b] Gradient descent

*) Gradient descent is an optimization algorithm used to minimize a function

*) In machine learning, the function is usually a loss (or cost) function that measures how far off the model's predictions are from the actual data

$J(w, b) \longrightarrow$ want $\min\limits_{w,b} (J(w,b))$

*] start with some w, b (set $w = 0$, $b = 0$)
*) keep changing w, b to reduce $J(w, b)$
*) Until we settle at or near a minimum

◎ If it has more than one minimum it is not linear regression (not squared error cost function)

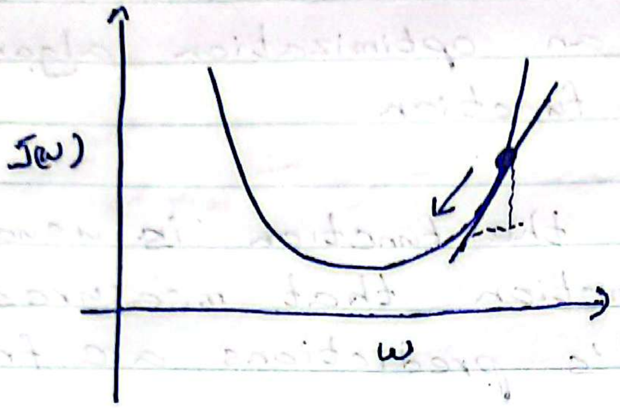
$$w = w - \alpha \boxed{\frac{\partial}{\partial w} J(w, b)}$$

$$b = b - \alpha \boxed{\frac{\partial}{\partial b} J(w, b)}$$

α = learning rate
partial derivative

*) Simultaneously update w and b until convergence.

Simple case $\rightarrow J(w)$



$$w = w - \alpha \boxed{\frac{\partial}{\partial w} J(w)}$$

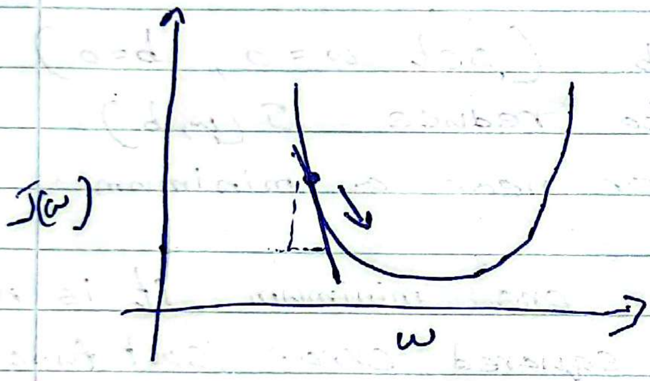
> 0

$w = w - \alpha \cdot$ (positive number)

w ~~Derivative~~ decrease

positive slope

positive derivative



$$w = w - \alpha \boxed{\frac{\partial}{\partial} Jw}$$

< 0

$w = w - \alpha$ (negative number)

w = increases

negative slope

negative derivative

7] Learning rate (α)

i] If α is too large :

*) Gradient descent overshoot, never reach minimum

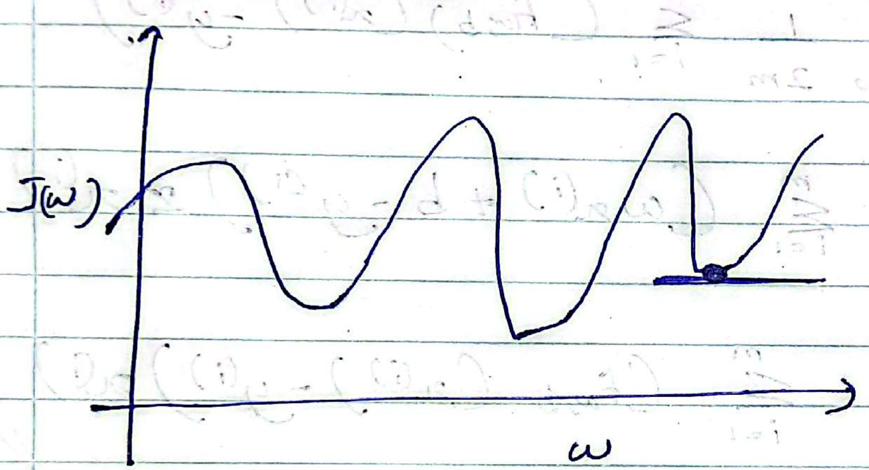
*) Fail to converge, diverge

ii) If α is too small:

*) Gradient descent may be slow

Local minimum

*) No matter what learning rate you use it will change nothing



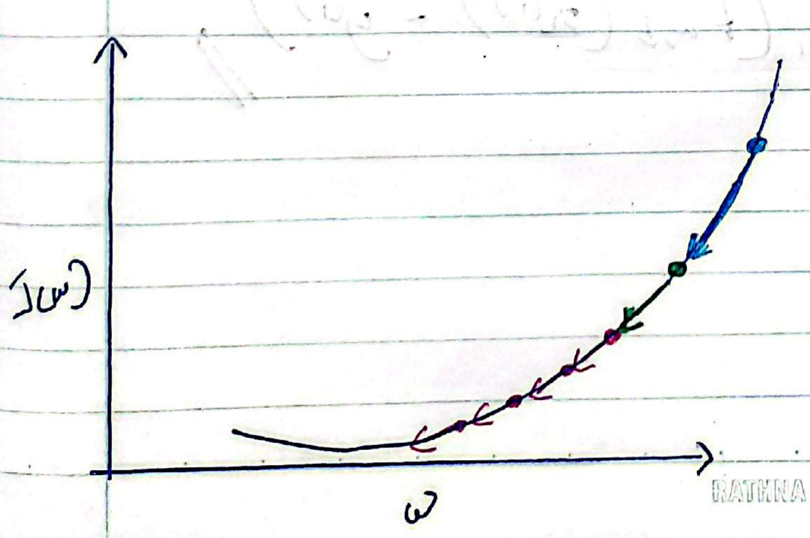
Slope = 0

(local minimum)

$$w = w - \alpha \boxed{\frac{d}{dw}(J(w))} = 0$$

$$\boxed{w = w}$$

Fixed learning rate



$$w = w - \boxed{\alpha \frac{d}{dw}(J(w))}$$

smaller (fixed)

not as large

large

Near a local minimum:

*) Derivative is smaller

*) Update steps become smaller

8] Gradient descent for linear regression

$$f_{w,b}(x) = wx + b$$

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2$$

*] $$\frac{\partial}{\partial w} J(w,b) = \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2$$

$$= \frac{1}{\cancel{2}m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)}) \cancel{2} x^{(i)}$$

$$= \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$$

*] $$\frac{\partial}{\partial b} J(w,b) = \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)})^2$$

$$= \frac{1}{\cancel{2}m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)}) \cancel{2}$$

$$= \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})$$

Gradient descent algorithm

repeat until convergence {

$$w = w - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}$$

*) For squared error cost it should only have one global minimum (= Convex function / bowl)

Batch gradient descent

*) Batch = Each step of gradient descent uses all the training examples

*) There are some methods other than batch gradient descent which use only a subset of training examples.