[178]

☆ Week 2 ☆

--(a) ☆ Multiple features (variables) - - -

Size (feet²)	No of bedroom	No of floors	Age of Home (years)	Price ($1000's)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178

m → no of entries [

n → no of features ⟷

$x^{(i)}$ → input (features) of i^{th} training examples.

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

in m ↕

in n ↔

↳ $x^{(2)}$ = [1416, 3, 2, 40] (vector)

$x_3^{(2)} = 2$

Hypothesis

previously: $h_\theta(x) = \theta_0 + \theta_1 x$ ~~only 1 feature~~

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

$= 80 + 0.1 x_1 + 0.01 x_2 + 3 x_3 - 2 x_4$

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

Let $x_0 = 1$ $(x_0^{(i)} = 1)$

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

$h_\theta(x) = \theta^T X$ ~~Multivariate linear regression~~

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\underbrace{[\theta_0\ \theta_1 \cdots \theta_n]}_{\theta^T} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}}_{X} \quad (n+1) \times 1$$

Gradient descent for Multiple Variables

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

Parameters: $\theta_0, \theta_1, \cdots \theta_n$ ($n+1$ dimensional vector)

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

$J(\theta)$ → vector

Gradient descent:

Repeat → $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \cdots \theta_n)$ $J(\theta)$

(simultaneously update for every $j = 0, \cdots, n$)

Gradient Descent

- Previously (n=1)

Repeat {

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

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

$x_1^{(i)}$

(simultaneously update θ_0, θ_1)

}

- New algorithm (n ≥ 1)

Repeat {

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

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

}

$x_0^{(i)} = 1$

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

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

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

Gradient Descent in Practice I

Page No: 10 Sadguru
Date: / /

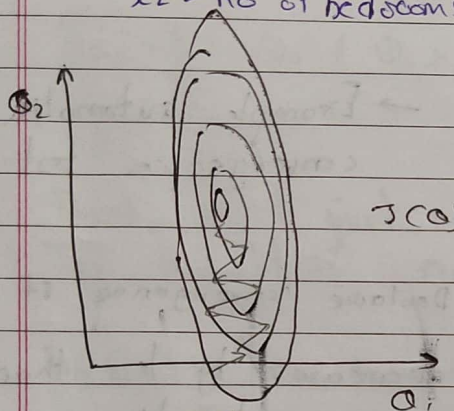
Tips to run gradient descent faster

1) Feature scaling in gradient descent algo.

Idea: Make sure features are on similar scale.

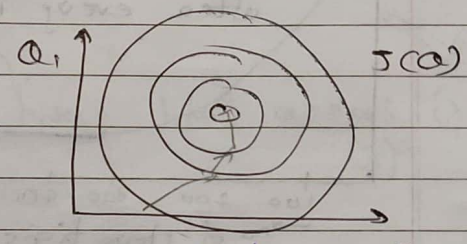
eg. x_1 = size (0-2000 feet^2) $\quad x_1 = \frac{\text{size (feet}^2)}{2000}$

x_2 = no of bedrooms (1-5) $\quad x_2 = \frac{\text{no of bedrooms}}{5}$



Takes more time.

$0 \le x_1 \le 1 \qquad 0 \le x_2 \le 1$



runs faster

* Feature scaling

Get every feature into approximately $-1 \le x_i \le 1$ range.

$x_0 = 1$ ← always right

$0 \le x_1 \le 3$ ✓ (−3 fine 3)

$-2 \le x_2 \le 0.5$ ✓

$-100 \le x_3 \le 100$ ✗ (−1/3 fine 1/3)

$-0.0001 \le x_4 \le 0.0001$ ✗

- Mean normalization:

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$)

$x_1 = \frac{\text{size} - 1000}{2000} \qquad x_2 = \frac{\#\text{bed} - 2}{5}$

$-0.5 \le x_1 \le 0.5 \qquad -0.5 \le x_2 \le 0.5$

$x_i = \frac{x_i - \mu_i}{s_i}$ ← average value of x_i in training set

range (max − min) / standard deviation

$x_1 = \frac{x_1 - \mu_1}{s_1}$

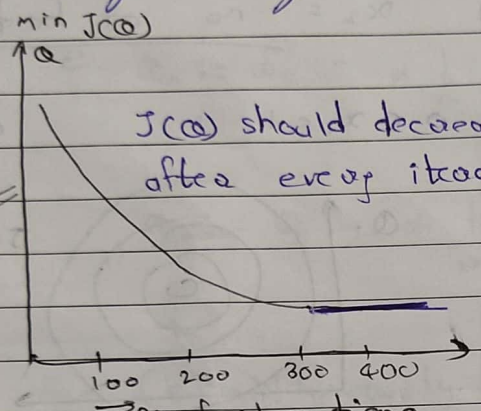
Gradient Descent in Practice II

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

2) Debugging : How to make sure gradient descent is working correctly.

How to choose learning rate α

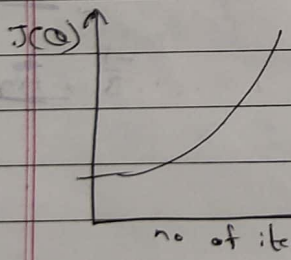
Making sure gradient descent is working correctly.



$J(\theta)$ should decrease after every iteration → Example automatic convergence test.

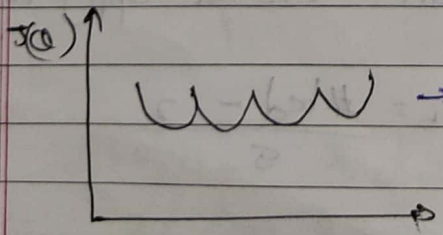
→ Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration

→ Here we are very near to minima.



→ Gradient descent is not working properly

→ use smaller α to work properly.



→ use smaller α.

- For sufficiently small α, $J(\theta)$ should decrease on every iteration
- But if α is too small gradient descent can be slow to converge.

To choose α try.

0.0001, 0.001, 0.01, 0.03, 0.1, 0.3, 1

--② Features and Polynomial Regression ---

House price prediction

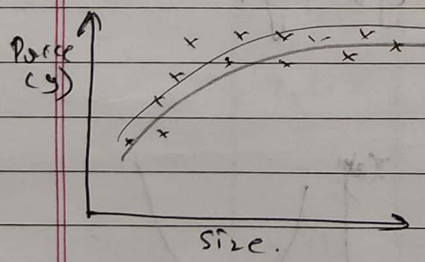
$h_\theta(x) = \theta_0 + \theta_1 \times \underbrace{\text{frontage}}_{x_1} + \theta_2 \times \underbrace{\text{depth}}_{x_2}$

Think, don't just use given,
Try to obtain new feature like area: $(x_1) * (x_2)$

$h_\theta(x) = \theta_0 + \theta_1 \times (\text{area})$

This may make model accuracy ↑.

Polynomial regression: ~~simple linear~~ regression



$\theta_0 + \theta_1 x + \theta_2 x^2$ ✓ size ↑, price ↑

$\theta_0 + \theta_1 x + \theta_2 x^2$ ✗ size ↑, price ↓

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

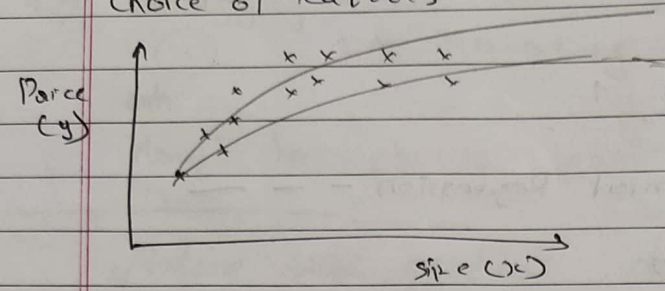
$= \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size}) + \theta_3(\text{size})$

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

feature scaling gets importance.

x_1 : 1 - 1000 (size)
x_2 : 1 - 1 000 000 $(\text{size})^2$
x_3 : 1 - 10^9 $(\text{size})^3$

choice of features. $h_\theta(x) = \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2$



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

que model: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2\sqrt{x}$ $\quad x \to 1 \sim 1000$ feet^2

$\sqrt{1000} = 32$

Ans $x_1 = \frac{\text{size}}{1000}$, $x_2 = \frac{\sqrt{\text{size}}}{32}$

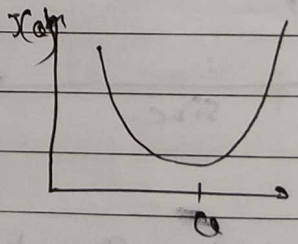
-- (a) Normal Equation — — —

Normal equation → Without iterating, (running gradient descent) calculate θ_j in one go.

Method to solve for θ analytically.

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

$J(\theta) = a\theta^2 + b\theta + c$



To ~~fin~~ make $J(\theta)$ minimum

$\frac{d}{d\theta} J(\theta) =$ set 0

solve for θ.

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

$\frac{\partial}{\partial \theta_j} J(\theta) =$ - set 0 (for every j)

solve for $\theta_0, \theta_1, \dots \theta_n$

e.g. $m = 4$

	size(feet)^2	# bed	# floors	Age	Price
We have added this	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

Design Matrix

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 36 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$m \times (n+1)$ matrix — m-dimensional vector

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

General case

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

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

$(n+1)$ dimensional vector

X = Design matrix

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

e.g. If $x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$

$$X = \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x_1^{(2)} \\ 1 & x_1^{(4)} \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

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

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

Feature scaling is not necessary.

$0 \le x_1 \le 1$
$0 \le x_2 \le 1000$
$0 \le x_3 \le 10^{-5}$
are okay.

octave : ~~pinv(X~~ pinv(x' * x) * x' * y

$X^T \longrightarrow X'$

m training examples, n features

Gradient Descent	Normal Eqn.
Need to choose α	no need
Need many iterations	no need.
Works well even when n is large.	$(X^T X)^{-1}$ slow to invest if n is very large
$n = 10^6$	$n = 100, 1000, 10000$

~~n = m here~~, n → no of features. I guess

$h(x) = \text{theta}^T * x$

$h(x) = x * \text{theta}$

Sadguru

Page No: 16

Date: / /

--@ Normal Equation Noninvertibility — — —

☆ Normal eq^n invertiablity. (cocurely)

Normal eq^n: $\theta = (X^T X)^{-1} X^T y$

pinv $(x' * x) * x' y$

Issue → what if $(X^T X)^{-1}$ is non-invertible?. (singular / degenerat)

octave has two fun to inverse :- i.e. pinv inv

even
works ~~eq~~ with $X^T X$ as singular.

It computes θ.

What if $X^T X$ is non-invertable?

→ where it can't be invertable

- Redundant features (linearly dependent)

e.g. x_1 = size in feet^2

x_2 = size in m^2 $\quad 1m = 3.28$ feet

dependent features make $(X^T X)$ ~~non-invertible~~.

$x_1 = (3.28)^2 x_2$

So, keep only one among x_1, x_2.

- Too many features ($m \leq n$)

Delete some features, or use regularization.

↓ latter

-- @ Basic operations - - -

* Octave

5+6 → ans=11

1/2 → 0.5000

2^6 → 64

1==2 % false → ans=0

1~=2 → 1

1&&0 → 0

1||0 → 1

xor(1,0) → 1

PS1('>> ');

a=3 → a=3

a=3; → nothing print

a → 3

b='hi'

a=pi;

a → 3.1416

disp(a); → printf(a)

disp(sprintf('2 decimals : %.0.2f', a))

↳ 2 decimals : 3.14

format short

format long

A=[1 2; 3 4; 5 6]

A= 1 2

3 4

5 6

A=[1 2;

3 4;

5 6] same.

start inc stop

V = 1 : 0.1 : 2

V =

Columns 1 through 7:

1 1.1 1.2 1.3 1.4 1.5 1.6

1.7 1.8 1.9 2

v = 1 : 6

v = 1 2 3 4 5 6

ones(2,3)

1 1 1

1 1 1

C = 2 * ones(2,3)

C = 2 2 2

2 2 2

w = zeros(1,3)

rand(3,3)

- - - 0 to 1 nos

- - -

- - -

w = randn(0,1,3)

w = -0.33 1.26 -0.2

w = -6 + sqrt(10) * (randn(1,10000))

hist(w)

hist(w, 50) → 50 bins

eye(4)

1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1

help eye.