Machine Learning.

Supervised learning !- In supervised learning we have give dataset to learn machine

① Regression !- In Regression we predict a no infinitely many possible outputs eg!- house price

② Classification!- In Classification predict Categories small no of possible outputs eg!- benign OR malignant, Cat or dog, 0 or 1

Unsupervised learning!- In unsupervised learning we don't have dataset to learn machine here machine find something in unlabeled data and learn

yha input milte h output already not given

yha ham tumor size, patient age de deate par ye nahi batate ki vo malignant h ya benign vo machine predict karti h

① Clustring :- Group similar data points together

eg!- Google news :- It find some key words in news and cluster all thoes news

DNA microarry !- cluster all DNA with some (genes) specification eg!- tall

Here we find Anomaly detection :- find unusual data points

Dimensionality reduction !- Compress data using fewer no

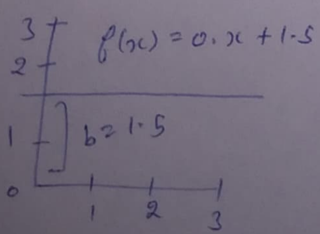
Linear Regression with one variable (univariate linear Regression)

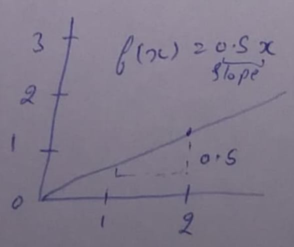
$\boxed{f_{w,b}(x) = wx + b}$

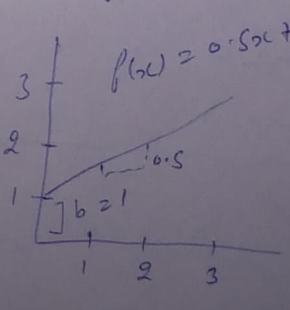
w, b = Cofficients weights

y = output variable

x = input variable feature







Cost function : Squared error cost function

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

m = no of training examples

$\hat{y}^{(i)}$ = hypothesis pridicted value

$y^{(i)}$ = give data set prediction (output variable)

$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$

model predication

$$\boxed{J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2}$$

Here our goal is to minimize J(w,b)

firstly start with some w, b (set w=0, b=0)

then keep changing w, b to reduce J(w,b)

until we settle at or near a minimum ye ham jab tak karenge jab tak cost function ki value min aate rahe previous value se

Note!- In this we may have more that 1 minimum in some cases

Gradient descent algorithm

partial derivative

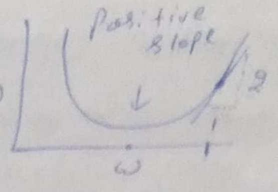
$$\boxed{\begin{aligned} temp\_w &= w - \alpha \frac{\partial}{\partial w} J(w,b) \\ temp\_b &= b - \alpha \frac{\partial}{\partial b} J(w,b) \\ w &= temp\_w \\ b &= temp\_b \end{aligned}}$$

α = learning rate

take it as small as you could

$w = w - \alpha$ (positive no)

-ve value get smaller value of w

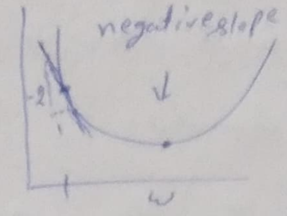


height / width = slope

$w = w - \alpha$ (negative no)

+ve value

get larger value of w



Gradiend descent algorithm

$$\text{temp\_w} = w - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)x^{(i)}\right]$$

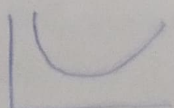
$$\text{temp\_b} = b - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)\right]$$

$w = \text{temp\_w}$

$b = \text{temp\_b}$

Note: In temp_w $x^{(i)}$ is the data (feature) that we have given like house size, house age etc.

Note:- In linear regression squared error cost function always end up with a bow shape or hammock shape



Note :- If $w = w - \alpha[0]$

$w = w$ so stop here and this is our value of w

Step 1 :- Phale line draw karne h $y = wx + b$

~~Step~~ Initially take w & b as 0

Step 2 :- find cost function

Step 3 :- update w and b by using gradient descent

Step 4 :- find cost function

if cost is smaller than previous cost so do these steps again and again

if cost is larger than previous cost then our value of w is previous w

if when finding ~~new~~ w from gradient descent the new w is same as previous w then this w is our ans

Linear Regression with Multiple Variables

Here we have multiple features

x_1 = first feature

x_2 = second feature

$\vdots$

x_n , , , , , , ,

$x_1^{(i)}$ = feature of ith training example (horizontally value $x_1 x_2 x_3$)

$x_2^{(i)}$ =

$\vdots$

$x_n^{(i)}$, , , , , , , , ,

n = no of feature

$x_1^{(i)}$ = value of feature 1 in ith training example ($x_1 x_2 x_3$)

$x_2^{(i)}$ = " " " 2 " " " "

$\vdots$

$x_n^{(i)}$ = , , , , , , , , , , ,

eg:- $\vec{x}^{(2)} = [1416 \quad 3 \quad 2 \quad 40]$

here $X_1 = 1416$ $\quad X_1^{(2)} = 1416$

$X_2 = 3$ $\quad x_2^{(1)} = 3$

$X_3 = 2$ $\quad x_3^{(2)} = 2$

$X_4 = 40$ $\quad x_4^{(3)} = 40$

$$\boxed{f_{w,b}(x) = w_1 x_1 + w_2 x_2 + \cdots\cdots w_n x_n + b}$$

It is multiple linear regression not multivariate regression

make a vector of all w $\vec{w} = [w_1\ w_2\ w_3 \cdots w_n]$

b is a no not a vector

vectorization

x $\vec{x} = [x_1\ x_2\ x_3 \cdots x_n]$

$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$

f = np.dot(w, x) + b

Cost function = $J_{w,b} = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{w,b}(\vec{x}^{(i)}) - y^{(i)}\right)^2$

Gradient descent

$$temp\_w = w - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) x_1^{(i)} \right]$$

$$temp\_b = b - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) \right]$$

$w = temp\_w$

$b = temp\_b$

Note:- Normal equation

- only for linear regression
- Solve for w, b without iterations

Disadvantages

- Doesn't generalize to other learning algorithms
- Slow when no of features is large (>10,000)

what you need to know

- Normal equation method may be used in machie learning libraries that implement linear regression
- Gradient descent is the recommended method for finding parameters w, b

eg:-	size	bedroom	Price Ans
	x_1	x_2	y
	300	0	
	⋮	⋮	⋮
	2000	5	$500k

$w_1 = 50$ $w_2 = 0.1$ $b = 50$
large small

price = $50 \times 2000 + 0.1 \times 5 + 50$

price = $1000050.5

$w_1 = 0.1$ $w_2 = 50$ $b = 50$
small large

price = $0.1 \times 2000k + 50 \times 5 + 50$

price = $500k

this is more reasonable

Here problem is to find the size of the parameters w_1, w_2

Note:- In this we notice that when a possible range of values of feature is large (300 - 2000) value of w is small
and when possible range of values of feature is small (0-5) value of w is large.

if ranges are diff than gradient descent give value slowly but when ranges are in same than it give value fastly

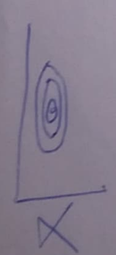
In ~~Linear~~ Linear regression we use some normalization

1. Feature scaling

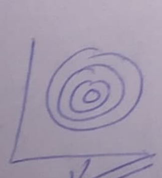
$\boxed{\min \le x_1 \le \max \qquad x_{1,scaled} = \frac{x_1}{\max}}$

$300 \le x_1 \le 2000$ $0 \le x_2 \le 5$

without scaling



with scalling



$x_{1,scaled} = \frac{x_1}{2000}$ $x_{2\,scaled} = \frac{x_2}{5}$

$0.15 \le x_{1,scaled} \le 1$ $0 \le x_{2,scaled} \le 1$

2. mean normalization

$$\boxed{\begin{array}{l} \min \le x_1 \le \max \\ x_{1\,normalized} = \frac{x_1 - mean}{\max - \min} \end{array}}$$

$300 \le x_1 \le 2000$

$0 \le x_2 \le 5$

x_1 normlized $= \dfrac{x_1 - \mu_1}{2000-300}$

x_2 normalized $= \dfrac{x_2 - \mu_2}{5-0}$

$-0.18 \le x_{1\,\text{normalized}} \le 0.82$

$-0.46 \le x_{2\,\text{normalized}} \le 0.54$

3. Z-score normalization

$$\boxed{\begin{gathered} min \le x_1 \le max \\ x_1 \text{ normalized} = \frac{x_1 - \text{mean}}{\text{standard deviation}} = \frac{x_1 - \mu_1}{\sigma_1} \end{gathered}}$$

$300 \le x_1 \le 2000$

$0 \le x_2 \le 5$

x_1 normalized $= \dfrac{x_1 - \mu_1}{\sigma_1}$

x_2 normalized $= \dfrac{x_2 - \mu_2}{\sigma_2}$

$-0.67 \le x_1$ normalized ≤ 3.1

$-1.6 \le x_2$ normalized ≤ 1.9

$\sigma \rightarrow$ standard deviation

Note:- here our motive is to normalize it as small as if after normalize it also big than again normalize

$0 \le x_1 \le 3$	ok
$-2 \le x_2 \le 0.5$	ok
$-100 \le x_3 \le 100$	too large $\rightarrow$ rescale it
$-0.001 \le x_4 \le 0.001$	too small $\rightarrow$ rescale it
$98.6 \le x_5 \le 105$	too large $\rightarrow$ rescale it

Choosing the learning rate

Values of α to try

... 0.001 → 0.003 →
3x ≅3x

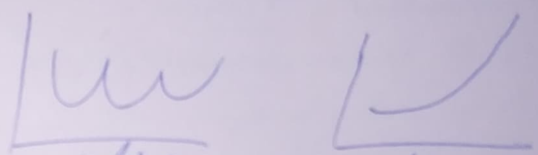
0.01 → 0.03 →
3x ≅3x

0.1 → 0.3 →
3x ≅3x

1 → 3 →
3x ≅3x

Debuggin tip

with a small enough α, $J(\vec{w}, b)$ should decrease on every iteration so if gradiend descent isn't working so just set Alpha to be very small no and see if that causes the cost to descrease on every iteration. if even with Alpha set to a very small no J doesn't decrease on every single iteration but instead sometimes increases then that usually means there's a bug somewhere in the code



before after

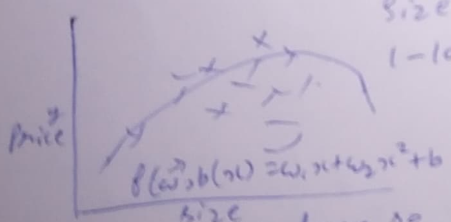
next page

Polynomial regression

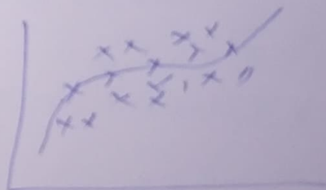
$$f_{w,b}(\vec{x}) = w_1 x + w_2 x^2 + w_3 x^3 + b$$

size: 1–10^3
size^2: 1–10^6
size^3: 1–10^9

feature scaling



Quadric It comes back down so



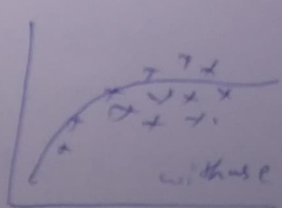
$f(\vec{w}, b(\vec{x})) = w_1 x + w_2 x^2 + w_3 x^3 + b$

so use cubic function that is new feature that we predict by feature engineering

Choice of features

$$f_{\vec{w},b}(x) = w_1 x + w_2 \sqrt{x} + b$$

↑ size ↑ $\sqrt{size}$



with use of root it look like this

Here we have problem what feature to use

Feature engineering

x_1 frontage | House
depth x_2

$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + b$

this model might work by ok but here's another option for how you might choose a diff way to use these features in the model that could be even more effective

you might notice that the area of land can be calculated as the frontage or width times the depth

area = frontage x depth

And you may have an intuition that the area of the land is more predictive of the price than the frontage and depth as separate features so you might define a new feature $x_3 = x_1 x_2$

$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$

step1 :- firstly find the hypothesis

$$f_{w,b}(x) = w_1 x_1 + w_2 x_2 \cdots\cdots w_n x_n$$

step2 :- find the cost

$$\min_{J(w,b)} \quad J_{w,b} = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

step3 :- find w & b from gradient descent

w ar b ko jab tak change करना है।

Jab Tak cost previous cost se increase na hojane

agar cost prev cost se inc hui to previous w, b return kar dane

Aga previous w and new w same aa jane To be w return kar dange

step4 :- Gradiend descent ko fast karne k lyea ham features ko ek range me kar lange scaling karke taaki gradient descent fast ho jane

step5 :- ham feature engineering karke bhi prediction fast kar sakte h