* Machine Learning *

→ Doing tasks without being programmed explicitly.

→ Experience E is obtained by doing task T, whose performance is measured by P.
Measure of Performance P in doing a task T, increases with ~~task~~ experience E

ML

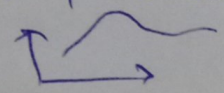
Supervised
↳ Learn and enhance your P using training dataset and later perform as per desired objective.

Unsupervised.
↳ Just club similar items/data without any objective.
↳ Youtube recommendations
↳ Seperating audio of two people.

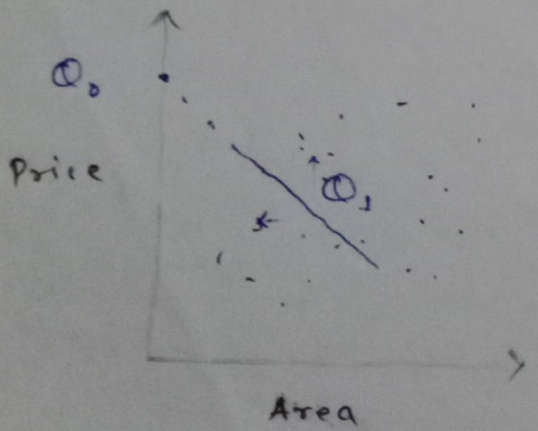
ML

Classification
0 / 1

Regression



* LINEAR REGRESSION *



1) Assumming linear relationship
This we will learn later

Find:
best $h: A \to P$
such that the squared error is min
overage

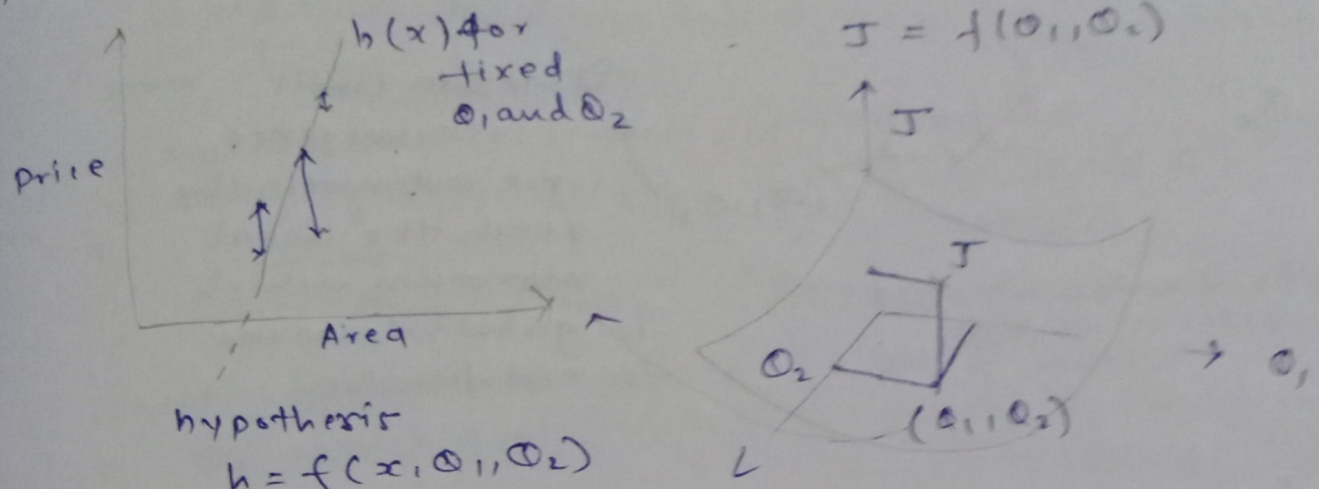
2) Error - squared function: $J(\theta_0, \theta_1)$

$$: \frac{1}{2m} \sum_{0}^{m} (h(x) - y)^2$$

* Gradient Descent *

↳ It's about finding local minima.

* Understand this condition



we need such θ_1 and θ_2 that J is min.

* Understand this equation.

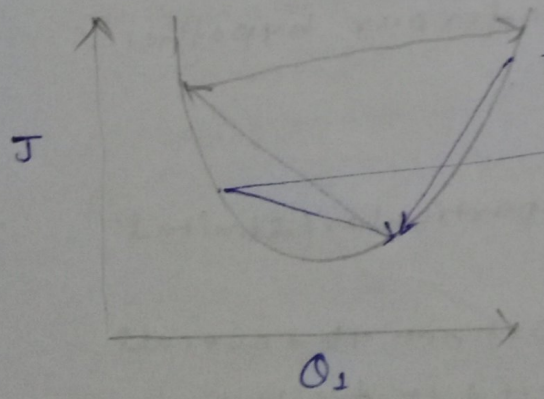
learning rate.

$$\left.\begin{aligned} \theta_1 &= \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1, \theta_2) \\ \theta_2 &= \theta_2 - \alpha \frac{\partial}{\partial \theta_2} J(\theta_1, \theta_2) \end{aligned}\right\} \text{Do until convergence}$$

This is incorrect
do simultaneous update.

* Understand the learning rate.

$J = f(\theta_1)$ suppose $\Rightarrow h = \theta_1 x$



if $\frac{\partial J}{\partial \theta}$ = +ve * +ve α $\}$ θ_1 decrease

$\frac{\partial J}{\partial \theta}$ = -ve * +ve α $\}$ θ_1 increase

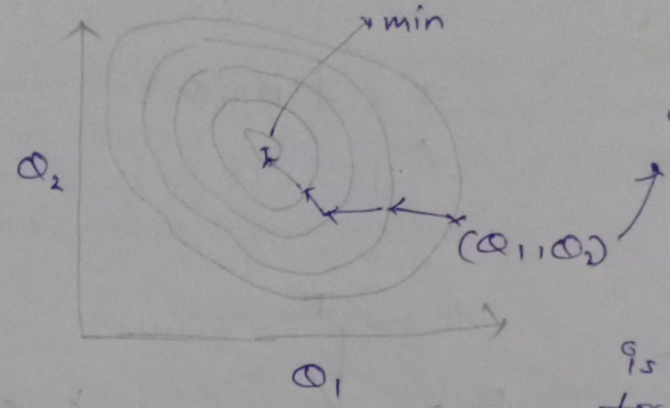
Be careful about
α being too large.

15/07/2020

Back to normal:

$J = f(\theta_1, \theta_2)$

Countour plot



if we run until convergence we automatically reach the local minimum which is global minimum for cost function

Conclusion

$h: A \to P \Rightarrow \theta_0 + \theta_1 x$

$J(\theta_1, \theta_2) = \frac{1}{2m} \sum_1^m (h^2 - y)^2$

$\Rightarrow$ find $\theta_0, \theta_1 \Rightarrow$ s.t $J = \min J(\theta_0, \theta_1)$

Gradient descent

Repeat until convergence {

$temp1 := \theta_1 - \alpha \frac{\partial J}{\partial \theta_1}$

$temp2 := \theta_2 - \alpha \frac{\partial J}{\partial \theta_2}$.

$\theta_1 = temp1$

$\theta_2 = temp2$ }

$\Rightarrow$ Output θ_1, θ_2 is fit line for our hypothesis - this is the best fit.

Batch Gradient Descent:

$\Rightarrow$ finding best fit line for particular/limited spectrum on x line.

Here $J_1 + J_2 + \ldots J_n$ } Defines the overall graph with $J_{1 \ldots n}$ being best fits for their respective domains

$J_1 = \frac{1}{2n_1} \sum_1^{n_1} (h - y)^2$

$J_2 = \frac{1}{2n_2} \sum_{n_1}^{n_2} (h - y)^2$

} Rest of the procedure is the same

* Linear Algebra *

Properties of Matrix Multiplication:

1. $A \times B \neq B \times A$ } Not always but may be.
2. $(A \times B) \times C = A \times (B \times C)$
3. $I \times A = A \times I = A$.
   - I: $m \times m$
   - A: $m \times n$
   - A: $m \times r$
   - I: $n \times n$

For square matrices inverse doesn't exist:

4. $A(A^{-1}) = A^{-1}A = I$ } Not for all A.
   where $A^{-1}_{LHS} = A^{-1}_{RHS}$

Note: What actually is an inverse of matrix.

Transpose of a matrix:

5. $B_{ji} = A^T = A_{ij}$

---

* Multidimensional linear regression *

Earlier: h: Area → Price

Now: h: Bedroomt Area + floors + Age → Price.

Nomenclature:

$x_n^{(i)}$ → i^{th} training set element

→ n^{th} attribute of that training set

$$h = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 \cdots \theta_n x_n. \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \end{bmatrix}$$

($x_0 = 1$)

$$\therefore \boxed{h = \theta^T * X}$$

$$X = \begin{bmatrix} x_0 \to 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

> Intuition:

θ_0 → base price

θ_1 → proportionality wrt no-of-bedrooms

θ_2 → prop. wrt. no-of-floors

⋮ ⋮

* Multivariate loss function *

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

$x^{(i)}$ → Training set

specially this don't confuse with $\theta_0 x_0 + \theta_1 x_1 \ldots$

Training set — Parameter.

1	$x_0^1, x_1^1, x_2^1, x_3^1$
2	$x_0^2, x_1^2, x_2^2, x_3^2$

$$\therefore \quad J(\theta) = \frac{1}{2m} \sum_{1}^{m} \left(h(x^{(i)}) - y^{(i)} \right)^2$$

$$= \frac{1}{2m} \sum_{1}^{m} \left(\theta^T x^{(i)} - y^{(i)} \right)^2$$

* Gradient Descent function *

Earlier.

$$\theta_k = \theta_k - \frac{\alpha}{2} \frac{\partial}{\partial \theta_k} J(\theta_0, \theta_1, \theta_2 \ldots)$$

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

$$= \theta_k - \frac{\alpha}{m} \sum_{1}^{m} \left(\theta^T x^{(i)} - y^{(i)} \right) x_k^{(i)}$$

* How to make Gradient Descent work faster in action *

Suppose :

(a) Parameter 1 {size} $\in (0, 2000)$

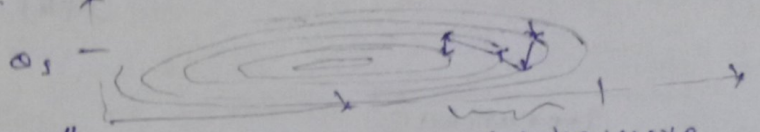
(b) Parameter 2 {bed} $\in (0, 2)$

Price = $\theta_0 + \theta_1 S + \theta_2 B$ → <<<

assume 0 ; θ_1 >>> ; θ_2 >>>

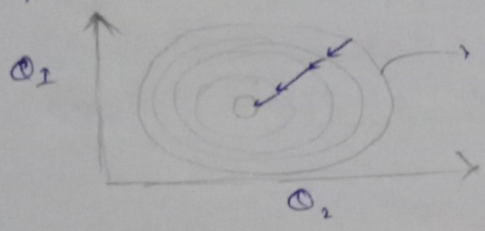
θ_1 wrt θ_2

(c) ∴ on θ_1 θ_2 axis <<<



But somehow ; will take more time to converge

But if θ_1 and θ_2 are of same order



→ Directly along radius so it saves time.

So you understood need for "feature Scaling"

Now lets work with normalisation :

Befor that :

for Price = $\theta_1 S + \theta_2 B$

↓ ↙ why these two terms should nearly be of same order?

Because their existence matter that is why we have included them

Easier way to digest scaling :

$$Price = \left(\theta_1 * 2 * 10^5\right)\left(\frac{S}{2 * 10^5}\right) + \dots$$

θ_1 ; S'

* Why normalisation ?

From where {which point} to begin gradient descent calculation.....

{ let Love (0, 0, 0, 0, ...) }

$$Para = \frac{Para - Mmean}{range\ (max - min)}$$

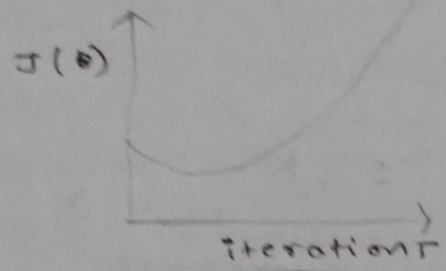
We can have std deviation also.

* Learning Rate *

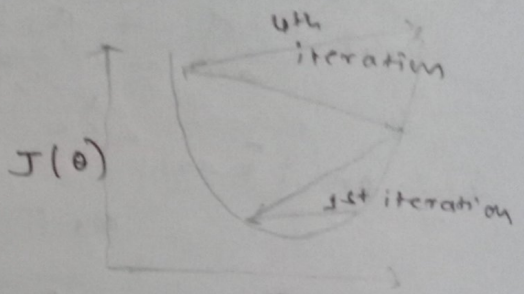
what should be learning rate :-

(A) How to make sure that gradient descent is working properly?

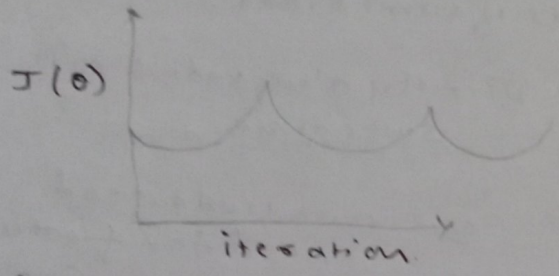
(i) α >>>



Because →

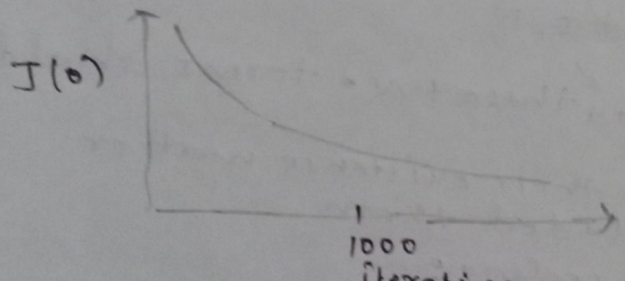


(ii) α >>>

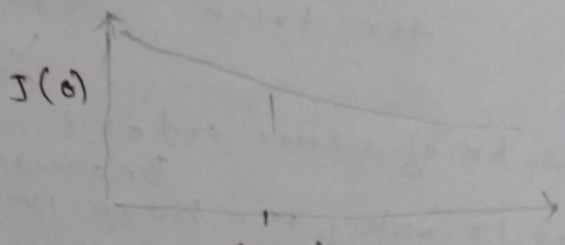


(iii) α is perfect :

" For α {sufficiently small λ} J(θ) should decrease with every iteration



perfect α is largest correct α.



α very small very

Mathematicians were pissed off and hence they couldn't derive a method for perfect α hence they said,

Try.

0.001 . 0.01 . 0.1 . . . 1 . . . } And check J(θ) vs iterations graph.

0.003 0.03

Pick the largest possible value.

* What feature to take *?

$h: A \rightarrow P$ or h: width, length $\rightarrow P$

* Polynomial regression *

Let hypothesis be.

h: size $\rightarrow P = \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2$

$\downarrow x_1 \qquad \downarrow x_2$

'size^2

'size^3

Then behave with $(\text{size})^2$ as a seperate entity, and normalise or feature scale for independently.

Ex. let size $\in (0, 1000)$

and perfect fit be ~~parabola~~ $\cap$

$\therefore h: \theta_0 + \theta_1 \text{size} + \theta_2 \sqrt{\text{size}}$

$\downarrow x_1 \qquad \downarrow x_2$

$\therefore x_1 \leftarrow \dfrac{\text{size}/x_1 - 500}{1000}$

$x_2 \leftarrow \dfrac{\sqrt{\text{size}}\ x_2 - M}{32}$

} This is how.

* How we do it in a vectorised manner *

$$X = \begin{bmatrix} x_0^1 & x_1^1 & \cdots & x_n^1 \\ x_0^2 & & & \\ \vdots & & & \end{bmatrix} \quad \begin{matrix} \leftarrow \text{number of features} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{No of training sed} \\ m \end{matrix}$$

$m \times n$

$$Y = \begin{bmatrix} y^1 \\ y^2 \\ y^3 \\ \vdots \end{bmatrix} \} m \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} n$$

For linear regression θ or polynomial regression mapped to linear regression

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

use for $n < 10{,}000$

* Normal equation for Linear Regression

~ No feature scaling required.
No iterations straight answer

Intuition

$v = y - \theta^T x$

We want to minimise $(v \cdot v^T)$ } that pt.