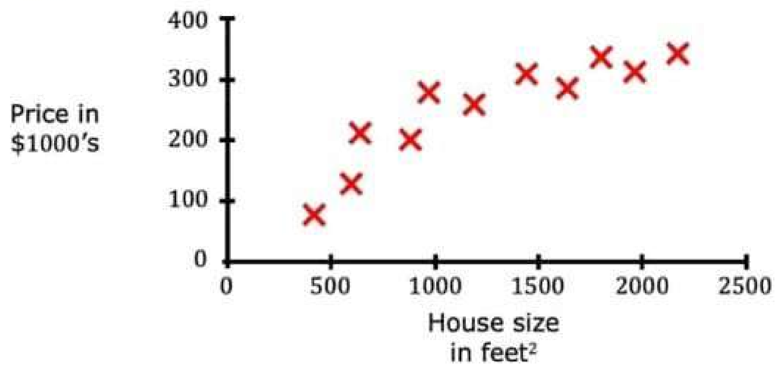
Supervised Learning

Linear Regression

Supervised Machine Learning

We give some examples with both inputs and outputs specified clearly. The machine uses these examples to learn and give an output label on the basis of the input provided by the user.

Regression: Housing price prediction



Some basic terminologies

(i) Dataset - A huge collection of data. Dataset can contain any type of information depending on the problem.

(ii) Datapoint - Any data entry or "point of information" in the dataset, can be referred to as a data point

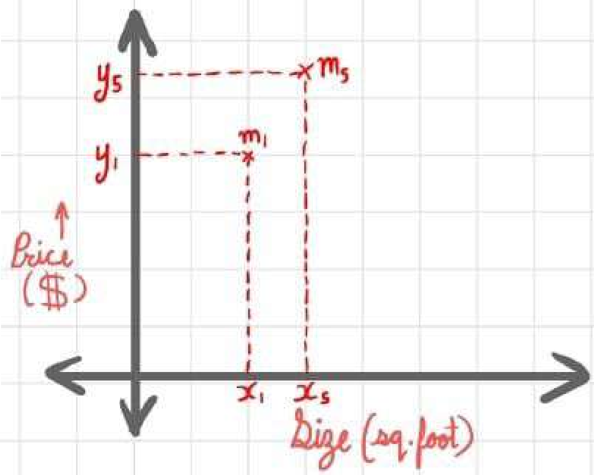
(iii) Input Label - Typically, what one may call "x" in context of coordinate - geometry

(iv) Output Label / Output target - Typically, what one may call "y" in context of coordinate - geometry

These terminologies are just the basic of the basics. As the course proceeds we will be facing more challenging problems and terminologies. Stay focussed and consistent and we'll pass through easily.

What does a typical dataset look like?

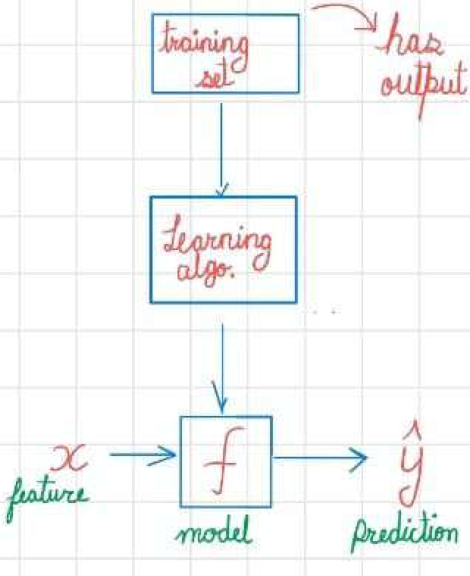
	Living area ($feet^2$)	Price (1000\$s)
→ m=	2104 (x_1)	400 (y_1)
→ m	1600 (x_2)	330 (y_2)
→ m=	2400 (x_3)	369 (y_3)
→ m	1416 (x_4)	232 (y_4)
→ m=	3000 (x_5)	540 (y_5)
	⋮ (x_i)	⋮ (y_i)



where m = index of the data set
x_i = input label
y_i = output label

→ the value of i depends on the index's progression

A Brief Working Structure :



has input features (x) and output targets (y) from dataset.

Linear Regression

Linear Regression is a simplified method of using Supervised Learning using basic theory of 2-D geometry and algebra to your advantage.

To be a bit more technical, linear regression is a way by which we predict values using pre-existing relationships b/w the dependent (y) and the independent variable (x).

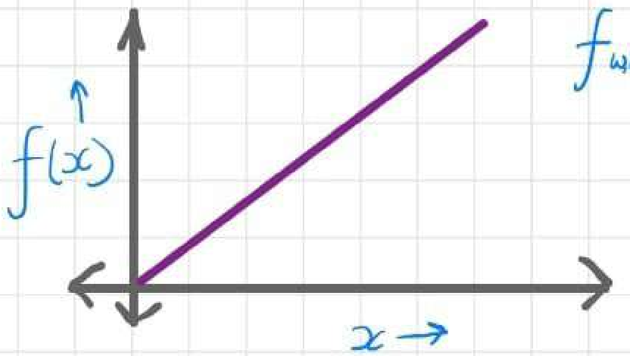
Model used in Linear Regression

To be exact, the model we refer to here is truly represented by the equation, $\boxed{f_{\omega,b}(x) = \omega x + b}$ or $\boxed{f(x) = \omega x + b}$, where ω = weight & b = bias

weight and bias (ω & b) are parameters that can be adjusted to make our model more accurate.

Notice anything familiar?

To simplify the figure representation, we assume $b=0$



$f_{\omega b}(x) = \omega x + \underbrace{b}_{\to 0}$

ex-1 $f(x) = wx + b$

x_train = [1, 2]
y_train = [300, 500]

Cost function

Cost function gives the margin of error for the model

$f_{\omega,b}(x_i) = \omega x_i + b = \hat{y}_i$ (predicted value)

Let y_i be the target value.

Cost fn.:- Squared error cost function

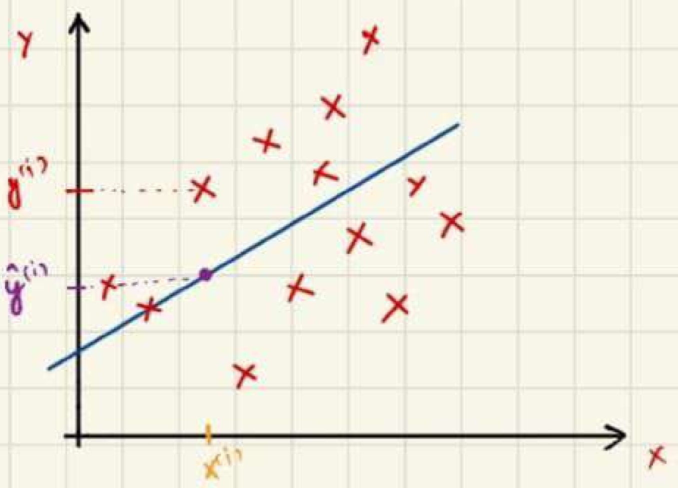
$J_{\omega,b} = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$ where m = number of training examples

* Goal: To minimise $J_{\omega,b}$

$f_{w,b}(x^{(i)})$

$$\boxed{J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2}$$

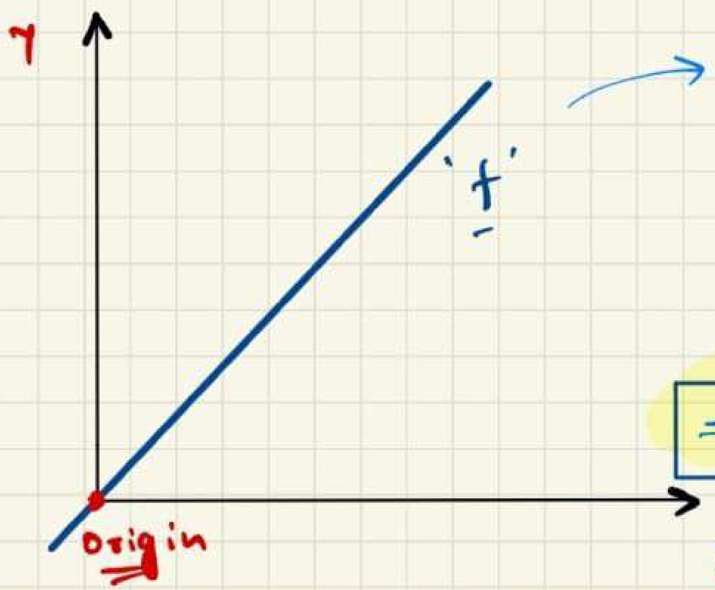
"m → no. of training examples."



$$\boxed{\hat{y}^{(i)} = f_{w,b}(x^{(i)})}$$

$$\boxed{f_{w,b}(x^{(i)}) = w x^{(i)} + b}$$

Assume that, for $f(x) = wx + b$; $b = 0$



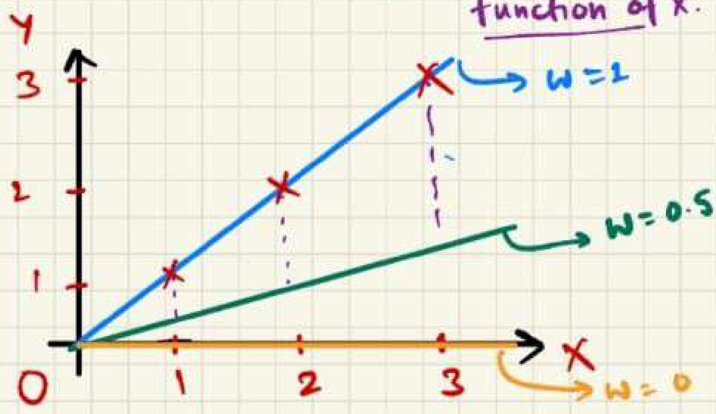
$f_w(x) = wx + \cancel{b} \to 0$

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

↳ $w x^{(i)}$

$\boxed{\Rightarrow \min_{w} J(w)}$

$f_w(x)$ ↳ for a fixed w, function of x.

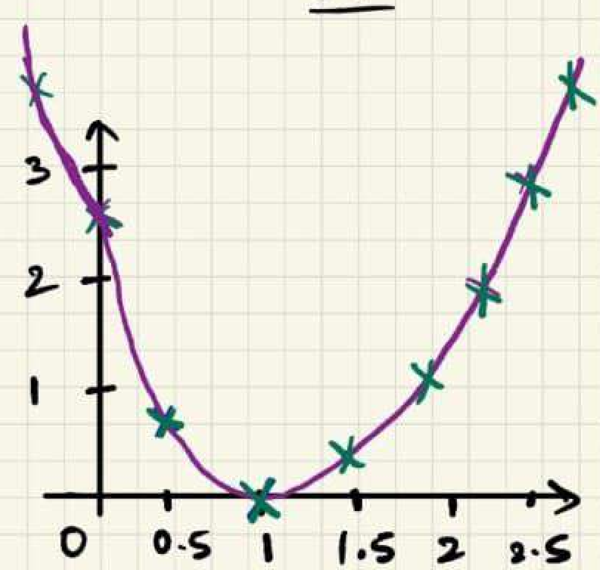


$\boxed{J(w) = \frac{1}{2m}\sum_{i=1}^{m} f_w(x^{(i)} - y^{(i)})^2}$

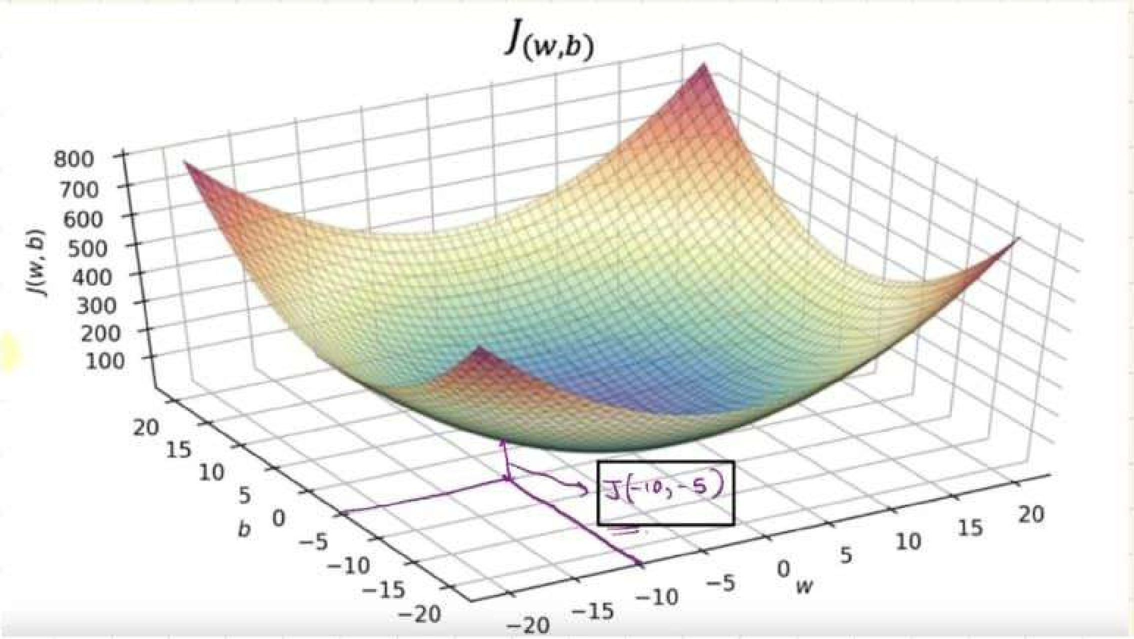
Q. Find the value of the cost funⁿ.
↳ assume $w = 1$

⇒

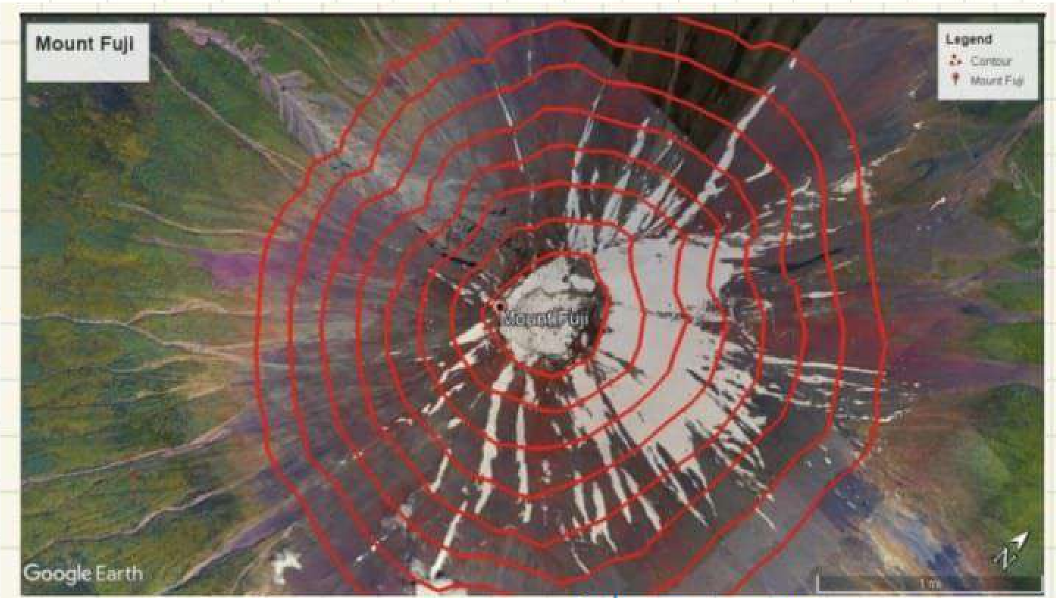
$J(w)$



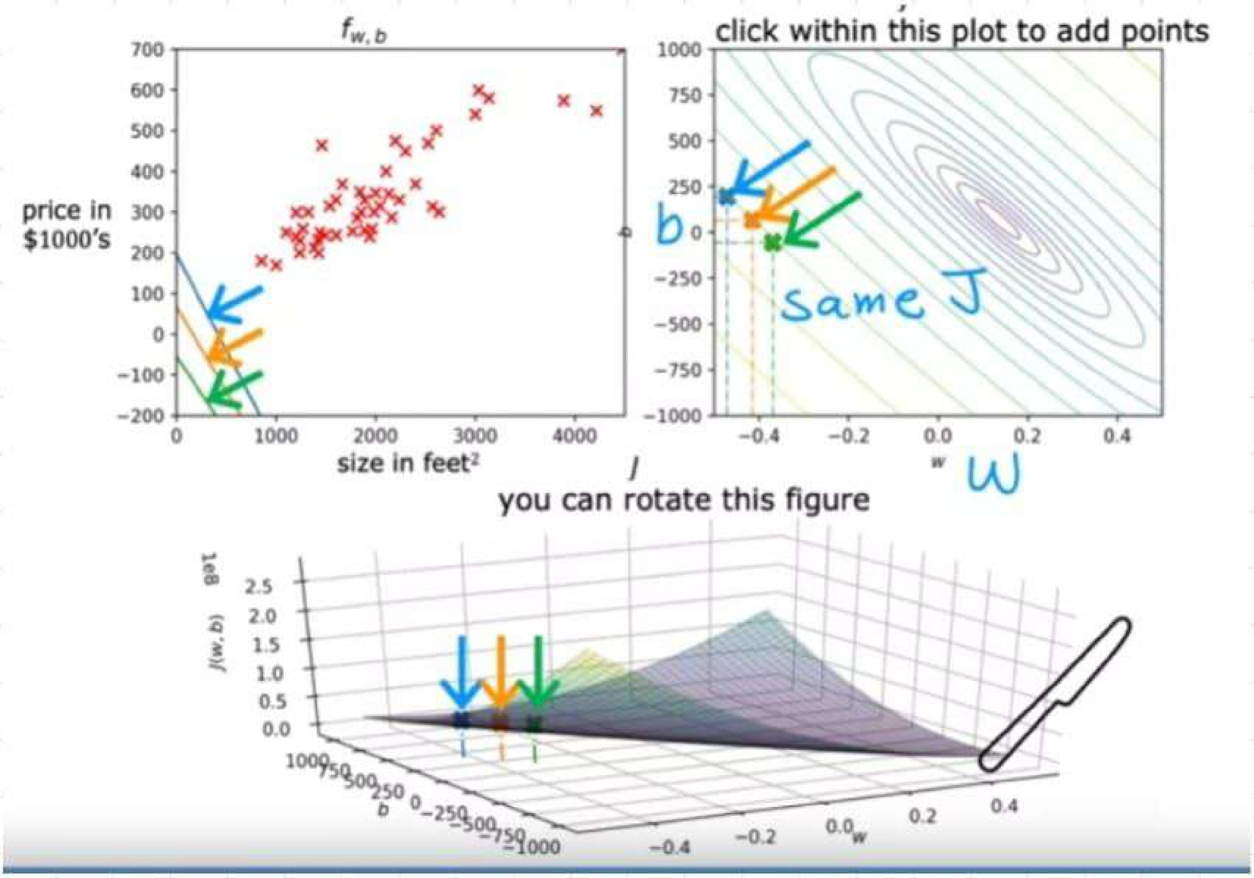
What does cost function look like? (3D intuition)



Contour plot: A plot which shows all the points at the same height for different heights.



eg:- Contour plot of Mt. Fuji (top view)



Gradient descent algorithm

Used to train model by finding optimal value of cost function.
Not only this, a gradient function can be used to minimise any fn.

$$\boxed{\omega = \omega - \alpha \frac{\partial J_{\omega,b}}{\partial \omega}}$$

where α = learning rate

Learning rate denotes the rate at which the weight (ω) descends to an optimal value for cost function's utilisation.

$$\boxed{b = b - \alpha \frac{\partial J_{\omega,b}}{\partial b}}$$

*imp the value of α is always b/w 0 & 1.

* The values of ω & b, must be calculated and updated simultaneously

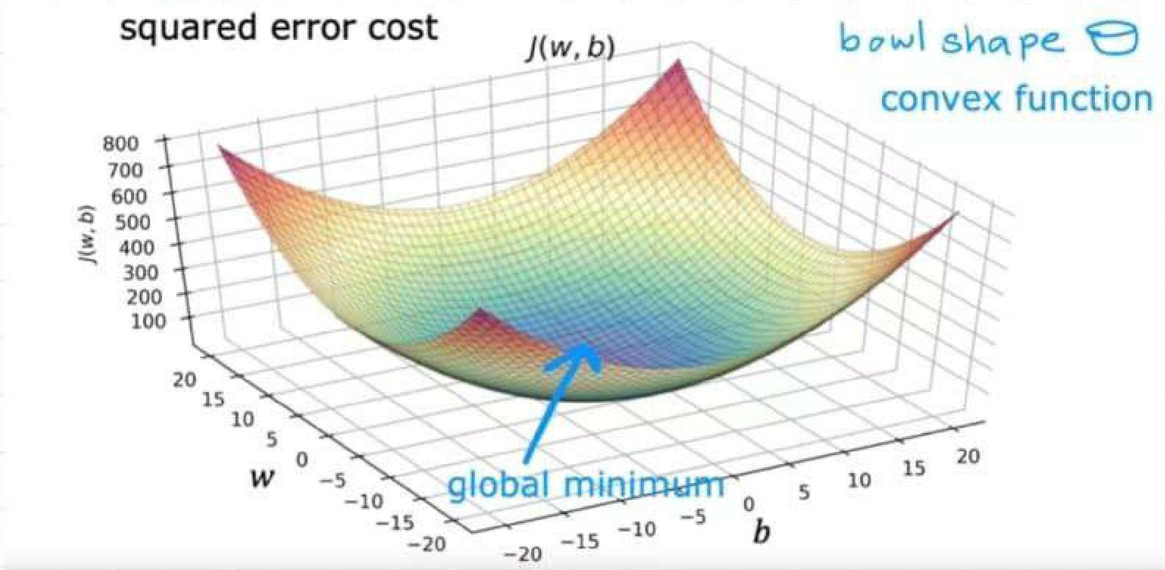
⇒ Correct method for simultaneous calculations

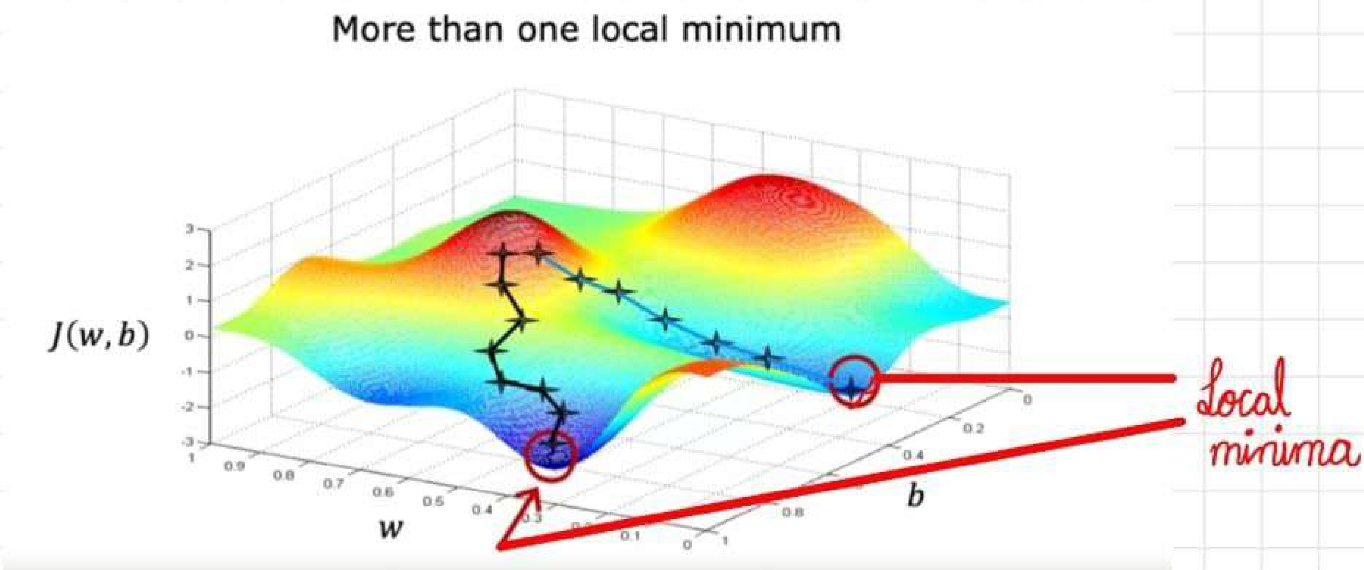
$$tmp\_\omega = \omega - \alpha \frac{\partial J_{\omega,b}}{\partial \omega}$$

$$tmp\_b = b - \alpha \frac{\partial J_{\omega,b}}{\partial b}$$

$$\omega = tmp\_\omega$$
$$b = tmp\_b$$

* Goal: To repeat the alogorithms till the values of the parameters converge, i.e., the values stop changing much.





Learning rate(α)

In the gradient descent algorithm, the learning rate should neither be too high, nor too low.

* v.imp How do we represent gradient descent algorithm in a code-friendly format?"

$$\Rightarrow \ w = w - \alpha \frac{\partial J_{w,b}}{\partial w} \rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x_i) - y_i)x_i$$

$$b = b - \alpha \frac{\partial J_{w,b}}{\partial b} \rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x_i) - y_i)$$

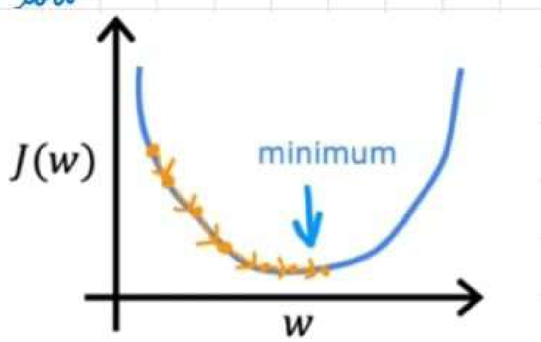
$\therefore$, the new eqns. are $\Rightarrow$ $\boxed{w = w - \alpha \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x_i) - y_i)x_i}$

$$\boxed{b = b - \alpha \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x_i) - y_i)}$$

What is an ideal value of Learning rate (α)?

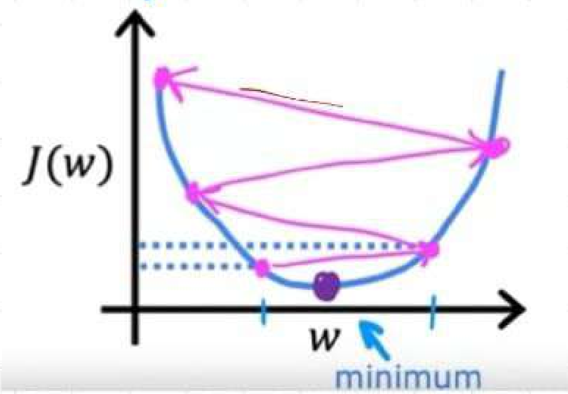
Learning rate (α), can neither be too low, nor too high, but why though?

Case-1 α too low



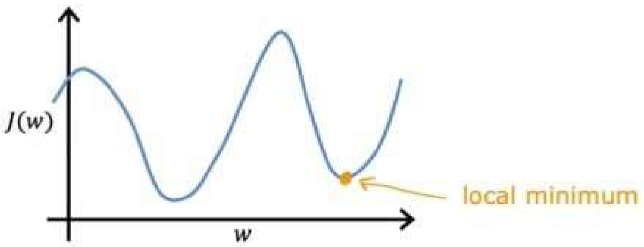
In such a case, not only the descent is slow, but also the resource consumption is very high.

Case-II α too high



If α is too high, then rather than approaching minimum value of cost function faster, the values bounce off further away

9



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