# 4.5 Batch

# Batch

## 도입

- 우리가 사용하는 데이터는 벡터화 되어 병렬 처리 된다.
- $X = [x^1, x^2, x^3, \dots, x^{100}, \dots, x^{5000}]$
- $Y = [y^1, y^2, y^3, \dots, y^{100}, \dots, y^{5000}]$

실제 처리되는 데이터의 단위는 어떻게 될까?

- 데이터가 너무 많음! 메모리 부족
- 한번 계산으로 최적화(optiization) 하기는 너무 어렵다.
- 여러번 학습시키기 위해 데이터를 나눈다.

## 데이터를 나누기 위한 단위

### Epoch

- 전체 sample 데이터를 이용하여 한 바퀴 돌며 학습하는 것을 1회 epoch이라 한다.
- 예) 2 epochs은 전체 sample을 이용하여 두 바퀴를 돌며 학습한 것이다.

### Step

- Weight와 Bias를 1회 업데이트하는 것을 1 Step이라 부른다.

### Batch Size

- 1 Step에서 사용한 데이터의 수이다.

- 예) Batch Size가 100이라고 가정하고 Step이 5이면 약 500개의 데이터를 이용한 것이다.

## 관계

아래와 같은 관계가 성립한다.

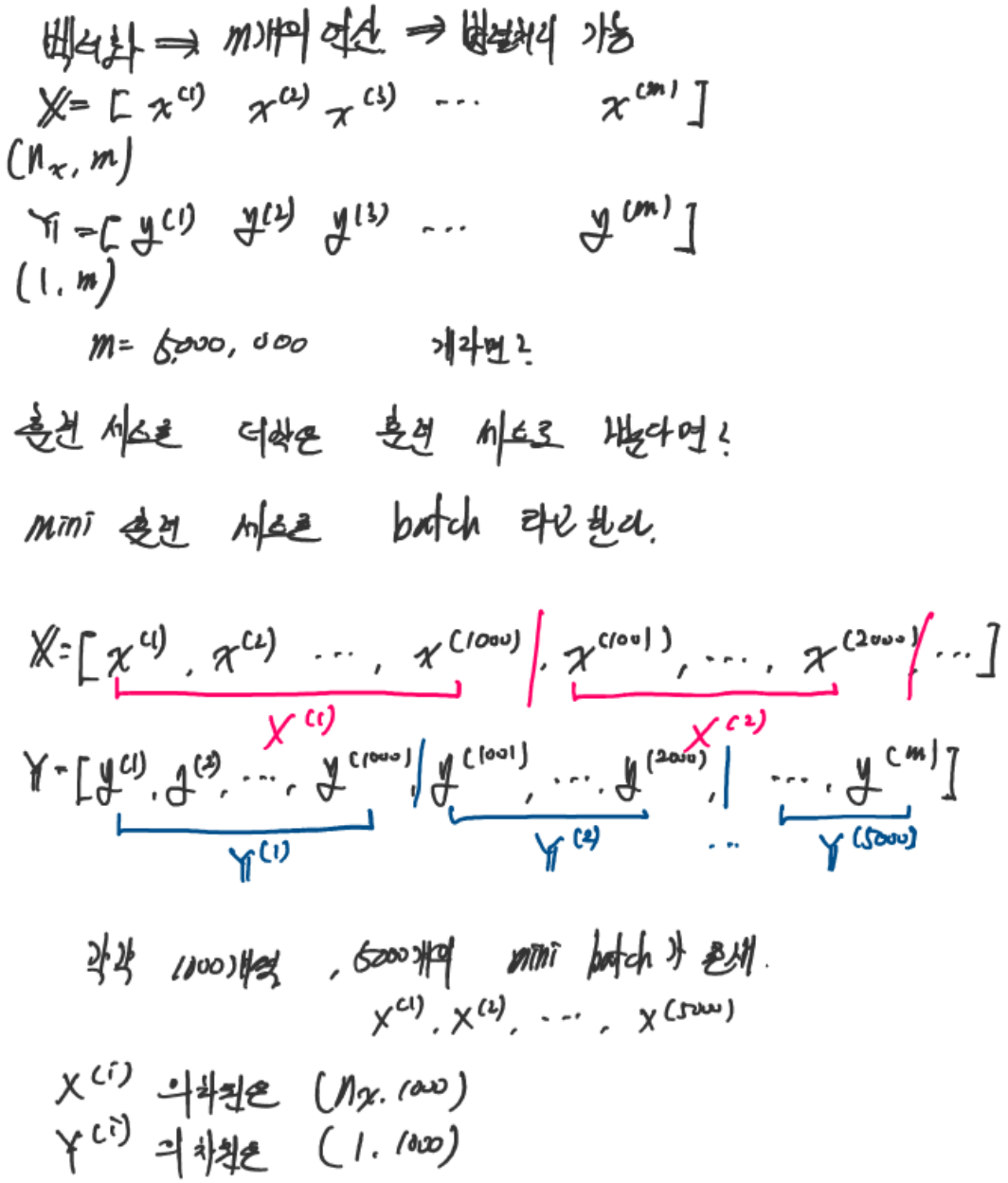
**s = (n * e) / b**

n = num of sample : 전체 학습할 데이터의 개수

e = epochs: Epoch 수

b = batch size: 배치 사이즈

s = steps: Step 수




# Batch Size 에 따른 Gradient Descent 과정

- Gradient Descent 의 Update 과정은 다음과 같이 이루어 지고 있다.
- Batch Size (m) 에 따른 학습률 변화는 어떻게 이루어 질까?

Cost Function – “One Half Mean Squared Error”:

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Objective:

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

Update rules:

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

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

Derivatives:

$$\frac{\partial}{\partial\theta_0}J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$$

$$\frac{\partial}{\partial\theta_1}J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)\cdot x^{(i)}$$

## Gradient Descent( Full Batch Gradient Descent )

- 일반적으로 GD 라고 알려진 방식은 Full Batch Gradient Descent 를 의미 한다.
- Batch Size = m = 전체 를 의미함.
- Batch Size 가 m 이면 1개의 Batch만 존재하고 1Epoch 당 1Batch 가 학습되어 Update 됨.

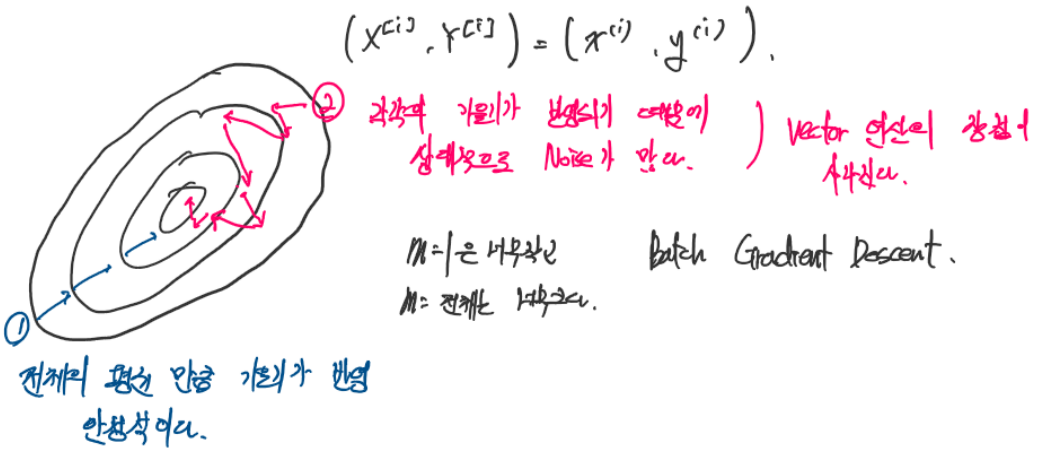
## Stochastic Gradient Descent

- Batch Size = m = 1 (랜덤으로 추출된 1개의 데이터 사용)
- Vector 연상의 장점이 사라지고 한개의 학습 데이터씩 가중치가 업데이트 된다.
- 가중치가 빠르게 업데이트 되므로 학습의 속도가 빨라진다.
- 상대적으로 적은 Epoch 가 사용됨

## Mini Batch Gradient Descent

- Batch Size = m = n
- $2^n$개로 Batch Size 를 지정해 준다. ( 2진법적인 관점)
- 미니 배치 확률적 경사하강법은 SGD의 노이즈를 줄이면서도 전체 배치보다는 더 빠르게 최적점을 구할 수 있습니다.

| Aa Property | 경사하강법 | 확률적 경사하강법 |
|---|---|---|
| 1회의 학습에 사용되는 데이터 | 모든 데이터 사용 | 랜덤으로 추출된 1개의 데이터 사용(중복 선택 가능) |
| 반복에 따른 정확도 | 학습이 반복 될 수록 최적해에 근접 | 학습이 반복 될 수록 최적해에 근접 |
| 노이즈 | 거의 없음 | 비교적 노이즈가 심함 |



```
model.compile(optimizer='adam', loss='sparse_categorical_cros
history = model.fit(train_X, train_Y, epochs=5, batch_size=1
```

참고자료

https://www.ritchieng.com/machine-learning-large-scale/