
Part. 04

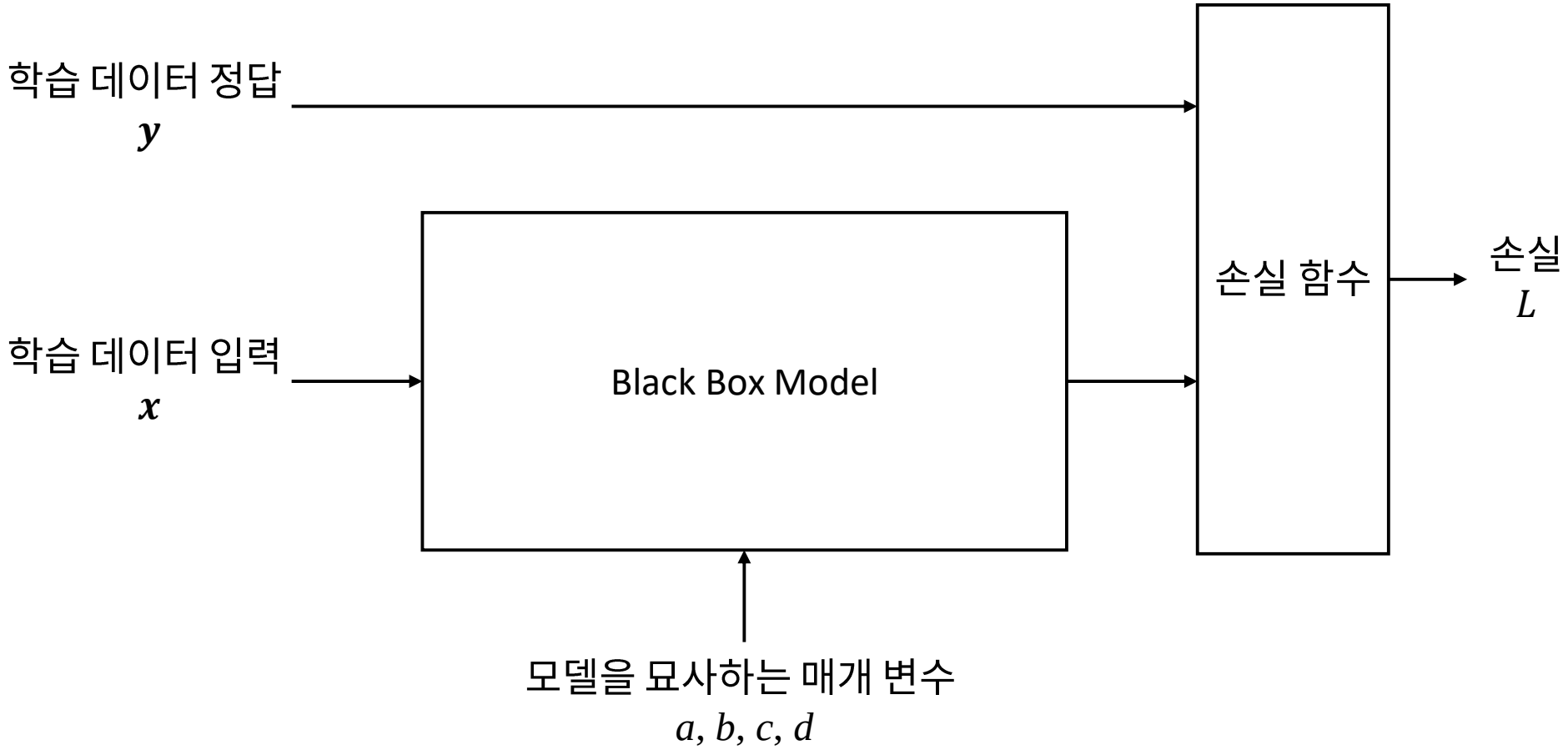
## Back Propagation

# 역전파 알고리즘의 필요성

FASTCAMPUS
ONLINE

강사. 신제용

# 블랙박스 모델

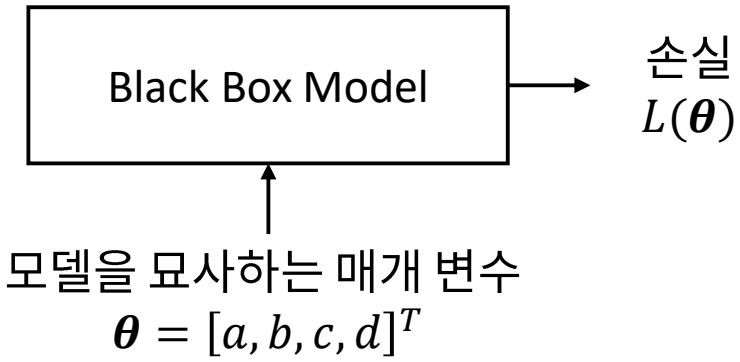



매개 변수에 의해 동작이 달라지는 Black Box Model

# 블랙박스 모델의 학습



학습 목표 : 손실을 최소화하는 매개 변수를 찾는다.




Gradient Descent

$$\boldsymbol{\theta}_{n+1} \leftarrow \boldsymbol{\theta}_n - \eta \nabla L(\boldsymbol{\theta}_n)$$

벡터 표현

$$a_{n+1} \leftarrow a_n - \eta \frac{\partial L(\boldsymbol{\theta}_n)}{\partial a}$$

$$b_{n+1} \leftarrow b_n - \eta \frac{\partial L(\boldsymbol{\theta}_n)}{\partial b}$$

$$c_{n+1} \leftarrow c_n - \eta \frac{\partial L(\boldsymbol{\theta}_n)}{\partial c}$$

$$d_{n+1} \leftarrow d_n - \eta \frac{\partial L(\boldsymbol{\theta}_n)}{\partial d}$$

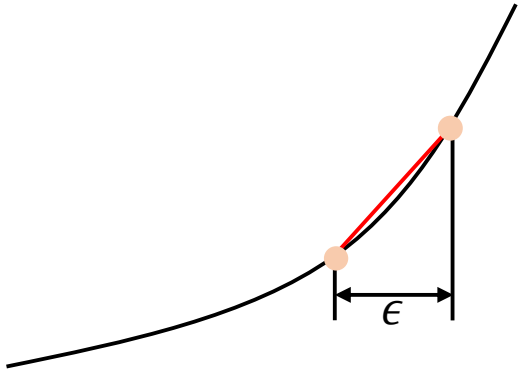
스칼라 표현

앞서 배운 Gradient Descent를 그대로 적용하면 된다.

# I 수치적 기울기



수치적 기울기 (Numerical Gradient) : 미분의 정의로부터 극한 연산을 근사해 수치적 기울기를 구할 수 있다.




$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

$$\approx \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

$\epsilon$의 값이 충분히 작다면, 수치적 기울기를 미분 값으로 사용할 수 있다.

# 블랙박스 모델의 수치적 기울기



$$\nabla L(\boldsymbol{\theta}_n) = \begin{bmatrix} \frac{\partial L(\boldsymbol{\theta}_n)}{\partial a} \\ \frac{\partial L(\boldsymbol{\theta}_n)}{\partial b} \\ \frac{\partial L(\boldsymbol{\theta}_n)}{\partial c} \\ \frac{\partial L(\boldsymbol{\theta}_n)}{\partial d} \end{bmatrix} \approx \begin{bmatrix} \frac{L(a_n+\epsilon, b_n, c_n, d_n) - L(\boldsymbol{\theta}_n)}{\epsilon} \\ \frac{L(a_n, b_n+\epsilon, c_n, d_n) - L(\boldsymbol{\theta}_n)}{\epsilon} \\ \frac{L(a_n, b_n, c_n+\epsilon, d_n) - L(\boldsymbol{\theta}_n)}{\epsilon} \\ \frac{L(a_n, b_n, c_n, d_n+\epsilon) - L(\boldsymbol{\theta}_n)}{\epsilon} \end{bmatrix}$$

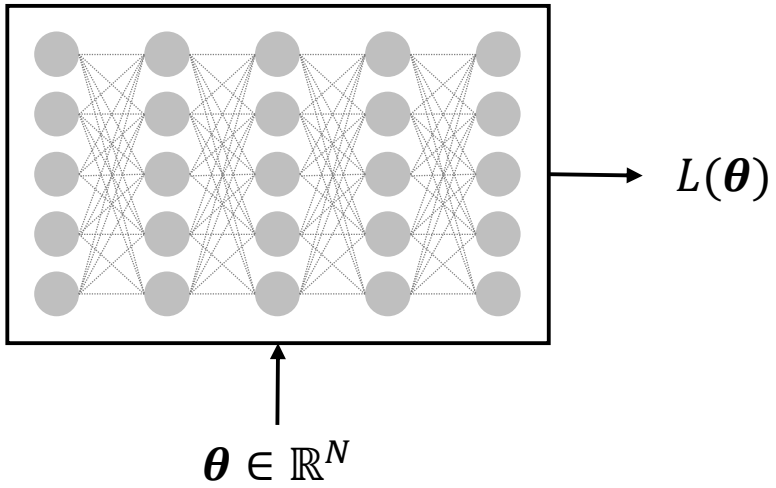
각 스칼라 변수를 각각 조금씩 바꾸어 대입해 보면서 수치적 기울기를 구한다.

$N$개의 매개 변수로 미분하기 위해 $(N+1)$번 더 손실함수를 평가해야 한다.

# I 심층 신경망의 수치적 기울기



손실 함수 평가에 필요한 곱연산 $\approx N$




$N$ : 심층 신경망의 파라미터 개수 (>> 10k)

미분을 위해 필요한 손실 평가 횟수 $N+1$

$$\nabla L(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_1} \\ \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_2} \\ \vdots \\ \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_N} \end{bmatrix} \approx \begin{bmatrix} \frac{L(\theta_1+\epsilon, \theta_2, \cdots, \theta_N) - L(\boldsymbol{\theta})}{\epsilon} \\ \frac{L(\theta_1, \theta_2+\epsilon, \cdots, \theta_N) - L(\boldsymbol{\theta})}{\epsilon} \\ \vdots \\ \frac{L(\theta_1, \theta_2, \cdots, \theta_N+\epsilon) - L(\boldsymbol{\theta})}{\epsilon} \end{bmatrix}$$

Gradient Descent 한 스텝 계산을 위한 연산은 $N(N+1)$번의 곱하기 연산이다.

10만개의 파라미터를 가진 경우 무려 100억 회 → 무언가 대책이 필요하다.