Multivariate Linear Regression

We have seen single variable (x) linear regression which is

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

Now, we have multivariate linear Regression:

Ex

Size (feet^2)	# of bedrooms	# of floors	Age of home	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	2	2	30	315
852	2	1	36	178
...				
x_1	x_2	x_3	x_4	

Now we have multiple variables of x and one y.

We call all x's $\Rightarrow$ $n = 4$ (in the above Ex)

Training set size $\Rightarrow$ $m = 4$ (4 Rows considered)

$x^{(i)}$ the input features of i^{th} training set (row)

Ex: $x^2 = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$ — $x^2_{(1)}$, $x^2_{(2)}$, $x^2_{(3)}$, $x^2_{(4)}$

$x^{(i)}_j$ value of feature of i^{th} training set (i^{th} row and j column:

Ex $x^2_{(4)} = 40$

multivariable Hypothesis :-

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

always $x_0 = 1$ (reason :- vector multiplication)

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n+1} \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n+1} \end{bmatrix}$$

we can do $\theta^T x$

$$h_\theta(x) = \begin{bmatrix} \theta_0, \theta_1, \theta_2 \cdots \theta_{n+1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n+1} \end{bmatrix} = \theta^T x$$

Gradient Descent for multiple variable :-

for single variable $n = 1$

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

repeat until converge {

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

partial derivative of $\frac{d}{d\theta_0}(J, \theta_0, \theta_1)$

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1}(J(\theta)) := \theta_1 - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta x^{(i)} - y^{(i)} \right) x^{(i)}}_{\frac{d}{d\theta_1} J(\theta_0, \theta_1)}$$

(Simultaneously update θ_0, θ_1)

$$\theta_j := \theta_j - \alpha \frac{d}{d\theta_j}(J(\theta))$$ - common algo for gradient descent.

multiple variable gradient descent algorithm

$(n \geq 1)$:

repeat until converges {

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

for $j = 0, 1, 2 \cdots n$ }

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

$x_0 = 1$ always.

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

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

$$\vdots$$

$$\theta_n := \theta_n - \alpha \frac{1}{m} \frac{\partial}{\partial \theta_n} \left(h_\theta(x^{(i)} - y^{(i)}\right) \cdot x_n^{(i)}$$

Gradient Descent in practice 1 :- Feature Scaling

Ex: x_1 = size (feet^2) (1-2000)

x_2 = # of bedrooms (1-5)

} when you have kind range for your feature it will skew your cost function as in proportion of size and # of bedroom is high

So you can scale the features

like $x_1 = \frac{\text{size}}{2000}$

$x_2 = \frac{\text{\# of bedrooms}}{5}$

$-1 \leq x \leq 1$ ✓

$-0.5 \leq x \leq 0.5$ ✓

$-3 \leq x \leq 3$ ✓

$-\frac{1}{3} \leq x \leq \frac{1}{3}$ ✓

$-100 \leq x \leq 100$ ✗

$-0.001 \leq x \leq 0.0001$ ✗

This way the values will come $0 \leq x_1 \leq 1$ and $0 \leq x_2 \leq 1$

Mean feature scaling:-

x_1 = Size feets (1-2000)

x_2 = # of bedroom (1-5)

You can mean value of i^{th} feature

you have i^{th} formula

for mean feature scaling.

$$X_i := \frac{x_i - \mu_i}{S_i}$$

x_i = feature

μ_i = mean value of i^{th} feature

S_i = range of i^{th} feature or (standard deviation of i^{th} feature)

Ex from above $\mu_1 = 1000$ $\mu_2 = 2$

$$x_1 := \frac{\text{Size(feet)} - 1000}{2000}$$ (2000-1) = 1999

$$x_2 := \frac{\#\text{of bed} - 2}{5}$$ (it can 5-1 = 4

Gradient Descent in practice II - Learning Rate

You can choose your α - learning rate like:

0.001 , 0.01 , 0.1 , 1 . . .

↓ ↓ ↓

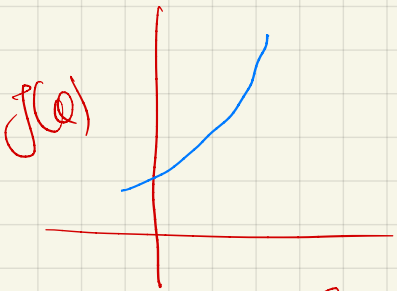
0.003 0.03 0.3 ? . . .

if your α is too small: slow convergence

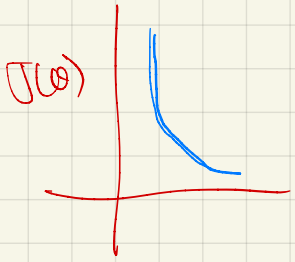
α is too high: may not decrease on every iteration and thus may not converge

So use α small enough to converge soon enough!

→ from this you can plot $J(\theta)$ vs iterations it took to see which is a good α.

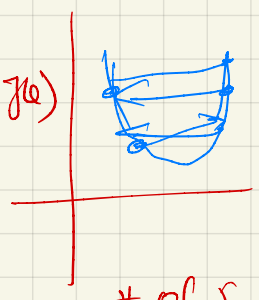


α = too high



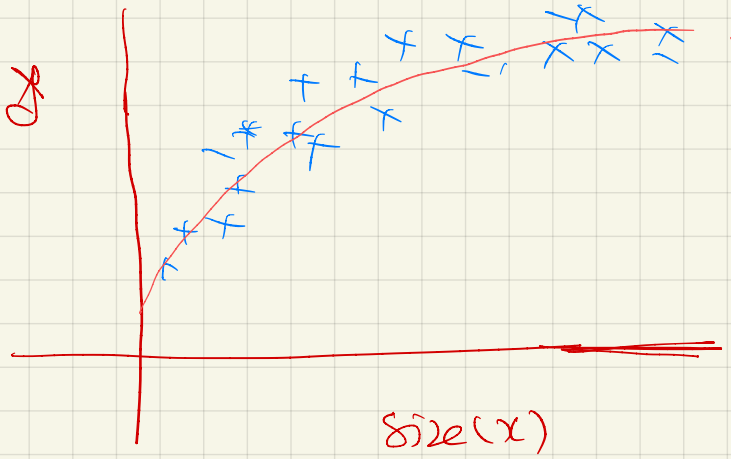
α = Middle

α = too slow.



α = too high

Feature and Polynomial Regression:-



You have hi training set like this and You can't fit wime linear Regression

So you use polynomial regression:-

if you use $h_\theta(x) = \theta_0 x_0 + \theta_1 (x_1) + \theta_2 (x_1)^2$ quadratic function there is a change that it will curve down

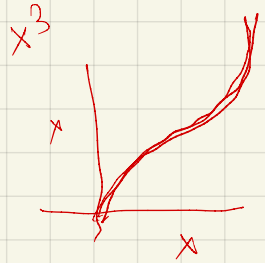
So use

$h_\theta(x) = \theta_0 x_0 + \theta_1 (x_1) + \theta_2 (\sqrt{x_1})$

square root function or

$h_\theta(x) = \theta_0 x_0 + \theta_1 (x_1) + \theta_2 (x_1)^2 + \theta_3 (x_1)^3$

Cube root function. It will not curve down also



But in Feature Scaling is very important when you do polynomial regression

Feature Scaling for Polynomial

Size of house (1 to 1000)

Then

$x_1 = \frac{size}{1000}$

$\sqrt{x_1} = \frac{\sqrt{size}}{\sqrt{1000}}$

$x_1^3 = \frac{(size)^3}{1000^3}$

We can Change behavior of or curve of our hypothesis function by making Quadratic, square root, Cube root.

Normal equation:- (2nd algorithm to solve for parameter θ)

Method only for specific model of linear regression

Gradient descent is one way of minimize J. and Normal equation is the second way.

Examples: $m = 4$.

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) 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

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \quad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

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

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

Ex shows $n = 4$ and we just make x_0 as all 1's and

X of the matrix

y of the matrix.

Normal equation $= (X^T X)^{-1} X^T y$

No need to feature Scaling for normal equation

Gradient Descent	Vs	Normal equation
1. need to choose α		1. no need to compute α
2. needs many iteration		2. no need to iterate
3 $O(kn^2)$		3 $O(n^3)$, need to calculat $(X^T X)^{-1}$
4. works well with large n		4. slow if n is large
5. Choose if $n \geq 10^6$		5. Choose if $n \leq 10^5$

In future when we get to more Complex algorithms, like classification and logistic algo, The normal equation method donot work and we have to Choose Gradient descent. Very useful algo Gradient descent

Normal equation Noninvertibility

normal when a matrix can't be inverted if it is linearly dependeno (Singular).

In normal equation we do

$(X^T X)^{-1}$ ⇒ There are some common issue.

1. Redundent features.

Ex: size of ñ house in $feet^2$ x_1

size of ñ house in m^2 x_2

Then you will end of linearly dependent Matrix which can't be invertible. So delete one of ñ future.

2. $m \leq n$, if $m=10$ and $n=101$, meh it is too many features for m so we have to see and remove some of ñ unnecessary features or use "regularization" (will come in furthe)

in octave

$pinv(X^T X)$ ⇒ solve ñ issue inself hoy

@ when you compute $inv(X^T X)$ will return non invertible because you have to manually correct it.