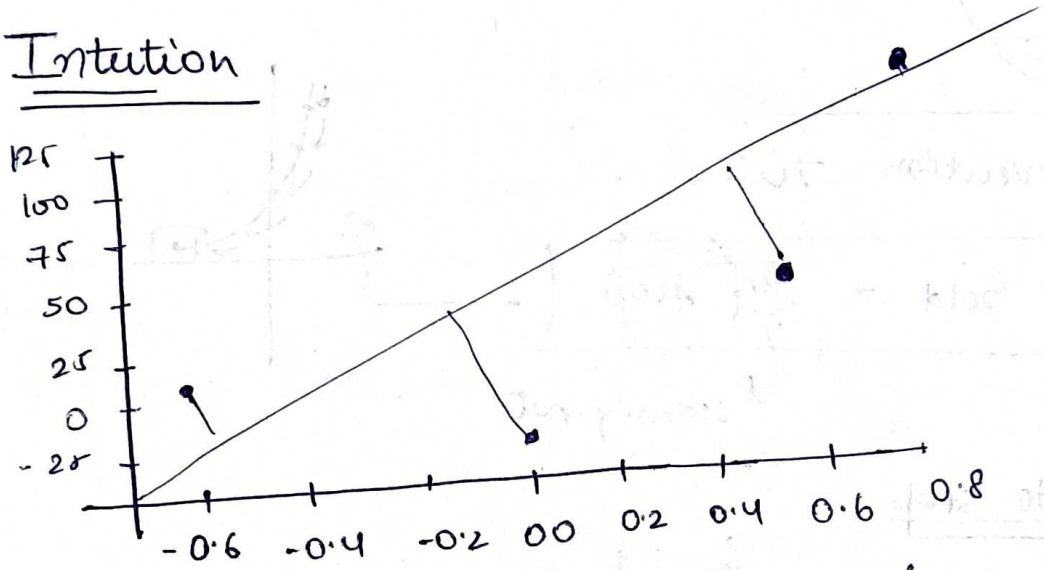
Gradient Descent

Intution

Gradient Descent



4 rows, 2 columns

OLS

sk learn

lib call -

cgpa	lpa
4 rows	

$\hat{y}_i = m y_i + b$

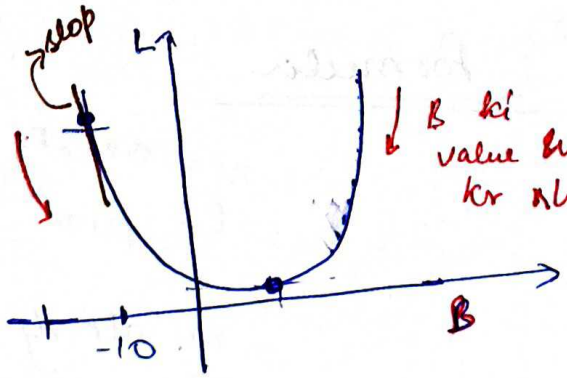
$m = 78.35$

$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

$L = \sum_{i=1}^{n} (y_i - m x_i - b)^2$

$L = \sum_{i=1}^{n} (y_i - 78.35 \times x_i - b)^2$

$L \to b^2$



① Step :- Select aandom b

let → random

b = -10

derivative $x = 5$

$y = x^2 + 2$

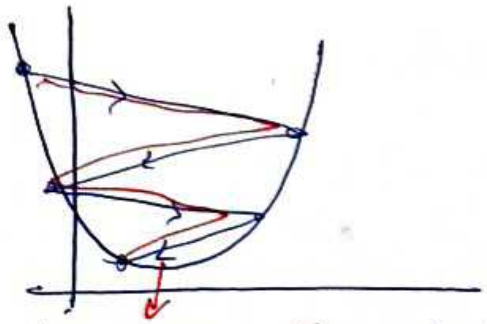
$y = 2x$

$y = 10$

slope = -ve → (B → value increase)

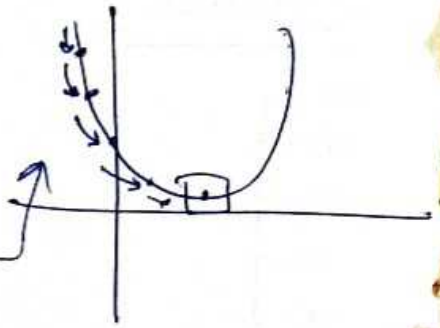
slope = +ve ← (B → value decrease)

$b_{new} = b_{old} - slop$



for prevention this

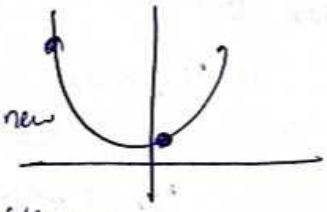
$$b_{new} = b_{old} - \eta \text{ slop}$$



↳ learning rate

When to stop

(1) difference betⁿ b_{old} b_{new} = 0.001



when multiple time difference came same result then need to stop.

$b_{new} - b_{old} = 0.001$

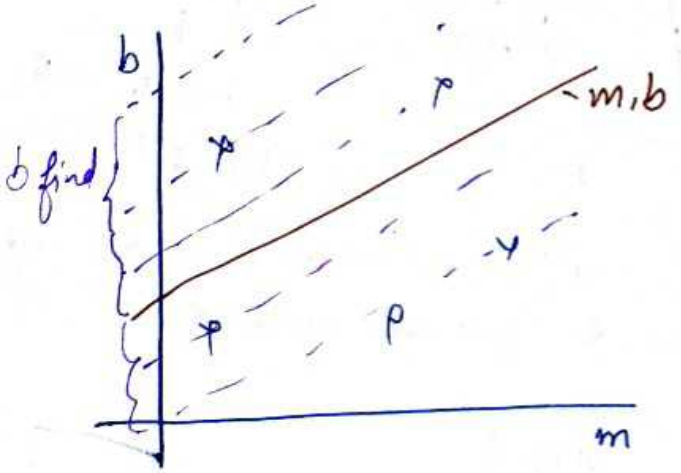
$b_{new} - b_{old} \approx 0$

is close to zero.

(2) Iteration ⇒ limit eg 1000 times, 100 times

epoch → set 1000 or 1000

Mathematical Formula



m = 78.35

↳ from sklearn

m already know we just find b

Step① → start with a random b

for i in epochs:

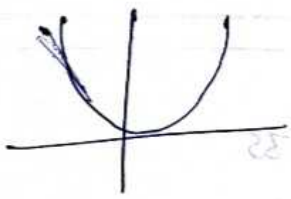
$$b_{new} = b_{old} - \eta \times slope$$

↳ learning rate let $\eta = 0.01$

$b_{new} = b_{old} - \eta \times slope$

↳ finding slope at b=b

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \longrightarrow$$



$$\frac{dL}{db} = \frac{d}{db} \left(\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right)$$

$$\frac{d}{db} \sum_{i=1}^{n} (y_i - mx_i - b)^2 = 2 \sum_{i=1}^{n} (y_i - mx_i - b)(-1)$$

$2 \sum_{i=1}^{n} (y_i - mx_i - b)(0 - 0 - 1)$

$$\text{slope} = -2 \sum_{i=1}^{n} (y_i - mx_i - b)$$

Let b = 0

$$= -2 \sum_{i=1}^{n} (y_i - mx_i - 0)$$

$$= -2 \sum_{i=1}^{n} (y_i - 78.35 x_i)$$

slope = after solve

at b=0

$$b_{new} = b_{old} - \eta \, slope$$

↳ Step size

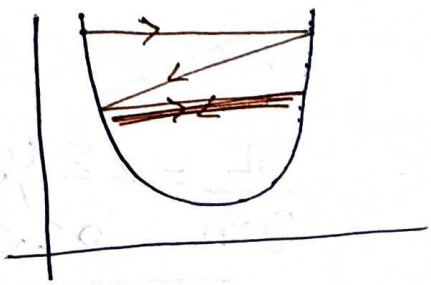
```
from sklearn.datasets import make_regression
import numpy as np

X, y = make_regression(n_samples=4, n_features=1,
                       n_informative=1, n_targets=1, noise=80,
                       random_state=13)
```

```
b = -100
m = 78.35
lr = 0.1
epochs = 15
for i in range(epochs):
    loss_slope = -2 * np.sum(y - m*X.ravel() - b)
    b = b - (lr * loss_slope)
    y_pred = m * X + b
    plt.plot(X, y_pred)
plt.scatter(X, y)
```

Convert [2D or 3D to 1D]

Visualization



if condition is not true,
learning rate ↓↓
epochs ↑↑

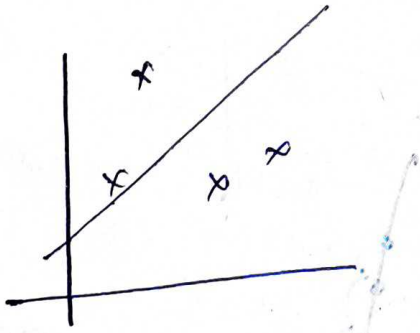
Adding m into the mix

Steps

① init random values of m and b
m=1 and b=0

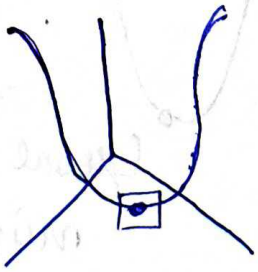
② epochs = 100 , lr = 0.01

for i in epochs:
b = b - n slope
m = m - n slope



$$L = \sum(y_i - \hat{y}_i) = \sum (y_i - mx_i - b)^2$$

$L(m, b)$ like $f(x, y)$



$$b\text{ slope} = \frac{dL}{db}$$

$$m\text{ slope} = \frac{dL}{dm}$$

$$\sum (y_i - mx_i - b)^2$$

for calculating b ↓

$$\frac{\partial L}{\partial b} = -2\sum (y_i - mx_i - b)$$

$$= -2\sum (y_i - mx_i - b)$$

slope at $b=0$

for calculating m ↓

$$\frac{\partial L}{\partial m} = 2\sum (y_i - mx_i - b)(-x_i)$$

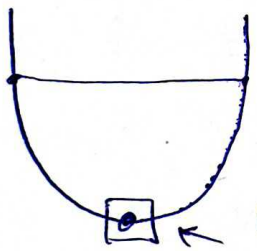
$$= -2\sum (y_i - mx_i - b)x_i$$

Slope m at m=1

Effect of loss function

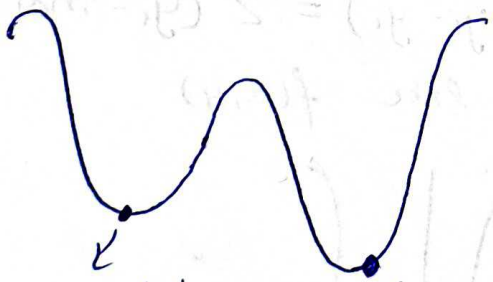
$L = \sum (y_i - \hat{y_i})^2$ → Convex function

Only 1 minima



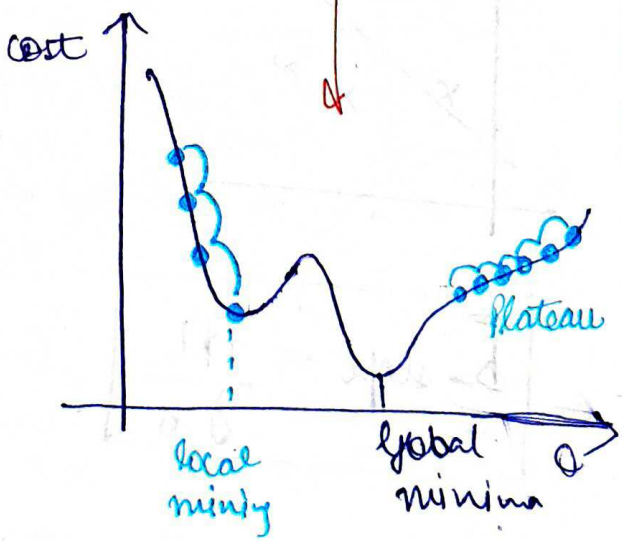
Global / minima

Non-convex ↓



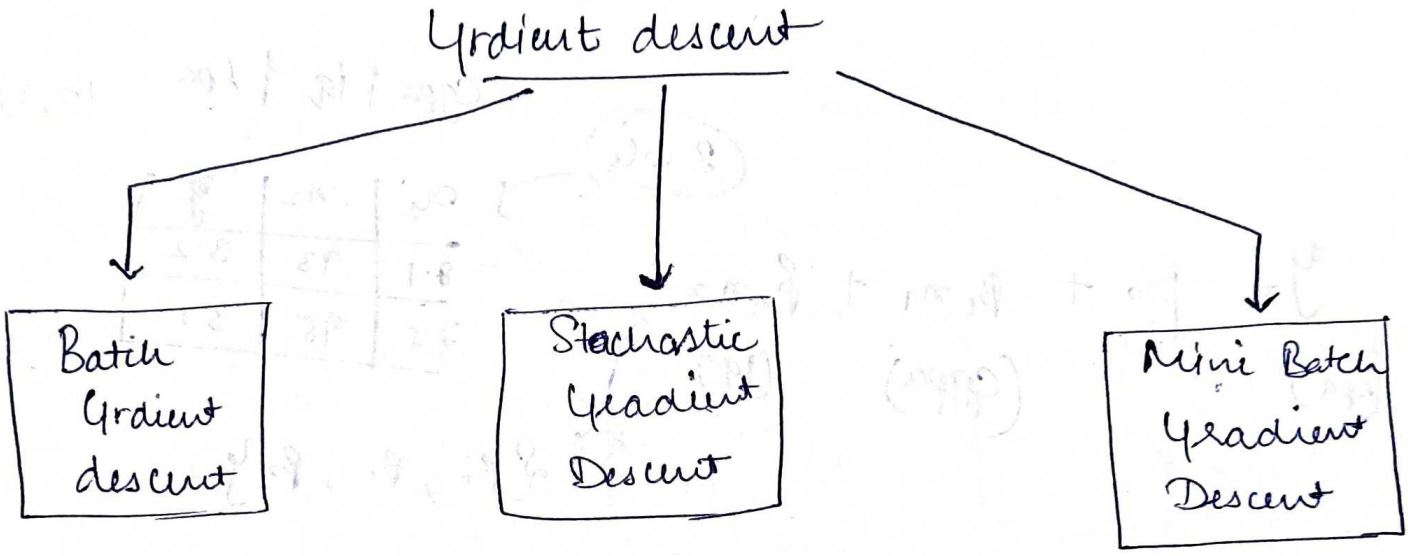
local minima

Global minima



* In this, we can not reach at global minima. we stuck at local minima.

Types of Gradient descent



$m = 1, \quad b = 0$

Iterate

Batch → slow

m, b

$\frac{dL}{dm_0} \rightarrow \frac{\partial L}{\partial m}$

$\rightarrow m_n = m_0 - \eta \times (\text{slope})_{m=0}$

$\frac{dL}{db} \rightarrow \frac{\partial L}{\partial b}$

$b_n = b_0 - \eta \times (\text{slope})_{b=0}$

300

300 Rows ko dekhne ke bad update krte he

Stochastic GD → fast
→ low computation
→ 1 row
→ for larger data

mini batch GD
→ both batch GD and Stochastic

Same Scale → Data

↳ bcz when data is are different scale so, it take much time to reach at Global minima.

Mathematical Formula

n-dimension dataset

cgpa | iq | lpa (2,3)

3 cols

x_1	x_2	y
8.1	93	3.2
7.5	95	3.5

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

(lpa) (cgpa) (iq)

$\{\beta_0, \beta_1, \beta_2\}$

① Step :- Random value

$\beta_0 = 0$, $\beta_1 ; \beta_2 = 1$

② Step epoch = 100, lr = 0.1

$L(\beta_0, \beta_1, \beta_2)$

$\beta_0 = \beta_0 - \eta$ slope → $\frac{\partial L}{\partial \beta_0} = \frac{\partial L}{\partial \beta_1} \frac{\partial L}{\partial \beta_2}$

$\beta_1 = \beta_1 - \eta$ slope

$\beta_2 = \beta_2 - \eta$ slope

if n - dim

(n+1) $\beta_0 - \beta_n$

Mean square error

$$L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

& row = 2, cols = 2 + 1 (y)

input → x, output → y

$$= \frac{1}{2}\left[(y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 \right]$$

$$L = \frac{1}{2}\left[(y_1 - (\beta_0 - \beta_1 x_{11} - \beta_2 x_{12}))^2 + (y_2 - (\beta_0 - \beta_1 x_{21} - \beta_2 x_{22}))^2 \right]$$

(0, (-1), 0, $\hat{y}_1$, 0; $\hat{y}_2$)

x_1	x_2	y
8·1 (x_{11})	93 (x_{12})	3·2
7·5 (x_{21})	95 (x_{22})	3·5

$$\frac{\partial L}{\partial \beta_0} = \frac{1}{2}\left[2(y_1 - \hat{y}_1)(-1) + 2(y_2 - \hat{y}_2)(-1) \right]$$

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$\hat{y}_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12}$$

$$\hat{y}_2 = \beta_0 + \beta_1 x_{12} + \beta_2 x_{22}$$

$$\frac{dL}{\partial \beta_0} = -\frac{2}{2}\left[(y_1 - \hat{y}_1) + (y_2 - \hat{y}_1) \right]$$

$$= \frac{-2}{n}\left[(y_1 - \hat{y}_1) + (y_2 - \hat{y}_2) + (y_3 - \hat{y}_3) \cdots (y_n - \hat{y}_n) \right]$$

$$= \boxed{\frac{-2}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i) = \frac{\partial L}{\partial \beta_0}}$$

$$L = \frac{1}{2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$L = \frac{1}{2} \left[(y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 \right]$$

$$L = \frac{1}{2} \left[(y_1 - \beta_0 - \beta_1 x_{11} - \beta_2 x_{12})^2 + (y_2 - \beta_0 - \beta_1 x_{21} - \beta_2 x_{22})^2 \right]$$

$$\frac{\partial L}{\partial \beta_1} = \frac{1}{2} \left[2(y_1 - \hat{y}_1)(-x_{11}) + 2(y_2 - \hat{y}_2)(-x_{21}) \right]$$

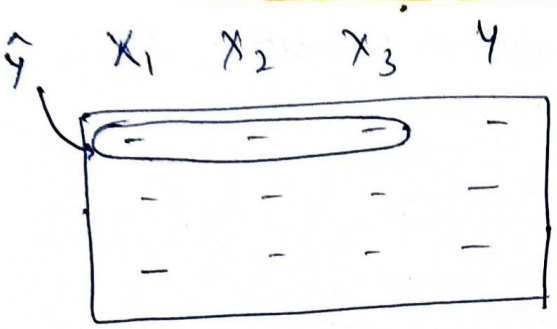
$$\frac{\partial L}{\partial \beta_1} = \frac{-2}{n} \left[(y_1 - \hat{y}_1)(x_{11}) + (y_2 - \hat{y}_2)(x_{21}) + (y_3 - \hat{y}_3)(x_{31}) + \cdots + (y_n - \hat{y}_n)(x_n) \right]$$

$$\frac{\partial L}{\partial \beta_1} = \frac{-2}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i) x_{i1}$$

$x_{i1} \rightarrow$ x_{11}, x_{21}, x_{31}, x_{41}, ⋮, x_{n1}

$$\frac{\partial L}{\partial \beta_2} = \frac{-2}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i) x_{i2}$$

$$\frac{\partial L}{\partial \beta_m} = \frac{-2}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i) x_{im}$$



$$\hat{y}_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \beta_3 x_{13}$$

$$\hat{y}_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \beta_3 x_{23}$$

β_0 + np.dot (X_train, coef_)

(353, 10) (10, 1)

$$\hat{y} = \beta_0 + \begin{bmatrix} x_{11} & x_{12} & x_{13} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$$

(353, 1) → $\hat{y}$ = np.dot (coef_, X_train) + β_0

↑ y_pred

How to find coef_

x_1	x_2	y	$\hat{y}$
1	2	5	6
3	4	7	8

y $\quad$ $\hat{y}$

[5 7] [6 8]

$y - \hat{y} = [-1 \quad -1]$

$$\frac{\partial L}{\partial \beta_1} = \frac{-2}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)\, x_{i1}$$

$$\frac{-2}{n} \left[[-1 \quad -1] \begin{bmatrix} 1 \\ 3 \end{bmatrix} \right]$$

$[-4 + -4]$

$= 8$

$$\frac{\partial L}{\partial \beta_2} = [y - \hat{y}] \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

$$= [-1 \quad -1] \begin{bmatrix} 2 \\ 4 \end{bmatrix} \times \frac{-2}{4}$$

if you want this overall ~~condition~~ or find all derivatives.

$$\left[[y - \hat{y}] \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} x - \frac{2}{4} \right]$$

$$\frac{dL}{\partial \beta_1} \cdots \frac{\partial L}{\partial \beta_{10}} = \left[(y_i - \hat{y}_i) \, x\text{-train} \right] x \, -\frac{2}{n}$$

y-train → transpose

$y_i = 3 \times 3$

(3 × 3, 1) (3 × 3, 10) (3 × 3, 1)

(1, 353) (353, 10) (1, 1)

$(1, 10) \times \frac{-2}{4} = (1, 10)$

↑ Coef-der

Disadvantage

1:- Very slow

2:- Large calculation

3:- Hardware (less RAM and Large Data)

Stochastic Gradient Descent

* Sklearn

Stochastic

- single row → update
- second row → update
- Third row → update

* Faster convergence
  - less no. of epochs

n rows → n update

1 epochs

↳ random select row

↳ steady 50/100 %

↳ correction

Time Comparison

no. of epochs → fixed

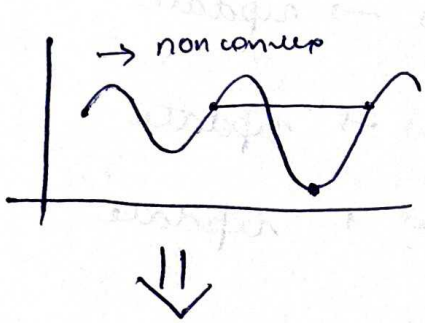
e = 100

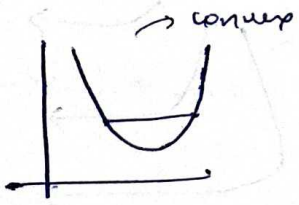
- batch → 100
- Stochastic → 100 x n → no. of rows
  - no need that much epochs.
  - ↳ decrease the rate

When to use Stochastic GD

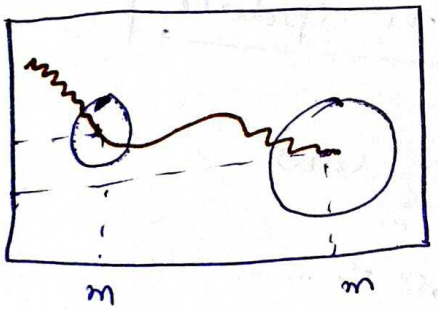
1) Big Data → SGD

2) Non convex function





3



Problem :- still ^ Fluctuate when near the destination

solution :- learning schedule → learning rate vary

```
t0, t1 = 5, 50

def learning_rate(t):
    return t0/(t+t1)

for i in range(epochs):
    for in range(X.shape[0]):
        ls = learning_rate(i * X.shape[0]+j)
```

Mini- Batch Gradient Descent

Batch

1 update / epoch

{ slow }

{ small dataset, convex data }

Stochastic (n rows)

n update / epoch

{ fast, big data, non- convex }

↳ Random

Mini batch

↳ group of rows

let n = 1000

↳ batches → 100 → 10 batches

batches update / epoch

let n = 1000

n = 100 → 10 batches of 1000 rows ⇒ 10 batches update / epoch