

## Week 2: Regression with multiple input variables.

⇒ Multiple linear Regression

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

### Video 1: Multiple features

Earlier we had seen only one input feature (size) corresponding to output  $\hat{y}$ , (price)

Now, suppose there are "multiple features example":

features →	size in feet <sup>2</sup>	no. of bedrooms	no. of floors	age of home	price
features	$x_1$	$x_2$	$x_3$	$x_4$	$y$
	204	1.5	1	45	460
	1416	3	2	40	283
	1534	3	2	30	365
	852	2	1	20	178

$x_1, \dots, x_4$  - features

→  $n$  = no. of features = 4

$x_{ij}$  =  $j^{\text{th}}$  feature

$x_{ij}^i$  → features of  $i^{\text{th}}$  training example

$x_3^2$  = no. of floors in second training example

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

-  $(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n) - y^{(i)}$

Previously, algorithm was implemented : Matrix

$$f(w, b)(x) = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b$$

vectorised version algorithm  $\Leftarrow$

$$f_{w,b}(x) = w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + b$$

vectorized with W :  $\Leftarrow$  adding

example:

$$f(w, b)(x) = 0.1x_1 + 0.4x_2 + 1.0x_3 + 2x_4 + 80$$

$\uparrow$

base price

$$f_{w,b}(x) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b$$

given months  $\rightarrow$  [2000, 10, 0.01, 2000, 10, 0.01, 10, 0.01, 2000, 10, 0.01]

$$\vec{w} = [w_0, w_1, w_2, w_3, \dots, w_n]$$

$b$  = number

$$\vec{x} = [x_0, x_1, x_2, \dots, x_n]$$

now

$$f_{\vec{w}, b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

wanted  $\rightarrow x_0, x_1, \dots, x_n$

$\Rightarrow$  general for all  $i \in n$

general  $\rightarrow$   $x_i$

Video 2: Vectorisation - Part (I)  $\rightarrow$  go through  $\rightarrow$   $x_i$

Non Vectorised Implementation :

$$f_{\vec{w}, b}(\vec{x}) = \left( \sum_{j=1}^n w_j x_j \right) + b$$

for  $j$  in range( $n$ ):

$$f = f + w[j] * x[j]$$

$$f = f + b$$

Vectorisation

introduction to ML

$$f_{\vec{w}, b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

$$f = np.dot(w, x) + b$$

### Video 3: Vectorisation - Part (II)

$np.dot(w, x)$  without loop instead of  $w[0]x[0] + w[1]x[1] + \dots + w[15]x[15]$

$$\begin{bmatrix} w[0] & w[1] & \dots & \dots & \dots & w[15] \end{bmatrix} \begin{bmatrix} * & * & \dots & \dots & \dots & * \end{bmatrix}$$

$$\begin{bmatrix} x[0] & x[1] & \dots & \dots & \dots & x[15] \end{bmatrix} \begin{bmatrix} w[0] & w[1] & \dots & \dots & \dots & w[15] \end{bmatrix}$$

$$[w[0]*x[0]] + [w[1]*x[1]] + \dots + [w[15]*x[15]]$$

Gradient Descent

$$\vec{w} = (w_1, w_2, \dots, w_{16})$$

$$\vec{d} = (d_1, d_2, \dots, d_{16})$$

$$w = np.array([0.5, 1.3, -2.5, 3.4])$$

$$d = np.array([0.3, 0.2, -0.6, 0.4])$$

$$\text{compute: } w_j = w_j - 0.1 d_j$$

$\uparrow$   
learning rate

Without vectorisation

$$w_1 = w_1 - 0.1d_1,$$

$$w_2 = w_2 - 0.1d_2,$$

.

.

$$w_{16} = w_{16} - 0.1d_{16}$$

$$\vec{d} + \vec{x} \cdot \vec{w} \rightarrow (\vec{w} - 0.1) \vec{d}$$

$$\vec{d} + (\vec{x}, w) \text{ total grad.} = \vec{0}$$

With vectorisation

$$\vec{d} + \vec{x} \cdot \vec{w} \rightarrow (\vec{w} - 0.1) \vec{d}$$

$$\vec{d} + (\vec{x}, w) \text{ total grad.} = \vec{0}$$

(II) fast - vectorisation & optim.

Video 4: Gradient Descent for multiple linear regression

Now,

$$\vec{w} = [w_1 \dots w_n]$$

$\vec{d}$  - still a number

$$f_{\vec{w}, b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

$[0]w | [0]w$

$[0]x | [0]x$

$$+ [0]w * [0]w$$

$$J(\vec{w}, b)$$

Gradient Descent:

repeat {

$$\begin{aligned} (\partial_w J(\vec{w}, b)) &= \vec{v} \\ (\partial_b J(\vec{w}, b)) &= \vec{b} \end{aligned}$$

$$w_j = w_j - \alpha \frac{\partial J(\vec{w}, b)}{\partial w_j} \quad \text{where } \alpha = 0.001, 0.01, 0.0001 \quad \text{for } j = 1, 2, \dots, n$$

$$b = b - \alpha \frac{\partial J(\vec{w}, b)}{\partial b} \quad \text{where } \alpha = 0.001, 0.01, 0.0001$$

for n features

repeat q

until w is fixed (friction)

$$j=1 \quad w_j = w_j - \alpha \frac{1}{m} \sum_{i=1}^m f(\tilde{w}, b)(\tilde{x}_i^{(i)}) - y_i^{(i)} \quad \text{for each } j \in \{1, 2, \dots, n\}$$

$$j=n \quad w_n = w_n - \alpha \frac{1}{m} \sum_{i=1}^m f(\tilde{w}, b)(\tilde{x}_i^{(i)}) - y_i^{(i)} \quad \text{for each } j \in \{1, 2, \dots, n\}$$

Same for b

Alternative for gradient descent

(complicated)

Normal equation:

- Only for linear regression
- Solve for w, b without iteration

Disadvantages

slow in step  
below

- Slow when number of features ( $> 10,000$ )
- doesn't generalise to other learning algorithms

(II) ~~friction - parallel vector~~ : ~~is subiv~~

$$2x_1 + 2x_2 = 0$$

$$2000 > 10 > 2000$$

$$\frac{x_1}{2} + \text{feature } x_2 = 0$$

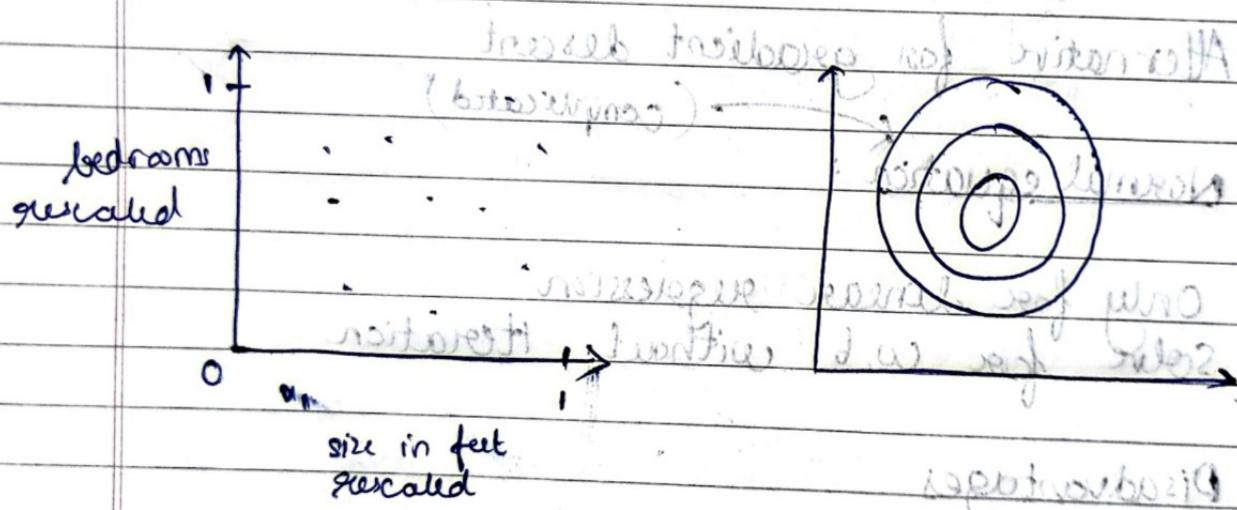
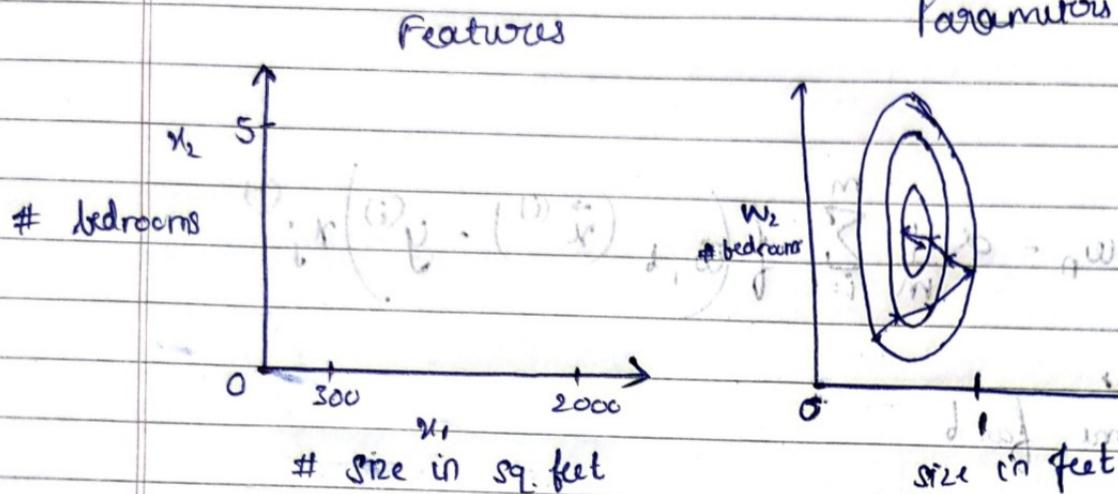
$$\frac{10}{2000} = \text{feature } 10$$

$$12 \cdot \text{feature } x_2 > 0$$

$$12 \cdot \text{feature } 10 > 21.7$$

## → Gradient Descent in Practice

### Video 1: Feature Scaling Part (I)



### Video 2: Feature Scaling - Part (II)

$$300 \leq x_1 \leq 2000$$

$$0 \leq x_1 \leq 1$$

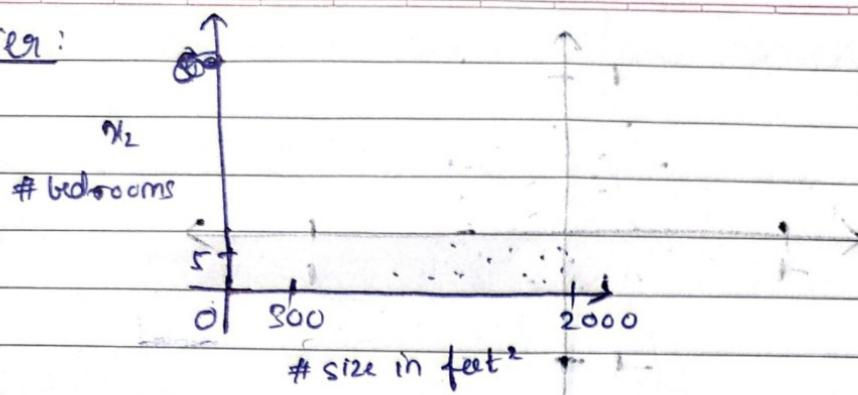
$$x_{1, \text{scaled}} = \frac{x_1}{2000}$$

$$x_{2, \text{scaled}} = \frac{x_2}{5}$$

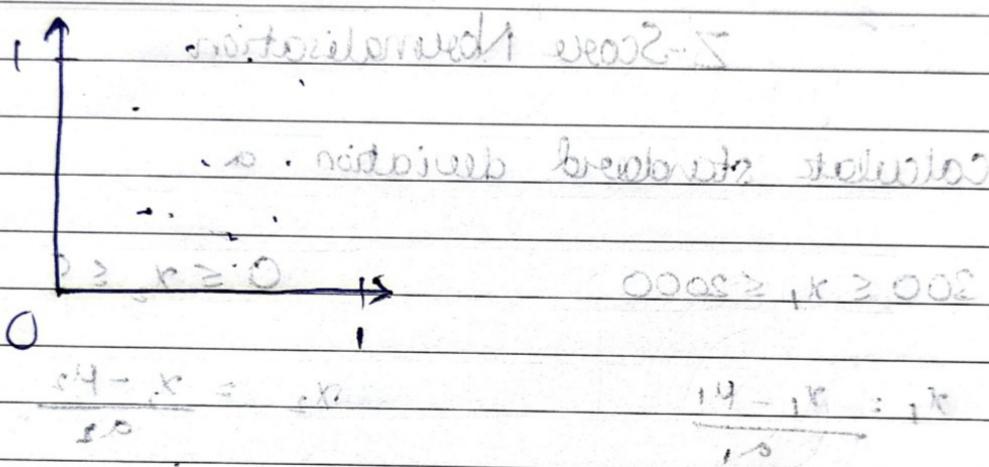
$$0.15 \leq x_{1, \text{scaled}} \leq 1$$

$$0 \leq x_{2, \text{scaled}} \leq 1$$

Earlier:



After Scaling:



### Mean Normalisation

- Step ① find mean (average) ( $\mu$ )  
 Step ② divide each term of the feature by (max-min)  
 after subtracting avg.

e.g.

$$300 \leq x_1 \leq 2000 \quad \sigma \leq x_2 \leq 55$$

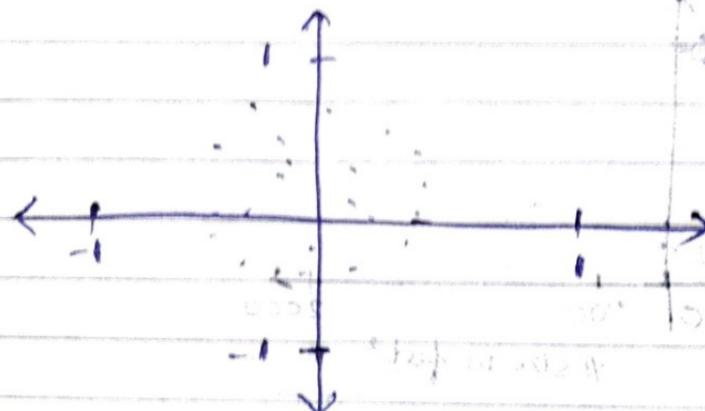
$$x_1 = \frac{x_1 - \mu_1}{2000 - 300}$$

$$-0.18 \leq x_1 \leq 0.82$$

$$x_2 = x_2 - \mu_2$$

$$\text{Upper value } 5.59$$

$$3.46 \leq x_2 \leq 0.5$$



## Z-Score Normalisation

calculate standard deviation  $\alpha$ .

$$300 \leq x_1 \leq 2000$$

$$\Leftrightarrow 0 \leq x_2 \leq 5$$

$$x_1 = \frac{x_1 - \mu_1}{\alpha_1}$$

$$x_2 = \frac{x_2 - \mu_2}{\alpha_2}$$

$$-0.67 \leq x_1 \leq 3.1$$

$$-1.6 \leq x_2 \leq 1.9$$

(1) spread) new diff. (2) pos.

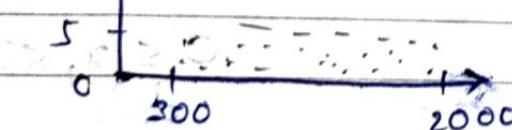
$\mu_1 = 2.3$

$\alpha_1 = 450$

#  $x_1$

$\alpha_2 = 1.4$

bedrooms



acceptable ranges:

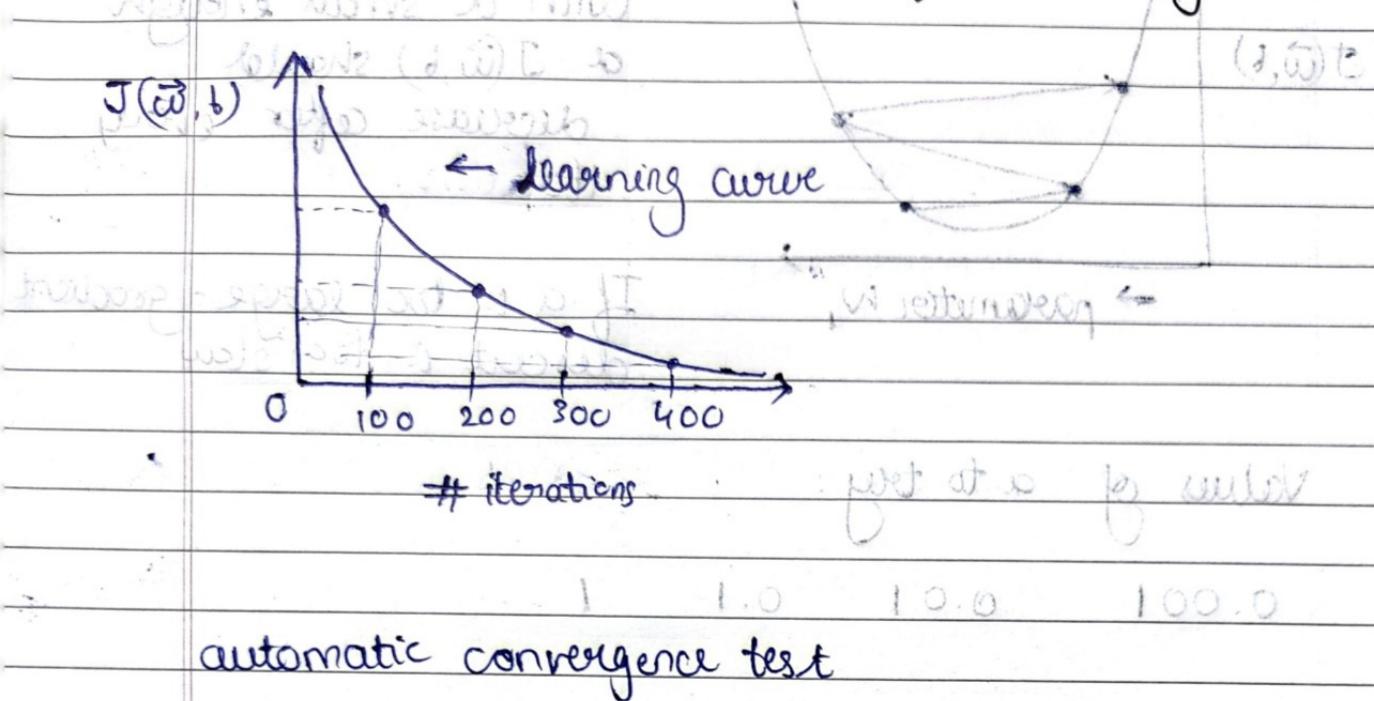
(examples)

$$-1 \leq x_j \leq 1$$

$$-3.6 \leq x_j \leq 3.6$$

$$-0.3 \leq x_j \leq 0.3$$

## Video 3: Checking Gradient Descent from Convergence.



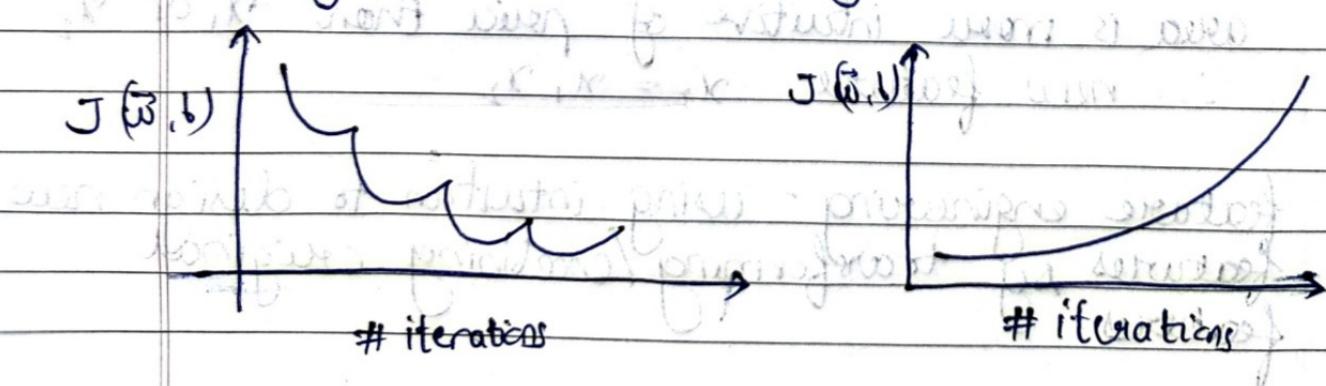
$$\epsilon = 10^{-3}$$

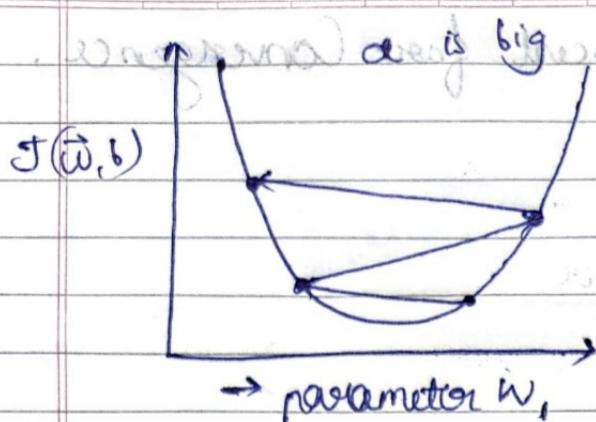
If  $J(\vec{w}, b)$  decreases by  $\leq \epsilon$  in one iteration, declare convergence.

found parameters  $(\vec{w}, b)$  to get close to global minimum

## Video 4: Choose the learning rate

Identify problem with learning rate





with a small enough  $\alpha$   $J(\vec{w}, b)$  should decrease after every iteration.

If  $\alpha$  is too large - gradient descent is too slow

Values of  $\alpha$  to try:

0.001    0.01    0.1    1 ...

test implementation

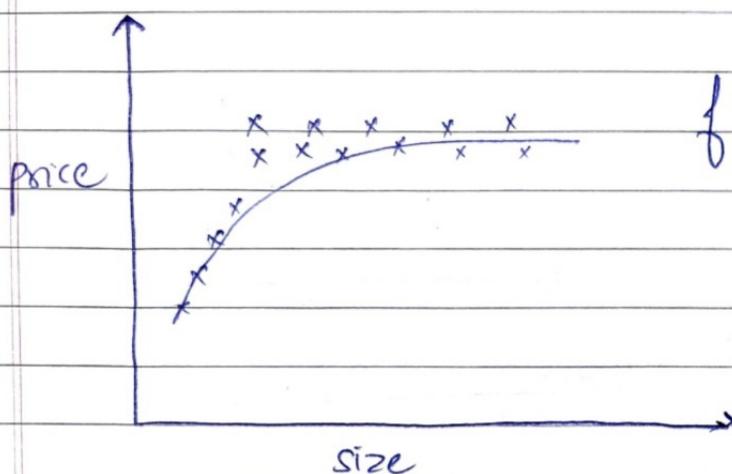
## Video 5: Feature Engineering



area is more intuitive of price than  $x_1$  or  $x_2$   
 $\therefore$  new feature:  $x_3 = x_1 \cdot x_2$

feature engineering - using intuition to design new features by transforming/combinining original features

## Video 6: Polynomial Regression



$$f \vec{w}_1(\omega_0 x + \omega_1 \sqrt{x} + b)$$