

# Mathematics for Artificial Intelligence

4강: 경사하강법 (매운맛)

---

임성빈

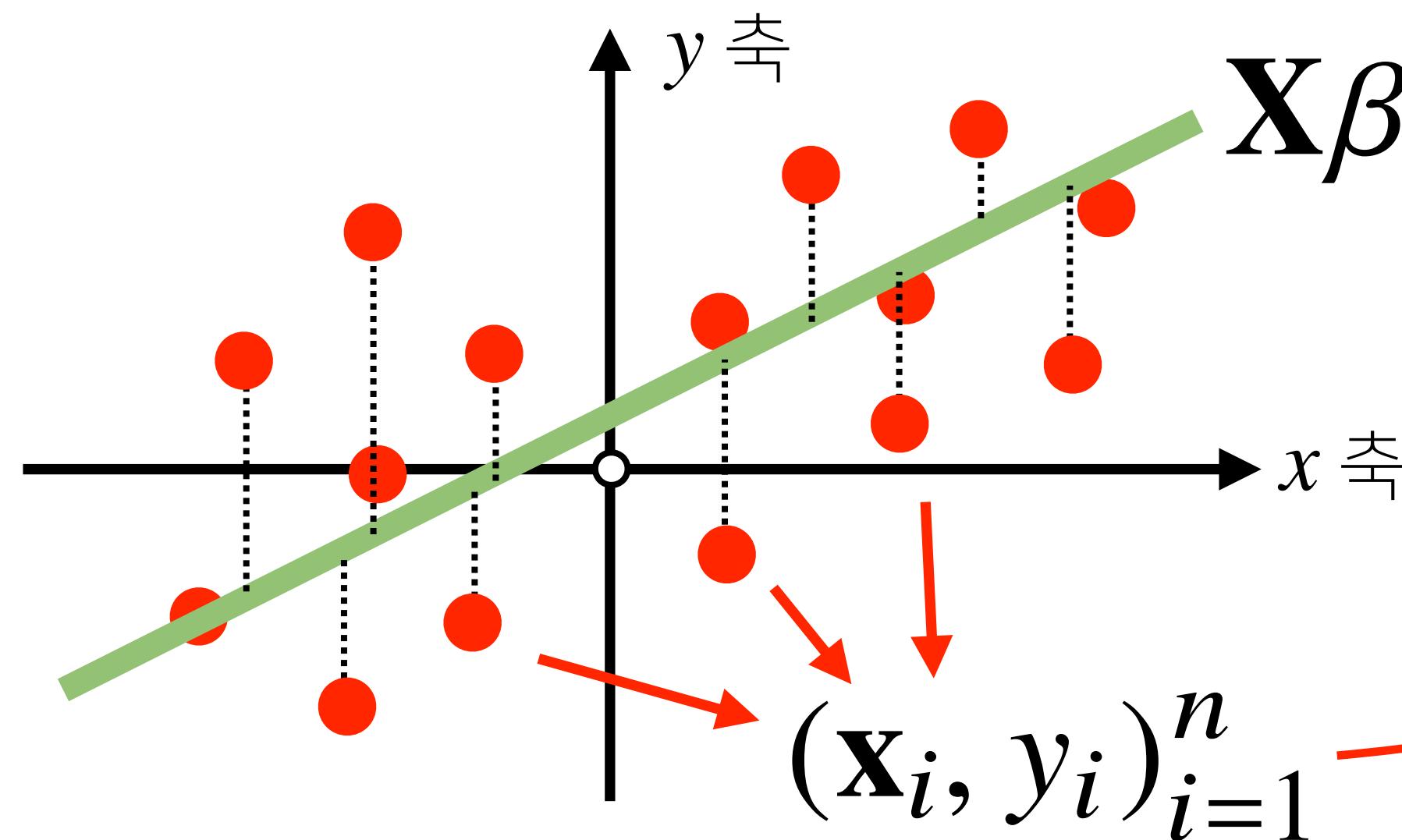


인공지능대학원 & 산업공학과  
Learning Intelligent Machine Lab



# 선형회귀분석 복습

- `np.linalg.pinv` 를 이용하면 데이터를 선형모델(linear model)로 해석하는 선형회귀식을 찾을 수 있다

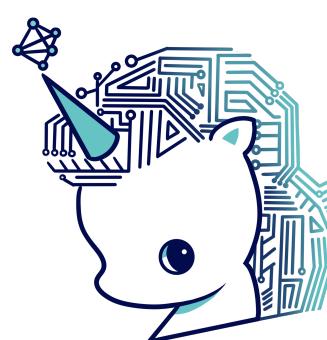


$n \geq m$  인 경우: 데이터가 변수 개수보다 많거나 같아야 함

$\times$

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} \neq \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$
$$\mathbf{X}\beta = \hat{\mathbf{y}} \approx \mathbf{y}$$
$$\min_{\beta} \|\mathbf{y} - \hat{\mathbf{y}}\|_2$$
$$\Rightarrow \beta = \mathbf{X}^+ \mathbf{y}$$
$$= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

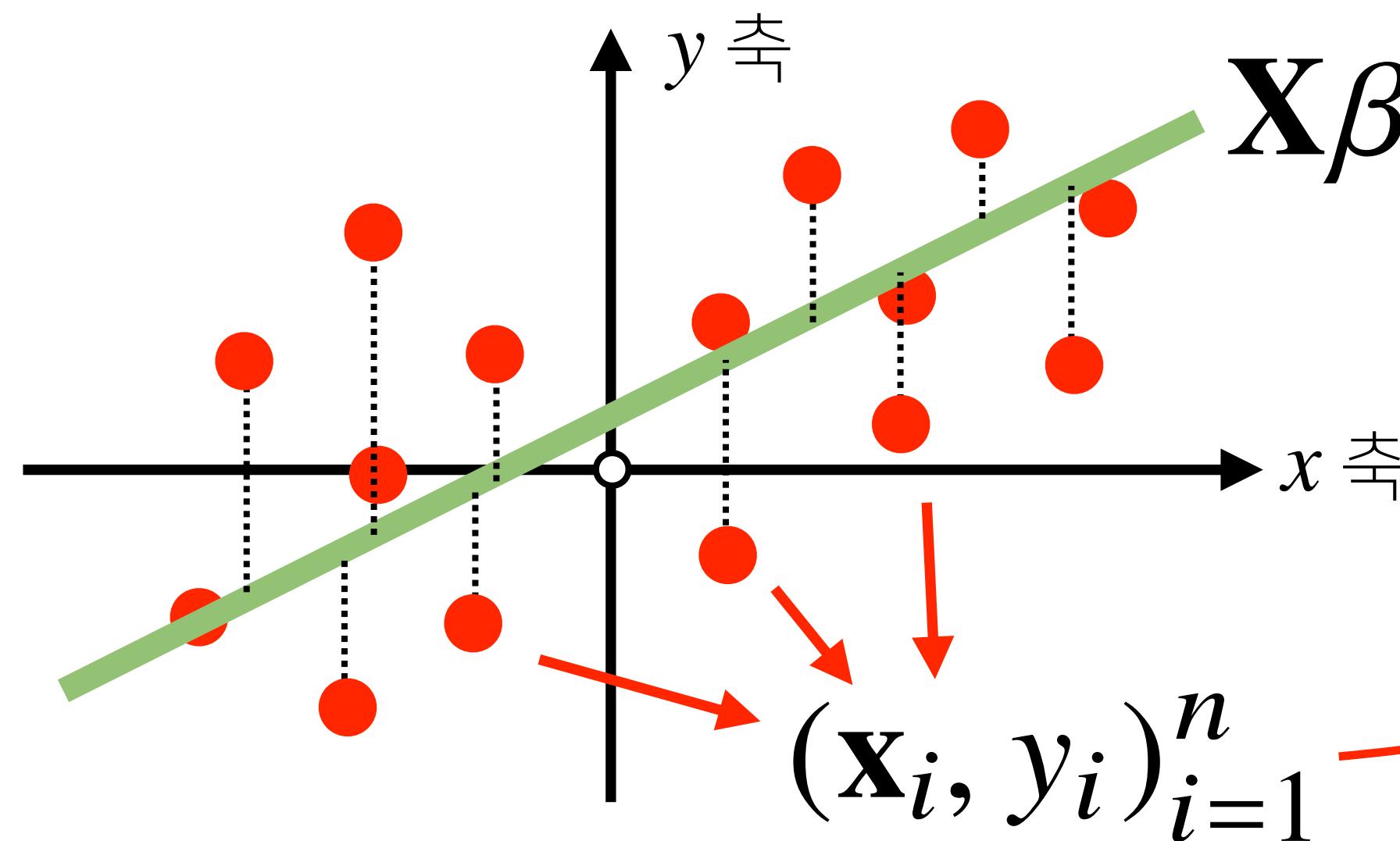
$L_2$ -노름을 최소화



선형모델의 경우 위와 같이 역행렬을 이용해서 회귀분석이 가능하다

# 선형회귀분석 복습

- `np.linalg.pinv` 를 이용하면 데이터를 선형모델(linear model)로 해석하는 선형회귀식을 찾을 수 있다



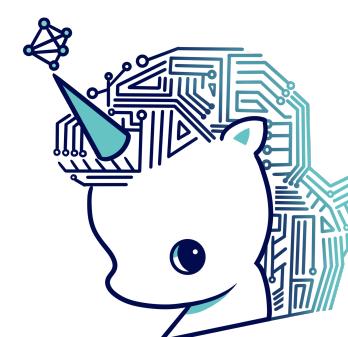
$n \geq m$  인 경우: 데이터가 변수 개수보다 많거나 같아야 함

$\times$

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} \neq \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$
$$\mathbf{X}\beta = \hat{\mathbf{y}} \approx \mathbf{y}$$
$$\min_{\beta} \|\mathbf{y} - \hat{\mathbf{y}}\|_2$$

$\Rightarrow \beta = \mathbf{X}^+ \mathbf{y}$   
 ~~$= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$~~

$L_2$ -노름을 최소화



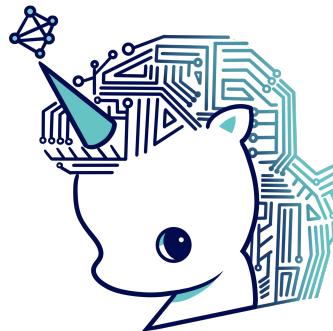
역행렬을 이용하지 말고 경사하강법을  
이용해 적절한 선형모델을 찾아보자

# 경사하강법으로 선형회귀 계수 구하기

---

- 선형회귀의 목적식은  $\|y - X\beta\|_2$ 이고 이를 최소화하는  $\beta$ 를 찾아야 하므로 다음과 같은 그레디언트 벡터를 구해야 한다

$$\nabla_{\beta} \|y - X\beta\|_2 = (\partial_{\beta_1} \|y - X\beta\|_2, \dots, \partial_{\beta_d} \|y - X\beta\|_2)$$



$\|y - X\beta\|_2$  가 아닌  $\|y - X\beta\|_2^2$  를 최소화해도 된다

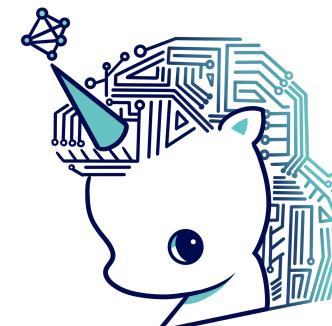
# 경사하강법으로 선형회귀 계수 구하기

---

- 선형회귀의 목적식은  $\|\mathbf{y} - \mathbf{X}\beta\|_2$ 이고 이를 최소화하는  $\beta$ 를 찾아야 하므로 다음과 같은 그레디언트 벡터를 구해야 한다

$$\nabla_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2 = (\partial_{\beta_1} \|\mathbf{y} - \mathbf{X}\beta\|_2, \dots, \partial_{\beta_d} \|\mathbf{y} - \mathbf{X}\beta\|_2)$$

$$\partial_{\beta_k} \|\mathbf{y} - \mathbf{X}\beta\|_2 = \partial_{\beta_k} \left\{ \frac{1}{n} \sum_{i=1}^n \left( y_i - \sum_{j=1}^d X_{ij} \beta_j \right)^2 \right\}^{1/2}$$



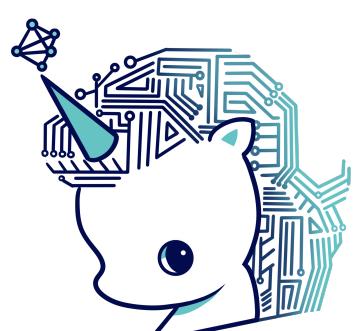
계산이 복잡해보이지만 끈기를 가지고 계산해보자

# 경사하강법으로 선형회귀 계수 구하기

- 선형회귀의 목적식은  $\|\mathbf{y} - \mathbf{X}\beta\|_2$ 이고 이를 최소화하는  $\beta$ 를 찾아야 하므로 다음과 같은 그레디언트 벡터를 구해야 한다

$$\nabla_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2 = (\partial_{\beta_1} \|\mathbf{y} - \mathbf{X}\beta\|_2, \dots, \partial_{\beta_d} \|\mathbf{y} - \mathbf{X}\beta\|_2)$$

$$\partial_{\beta_k} \|\mathbf{y} - \mathbf{X}\beta\|_2 = \partial_{\beta_k} \left\{ \frac{1}{n} \sum_{i=1}^n \left( y_i - \sum_{j=1}^d X_{ij} \beta_j \right)^2 \right\}^{1/2}$$



행렬  $\mathbf{X}$ 의  $k$  번째 열(column) 벡터  
를 전치시킨 것이다

×

$$= -\frac{\mathbf{X}_{\cdot k}^\top (\mathbf{y} - \mathbf{X}\beta)}{n \|\mathbf{y} - \mathbf{X}\beta\|_2}$$

# 경사하강법으로 선형회귀 계수 구하기

---

- 선형회귀의 목적식은  $\|y - X\beta\|_2$ 이고 이를 최소화하는  $\beta$ 를 찾아야 하므로 다음과 같은 그레디언트 벡터를 구해야 한다

$$\begin{aligned}\nabla_{\beta} \|y - X\beta\|_2 &= (\partial_{\beta_1} \|y - X\beta\|_2, \dots, \partial_{\beta_d} \|y - X\beta\|_2) \\ &= \left( -\frac{\mathbf{X}_{\cdot 1}^T(y - X\beta)}{n\|y - X\beta\|_2}, \dots, -\frac{\mathbf{X}_{\cdot d}^T(y - X\beta)}{n\|y - X\beta\|_2} \right)\end{aligned}$$

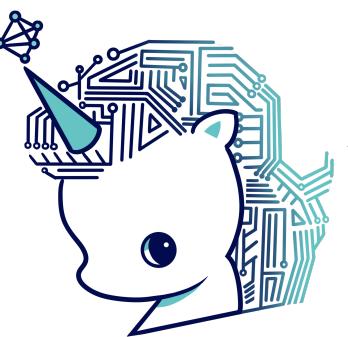
# 경사하강법으로 선형회귀 계수 구하기

---

- 선형회귀의 목적식은  $\|y - X\beta\|_2$ 이고 이를 최소화하는  $\beta$ 를 찾아야 하므로 다음과 같은 그레디언트 벡터를 구해야 한다

$$\nabla_{\beta} \|y - X\beta\|_2 = (\partial_{\beta_1} \|y - X\beta\|_2, \dots, \partial_{\beta_d} \|y - X\beta\|_2)$$

$$= -\frac{X^T(y - X\beta)}{n\|y - X\beta\|_2}$$



복잡한 계산이지만 사실  $X\beta$ 를 계수  $\beta$ 에 대해 미분한 결과인  $X^T$  만 곱해지는 것

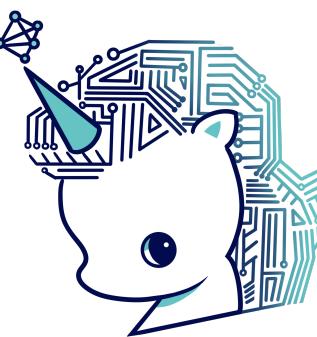
# 경사하강법으로 선형회귀 계수 구하기

---

- 선형회귀의 목적식은  $\|y - X\beta\|_2$ 이고 이를 최소화하는  $\beta$ 를 찾아야 하므로 다음과 같은 그레디언트 벡터를 구해야 한다

$$\nabla_{\beta} \|y - X\beta\|_2 = (\partial_{\beta_1} \|y - X\beta\|_2, \dots, \partial_{\beta_d} \|y - X\beta\|_2)$$

$$= -\frac{X^T(y - X\beta)}{n\|y - X\beta\|_2}$$



지난 시간에 배운 다변수 함수의 경사하강법 알고리즘을 기억해보자

- 이제 목적식을 최소화하는  $\beta$ 를 구하는 경사하강법 알고리즘은 다음과 같다

$$\beta^{(t+1)} \leftarrow \beta^{(t)} - \lambda \nabla_{\beta} \|y - X\beta^{(t)}\|$$

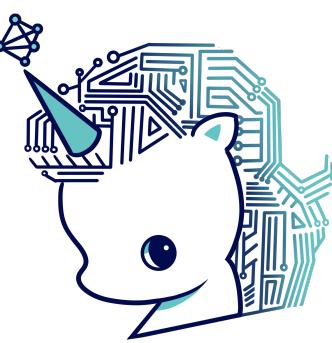
# 경사하강법으로 선형회귀 계수 구하기

---

- 선형회귀의 목적식은  $\|y - X\beta\|_2$ 이고 이를 최소화하는  $\beta$ 를 찾아야 하므로 다음과 같은 그레디언트 벡터를 구해야 한다

$$\nabla_{\beta} \|y - X\beta\|_2 = (\partial_{\beta_1} \|y - X\beta\|_2, \dots, \partial_{\beta_d} \|y - X\beta\|_2)$$

$$= -\frac{X^T(y - X\beta)}{n\|y - X\beta\|_2}$$



$\lambda$  앞에 붙은 마이너스(-) 부호가  
플러스(+) 부호로 바뀐것에 주의할 것!

- 이제 목적식을 최소화하는  $\beta$ 를 구하는 경사하강법 알고리즘은 다음과 같다

$$\beta^{(t+1)} \leftarrow \beta^{(t)} + \frac{\lambda}{n} \frac{X^T(y - X\beta^{(t)})}{\|y - X\beta^{(t)}\|}$$

×

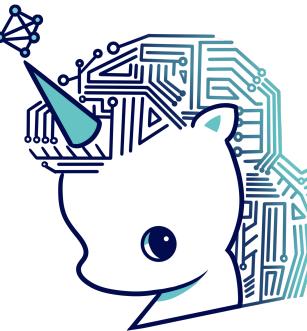
# 경사하강법으로 선형회귀 계수 구하기

---

- 선형회귀의 목적식은  $\|\mathbf{y} - \mathbf{X}\beta\|_2^2$ 이고 이를 최소화하는  $\beta$ 를 찾아야 하므로 다음과 같은 그레디언트 벡터를 구해야 한다

$$\nabla_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 = (\partial_{\beta_1} \|\mathbf{y} - \mathbf{X}\beta\|_2^2, \dots, \partial_{\beta_d} \|\mathbf{y} - \mathbf{X}\beta\|_2^2)$$

$$= -\frac{2}{n} \mathbf{X}^\top (\mathbf{y} - \mathbf{X}\beta)$$



$\|\mathbf{y} - \mathbf{X}\beta\|_2$  대신  $\|\mathbf{y} - \mathbf{X}\beta\|_2^2$ 을 최소화하면 식이 좀 더 간단해진다

- 이제 목적식을 최소화하는  $\beta$ 를 구하는 경사하강법 알고리즘은 다음과 같다

$$\beta^{(t+1)} \leftarrow \beta^{(t)} + \frac{2\lambda}{n} \mathbf{X}^\top (\mathbf{y} - \mathbf{X}\beta^{(t)})$$

×

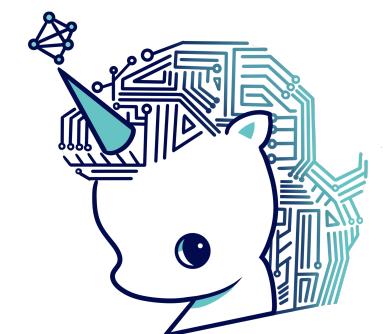
# 경사하강법 기반 선형회귀 알고리즘

---

Input: X, y, lr, T, Output: beta

---

```
# norm: L2-노름을 계산하는 함수  
# lr: 학습률, T: 학습횟수  
  
for t in range(T):  
    error = y - X @ beta  
    grad = - transpose(X) @ error  
    beta = beta - lr * grad
```



종료조건을 일정 학습횟수로 변경한 점만 빼고  
앞에서 배운 경사하강법 알고리즘과 같다

# 경사하강법 기반 선형회귀 알고리즘

---

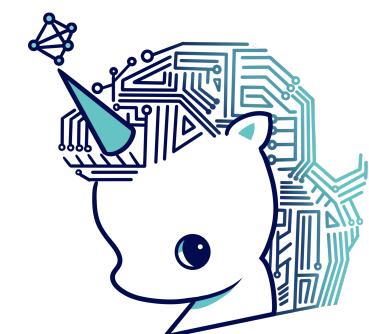
Input:  $X, y, lr, T$ , Output:  $\beta$

---

# norm: L2-노름을 계산하는 함수

# lr: 학습률, T: 학습횟수

```
for t in range(T):
    error = y - X @ beta
    grad = - transpose(X) @ error
    beta = beta - lr * grad
```



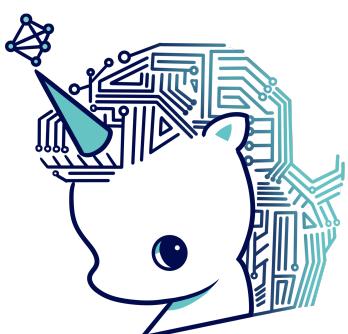
$\nabla_{\beta} \|y - X\beta\|_2^2$  항을 계산해서  $\beta$ 를 업데이트한다

# 경사하강법 기반 선형회귀 알고리즘

Input: X, y, lr, T, Output: beta

```
# norm: L2-노름을 계산하는 함수
# lr: 학습률, T: 학습횟수

for t in range(T):
    error = y - X @ beta
    grad = - transpose(X) @ error
    beta = beta - lr * grad
```



이제 경사하강법 알고리즘으로 역행렬을 이용하지 않고 회귀계수를 계산할 수 있다

```
1 x = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
2 y = np.dot(x, np.array([1, 2])) + 3
3
4 beta_gd = [10.1, 15.1, -6.5] # [1, 2, 3] 이 정답
5 x_ = np.array([np.append(x,[1]) for x in x]) # intercept 항 추가
6
7 for t in range(5000):
8     error = y - x_ @ beta_gd
9     # error = error / np.linalg.norm(error)
10    grad = - np.transpose(x_) @ error
11    beta_gd = beta_gd - 0.01 * grad
12
13 print(beta_gd)

[1.00000367 1.99999949 2.99999516]
```

# 경사하강법 기반 선형회귀 알고리즘

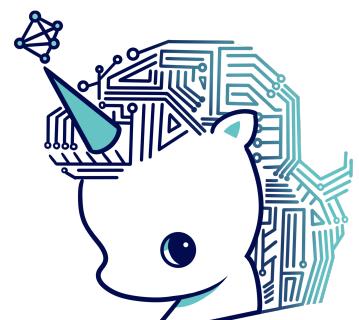
Input: X, y, lr, T, Output: beta

```
# norm: L2-노름을 계산하는 함수
# lr: 학습률, T: 학습횟수

for t in range(T):
    error = y - X @ beta
    grad = - transpose(X) @ error
    beta = beta - lr * grad
```

```
1 X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
2 y = np.dot(X, np.array([1, 2])) + 3
3
4 beta_gd = [10.1, 15.1, -6.5] # [1, 2, 3] 이 정답
5 X_ = np.array([np.append(x,[1]) for x in X]) # intercept 항 추가
6
7 for t in range(100):
8     error = y - X_ @ beta_gd
9     # error = error / np.linalg.norm(error)
10    grad = - np.transpose(X_) @ error
11    beta_gd = beta_gd - 0.01 * grad
12
13 print(beta_gd)
```

[ 3.41314549 4.63604548 -6.69249764]

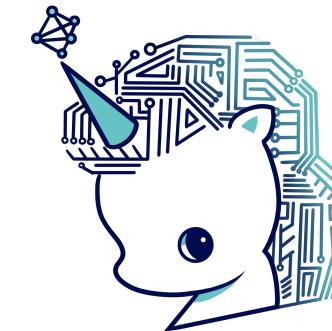


그러나 경사하강법 알고리즘에선 **학습률**과 **학습횟수**가 중요한 hyperparameter 가 된다

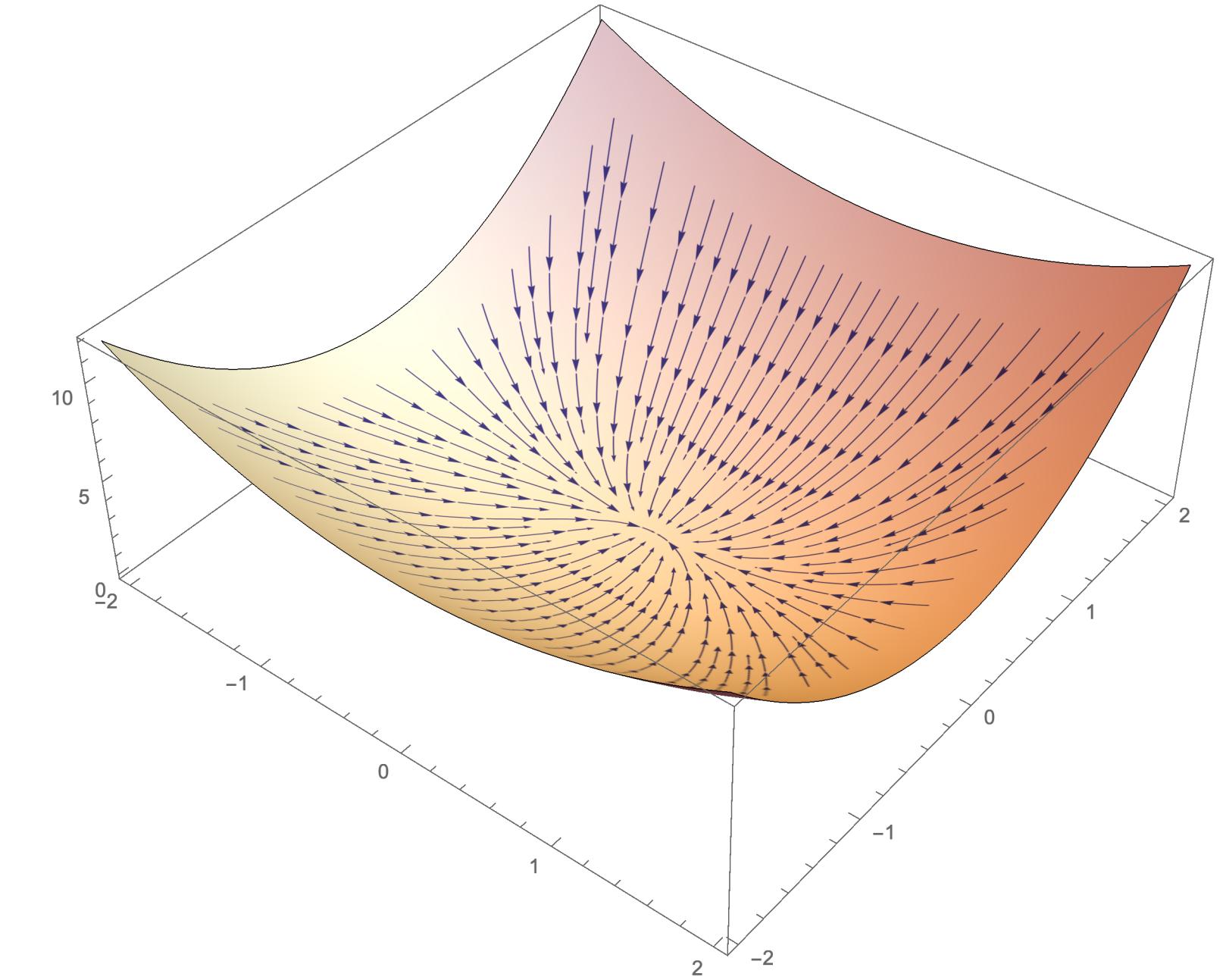
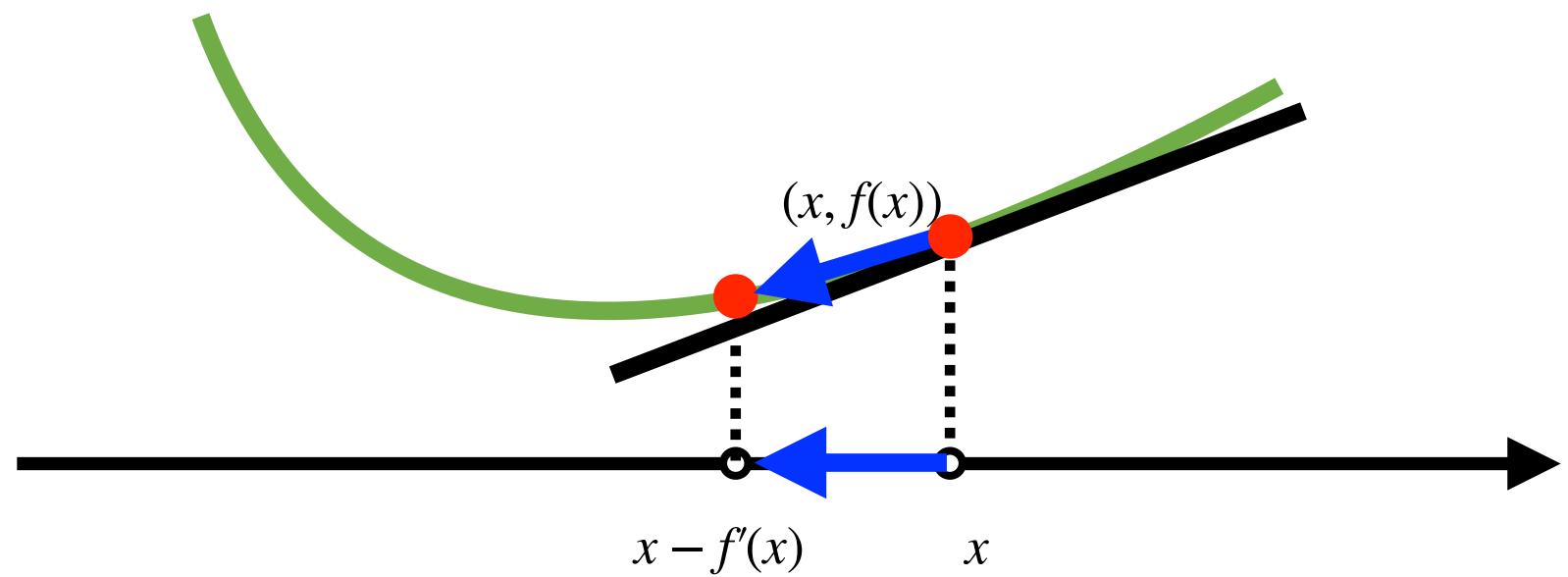
X

# 경사하강법은 만능일까?

- 이론적으로 경사하강법은 미분가능하고 볼록(convex)한 함수에 대해선 적절한 학습률과 학습횟수를 선택했을 때 수렴이 보장되어 있습니다

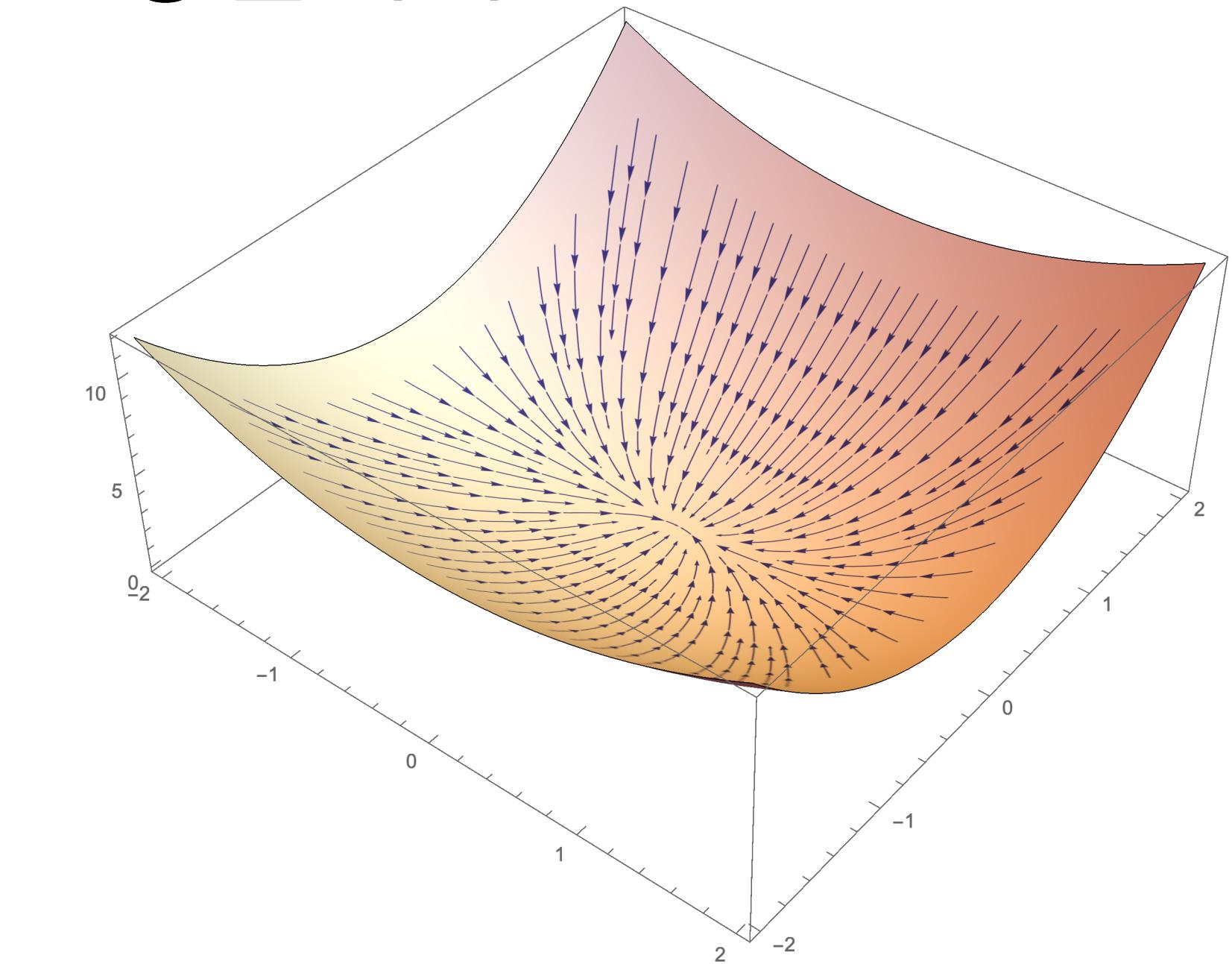
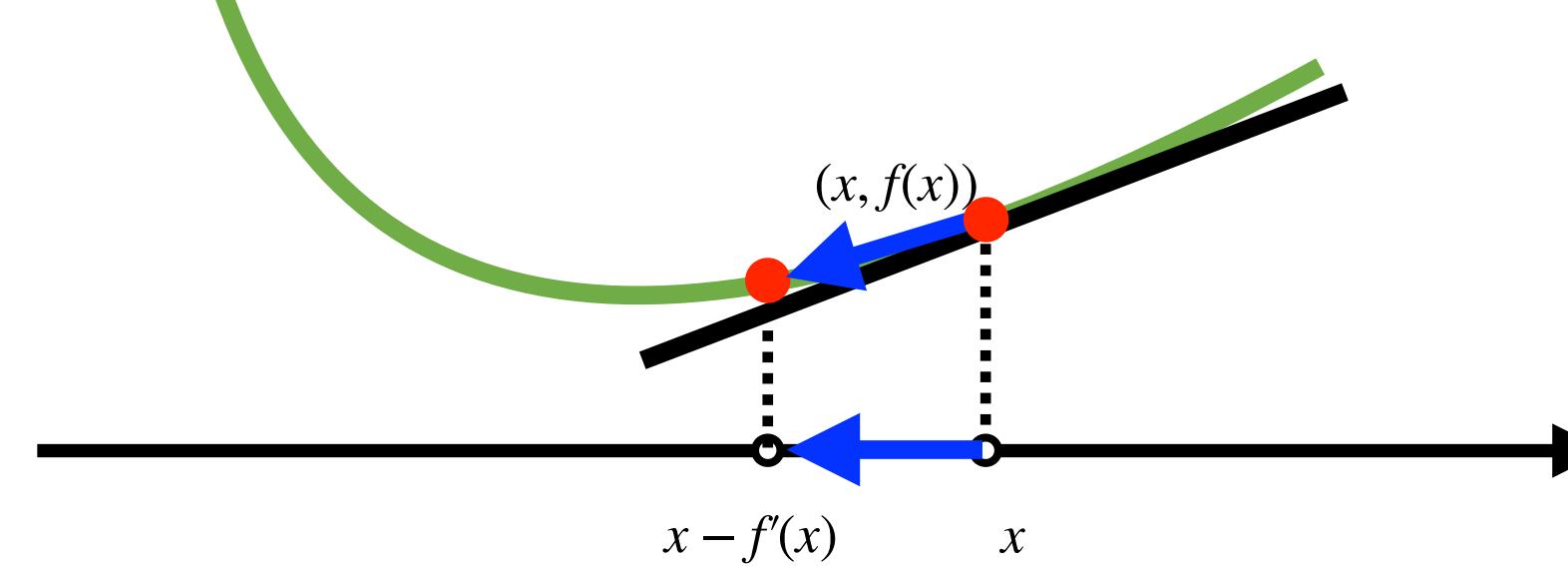
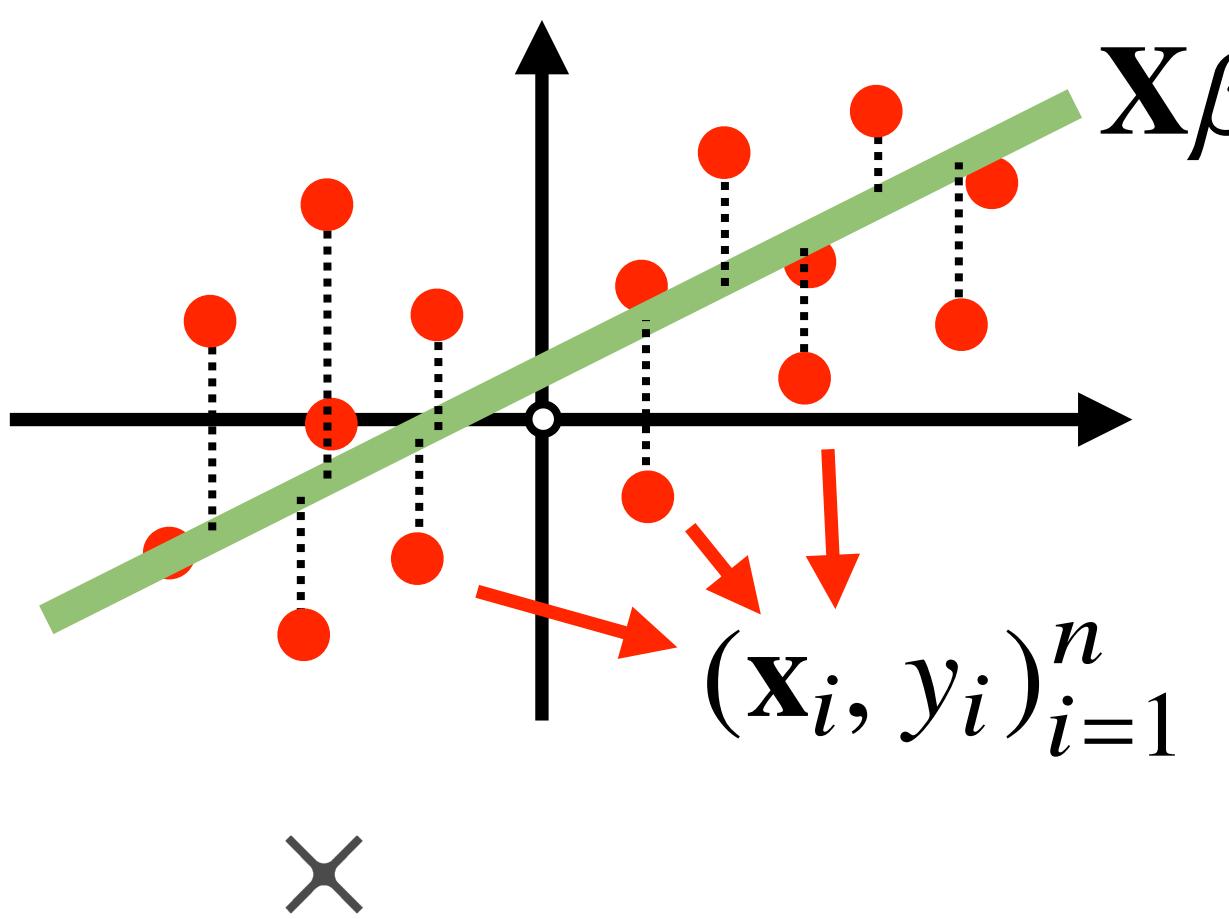


볼록한 함수는 그레디언트 벡터가 항상 최소점을 향한다



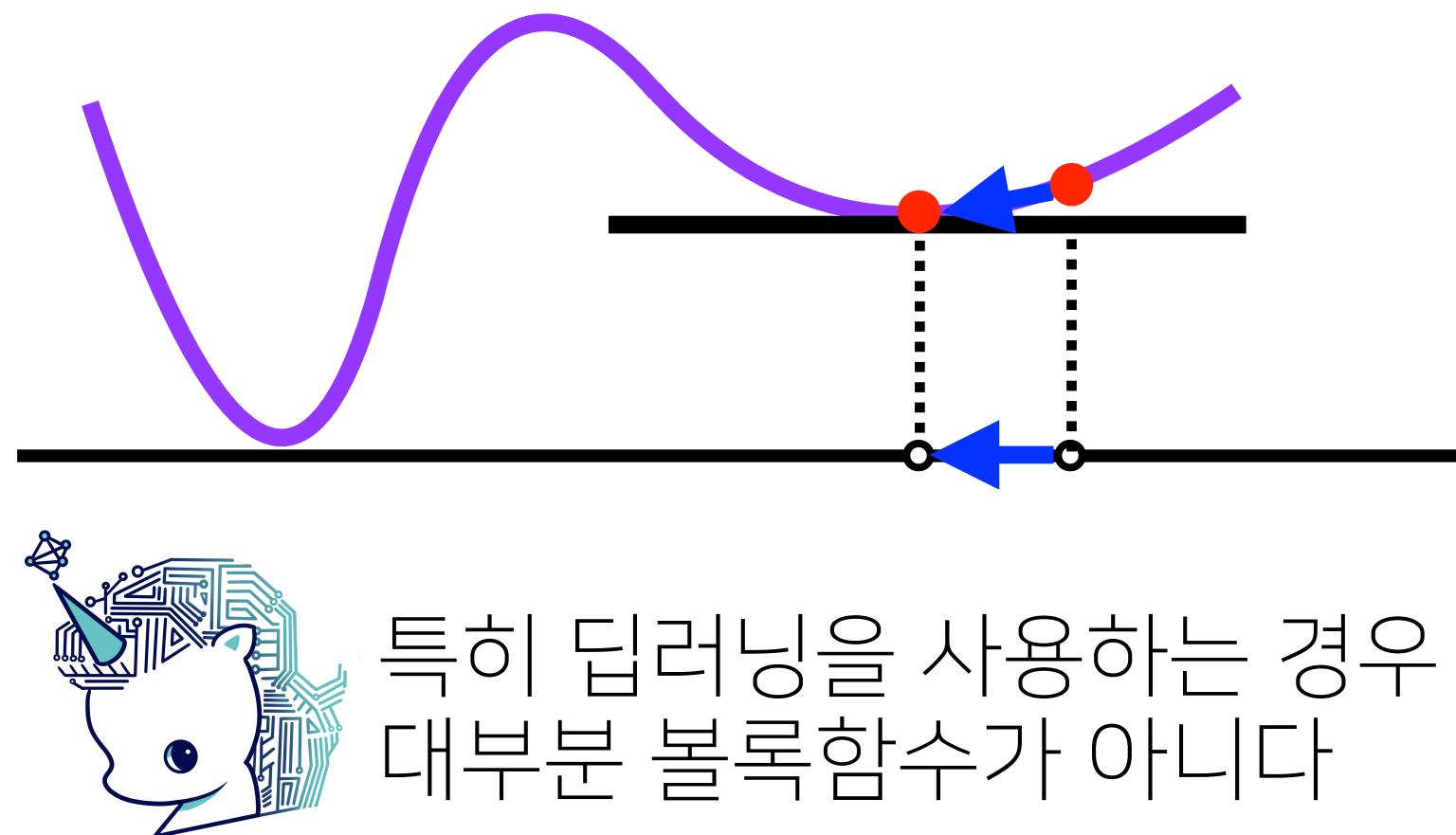
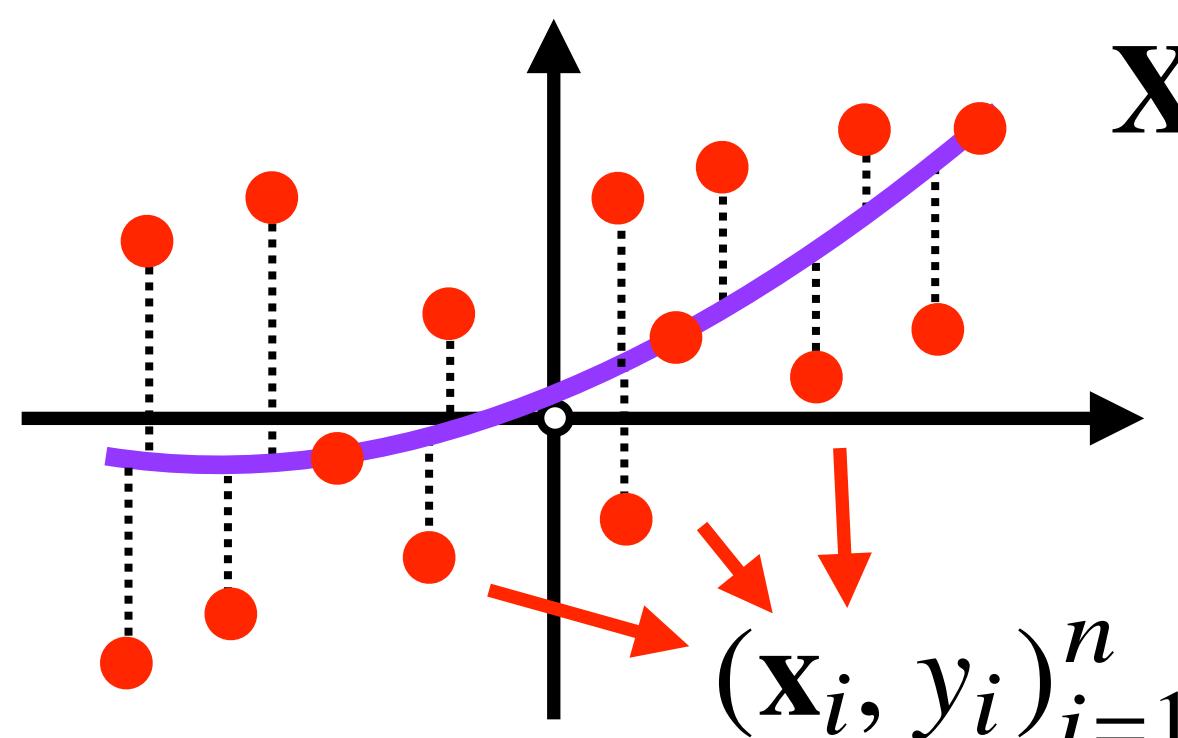
# 경사하강법은 만능일까?

- 이론적으로 경사하강법은 미분가능하고 볼록(convex)한 함수에 대해선 적절한 학습률과 학습횟수를 선택했을 때 수렴이 보장되어 있습니다
- 특히 선형회귀의 경우 목적식  $\|y - X\beta\|_2$ 은 회귀계수  $\beta$ 에 대해 볼록함수이기 때문에 알고리즘을 충분히 돌리면 수렴이 보장됩니다



# 경사하강법은 만능일까?

- 이론적으로 경사하강법은 미분가능하고 볼록(convex)한 함수에 대해선 적절한 학습률과 학습횟수를 선택했을 때 수렴이 보장되어 있습니다
- 특히 선형회귀의 경우 목적식  $\|y - X\beta\|_2$ 은 회귀계수  $\beta$ 에 대해 볼록함 수이기 때문에 알고리즘을 충분히 돌리면 수렴이 보장됩니다
- 하지만 비선형회귀 문제의 경우 목적식이 볼록하지 않을 수 있으므로 수렴이 항상 보장되지는 않습니다



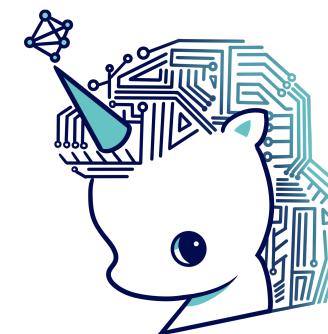
특히 딥러닝을 사용하는 경우 목적식은 대부분 볼록함수가 아니다

# 확률적 경사하강법

---

- 확률적 경사하강법(stochastic gradient descent)은 모든 데이터를 사용해서 업데이트하는 대신 데이터 한개 또는 일부 활용하여 업데이트합니다
- 볼록이 아닌(non-convex) 목적식은 SGD를 통해 최적화할 수 있습니다

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \widehat{\nabla_{\theta} \mathcal{L}}(\theta^{(t)}) \quad \mathbb{E}[\widehat{\nabla_{\theta} \mathcal{L}}] \approx \nabla_{\theta} \mathcal{L}$$



SGD 라고 해서 만능은 아니지만 딥러닝의 경우 SGD 가  
경사하강법보다 실증적으로 더 낫다고 검증되었다

# 확률적 경사하강법

---

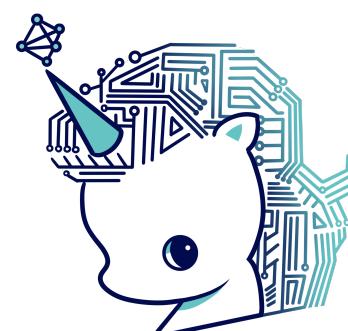
- 확률적 경사하강법(stochastic gradient descent)은 모든 데이터를 사용해서 업데이트하는 대신 데이터 한개 또는 일부 활용하여 업데이트합니다
- 볼록이 아닌(non-convex) 목적식은 SGD를 통해 최적화할 수 있습니다

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \widehat{\nabla_{\theta} \mathcal{L}}(\theta^{(t)}) \quad \mathbb{E}[\widehat{\nabla_{\theta} \mathcal{L}}] \approx \nabla_{\theta} \mathcal{L}$$

- SGD는 데이터의 일부를 가지고 패러미터를 업데이트하기 때문에 연산자원을 좀 더 효율적으로 활용하는데 도움이 됩니다

$$\beta^{(t+1)} \leftarrow \beta^{(t)} + \frac{2\lambda}{n} \mathbf{X}^{\top} (\mathbf{y} - \mathbf{X}\beta^{(t)}) \xrightarrow{O(d^2n) \rightarrow O(d^2b)} \beta^{(t+1)} \leftarrow \beta^{(t)} + \frac{2\lambda}{b} \mathbf{X}_{(b)}^{\top} (\mathbf{y}_{(b)} - \mathbf{X}_{(b)}\beta^{(t)})$$

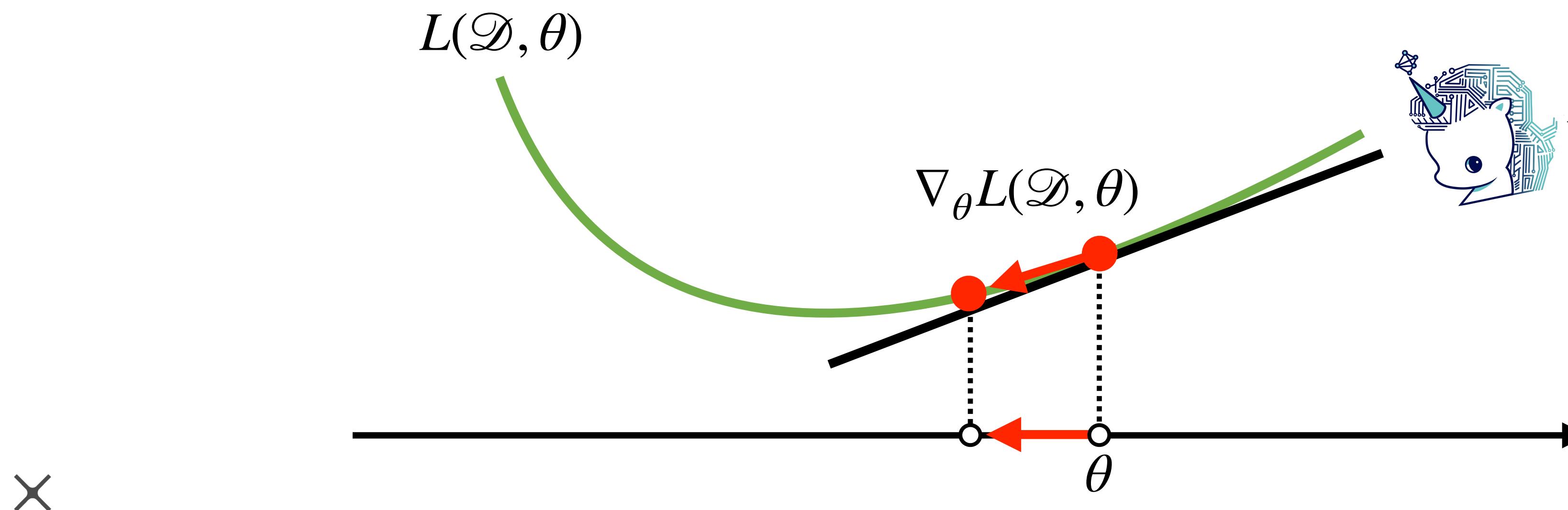
×



전체 데이터  $(\mathbf{X}, \mathbf{y})$  를 쓰지 않고 미니배치  $(\mathbf{X}_{(b)}, \mathbf{y}_{(b)})$  를  
써서 업데이트 하므로 연산량이  $b/n$  로 감소한다

# 확률적 경사하강법의 원리: 미니배치 연산

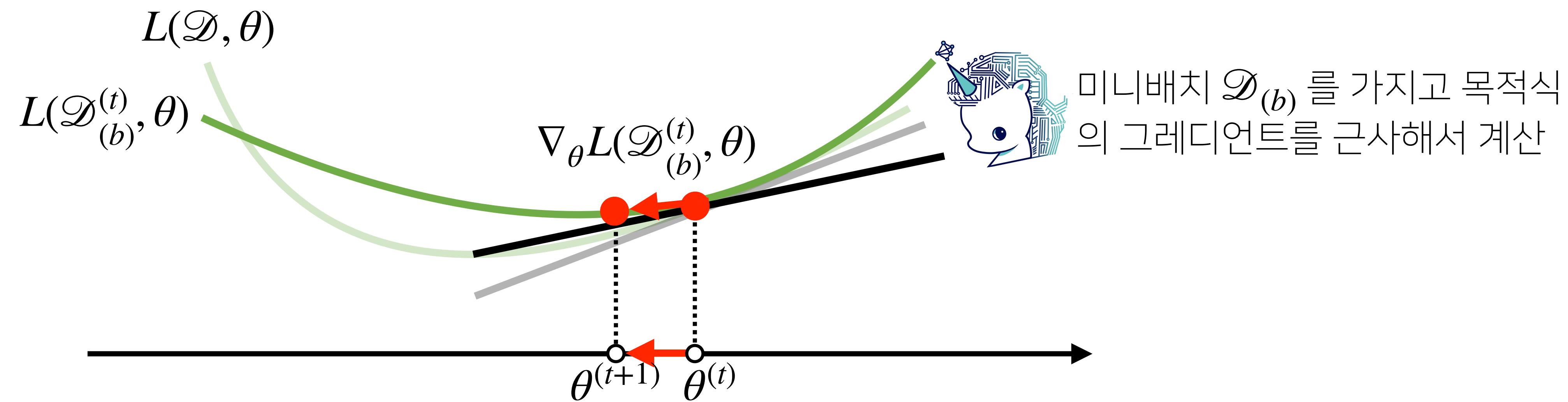
- 경사하강법은 전체데이터  $\mathcal{D} = (\mathbf{X}, \mathbf{y})$  를 가지고 목적식의 그레디언트 벡터인  $\nabla_{\theta} L(\mathcal{D}, \theta)$  를 계산합니다



$L(\mathcal{D}, \theta)$  은 전체데이터  $\mathcal{D}$  와  
파라미터  $\theta$ 로 측정한 목적식

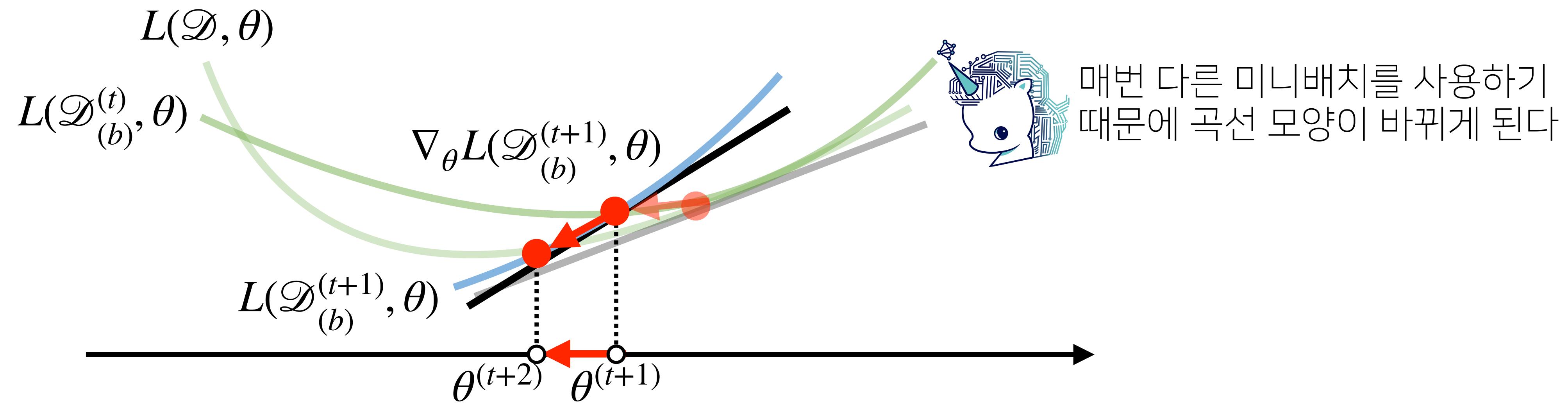
# 확률적 경사하강법의 원리: 미니배치 연산

- 경사하강법은 전체데이터  $\mathcal{D} = (\mathbf{X}, \mathbf{y})$  를 가지고 목적식의 그레디언트 벡터인  $\nabla_{\theta} L(\mathcal{D}, \theta)$  를 계산합니다
- SGD 는 미니배치  $\mathcal{D}_{(b)} = (\mathbf{X}_{(b)}, \mathbf{y}_{(b)}) \subset \mathcal{D}$  를 가지고 그레디언트 벡터를 계산합니다



# 확률적 경사하강법의 원리: 미니배치 연산

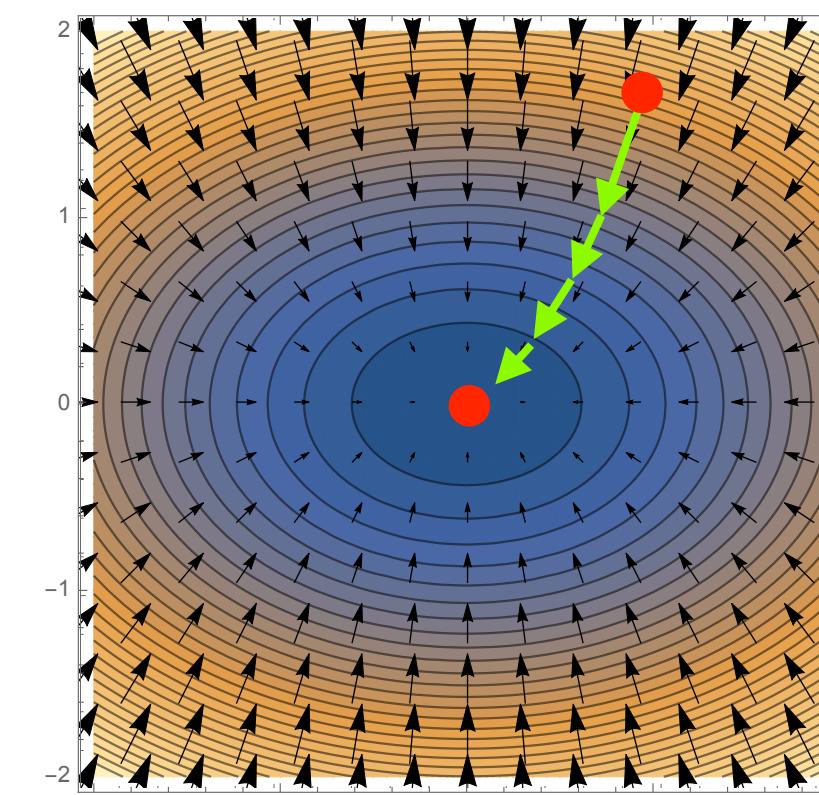
- 경사하강법은 전체데이터  $\mathcal{D} = (\mathbf{X}, \mathbf{y})$  를 가지고 목적식의 그레디언트 벡터인  $\nabla_{\theta} L(\mathcal{D}, \theta)$  를 계산합니다
- SGD 는 미니배치  $\mathcal{D}_{(b)} = (\mathbf{X}_{(b)}, \mathbf{y}_{(b)}) \subset \mathcal{D}$  를 가지고 그레디언트 벡터를 계산합니다. 미니배치는 확률적으로 선택하므로 목적식 모양이 바뀌게 됩니다



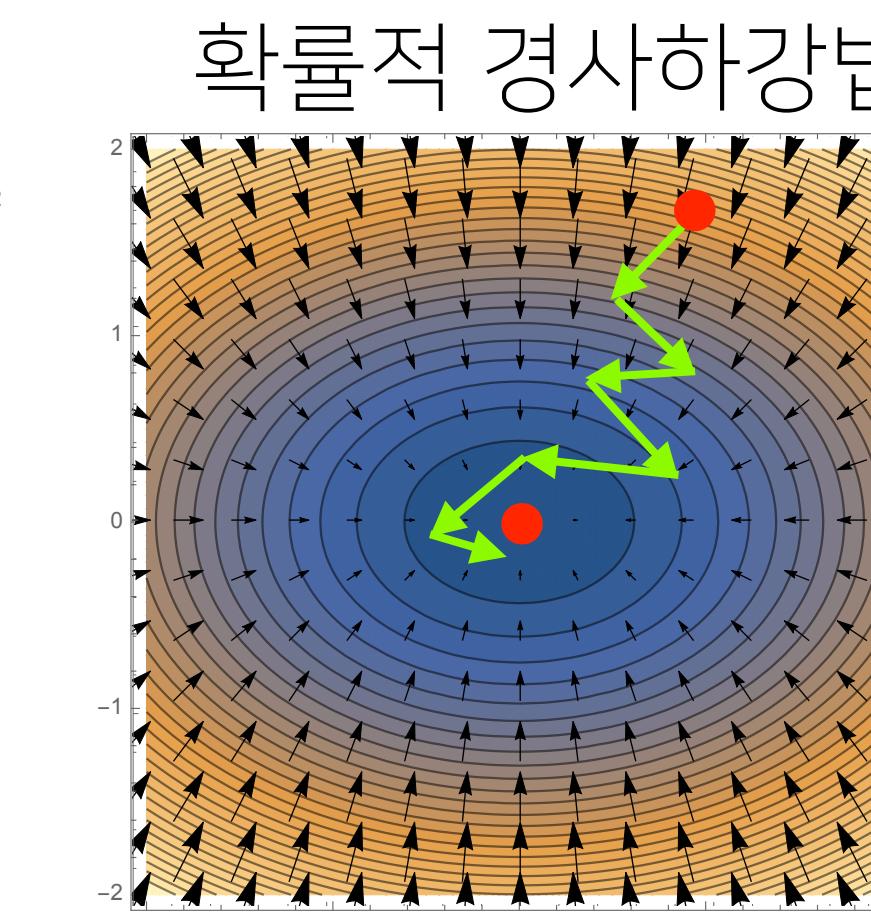
# 확률적 경사하강법의 원리: 미니배치 연산

- 경사하강법은 전체데이터  $\mathcal{D} = (\mathbf{X}, \mathbf{y})$  를 가지고 목적식의 그레디언트 벡터인  $\nabla_{\theta} L(\mathcal{D}, \theta)$  를 계산합니다
- SGD는 미니배치  $\mathcal{D}_{(b)} = (\mathbf{X}_{(b)}, \mathbf{y}_{(b)}) \subset \mathcal{D}$  를 가지고 그레디언트 벡터를 계산합니다. 미니배치는 확률적으로 선택하므로 목적식 모양이 바뀌게 됩니다
- SGD는 볼록이 아닌 목적식에서도 사용 가능하므로 경사하강법보다 **머신러닝 학습에 더 효율적**입니다

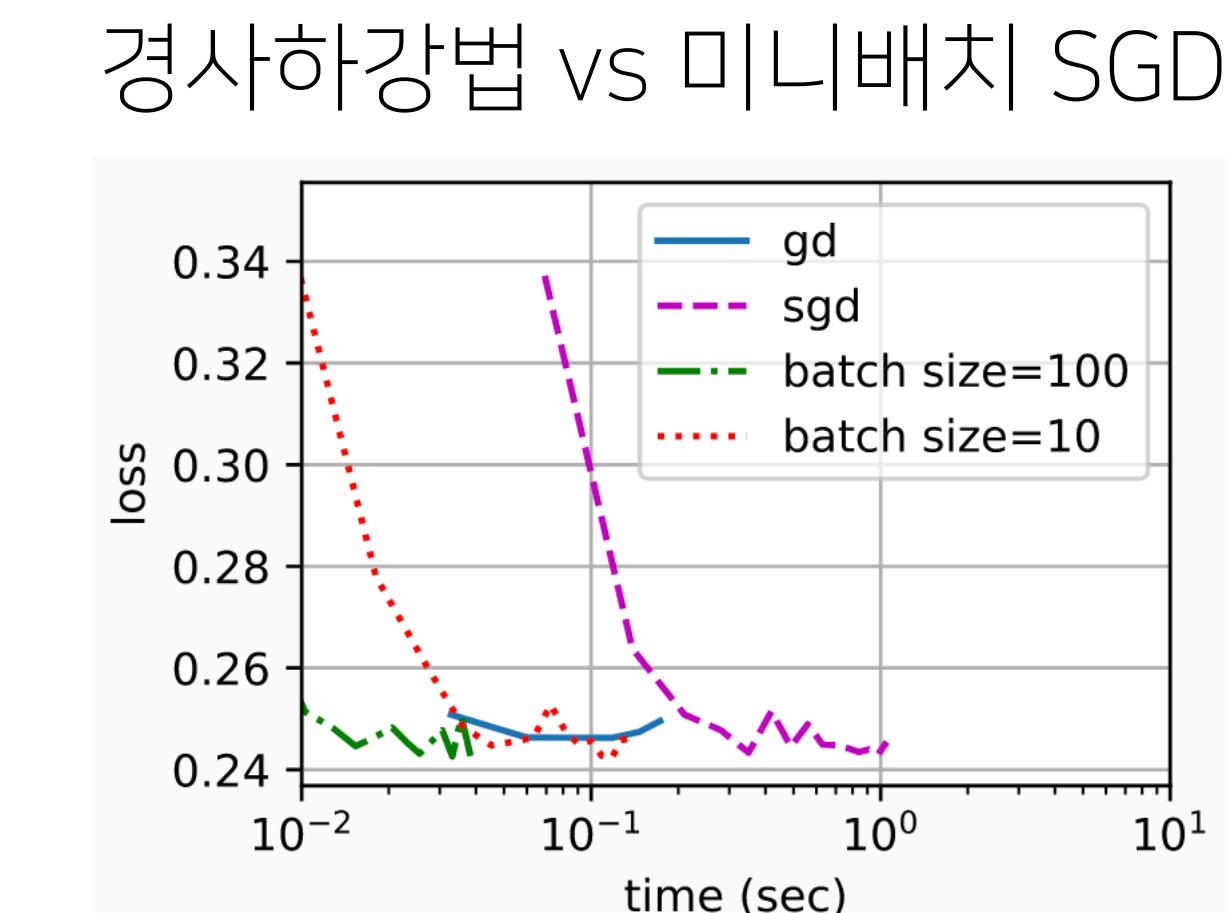
X



경사하강법

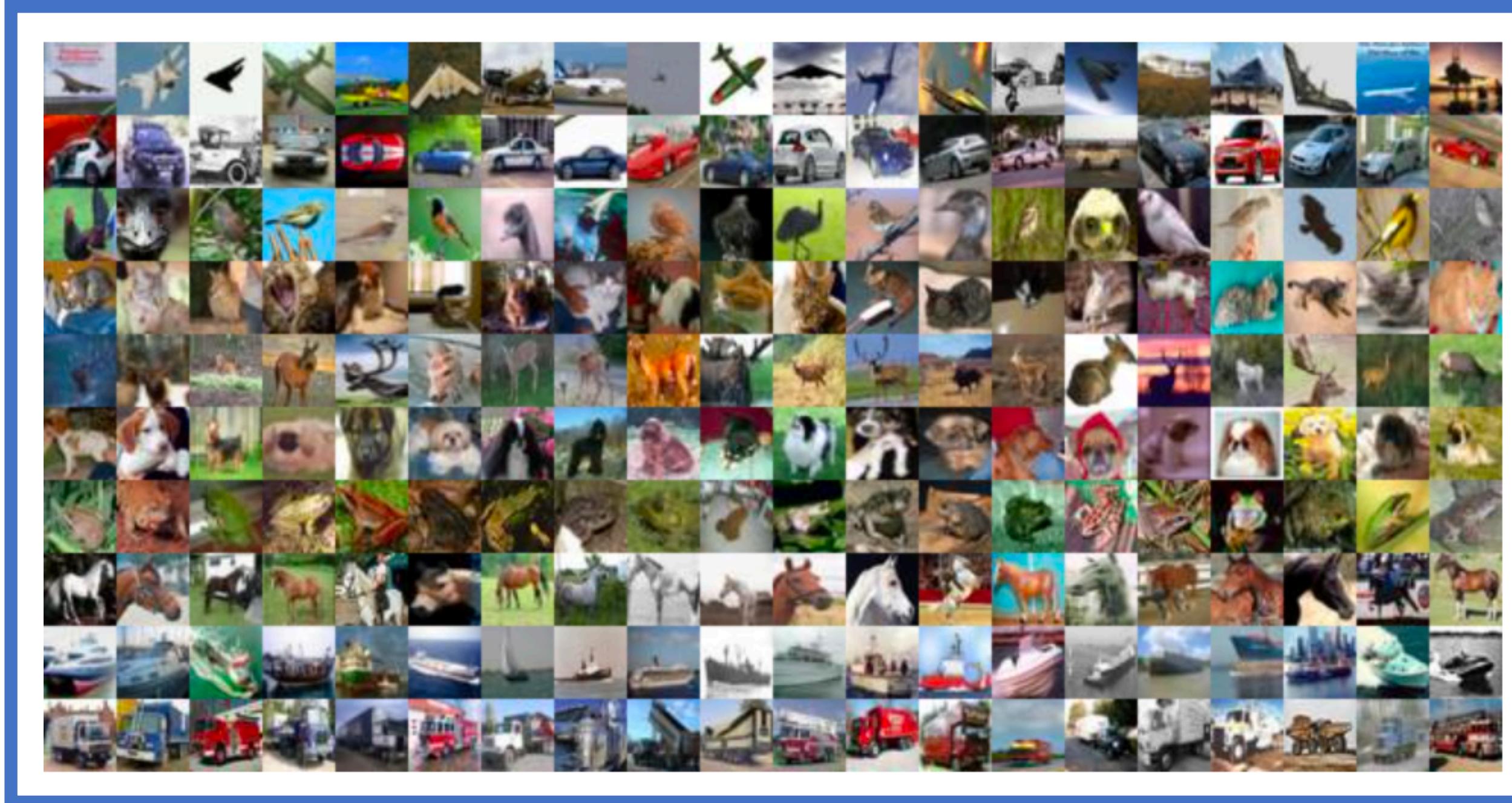


확률적 경사하강법



(출처: Dive into Deep Learning)

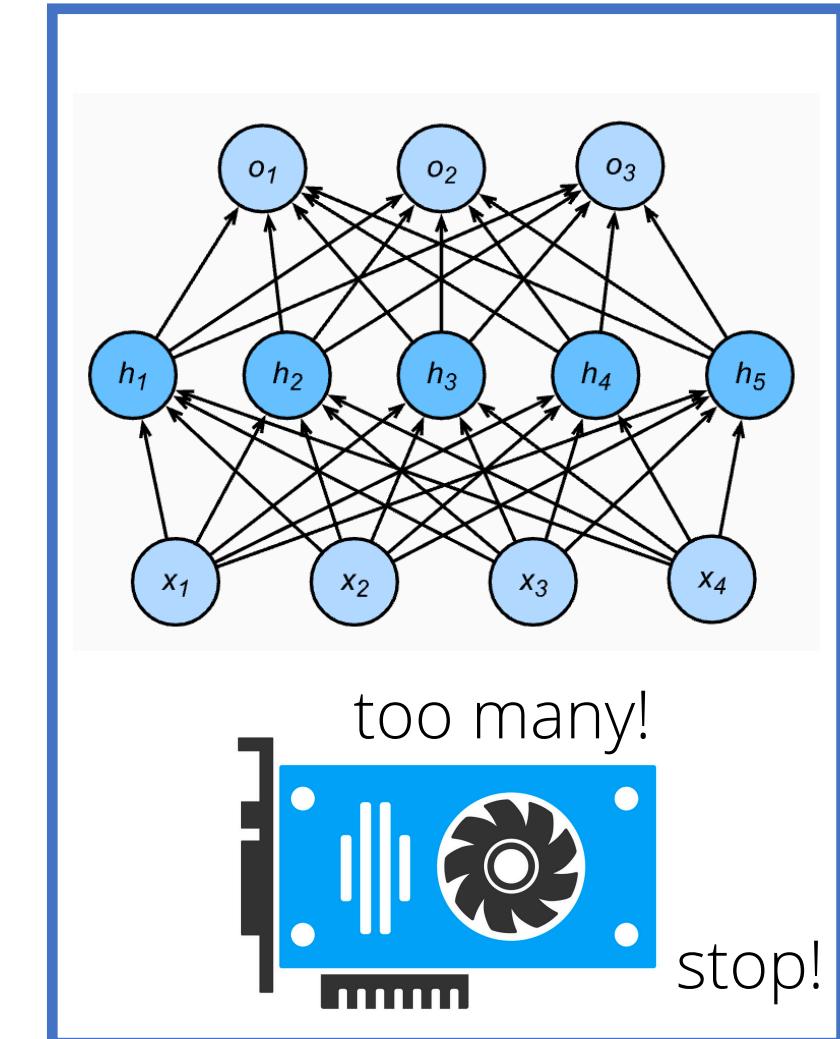
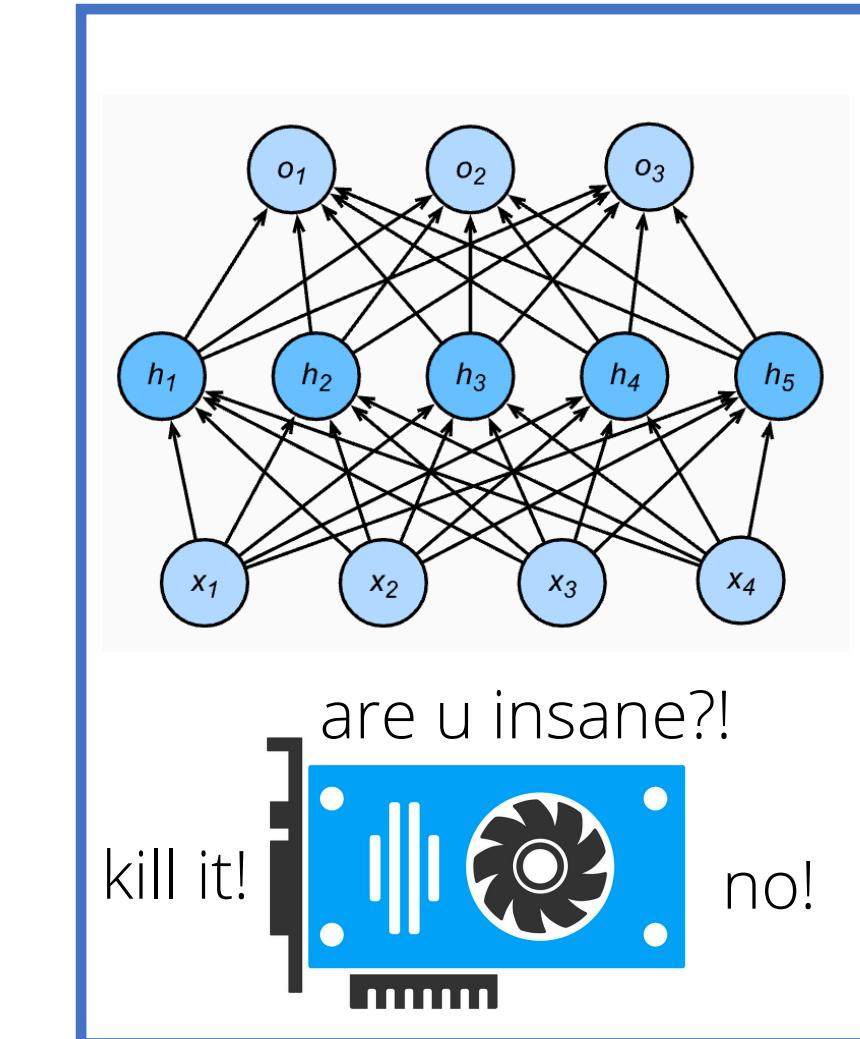
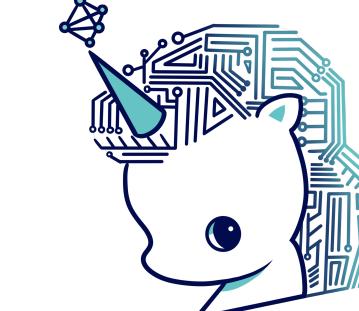
# 확률적 경사하강법의 원리: 하드웨어



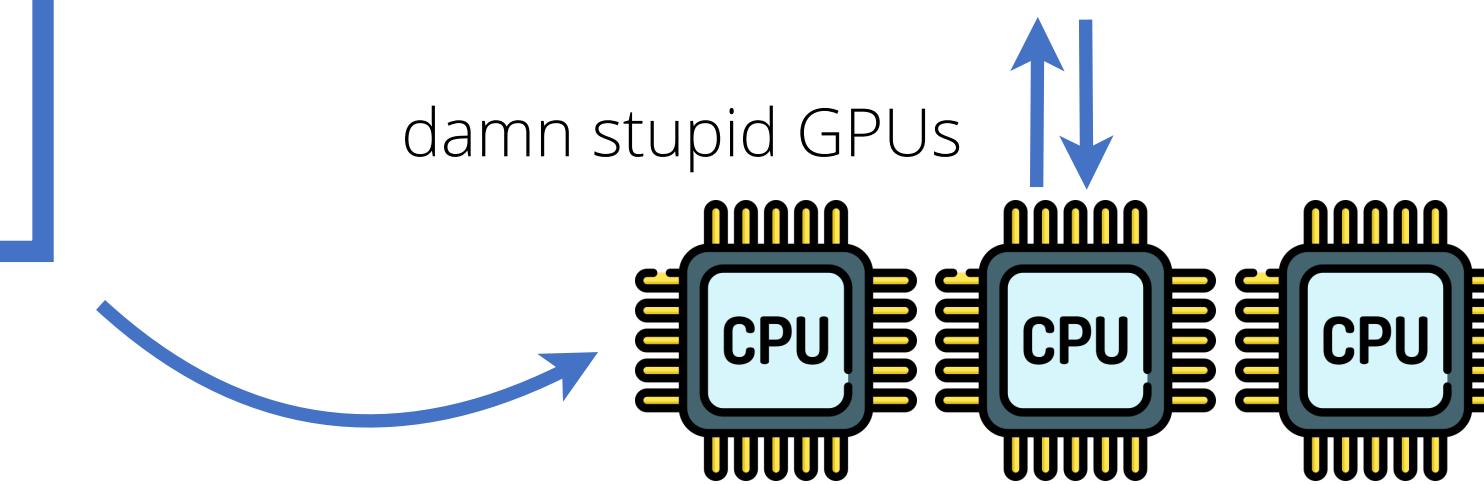
$$256 \times 256 \times 3 \times 1,000,000 \approx 2^{37} \text{ bytes}$$

이미지 데이터

X



damn stupid GPUs

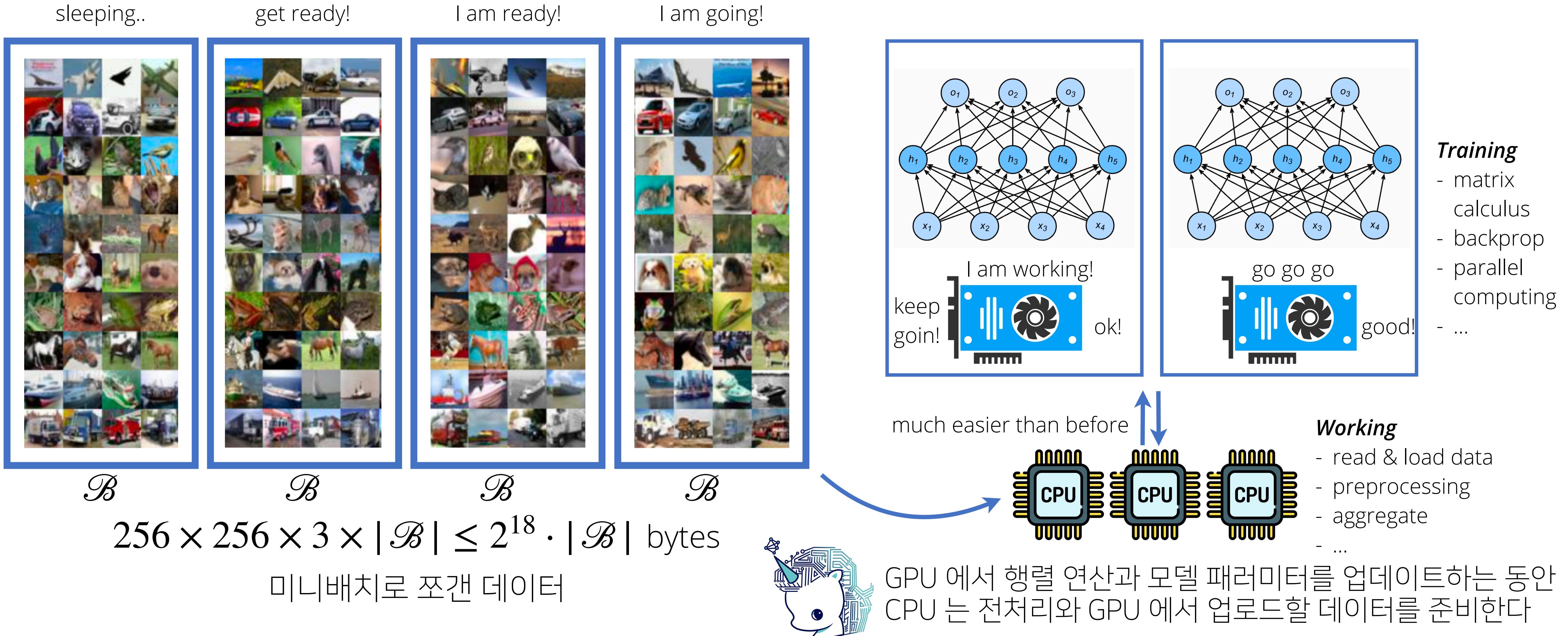


Working

- read & load data
- preprocessing
- aggregate
- ...

만일 일반적인 경사하강법처럼 모든 데이터를 업로드하면  
메모리가 부족하여 Out-of-memory 가 발생한다

# 확률적 경사하강법의 원리: 하드웨어



# THE END

---

다음 시간에 보아요!

# Layer Selection ↵ MDP

