House sizes:
2104
1416
1534
852

Have 3 competing hypothesis

1. $h_\theta(x) = -40 + 2.5x$
2. $h_\theta(x) = 200 + 0.1x$
3. $h_\theta(x) = -150 + 0.4x$

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix} \times \begin{bmatrix} -40 & 200 & -150 \\ 0.25 & 0.1 & 0.4 \end{bmatrix} = \begin{bmatrix} 486 & 410 & 692 \\ 314 & 342 & 416 \\ 344 & 353 & 464 \\ 173 & 285 & 191 \end{bmatrix}$$

Properties for matrix multiplication

→ A and B. $AB \neq BA$. (not commutative)

→ $3 \times (5 \times 2) = (3 \times 5) \times 2$, $A \times (B \times C) = (A \times B) \times C$

Associative. (Matrix mult is associative)

→ Identity I ($I_{n \times n}$) $I \times A = A \times I = A$.

→ Inverse. $A A^{-1} = A^{-1} A = I$. (A must be a square matrix)

(Singular, degenerate matrices) → don't have inverse.

⇒ Multiple features

Earlier, we had only size of house on the basis of which we had to predict price

Size x_1	No. of bedrooms (x_2)	No. of floors (x_3)	Age of home (x_4)	Price ($1000's) ($y$)	
2104	5	1	45	460	$n = 47$
1416	3	2	40	232	
1534	3	2	30	315	
852	2	1	36	178	

Notation: n = number of features $n = 4$ here.

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

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

→ $x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$ → 4 features that determine price of 2nd house.

vector.

$x^{(i)}$ → n dimensional vector.

→ $x_3^{(2)} = 2$ $\quad x_1^{(2)} = 1416$

Hypothesis

Previously $h_\theta(x) = \theta_0 + \theta_1 x$

Now, $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$.

eg. $h_\theta(x) = 80 + 0.1x_1 + 0.01x_2 + 3x_3 - 2x_4$

$h_\theta(x)$: Price of house.

$0.1x_1$: effect of size on price

$0.01x_2$: effect of bedrooms on price

$3x_3$: effect of floors on price

$-2x_4$: effect of age on price.

→ n features

→ $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots \theta_n x_n$

we define $x_0 = 1$. $\quad (x_0^{(i)} = 1)$

↳ we have basically defined an additional feature that always has value of 1 for any training set.

So $X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ → (n+1) dimensional vector.

$\in \mathbb{R}^{n+1}$.

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$.

→ 0 indexed vector.

$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 \dots \theta_n x_n$

↳ we can write this as $\theta^T x$.

$= \underbrace{\begin{bmatrix} \theta_0 \theta_1 \dots \theta_n \end{bmatrix}}_{1 \times n} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}}_{n \times 1} = \theta_0 x_0 + \theta_1 x_1 \dots \theta_n x_n$.

parameter vector.

feature vector

This is called multivariate linear regression.

⇒ Gradient Descent for multiple features

Hypothesis : $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_n x_n$.

Parameters : $\theta_0, \theta_1 \dots \theta_n \longrightarrow \theta \rightarrow n+1$ dimensional vector.

cost function : $J(\theta_0, \theta_1 \dots \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

we will call this as $J(\theta)$ where θ is vector $(n+1)$ dim.

Gradient Descent : Repeat {

$$\theta_j := \theta_j - \alpha \frac{d}{d\theta_j} J(\theta_0 \dots \theta_n)$$ ← $J(\theta)$

} (simultaneously update for every $j = 0, 1, \dots n$)

Previously ($n=1$) :

Repeat { $\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)}_{\frac{d}{d\theta_0} J(\theta)}$

$\rightarrow \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$ → Now $x_1^{(i)}$

(simultaneously update θ_0, θ_1)

}.

Now ($n \geq 1$) : Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

(simultaneously update θ_j for $j = 0, \dots n$).

}.

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_2^{(i)}$$

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

$$= \frac{1}{2m} \left(h_\theta x^{(1)} - y^{(1)} \right)^2 + \left(h_\theta x^{(2)} - y^{(2)} \right)^2 + \left(h_\theta x^{(3)} - y^{(3)} \right)^2 \dots$$

$$= \frac{1}{2m} \left(\theta_0 x_0^{(1)} + \theta_1 x_1^{(1)} \dots \theta_n x_n^{(1)} - y^{(1)} \right)^2 + \left(\theta_0 x_0^{(2)} + \theta_1 x_1^{(2)} \dots \theta_n x_n^{(2)} - y^{(2)} \right)^2 \dots$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{2m} \cdot \left(2 \left(h_\theta x^{(1)} - y^{(1)} \right) \cdot x_j^{(1)} + 2 \left(h_\theta x^{(2)} - y^{(2)} \right) \cdot x_j^{(2)} \dots \right)$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta x^{(i)} - y^{(i)} \right) x_j^{(i)}$$

⇒ GD in practice : feature scaling

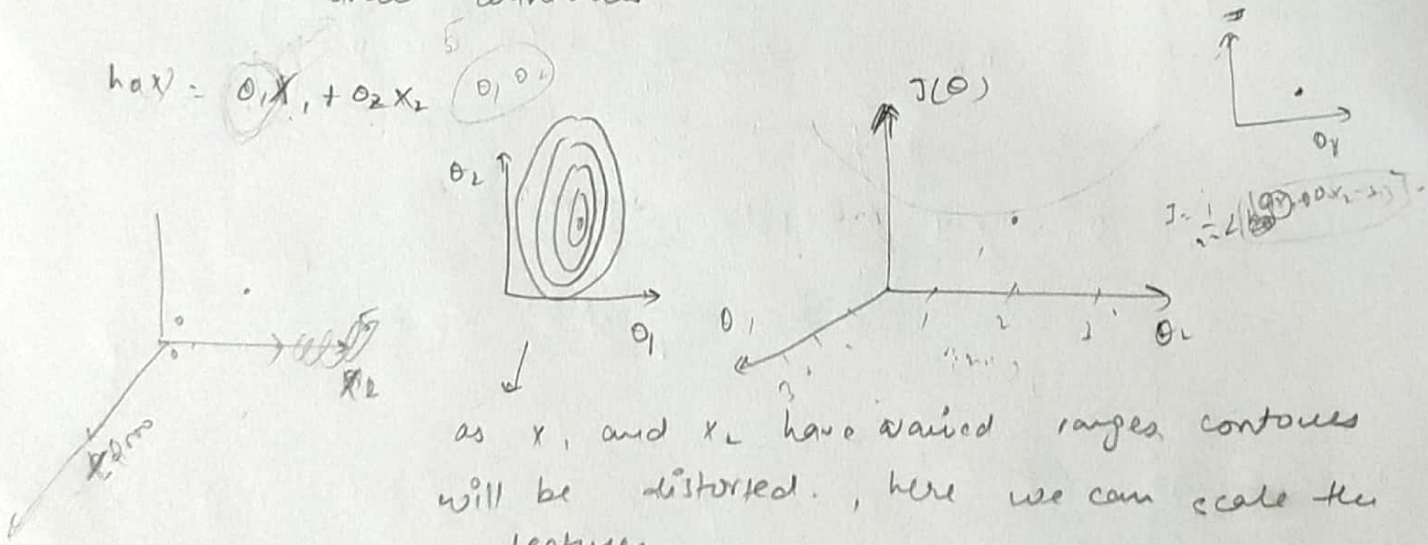
Idea: Make sure features are on a similar scale.

Eg. x_1 = size (0-2000 feet^2)
x_2 = No. of bedrooms (1-5)

↳ makes GD converge more quickly.

↳ let's draw contours

$h_\theta x = \theta_1 x_1 + \theta_2 x_2$

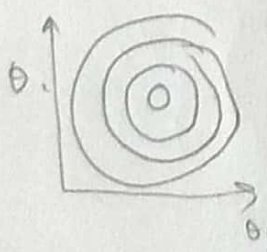


as x_1 and x_2 have varied ranges, contours will be distorted., here we can scale the features

$$x_1 = \frac{\text{size (feet}^2\text{)}}{200}, \qquad x_2 = \frac{\text{No. of bedrooms}}{5}$$

→ Now, contours would be circular.

GD would work much easily here as the trajectory would be easy



→ So generally, get every feature into approx. $-1 \le x_i \le 1$ range

also like $0 \le x_1 \le 3$, $-2 \le x_2 \le 0.5$ } → These would also work as range is not far however.

$-100 \le x_3 \le 100$ } X → This wouldn't work.

$-0.0001 \le x_4 \le 0.0001$ } → not workable.

→ Mean normalisation

Replace x_i with $x_i - \mu_i$ to make features have approx. zero mean (not for $x_0 = 1$)

eg. $x_1 = \frac{\text{size} - 1000}{2000}$ → Avg = 1000

$x_2 = \frac{\text{bedrooms} - 2}{5}$ → Avg = 2

So $-0.5 \le x_1 \le 0.5$ and $-0.5 \le x_2 \le 0.5$.

General rule: replace x_i with $\frac{x_i - \mu_i}{s_i}$

where → μ_i → average

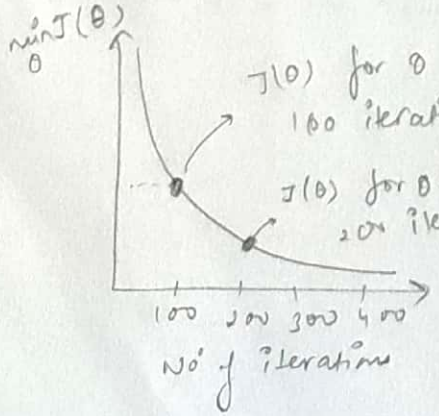
s_i → range = (max - min value)

⇒ Gradient Descent in Practice 2: Learning rate (α)

GD: $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$

- 'Debugging': How to make sure GD is working correctly.
- How to choose learning rate α.

Making sure GD is working correctly



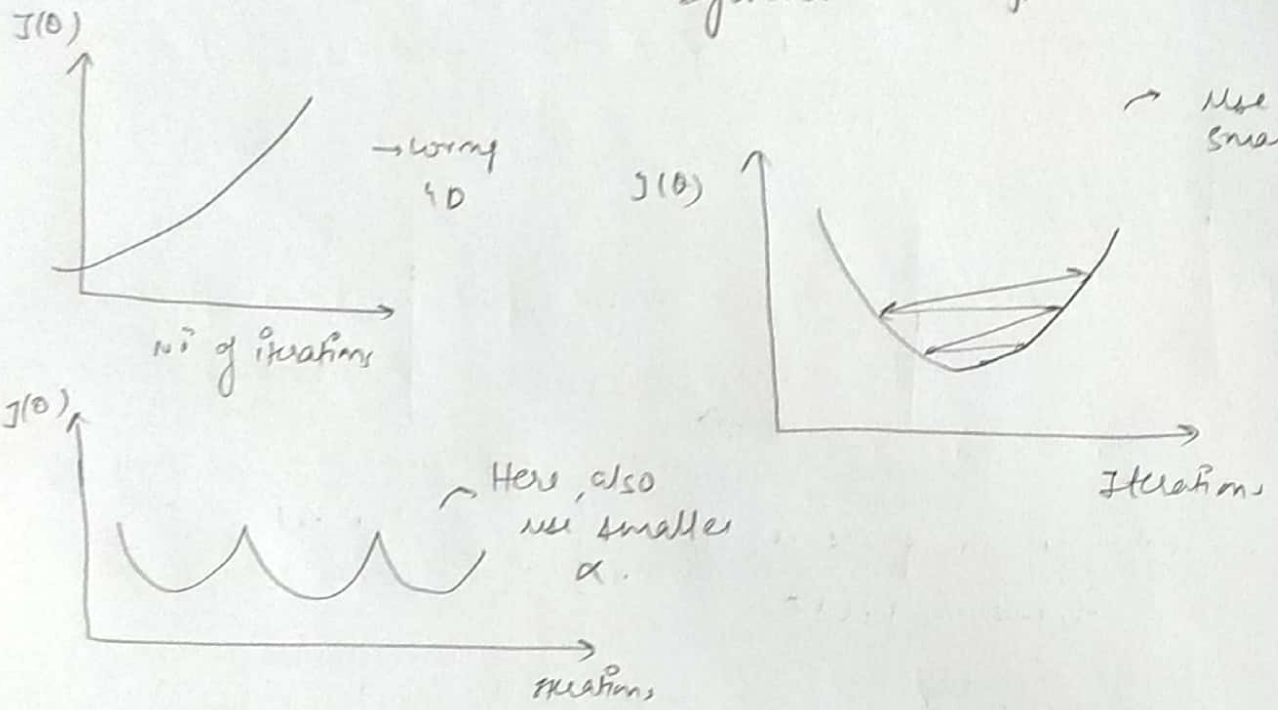
Thus, if GD is working correctly, then $J(\theta)$ decreases with number of iterations

Also, from the curve, it is clear that $J(\theta)$ decreases slowly as iteration increase.

We can also do automatic convergence test

↳ ex. Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration.

↳ However, guessing the number 10^{-3} beforehand is difficult.



→ For sufficiently small α, $J(\theta)$ should decrease on every iteration

→ But if α is too small, GD can be slow to converge.

Summary :- - If α is too small : slow convergence

- If α is too large : $J(\theta)$ may not decrease on every iteration, may not converge (slow converge also possible)

Now, to debug all this, graph can help

To choose α, try 0.001 0.01 0.1 1

, 0.001, 0.01, 0.1, 1 and choose the α the seems to cause $J(\theta)$ to decrease rapidly.

⇒ Features and Polynomial Regression

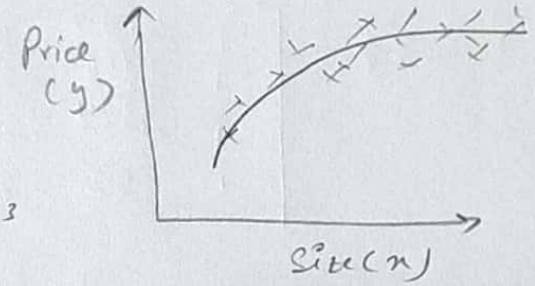
Housing price prediction.

$h_\theta(x) = \theta_0 + \theta_1 \times \underbrace{\text{frontage}}_{x_1} + \theta_2 \times \underbrace{\text{depth}}_{x_2}$

Area = x = frontage × depth.

let $h_\theta(x) = \theta_0 + \theta_1 x$ → land area.

Polynomial Regression



may be $\theta_0 + \theta_1 x + \theta_2 x^2$ fit

or $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$

$= \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2 + \theta_3(\text{size})^3$ } set features

$x_1 = \text{size}$ $x_2 = \text{size}^2$ $x_3 = \text{size}^3$

feature scaling becomes important

- size - 1-1000
- size² - 1-1000,000
- size³ - 109

Choice of features is important.

Ex.

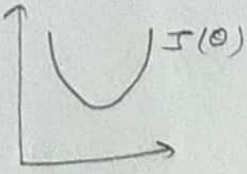
$h_\theta(x) = \theta_0 + \theta_1(\text{size}) + \theta_2(\sqrt{\text{size}})$

⇒ Normal equation → Helps to solve parameters.

GD → Iterative algorithm, but now, we solve for $J(\theta)$ min analytically

Intuition: If 1D ($\theta \in \mathbb{R}$)

$J(\theta) = a\theta^2 + b\theta + c$ $\frac{dJ(\theta)}{d\theta} = 0$ → and get θ.



but for us

$(\theta \in \mathbb{R}^{n+1})$ $J(\theta_0, \theta_1, \theta_2 \cdots \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} ((h_\theta(x^{(i)}) - y^{(i)})^2)$

So here, $\frac{\partial}{\partial \theta_j} J(\theta) = 0 \cdots = 0$ (for every j)

then solve for $\theta_0, \theta_1 \cdots \theta_n$

Examples $m = 4$

	Size (feet²)	No of bedrooms	No of floors	Age of home	Price (×1000)
x_0	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 2 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 38 \end{bmatrix} \quad \hookrightarrow m \times (n+1)$$

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix} \quad \hookrightarrow (m \times 1)$$

$$\boxed{\theta = (X^T X)^{-1} X^T y}$$

θ the the parameters for minimum $J(\theta)$ (cost function)

→ m examples $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \dots (x^{(m)}, y^{(m)})$; n features.

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$X = \begin{bmatrix} \text{——} (x^{(1)})^T \text{——} \\ \text{——} (x^{(2)})^T \text{——} \\ \vdots \\ \text{——} (x^{(m)})^T \text{——} \end{bmatrix}$$

(design matrix)

eg. If $x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$ → $X = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ & \vdots \\ 1 & x^{(m)} \end{bmatrix}$ $y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$

$$\boxed{\theta = (X^T X)^{-1} X^T y}$$

Also, feature scaling is unnecessary in this method, or eventually we'll get same value of θ

GD	Normal equation
→ Need to choose α	→ No need to choose α
→ Needs many iterations	→ Don't need to iterate
→ Works well even when n is large	→ Need to compute $(X^T X)^{-1}$ — slow if n is large.
	$n = 1000$ → works fine for normal.
	$n = 10000$ → normal. $n = 10^6$ → GD

⇒ Normal equation and non-invertability

$\theta = (X^T X)^{-1} X^T y$ → what if $X^T X$ is non invertible (singular / degenerate)?

$(X^T X)$ → is non invertible.

↳ this can be because of → (i) Redundant features (linearly dep.)

eg. x_1 = size in ft^2

x_2 = size in m^2

(ii) Too many features (eg. $m \leq n$)

- Delete some features, or use regularisation. → later.

⇒ MATLAB Onramp

→ ; , usually command output is suppressed

→ MATLAB does not calculate all variables if some previous one is changed.

→ save filename - saves workspace variables

→ load filename - loads workspace variable

→ clear - clears workspace clc → clears command video

→ x = 1:4 → x = [1 2 3 4]

→ x = [1;2;3] → $x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

→ x = 3:2:13 → x = [3 5 7 9 11 13]

→ x = linspace (first, last, number of elements)

x = linspace (1, 10, 5) → x = [1 3.25 5.5 7.75 10]

→ x = x' (transpose operator)

→ x = rand (3) → [. . .] random numbers.

random number matrix

→ x = rand (2, 3) → 2×3 matrix is formed.

→ x = zeros (2, 3) → $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

→ size (x) → 2 × 3.

→ x(3) → index of array x(row, col) data is a 7×4 matrix

→ x = data(6,3) → element in 6th row, 3rd column

→ x = data(end, 3) → last row, 3rd colm

→ x = data(11) → traverses columnwise

→ x = data(:, 2) → give 2nd column all rows

→ [3 4] * [10 20] → Error. (Matrix multiplication)

→ [3 4] .* [10 20] = [30 80] (element wise multiplication)

→ plot(x, y) → plot same sized vectors against each other

→ plot(x, y, 'r--o') → red colour
→ -- → line style
→ o → marker style

→ hold on → plots on same curve
→ hold off → plot on diff curves.

→ plot(v1) → plotting single vector plots values against index of array.

→ plot(sample, v1, 'ro-', "LineWidth", 4)

→ title("Sample Mass") → give title to previous plot

→ ylabel("Mass (g)") → y axis label.

→ legend("Exp A", "Exp B") → add legend over graph □ → Exp A ☆ → Exp B.

→ Tables = → load datafile elements

d = elements.Density → assigns density column to variable d