Stanford CS229

Linear Regression

↳ Supervised learning.

• Goal is to fit a straight line

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

↳ Output

Input features

Ex:-

$h(x)$ cost	x_1 size	x_2 no. of rooms
1300\$	1600	3
1200\$	1700	4
600\$	1200	1
700\$	1300	2

$$h(x) = \sum_{j=0}^{2} \theta_j x_j$$

$$x_1 = 0$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \qquad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

↳ Parameters

* m = training examples
x = inputs / feature
y = output
(x, y) = training ex.
n = no of features

Goal of L.R (Linear regression)

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

$h_\theta x$:
h depends on θ & x

⇓

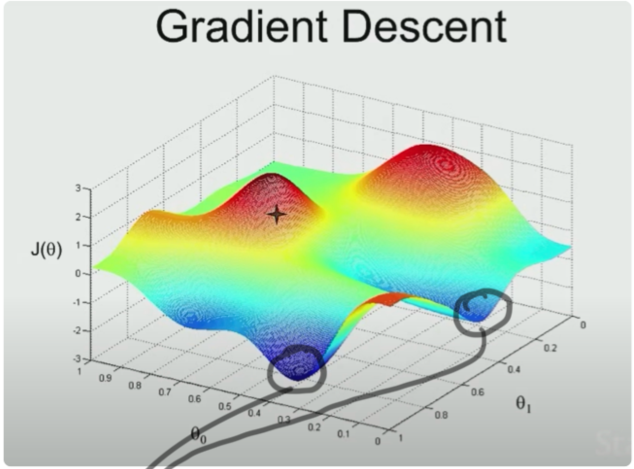
minimize $J(\theta)$

Gradient descent (Algorithm to minimize $J(\theta)$)

Algorithm :-

1) Start with som θ
2) change θ to reduce $J(\theta)$

- Start at some point
- Take 360° look & figure out which direction baby step leads to lowest value
- Repeat



Lowest point

→ here θ_2 is not included to make a 3-D diagram

* Slope ⇒ Give rate of change
Derivative ⇒ Slope
-Derivative ⇒ -ve slope which means the rate of lowest descent.

'a = b'
↳ expression
a := a+1
↳ assignment

Ex:- $f(x, y)$ ⇒ here unlike slope the function doesn't change in one direction like the above fig. We use gradient instead of slope (a vector that points in the direction of greatest rate of increase of the function

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

If the gradient vector at $(2,3)$ is $\nabla f = (4,5)$, it means:

- Moving in the direction where x increases by 4 units and y increases by 5 units will make you climb the hill the fastest.

$x := x - \alpha \, \partial J(\theta)$] Gradient descent algo

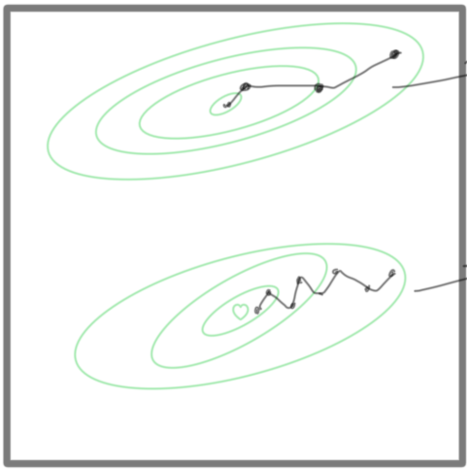
∂x

$f(x, y)$

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

↳ Learning rate

↳ the x & y we obtain gives the steepest descent which we keep repeating until we obtain a minim for x, y



high α (might overshoot)

low α (better but more iterations)

How to find Local Minima? $\Rightarrow \frac{d}{dx} = 0$

$\nabla_\theta J(\theta) = 0$

trace (A) = Sum of all diagonal elements of a matrix
$tr(A) = tr(A)^T$