Week - 2

⇒ Notations :-

n = number of features
x^i = input (features) of i^{th} training example
x^i_j = value of feature j in i^{th} example
m = no of rows in the training set
y = output of training set.

⇒ Multivariate linear regression

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_{(n+1)\times 1} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}_{(n+1)*1} \in \mathbb{R}^{n+1}$$

⇒

$$h(x) = \theta_0 \overset{1}{x_0} + \theta_1 x_1 + \theta_2 x_2 \dots \theta_n x_n$$
$$= \boxed{\theta^T x}$$

⇒ Cost f^n ⇒

$$J(\theta_0, \theta_1, \theta_2 \dots \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i\right)^2$$

$$\Downarrow$$

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

OR

$$\frac{1}{2m} \sum_{i=1}^{m} \left(\sum_{j=1}^{n} \theta_j x_j^i - y^i\right)^2$$

⇒ Gradient Descent $(n \geq 1)$ ⟶

Repeat {

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

}

⇒ **Feature Scaling** :- Make sure features are on a similar scale (have similar value ranges) then the Gradient Descent reduces quickly.

(divide the feature by its upper bound or (range)
⇒ get in range of $-1 \leq x_i \leq 1$

$$x_1 = \frac{\text{size}}{(\text{max} - \text{min})}$$

⇒ **Mean normalization** :→ Replace x_i with $x_i - \mu_i$ (avg value) to make features have approximately zero mean

Eg → $x_1 = \frac{\text{size} - 1000}{2000}$, $-0.5 \leq x_1 \leq 0.5$

(size of house) → (max − min)

⇒ The use of gradien descent is to find a value of θ that minimises the cost f^n → $J(\theta)$

Finding the right α

$\Rightarrow$ If α is too small :$\rightarrow$ slow convergence
$\Rightarrow$ If α is too large $\rightarrow$ $J(\theta)$ may not decrease on every iteration, may not converge

To ~~fin~~ help choose the suitable α, plot a graph of $J(\theta)$ (Y) vs no of iterations (X axis)

$\Rightarrow$ POLYNOMIAL Regression

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

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

OR

$\Rightarrow \theta_0 + \theta_1(\text{size}) + \theta_2(\sqrt{\text{size}})$

that means $\Rightarrow$ $x_2 = (x_1)^2$ or $\sqrt{x_1}$ or $(x_1)^3$

$\Rightarrow$ For linear regression, if we solve the ~~coll~~ ~~f^n (J(\theta))~~ $h(\theta) = \theta_0 + \theta_1 x$ for the value of θ, we get

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

X & y are matrices.

In this method, no need for feature scaling

$\Rightarrow$ $h_\theta(x) = \theta_0 + \theta_1 x \longrightarrow$ hypothesis f^n

⇒ If n is large, use gradient descent
(if $n > 10{,}000$) ⇒ Time ⇒ $O(kn^2)$

⇒ If n is small, use Normal equation
Time ⇒ $O(n^3)$

⇒ Sometime, $(X^T X)$ can be non-invertible

↳ Reasons ⇒

i) = two features have a relation
ie $x_1 = k x_2$ (linearly dependent).

ii) = too many features ($m \leq n$).

<u>Solutions</u> ⇒ Delete some features or regularization

⇒ Still the octave f^n → pinv(X' * X) will give the right value of θ
(even if $(X^T \cdot X)$ is non-invertible)

Scanned with CamScanner

$a^T b = b^T a \rightarrow$ if a & b are vectors

OCTAVE TUTORIAL

$\Rightarrow$ ~= is $\Rightarrow$ means not equal to

$\Rightarrow$; semicolon suppress the output.

$\Rightarrow$ sprintf() $\rightarrow$ used for printing.

$\Rightarrow$ Syntax is same as C language & MATLAB

$\Rightarrow$ disp() $\rightarrow$ used to display output on terminal

$\Rightarrow$ format long $\longrightarrow$ changes the variable type to long

$\Rightarrow$ V = 1 : 0.1 : 2

V generates a ~~m~~ vector having values ~~starting~~ from 1 to 2 with intervals of 0.1

$\Rightarrow$ $C = \text{ones}(2,3) \rightarrow C = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

$\Rightarrow$ $W = \text{zeros}(2,3) \rightarrow W = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

$\Rightarrow$ rand (1,3) $\rightarrow$ random values less than 1

$\Rightarrow$ randn (1,3) $\rightarrow$ gives gaussian random values

$\Rightarrow$ hist (W) $\rightarrow$ plots a histogram

$\Rightarrow$ $\text{eye}(2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $\rightarrow$ generates an ~~Iden~~ identity matrix

$\Rightarrow$ help $\rightarrow$ brings out documentation

Scanned with CamScanner

⇒ size(A) → gives size of matrix
size(A,1) → gives rows.
size (A,2) → gives columns

⇒ length (A) = gives size of longer dimension

⇒ who → shows variables in current workspace
whos → gives detailed ~~view~~ view

⇒ ~~cl~~ clear A → clears variable A from workspace
clear → deletes all variables

⇒ save file.mat A → saves variable A in file.mat.

⇒ save hello.txt A -ascii → saves as text

⇒ A(3,2) → fetches the 6th element of A

⇒ A(2,:) → fetch everything from 2nd row

⇒ A([1 3],:) → fetch everything from 1st & 3rd row

⇒ A = [A, [100; 110; 120]] ⇒ appends another row to A.

⇒ A(:) → puts all elements of A in single vector

⇒ C = [A B] → concatenates 2 matrices (horizontally)

⇒ C = [A ; B] ⇒ concatenates vertically.

⇒ A .* B → multiplies every element of A with their corresponding element in B.

⇒ A .^ 2 → gives element wise squaring

⇒ 1 ./V ⇒ gives reciprocal of elements of V

⇒ log(V) ⇒ element wise log
exp(V) = " " exponent
abs (V) = absolute value

⇒ [val, ind] = max(a) → gives value & index of max element of A

⇒ find (a<3) → gives elements of A, less than 3

⇒ A = magic(3) → A magic matrix, has all sum of rows & columns equal (even the diagonals add to same sum)

⇒ sum (a) & prod(a) → gives sum of all elements as same for product.

⇒ max(max(A)) → gives max element of A
max(A) → gives max elements of every row

⇒ sum (A,1) → gives sum of every element in raw
sum (A,2) → in colum
Sum (sum(A.* eye(9))) → gives sum of diagonal.
sum(sum(A.* flipud(eye(9))) → other diag sum

⇒ flipud (A) → flips a matrix (mirror image)

⇒ pinv(A) → inverse of A

⇒ plot (X, Y) → plots a graph

⇒ hold on → makes all changes on same plot

⇒ xlabel ('...') & ylabel ('...') → to label axis.

⇒ print -dpng 'file.png' → to save a plot

⇒ figure(1); plot(t, y1); → helps plots more than 1 plot at a time.

⇒ axis ([0.5 1 -1 1]) → changes axis.

⇒

```
for i = 1:10,
    v(i) = 2^i;
end;
```

```
i = 1;
while i <= 5,
    v(i) = 100;
    i = i+1
end;
```

⇒ addpath ('c....') → add the path to octave search.

⇒ Octave f^n can return multiple values.

⇒ <u>Vectorization</u> ⟶ $h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$

$= \theta^T x$