

분류 & 회귀 & 학습



19기 분석 오효근

목차

선형/다항 회귀

회귀모형, 정규방정식

경사 하강법

배치경사하강법,
확률적경사하강법,
미니배치경사하강법

학습 곡선 & 규제

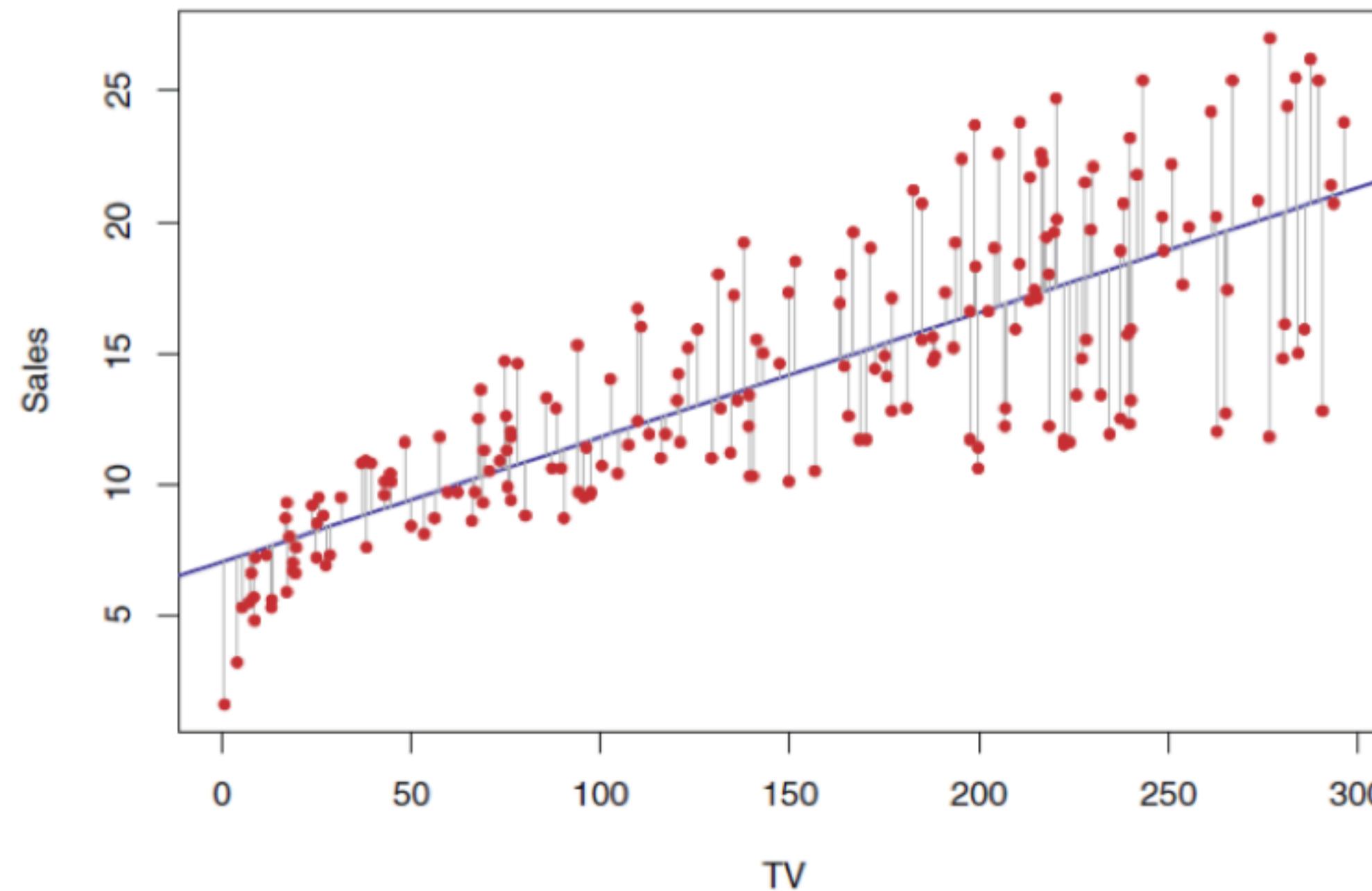
과대적합, 과소적합
분산/편향 트레이드오프
릿지, 라쏘, 엘라스틱넷,
조기 종료

분류 회귀

로지스틱 회귀,
소프트맥스 회귀

회귀

회귀의 개념, 모형식, 목적



TV(설명변수)에 대한
Sales(반응변수)의 회귀모형

회귀

: 변수들간 함수관계를 탐색하는 방법
반응변수(target) ~ 설명변수(input)

회귀 모형

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$
$$\hat{y} = h_{\beta}(x) = \boldsymbol{\beta} \cdot \mathbf{x}$$

목적

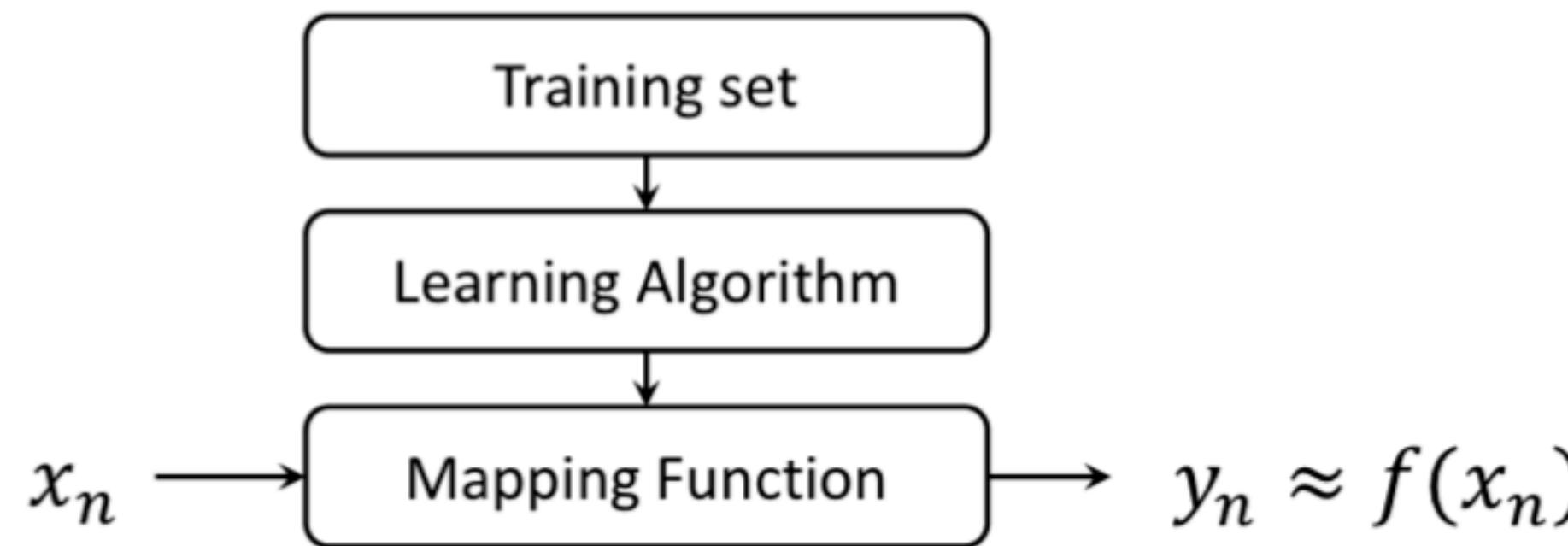
- 1) 예측: 새로운 input x 에 대해 output 예측
- 2) 해석: output에 대한 input의 영향 이해

이를 위해 x 와 y 의 관계를 잘 설명하는 모델을 찾아야 함!
예측값과 실제값의 차이 즉, 잔차가 작아야 ♡

좋은 모델?

회귀의 비용함수

$$y_n \approx f(x_n), \text{ for all } n$$



회귀 모형의 예측값과 실제값이 근사하도록!

잔차를 최소화하는 최적해 β^* 찾기

비용함수 MSE *mean squared error*

: 모델이 얼마나 데이터를 잘 설명하고 있는지 측정; how costly our mistakes are

$$MSE(\mathbf{X}, h_{\beta}) = \frac{1}{m} \sum_{i=1}^m (\beta^T \mathbf{x}^{(i)} - y^{(i)})^2$$

최적화 목표

$$\beta^* = \operatorname{argmin}_{\beta} MSE(\mathbf{X}, h_{\beta}) = \operatorname{argmin} \frac{1}{m} \sum_{i=1}^m (\beta^T \mathbf{x}^{(i)} - y^{(i)})^2$$

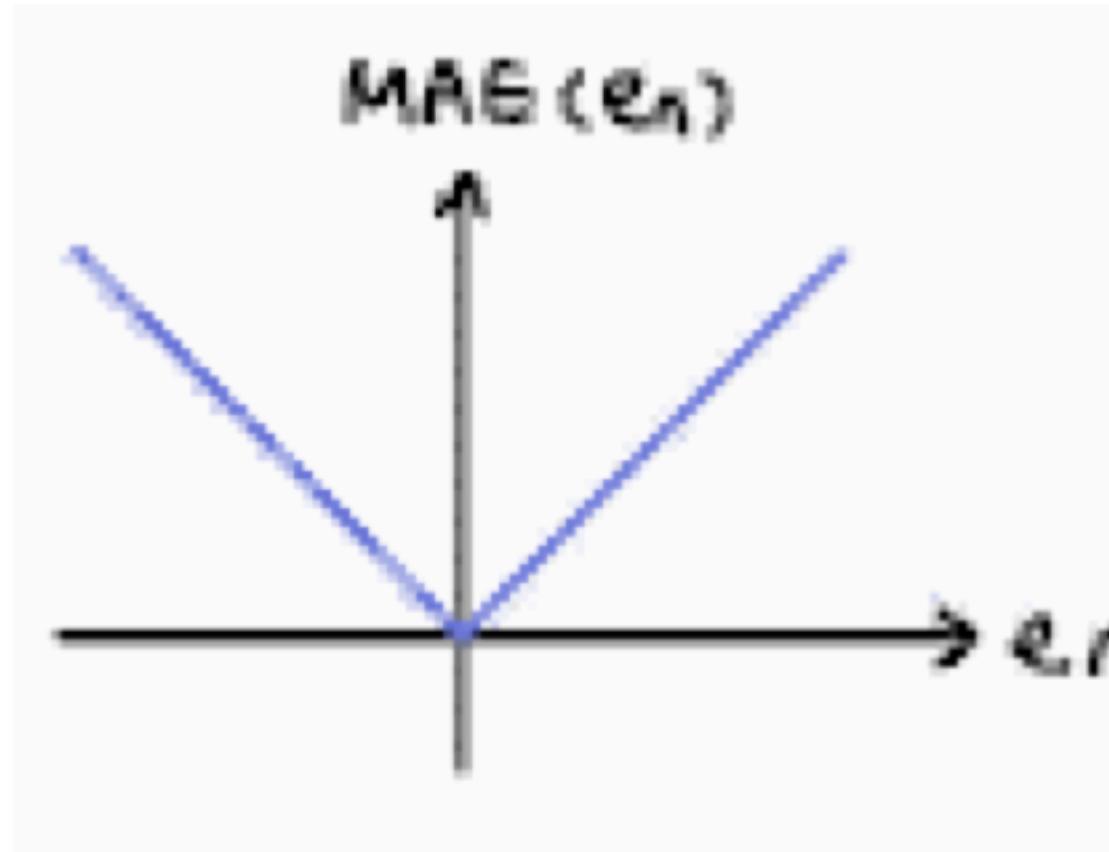
정규방정식

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X} \mathbf{y} = \mathbf{X}^+ \mathbf{y}$$

데이터에 가장 잘 맞는 모델 파라미터 (즉, 데이터에 대해 비용 함수를 최소화하는 모델 파라미터)
공식으로 계산해서 도출하는 해석적 방법

좋은 모델?

회귀의 비용함수



MAE는 l1 norm을 이용한 error
미분 불가능

좋은 비용 함수의 기준

- ✓ 0을 기준으로 symmetric 해서
+, - error들이 동등하게 penalized
- ✓ 큰 error와 매우 큰 error가 비슷하게 penalized
- ✓ 미분 가능

비용함수 MAE *mean absolute error*

: 모델이 얼마나 데이터를 잘 설명하고 있는지 측정; how costly our mistakes are

$$MAE(\mathbf{X}, h_{\beta}) = \frac{1}{m} \sum_{i=1}^m |\beta^T \mathbf{x}^{(i)} - y^{(i)}|$$

최적화 목표

$$\boldsymbol{\beta}^* = \operatorname{argmin}_{\beta} MSE(\mathbf{X}, h_{\beta}) = \operatorname{argmin} \frac{1}{m} \sum_{i=1}^m (\beta^T \mathbf{x}^{(i)} - y^{(i)})^2$$

정규방정식

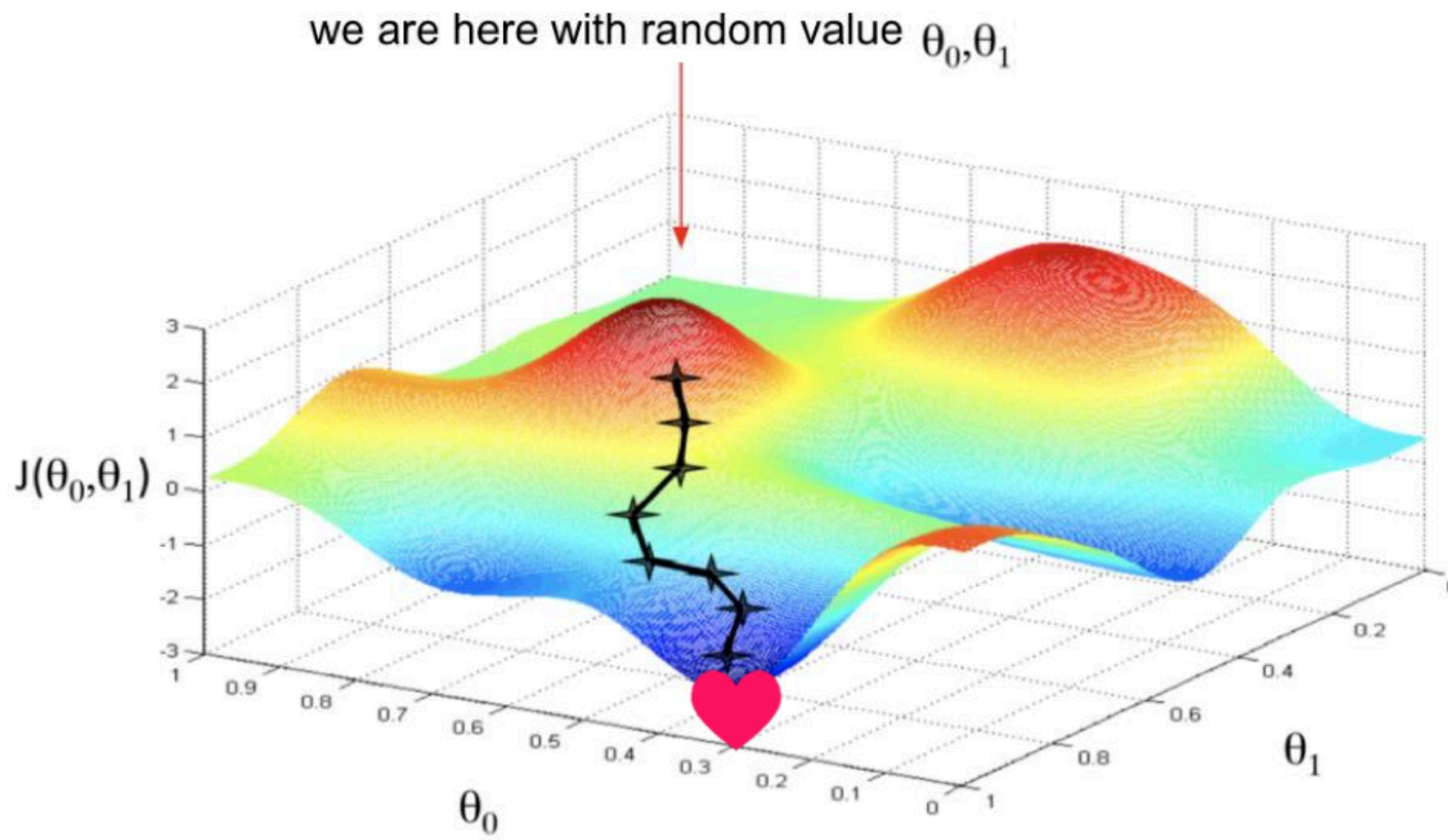
$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X} \mathbf{y} = \mathbf{X}^+ \mathbf{y}$$

데이터에 가장 잘 맞는 모델 파라미터 (즉, 데이터에 대해 비용 함수를 최소화하는 모델 파라미터)
공식으로 계산해서 도출하는 해석적 방법

경사 하강법

Gradient descent; GD

여러 종류의 문제에서 최적의 해법을 찾을 수 있는 일반적인 알고리즘



Follow the Gradient!

파라미터 벡터 β 에 대해 비용 함수의 gradient 계산
→ gradient가 감소하는 방향으로 진행
→ gradient가 0이 되면 최솟값에 도달한 것

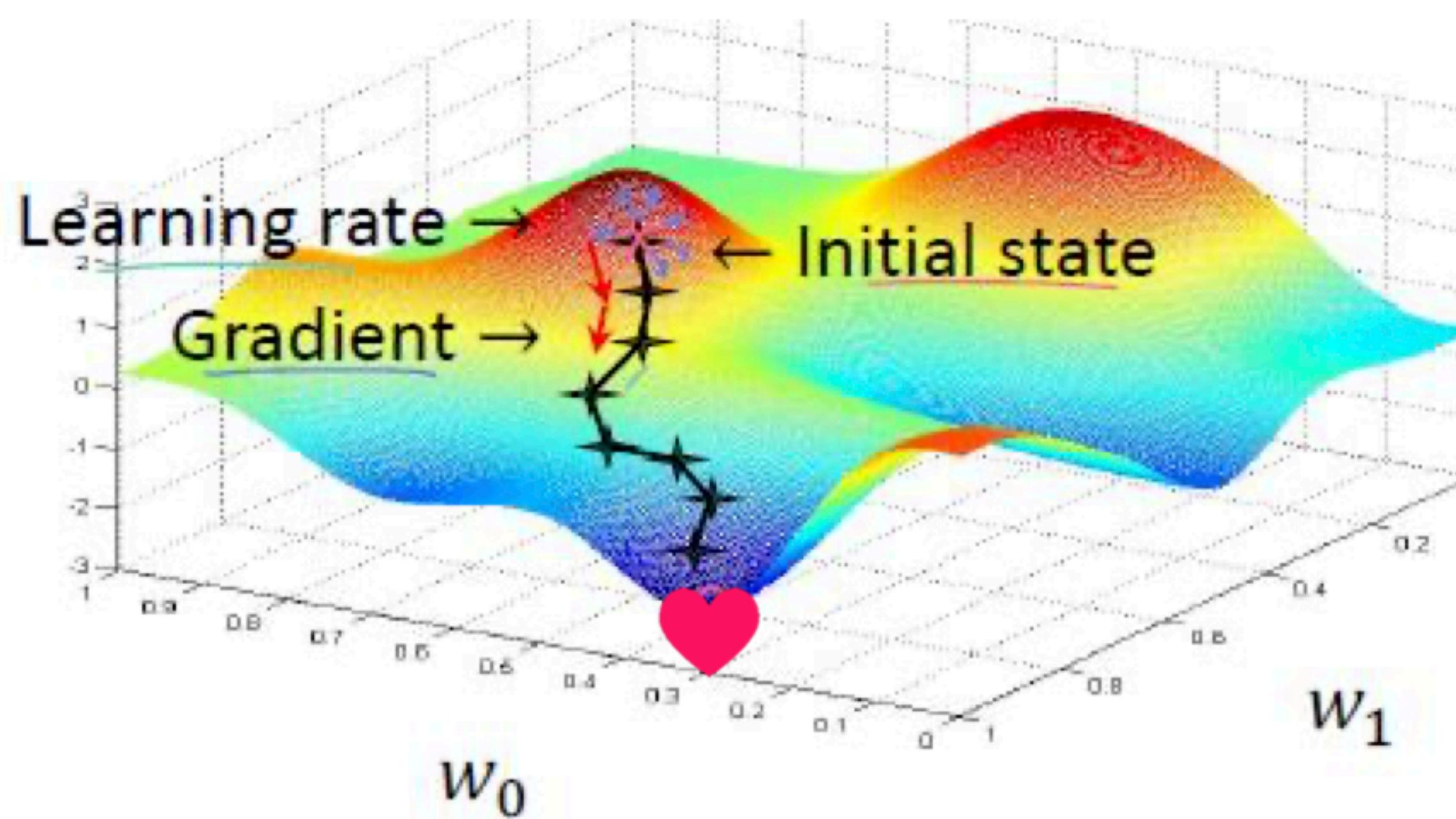
1. 어느 방향으로?
2. 어디서 시작?
3. 얼마나 이동?

발리 골짜기로 내려가는 좋은 방법은 가장 가파른 길을 따라 아래로 내려가는 것!
→ 그레이디언트

경사 하강법

Gradient descent; GD

여러 종류의 문제에서 최적의 해법을 찾을 수 있는 일반적인 알고리즘



발리골짜기로 내려가는 좋은 방법은 가장 가파른 길을 따라 아래로 내려가는 것!
→ 그레이디언트



Follow the Gradient!

파라미터 벡터 β 에 대해 비용 함수의 gradient 계산
→ gradient가 감소하는 방향으로 진행
→ gradient가 0이 되면 최솟값에 도달한 것

1. 어느 방향으로?

Gradient

2. 어디서 시작?

Random initialization

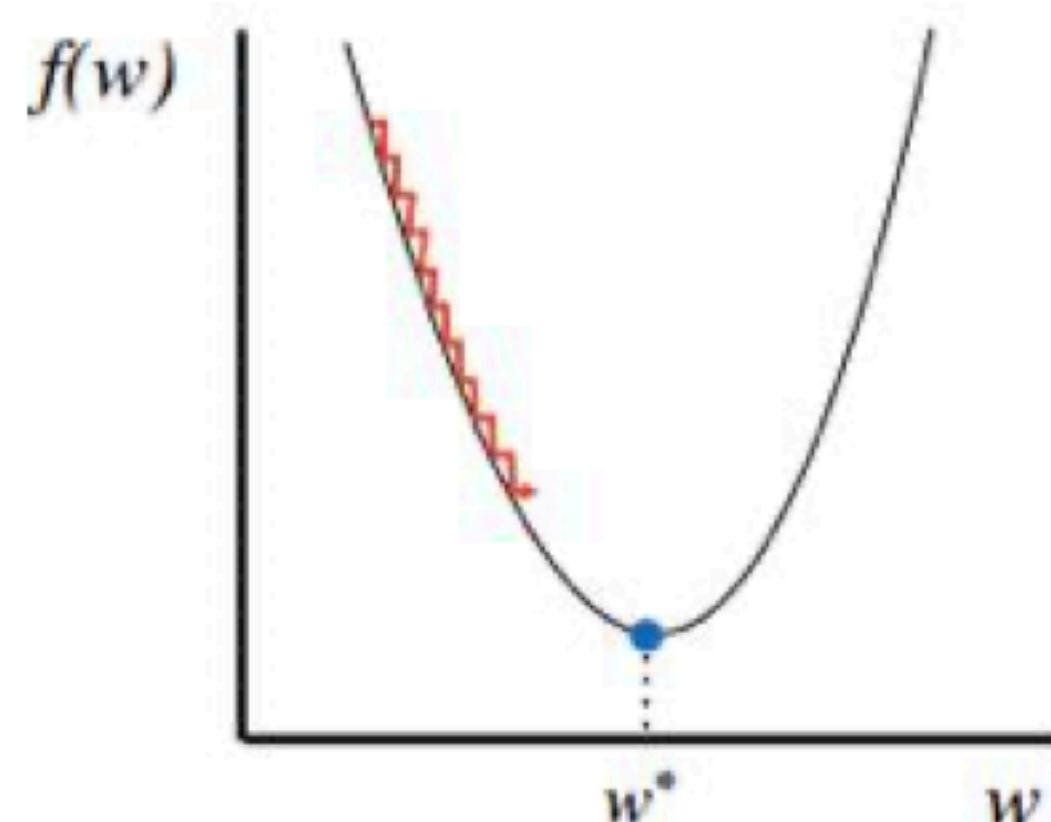
3. 얼마나 이동?

Learning rate

경사 하강법

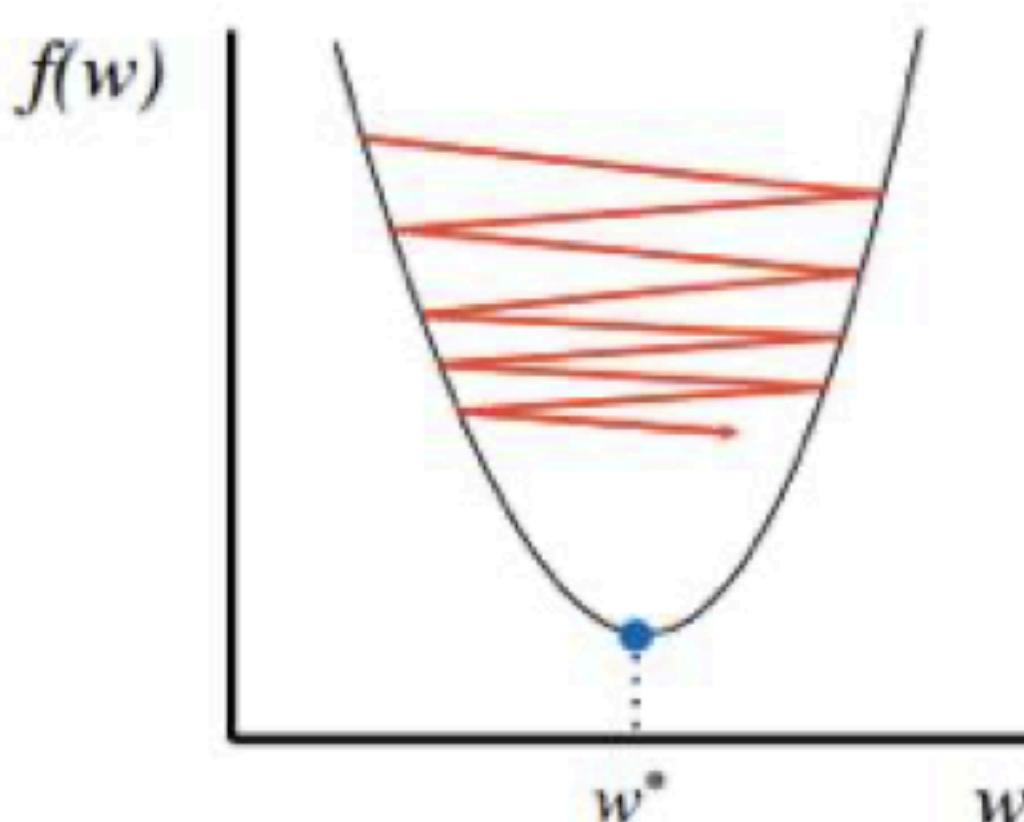
Gradient descent, GD

여러 종류의 문제에서 최적의 해법을 찾을 수 있는 일반적인 알고리즘



Too small: converge
very slowly

- 1) 너무 작으면 시간 오래 걸림



Too big: overshoot and
even diverge

- 2) 너무 크면 발산해서 최적해 못 찾을지도

→ 시작할 때는 α 를 크게하고 점진적으로 감소!! “learning schedule”

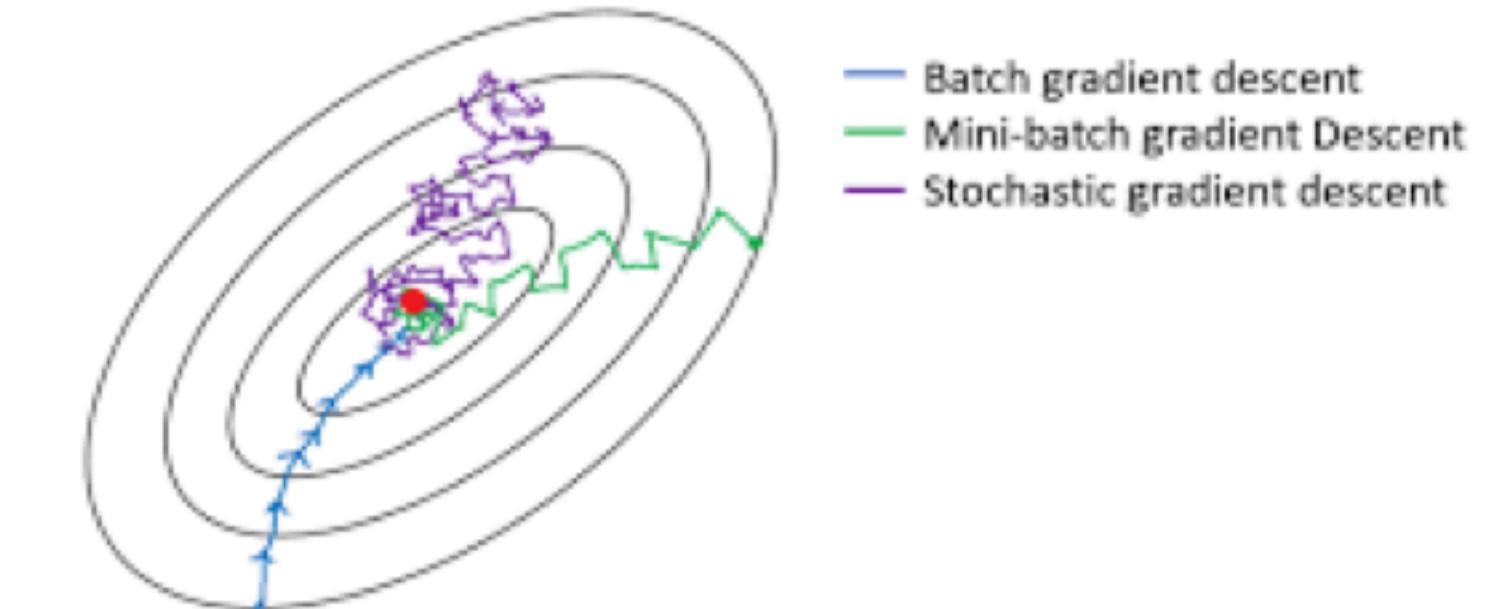


Follow the Gradient!

1. 그레이디언트 *gradient*: (+) gradient 이면 (-) 곡, (-) gradient 이면 (+) 곡
2. 무작위 초기화 *random initialization*: 시작되는 임의의 β 값
3. 학습률 *learning rate*: 스텝의 크기 α

경사 하강법

feature이 매우 많고 훈련 샘플이 너무 많아
메모리에 모두 담을 수 없을 때 적합

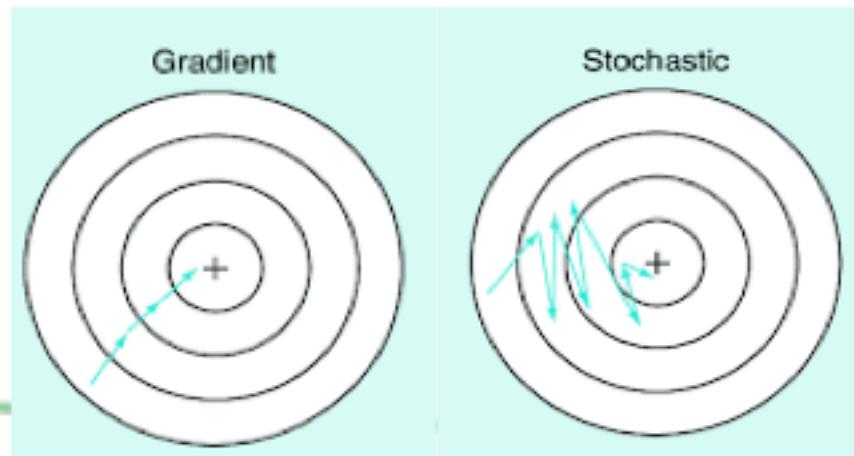


배치 경사 하강법

```
Repeat until convergence {  
     $\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$   
}
```

$$\frac{\partial}{\partial \theta_j} \text{MSE}(\theta) = \frac{2}{m} \sum_{i=1}^m (\theta^T \mathbf{x}^{(i)} - y^{(i)}) x_j^{(i)}$$

- ✓ 매 경사 하강법 스텝에서 전체 훈련셋에 대해 계산
→ 매우 큰 데이터에서는 느림, m에 민감 x



확률적 경사 하강법 SGD

```
For N Epochs through training set  $\{(x_n, y_n)\}_{n=1}^N$ {  
     $w \leftarrow w - \alpha \frac{\partial}{\partial w} L(w)$   
}
```

* “1 epoch”: pass through training set.

- ✓ 매 스텝에서 한 개의 샘플을 무작위로 선택, 그에 대한 그레이디언트 계산
→ 계산 빠름, 매우 큰 훈련셋에서도 가능
- ✓ 에폭 epoch x m 번 반복
- ✓ 확률적 → 불안정, 최적값은 아닐 수도 ↗👎
- ✓ 지역 최솟값에서 탈출 가능성 ↑ 👍
- 학습 스케줄 learning schedule



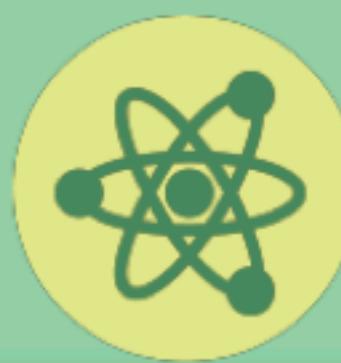
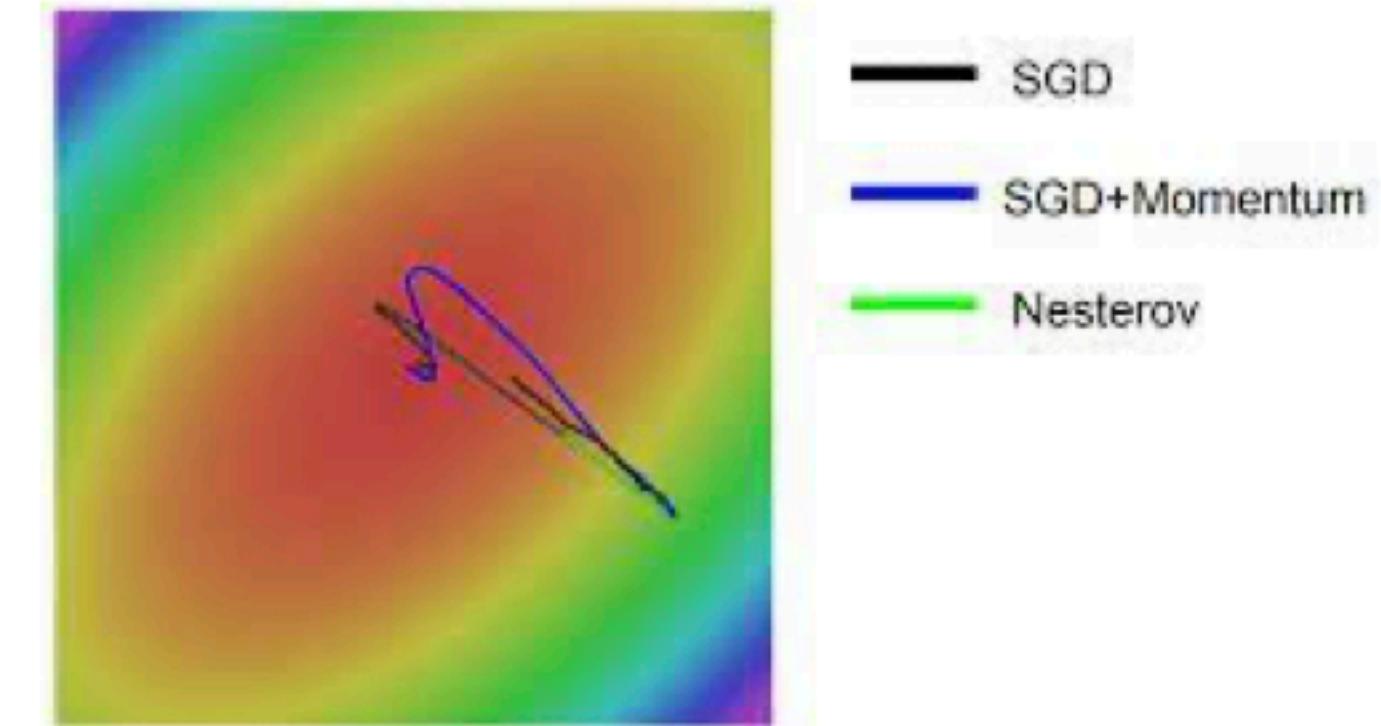
미니배치 경사 하강법

```
For N Epochs{  
    For each training batch  $\{(x_b, y_b)\}_{b=1}^B$ {  
         $w \leftarrow w - \alpha \frac{\partial}{\partial w} L(w)$   
    }  
}
```

- ✓ 임의의 작은 샘플 세트, 미니배치에 대해 그레이디언트 계산
- ✓ 미니배치를 어느 정도 크게 하면 파라미터 공간에서 SGD보다 덜 불규칙
→ 최적해에 근사 ↑, localmin 탈출 ↓
- ✓ GPU로 성능 향상 👍

확률적 경사 하강법

문제점과 대안



확률적 경사 하강법 SGD

For N Epochs through training set $\{(x_n, y_n)\}_{n=1}^N$

$$w \leftarrow w - \alpha \frac{\partial}{\partial w} L(w)$$

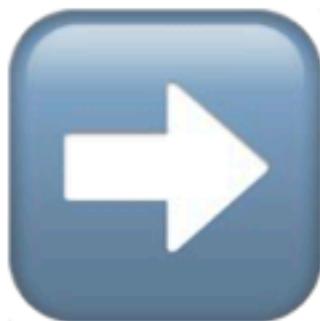
속도
위치 α 를 미분

* “1 epoch”: pass through training set.

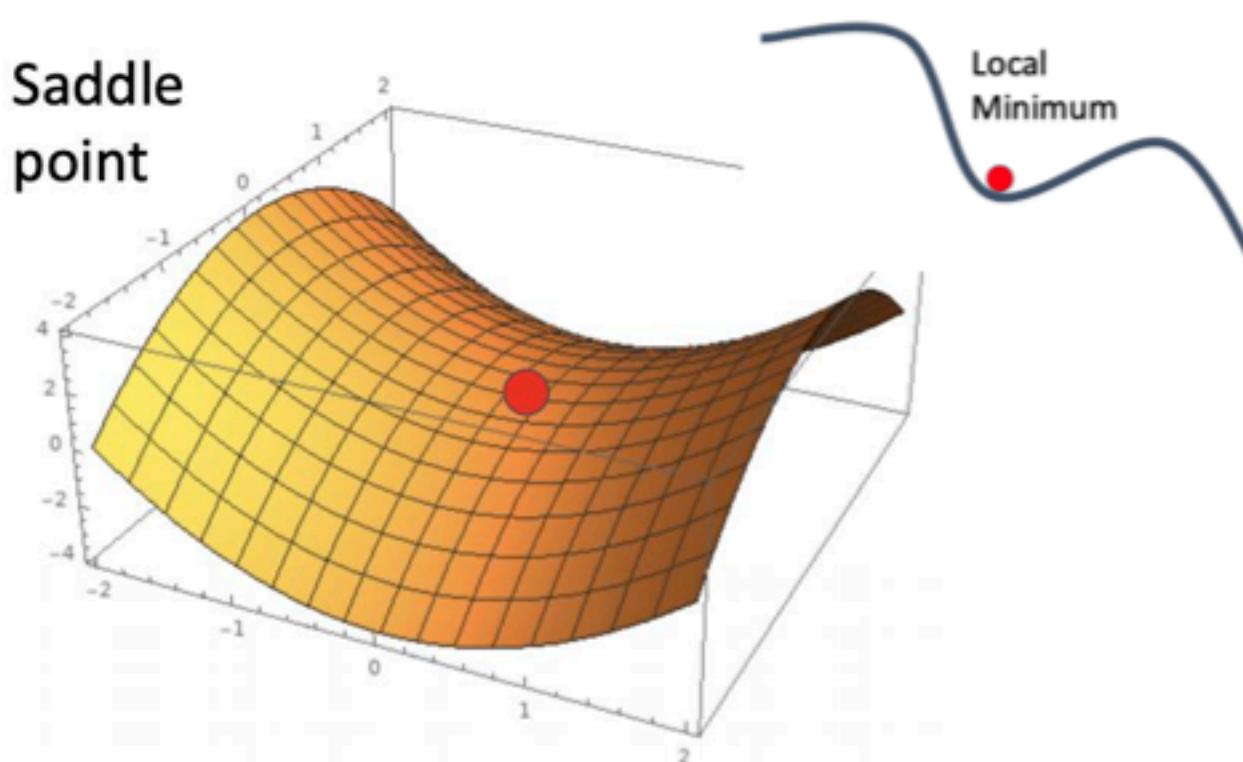
- ✓ 매 스텝에서 한 개의 샘플을 무작위로 선택, 그에 대한 그레이디언트 계산 → 계산 빠름, 매우 큰 훈련 셋에서도 가능
- ✓ 에폭 epoch x m 번 반복

- ✓ 확률적 → 불안정, 최적값은 아닐 수도
- ✓ 지역 최솟값에서 탈출 가능성 ↑
- 학습 스케줄 learning schedule

파라미터 간 스케일 다를 때 불안정



Saddle point에 빠지기 쉬움



1. SGD+Momentum

- ✓ 가속도 개념 도입

$$v_{t+1} = \rho v_t - \alpha \nabla f(w_t)$$

velocity update

$$w_{t+1} = w_t + v_{t+1}$$

weight update

2. Nesterov Momentum

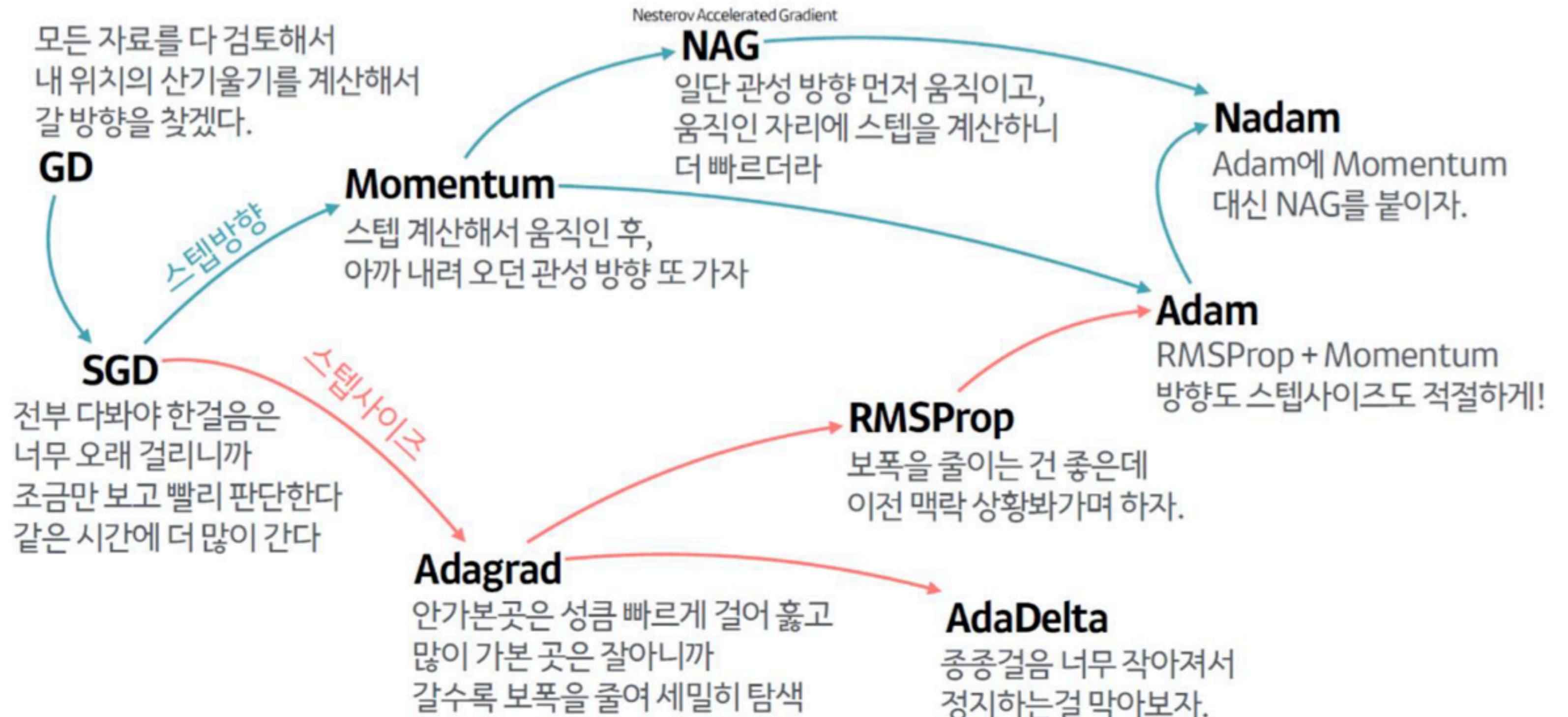
- ✓ 가속도 한번 step 가서 계산 → actual update

$$v_{t+1} = \rho v_t - \alpha \nabla f(w_t + \rho v_t)$$

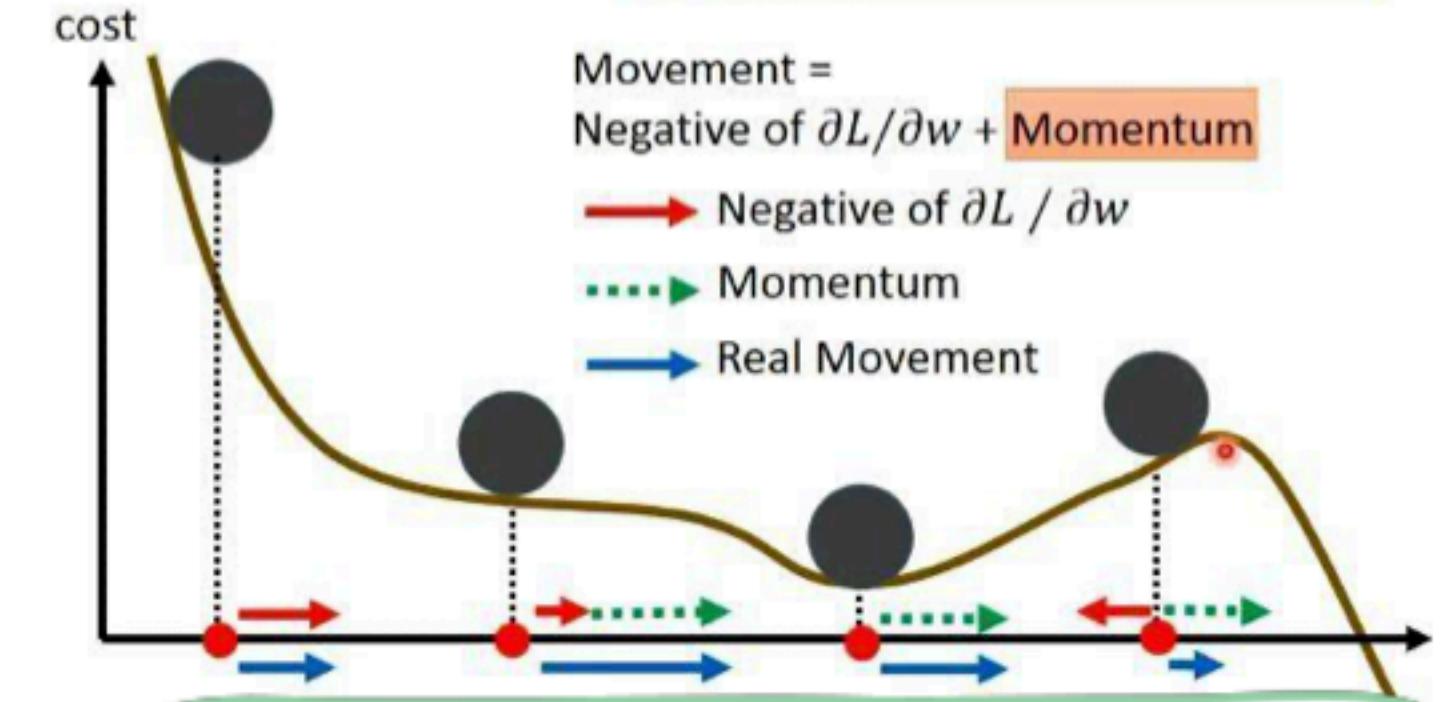
$$w_{t+1} = w_t + v_{t+1}$$

확률적 경사 하강법

문제점과 대안



Momentum



1. SGD + Momentum

- ✓ 가속도 개념 도입

$$v_{t+1} = \rho v_t - \alpha \nabla f(w_t)$$

$$w_{t+1} = w_t + v_{t+1}$$

velocity update

2. Nesterov Momentum

- ✓ 가속도 한번 step 가서 계산 → actual update

$$v_{t+1} = \rho v_t - \alpha \nabla f(w_t + \rho v_t)$$

$$w_{t+1} = w_t + v_{t+1}$$

weight update

분산 편향 트레이드 오피

회귀의 개념, 모형식, 목적

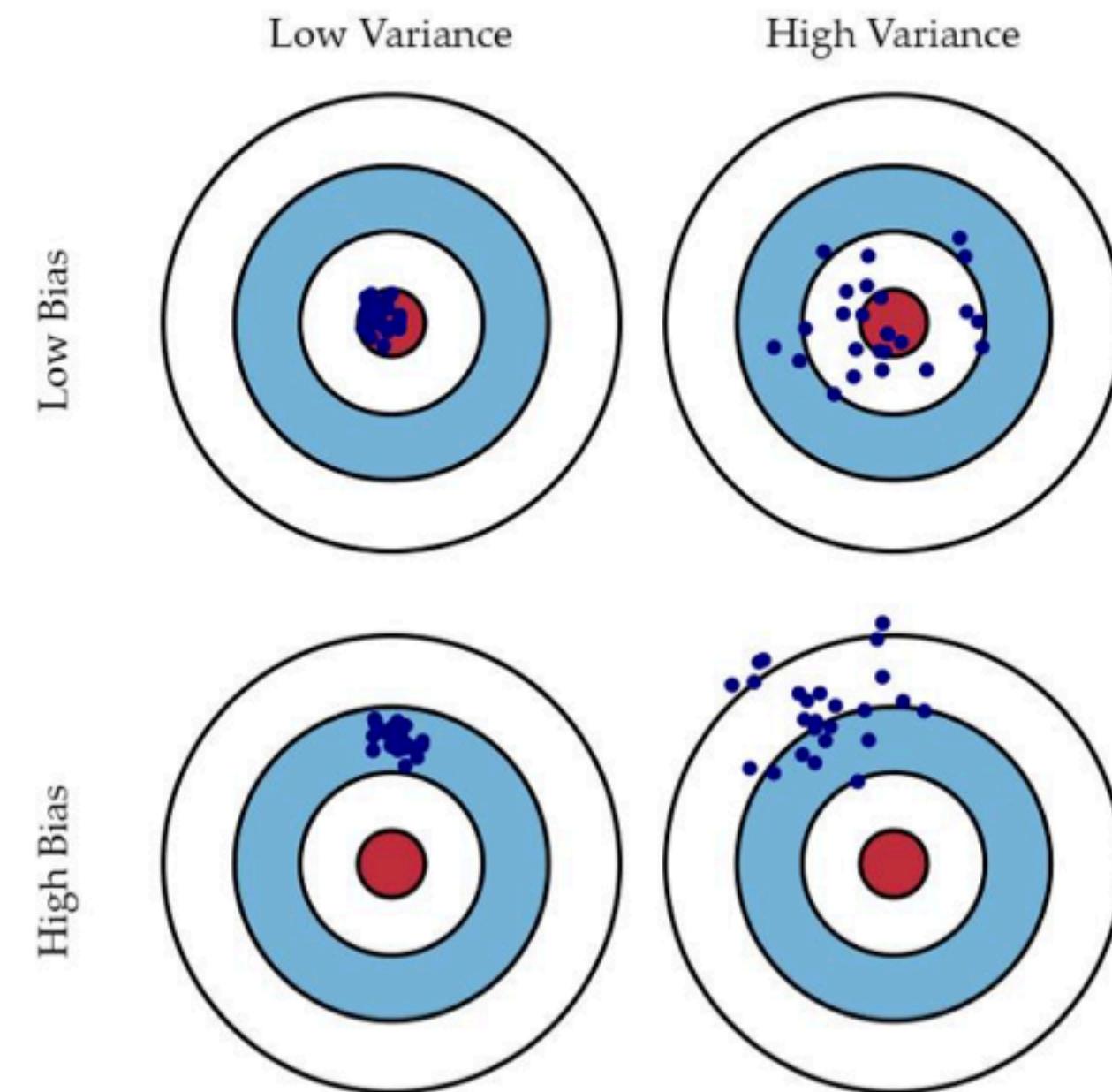
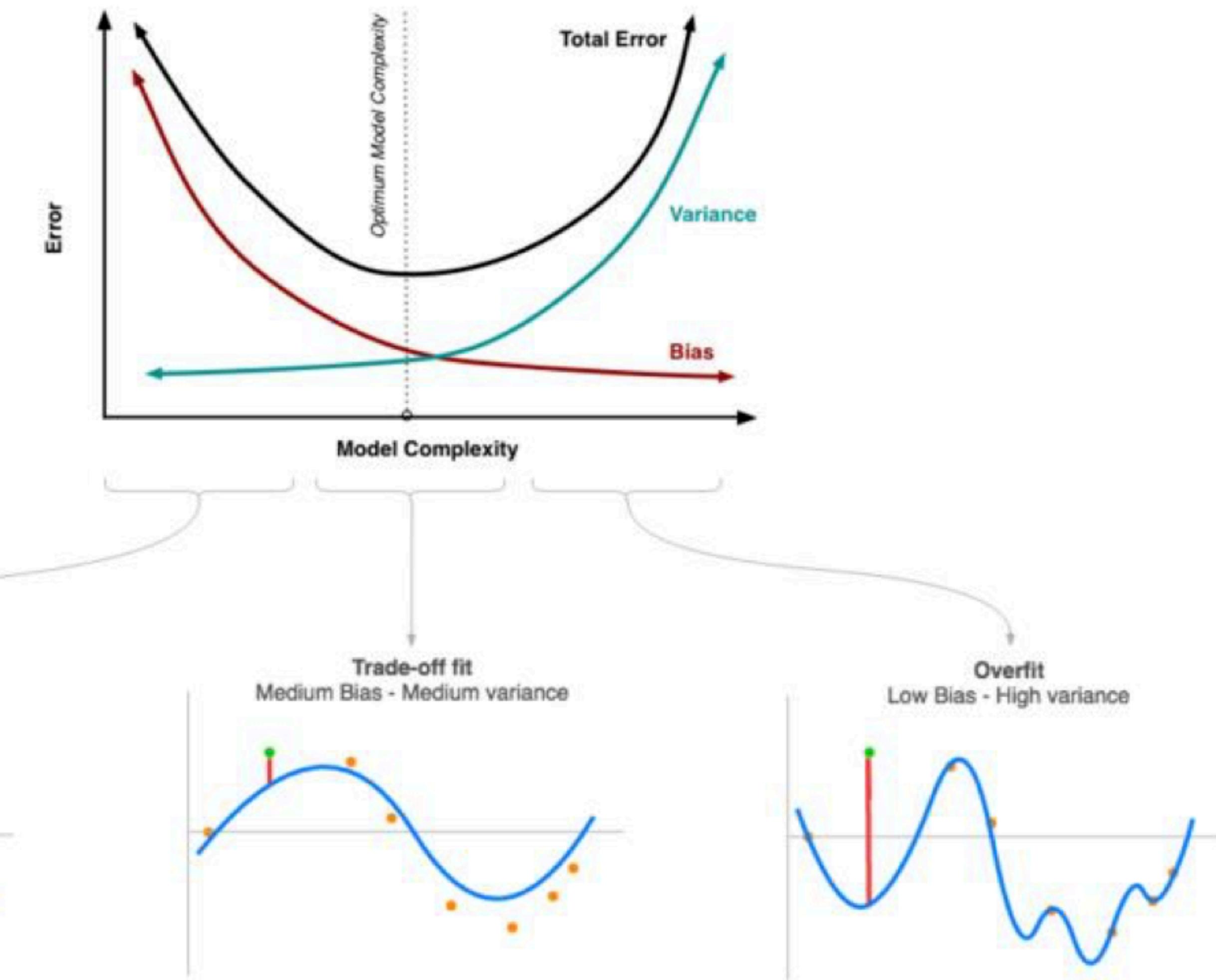
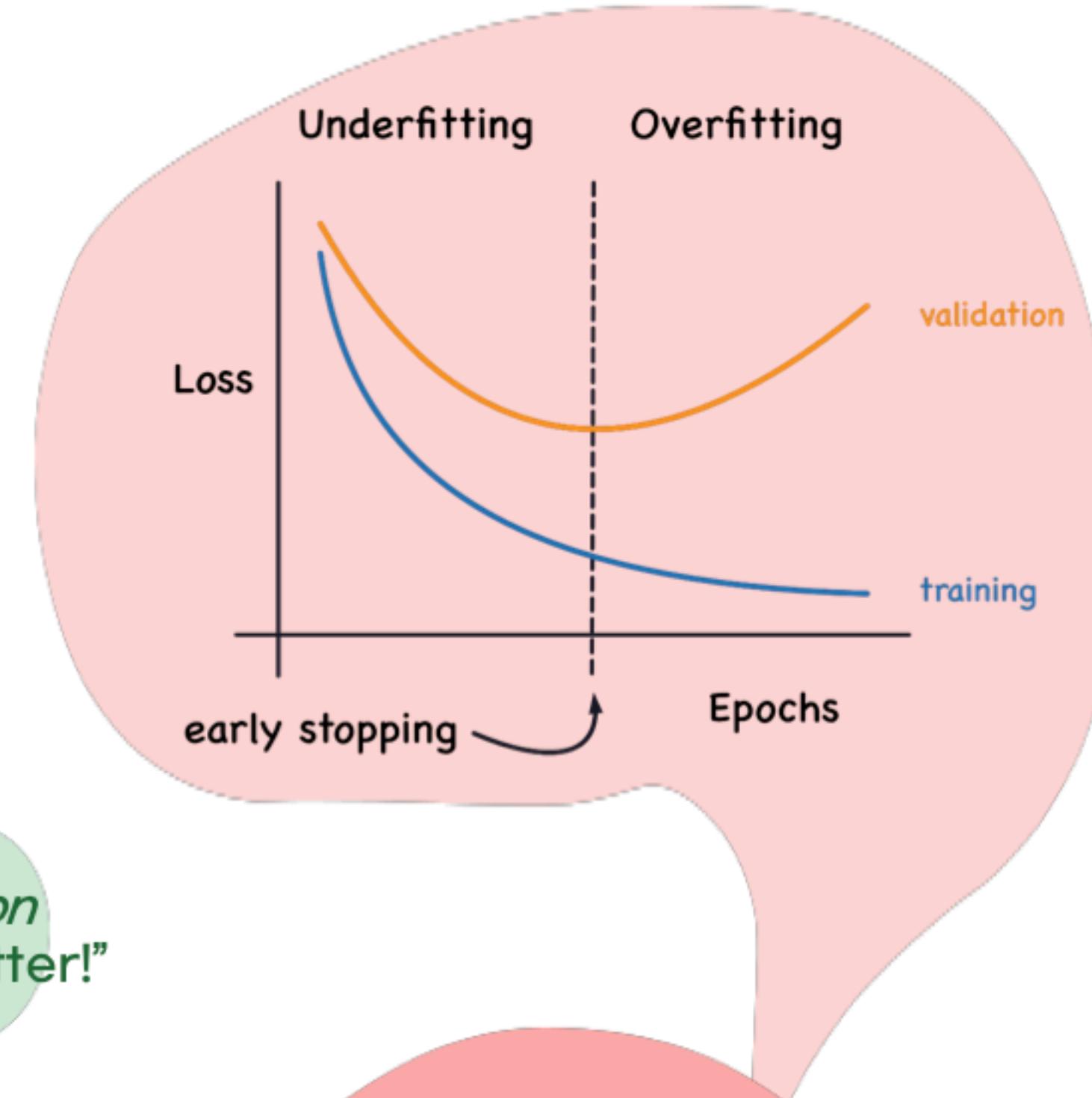


Fig. 1 Graphical illustration of bias and variance.



과소·과대적합



이상치의 영향 ↓
데이터 하나하나에 의존도 ↓

변수 선택 *model selection*
“Simple models are better!”

훈련 데이터셋
추가

차수
줄이기

과대적합
된다면?!
overfit

과적합 발생 전 학습 중지

조기종료

릿지(L2), 라쏘(L1),
엘라스틱넷

규제
regularization

규제

: 가중치의 절댓값을 가능한 작게 만듦
큰 가중치에 대해서 패널티 부과

$$\underset{w}{\operatorname{argmin}}(L(w) + \Omega(w))$$

regualizer

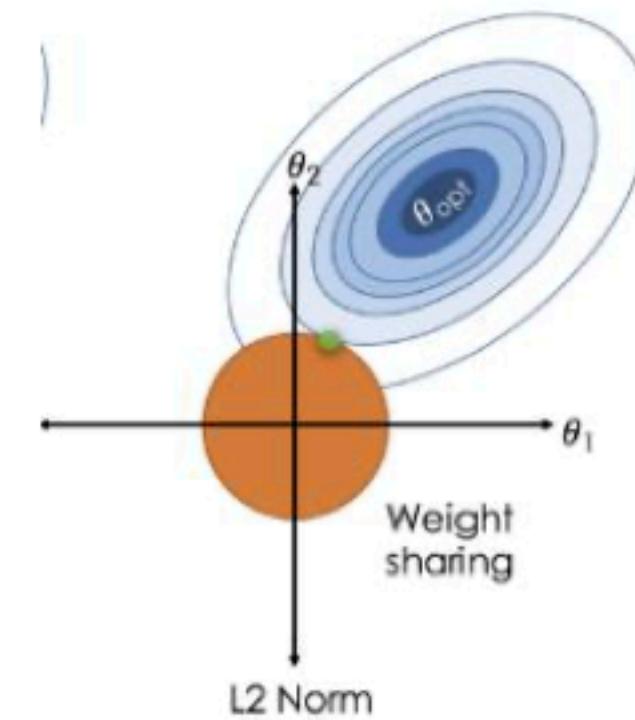
단순한 mapping function ❤
복잡한 모델 penalize (규제) ❤



릿지회귀(L2)

$$J(\theta) = \text{MSE}(\theta) + \alpha \frac{1}{2} \sum_{i=1}^n \theta_i^2$$

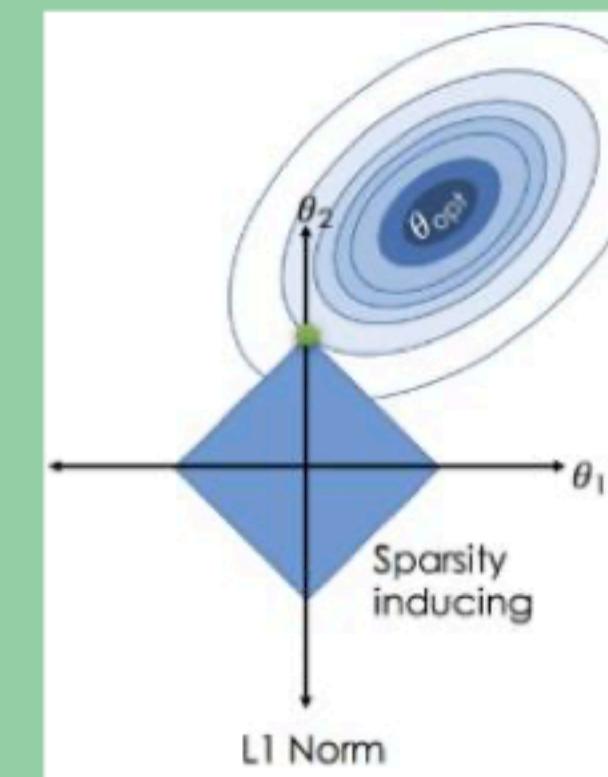
- ✓ α 에 따라 모델을 얼마나 규제할지 조절
- ✓ 편향 θ_0 은 규제되지 않음
- ✓ 입력 feature의 스케일에 민감 → 스케일링!



라쏘회귀(L1)

$$J(\theta) = \text{MSE}(\theta) + \alpha \sum_{i=1}^n |\theta_i|$$

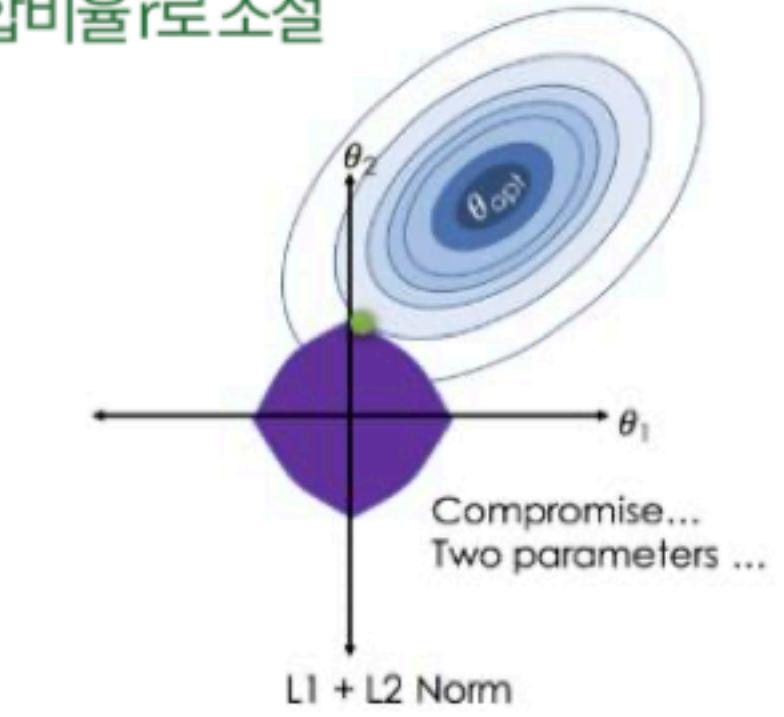
- ✓ Feature selection, 희소모델 만들



엘라스틱넷회귀

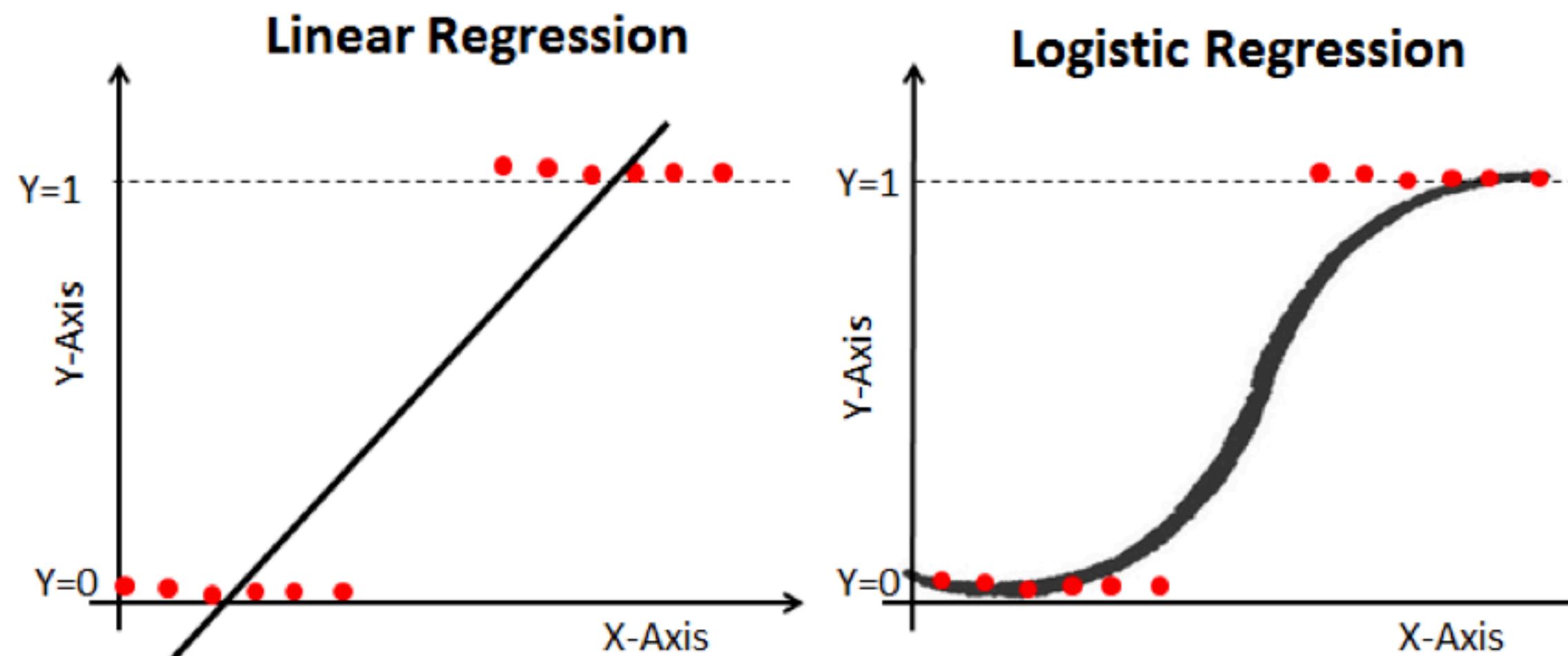
$$J(\theta) = \text{MSE}(\theta) + r\alpha \sum_{i=1}^n |\theta_i| + \frac{1-r}{2}\alpha \sum_{i=1}^n \theta_i^2$$

- ✓ 릴지회귀와 라쏘회귀를 절충한 모델
- ✓ 혼합정도는 혼합비율 r 로 조절



로지스틱 회귀

로지스틱 회귀의 개념, 모형식, 분류



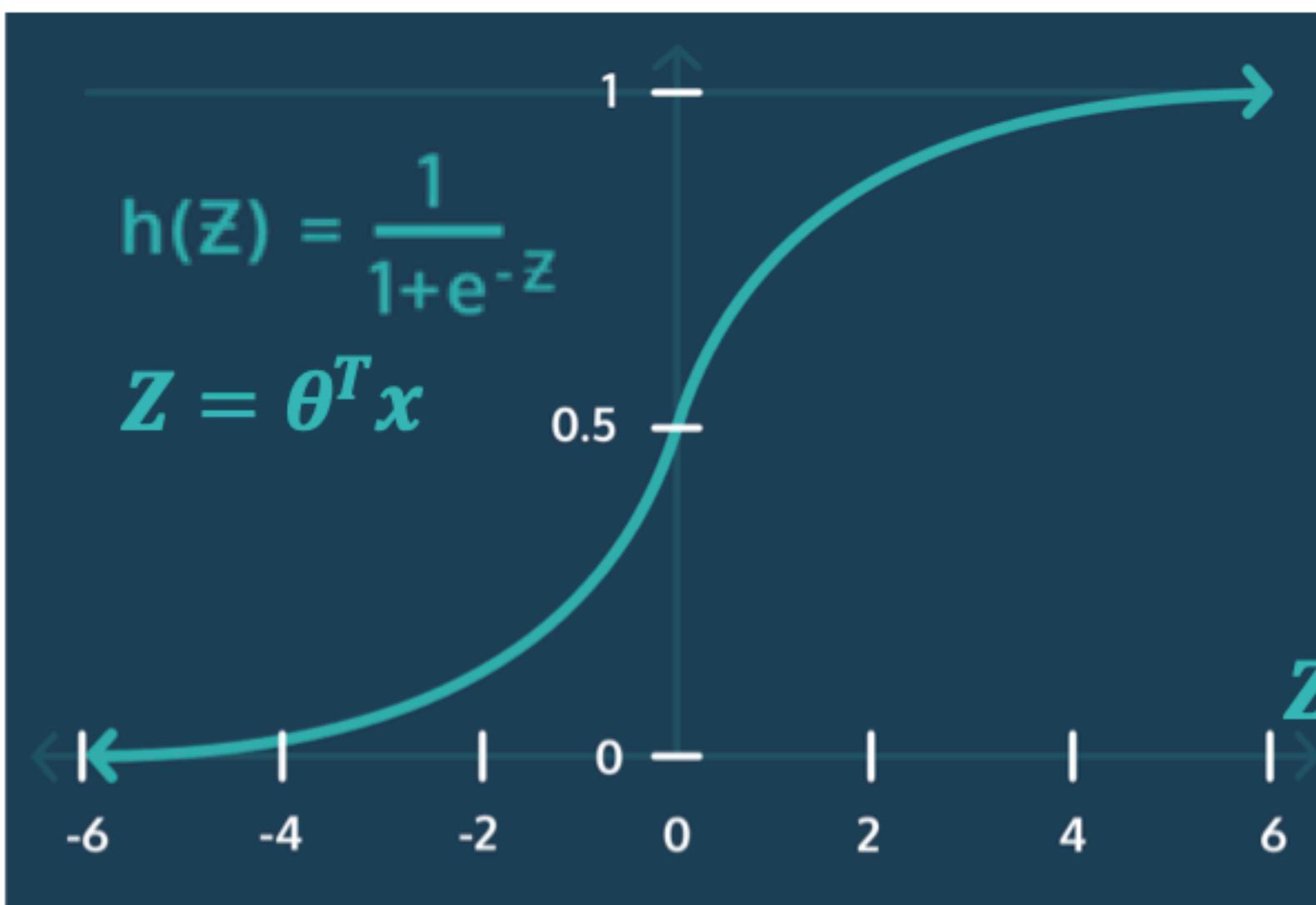
회귀 알고리즘으로 분류? X

선형 회귀로는 분류할 수 없음! 이상치에 영향 많이 받음

Solution? 💡

- 1) Mapping function $h_{\theta}(x) = \theta^T x$ 을 0~1 범위로 정규화 (normalize)
- 2) threshold를 0.5로 설정해 분류

→ 시그모이드 함수

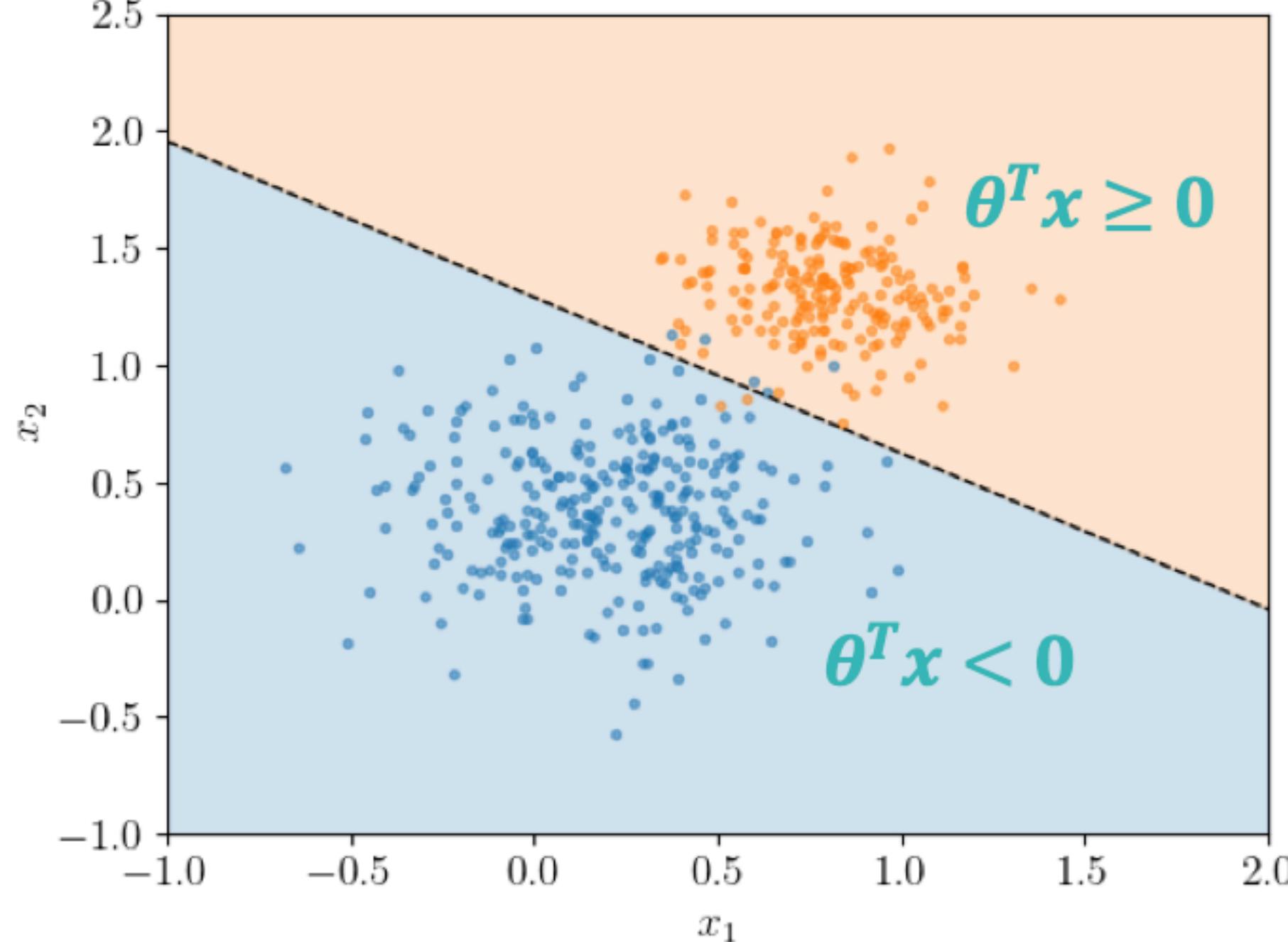


로지스틱 회귀 분류

- ✓ $\theta^T x \geq 0 \rightarrow h_{\theta}(x) = \theta^T x \geq 0.5 \rightarrow \hat{y} = 1$ 양성
- ✓ $\theta^T x < 0 \rightarrow h_{\theta}(x) = \theta^T x < 0.5 \rightarrow \hat{y} = 0$ 음성

로지스틱 회귀

이진분류에 사용되는 회귀 알고리즘



실제 양성 → 양성 예측: 비용 ↓
실제 양성 → 음성 예측: 비용 ↑

실제 음성 → 음성 예측: 비용 ↓
실제 음성 → 양성 예측: 비용 ↑

비용함수 MSE *meansquarederror*

: 모델이 얼마나 데이터를 잘 설명하고 있는지 측정; how costly our mistakes are

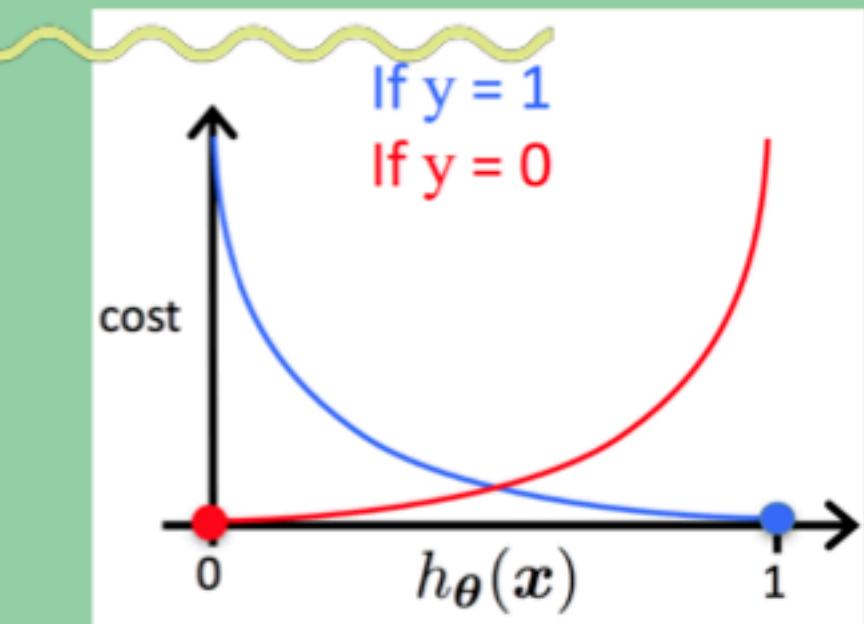
$$MSE(\mathbf{X}, h_{\theta}) = \frac{1}{m} \sum_{i=1}^m (\sigma(\theta^T \mathbf{x}^{(i)}) - y^{(i)})^2$$

비용함수

$$\begin{aligned} C(h_{\theta}, y_n) &= \begin{cases} -\log(h_{\theta}(x_n)), & \text{if } y_n = 1 \\ -\log(1 - h_{\theta}(x_n)), & \text{if } y_n = 0 \end{cases} \\ &= -y_n \log(h_{\theta}(x_n)) - (1 - y_n) \log(1 - h_{\theta}(x_n)) \end{aligned}$$

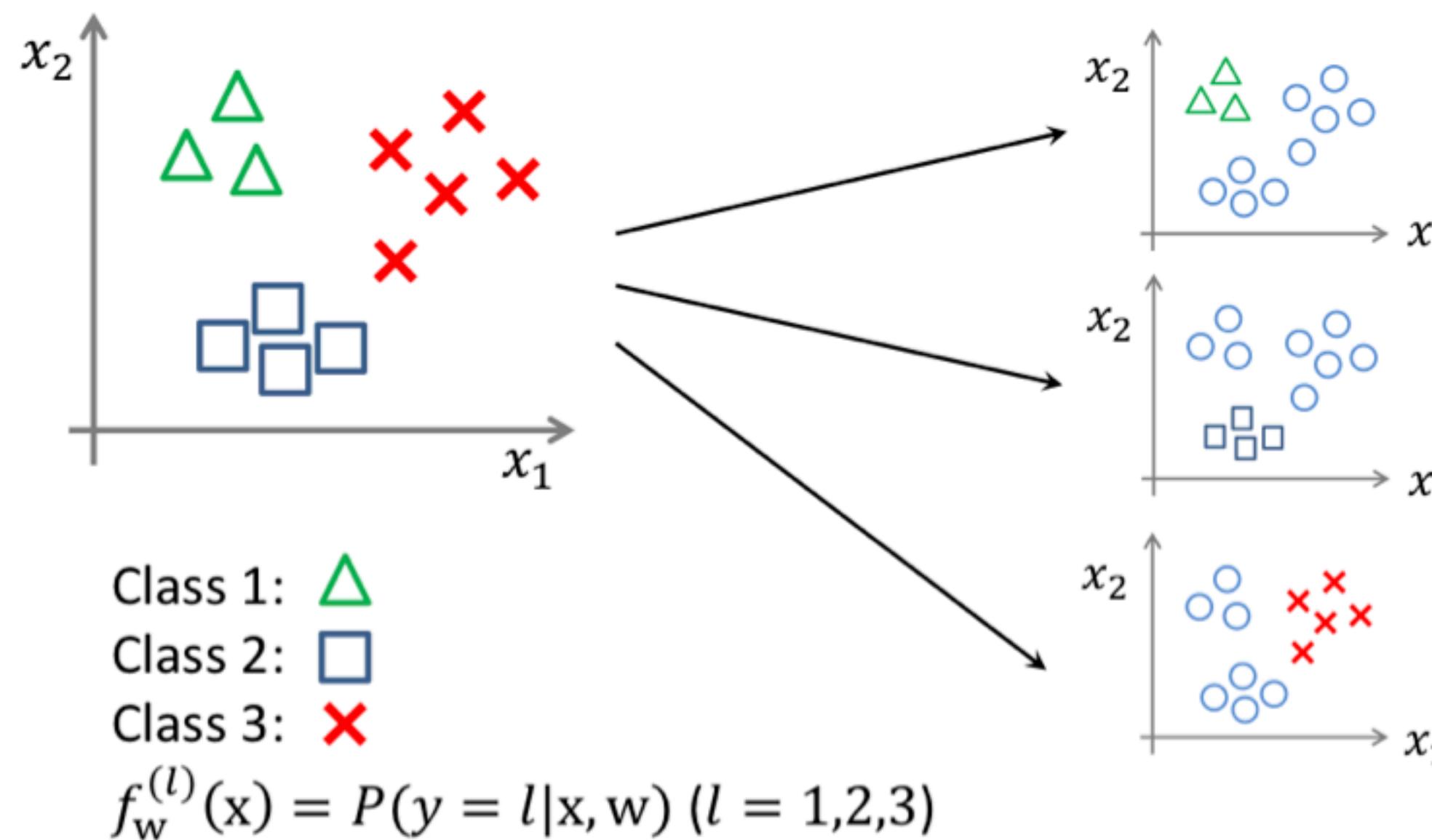
로지스틱 회귀의 비용 함수 log loss

$$L(\theta) = -\frac{1}{m} \sum_{n=1}^m [-y_n \log(h_{\theta}(x_n)) - (1 - y_n) \log(1 - h_{\theta}(x_n))]$$
$$\theta^* = \operatorname{argmin}_{\theta} L(\theta)$$



소프트맥스 회귀

소프트맥스 회귀의 개념, 모형식, 분류



다중 분류?

클래스 개수가 k 개일 때, $\sum_{l=1}^k p_l$ 의 이진분류 & 추정확률 합이 1이 되도록

Solution?

→ 소프트맥스 함수

1) 각 분류기의 회귀식에 대입 $z_l = (w^{(l)})^T x$

$$2) p_l = softmax(z_l)_l = \frac{\exp(z_l)}{\sum_{l=1}^k \exp(z_l)} = \frac{\exp((w^{(l)})^T x)}{\sum_{l=1}^k \exp((w^{(l)})^T x)}$$
$$f_w^{(l)}(x)$$

소프트맥스 회귀 분류

✓ 양성추정 확률이 가장 높은 클래스로 예측

✓ $\hat{y} = argmax_l p_l = argmax_l softmax(z_l)$

소프트맥스 회귀

다중분류에 사용되는 회귀 알고리즘

비용함수 Cross Entropy

$$C(\hat{y}_n, y_n) = - \sum_{l=1} y_l \log(\hat{p}_l)$$

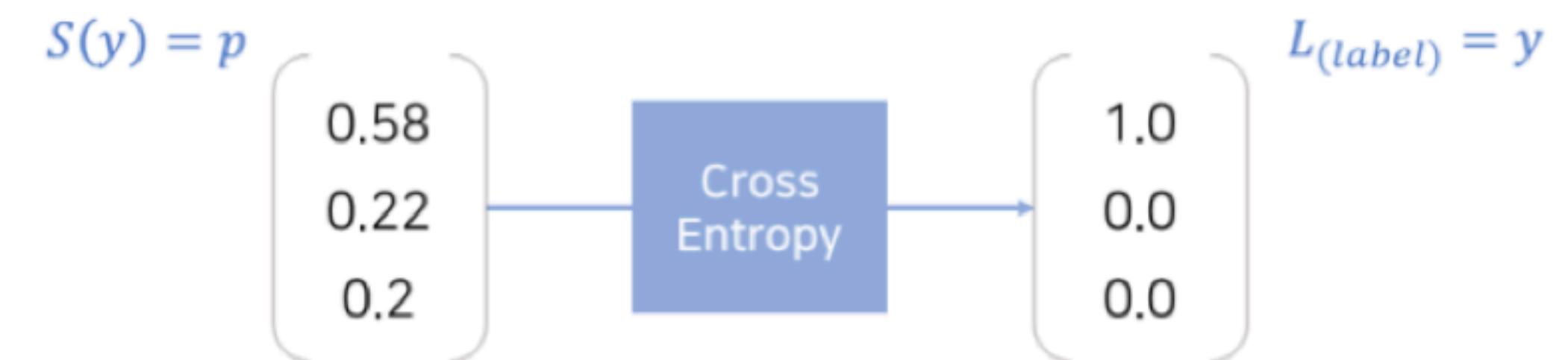
- ✓ 추정된 클래스의 확률이 타깃 클래스에 얼마나 잘 맞는지 측정
- ✓ 크로스엔트로피를 최소화
=타깃 클래스에 대해 낮은 확률을 예측하는 모델 억제

소프트맥스 회귀의 비용 함수 cross entropy

$$L(\theta) = -\frac{1}{m} \sum_k \sum_{l=1}^m y_l \log(\hat{p}_l)$$
$$\theta^* = \operatorname{argmin}_{\theta} L(\theta)$$

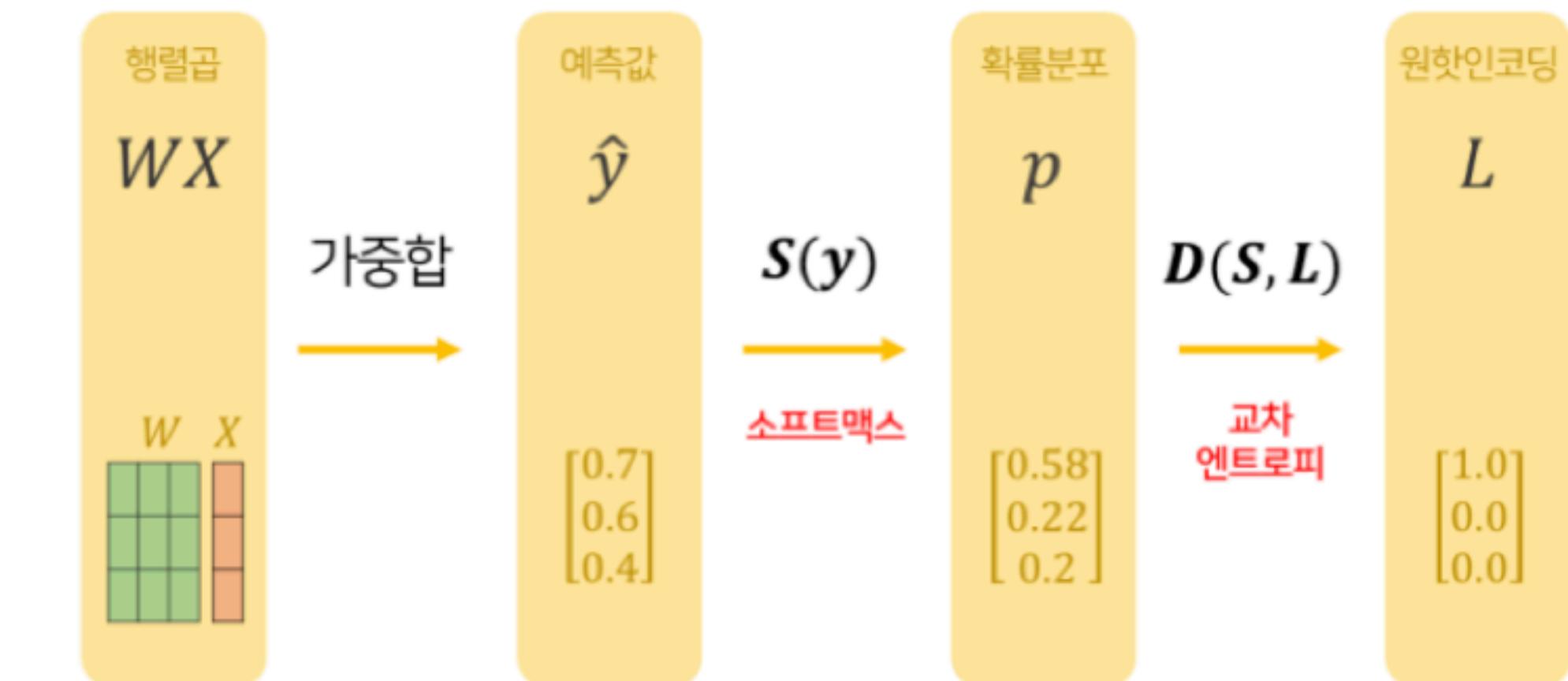
비용함수로 원핫인코딩 한다

이때 사용하는 함수가 교차 엔트로피 함수!



$$D(S, L) = - \sum_i L_i \log(S_i)$$

Softmax Regression 한 장 정리



감사합니다

질문이 있다면 말씀해주세요.