Week 02

1] Multiple Features

x_j = j^{th} feature

n = number of features

$\vec{x}^{(i)}$ = features of i^{th} training example

$x_j^{(i)}$ = value of feature j in i^{th} training example

Multiple Linear Regression Model

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

$\vec{w} = [w_1 \; w_2 \; \dots \; w_n]$ } Parametres of the model

b = is a number

$\vec{x} = [x_1 \; x_2 \; \dots \; x_n]$ Vector of features

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

2] Vectorization

Parameters and features

*) $\vec{w} = [w_1 \quad w_2 \quad w_3]$

*) b is a number

*) $\vec{x} = [x_1 \quad x_2 \quad x_3]$

Linear algebra count from 1

```
w = np.array([1.0, 2.5, -3.3])
b = 4
x = np.array([10, 20, 30])
```

→ Code Snippet

Code count from 0

Without vectorization

1] $f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$

```
f = w[0] * x[0] +
    w[1] * x[1] +
    w[2] * x[2] + b
```

} code Snippet

2] $f_{\vec{w},b}(\vec{x}) = \left(\sum_{j=1}^{n} w_j x_j\right) + b$ $\quad \sum_{j=1}^{n} \rightarrow j = 1 \ldots n$ (1, 2, 3)

range (0, n) / rangen → $j = 0 \ldots n-1$

```
f = 0
for j in range(0, n):
    f = f + w[j] * x[j]
f = f + b
```

} Code Snippet

With Vectorization

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

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

3 code Snippet

Without Vectorization	Vectorization
`for j in range(0,16)` `  f = f + w[j] * x[j]`	`np.dot(w, x)`
t_0, $f + w[0] * x[0]$	t_0, $w[0] \mid w[1] \mid \dots \mid w[15]$
t_1, $f + w[1] * x[1]$	$x[0] \mid x[1] \mid \dots \mid x[15]$ (each $*$)
⋮	in parallel
t_{15}, $f + w[15] * x[15]$	t_1, $w[0]*x[0] + w[1]*x[1] + \dots + w[15]*x[15]$

*) Vectorization is fast, shorter code and efficient specially when scaled to large datasets

3) Gradient descent for Multiple linear regression

	previous notation	Vector notation
Parameters	$w_1, \dots w_n$ b	$\vec{w} = [w_1, \dots w_n]$ b
Model	$f_{\vec{w},b}(\vec{x}) = w_1x_1 + \dots + w_nx_n + b$	$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$
Cost function	$J(w_1, \dots w_n, b)$	$J(\vec{w}, b)$

Gradient descent	repeat {
	$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_1, \dots w_n, b)$
	$b = b - \alpha \frac{\partial}{\partial b} J(w_1, \dots w_n, b)$
	}
		repeat {
		$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$
		$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b)$
		}

One feature	n features ($n \geq 2$)
repeat {	repeat {
$w = w - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$	$w_j = w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_j^{(i)}$
$\Downarrow$	$\Downarrow$
$\frac{\partial}{\partial w} J(w,b)$	$\frac{\partial}{\partial w_j} J(\vec{w}, b)$
	for $j = 1, \ldots, n$
$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})$	$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})$
simultaneously update w, b	simultaneously update w_j (for $j = 1, \ldots, n$) and b
}	}

An alternative for gradient descent

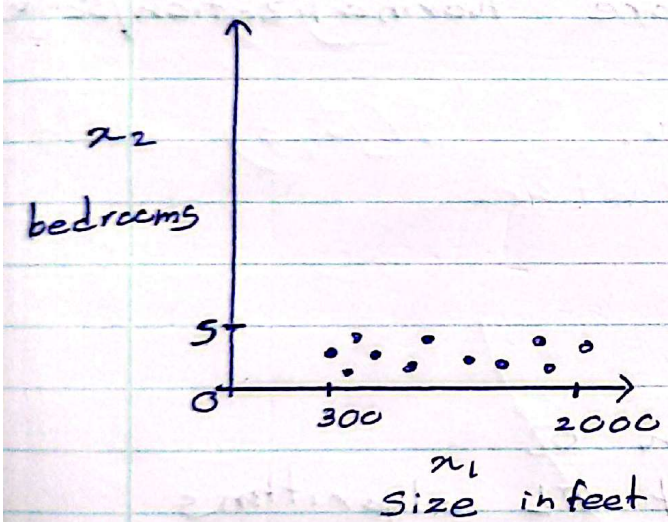
*) while gradient descent is the recommended method for finding parameters w, b Normal equation (solve w, b without iterations) may be used in ML libraries that implement linear regression)

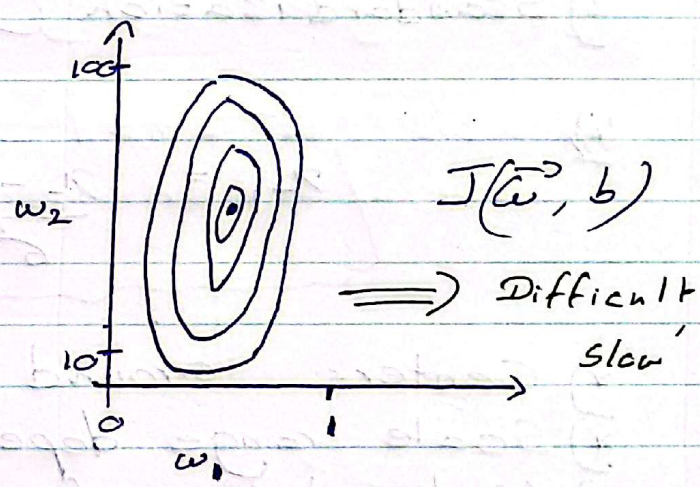
4] Feature Scaling

*) Feature scaling is a preprocessing step in ML where you adjust the range of input features (variables) so that they are on a similar scale.

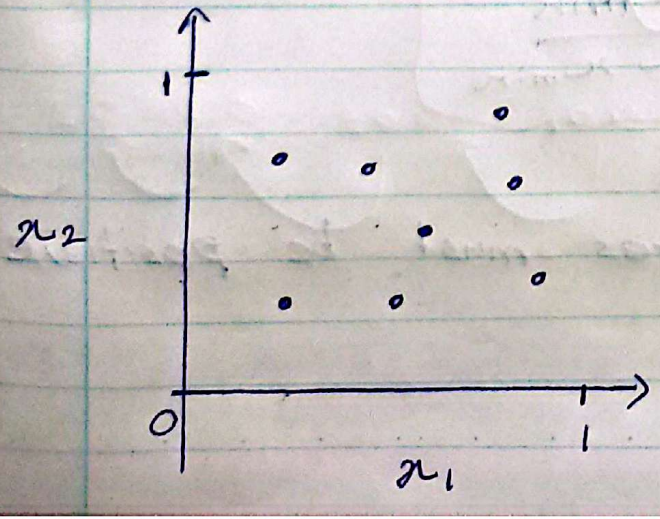
*) Many ML algorithms work best when all the features have similar range.

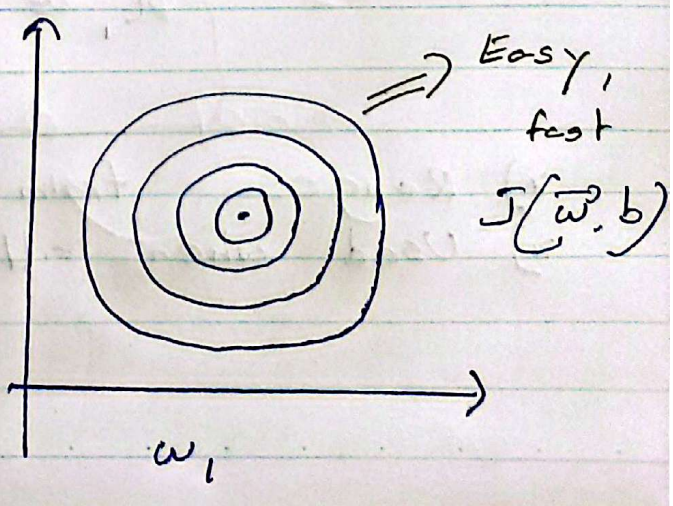
Feature size and Gradient ~~Descent~~ descent





After Rescaling





Feature Scaling Method

1) Mean Normalization

$$x_1 = \frac{x - \mu}{x_{max} - x_{min}}$$

*) Centers around 0
*) Scale range usually $[-1 \Rightarrow 1]$
*) Sensitive to outliers

2) Standardization (Z-score normalization/Scaling

$$x_1 = \frac{x - \mu}{\sigma}$$

*) Centers around 0
*) Scale range depends on σ
*) Work better for most ML algorithms

3) Min max Scaling

$$x_1 = \frac{x - x_{min}}{x_{max} - x_{min}}$$

*) Ranges from 0 - 1
*) Used when all features must be positive

$0 \leq x_1 \leq 3$ (No rescale needed

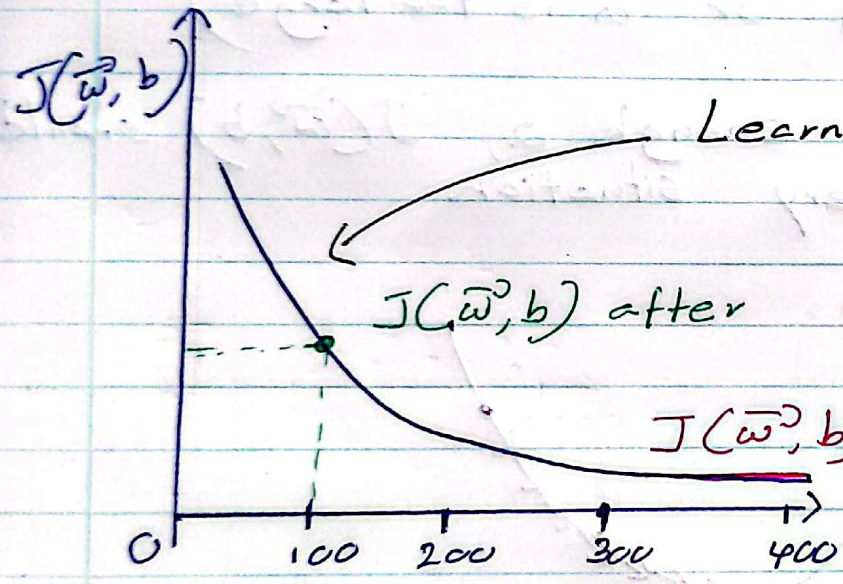
$-2 \leq x_2 \leq 0.5$ (No rescale needed)

$-100 \leq x_3 \leq 100$ (Too large, rescale)

$-0.001 \leq x_4 \leq 0.001$ (Too small, rescale)

$98.6 \leq x_5 \leq 105$ (Too large → rescale)

5) Checking gradient descent and convergence



$J(\vec{w}, b)$ should be decrease after every iteration

⑩ Automatic convergence test:

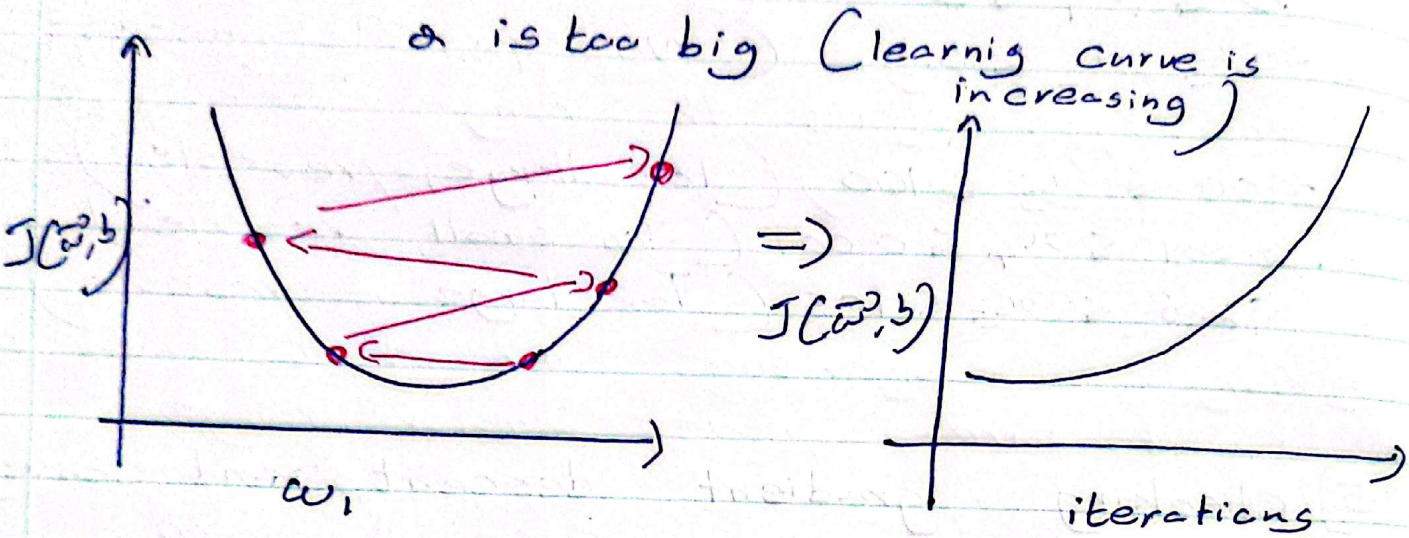
Let $\epsilon = 10^{-3}$

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

*) Convergence => (Found parameters $\vec{w}, b$ to get close to global minimum)

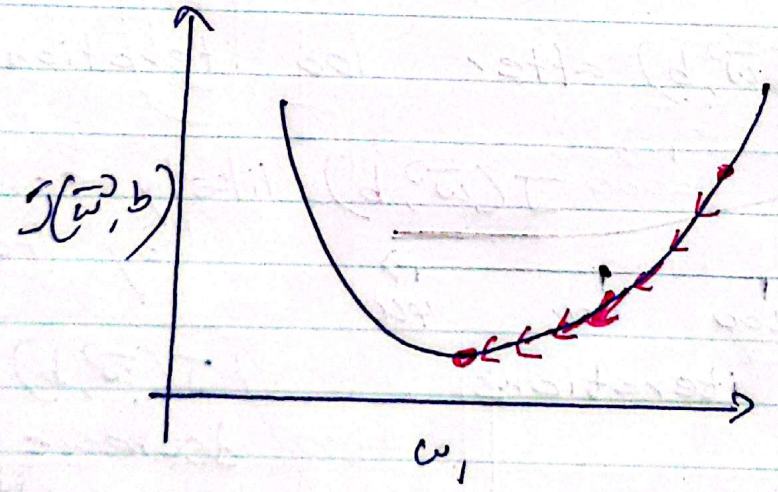
b) Chosing the learning rate

α is too big (learnig curve is increasing)

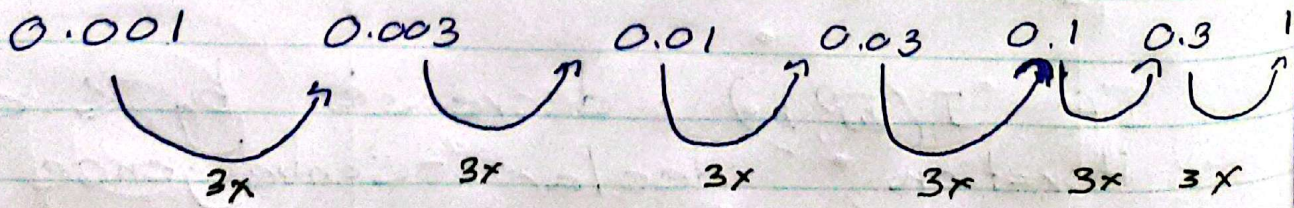


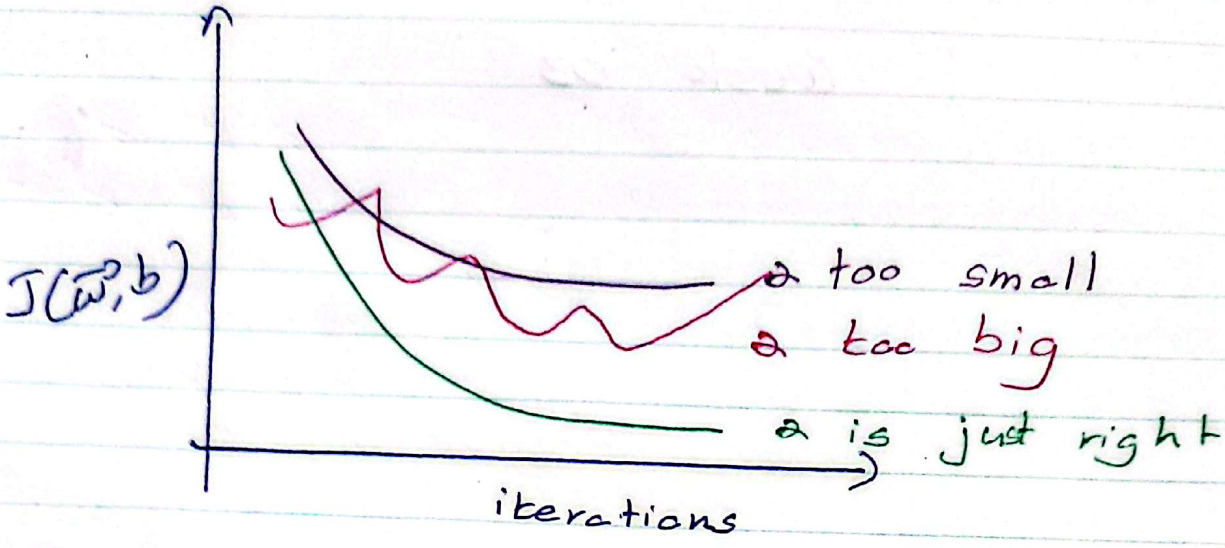
*) Not good (bug or α is too large)

*) With a small enough α, $J(\vec{w},b)$ should decrease on every situation



Values of α to try:





7) Feature Engineering

*) Using intution to design new features, by combining or transforming original features

*) It improves model's performance as the algorithm generalize better to new data and learn faster

Turning raw data → Useful form

8) Polynomial Regression

*) Polynomial regresion is a type of linear regression where the relationship between input x and output y is modeled as an nth degree polynomial (curve) instead of a straight line

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 - - + w_n x^n$$