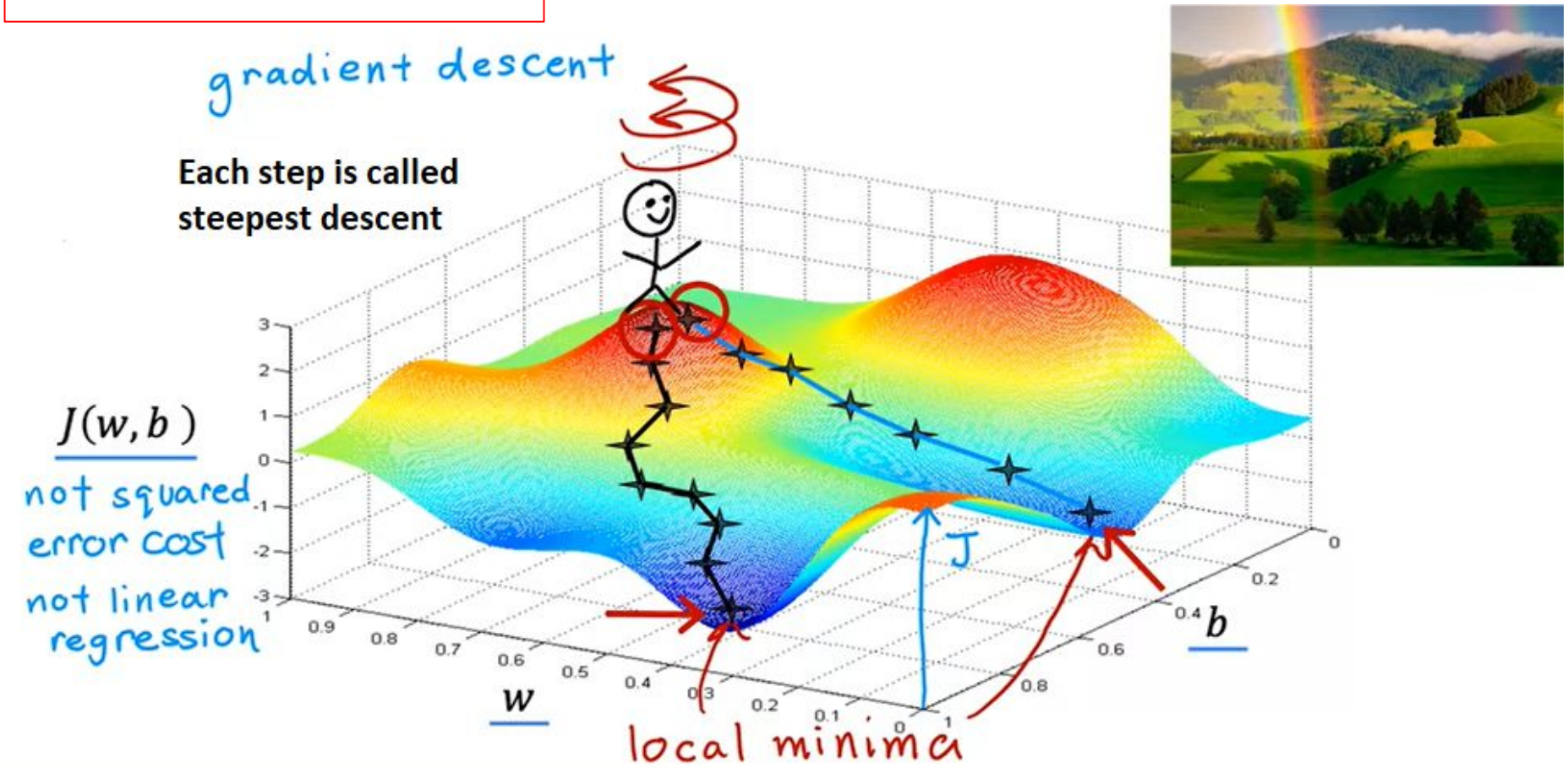
Gradient descent



Gradient descent algorithm

Repeat until convergence

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

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

Learning rate

Derivative

Simultaneously update w and b

Assignment

$a = c$

$a = a + 1$

Code

Truth assertion

$a = c$

$a = a + 1$

Math

a==c

Correct: Simultaneous update

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

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

$$w = tmp\_w$$

$$b = tmp\_b$$

Incorrect

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

$$w = tmp\_w$$

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

$$b = tmp\_b$$

Notes:

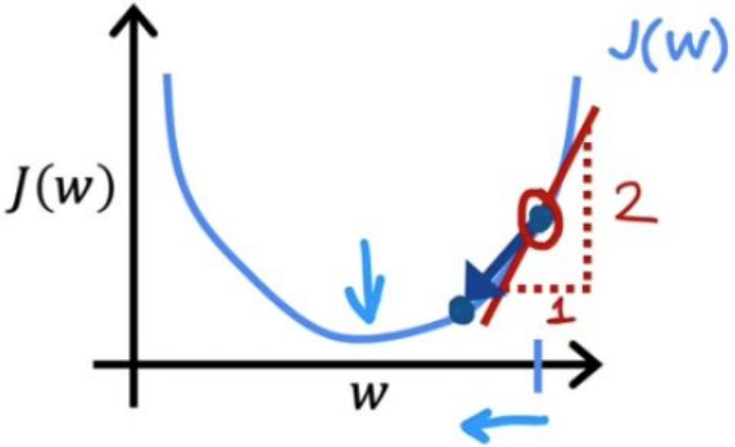
- Alpha: is a small +ve number between 0 and 1. It controls the size of each step (steepest descent).

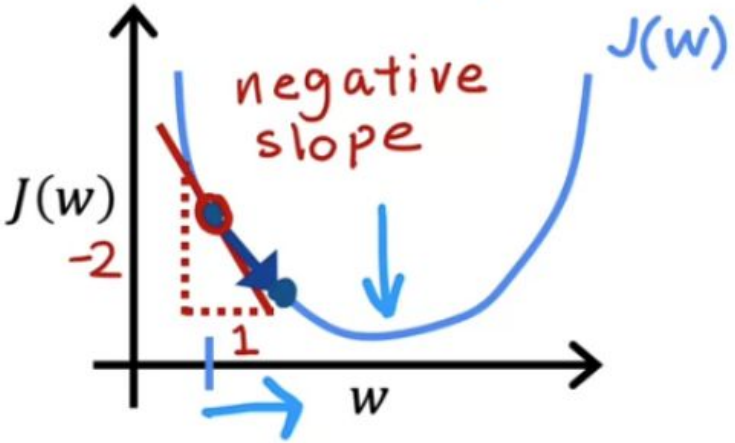
- Derivative term: telling us in which direction to take the step. The slope of tangent of the point

---

These two updates executes until the algorithm converges(which mean reaching the point at a local minimum[where the cost function for each steps is almost equal to the previous one]).

Derivative term





$$w = w - \alpha \underbrace{\frac{d}{dw} J(w)}_{>0}$$

$$w = w - \alpha \cdot (\text{positive number})$$

$$\underbrace{\frac{d}{dw} J(w)}_{<0}$$

$$w = w - \alpha \cdot (\text{negative number})$$

Learning Rate

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

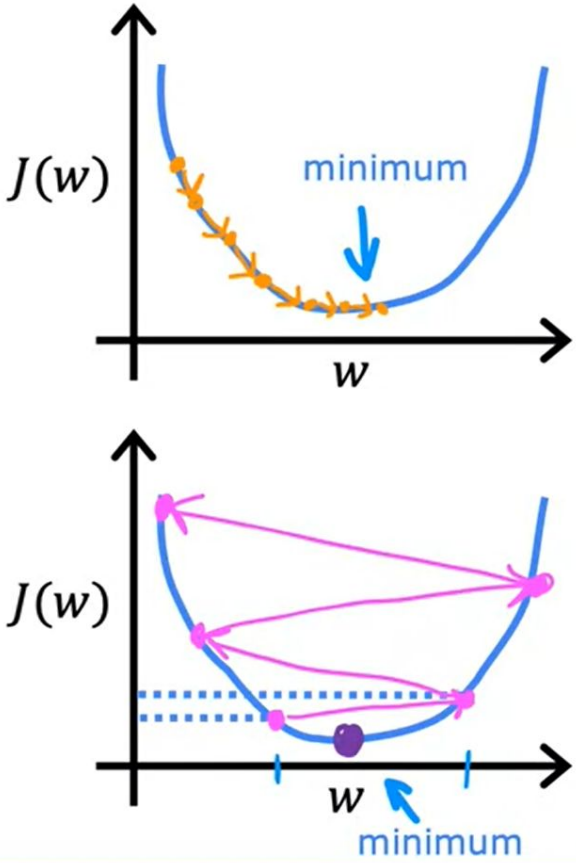
If α is too small...

Gradient descent may be slow.

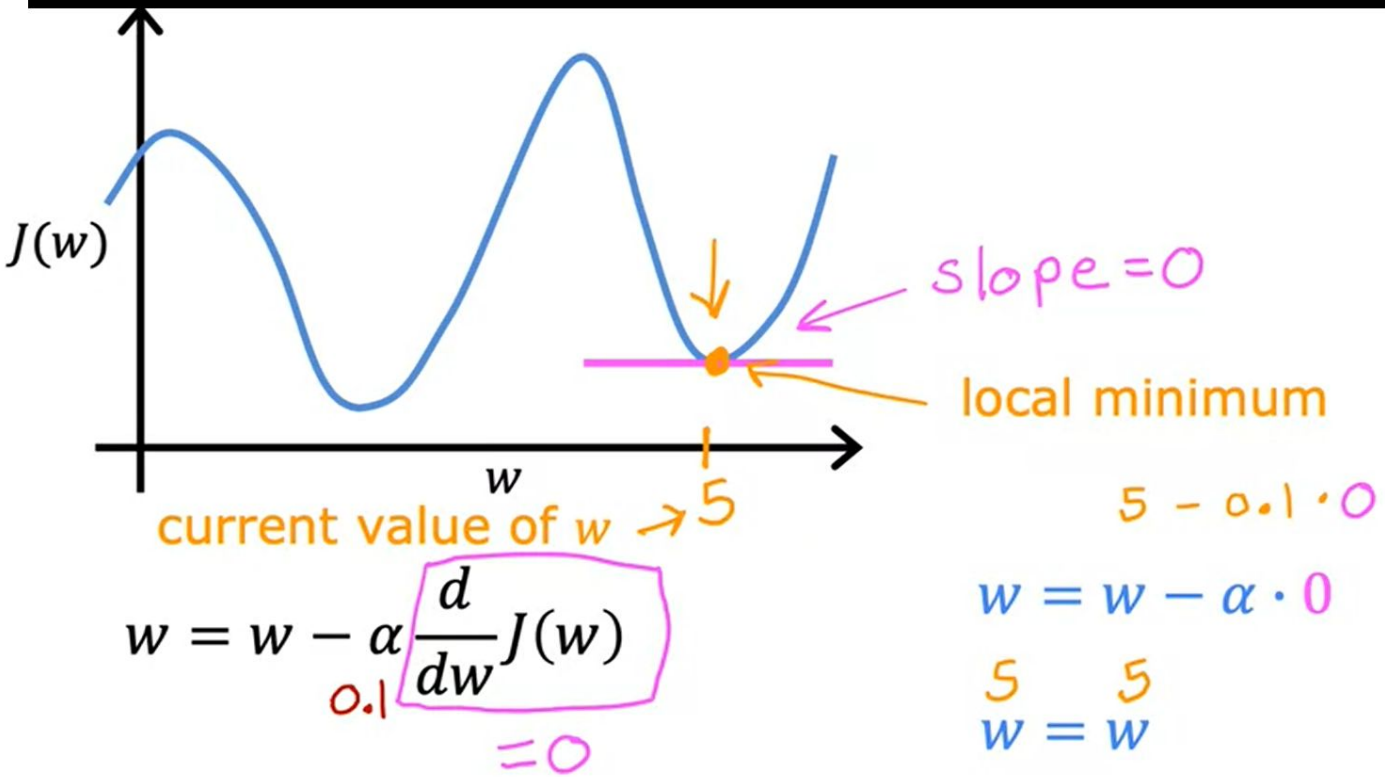
If α is too large...

Gradient descent may:

- Overshoot, never reach minimum
- Fail to converge, diverge



If we're already at a local minimum, gradient descent leaves (w) unchanged.



This explains how gradient descent reach the local minimum while alpha is constant.

Can reach local minimum with fixed learning rate

α

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

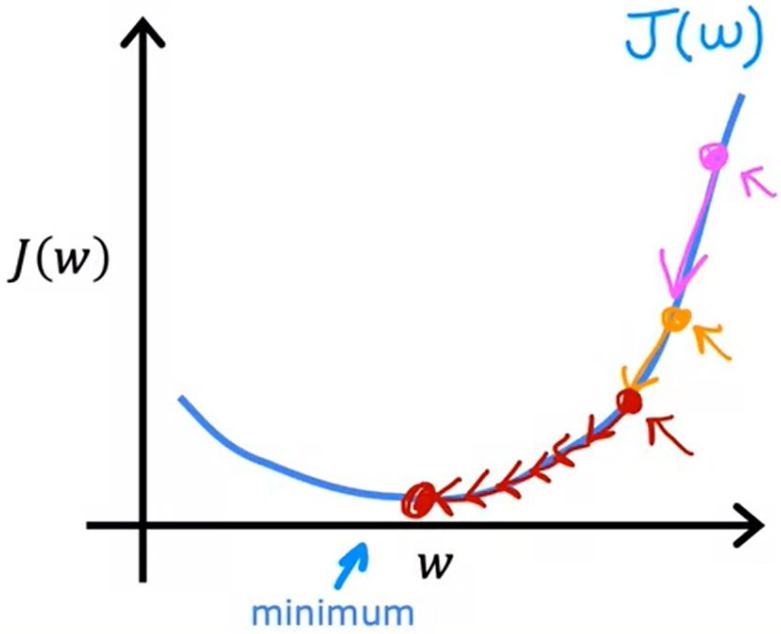
smaller

not as large

large

Near a local minimum,
- Derivative becomes smaller
- Update steps become smaller

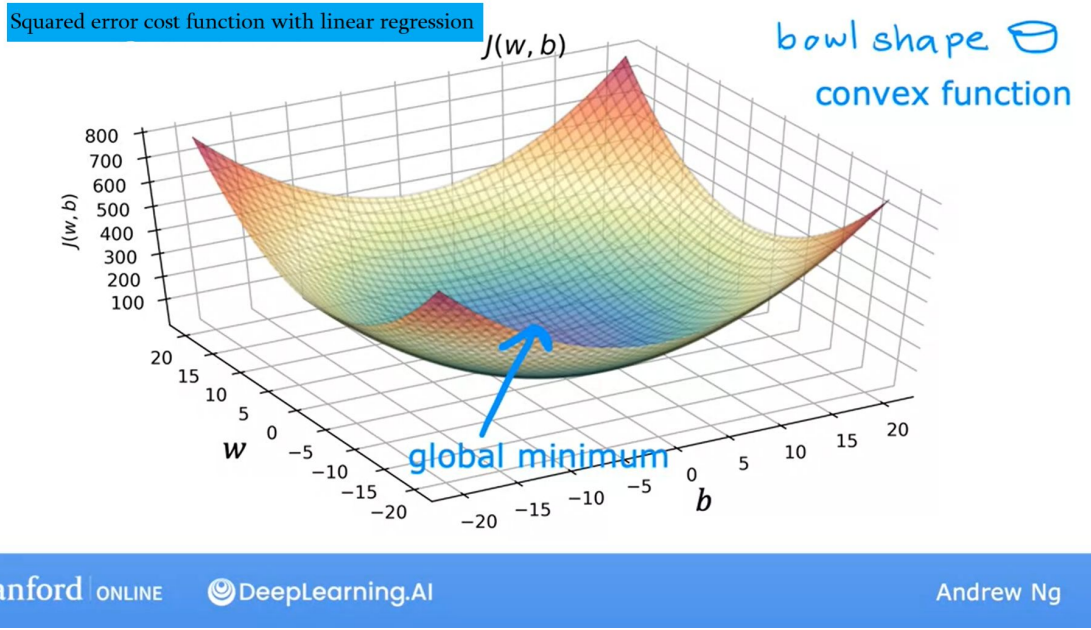
Can reach minimum without decreasing learning rate α



Global Minimum → The point that has the lowest possible value for the cost function of all possible points(the lowest local minima).

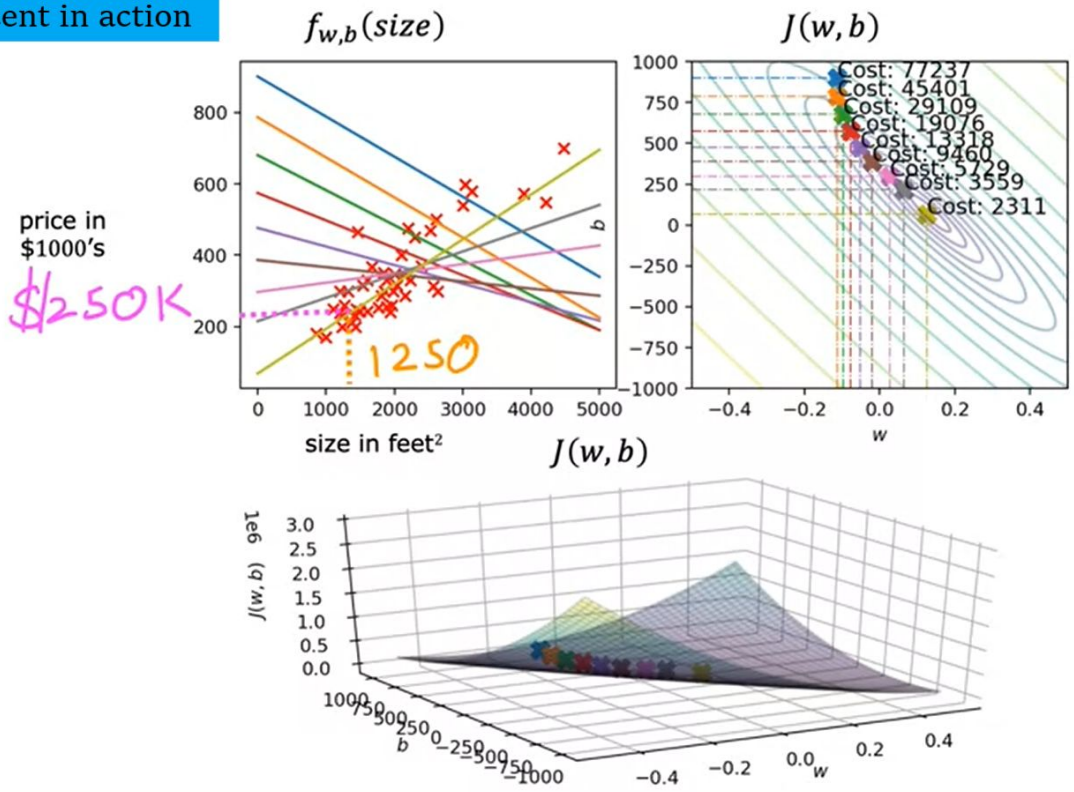
Generally, the local minima differs depending on where we initialized the parameters w & b. But, in linear regression, the cost function is (Convex function).

This means that the squared error cost function with linear regression does not has multiple local minima. This provides a great property which is getting the same local minima whatever was the value of learning rate(Alpha).



Gradient descent in action

Gradient descent in action



Batch gradient descent

"Batch" gradient descent

other gradient descent: subsets

"Batch": Each step of gradient descent uses all the training examples.

	x size in feet²	y price in $1000's
(1)	2104	400
(2)	1416	232
(3)	1534	315
(4)	852	178
...	...	...
(47)	3210	870

$m = 47$

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