

Soft computing



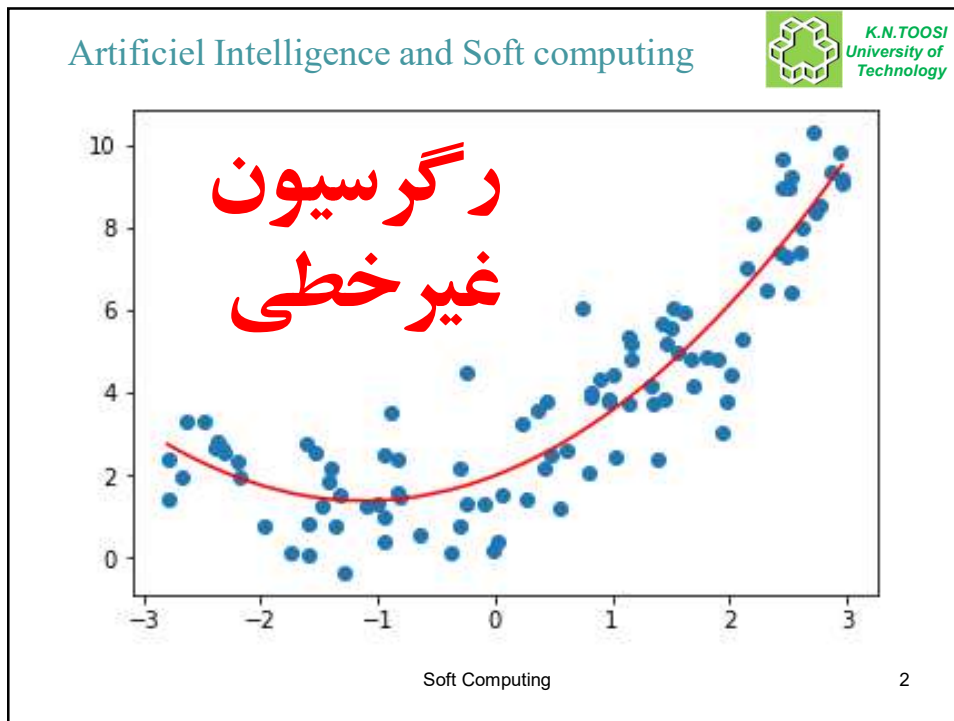
K.N. Toosi  
University of  
Technology

محاسبات نرم



Hasan Ghasemzadeh  
<http://wp.kntu.ac.ir/ghasemzadeh>

Soft Computing



## رگرسیون یک متغیره

آنالیز رگرسیون (واکاوی وایزشی)

One feature (Univariate).

Size (m <sup>2</sup> ) $x$	Price (Milliard Toman) $y$
210.4	46
141.6	23.2
153.4	31.5
85.2	17.8
...	...

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

رگرسیون خطی تک متغیره

$x$  متغیر

Soft Computing

3

## رگرسیون چند متغیره

Multiple features (variables). آنالیز رگرسیون (واکاوی وایزشی)

Size (m <sup>2</sup> ) $x_1$	Number of bedrooms $x_2$	Number of floors $x_3$	Age of home (years) $x_4$	Price (Milliard Toman) $y$
210.4	5	1	45	46
141.6	3	2	40	23.2
153.4	3	2	30	31.5
85.2	2	1	36	17.8
...	...	...	...	...

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

متغیر  $x_i$  یک بردار است

هر متغیر  $j$  نمونه یا مثال دارد که با آنها برنامه آموزش خواهد دید تا  $y$  را حدس بزند

Soft Computing

4

## رگرسیون چند متغیره

Multiple features (variables).

متغیرها →	Size (m <sup>2</sup> ) $x_1$	Number of bedrooms $x_2$	Number of floors $x_3$	Age of home (years) $x_4$	Price (Milliard Toman) $y$
	210.4	5	1	45	46
	141.6	3	2	40	23.2
مثال سوم → $x^3$	153.4	3	2	30	31.5
	85.2	2	1	36	17.8
	300	4	1	38	54
	289	5	3	20	60
Notation:	...	...	...	...	...

$n =$  number of features  $n = 4$  تعداد متغیرها  
 $x^{(i)}$  = input (features) of  $i^{th}$  training example.  $x^3 = [153.4 \ 3 \ 2 \ 30]^T$   
 $x_j^{(i)}$  = value of feature  $j$  in  $i^{th}$  training example.  $x_4^3 = 30$

Soft Computing  $y = [46 \ 23.2 \ 31.5 \ 17.8 \ 54 \ 60 \dots]^T$

نام متغیر در زیرنویس و شماره مثال در بالانویس می آید

## رگرسیون چند متغیره

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

$$h_{\theta}(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

For convenience of notation, define  $x_0 = 1$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in R^{n+1} \quad X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in R^{n+1} \quad h_{\theta} = \theta^T X = [\theta_0 \ \theta_1 \ \dots \ \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$h_{\theta} = \theta^T X = \sum_0^n \theta_n x_n$$

training set: design matrix  $X$

## رگرسیون چند متغیره

$$h_{\theta}(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n \quad h_{\theta} = \theta^T X = \sum_0^n \theta_n x_n$$

$$J(\theta) = \frac{1}{2} \sum_1^j (h_{\theta}(x^{(j)}) - y^{(j)})^2$$

تابع هزینه بایستی حداقل شود

مشتق گیری (در این درس مورد نیاز نیست)

$$\begin{aligned} \nabla_{\theta} J(\theta) &= \nabla_{\theta} \frac{1}{2} (X\theta - \bar{y})^T (X\theta - \bar{y}) \\ &= \frac{1}{2} \nabla_{\theta} (\theta^T X^T X \theta - \theta^T X^T \bar{y} - \bar{y}^T X \theta + \bar{y}^T \bar{y}) \\ &= \frac{1}{2} \nabla_{\theta} (\theta^T X^T X \theta - \theta^T X^T \bar{y} - \bar{y}^T X \theta + \bar{y}^T \bar{y}) \quad Ax = b \rightarrow A^T Ax = A^T b \Rightarrow x = (A^T A)^{-1} A^T b \\ &= \frac{1}{2} \nabla_{\theta} (\text{tr}(\theta^T X^T X \theta) - 2 \text{tr}(\bar{y}^T X \theta)) \\ &= \frac{1}{2} (X^T X \theta + X^T X \theta - 2 X^T \bar{y}) \\ &= X^T X \theta - X^T \bar{y} \end{aligned}$$

$$X^T X \theta = X^T y$$

مشتق برابر صفر

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

تعیین ضرایب رگرسیون

Soft Computing

7

## رگرسیون چند متغیره

Hypothesis:  $h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$

Parameters:  $\theta_0, \theta_1, \dots, \theta_n$   $x_0 = 1$

Cost function:

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

Gradient descent:

$$\text{Repeat } \left\{ \begin{array}{l} \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \dots, \theta_n) \\ \end{array} \right\}$$

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

Soft Computing

8

## رگرسیون چند متغیره

### Gradient Descent

Previously ( $n=1$ ):

Repeat {

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

$\underbrace{\hspace{10em}}_{\frac{\partial}{\partial \theta_0} J(\theta)}$

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

(simultaneously update  $\theta_0, \theta_1$ )

}

New algorithm ( $n \geq 1$ ) :

Repeat {

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

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

}

---


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

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

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

## مقیاس کردن متغیرها

### Feature Scaling

Idea: Make sure features are on a similar scale.

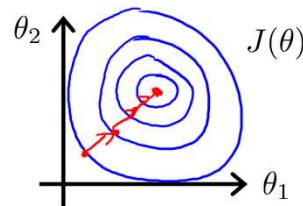
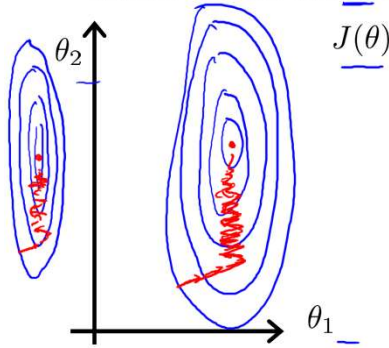
E.g.  $x_1 = \text{size (0-300 m}^2)$  ←

→  $x_1 = \text{size (m}^2)/300$  ←

$x_2 = \text{number of bedrooms (1-5)}$  ←

→  $x_2 = \frac{\text{number of bedrooms}}{5}$  ←

$0 \leq x_1 \leq 1$       $0 \leq x_2 \leq 1$



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

### Mean normalization

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

E.g.  $x_1 = (\text{size} - 150) / 300$

$$\text{Ave}(x_i) = 150$$

$$\max x_i - \min x_i = 300$$

$$x_2 = (\# \text{ bedrooms} - 2) / 4$$

$$-0.5 \leq x_1 \leq 0.5, -0.5 \leq x_2 \leq 0.5$$

$$x_i = (x_i - \text{Ave}(x_i)) / (\max x_i - \min x_i)$$

Soft Computing

11

## نرخ یادگیری

### Gradient descent

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

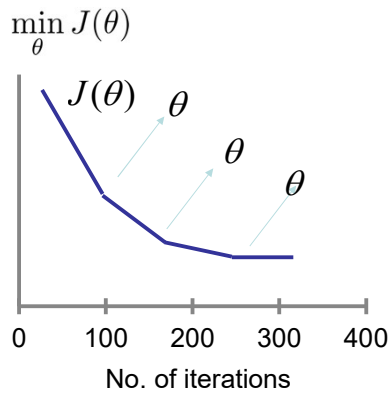
- “Debugging”: How to make sure gradient descent is working correctly.
- How to choose learning rate  $\alpha$ .

Soft Computing

12

## رگرسیون چند متغیره

Making sure gradient descent is working correctly.



Example automatic convergence test:

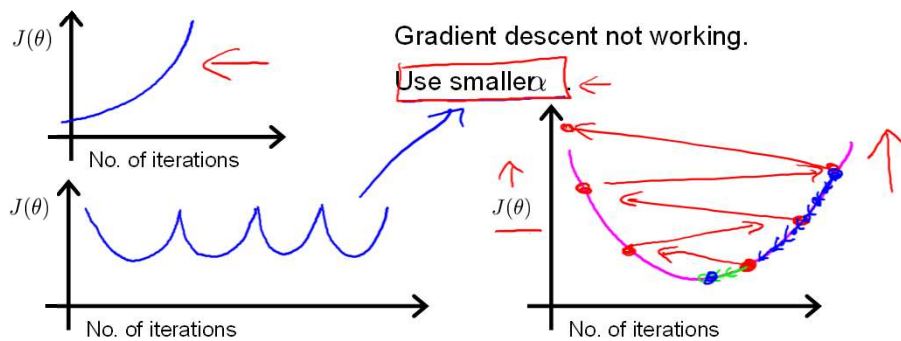
Declare convergence if  $J(\theta)$  decreases by less than  $10^{-3}$  in one iteration.

تابع هزینه پس از هر تکرار بایستی کاهش یابد  
Soft Computing

13

## رگرسیون چند متغیره

Making sure gradient descent is working correctly.



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

Soft Computing

14

## رگرسیون چند متغیره

### Summary:

- If  $\alpha$  is too small: slow convergence.
- If  $\alpha$  is too large:  $J(\theta)$  may not decrease on every iteration; may not converge.

To choose  $\alpha$  , try

..., 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, ...



Soft Computing

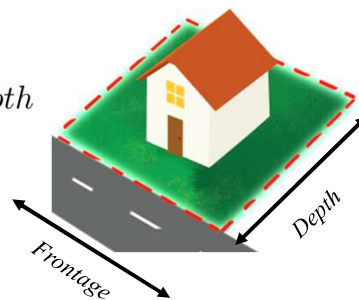
15

## رگرسیون چند متغیره

### Housing prices prediction

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

$$\text{Area} = \text{Frontage} * \text{Depth}$$



گاهی اوقات یک متغیر ممکن است به چند متغیر تبدیل شود و جواب بهتری از رگرسیون بدست آید

Soft Computing

16



### رگرسیون چند متغیره - معادله نرمال

Size (m <sup>2</sup> )	Number of bedrooms	Number of floors	Age of home (years)	Price (Milliard Toman)
$x_1$	$x_2$	$x_3$	$x_4$	$y$
210.4	5	1	45	46
141.6	3	2	40	23.2
153.4	3	2	30	31.5
85.2	2	1	36	17.8
300	4	1	38	54
289	5	3	20	60
...	...	...	...	...

$$X = \begin{bmatrix} 210.4 & 5 & 1 & 45 \\ 141.6 & 3 & 2 & 40 \\ 153.4 & 3 & 2 & 30 \\ 85.2 & 2 & 1 & 36 \\ 300 & 4 & 1 & 38 \\ 280 & 5 & 3 & 20 \end{bmatrix}$$

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

معادله نرمال

Soft Computing

$$y = \begin{bmatrix} 46 \\ 23.2 \\ 31.5 \\ 17.8 \\ 54 \\ 60 \end{bmatrix}$$

17

### رگرسیون چند متغیره - معادله نرمال

**Examples:**  $m = 5$ .

	Size (m <sup>2</sup> )	Number of bedrooms	Number of floors	Age of home (years)	Price (Milliard Toman)
$x_0$	$x_1$	$x_2$	$x_3$	$x_4$	$y$
1	210.4	5	1	45	46
1	141.6	3	2	40	23.2
1	153.4	3	2	30	31.5
1	85.2	2	1	36	17.8
1	300	4	1	38	54
1	289	5	3	20	60
...	...	...	...	...	...

$$X = \begin{bmatrix} 1 & 210.4 & 5 & 1 & 45 \\ 1 & 141.6 & 3 & 2 & 40 \\ 1 & 153.4 & 3 & 2 & 30 \\ 1 & 85.2 & 2 & 1 & 36 \\ 1 & 300 & 4 & 1 & 38 \\ 1 & 280 & 5 & 3 & 20 \end{bmatrix}$$

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

معادله نرمال

Soft Computing

$$y = \begin{bmatrix} 46 \\ 23.2 \\ 31.5 \\ 17.8 \\ 54 \\ 60 \end{bmatrix}$$

18

## رگرسیون چند متغیره - معادله نرمال

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

```
X = np.array([
    [1, 210.4, 5, 1, 45],
    [1, 141.6, 3, 2, 40],
    [1, 153.4, 3, 2, 30],
    [1, 85.2, 2, 1, 36],
    [1, 300, 4, 1, 38],
    [1, 280, 5, 3, 20] ])

XT X_transpose = X.T
XT [[ 1.  1.  1.  1.  1.  1. ]
     [210.4 141.6 153.4 85.2 300. 280. ]
     [ 5.  3.  3.  2.  4.  5. ]
     [ 1.  2.  2.  1.  1.  3. ]
     [ 45. 40. 30. 36. 38. 20. ]]

A = XT X
```

نام گذاری

Soft Computing

19

## رگرسیون چند متغیره - معادله نرمال

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

$(X^T X)^{-1}$  is inverse of matrix  $A = X^T X$

```
A = np.dot(X_transpose, X)
[[[6.0000000e+00 1.1706000e+03 2.2000000e+01 1.0000000e+01 2.0900000e+02]
 [1.1706000e+03 2.6350932e+05 4.7074000e+03 2.0256000e+03 3.9801200e+04]
 [2.2000000e+01 4.7074000e+03 8.8000000e+01 3.8000000e+01 7.5900000e+02]
 [1.0000000e+01 2.0256000e+03 3.8000000e+01 2.0000000e+01 3.1900000e+02]
 [2.0900000e+02 3.9801200e+04 7.5900000e+02 3.1900000e+02 7.6650000e+03]]

if np.linalg.det(A) != 0: # کنترل دترمینان ماتریس
    inverse_A = np.linalg.inv(A) # محاسبه معکوس ماتریس
    print(inverse_A)
else: print("ماتریس معکوس پذیر نیست")

(XTX)-1 [[ 2.52802649e+01 -2.28181124e-02 1.16806707e+00 -4.75589400e+00 -4.88560969e-01]
         [-2.28181124e-02 1.10911772e-04 -6.61431345e-03 4.63626218e-03 5.08266165e-04]
         [ 1.16806707e+00 -6.61431345e-03 5.44889272e-01 -3.82637813e-01 -3.55352772e-02]
         [-4.75589400e+00 4.63626218e-03 -3.82637813e-01 1.17990985e+00 9.43879852e-02]
         [-4.88560969e-01 5.08266165e-04 -3.55352772e-02 9.43879852e-02 1.04032808e-02]]
```

Soft Computing

20

## رگرسیون چند متغیره - معادله نرمال

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

```
C = np.dot(inverse_A, X_transpose)  C = (X^T X)^{-1} X^T
print(C)
[[-4.21468217e-01 -3.50080537e+00  1.11555059e+00  3.32820698e+00 -2.14111370e-01  6.92627391e-01]
 [-5.04560327e-03  2.64722510e-03 -1.12667765e-03 -3.66321222e-03  7.94854176e-03 -7.60273724e-04]
 [ 5.19136591e-01 -3.20538615e-01 -4.32347418e-02  3.23983133e-02 -3.69648226e-01  1.81886679e-01]
 [-2.66244322e-01  8.88026388e-01 -1.14557031e-03 -5.48282774e-01 -1.28913313e-01  5.65595910e-02]
 [ 3.23746747e-03  8.17108909e-02 -1.63243765e-02 -4.74211521e-02  1.14904274e-02 -3.26932572e-02]]
```

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

```
y_transpose
=np.array([46,23.2,31.5,17.8,54,60])
y=y_transpose.T
Tetha = np.dot(C,y)
print(Tetha)
```

```
 $\theta^T$  [23.77133468  0.11222718  6.61077944 -5.00828888 -0.65481055]
```

Soft Computing

21

## رگرسیون چند متغیره - معادله نرمال

```
 $\theta^T$  [23.77133468  0.11222718  6.61077944 -5.00828888 -0.65481055]
```

Hypothesis:  $h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$

مثال: قیمت یک منزل دوطبقه ده ساله با مساحت دویست متر و دارای سه

اتاق خواب چقدر است؟

$$h_{\theta}(x) = \theta^T x \quad x = \begin{bmatrix} 1 \\ 200 \\ 3 \\ 2 \\ 10 \end{bmatrix} \quad \Rightarrow h_{\theta}(x) = 49.48$$

```
x = np.array([[1],[200],[3],[2],[10]]) # can be written as below
# x = np.array([1,200,3,2,10]) # just in one dimension
hx = np.dot(Tetha,x)
print("h(x):",hx)
```

Soft Computing

22

## رگرسیون با تابع چند جمله ای

### Polynomial regression

رگرسیون غیر خطی مشابه رگرسیون چند متغیره است

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

Price (y)

Size (x)

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

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

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

Size (m <sup>2</sup> )	Number of bedrooms	Number of floors	Age of home (years)	Price (Milliard Toman)
$x_1 = 1$	$x_2 = x$	$x_3 = x^2$	$x_4 = x^3$	$y$
1	210.4	44268.2	9,314,020.8	46
1	141.6	20,050.6	2,839,159.3	23.2
1	153.4	23,531.6	3,609,741.3	31.5
1	85.2	7,259.0	618,470.2	17.8
1	300	90000	27000000	54
1	289	83521	24,137,569	60
...	...	...	...	...

23

## رگرسیون غیر خطی

### Choice of features

Price (y)

Size (x)

$$\rightarrow h_{\theta}(x) = \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2$$

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

Soft Computing

24

## رگرسیون چند متغیره با تابع چند جمله ای

### Polynomial regression

$(x + y + z + \dots)^n$  بسط توانی مرتبه  $n$   
معمولا حاصلضرب متغیرها با مجموع توان ۳ نیز در نظر گرفته می شود

$x, y, x^2, y^2, xy$  تابع چند جمله ای مرتبه ۲  
علاوه بر جملات مرتبه دو معمولا جملات مرتبه یک در نظر گرفته می شوند

$(x + y + z + \dots)^3$  ضرایب یک بسط توانی مرتبه ۳  
تابع چند جمله ای مرتبه ۳ دو متغیره شامل توانهای مساوی یا کمتر از ۳

$x, y, x^2, y^2, xy, x^3, y^3, x^2y, xy^2$

Soft Computing

25

## محاسبات زیاد تعداد ضرایب در رگرسیون

$(x + y + z + \dots)^n$  تعداد ضرایب یک بسط توانی شامل  $k$  جمله به توان  $n$

$$\binom{n+k-1}{k-1} = \frac{(n+k-1)!}{(k-1)!(n)!}$$

$$\binom{2+k-1}{k-1} = \frac{(k+1)!}{(k-1)!(2)!} = k(k+1)/2 \quad n=2 \text{ برای حالت کودراتیک}$$

$$5 * 6/2 = 15 \quad n=2 \text{ برای مساله قبلی با ۵ متغیر در حالت کودراتیک}$$

$$1000 * 1001/2 = 50500 \quad n=2 \text{ برای مساله قبلی با 1000 متغیر در حالت کودراتیک}$$

تعداد پارامترهایی که باید حساب شود به سرعت زیاد می شود

Soft Computing

26

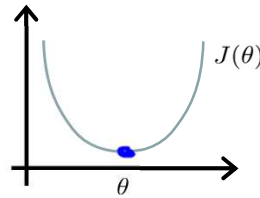
## رگرسیون غیر خطی

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

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

$\frac{\partial}{\partial \theta} J(\theta) = \dots \stackrel{\text{set}}{=} 0$

Solve for  $\theta$



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

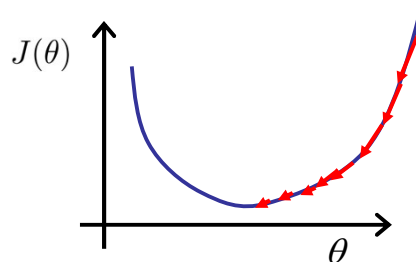
$$\frac{\partial}{\partial \theta_j} J(\theta) = \dots = 0 \quad j \text{ (for every )}$$

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

## رگرسیون غیر خطی - معادله نرمال

Gradient Descent

$$h_{\theta} = \theta^T X = \sum_0^n \theta_n x_n$$



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

Normal Equation

- No need to choose  $\alpha$ .
- Don't need to iterate.
- Need to compute  $(X^T X)^{-1}$
- Slow if  $n$  is very large.

$m$  training examples,  $n$  features.

Gradient Descent

- Need to choose  $\alpha$ .
- Needs many iterations.
- Works well even when  $n$  is large.

## دترمینان صفر و عدم وجود معکوس ماتریس

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

- Redundant features (linearly dependent).  
E.g.  $x_1 = \text{size in m}^2$   
 $x_2 = \text{size in feet}^2$
- Too many features (e.g.  $m \leq n$ ).
  - Delete some features, or use regularization.

Soft Computing

29

## رگرسیون چند متغیره - معادله نرمال

Rank of matrix

رتبه ماتریس برابر با کمترین مقدار بین بیشترین تعداد ستون‌ها یا سطرهایی که مستقل از هم هستند.

```
rank_X = np.linalg.matrix_rank(X)
print("Ranx X=", rank_X )
```

$$X = \begin{bmatrix} 1 & 210.4 & 5 & 1 & 45 \\ 1 & 141.6 & 3 & 2 & 40 \\ 1 & 153.4 & 3 & 2 & 30 \\ 1 & 85.2 & 2 & 1 & 36 \\ 1 & 300 & 4 & 1 & 38 \\ 1 & 280 & 5 & 3 & 20 \end{bmatrix}$$

Ranx x= 5

$$X = \begin{bmatrix} 1 & 210.4 & 5 & 1 & 45 & 2104 \\ 1 & 141.6 & 3 & 2 & 40 & 1416 \\ 1 & 153.4 & 3 & 2 & 30 & 1534 \\ 1 & 85.2 & 2 & 1 & 36 & 852 \\ 1 & 300 & 4 & 1 & 38 & 3000 \\ 1 & 280 & 5 & 3 & 20 & 2800 \end{bmatrix}$$

Ranx x= 5

Soft Computing

30

## رگرسیون چند متغیره - معادله نرمال

$(X^T X)^{-1}$  is inverse of matrix  $X^T X$

اگر ماتریس دارای دترمینان صفر باشد یا غیر مربعی باشد به جای معکوس از شبه معکوس (pseudo inverse) استفاده می شود. برای اینکار از دستور زیر در برنامه استفاده کنید.

```
import numpy as np
# Assuming a is your rectangular matrix
a = np.array([[...], [...]]) # your matrix
# Compute the pseudo-inverse of a
pseudo_inverse = np.linalg.pinv(a)
```

$$X = \begin{bmatrix} 2 & 3 \\ 4 & 6 \end{bmatrix} \quad \det(X) = 0 \quad \text{Pseudo Inverse}(X) = \begin{bmatrix} 0.0615 & 0.0308 \\ 0.0923 & 0.0462 \end{bmatrix}$$

Soft Computing

31

## رگرسیون چند متغیره - معادله نرمال

➤ در بسیاری از کاربردهای عملی، مانند پردازش داده‌ها، معمولاً تعداد معادلات بیشتر یا کمتر از تعداد متغیرها است. در این شرایط، شبه معکوس یک روش برای یافتن بهترین جواب با حداقل کردن خطای کمترین مربعات است.

➤ جواب‌هایی که از طریق شبه معکوس به دست می‌آیند، دقیقاً مانند حل معادلات با معکوس معمولی نیستند؛ زیرا شبه معکوس کمترین خطا را پیدا می‌کند اما نمی‌تواند شرایط دقیق را برآورده کند. در مواردی که داده‌ها نویزی یا دارای خطا هستند، این روش اغلب جواب قابل اعتمادی فراهم می‌کند.

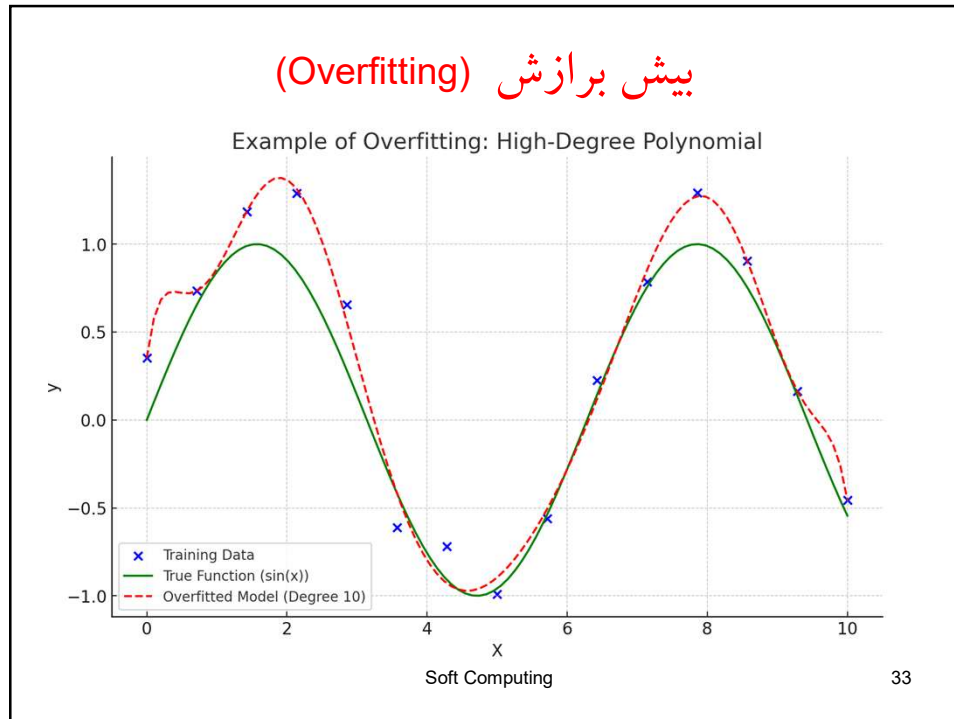
➤ در مواردی که ماتریس اصلی دارای دترمینان صفر باشد، استفاده از شبه معکوس تنها راه حل معادله خواهد بود که یک جواب تقریبی بهینه از نظر کمترین مربعات خطا ارائه می‌دهد.

➤ در شرایطی که تعداد معادلات کمتر از تعداد متغیرها باشد جواب‌های متعددی ممکن است وجود داشته باشد، و شبه معکوس یک جواب خاص از بین جواب‌های ممکن ارائه می‌دهد که بهینه است.

Soft Computing

32





33

### بیش برازش (Overfitting)

روش‌های جلوگیری از بیش برازش داده‌ها

#### تقسیم داده‌ها به مجموعه‌های آموزشی، ارزیابی و تست

این کار به مدل کمک می‌کند تا روی مجموعه تست عمومی‌سازی (generalization) بهتری داشته باشد. مجموعه ارزیابی برای تنظیم هایپرپارامترها و انتخاب بهترین مدل استفاده می‌شود، و مجموعه تست برای ارزیابی نهایی مدل به کار می‌رود.

#### ۲. استفاده از روش‌های منظم‌سازی (Regularization)

• روش‌های منظم‌سازی ریب و لسو به مدل کمک می‌کنند تا از یادگیری بیش از حد جزئیات و نوسانات نویز جلوگیری کنند. با افزودن جریمه‌ای به تابع هزینه که بزرگی وزن‌ها را کاهش می‌دهد، مدل ساده‌تر می‌شود.

#### ۳. کاهش تعداد ویژگی‌ها (Feature Selection)

• در داده‌های حجیم، معمولاً تعداد زیادی ویژگی وجود دارد. با استفاده از روش‌های کاهش ویژگی‌ها، مدل با ویژگی‌های غیرضروری دچار بیش‌برازش نمی‌شود.

• روش‌های دیگری نیز وجود دارد که در صورت نیاز می‌توانید در باره آنها تحقیق نمایید

Soft Computing

34

## نمونه رگرسیون مرتبه دو

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import root_mean_squared_error, r2_score
from mpl_toolkits.mplot3d import Axes3D

# Load the data
file_path = '3D_data.xlsx'
data = pd.read_excel(file_path)

# Extract X, Y, Z columns
X_data = data[['X', 'Y']]
Z_data = data['Z']

# Create polynomial features of degree 2
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X_data)

# Fit a linear regression model on the polynomial features
model = LinearRegression()
model.fit(X_poly, Z_data)

# Predict Z values using the model
Z_pred = model.predict(X_poly)

# Get coefficients for the polynomial equation
intercept = model.intercept_
coefficients = model.coef_
```

روش معادله نرمال

Soft Computing

35

## نمونه رگرسیون مرتبه دو

```
# Format the polynomial equation as a string
equation = (f"Z = {intercept:.2f} "
            f"+ ({coefficients[1]:.2f})*X "
            f"+ ({coefficients[2]:.2f})*Y "
            f"+ ({coefficients[3]:.2f})*X^2 "
            f"+ ({coefficients[4]:.2f})*X*Y "
            f"+ ({coefficients[5]:.2f})*Y^2")

# Create a mesh grid for X and Y for a smooth surface plot
x_range = np.linspace(X_data['X'].min(), X_data['X'].max(), 100)
y_range = np.linspace(X_data['Y'].min(), X_data['Y'].max(), 100)
x_mesh, y_mesh = np.meshgrid(x_range, y_range)

# Transform mesh grid data for prediction
X_mesh_poly = poly.transform(np.c_[x_mesh.ravel(),
                                   y_mesh.ravel()])
z_mesh_pred = model.predict(X_mesh_poly).reshape(x_mesh.shape)
```

Soft Computing

36

## نمونه رگرسیون مرتبه دو

```
# Plotting the 3D scatter and regression surface
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')
# Plot the original data points
ax.scatter(X_data['X'], X_data['Y'], Z_data, color='blue',
label="Original Data", alpha=0.5)
# Plot the regression surface
ax.plot_surface(x_mesh, y_mesh, z_mesh_pred, color='orange',
alpha=0.6, rstride=100, cstride=100)
# Add the polynomial equation to the plot
ax.text2D(0.05, 0.95, equation, transform=ax.transAxes,
fontSize=10, verticalalignment='top',
bbox=dict(boxstyle="round,pad=0.3", edgecolor="black",
facecolor="white"))
# Labels and title
ax.set_title("3D Polynomial Regression Surface")
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
plt.legend()
plt.show()
```

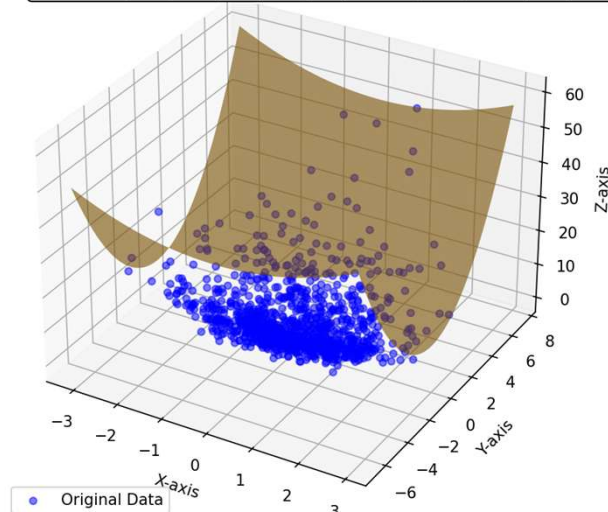
Soft Computing

37

## نمونه رگرسیون مرتبه دو

3D Polynomial Regression Surface

$$Z = 0.21 + (-0.02)*X + (0.02)*Y + (1.00)*X^2 + (-0.02)*X*Y + (0.90)*Y^2$$



38

## نمونه رگرسیون مرتبه دو - نزول گرادیان

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Load the data
file_path = '3D_data.xlsx'
data = pd.read_excel(file_path)

# Extract X, Y, Z columns
X_data = data[['X', 'Y']].values
Z_data = data['Z'].values

# Generate polynomial features of degree 2 manually
X_poly = np.c_[np.ones(X_data.shape[0]), X_data[:, 0],
               X_data[:, 1], X_data[:, 0]**2, X_data[:, 0]*X_data[:, 1],
               X_data[:, 1]**2]

# Initialize parameters for gradient descent
theta = np.random.randn(X_poly.shape[1])
learning_rate = 0.01
num_iterations = 10000

# Define the cost function
def compute_cost(X, y, theta):
    m = len(y)
    predictions = X @ theta
    cost = (1 / (2 * m)) * np.sum((predictions - y) ** 2)
    return cost
```

Soft Computing

39

## نمونه رگرسیون مرتبه دو - نزول گرادیان

```
# Define the gradient descent function
def gradient_descent(X, y, theta, learning_rate,
                    num_iterations):
    m = len(y)
    cost_history = []

    for i in range(num_iterations):
        # Calculate predictions
        predictions = X @ theta

        # Calculate the gradients
        gradients = (1 / m) * X.T @ (predictions - y)

        # Update the parameters
        theta -= learning_rate * gradients

        # Store the cost for this iteration
        cost = compute_cost(X, y, theta)
        cost_history.append(cost)

    # Optional: print cost every 1000 iterations for
    monitoring
    if i % 1000 == 0:
        print(f"Iteration {i}: Cost = {cost}")

    return theta, cost_history
```

Soft Computing

40

## نمونه رگرسیون مرتبه دو - نزول گرادیان

```
# Run gradient descent
theta, cost_history = gradient_descent(X_poly, Z_data,
theta, learning_rate, num_iterations)

# Display final parameters
intercept = theta[0]
equation = (f"Z = {intercept:.2f} "
            f"+ ({theta[1]:.2f})*X "
            f"+ ({theta[2]:.2f})*Y "
            f"+ ({theta[3]:.2f})*X^2 "
            f"+ ({theta[4]:.2f})*X*Y "
            f"+ ({theta[5]:.2f})*Y^2")
print("Equation:", equation)
# Plotting the 3D scatter and regression surface
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')
# Original data points
ax.scatter(X_data[:, 0], X_data[:, 1], Z_data, color='blue',
label="Original Data", alpha=0.5)
```

Soft Computing

41

## نمونه رگرسیون مرتبه دو - نزول گرادیان

```
# Create mesh grid for X and Y for the regression surface
x_range = np.linspace(X_data[:, 0].min(), X_data[:, 0].max(), 100)
y_range = np.linspace(X_data[:, 1].min(), X_data[:, 1].max(), 100)
x_mesh, y_mesh = np.meshgrid(x_range, y_range)
X_mesh_poly = np.c_[np.ones(x_mesh.ravel().shape[0]), x_mesh.ravel(),
y_mesh.ravel(), x_mesh.ravel()*2, x_mesh.ravel()*y_mesh.ravel(),
y_mesh.ravel()*2]
# Predict Z values for the grid
z_mesh_pred = X_mesh_poly @ theta
z_mesh_pred = z_mesh_pred.reshape(x_mesh.shape)
# Plot the regression surface
ax.plot_surface(x_mesh, y_mesh, z_mesh_pred, color='orange',
alpha=0.6, rstride=100, cstride=100)
# Add the polynomial equation to the plot
ax.text2D(0.05, 0.95, equation, transform=ax.transAxes, fontsize=10,
verticalalignment='top', bbox=dict(boxstyle="round,pad=0.3",
edgecolor="black", facecolor="white"))
# Labels and title
ax.set_title("3D Polynomial Regression Surface using Gradient
Descent")
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
plt.legend()
plt.show()
```

Soft Computing

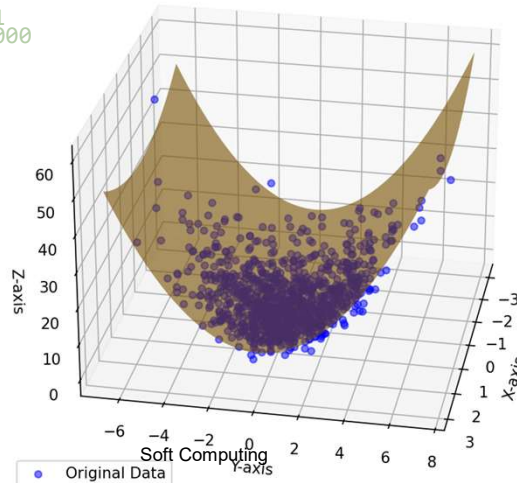
42

## نمونه رگرسیون مرتبه دو - نزول گرادیان

3D Polynomial Regression Surface using Gradient Descent

$$Z = 0.21 + (-0.02)*X + (0.02)*Y + (1.00)*X^2 + (-0.02)*X*Y + (0.90)*Y^2$$

learning\_rate = 0.01  
num\_iterations = 10000



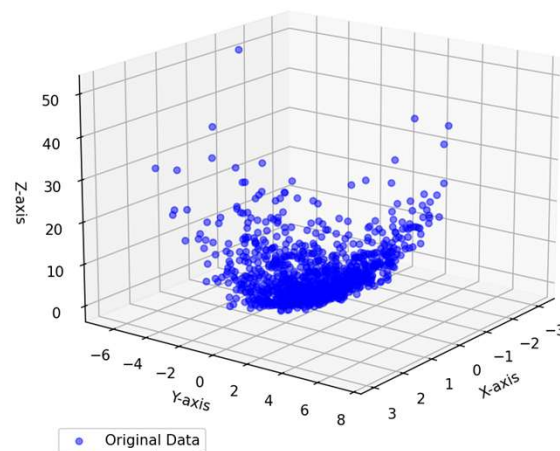
43

## نمونه رگرسیون مرتبه دو - نزول گرادیان

3D Polynomial Regression Surface using Gradient Descent

$$Z = \text{nan} + (\text{nan})*X + (\text{nan})*Y + (\text{nan})*X^2 + (\text{nan})*X*Y + (\text{nan})*Y^2$$

learning\_rate = 0.1  
num\_iterations = 10000



## تمرین برنامه نویسی

تمرین ششم: یک برنامه به زبان پایتون بنویسید که یک فایل داده را خوانده و رگرسیون درجه ۲ برای دو متغیر حساب نماید.

۱- به روش معادله نرمال

۲- با استفاده از نزول در راستای گرادیان

۳- مقایسه زمان دو روش حل فوق