bmatrix} = \mathbf{X}\boldsymbol{\theta}$$

Linear Model

3. 비용 함수

평균 제곱 오차 (Mean Squared Error)

- **Classic form**

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)^2$$

- **Vector form**

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\|\mathbf{h}_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}\|_2^2$$

$$= \frac{1}{2m}\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|_2^2$$

$$= \frac{1}{2m}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

> MSE 말고도 절댓값의 형태를 사용하는 평균 절대 오차 (MAE)도 있으나, 대부분 MSE 사용

4. 최적화

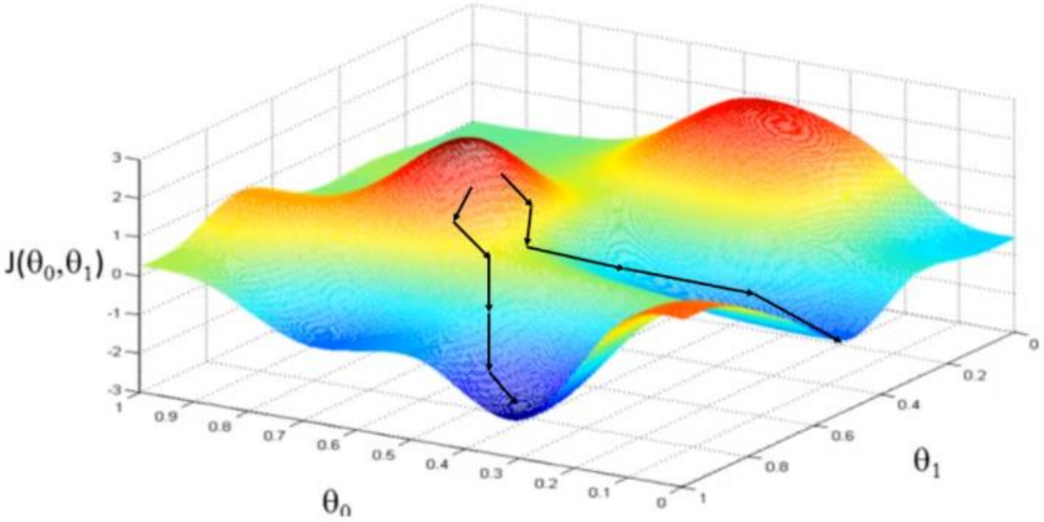
최적화 방법 비교

① 경사 하강법	② 정규 방정식
$\nabla J(\boldsymbol{\theta}) = \frac{1}{m}\mathbf{X}^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$	$\boldsymbol{\theta}^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$
학습률 정할 필요 있음	학습률 정할 필요 없음
많은 반복이 필요	반복이 불필요함(한 번의 연산으로 계산 가능)
시간 복잡도 O(kn^2)	시간 복잡도 O(n^3), X^TX의 역행렬 계산이 필요

> 특징 n이 많아질수록 경사 하강법이 계산에 유리하고, 컴퓨터는 역행렬을 구하는 과정이 불완전함

4. 최적화

경사 하강법

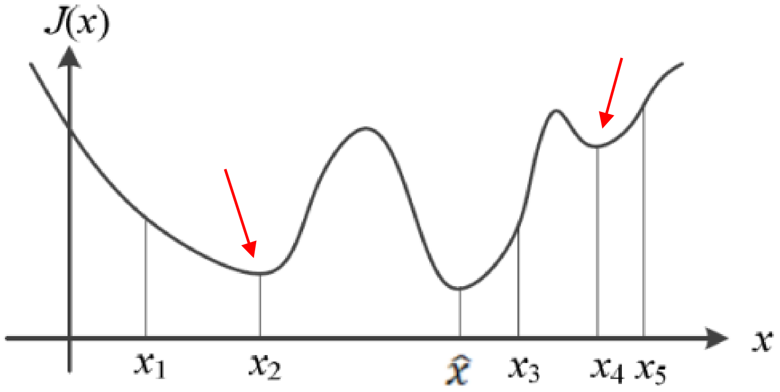


$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

비용 함수에 대해 각 지점에서 미분 값을 구하여, **비용 함수 값이 작아지는 방향으로** 진행하며 파라미터를 업데이트

4. 최적화

지역 최적해 해결 방법



$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} J(\Theta)$$

> 비용 함수를 아래 볼록 함수(2차 함수)로 만든다. -> MSE

> 딥러닝처럼 파라미터를 수 만개 사용하는 경우에는 gradient가 0이 되는 경우가 매우 적다.

4. 최적화

비용 함수와 Gradient

- **Classic form**

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)^2$$

- **Vector form**

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\|\mathbf{h}_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}\|_2^2$$

$$= \frac{1}{2m}\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|_2^2$$

$$= \frac{1}{2m}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

→

- **Classic form**

For j=0,…,n :

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)x_j^{(i)}$$

- **Vector form**

$$\nabla J(\boldsymbol{\theta}) = \frac{1}{m}\mathbf{X}^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

4. 최적화

학습 모드

- **Classic form**

$$\theta_j := \theta_j - \alpha \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)} \right) x_j^{(i)}$$

- **Vector form**

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha \nabla J(\boldsymbol{\theta}) = \boldsymbol{\theta} - \alpha \frac{1}{m} \mathbf{X}^T (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

> 수렴할 때까지 반복

- Online mode

  모든 훈련 모델에 대해서 파라미터를 업데이트
  매 훈련 모델에 적용하기 때문에 잡음이 큼

- Batch mode

  전체 훈련 모델을 마친 후 평균 값으로 파라미터를 업데이트
  속도가 너무 느림

- Mini-batch mode (Stochastic Gradient Descent)

  특정 훈련 모델 set마다 파라미터를 업데이트

4. 최적화

SGD(Stochastic Gradient Descent)

알고리즘 2-5 스토캐스틱 경사 하강 알고리즘(SGD)

입력 : 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$, 학습률 ρ
출력 : 최적해 $\hat{\mathbf{\Theta}}$

난수를 생성하여 초기해 $\mathbf{\Theta}$를 설정한다.
repeat
  $\mathbb{X}$의 샘플의 순서를 섞는다.
  for (i=1 to n)
    i번째 샘플에 대한 그레이디언트 $\mathbf{\nabla}_i$를 계산한다.
    $\mathbf{\Theta} = \mathbf{\Theta} - \rho\mathbf{\nabla}_i$
until(멈춤 조건)
$\hat{\mathbf{\Theta}} = \mathbf{\Theta}$

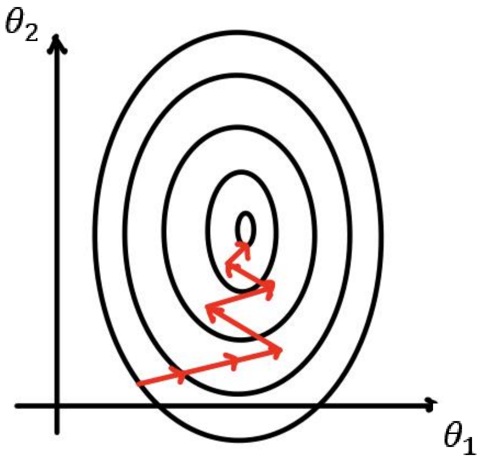
- **훈련 data들의 순서에 따른 의존을 방지**

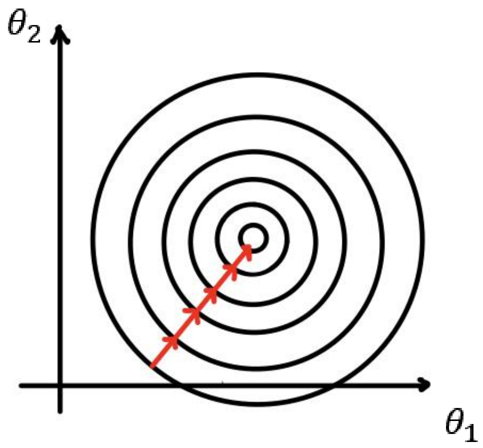
  ① Random Shuffling
  순서를 뒤섞음

  ② Random Sampling
  복원 추출(중복 허용)

5. 정규화, 규제

정규화





- **Sklearn의 StandardScaler**

```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
X_scaled = ss.fit_transform(X_train)
```

Gradient는 **feature의 범위에 비례**하기 때문에, 각 변수에 따른 gradient 값이 너무 다르다

> 결과적으로 수렴 속도가 느려지므로, 정규화 필요

5. 정규화, 규제

규제

- **릿지 회귀 (Ridge Regression, L_2-norm regularization)**

$$\hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

➡ 필요 없는 특성들의 회귀계수를 감소시키며 영향력을 낮춤

- **라쏘 회귀 (Lasso Regression, L_1-norm regularization)**

$$\hat{\beta}^{lasso} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

➡ 학습에 필요 없는 일부 특성들을 0으로 만들어 변수 선택

규제를 통해, 훈련 세트의 **과대 적합을 방지**하고 테스트 세트에서의 좋은 성능을 얻음

5. 정규화, 규제

규제

- L_2-norm regularization

Regularized Cost & Gradient

$$\underbrace{J_{regularized}(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y})}_{\text{규제를 적용한 목적함수}} = \underbrace{J(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y})}_{\text{목적함수}} + \lambda \underbrace{\|\mathbf{\Theta}\|_2^2}_{\text{규제 항}}$$

$$\nabla J_{regularized}(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y}) = \nabla J(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y}) + \underline{2\lambda\mathbf{\Theta}}$$

Parameter Update

$$\begin{aligned}\mathbf{\Theta} &= \mathbf{\Theta} - \rho \nabla J_{regularized}(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y}) \\ &= \mathbf{\Theta} - \rho(\nabla J(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y}) + 2\lambda\mathbf{\Theta}) \\ &= \boxed{(1 - 2\rho\lambda)}\mathbf{\Theta} - \rho \nabla J(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y})\end{aligned}$$

→ 필요 없는 특성들의 회귀계수를 감소시키며 영향력을 낮춤

- L_1-norm regularization

Regularized Cost & Gradient

$$\underbrace{J_{regularized}(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y})}_{\text{규제를 적용한 목적함수}} = \underbrace{J(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y})}_{\text{목적함수}} + \lambda \underbrace{\|\mathbf{\Theta}\|_1}_{\text{규제 항}}$$

$$\nabla J_{regularized}(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y}) = \nabla J(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y}) + \underline{\lambda \mathbf{sign}(\mathbf{\Theta})}$$

Parameter Update

$$\begin{aligned}\mathbf{\Theta} &= \mathbf{\Theta} - \rho \nabla J_{regularized}(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y}) \\ &= \mathbf{\Theta} - \rho(\nabla J(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y}) + \lambda \mathbf{sign}(\mathbf{\Theta})) \\ &= \mathbf{\Theta} - \rho \nabla J(\mathbf{\Theta}; \mathbb{X}, \mathbb{Y}) \boxed{- \rho\lambda \mathbf{sign}(\mathbf{\Theta})}\end{aligned}$$

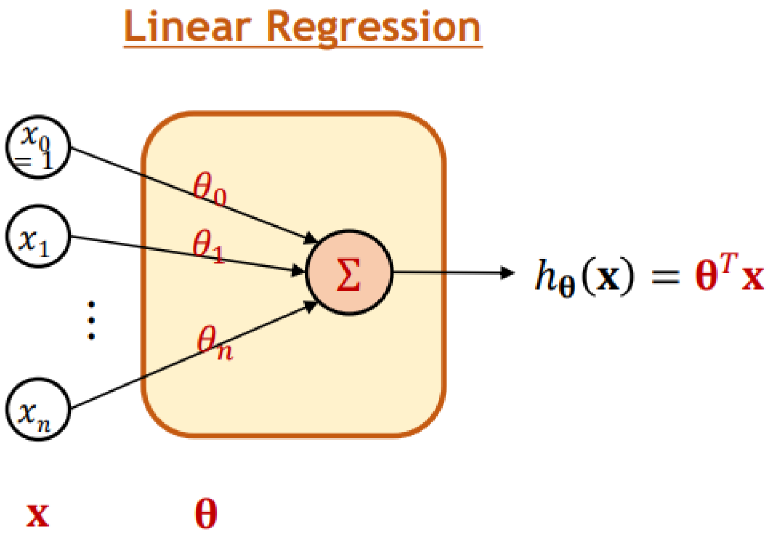
→ 학습에 필요 없는 일부 특성들을 0으로 만들어 변수 선택

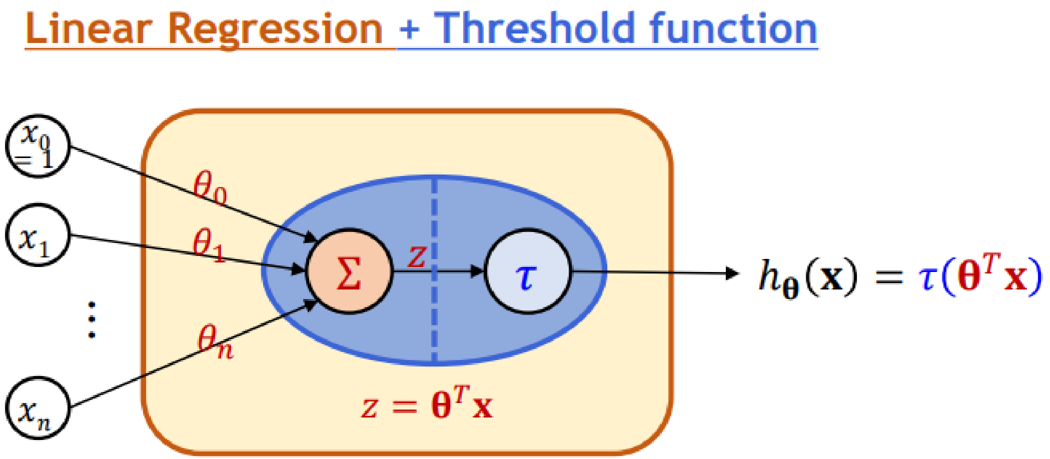
로지스틱 회귀

1. 선형 회귀의 확장
2. 로지스틱 회귀 모델
3. 비용 함수
4. 최적화

1. 선형 회귀의 확장

선형 회귀를 로지스틱 회귀로 확장

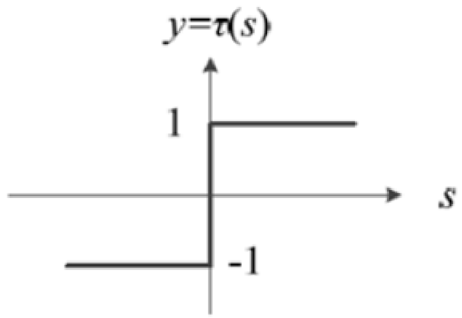




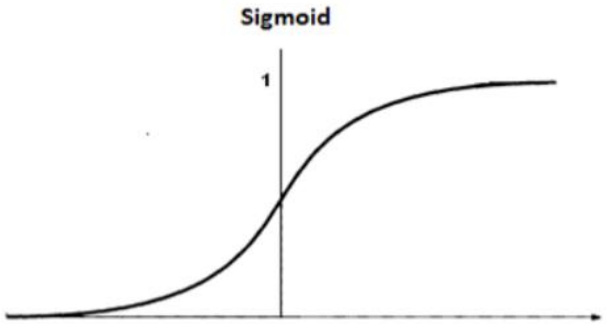
> 선형 회귀 모델에 활성 함수를 추가하여, **특정 범주로 분류될 확률**을 출력

2. 로지스틱 회귀 모델

활성 함수



$$\tau(s) = \begin{cases} 1 & s \geq 0 \\ -1 & s < 0 \end{cases}$$



$$sigmoid(z_j) = \frac{1}{1 + \exp(-z_j)}$$

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for j = 1,...,K}$$

① 계단 함수 (퍼셉트론, MLP)

결과값으로 -1과 1 또는 0과 1 만을 가지는 함수

② Sigmoid 함수 (이진 분류)

결과로 0과 1 사이의 값을 가지는 함수

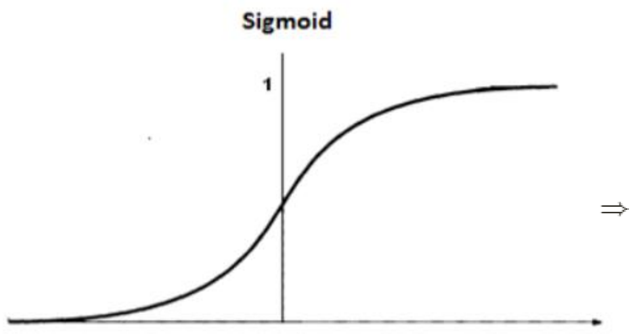
③ Softmax 함수 (다중 분류)

다중 분류에서 출력 결과를 정규화하여.
전체 합이 1이 되도록 만드는 함수

* 이외에도 ReLU, 소프트 플러스, 하이퍼볼릭 탄젠트 등의 활성 함수들이 존재

2. 로지스틱 회귀 모델(이진 분류)

활성 함수 (Sigmoid 함수)



$\Rightarrow 0 \leq h_\theta(x) \leq 1$

$$sigmoid(z_j) = \frac{1}{1 + \exp(-z_j)}$$

- **선형 모델 :** $z = \boldsymbol{\theta}^T \boldsymbol{x}$
- **로지스틱 함수 (활성 함수) :** $h_\theta(z) = \sigma(z) = \dfrac{1}{1 + e^{-z}}$

> $h_\theta(z)$는 이진 분류에서 결과가 1일 **확률**을 나타낸다.

3. 비용 함수

MSE 안 쓰는 이유

- **선형 함수 :** $h_\theta(x) = h_\theta(x_1, ...x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 ..... + \theta_n x_n$
- **로지스틱 함수 (활성 함수) :** $h_\theta(x) = \sigma(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$

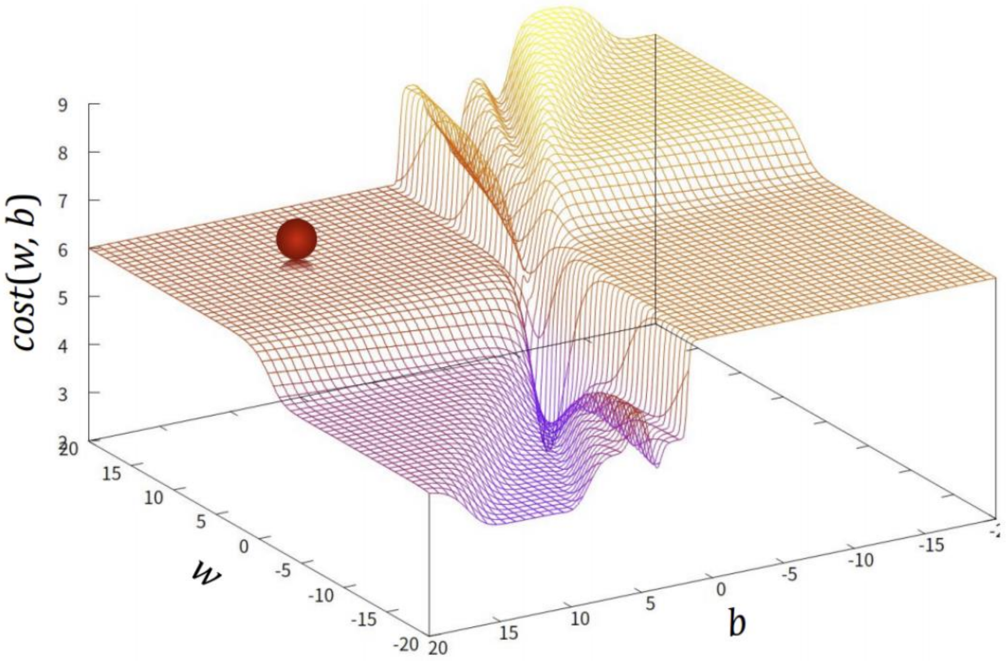
- **MSE**

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2$$

로지스틱 회귀 모델에서는 MSE를 써도 파라미터에 대한 함수가 아래 볼록 함수 형태(Convex)가 아님

3. 비용 함수

MSE 안 쓰는 이유



- **가설로 sigmoid 함수, 비용 함수로 MSE를 적용한 모델**

옆 그림처럼 평평한 부분이 많기 때문에, 아무리 학습을 많이 하더라도 MSE를 사용해서 구현한다면 경사 하강법을 제대로 적용하기가 어려움

> **Cross-Entropy(CE) 비용함수를 사용**

3. 비용 함수

Cross-Entropy (CE)

정답이 1인 경우 정답이 0인 경우

- **Classic form :** $J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$

확률 정답 값 $h_\theta(x^{(i)})$ 의 정보량

- **Vector form:** $J(\boldsymbol{\theta}) = -\frac{1}{m}(\boldsymbol{y}^T\log(\boldsymbol{h}) + (\boldsymbol{1}-\boldsymbol{y})^T\log(\boldsymbol{1}-\boldsymbol{h}))$

$I(x) = log(\frac{1}{p(x)}) = - log(p(x))$

- **Entropy : 평균 정보량**

X가 발생할 수 있는 모든 경우에 대한 정보량의 평균
X의 발생 확률이 높으면, 정보량이 감소

Cross-Entropy 비용 함수는 로그 함수의 형태라서, 0에서 1사이에서 **볼록 함수 (convex)의 형태를 나타냄**

4. 최적화

경사 하강법

CE에 대한 Gradient

- **Classic form :** $\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(\sigma\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$ for j=0,…,n
- **Vector form:** $\nabla J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left(\sigma\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\right) - y^{(i)}\right)\mathbf{x}^{(i)} = \frac{1}{m}\mathbf{X}^T\left(\sigma(\mathbf{X}\boldsymbol{\theta}) - \mathbf{y}\right)$

MSE에 대한 Gradient

- **Classic form :** $\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right) x_j^{(i)}$ For j=0,…,n

> 선형 회귀와 로지스틱 회귀 모두 gradient 값이 비슷한 형태를 나타냄