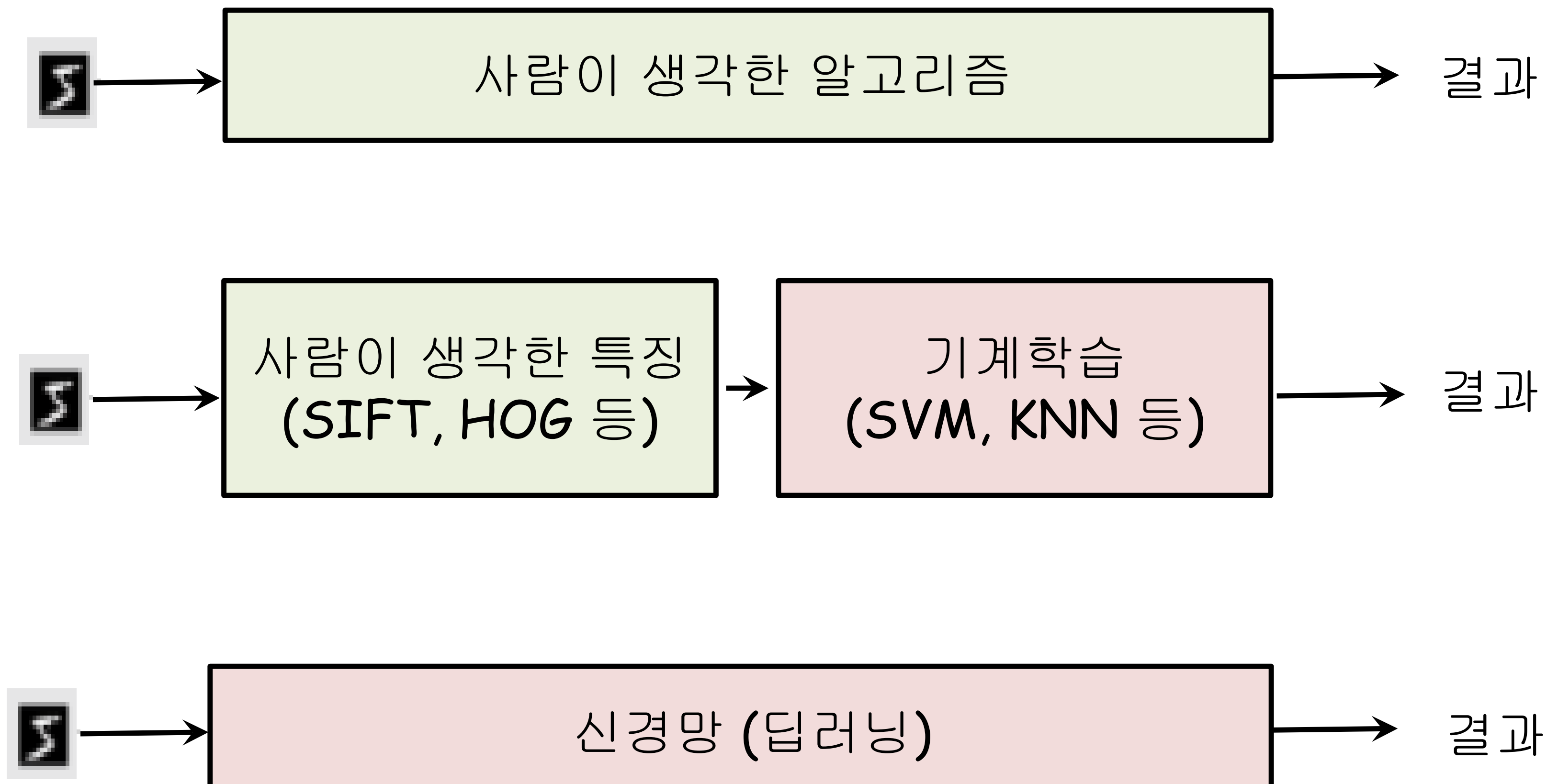
Neural Network

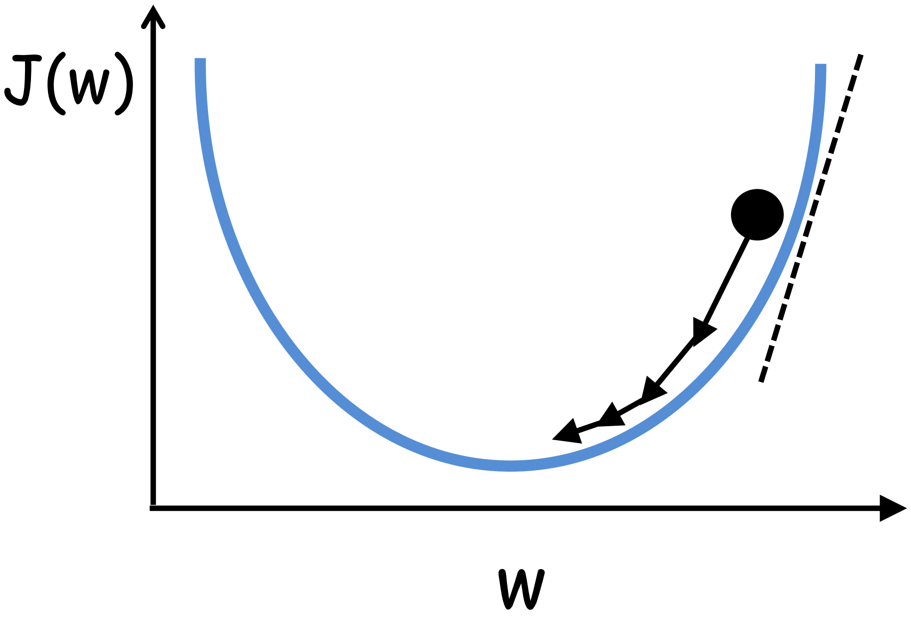
4. 신경망 학습

Rule based vs Data driven


사람이 생각한 알고리즘
결과
사람이 생각한 특징
(SIFT, HOG 등)
기계학습
(SVM, KNN 등)
결과
신경망 (딥러닝)
결과

Loss function

- 인공신경망 학습 = Loss function 을 낮추는 과정



One-hot encoding

- Classification 을 위해 Label 의 데이터 변환

$0 = [1,0,0,0,0,0,0,0,0,0]$

$1 = [0,1,0,0,0,0,0,0,0,0]$

$2 = [0,0,1,0,0,0,0,0,0,0]$

$3 = [0,0,0,1,0,0,0,0,0,0]$

$4 = [0,0,0,0,1,0,0,0,0,0]$

$5 = [0,0,0,0,0,1,0,0,0,0]$

$6 = [0,0,0,0,0,0,1,0,0,0]$

$7 = [0,0,0,0,0,0,0,1,0,0]$

$8 = [0,0,0,0,0,0,0,0,1,0]$

$9 = [0,0,0,0,0,0,0,0,0,1]$

Mean squared error

- 신경망 결과 Probability vector 와 Label vector 의 distance

$$E = \frac{1}{2}\sum_{k}\left(y_k - t_k\right)^2$$

y_k : 신경망이 k라고 예측한 확률

t_k : 라벨을 원 핫 인코딩한 후 k번째 좌표

데이터 셋의 평균 제곱 오차=각 데이터의 평균제곱오차의 평균

Cross entropy error

- Information theory 에서 확률분포사이의 거리

$$E = -\sum_k t_k \log y_k$$

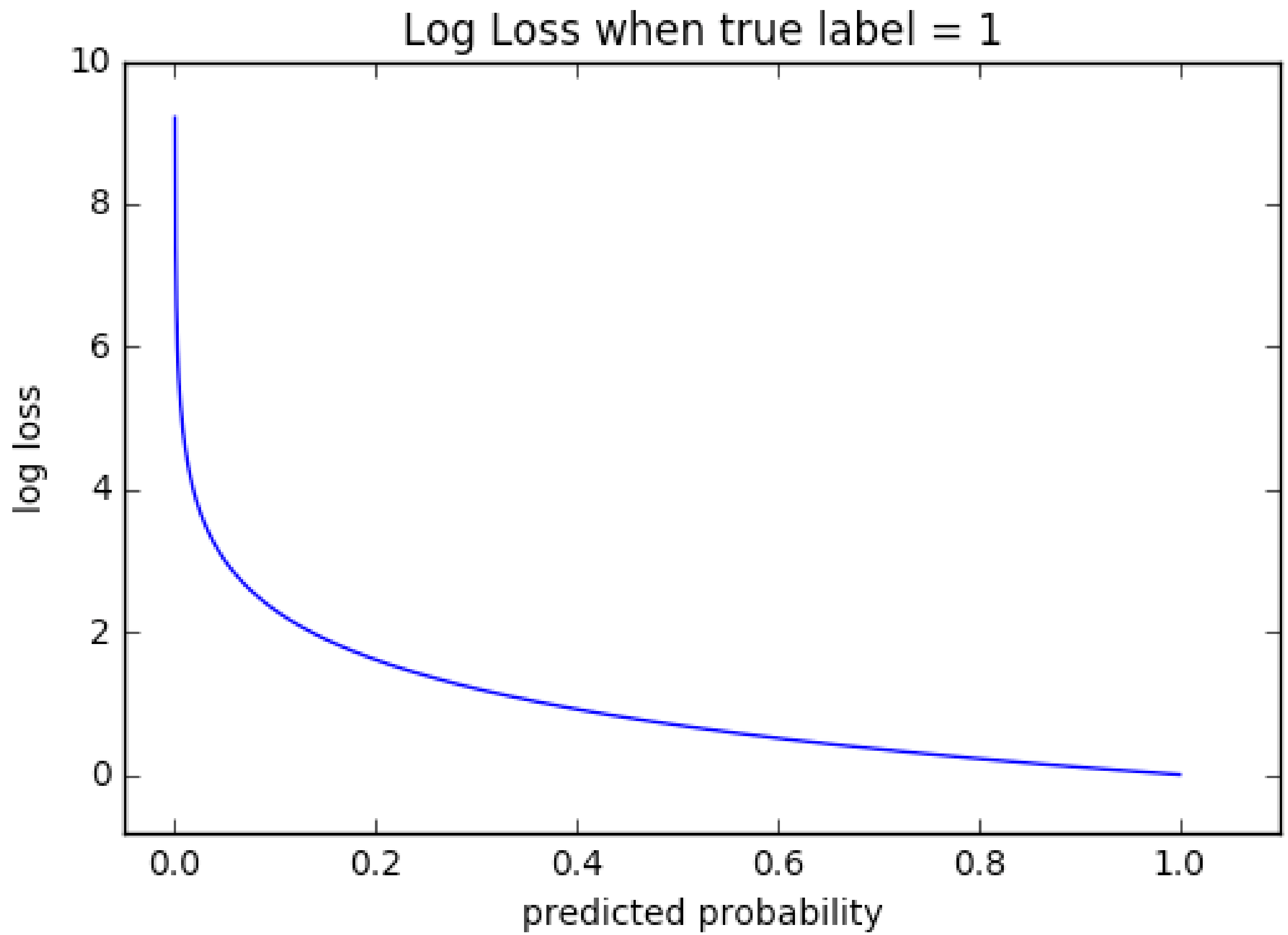
y_k : 신경망이 k라고 예측한 확률

t_k : 라벨을 원 핫 인코딩한 후 k번째 좌표

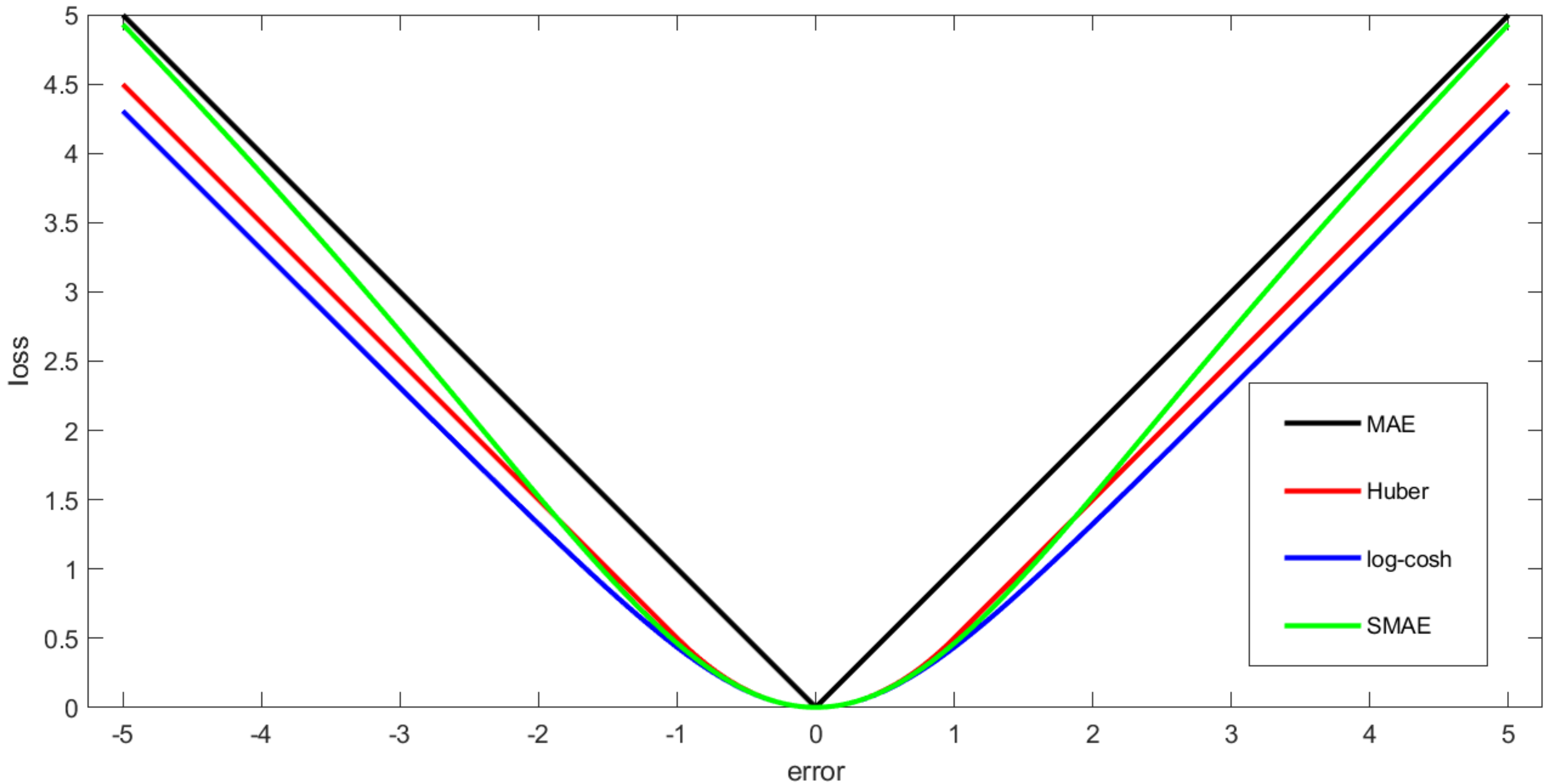
라벨이 k_0라고 하면 $E = -\log y_{k_o}$

데이터 셋 교차 엔트로피 오차=각 데이터 교차 엔트로피 오차 평균

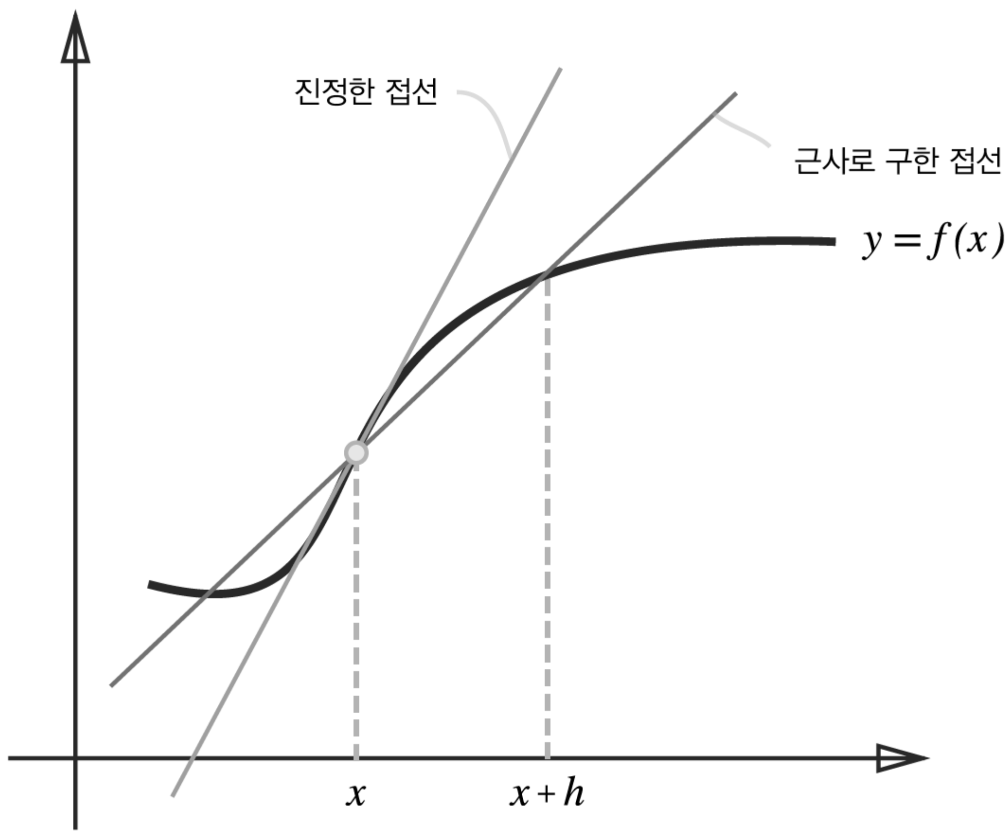
Cross entropy error



Loss functions



Numerical Differentiation



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

Numerical Differentiation

$$\frac{dy}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

$$\frac{f(x+h) - f(x-h)}{2h}$$

Univariable vs Multivariable

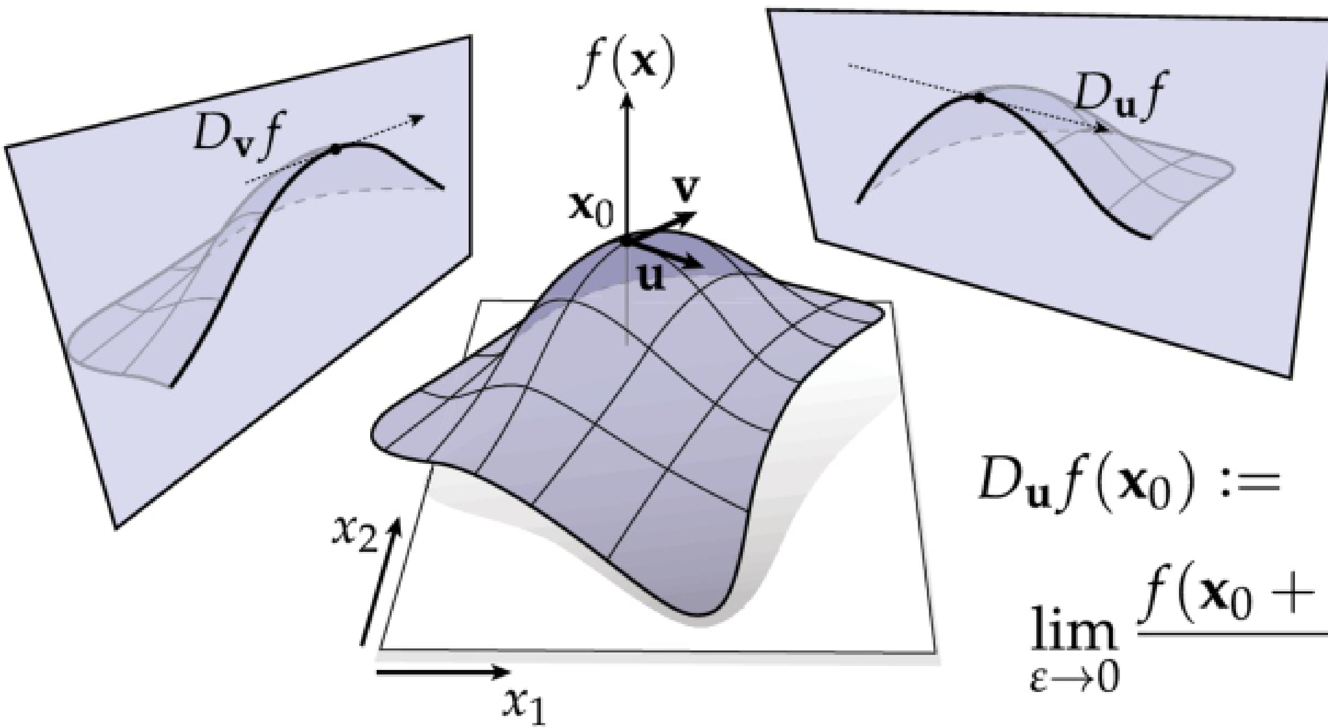
- 1변수 함수 미분

$$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

- 다변수 함수의 방향 미분

$$D_{\mathbf{v}}f(\mathbf{x})=\lim_{h\to 0}\frac{f(\mathbf{x}+h\mathbf{v})-f(\mathbf{x})}{h}$$

Directional derivative



take a small step along u

$$D_{\mathbf{u}} f(\mathbf{x}_0) := \lim_{\varepsilon \to 0} \frac{f(\mathbf{x}_0 + \varepsilon \mathbf{u}) - f(\mathbf{x}_0)}{\varepsilon}$$

Partial derivative

- 각 축 방향으로 미분

$$D_{(1,0,\cdots,0)}f(\mathbf{x}) = \lim_{h\to 0}\frac{f(\mathbf{x}+h(1,0,\cdots,0)) - f(\mathbf{x})}{h}$$

$$\vdots$$

$$D_{(0,0,\cdots,1)}f(\mathbf{x}) = \lim_{h\to 0}\frac{f(\mathbf{x}+h(0,0,\cdots,1)) - f(\mathbf{x})}{h}$$

$$\frac{\partial f}{\partial x_i}$$

Gradient (vector)

$$\nabla f = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n})$$

$$D_{\mathbf{v}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{v}$$

$$z = \nabla f(x_0, y_0) \cdot \big((x, y) - (x_0, y_0)\big) + f(x_0, y_0)$$

- 함수 f와 점 $\mathbf{x}$ 에 대해

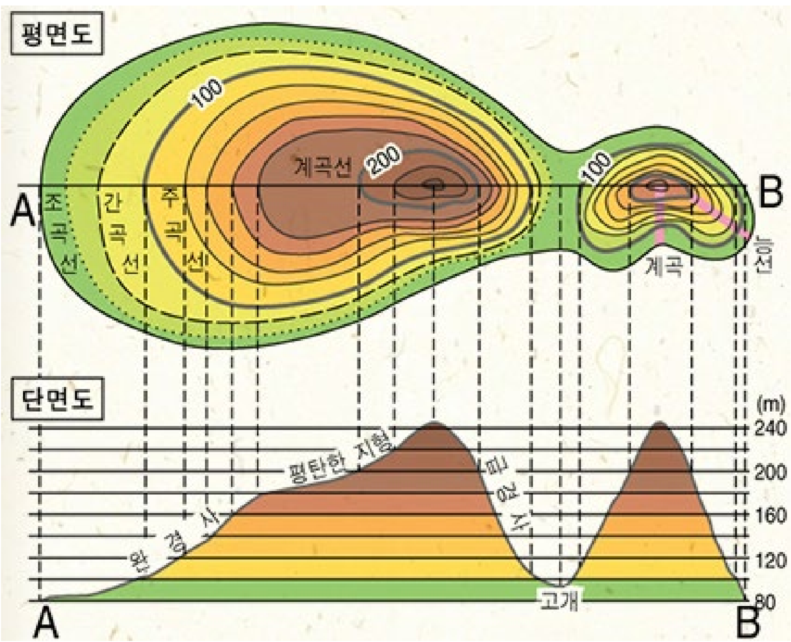
$$D_{\mathbf{v}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{v} = \left\| \nabla f(\mathbf{x}) \right\| \cdot \left\| \mathbf{v} \right\| \cos\theta = \left\| \nabla f(\mathbf{x}) \right\| \cos\theta$$

- 방향 미분이 가장 커지는 방향 = gradient 방향, gradient의 크기

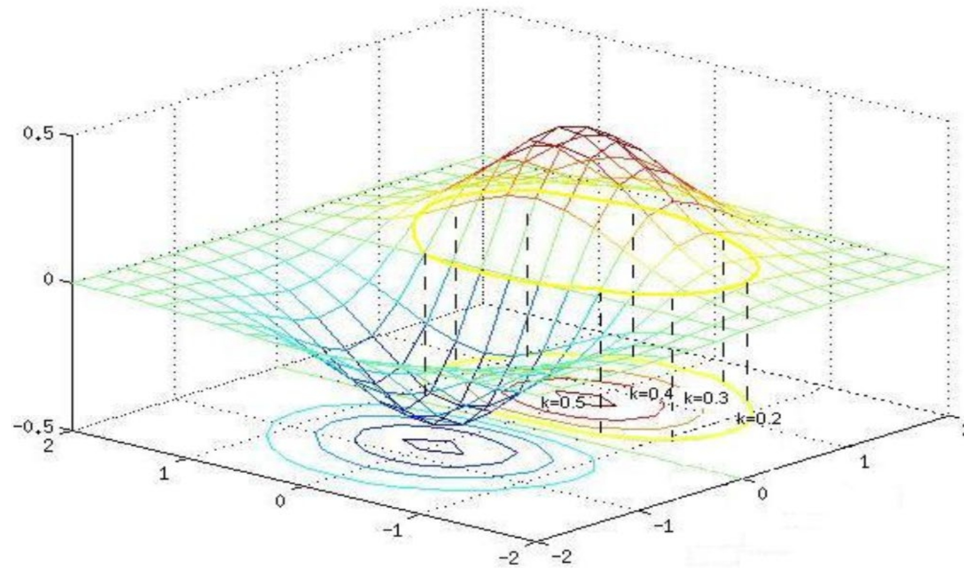
- 가장 작아지는 방향 = gradient 반대 방향, 마이너스 gradient의 크기

- 방향 미분이 0이 되는 방향 = gradient와 수직 방향

Level curve

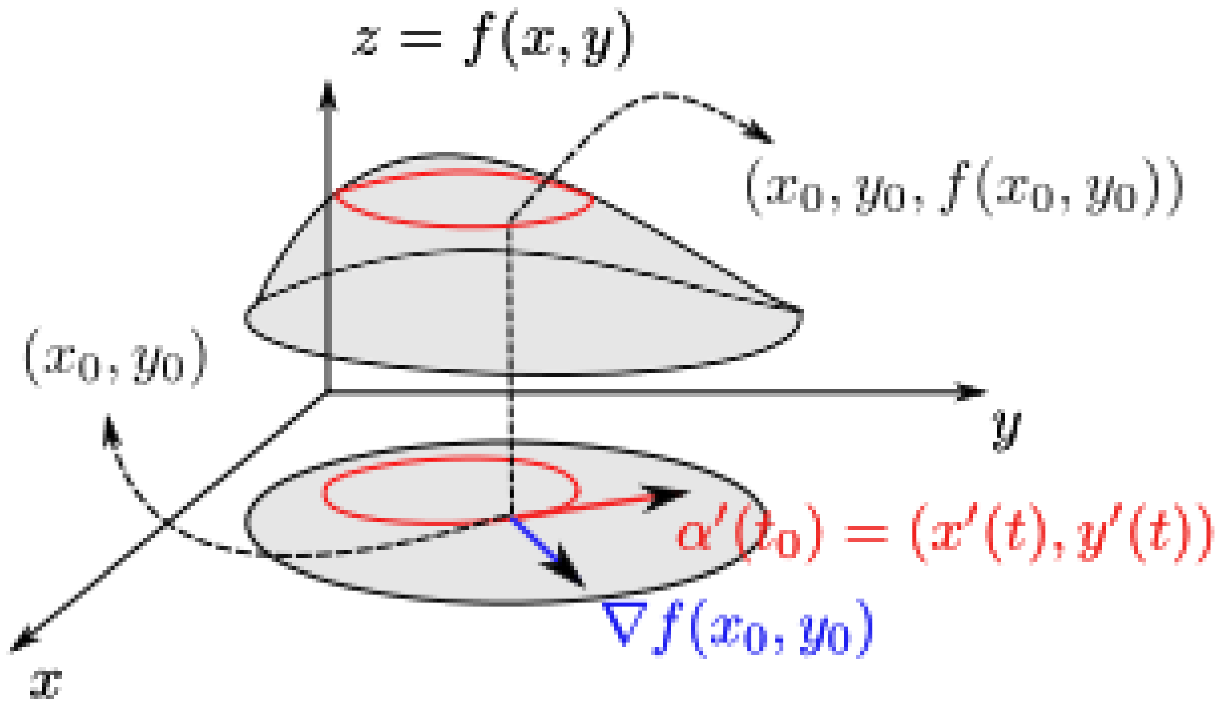


평면도
100
계곡선
200
100
A
B
조곡선
간곡선
주곡선
계곡
능선
단면도
(m)
240
200
160
120
80
평탄한 지형
완경사
급경사
고개
A
B

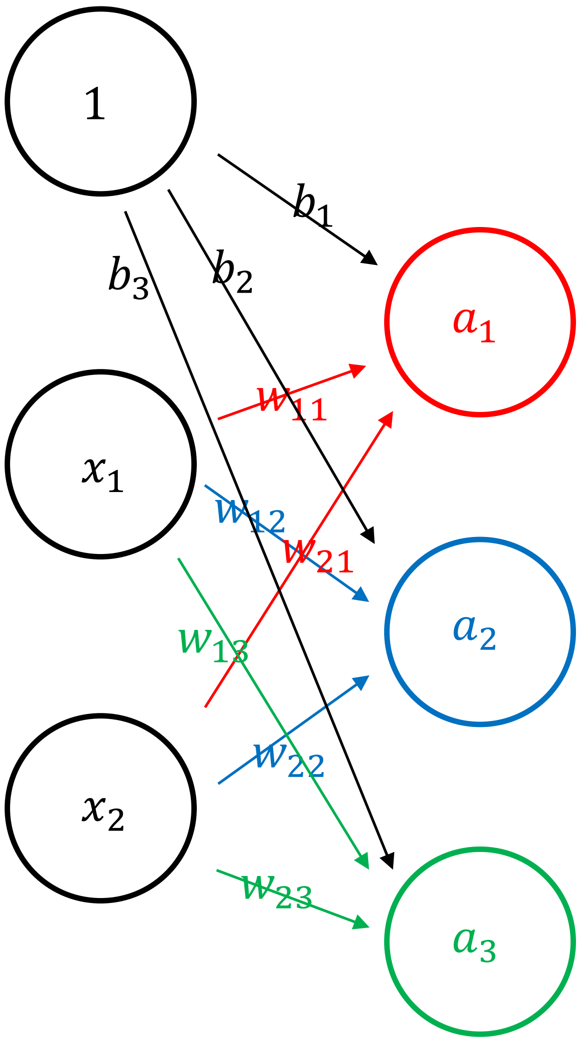


0.5
0
-0.5
k=0.5
k=0.4
k=0.3
k=0.2
2
1
0
-1
-2
-2
-1
0
1
2

- 함수 f의 등위선(면)과 gradient는 수직



Variables

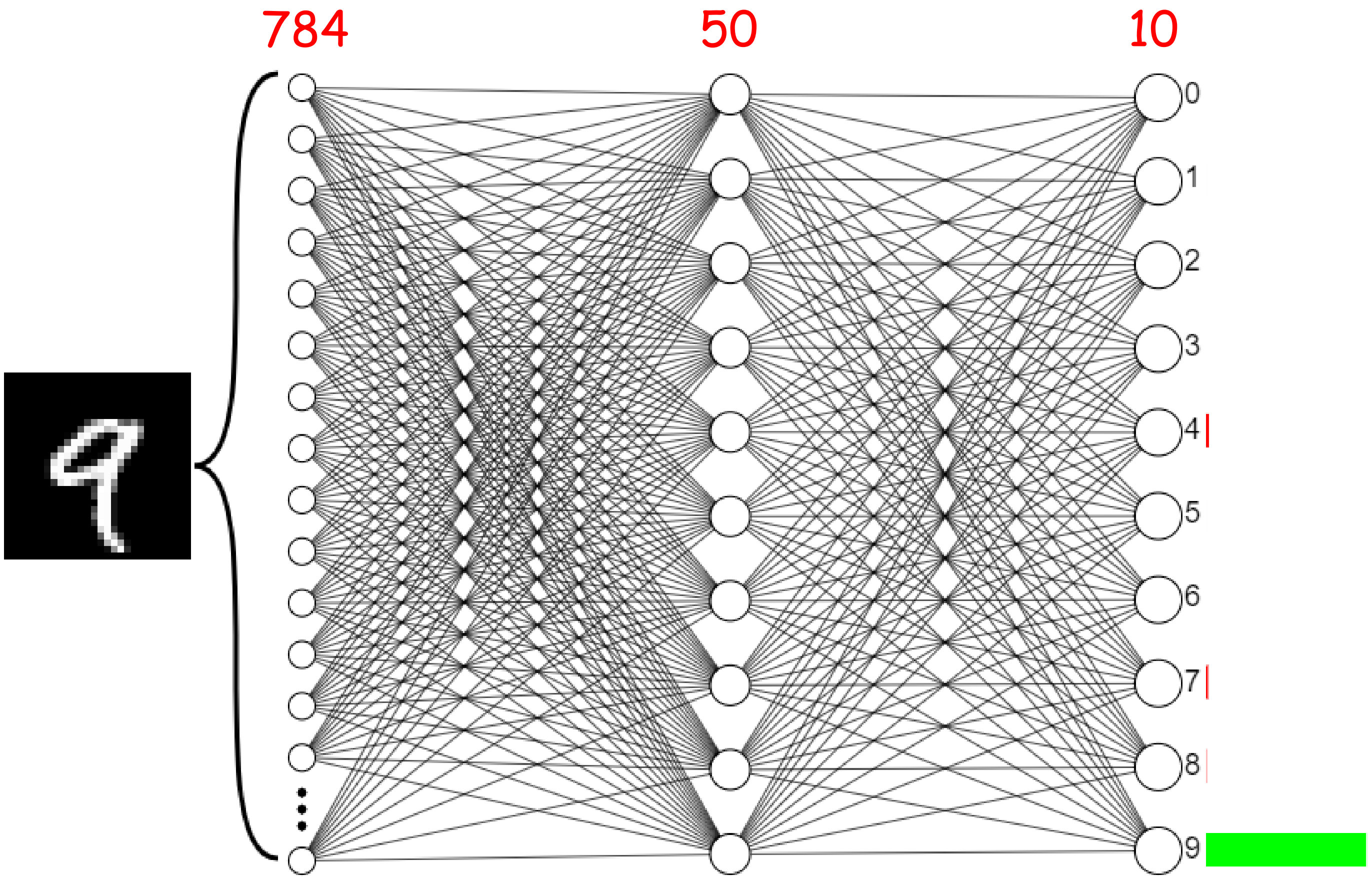


- weight, bias → variables
- input → coefficients

- Label = (0,0,1)

$$L\left(\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}, \begin{pmatrix} b_1 & b_2 & b_3 \end{pmatrix}\right)$$

$$= -\log \frac{e^{w_{13}x_1 + w_{23}x_2 + b_3}}{e^{w_{11}x_1 + w_{21}x_2 + b_1} + e^{w_{12}x_1 + w_{22}x_2 + b_2} + e^{w_{13}x_1 + w_{23}x_2 + b_3}}$$

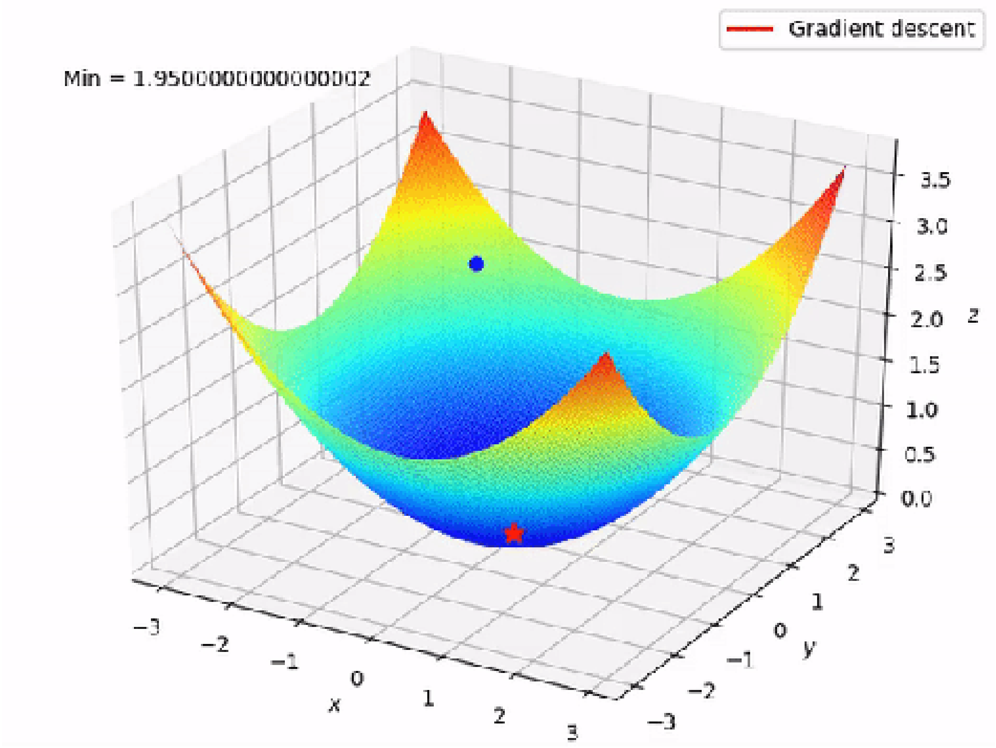


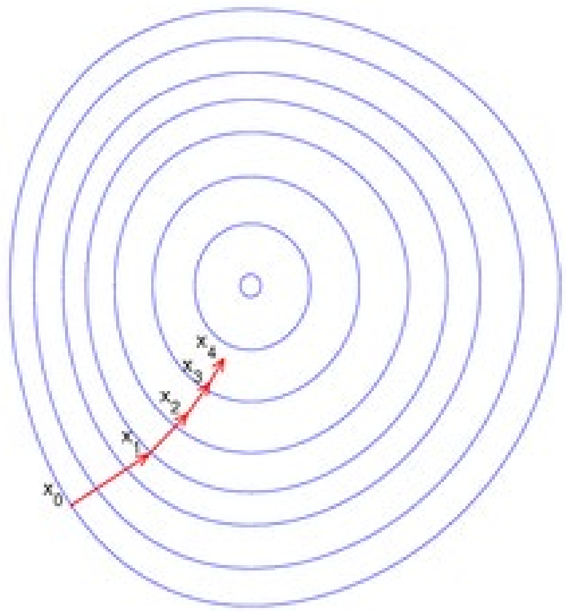
- # of variables

  $= 784 \times 50 + 50 + 50 \times 10 + 10 = 39{,}760$

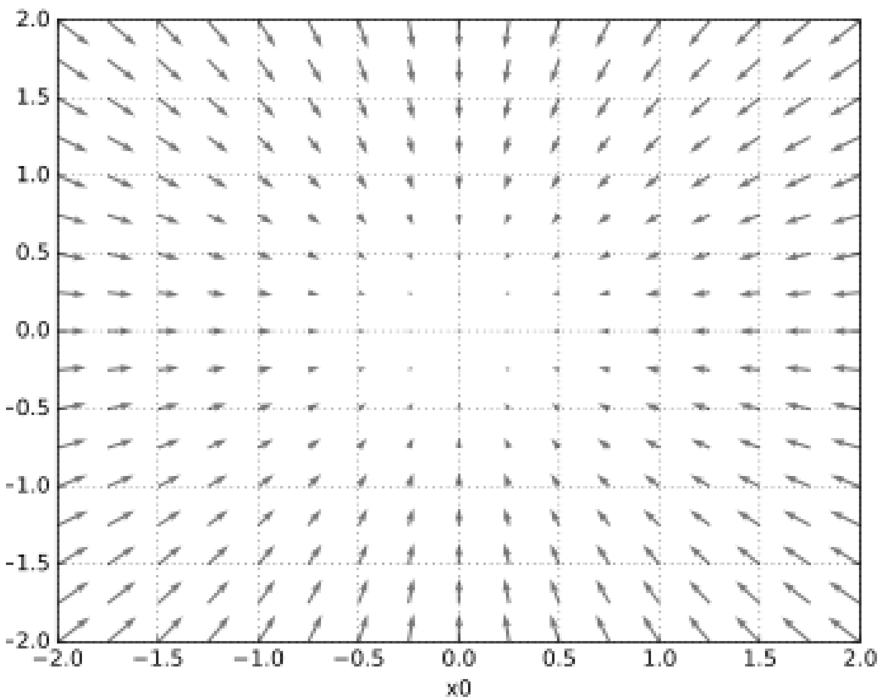
Gradient Descent Method

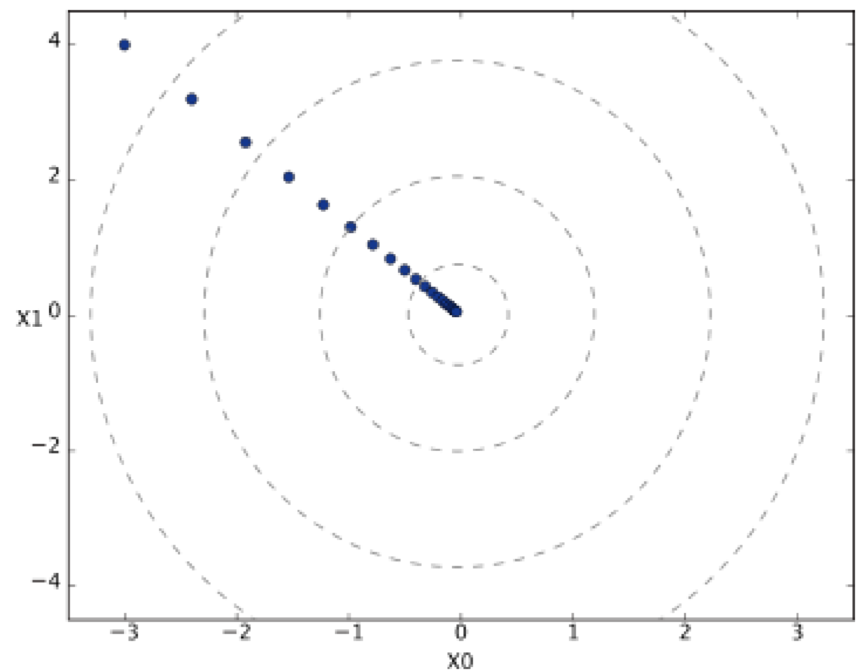
$$\mathbf{x}_{n+1} = \mathbf{x}_n - \eta \cdot \nabla f$$



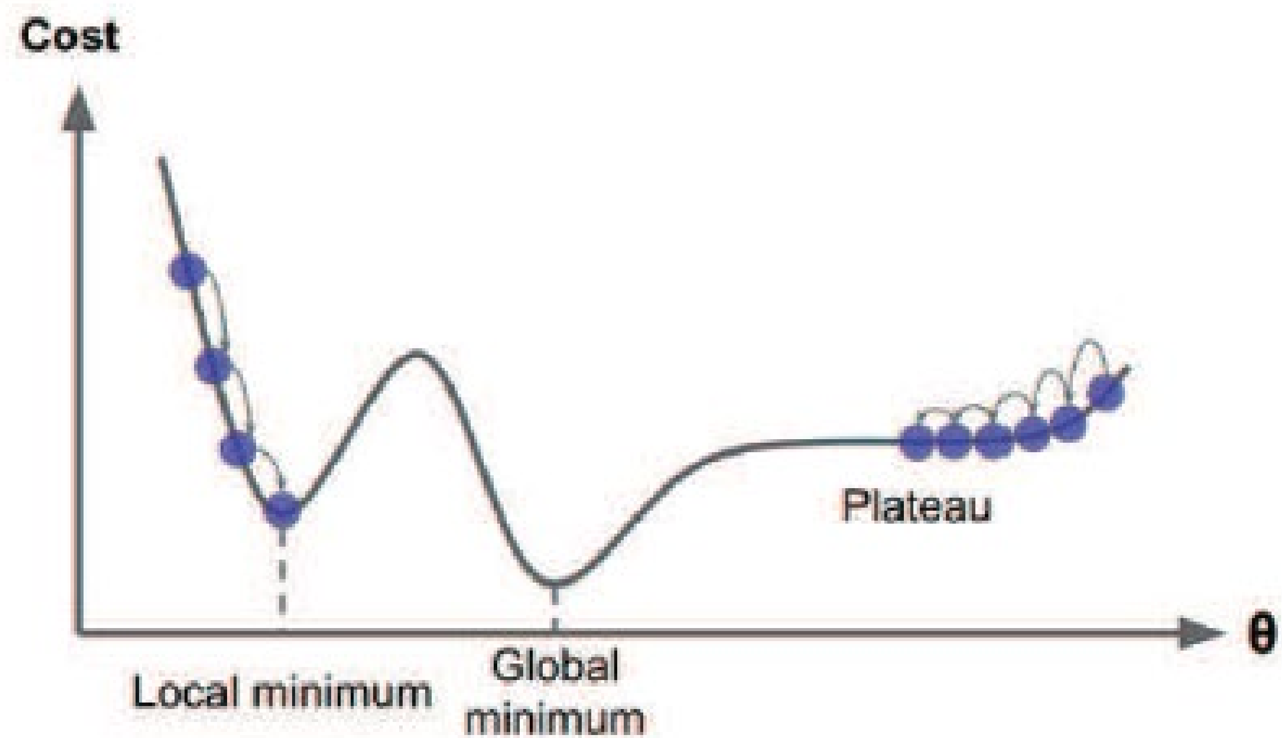


$$f(x_0, x_1) = x_0^{\,2} + x_1^{\,2}$$

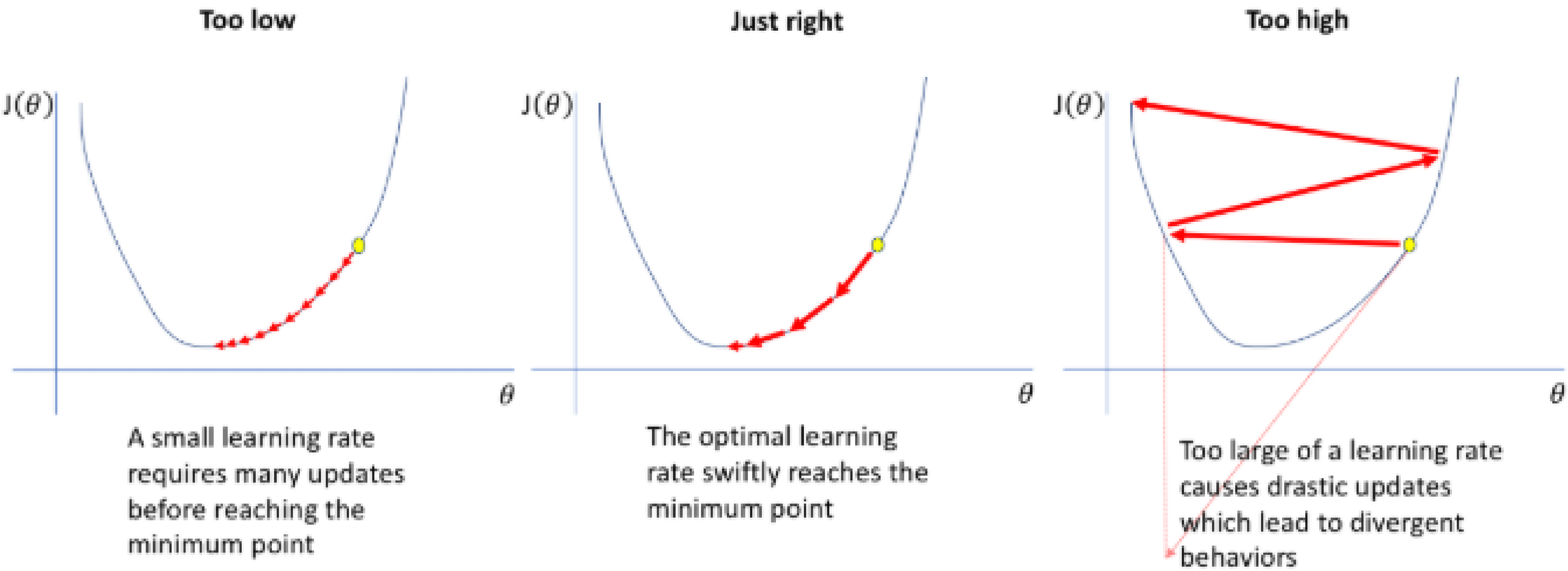




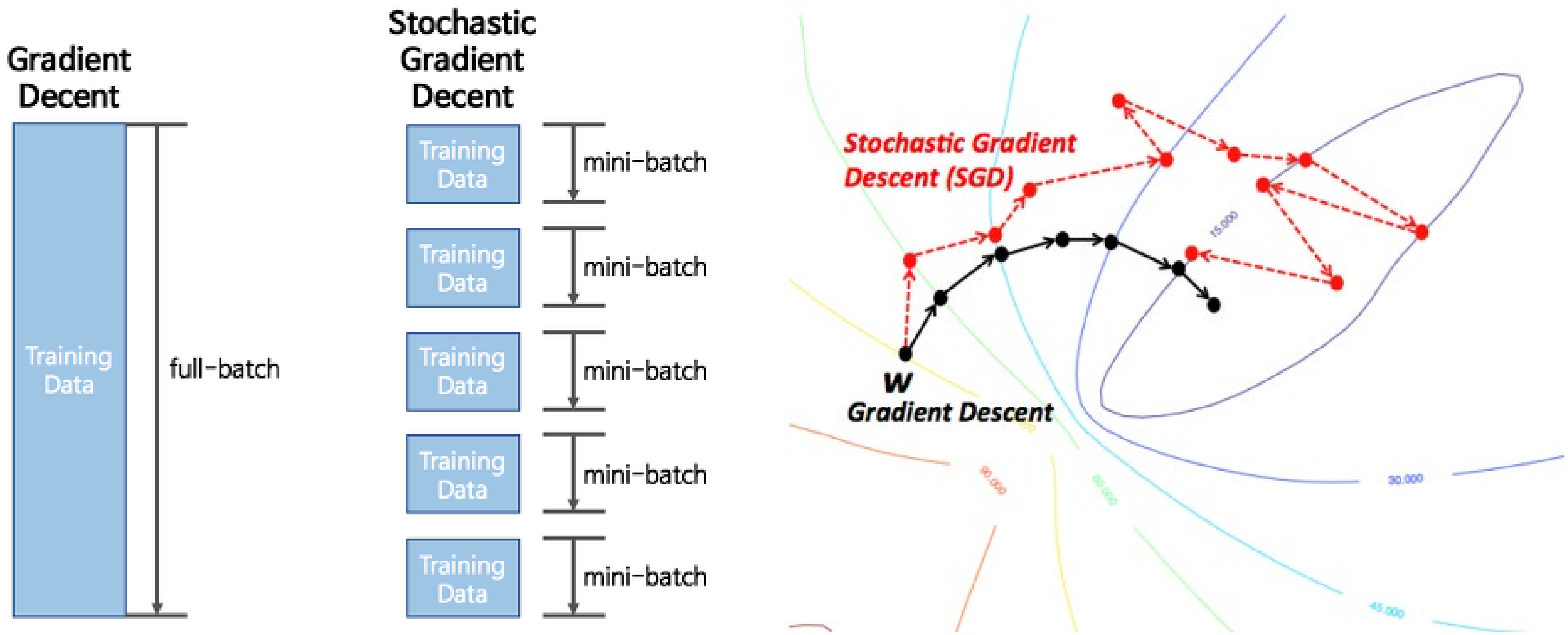
- Local minimum 또는 saddle point 도출
- 초기 위치 중요

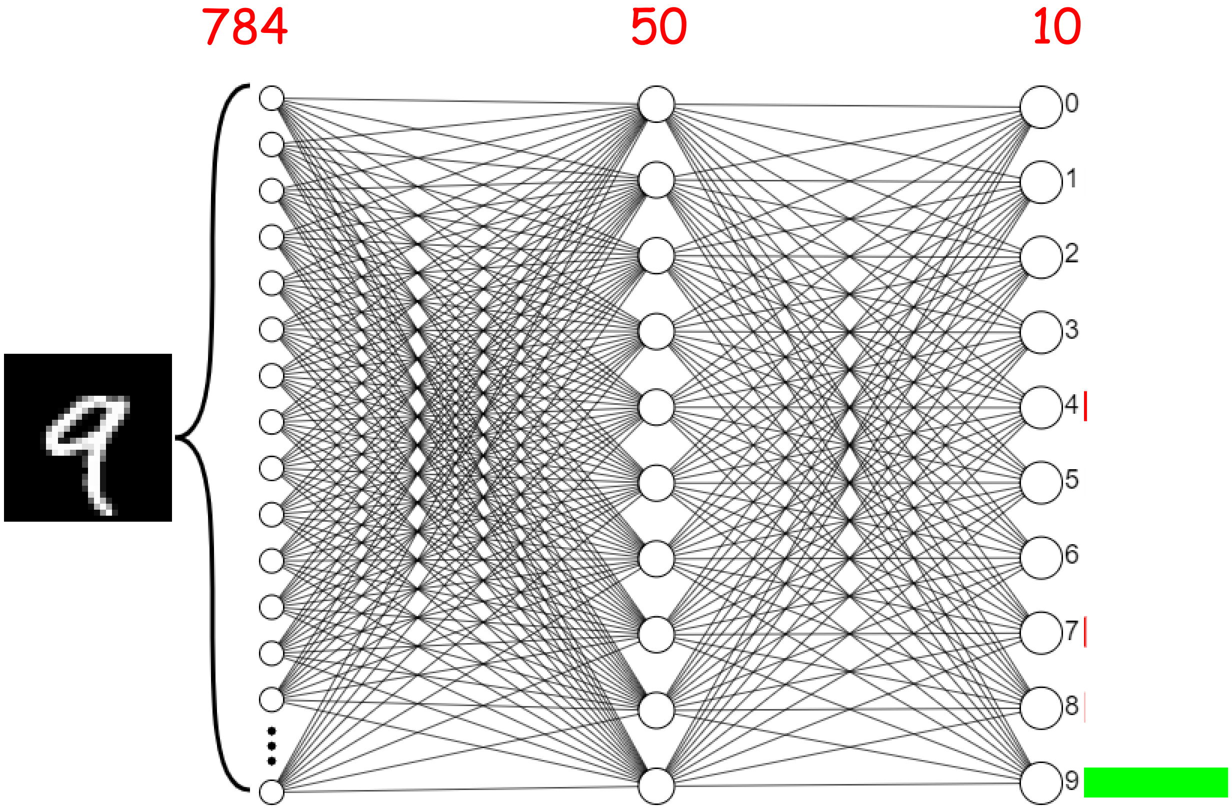


Learning rate



Stochastic Gradient Descent





- 2층 신경망, 뉴런의 수 #784, #50, #10
- 손실함수의 # of variables = 39,760
- Learning rate = 0.1, # of epochs = 10,000
- # of epochs 만큼 weight 와 bias 업데이트 $w_j := w_j - \eta \cdot \nabla f$

변수	설명
params	신경망의 매개변수를 보관하는 딕셔너리 변수(인스턴스 변수) params['W1']은 1번째 층의 가중치, params['b1']은 1번째 층의 편향 params['W2']는 2번째 층의 가중치, params['b2']는 2번째 층의 편향
grads	기울기 보관하는 딕셔너리 변수(numerical_gradient() 메서드의 반환 값) grads['W1']은 1번째 층의 가중치의 기울기, grads['b1']은 1번째 층의 편향의 기울기 grads['W2']는 2번째 층의 가중치의 기울기, grads['b2']는 2번째 층의 편향의 기울기

메서드	설명
__init__(self, input_size, hidden_size, output_size)	초기화를 수행한다. 인수는 순서대로 입력층의 뉴런 수, 은닉층의 뉴런 수, 출력층의 뉴런 수
predict(self, x)	예측(추론)을 수행한다. 인수 x는 이미지 데이터
loss(self, x, t)	손실 함수의 값을 구한다. 인수 x는 이미지 데이터, t는 정답 레이블(아래 칸의 세 메서드의 인수들도 마찬가지)
accuracy(self, x, t)	정확도를 구한다.
numerical_gradient(self, x, t)	가중치 매개변수의 기울기를 구한다.
gradient(self, x, t)	가중치 매개변수의 기울기를 구한다. numerical_gradient()의 성능 개선판! 구현은 다음 장에서...

Numerical Differentiation

- $39{,}760 \times 10{,}000$번의 수치 미분 계산이 필요
- affine함수, sigmoid함수, softmax, 교차 엔트로피를 합성한 함수
- Computation cost 과다

Backpropagation

- affine층, sigmoid층, softmax층, 손실함수층에서 analytic derivative 와 chain rule을 써서 전체 미분을 구함
- 컴퓨터가 가장 빨리 할 수 있는 덧셈과 곱으로 미분값 도출

Question?

자료 출처

Deep learning from scratch, 한빛미디어, 사이토고키

https://github.com/youbeebee/deeplearning_from_scratch