Ayush thaze

Rapidly search topic

Machine Learning

1) Linear Regression

Superwised learning

linear regression
- simple linear regression
- multiple linear regression
- polynomial linear regression
- regularization.

let say, we have data of CGPA, and placement package

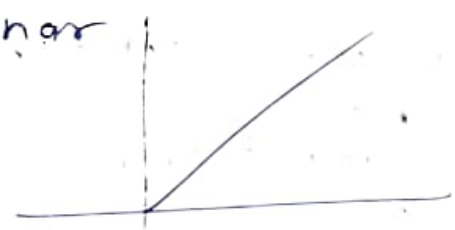
CGPA	Package
—	—
—	—
—	—
—	—

← simple linear regression

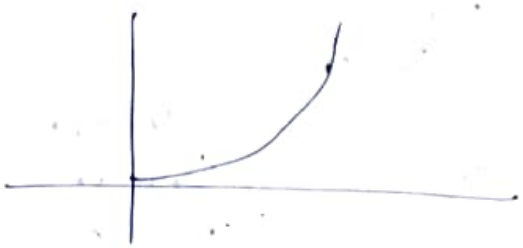
CGPA	gender	12th	State	package

multiple linear regression

linear



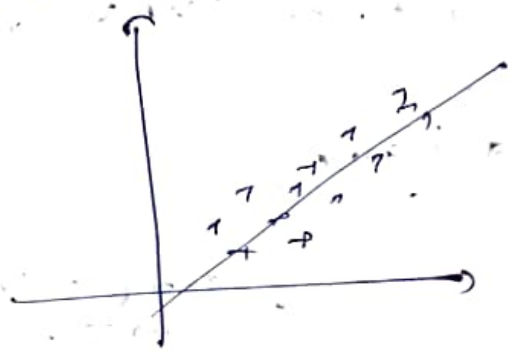
if we don't have linear data



CGPA	placement package

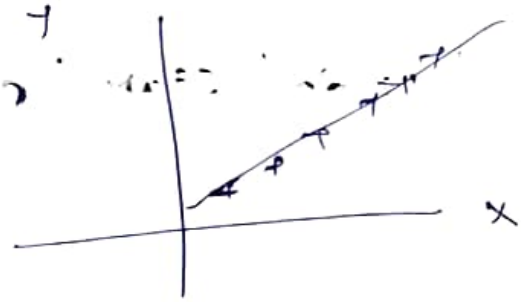
our task to predict package on basis of CGPA.

① We need to plot the data



← best fit line

why not linear
It mostly doesn't exist in real world.



$y = mx + c$

What it does?

It try to cover all datapoints such as to minimise error and closness.

```
from sklearn.model_selection import
    train_test_split
from sklearn.linear_model import
LinearRegression
lr = LinearRegression()
lr.fit(x_train)
lr.predict(x_test)
```

m = lr.coef_ — weightage

c = lr.intercept_ — offset in case it y.4X

techniques to find M and C:

To find m & b, there are two solutions
1) closed form solution
2) Non closed form solution

closed form solution?
It is a mathematical form of expression, can contain number, variables, operation functions, but no limits, differentiation and Integration.

from example
To solve ax^2+bx+c roots,

$$\text{roots} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$$

In Non closed form, we need to use approximation technique to reach there.

In Linar regression, there are two methods of calculating m and b.
1) OLS (ordinary least squares) used by Sklearn in internal
2) Approximation (Gradient descent)

Most of people explains it using gradient descent.

But why?
→ when we work with higher dimensions, to get answers from direct formula becomes difficult. There, so grad-descent becomes efficient technique

OLS → sklearn.LinearRegression()
gradient-descent → SGDregression

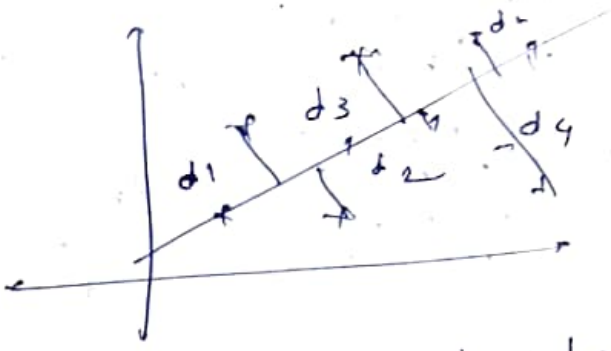
OLS direct formulaes

formula is,

$$m = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$c = \bar{y} - m\bar{x}$$

$\bar{x}$ and $\bar{y}$ are means
i = current row



Error = $d_1 + d_2 + d_3 + d_4 + d_5 \dots d_n$

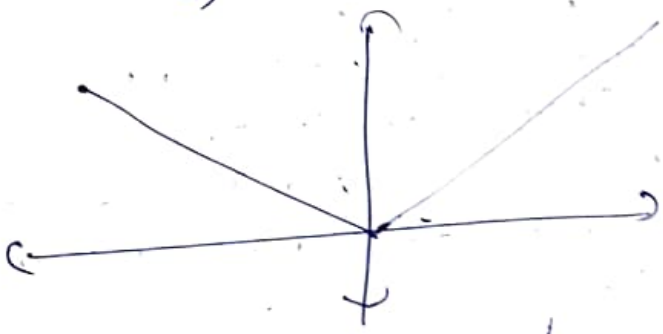
↓ To handle +ve & -ve both

errors = $d_1^2 + d_2^2 \dots d_n^2$

why not (d_i) ?

1) outliers stabalization

2) Next step involve differentiation



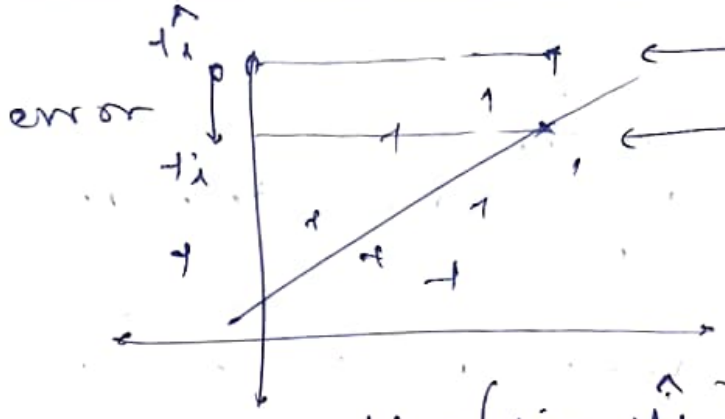
mod graph
→ continuous
but not diff

square can
do differentiation

$$E = \sum_{i=1}^{n} d_i^2$$

J
loss
function
error

I would say,
I need clost pass to every point.
In fact I can say,
I need to find m and c such that the value of eqn (E) should be minimum.



$$d_i = (y_i - \hat{y}_i)$$

actual Predicted

1 error !!

total $$E = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$\text{avg error} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Lines can be any, But we need only one with minimum E.

$\hat{y}_i = mx_i + c$ ∴ using line eqn

$$E(m,c) = \sum_{i=1}^{n} (y_i - mx_i - c)^2$$

So, we need m and b.

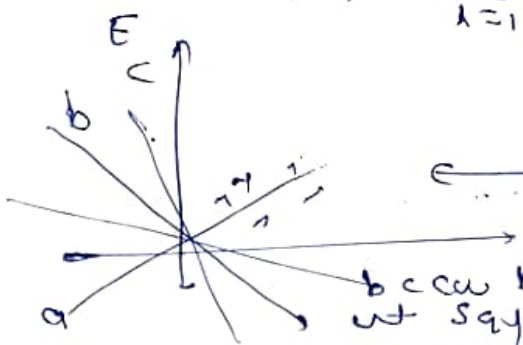
by changing m & b, we can increase & decrease error.

$y = f(x)$ → means, after changing x we get y changed.

So, we can modify alone m; alone c, or alone both. m and c.

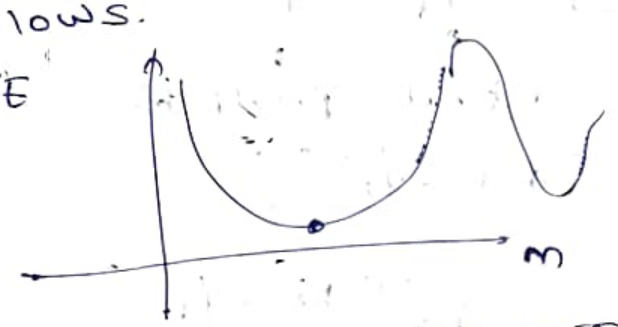
we start with m, :
we assume $c = 0$

$$E(m) = \sum_{i=1}^{n} (y_i - mx_i)^2$$



← possible lines. In around origin, we can rotate 360°.

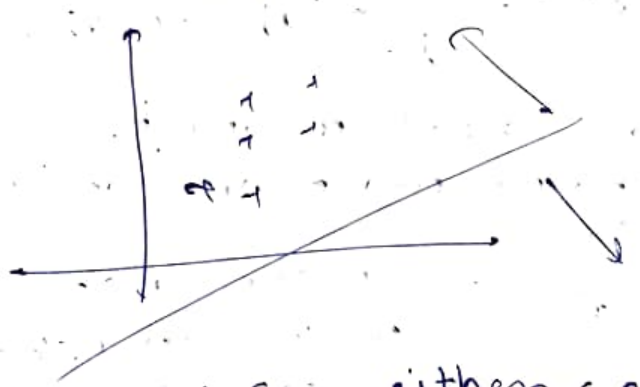
at assume
m = 10, E = 50.
Now, at I rotate line
line b will do less errors then line a.
at a, more more.
line c will do less errors then line b
so, after rotating. m, E changes as follows.



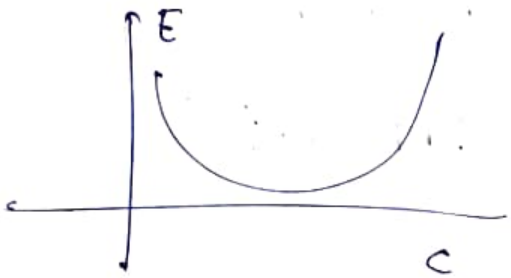
after a point, again errors will be increasing.

Now, we go with c.

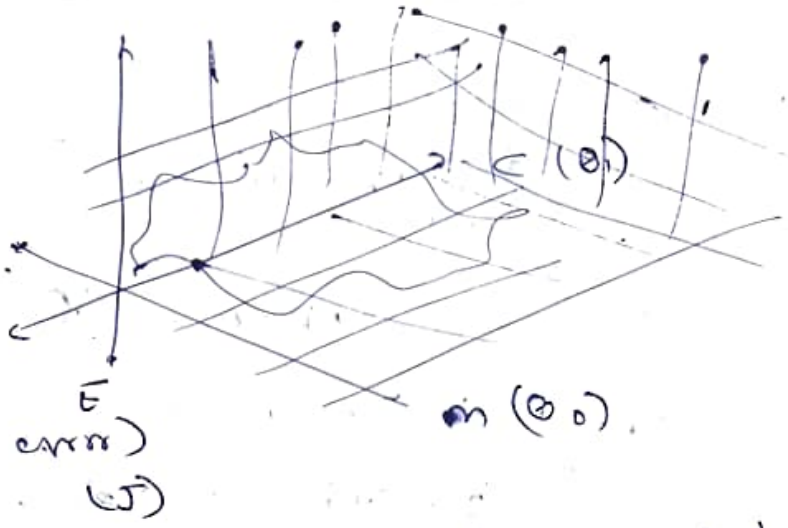
$$\therefore E(b) = \sum_{i=1}^{m} (y_i - x_i - b)^2, \quad (m = 1)$$



Now, we can either go up or down so, start there, calculate errors, more and minimize errors.



This is for c and E.



for certain values of m and b, error will be maximum. Similarly, for certain it will be minimum.

But how we can get maximums and minimums (heights and depths) for errors there?

-by maxima and minima in calculus.

On minimum, our slope becomes 0. for any equation of a line, we need to find its derivative, and put it to 0.

So, we will differenciate our ~~error~~ and error function E/J, and (with respect to both m and c) & put it 0.

If it was an equation, which was only dependent on x, then I, after taking,

$E(x)$ $\quad \frac{dE}{dx} = 0$ if it was only dep m x.

but if $f(x,y)$,

so, use partial diff

$$\frac{\partial E}{\partial m} = 0 \quad \& \quad \frac{\partial E}{\partial c} = 0$$

by this, I will get two eqns, & from them eqns, I will calculate m and c

solve the eqns,

$$\frac{\partial E}{\partial c} = \frac{\partial}{\partial c}\sum_{i=1}^{n}(y_i - mx_i - c)^2$$

$$= \sum_{i=1}^{n}\frac{\partial}{\partial c}(y_i - mx_i - c)^2$$

apply chain rule

$$= \sum_{i=1}^{n}\left(2\cdot(y_i - mx_i - c)^{2-1}\cdot \frac{\partial}{\partial c}(y_i - bx_i - c)\right)$$

$$= \sum_{i=1}^{n}\left(2(y_i - mx_i - c)\cdot(0 - 0 - 1)\right)$$

$$= \sum_{i=1}^{n}(-2)(y_i - mx_i - c) = 0$$

$$= \sum_{i=1}^{n}(y_i - mx_i - c) = 0$$

$$= \frac{\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} mx_i - \sum_{i=1}^{n} c = 0}{n}$$

$$= \frac{\sum_{i=1}^{n} y_i}{n} - \frac{\sum_{i=1}^{n} mx_i}{n} - \sum_{i=1}^{n}\frac{c}{n} = 0$$

$$= \bar{y} - m\bar{x} - \frac{nc}{n} = 0$$

$$= \bar{y} - m\bar{x} - c = 0$$

$$\boxed{c = \bar{y} - m\bar{x}}$$

$$E = \sum_{i=1}^{n}(y_i - mx_i - \bar{y} - m\bar{x})^2$$

$$\frac{\partial E}{\partial m} = \sum_{i=1}^{n} \frac{\partial}{\partial m} (y_i - mx_i - \bar{y} + m\bar{x})^2$$

$$\frac{\partial E}{\partial m} = 2 \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x}) \cdot \frac{\partial}{\partial m} (y_i - mx_i - \bar{y} + m\bar{x})$$

$$0 = 2 \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x}) (0 - x_i - 0 + \bar{x})$$

$$0 = (-2) \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})(x_i - \bar{x})$$

$$0 = \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})(x_i - \bar{x})$$

$$0 = \sum_{i=1}^{n} [(y_i - \bar{y}) - m(x_i - \bar{x})](x_i - \bar{x})$$

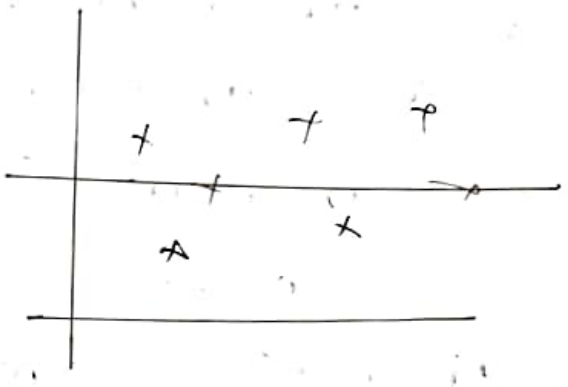
$$0 = \sum_{i=1}^{n} [(y_i - \bar{y})(x_i - \bar{x}) - m(x_i - \bar{x})^2] = 0$$

$$\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x}) = m \sum_{i=1}^{n} (x_i - \bar{x})^2$$

$$\boxed{m = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}$$

Regression metrics

1) MAE - mean absolute error



Simply the distance between actual and predicted.

$$MAE = \frac{|y_1 - \hat{y}_1| - |y_2 - \hat{y}_2| \dots - |y_n - \hat{y}_n|}{n}$$

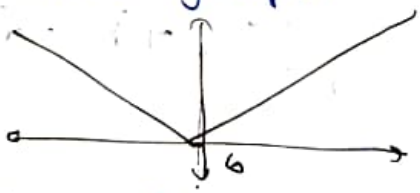
$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$$

Advantage

1) same unit
2) Robust to outliers

Disadvantage

1) Absolute mod graph



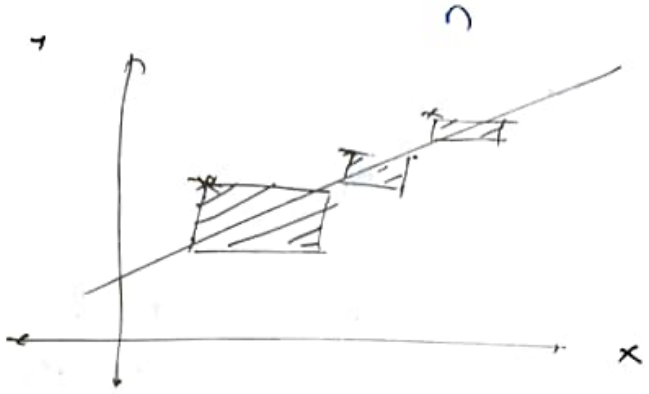
cannot differentiable at 0.

We apply optimization, so we need differentiation techniques

So, thats why mean squared error was introduced.

2) Mean squared error (MSE)

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$



Advantage

can use as loss function, as it is differentiable.

Disadvantage

1) unit squares-

$y^2 \rightarrow y$ Interpretation.

2) Not robust to outliers to square

3) RMSE (root mean squared error)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{N}}$$

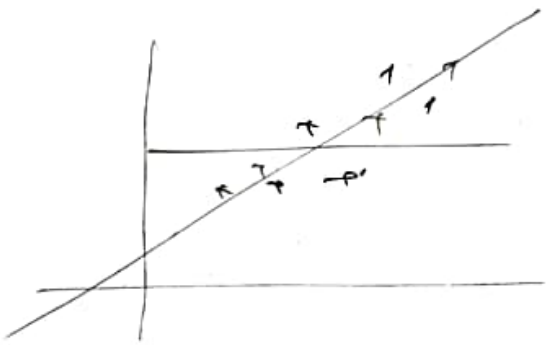
Advantage

same unit

Disadvantage

Not much robust to outliers.

4) R2 score



for one variable Independent

when we calculate R2 score, we just compare how much better our LR line, then mean line.

Coefficient of determination / Goodness of fit

$$R2 \text{ score} = \frac{SSR}{SS_m}$$

SSr = sum of squared error in LR

SSm = sum of squared error in mean.

$$R2 \text{ score} = 1 - \frac{\left[\sum_{i=1}^{n} (y_i - \hat{y})^2\right]_{reg}}{\left[\sum_{i=1}^{n} (y_i - \bar{y})^2\right]_{mean}}$$

comparision of worst to the best

special cases

1) Case 1) r2 score = 0

$$\frac{\left[\sum_{i=1}^{n} (y_i - \hat{y})^2\right]_{reg}}{\left[\sum_{i=1}^{n} (y_i - \bar{y})^2\right]_{mean}} = 1$$

both errors are same No advantage of regression, As model is unable to pick up ~~disad~~ advantage of model features.

2) Case 2) R2 score = 1

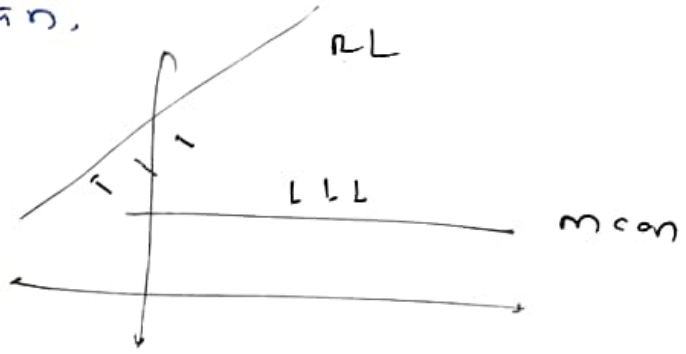
$$\frac{\left[\sum_{i=1}^{n} (y_i - \hat{y})^2\right]_{reg}}{\left[\sum_{i=1}^{n} (y_i - \bar{y})^2\right]_{mean}} = 0$$

mean, no error by regression. Perfect line

case 3) Negative R2 score

if SSR > SSM

means RL is very bad then mean.



Need to look at data & algorithm

Interpretation:

We say R2 score is 0.8

It means our input columns are able to explain 80% of variance in target column. Rest 20% we don't have actually.

Advantage

apply universally on all datasets

Disadvantage

if we add more columns in input, our R2 score starts increasing.

Adjusted R2 score

$$R2adj^2 = 1 - \left[\frac{(1-R2)(n-1)}{(n-k)}\right]$$

r_2 = Normal r_2 score

n = total number

k = total no of input columns

let us explain this by two cases:

1) case 1)

adding irrelivant column.

case 2)

adding relevent column

1) case - I

our R^2 score will either somewhat decrease or it will constant.

$$R^2_{adj} = \frac{(1-R^2)(n-1)}{(n-1-k)} \downarrow$$

$$(1-\downarrow) \downarrow$$

So overall R^2_{adj} ↓.

2) Case - II

$$R^2_{adj} = \frac{(1-R^2)(n-1)}{(n-1-k)} \downarrow$$

$1 - \updownarrow \Rightarrow \uparrow$

So, overall Increased.

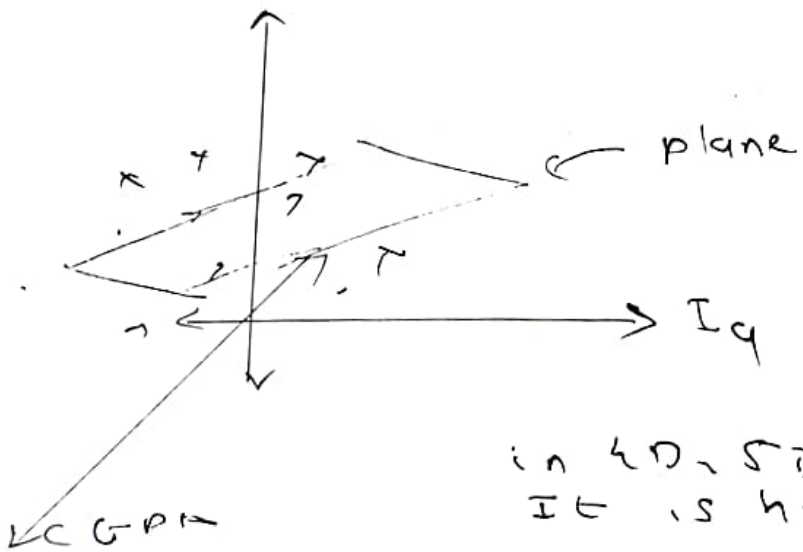
Multiple Linear regression.

Comes when there are many inputs.

Iq	subject	CGPA	package
-	-	-	-
-	-	-	-
-	-	-	-

let assume it takes 2. ignore sub for now

package



in 4D, 5D — nD
It is hyperplane

In 2D.

$y = mx + c$

In 3D

$y = mx_1 + mx_2 + c$

i.e. $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

need to find $\beta_0, \beta_1, \beta_2$.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \dots \beta_n x_n$$

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i$$

n column = n+1 coefficient

coefficient = weight.

less weight means no much contribution

high weight means significant contribution

mathematical Intution

cgpa	id	gender	package
x_1	x_2	x_3	y

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

$$\hat{Y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} \beta_0 \beta_0 x_{11} & \beta_0 x_{12} & \dots & \beta_n x_{1m} \\ \beta_0 \beta_0 x_{21} & \beta_0 x_{22} & \dots & \beta_n x_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_0 & \beta_n x_{n1} & \beta x_{n2} & \dots \beta_n x_{nm} \end{bmatrix}$$

$\hat{y}_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} \dots \beta_n x_{1n}$

m columns

$$\begin{bmatrix} 1 & x_{11} & x_{12} & \dots x_{1m} \\ 1 & x_{21} & x_{22} & \dots x_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots x_{nm} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}$$

$$\hat{Y} = X\beta \quad \text{---------- (1)}$$

$\hat{Y}$ ← prediction of all output rows

X ← coefficient matrics

β

$X =$

	x_1	x_2	x_3
1	x_{11}	x_{12}	x_{13}
1	x_{21}	x_{22}	x_{23}
1	x_{31}	x_{32}	x_{33}

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \qquad E = y - \hat{y}$$

$$E = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \hat{y_1} \\ \hat{y_2} \\ \vdots \\ \hat{y_n} \end{bmatrix}$$

$$E = \begin{bmatrix} y_1 - \hat{y_1} \\ y_2 - \hat{y_2} \\ \vdots \\ y_n - \hat{y_n} \end{bmatrix}$$

In simple linear regression

$$E = \sum_{i=1}^{n} (y_i - \hat{y_i})^2$$

so we can write E as at transpose e,

$$\underset{(1\times n)}{\begin{bmatrix} y_1 - \hat{y_1} & y_2 - \hat{y_2} & \cdots & y_n - \hat{y_n} \end{bmatrix}} \underset{(n\times 1)}{\begin{bmatrix} y_1 - \hat{y_1} \\ y_2 - \hat{y_2} \\ \vdots \\ y_n - \hat{y_n} \end{bmatrix}}$$

$: (1 \times 1)$
Scalar

$$= (y_1 - \hat{y_1})^2 + (y_2 - \hat{y_2})^2 + \cdots + (y_n - \hat{y_n})^2$$

$$E = \sum_{i=1}^{n} (y_i - \hat{y_i})^2$$

$$\boxed{E = E^T e}$$

Now:

$$E = e^T e$$

$$= (y - y')^T \cdot (y - y')$$

$$(A+B)^T = A^T + B^T$$
$$(A-B)^T = A^T - B^T$$

$$= (y^T - \hat{y}^T)(y - \hat{y})$$

$$= \left[y^T - (X\beta)^T \right] (y - X\beta)$$

$$= y^T y - \underline{y^T X\beta - (X\beta)^T y} + (X\beta)^T X\beta$$

To prove ——— (V)

$$= \underline{y^T X\beta} = (X\beta)^T y$$

$$A^T B = B^T A \quad \text{—— (1)}$$

let say $(A^T B)^T = B^T (A^T)^T$

$$= B^T A \quad \text{—— (2)}$$

RHS eqn (1) = eqn (2) RHS

in fact, we say

$$A^T B = (A^T B)^T$$

we say,

$$A^T B = c$$

$$c = c^T \leftarrow \text{To prove}$$

$$A^T B = \underline{y^T X\beta}$$

$$(y^T X\beta)^T = (y^T X\beta) \quad \text{—— (3)}$$

$[(1 \times n)]$ $[(m+1)(1)]$

$[(n \times (m+1))]$

$[1 \times (m+1)]$ $[(m+1)(1)]$

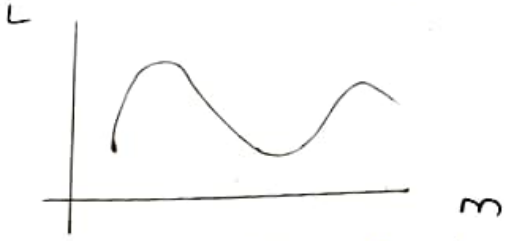
$= [(1 \times 1)]^T = [1]^T$

Coming back at eqn V

$$E = y^T y - 2 y^T x\beta + \beta^T x^T x\beta$$

loss function

like in SLR, we need to diff it to find slope,



similarly, we will find $\frac{dE}{d\beta}$.

$$\frac{dE}{d\beta} = \frac{d}{d\beta}\left[y^T y - 2 y^T x\beta + \beta^T x^T x\beta\right]$$

$$= 0 - 2y^T x + \frac{d}{d\beta}\left[\beta^T x^T x\beta\right]$$

matrix differentiation

Going in depth

Let us say Ax.

vector under transformation

Let say

$$A = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$$

$$Ax = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$f_1(x_1, x_2) = x_1 + 2x_2$$

$$f_2(x_1, x_2) \longrightarrow 3x_1 + 4x_2$$

$$\begin{pmatrix} x_1 + 2x_2 \\ 3x_1 + 4x_2 \end{pmatrix}$$

How to find

$$\frac{d}{dx}(Ax) = ?$$

$$\frac{\partial}{\partial} Ax = \begin{bmatrix} \frac{dF_1}{dx_1} = 1 & \frac{dF_4}{dx_2} \; 2 \\ \frac{dF_{\cdot 2}}{dx_1} = 3 & \frac{dF_2}{dx_2} = 4 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix} = A$$

$$\therefore \frac{d}{dx} Ax = A$$

So, talk about $x^T Ax$

$$= (x_1 \quad x_2 \; - \; - \; x_n) \begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ \cdot & & \cdot \\ a_{n1} & \cdot \; - & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$$\begin{pmatrix} a_{11}x_1 + a_{12}x_2 - \; - a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 \; - \; - a_{2n}x_n \\ \cdot \\ a_{n1}x_1 + a_{n2}x_2 \; - a_{nn}x_n \end{pmatrix}$$

Let now try to just look it as symetric. supm $a_{24} = a_{42}$ say A

So, we say

$$(x_1 \quad x_2 \; - \; - x_n) \begin{pmatrix} a_{11} & a_{12} & \cdot & a_{1n} \\ a_{21} & a_{22} & \cdot & a_{2n} \\ a_{31} & & & \\ \vdots & & & \\ a_{n1} & \cdot & - & a_{nn} \end{pmatrix}$$

if we say $n = 2$,

$$a_{11}x_1^2 + a x_1 x_2 + a x_1 x_2 + a_{22}x_2^2$$

$= a_{11}x_1^2 + 2a_{12}x_2 + a_{22}x_2^2$

$= f(x_1, x_2)$

So, $\frac{d}{dx}(x^T A x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{df}{dx_2} \end{pmatrix}$

$\therefore \begin{pmatrix} 2a_{11}x_1 + 2a_1x_2 \\ 2ax_1 + 2a_{22}x_2 \end{pmatrix}$

$= 2\begin{pmatrix} a_{11}x_1 + a x_2 \\ a x_1 + a_{21}x_{22} \end{pmatrix}$

$= 2\begin{pmatrix} a_{11} & a \\ a & a_{22} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

$= 2AX$

$\frac{\partial}{\partial x} 2AX = \frac{\partial}{\partial x} x^T A x$

---

come back to step

$: 0 - 2y^T X + \frac{\partial}{d\beta}[\beta^T X^T X \beta]$

$= -2y^T X + 2X^T X \beta^T$

$\therefore 2X^T X \beta^T = 2y^T X$

$\beta^T = y^T X \cdot \frac{1}{X^T X}$

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

$(\beta^T)^T = [(X^T X)^{-1}]^T (y^T X)^T$

$$\beta = \left[(X^T X)^{-1}\right]^T X^T y$$

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

$X = X_{train}$

$y = y_{train}$

$$\begin{bmatrix} \beta_1 \\ \beta_2 \\ \\ \beta_n \end{bmatrix} = [(m+1) \times (m+1)]\,[(m+1)(n)]$$

$$[(m+1) \times 1] = [(m+1) \cdot n]$$

This was OLS.

Still why gradient-descent?

Inverse time complexity : n^3.

a lot of computing. lot of time

OLS = Linear Regression

SGD regressor = high dimensions.

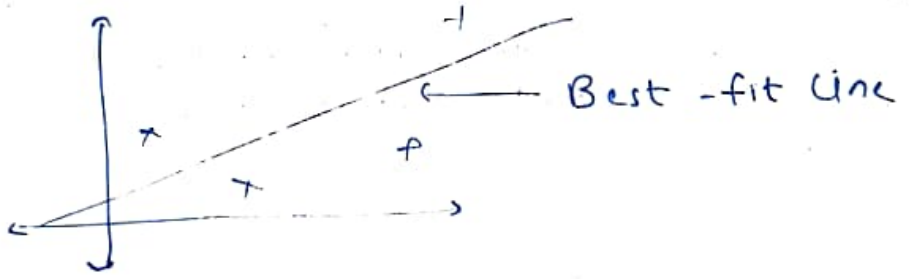
Gradient-descent

A technique to find out minima effectively.

1) Linear reg
2) Logistic reg
3) T-SNE reg
4) DL backbone

Differential function → Gradient descent → minima

let I have a dataset



we can do it by OLS

$$\hat{y} = mx + c$$

$$L_i = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

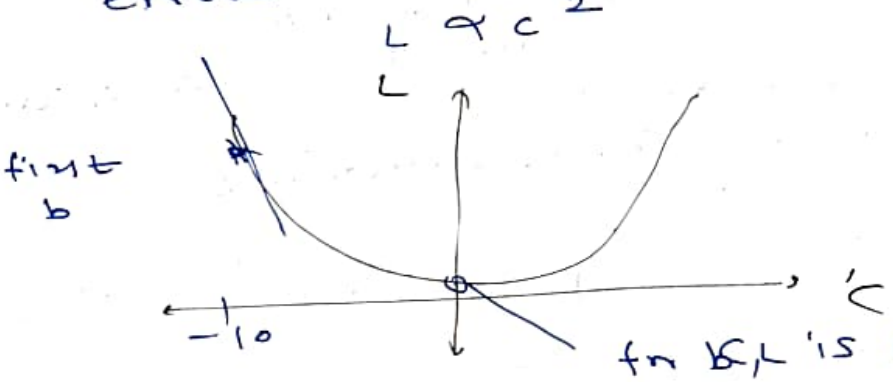
$$= \sum_{i=1}^{n} (y_i - mx_i - c)^2$$

Depends on m & c both.

let I assume that I know m, so, let it be 10.

$$L = \sum_{i=1}^{n} (y_i - 10x_i - c)^2$$

Now, I just simplied it, we need to find such c by adjusting height that do minimum errors.



we can find it by OLS, but cant in higher dimension.

so gradient descent

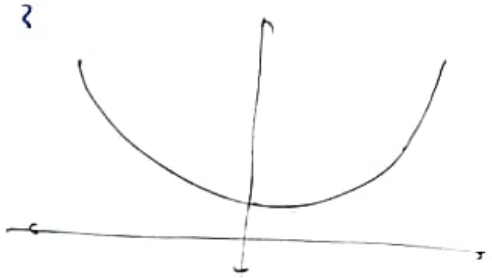
① Select a value for c. (random)

let b = -10

Algorithm have two direction for going towards minima either upper or lower. basically increasing or decreasing c,

we will simply find slope

how to find slope of a point to a curve?



$$y = mx + c$$

$$\frac{dy}{dx} = m \text{ (slope)}$$

Step ②

So, find slope of that point wit curve

find $\frac{dy}{dx} = m$, if put x

if

$m < 0 \rightarrow$ increase c

$m > 0 \rightarrow$ decrease c

$$C_{new} = C_{old} - m$$

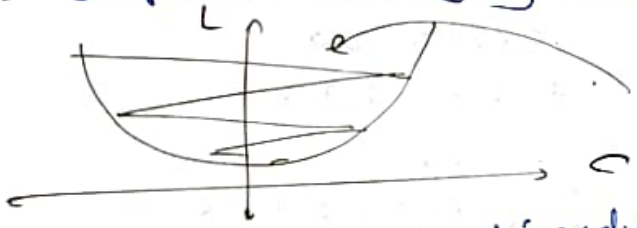
Gradient - descent equation.

back to the definatin repeated steps
The idea is to take opposite direction of the gradient. (derivate)

This will happening in loop.

But a point to be noted is,

As slope is changing whole value,

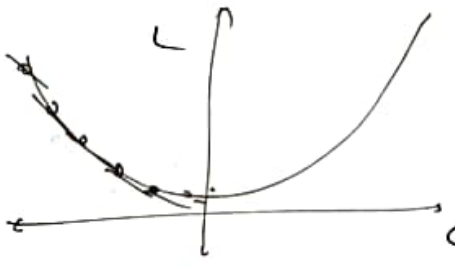


Too many diffections

To handle this tragedy, We multiply slope with a thing called learning rate

$$C_{new} = C_{old} + \eta m$$

η - learning rate = usually 0.01



if we take learning rate (η) as

so small = slow speed

so large = jumps and can skip eqn.

and this is an Iterative algorithm

when to stop

$C_{new} - C_{old} = 0.00001$

A very small number

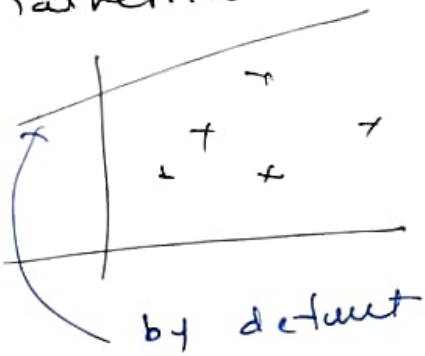
Increment / Decrement

so this was approach 1.

approach 2 : Limit ization.

| Iteration = epoxes |

Mathematical Intution



m & c.

we know m,

we ~~don't~~ only need c

by default we

1) start with a random of c.

2) for i in epox

b new = b old − 2 slope

λ - learning rate

at say 0.01.

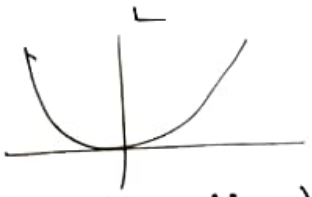
In first loop,

we have to find b new.

we have b old, that orginal we.

But how to find slope?

$$L = \sum_{i=1}^{n} (y_i - \hat{y})^2$$



from this eqn we will find slope eqn

& let say we have a c value.

$$\frac{dL}{dk} = \frac{d}{dc}\left(\sum_{i=1}^{n} (y_i - \hat{y_i})^2\right)$$

$$= \frac{d}{dc}\left(\sum_{i=1}^{n} (y_i - mx_i - c)^2\right)$$

$$= 2(y_i - mx_i - c)\frac{d}{dc}(y_i - mx_i - c)$$

$$= 2(y_i - mx_i - c)(0 - 0 - 1)$$

$$= -2\sum_{i=1}^{n}(y_i - mx_i - c)$$

let us say c is 5.
now, I can write,
as m=10 already assumed

$$= -2 \sum_{i=1}^{2} (y_i - 10x_i - 5)$$

after this, we get our slope at b=5.

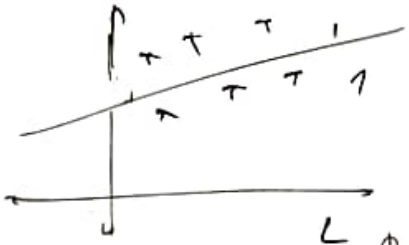
$$c_{new} = c_{old} - \eta \cdot slope_{(c=5)}$$

do it until n times.

Now, let us find m.

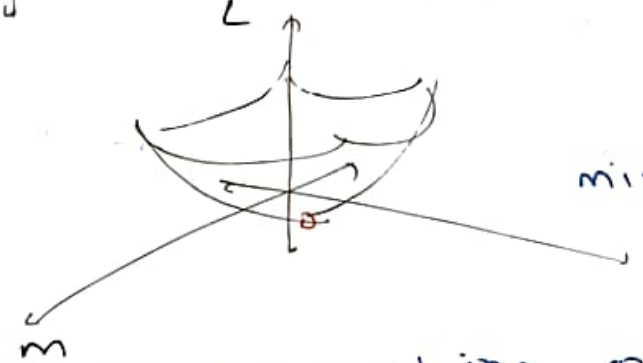
step 1) Initialize for m & c as random.
step 2) decide epoch & learning rate

for in in epochs:
$b = b - \eta \cdot slope$
$m = m - \eta \cdot slope$



$$loss = \sum_{i=1}^{2} (y - \hat{y}_i)^2$$

$$f(m,c) = \sum_{i=1}^{2} (y_i - mx_i - c)^2$$



we are applying gradient descent in both's terms.

In terms of c and m
slope has two compounds, m & c.
so we need diff wrt. m & wrt c.
m wrt dor is m direction
c dor is c direction
It is a combination of two diff variables.

$\frac{\partial L}{\partial b}$ for b | $\frac{\partial L}{\partial m}$ for m

$\sum_{i=1}^{n}(y_i - mx_i - c)$

wrt c | wrt m

$\frac{\partial L}{\partial c} = \sum_{i=1}^{n}(y_i - mx_i - c)(0 - 0 - 1)$

$= -2\sum_{i=1}^{n}(y_i - mx_i - c)$

we can take $b = 0$, as a starter

$\frac{\partial L}{\partial c} = 2\sum_{i=1}^{n}(y_i - mx_i - c)(0 - 0 - x_i - c)$

$-2\sum_{i=1}^{n}(y_i - mx_i - c)x_i$

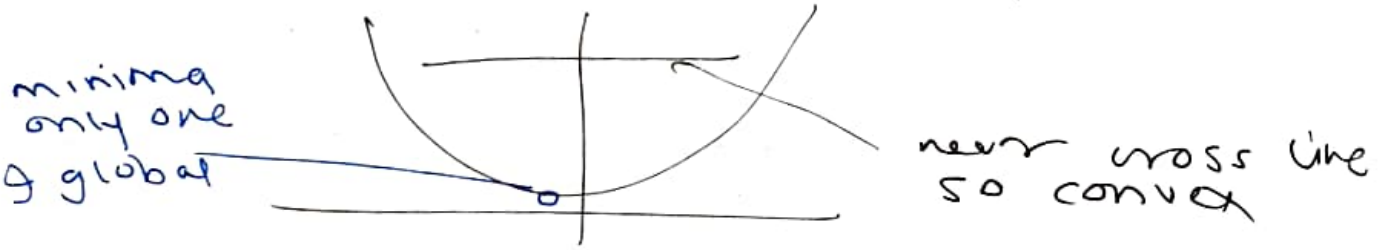
effect of too much high or low learning rate

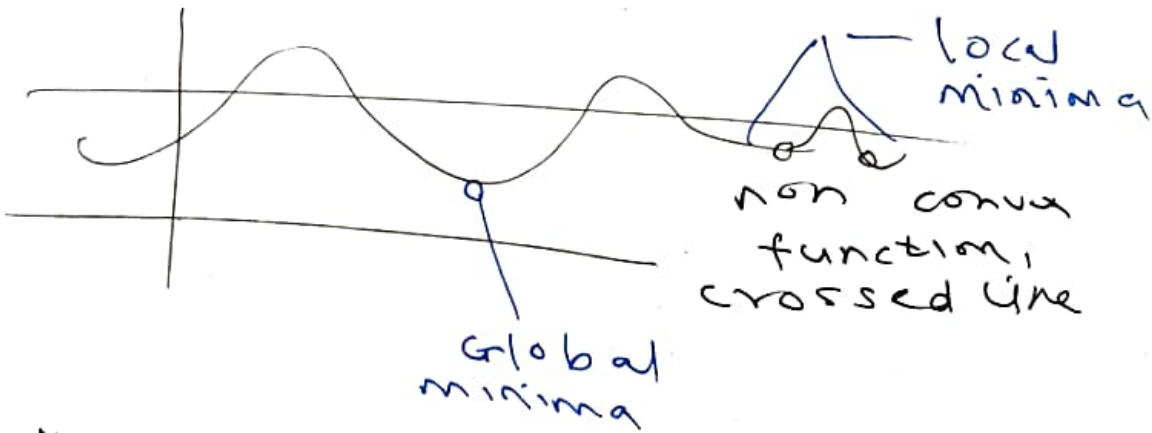
less learning rate - time to train

more learning rate - To jumps, no reach.

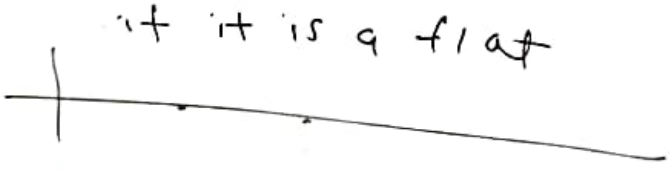
effect of loss function

$L = \Sigma(y - \hat{y})^2$ ← convex fun





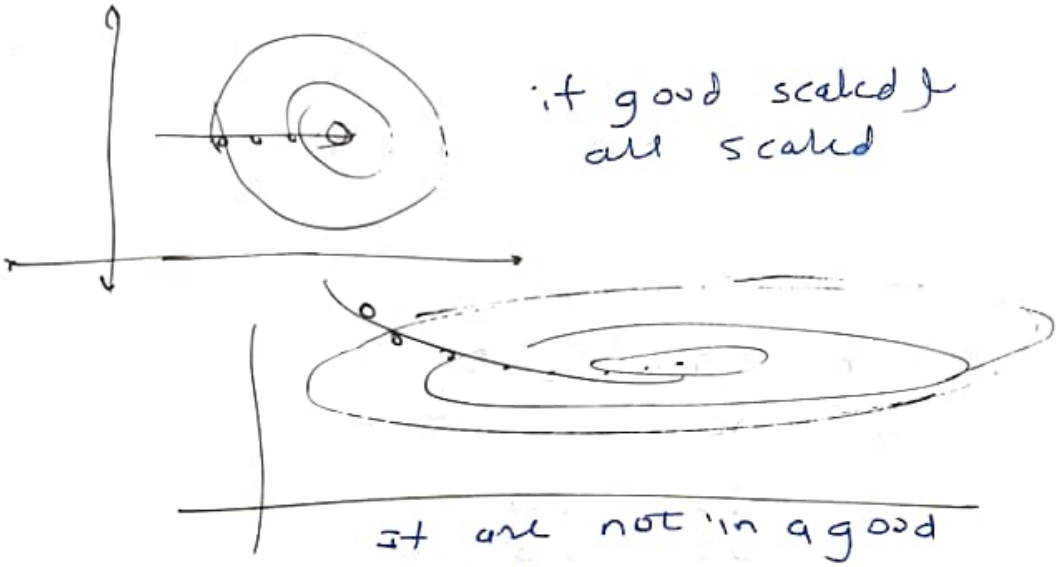
secondly,



less epoches can't let reach you to desired place.

effect of data

if columns in are same scale, we will defend on high speed.



if good scaled if all scaled

if are not in a good scale

Types of gradient descent

batch GD

stochastic GD

mini batch GD

basis of the types

$\frac{\partial L}{\partial m}$

$m = 1, b = 0$

iterate

$m_n = m_0 - \eta(\text{slope})$

$b_n = b_0 - \eta(\text{slope})$

$b = 0$

$\frac{\partial L}{\partial b}$

In this we look all rows literally every row.

This is a batch gradient descent.

which is slow.

To prevent this computational problems such kinds we just look a single row.

fast, suitable for larger datasets.

and if we say, we wish mid of batch and stochastic then,

it is a mini batch.

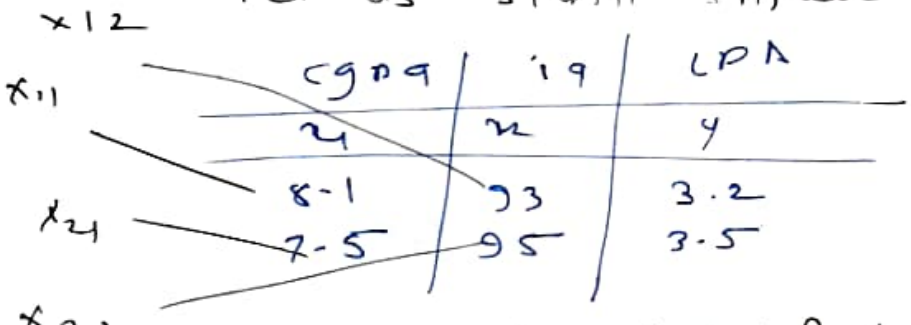
we define a batch size there update m & b according to m batch size

min-batch is somewhat rare.
batch is very rare and sometimes for smaller datasets, and convex loss function
stochastic is for large dataset, and it is common.

The past was one of batch GD, but it was 2D.

Let us study > 2D.

let us start small



$$y = B_0 + B_1 x_1 + B_2 x_2$$

LPA cgpa Iq

To find : B_0, B_1, B_2

step ① Start random

$B_0 = 0, \; B_1 = B_2 = 1$

step ② decide

epoch = 100

learning rate = 0.1

$B_0 = B_0 - \eta \text{ slope}$

$B_1 = B_1 - \eta \text{ slope}$

$B_2 = B_2 - \eta \text{ slope}$

$L(B_0, B_1, B_2)$ ⟶ 4D graph

3 components there

In direction of B_0, B_1 & B_2.

$\frac{\partial L}{\partial B_0}$ $\frac{\partial L}{\partial B_1}$ $\frac{\partial L}{\partial B_2}$

n columns → n+1 derivatives → update in equations

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

↓

$$= \frac{1}{2}\left[(y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2\right]$$

$\hat{y} = B_0 + B_1 x + B_2 x_2$

$\hat{y}_1 = B_0 + B_1 x_{11} + B_2 x_{12}$

$\hat{y}_2 = B_0 + B_1 x_{21} + B_2 x_{22}$

$$L = \frac{1}{2}\left[(y_1 - B_0 - B_1 x_{11} - B_2 x_{12})^2 + (y_2 - B_0 - B_1 x_{21} - B_2 x_{22})^2\right]$$

$$\frac{\partial L}{\partial B_0} = \frac{1}{2} \times \left[2(y_1 - B_0 - B_1 x_{11} - B_2 x_{12})(0 - 1 - 0 - 0) + 2(y_2 - B_0 - B_1 x_{21} - B_2 x_{22})(-1)\right]$$

$$= \frac{-2}{2}\left[(y_1 - \hat{y}_1) + (y_2 - \hat{y}_2)\right]$$

= But But But for n,

$$= \frac{-2}{n}\left((y_1 - \hat{y}_1) + (y_2 - \hat{y}_2) + (y_3 - \hat{y}_3) \cdots (y_n - \hat{y}_n)\right) \quad \text{---(1)}$$

$$\frac{\partial L}{\partial B_0} = \frac{-2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$$

$$\frac{\partial L}{\partial B_1} = \frac{1}{2}\left[2(y_1 - \hat{y}_1)(-x_{11}) + 2(y_2 - \hat{y}_2)(-x_{21})\right]$$

$$\frac{\partial L}{\partial B_1} = \frac{-2}{n}\left[(y_1 - \hat{y}_1)(x_{11}) + (y_2 - \hat{y}_2)(x_{21}) + (y_3 - \hat{y}_3)(x_{31}) \cdots + (y_n - \hat{y}_n)(x_{n1})\right]$$

$$\frac{\partial L}{\partial B_1} = \frac{-2}{n} \cdot \sum_{i=1}^{n} (y_i - \hat{y}_i)\, x_{i1} \quad \text{---------- (2)}$$

what x_{i1} represents ?
first column all.

x_{11} x_{12} x_{13}
x_{21} x_{22}
x_{31} x_{32}
x_{41} x_{42}

$$\frac{\partial L}{\partial B_2} = \frac{-2}{n} \cdot \sum_{i=1}^{n} (y_i - \hat{y}_i) x_{i2} \quad \text{---------- (3)}$$

what is x_{i2}
all values of second column.

$$\boxed{\frac{\partial L}{\partial B_m} = \frac{-2}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)\, x_{im}}$$

m = no of columns
n = no of rows

Stochastic gradient descent

problem in batch gradient descent

$$\frac{\partial L}{\partial B_n} = \frac{-2}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)\, x_{ii}$$

we are calculating n derivatives for each and every coeff + 1.

ex n = 1000 (no of rows)
col = 5 (no of columns)
coe. = 5 + 1 = 6

epoch = 50

50
1 → 1000 drv
6 → 6000 der

= 50 × 6,0,000
= 3,00,000 der

but what if it is a large number ?

A lot of computation

Hardware limitation,
While calculating y hat, we ~~can't~~ need to load x-train entirely in memory. Can't deal in large dataset.

updating coef's and intercept

batch GD
look whole data

stochastic GD
look a single row

while updating rows, we select at random

Advantage
faster rate of convergence

disadvantage
no steady solution.

for same no of epoches

batch

faster

update for no of epoches

stochastic

faster, But reach to sol for

update for no of epoches x no of row

slow calculation, because too many computes
But goes to solution fast, and can vary results.

when to use stochastic?

1) Big data
2) non convex function

Disadvantages

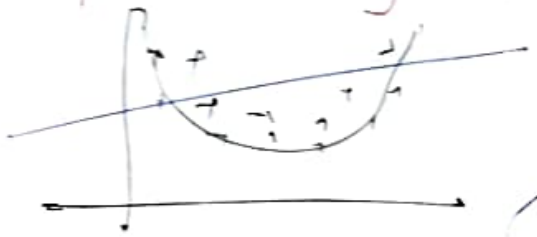
Even after reaching near of actual sol^n, we even struggle & fluctuate there

→ solve this,
we can change learning rate later

mini-batch gradient descent

In between batch and stochastic

Polynomial regression



for a linear regression,
I have eqn $y = mx + c$

$y = B_0 + B_1 x$

$y = B_0 + B_1 x_1 \ldots B_n x_n$

multiple linear regression
simple linear regression

If I wish to bring polynomial, what I will do is I will convert it to polynomial features in preprocessing.

x	y

(1,2)

(1,9)

at degree 2

x_0, x_1, x_2

(35, 100)

x_0	x^1	x^2
1	35	1225

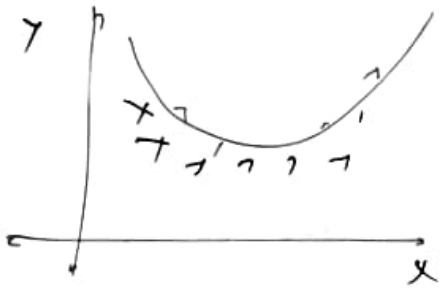
for deg 3

x_0	x_1	x_2	x_3
1	35	1225	150625

degree → Hyperparameter

eqn = $B_0 + B x_1 + B_2 x_2^2 - - B x^n$

effals of choosing degree
for ~~too~~ too small,



will not cover datapoints properly and set attributes

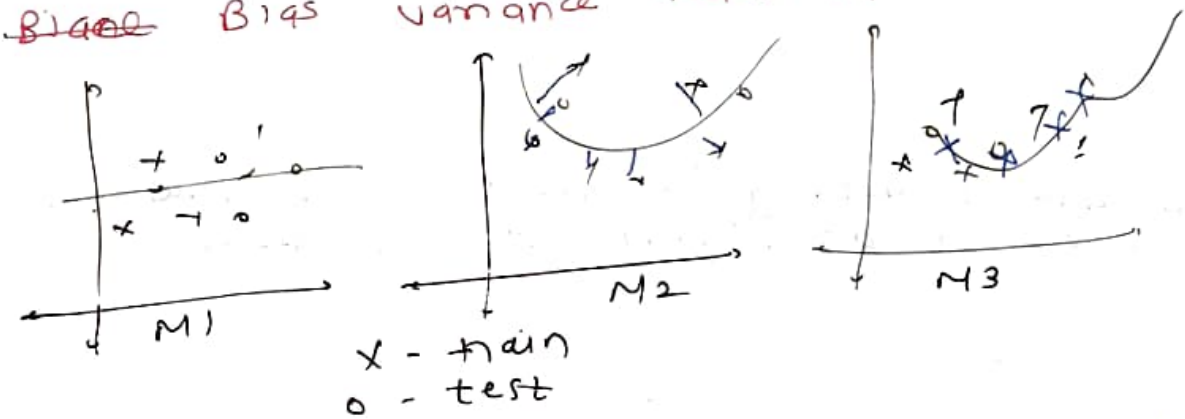
for too large

overfitting - need to find optimal value

wt we say, for 2 cols

$$y = B_0 + B_1 X_1 + B_2 X_1^2 + B_3 X_2 + B_4 X_2^2$$

~~Bias~~ Bias variance trade-off



Bias:
The disability of a ML model to capture relationship between training data
M3 - Now bias
M1 - High bias

Variance
diff bet^n train data error & test data error.
so M1 & M2 - low variance
as they cover train & test both as well. So, LV.
In M3, test error will very high & train error low.
So High variance

overfitting and underfitting

overfitting

if ML model works very good on train, but not on test.

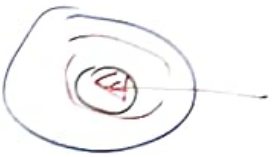
train error <<< test error

model M3

underfitting

it model unable to understand data properly.

model M1



low bias + low variance

To methods to deal with this

regularization

bagging

boosting

Ridge Regression

L2 Regularization

we add something in ML model so that our probabuty of overfitting get decreased.

Techniques available:

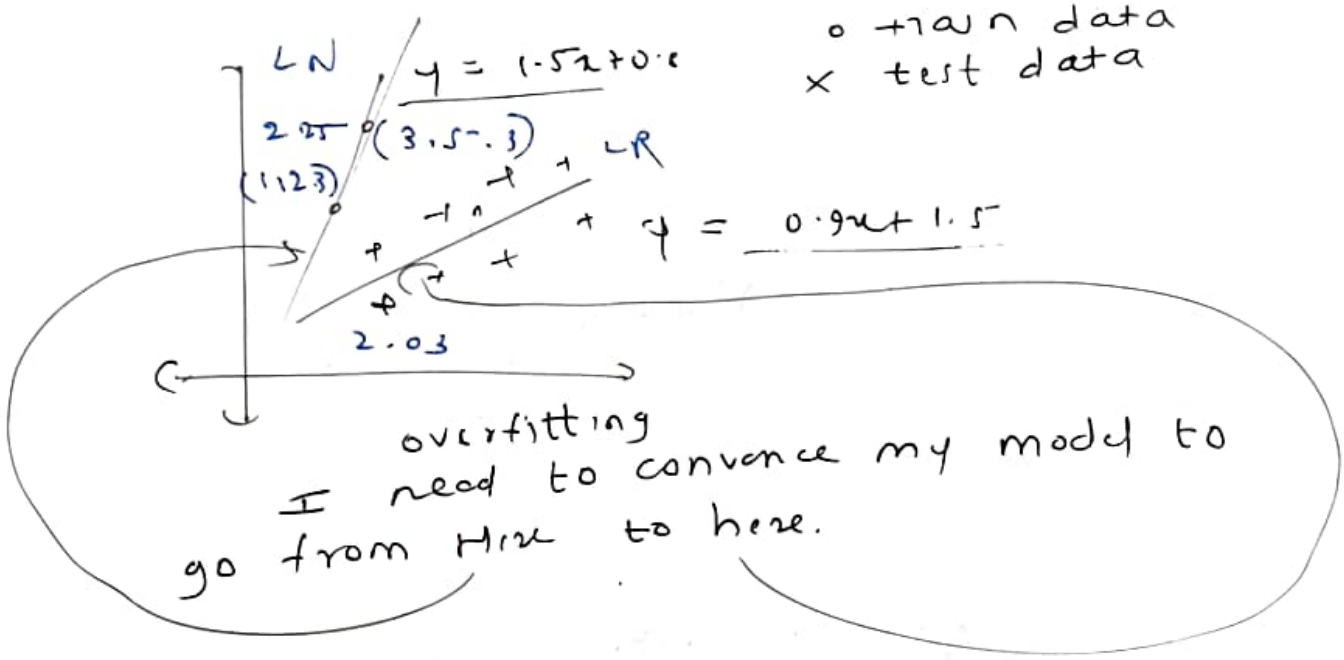
1) Ridge regression (regularization)
2) Lasso regression — | —
3) Elastic Net regression — | —

Overfitting in second term,

$y = mx + c$

extreme high value
high dependince of model

so see the next page



overfitting

I need to convince my model to go from Hire to here.

To minimize error,
I do like this

$$LF = \sum_{i=1}^{n} (y_i - \hat{y})^2 + \lambda(m^2)$$

I add this term.

λ = hyperparameter $(0 - \infty)$

m = slope

But how I will convince my ML model that?

we are assuming test data best fit line here

I will calculate errors

Loss LN	Loss LR
$\lambda = 1$	$\lambda = 1$
$0 + (1.5)^2 =$	$(2.3 - 0.9 - 1.5)^2$
2.25	$+ (5.3 - 2.7 - 1.5)^2$
	$+ (0.9)^2$
	$= (0.1)^2 + (1.1)^2 + (0.9)^2$
	$= 2.03$

(L2 regularization)

what if we had more axes?
we will add $\lambda(m_1^2 + m_2^2 \ldots)$

mathematics from scratch

see LR overview first, and continue

$$L = \sum_{i=1}^{n} (y - \hat{y}_i)^2 + \lambda m^2$$

also $L = \sum_{i=1}^{n} (y_i - mx_i - b)^2$ — $\frac{\partial L}{\partial m}$, $\frac{\partial L}{\partial b}$

$$\frac{\partial L}{\partial b} = (2) \sum_{i=1}^{n} (y_i - mx_i - b)(0 - 0 - 1)$$

$$\sum_{i=1}^{n} (y_i - mx_i - b) = 0$$

$$\sum_{i=1}^{n} (y_i) - m \sum_{i=1}^{n} (x_i) = nb$$

$$\frac{\sum y_i}{n} \qquad \frac{\sum x_i}{n} \qquad \frac{nb}{n}$$

$$b = \bar{y} - m\bar{x} \quad \text{——— (1)}$$

$$L = \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})^2 + \lambda m^2$$

$$= 2 \sum_{i=1}^{n} (y_i - mx_i - \bar{y} - m\bar{x})(0 - x_i - 0 + \bar{x}) + 2\lambda m$$

$$= -2 \left(\sum_{i=1}^{n} (y_i - \bar{y} - mx_i + m\bar{x})(x_i - \bar{x}) \right) + 2\lambda m = 0$$

$$= \sum_{i=1}^{n} (y_i - \bar{y} - mx_i + m\bar{x})(x_i - \bar{x}) + \lambda m = 0$$

$$\sum_{i=1}^{n} ((y_i - \bar{y}) - m(x_i - \bar{x}))(x_i - \bar{x}) + \lambda m = 0$$

$$\lambda m - \sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x}) - m(x_i - \bar{x})^2 = 0$$

$$= \lambda m - \sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x}) + m \sum_{i=1}^{n} (x_i - \bar{x})^2 = 0$$

$$= \lambda m + m \sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x})$$

$$m = \frac{\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n} (x_i - \bar{x})^2 + \lambda} \quad \text{---------- (2)}$$

new one added compare to original linear regression

let us go in n dimensions

x_1 x_2 -- x_n	y

(n+1) cols, m rows

$$L = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

	x_1 x_2 . x_n
m	w_1 w_2 . w_n

(coef) and no intercept

$$= (XW - Y)^T (XW - Y)$$

$$Y = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}_{m} \quad W = \begin{bmatrix} w_0 \\ w_1 \\ \cdot \\ w_n \end{bmatrix}_{n \times 1} \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{1n} \\ 1 & x_{22} & x_{22} & x_{2n} \\ \vdots & & & \\ 1 & x_{m1} & x_{m2} & x_{mn} \end{bmatrix}$$

$L = (XW - Y)^T (XW - Y)$ for normal.

Now, let add λ. term (regularization)

$$L = (XW - Y)^T (XW - Y) + \lambda \|W\|^2$$

$$\begin{cases} \lambda w_1^2 + \lambda w_2^2 - \lambda w_n^2 \\ \lambda (w_0^2 + w_1^2 - w_n^2) \end{cases}$$

$$[w_0 \; w_1 \; w_2 \; \cdots \; w_n] \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \cdots \end{bmatrix}$$

$$L = (Xw - y)^T (Xw - y) + \lambda w^T w$$

$$L = [(Xw)^T - (y)^T][Xw - y] + \lambda w^T w$$

$$= [w^T X^T - y^T][Xw - y] + \lambda w^T w$$

$$= (w^T X^T X w - w^T X^T y - y^T X^T w + y^T y) + \lambda w^T w$$

$a^T - b^T = (a-b)^T$

$(ab)^T = b^T a^T$

$$= w^T X^T X w - 2 w^T X^T y + y^T y + \lambda w^T w$$

$$\frac{dL}{dw} = 2X^T X w - 2X^T y + 0 + 2\lambda w = 0$$

$$= X^T X w + \lambda w = X^T y$$

$$(X^T X + \lambda I) w = X^T y$$

$$\boxed{w = (X^T X + \lambda I)^{-1} X^T y}$$

In LR it is

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

by gradient descent

vector form

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$L = (Xw - y)^T (Xw - y) + \lambda \|w\|^2$$

$$L = (Xw - y)^T (Xw - y) + \lambda w^T w$$

we know that we have $w_0, w_1 \ldots w_n$

$$w_0 = w_0 + \eta \frac{\partial L}{\partial w_0}$$

$$w_1 = w_1 + \eta \frac{\partial L}{\partial w_1}$$

$$\vdots$$

$$w_n = w_n + \eta \frac{\partial L}{\partial w_n}$$

$$w_{new} = w_{old} - \eta \frac{\Delta L}{\Delta w}$$

Gradient $\begin{bmatrix} \frac{\partial L}{\partial w_0} \\ \frac{\partial L}{\partial w_1} \\ \vdots \\ \frac{\partial L}{\partial w_n} \end{bmatrix}$

$$L = (Xw - Y)^T (Xw - Y) + \lambda w^T w$$

$$= \frac{1}{2}(w^T X^T - Y^T)(Xw - Y) + \frac{1}{2}\lambda w^T w$$

$$= \frac{1}{2}\left[w^T X^T X w - \cancel{Y^T X w} - w^T X^T Y - Y^T X w + Y^T Y \right] + \frac{1}{2}\lambda w^T w$$

$$= \frac{1}{2}\left[w^T X^T X w - \cancel{2 Y^T X w} + Y^T Y \right] + \frac{1}{2}\lambda w^T w$$

$$\left(2 w^T X^T Y\right)$$

$$\frac{\partial L}{\partial w} = \frac{1}{2}\left[2 X^T X w - \cancel{2 X^T Y} \right] + \frac{1}{2} \cdot 2(\lambda w)$$

$$\left(2 X^T Y\right)$$

$$\frac{\partial L}{\partial w} = X^T X w - \cancel{X^T Y} + \lambda w = \boxed{X^T X w - X^T Y - \lambda w}$$

$$\left(X^T Y\right)$$

$w = [0 \; 1 \; 1 \; \cdots]$

Start point

in epoch

$$w = w - \eta \frac{\partial L}{\partial w}$$

effects of ridge regression.

1) affect on coefficient

on higher value of alpha, coefficients goes closer to 0, but not exact 0.

2) higher values are impacted more

ex say,

Linear regression

$w = m_1 + m x_1 + m x_2$

high med low.

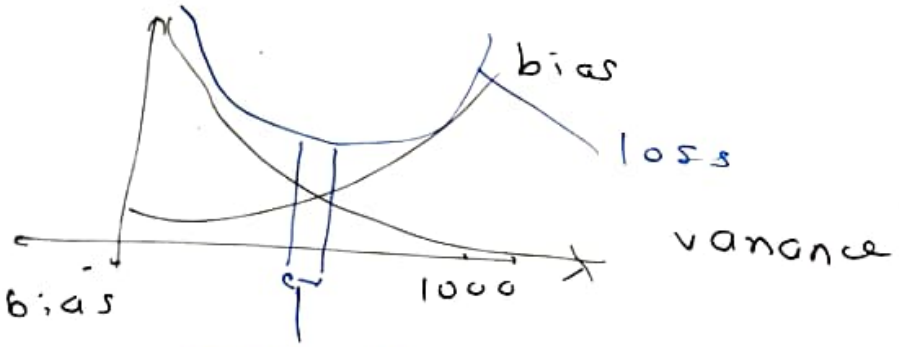
if we take ridge & go on inc value of α, what will impact?

Higher values of coefficients on higher values goes closer to 0.

3) Bias - variance tradeoff

if

	bias	model	variance
x	less ~ 0	↓	overfit ↑
x	greater	↑	underf ↓



take this value

4) Impact of loss function

$$L = \sum_{i=1}^{n} (y_i - \hat{y})^2 + \lambda \|w\|^2$$

x.y s m,b ⟶ b is constant

$$L = \sum_{i=1}^{n} (\hat{y} - mx)^2 + \lambda m^2$$

Now, the minimas,
As we are going to λ higher, It is moving the minima left side, means less and less minima, closer to orgin. Thus our loss function also.

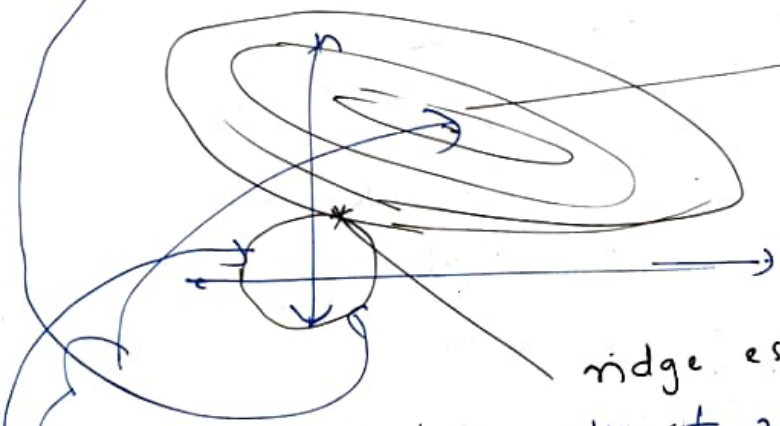
r) Why "ridge" name?
for this, we need to study hard constraint ridge regression.
but let say, we have two eqns, for loss funn, & let say

$$L = \underbrace{MSE}_{\textcircled{1}} + \underbrace{\lambda \|w\|^2}_{\textcircled{2}}$$

$$\underbrace{(y_i - \hat{y}_i)^2}_{\textcircled{1}}$$

$$\hat{\Sigma}\left(y_i - (\beta_0 + \beta_1 x_1 + \beta_2 x_2)\right)^2$$

$$+ \lambda(\beta_1^2 + \beta_2^2)$$



Tip
apply ridge for inputs >2

for a certain value of radious (hyperparam)
This is 3D → 2D plot projected.
if we dont have regularization, then our soln should be this one

loss min fn β_1 & β_2

But since we need to both minimize terms. Our solution will be on boundries of both curves, (*) marked. somewhere still it is by ridge of that circle, so, this is called ridge regression.

Lasso regression

L1 regularization

In ridge regression

$$L = \sum_{i=1}^{n} (y_i - \hat{y})^2 + \lambda \|w\|_2^2$$ (L2) (2nd norm) (square)

In lasso regression

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda |w|$$ (L1) (first norm) (mod)

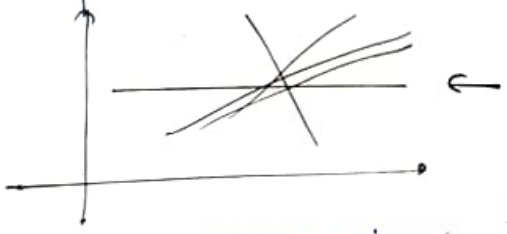
↑ absolute

In case if we increase λ alot, can underfit.
In case, if we decrease λ alot, can overfit.

$\lambda > 0$

if $\lambda = 0$, then it becomes linear regression

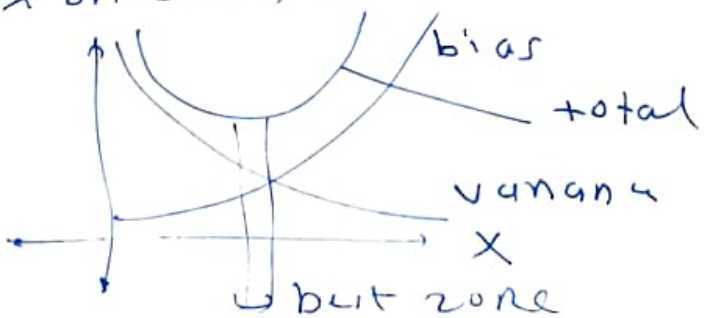
The first point to be noted is, unlike ridge regression, we can obtain coefficients equals to 0.



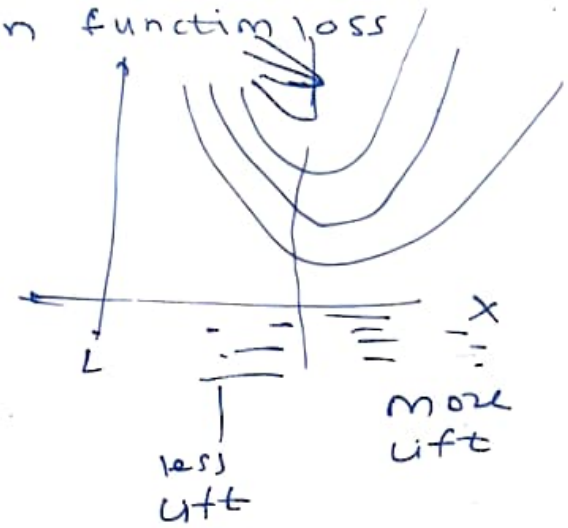
So, In higher dimensions, There is alot a call of overfitting. If we apply ridge, we would have to consider columns also who are not much important, due to their closer significance to 0. But if we apply lasso, the coefficient of less important features become 0.

Key points

1) Feature selection
2) higher coef's impact more
3) λ on bias & varnce



a) on function loss



At high values of λ, it becomes 0. but will not go behind 0.

Why sparsity in lasso?
I mean why some values become 0?
why lasso can do this & why dont ridge?

Simple example,

$x, y \rightarrow y = mx + b$, $b = \bar{y} - m\bar{x}$ (same)

Simple linear $= m = \dfrac{\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$

ridge $= m = \dfrac{\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n} (x_i - \bar{x})^2 + \lambda}$

let us drive for lasso.

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda |m|$$

$$L = \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})^2 + \lambda |m|$$

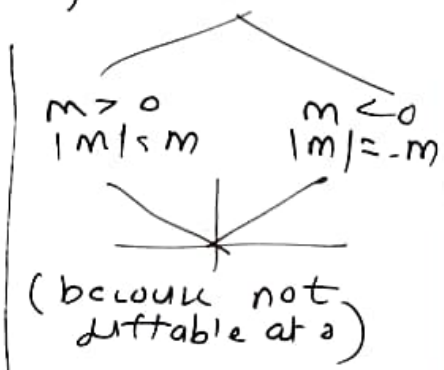
for, $|m| = m$ (positive)

$$L = \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})^2 + \lambda m$$

$$\frac{\partial L}{\partial m} = 2 \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})(-x_i + \bar{x}) + \lambda$$

$$= -2\left[(y_i - \bar{y}) - \frac{m}{m}(x_i - \bar{x})\right](x_i - \bar{x}) + \lambda = 0$$

$$= -\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x}) + m \sum_{i=1}^{n} (x_i - \bar{x})^2 + \lambda = 0$$



3. $\sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum (y_i - \bar{y})(x_i - \bar{x}) - \lambda$

$$m = \frac{\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x}) - \lambda}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \quad \therefore m > 0 \quad \text{---(1)}$$

for $m = 0$,

$$m = \frac{\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \quad \text{(linear)} \quad \text{---(2)}$$

for $m < 0$,

$$L = \sum_{i=1}^{n} (y_i - \bar{y})^2 - \lambda m$$

$$L = \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})^2 - 2\lambda m$$

$$\frac{\partial L}{\partial m} = 2 \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})(x_i - \bar{x}) - 2\lambda$$

$$\frac{\partial L}{\partial m} = 2 \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})(x_i - \bar{x}) - 2\lambda = 0$$

$$0 = -2 \sum_{i=1}^{n} \left[(y_i - \bar{y}) - m(x_i - \bar{x}) \right] (x_i - \bar{x}) - 2\lambda = 0$$

$$= -2 \sum_{i=1}^{n} \left[(y_i - \bar{y})(x_i - \bar{x}) - m(x_i - \bar{x})^2 \right] - 2\lambda = 0$$

$$= m = \frac{\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x}) + \lambda}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \quad \text{---(3)}$$

According to formula,

if $\frac{\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x})}{①} - \lambda$, travels until,

└ This and └ this become equ

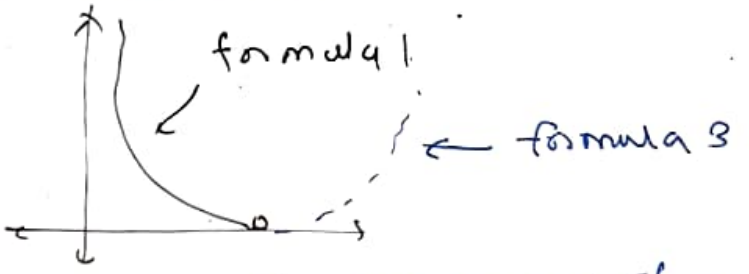
At negative, we add λ. (?)

So, If we decrease on and on, (Increase) λ, then algorithm stop there, due to bad case situation.

means. In case of possitive slope

if $$\frac{\sum_{i=1}^{n}(y_i-\bar{y})(x_i-\bar{x})-\lambda}{\sum_{i=1}^{n}(x_i-\bar{x})^2} = m$$

if we increase λ, numerator start decreasing see the black line.



as soon as for the terms of,

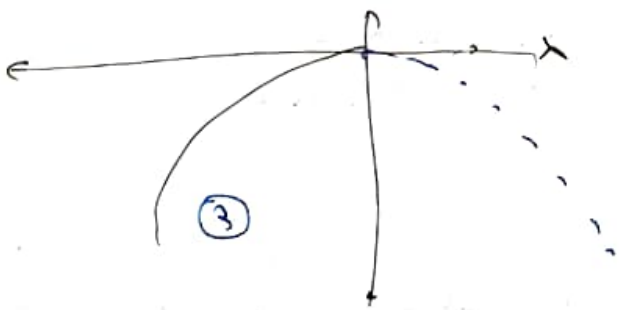
$$\sum_{i=1}^{n}(y_i-\bar{y})(x_i-\bar{x}) \leq \lambda,$$

The formula changes & becomes (3), Thats why, It dont goes down at 0, and start increasing again

for negative slope

$$\frac{\sum_{i=1}^{n}(y_i-\bar{y})(x_i-\bar{x})+\lambda}{\sum_{i=1}^{n}(x_i-\bar{x})^2} = m$$

if we ~~decrease~~ increase λ and then if I decrease λ, It goes towards 0.



as soon as for m term of

$$\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y}) \leq \lambda,$$

the slope comes +ve & here (3) thats why, It goes up at 0. so algo. dont stops it.

so why lasso create sparsity? why equal to 0?

λ comes in numerator in lasso
λ comes in denominator in ridge

so It is able to ride to 0.
so it is not able to ride to 0
so It close to 0 but dont get 0.

Elastic Net regression

If we don't know, In a large dataset, for a column in it, It's importance to predict.
In short, we confuse to choose between ridge & lasso,
we use elasticnet regression.

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + a||w||^2 + b||w||$$

In sklearn,
we get hyperparameters,

$\lambda = a + b$

$l1\ ratio = \frac{a}{a+b}$

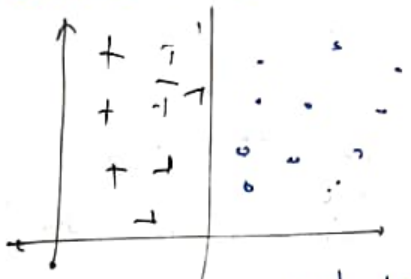
we decide a and b from here,
so, usually, λ 1 & l1 ratio 0.5 by def.
$\therefore a = 0.5, b = 0.5$.

means 50% ridge & 50% lasso.
if l1 ratio is ~~90~~ 0.9
means 90% ridge & 90% lasso
If our dataset are multi-co-linear columns alot,
2 input cols depends on each other.
we should use elasticNet regression.

Logistic regression

Two approaches
probabilistic
geometric

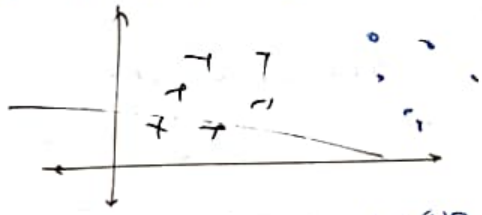
scenario & requirements
we should have linearly separable data
or almost linear separable data



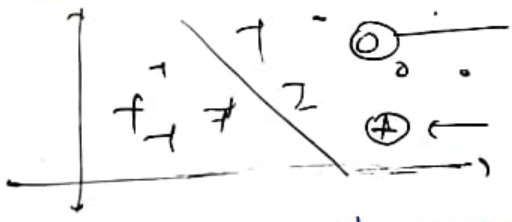
General form eqn
$ax+by+c=0$

Now we just need to find eqn line that can classify both data in parts.

Preption Trick
This algorithm does like this.
It starts with first random a, b & c.



Then It in a loop select a random point, and asks it if it is correct directed on the line, towards the line.
if point, say yes, then It will do nothing.
But if it is not,
It have to move such that I can take that point on right side.

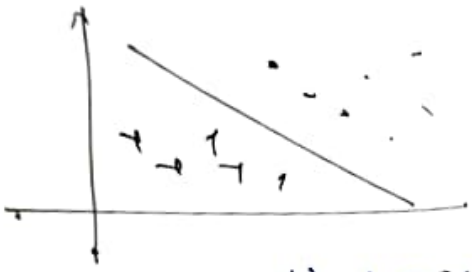


In first ask, you will not do anything
In second one, you will come just aside of that point.

For loops. How much time to travel.
1) either a fixed no iteration
2) until a converse.

How this transformation a,b,c does to move line?

First of all, How I will classify ? ie how I know side of line, we say +ve & -ve side?



so, How I will know that point direction, ie it is the or -ve?

well, I know line eqn, we it to

$ax+by+c=0$

and we say, I have a point (10,20)

so, I will put

$a(10)+b(20)+c=0$

if it is >0, +ve region.

if it is <0, -ve region

if it is 0, on the line

The transformation changes comes from changing a,b & c.

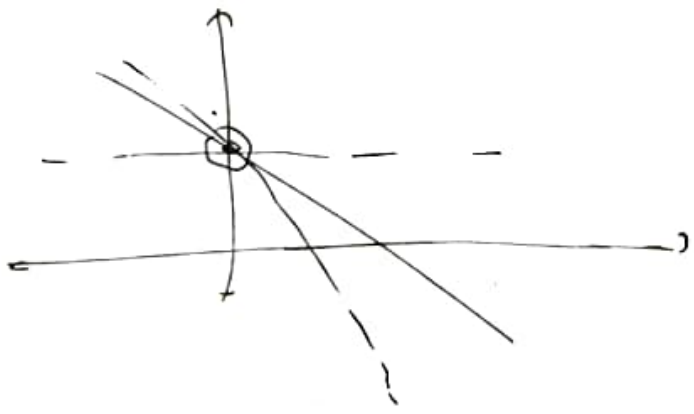
if we change C. It move like this

(parallel)

inc c → down

dec c → ~~down~~ up

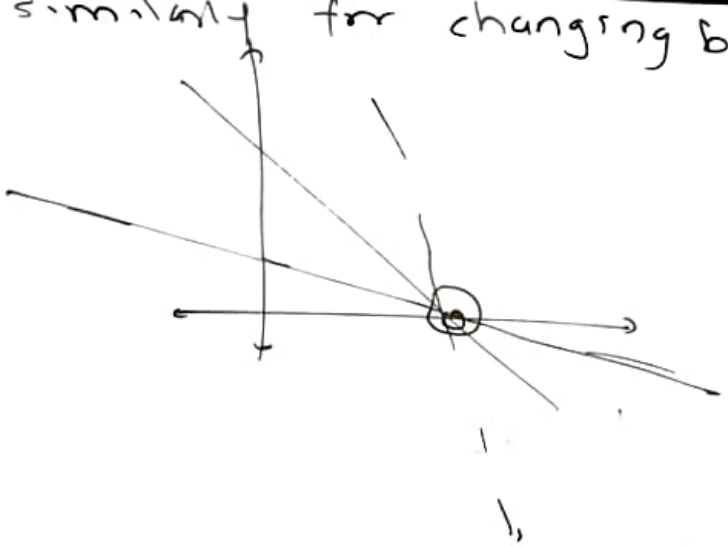
If I change a, I rotate on axis



dec → down

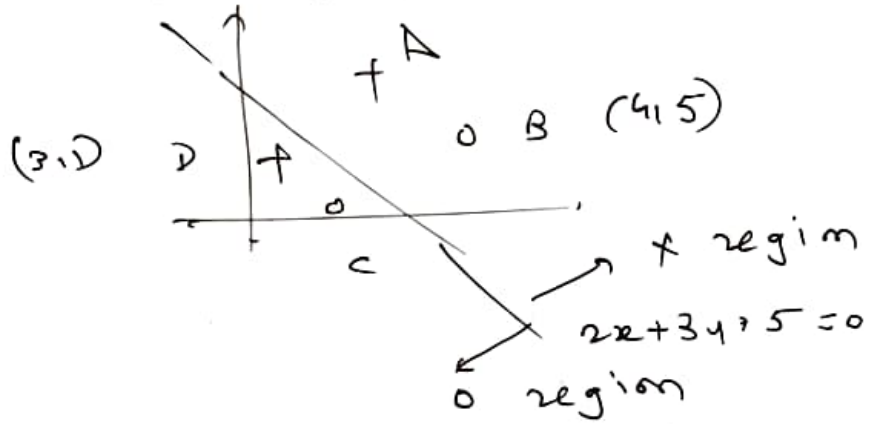
inc → up

similarly for changing b,



Now we know how transformation are getting applied.

But how will I know how to & what to apply?



point A → correct else incorrect
point C → correct
So, I will first ark point B.
It is not. So, I wish to move there
So, what preceptn trick is,
we say, I have line
(2, 3, 5)
I will consider 1 at end of point.
(4 5 1)
subtract to move towards +ve dire
if a negative pt is in pos region (-2, -2, 4)
this, $-2x - 2y + 4 = 0$, will move point B in negative direction.
we say, point D,
(2, 3, 5)
at our pos pt is in neg xy (3, 3, 1)
add to move in negative.
$3x + 5y + 6 = 0$

But In ML,
we can't do this kind of large transforms.
So, To control, we use learning rate (2)
we multiply each coef in point & then
add pow- to coefs of line / subtract.
i.e, 2 3 5
0.02 0.02 0.01
like this.

new coef = old coef − η × (cord)

so change happens gradually.

Algorithm:

	cgpa	Ig	placed
predict →	7.5	61	1
	5.9	109	1
	7.0	61	0

$ax + by + c = 0$

↓

$w_0 + w_1x_1 + w_2x_2$

$w_0 = c, \quad w_1 = A, \quad w_2 = B$

$w_0x_0 + w_1x_1 + w_2x_2 = 0$

$(\because x_0 = 1)$

$$\sum_{i=0}^{2} w_i x_i = 0$$

say this is my model.
But how I will predict?

So, we put
$w_0(1) + w_1(7.5) + w_2(61) = 0$ n

So, if this number n,
if $n \geq 0$ predict 1.
if $n < 0$ predict 0

$$\begin{bmatrix} w_0 & w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = n$$

$n \geq 0 \rightarrow 1$
$n < 0 \rightarrow 0$

algorithm

epoch define

define learning rate

for i in range (epoch)

random select a point.

check if x_i(row) $\in$ Negative region

$\sum_{i=1}^{n} w_i x_i \geq 0$. (according to model)

(so, it is a negative point, stuck in +ve region)

$w_{new} = w_{old} - \eta[x_0\ x_1\ x_i]$ (x_i)

check if x_0(row) $\in$ positive region

$\sum_{i=1}^{n} w_i x_i < 0$ (according to model)

(so, It is a positive point, stuck in -ve region)

$w_{new} = w_{old} + \eta(x_i)$

means, simply

$x_i \in N$. $w_i < 0$ ✓ } No change

$x_i \in P$ $w_i \geq 0$. ✓

simplifying algorithm!

only one condn instead of two

rather than this,

instead {

$x_i \in N$ and $\sum w_i x_i \geq 0$

$w_n = w_{old} - \eta x_i$

$x_i \in P$ and $\sum w_i x_i < 0$

$w_n = w_{old} + \eta x_i$

for i in (epoch)

select random row

$w_n = w_o + 2(y_i - \hat{y}_i) x_i$

(update rule directly)

if model & I or more this with 0.

$\therefore w_n = w_o + 0$. (same) (no update)

if model & I miss match

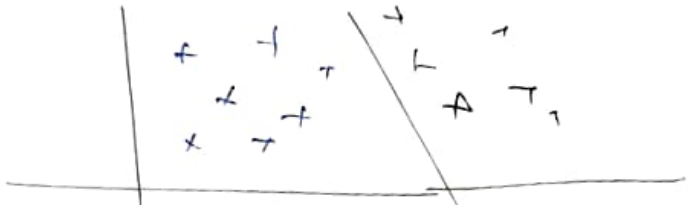
this will $\neq 0$

$w_n = w_o \pm (\eta)(y_i - \hat{y}_i)$

(change occurs)

act	pred	$y_i - \hat{y}_i$	mark
0	0	0	✓
0	1	-1	✗
1	0	+1	✗
1	1	0	✓

Issue with this

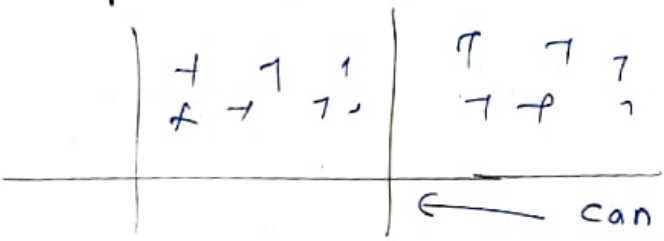


cant handle test data more efficiently

line by algorithm

It stops as soon as it classify all points. But it is not more practical.

more practical should be



← can handle more efficiently.

upgraded algorithm

until now, ~~co~~ missclisified points used to attract (pull) towards it.

Now we will change the approach

The corectly classified points will send long the line. (push) from it. and also mis classified point - each and every point

miss - pull

corect - push

The magnitude of pushing / pulling line will depend on the distance.

point	action	magnitude	distance
misclass	pull	less	near
Missclass	pull	high	long
corectclass	push	~~less~~ high	~~less~~ near
corectclass	push	low	long

so this is the case that will help us.

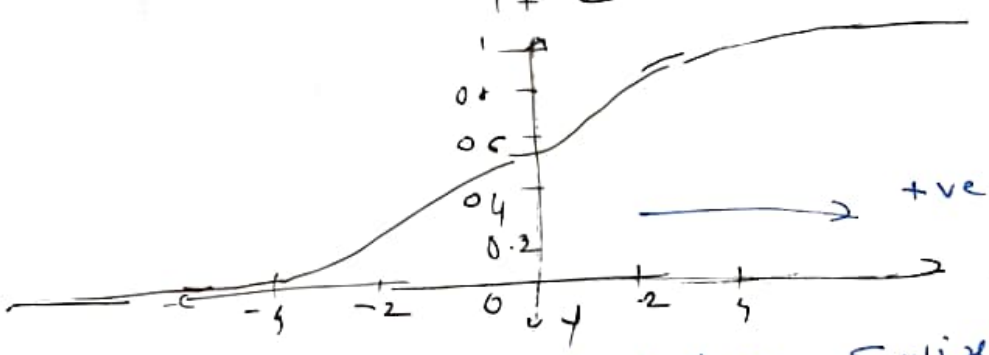
$$w_n = w_o + \eta (y_i - \hat{y}_i) x_i$$

y_i	$\hat{y}_i$	$y_i - \hat{y}_i$
1	1	0
0	0	0
1	0	1
0	1	1

if we wish to change in both cases.
avoild to create $(y_i - \hat{y}_i) = 0$
So mean $(y_i - \hat{y}_i) \neq 0$;
we can change only y_i , can't y.
so line eqn of 2-D can be
$w_o + w_1 x_1 + w_2 x_2 = 0$
I have to change step function
so we have to understand signoid

signoid function

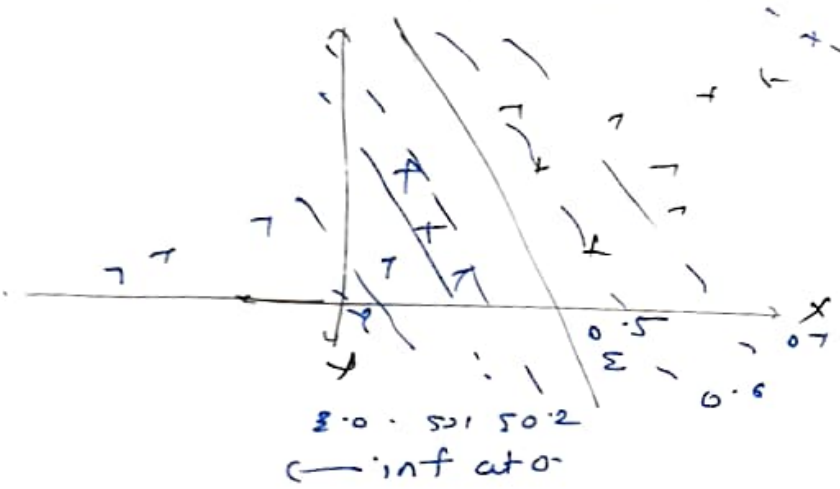
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



so we were calculating $\sum w_i x_i$

this
- if positive no $\geq 0.5 \rightarrow 1$
- if negative no $< 0.5 \rightarrow 0$

But what is disadvantage of that?



Basically it give a gradiat

← inf at o-

also signoid stats probabulty also.
it σ is 0.5, It means 50 % chance / probabuty

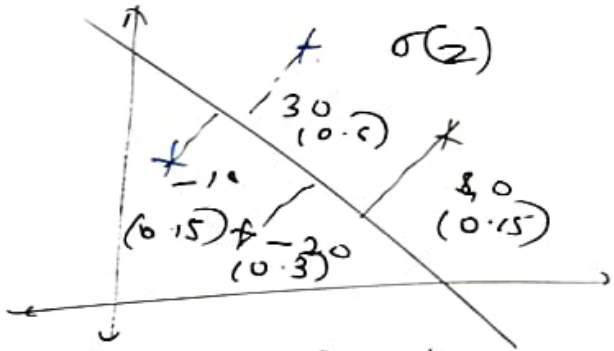
so, signoid gives a probabilitic interpretation.

Derivation
effect of signoid function

$$w_n = w_o + 2(y_i - \hat{y}_i)x_i$$

$$\hat{y}_i = \sigma(z) \quad \text{(signoid)}$$

$$z = \sum w_i x_i$$



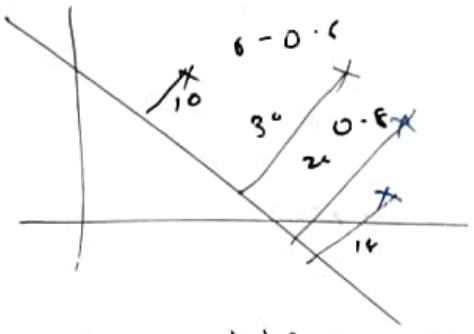
If we put large one, go towards 1.
if we put greatry one go towards 0

y_i	$\hat{y}_i$	$y_i - \hat{y}_i$
	0.8	0.2
1		
0	0.65	-0.65
1	0.3	0.7
0	0.15	-0.15

$x_n = x_0 + \eta\, 0.2\, x_i$ ——— push down

$x_n = x_0 - \eta\, 0.65\, x_i$ ——— uft up

magnitude of pushing and pulling



y_i	$\hat{y}$	$y_i - \hat{y}$
1	0.6	0.4
1	0.8	0.2

$w_n = w_0 + \eta\, 0.4 x_i \longrightarrow x_1$

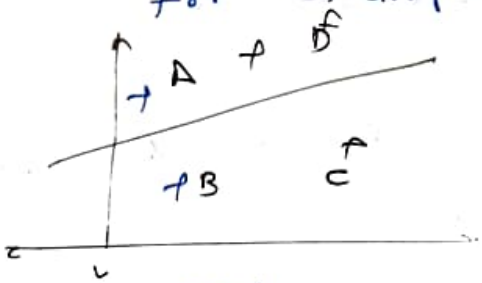
$w_n = w_0 + \eta\, 0.2 x_i \longrightarrow x_2$

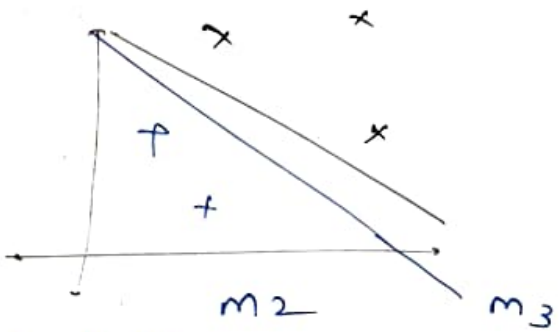
$x_1 > x_2$

In this approach, As we are towards solution, and But not to a optimal soln. and secondly we are generating points randomly. So we cannot tell if we are right there on optimal solution.

In machine learning, Things doesn't work this way. we always need a loss function. we take minima, where loss function is minimum & that coef's are our actual solution.

find a loss function for example





M1

for now, M2 is better.

But now, I can't tell if M2 and M3, which one better.

To calculate that, we need a loss function.

So, we will use maximum likelihood function these related to probability.

so, for two models,
let us calculate $\hat{y}$ first.

$\hat{y} = \sigma(z)$

$\hookrightarrow z = \sum w_i x_i$

so we will calculate ~~green~~ blue prob and black prob

points	P(blue)	P(black)
A	0.6	0.4
B	0.2	0.8
C	0.4	0.6
D	0.7	0.3

for m_1

points	P(blue)	P(red)
A	0.7	0.3
B	0.6	0.4
C	0.4	0.6
D	0.3	0.7

for m_2

maximum likelihood says to multiply all the probabilities. one of the better prob, will better model.

$m_1 = P(\text{blue}) * P(\text{Blue}) * P(\text{black}) * P(\text{blue}$

$= 0.7 \times 0.4 \times 0.4 \times 0.8 = 0.089$

$m_2 = 0.7 \times 0.6 \times 0.2 \times 0.7 = 0.176$

~~(9)~~

$m_2 > m_1$

But this way good; problamatic for large datasets.

So we will take log to handle it.

$m_1 = \log(ab) = \log(a) + \log(b)$

But still a problem.

log is always between 0 & 1 & if I calculate log between (0-1), then it is always negative

To tackle it, I will take - .

$= -\log(a) - \log(b)$

It is called cross entropy.

when I do summation of all negatived terms of logs of maximum likelihood, It is called cross entropy.

like we have to maximise ~~cross entropy~~ likelihood, we need to minimize cross entropy

we need to consider $\hat{y}$s as that blue/black points probabuty.

for a moment, can 4 for 4 points, can we do this?

$-\log(\hat{y_1}) - \log(\hat{y_2}) - \log(\hat{y_3}) - \log(\hat{y_4})$

No.

Because, We need to see probabuty of classification, of that orginal point.

So we convert it.

$$L := -y_i \log(\hat{y_i}) - (1-y_i) \log(1-\hat{y_i})$$

$=$ for point 1, target is 1. (y_1)

$-y_i \log(0.7) - (0)$

For 2nd point, target 0

$-0 - \log(1-0.2)$

for 3rd point target 1

$-0 - \log(0.4)$

for 4th point, target 0

~~$0 = (1 - \log(0.2)$~~

~~$\log(0.2)$~~

$-0 - \log\left(1 - \frac{0.6}{0.2}\right)$

loss function $=$

$$L = \sum_{i=1}^{2} -y_i \log(\hat{y_1}) - (1-y_i) \log(1-\hat{y_1})$$

total error

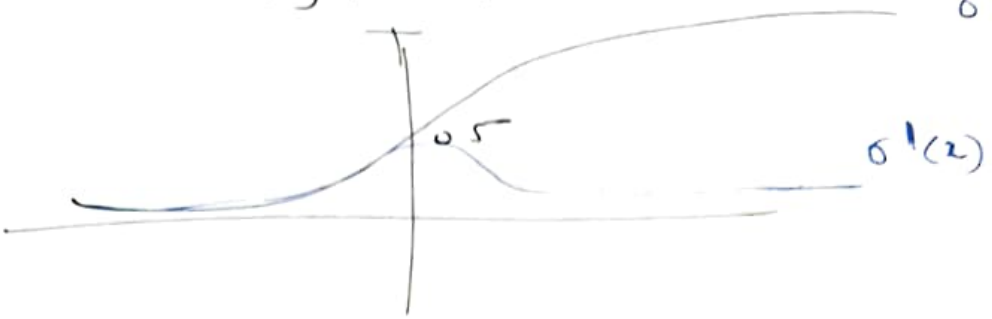
log-loss error

binary-cross entropy fun?

no closed form solution

(7)

Derivative of sigmoid function



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \frac{d}{dx}(1+e^{-x})^{-1}$$

$$= \frac{-1}{(1+e^{-x})^2} \frac{d}{dx}(1+e^{-x})$$

$$= \frac{-1}{(1+e^{-x})^2} \frac{d}{dx}(e^{-x})$$

$$= -\frac{e^{-x}}{(1+e^{-x})} \frac{d}{dx}(-x)$$

$$= \frac{e^{-x}}{(1+e^{-x})^2}$$

$$= \frac{1}{(1+e^{-x})} \cdot \frac{e^{-x}}{(1+e^{-x})}$$

$$= \sigma(x) \cdot \frac{e^{-x}}{(1+e^{-x})}$$

$$= \sigma(x)\left[\frac{1+e^{-x}-1}{1+e^{-x}}\right]$$

$$= \sigma(x)\left[\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{(1+e^{-x})}\right]$$

$$\boxed{\sigma'(x) = \sigma(x)(1-\sigma(x))} \quad \text{---} \quad (1)$$

Gradient - Descent approach

dataset

$$\begin{matrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{matrix} \qquad \begin{matrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{matrix}$$

$$[w_0 \quad w_1 \quad w_2 \quad w_3 \quad \cdots \quad w_n]$$

coefficients

for first one,

$$\sigma(w_0 + w_1 x_{11} + w_2 x_{12} + w_3 x_{13} \cdots w_n x_{1n}) = \hat{y}_1$$

$$\sigma(w_0 + w_1 x_{21} + w_2 x_{22} + w_3 x_{23} \cdots w_n x_{2n}) = \hat{y}_2$$

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} \sigma(w_0 + w_1 x_1 + w_2 x_{12} \cdots w_n x_{1n} \\ \sigma(w_0 + w_1 x_{21} + w_2 x_{22} \cdots w_n x_{2n} \\ \sigma(w_0 + w_1 x_{m1} + w_2 x_{m2} \cdots w_n x_{mn} \end{bmatrix}$$

$$= \sigma \begin{bmatrix} w_0 + w_1 x_{11} + w_2 x_{12} \cdots w_n x_{1n} \\ w_0 + w_1 x_{21} + w_2 x_{22} \cdots w_n x_{2n} \\ w_0 + w_1 x_{m1} + w_2 x_{m2} \cdots w_n x_{mn} \end{bmatrix}$$

$$= \sigma \left(\underbrace{\begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & \cdots & x_{2n} \\ 1 & x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}}_{W} \right)$$

$$\hat{y} = \sigma(XW)$$

from the equation,

$$L = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log(\hat{y_i}) + (1-y_i)\log(1-\hat{y_i})\right]$$

$$= -\frac{1}{m}\left[\underbrace{\sum_{i=1}^{m} y_i \log(\hat{y_i})}_{①} + \underbrace{\sum_{i=1}^{m}(1-y_i)\log(1-\hat{y_i})}_{②}\right]$$

subject to ①,

$$S_1 = \sum_{i=1}^{m} y_i \log(\hat{y_i})$$

$$= y_1 \log \hat{y_1} + y_2 \log \hat{y_2} + y_3 \log \hat{y_3} \cdots y_m \log \hat{y_m}$$

$$= [y_1\ y_2 \cdots y_m] \begin{bmatrix} \log \hat{y_1} \\ \log \hat{y_2} \\ \log \hat{y_m} \end{bmatrix}$$

$$= [y_1\ y_2 \cdots y_m] \log\left(\begin{bmatrix} \hat{y_1} \\ \hat{y_2} \\ \hat{y_m} \end{bmatrix}\right)$$

$$= Y \log \hat{Y}$$

$$= Y \log(\sigma(XW)) \quad \text{——— ①}$$

$$S_2 = \sum_{i=1}^{m}(1-y_m)\log(1-y_m)$$

$$= (1-y_1)\log(1-\hat{y_1}) + (1-y_2)\log(1-\hat{y_2}) \cdots + (1-y_m)\log(1-\hat{y_m})$$

$$= [(1-y_1)(1-y_2)\cdots(1-y_m)] \begin{bmatrix} \log(1-\hat{y_1}) \\ \log(1-\hat{y_2}) \\ \log(1-\hat{y_m}) \end{bmatrix}$$

$$= \begin{bmatrix} (1-y_1) & (1-y_2) & \cdots & (1-y_m) \end{bmatrix} \left(\log \begin{bmatrix} 1-\hat{y}_1 \\ 1-\hat{y}_1 \\ \vdots \\ 1-\hat{y}_m \end{bmatrix} \right)$$

$$= (1-y) \log(\sigma(1 - xw)) \quad \text{---------- (2)}$$

from s1 and s2

$$L = \frac{-1}{m} \left[y \log \hat{y} + (1-y) \log(1-\hat{y}) \right]$$

where $\hat{y} = \sigma(x, w)$

loss in matrix form

Gradient descent approach

define w and epoch

for i in epoch:

$$w_n = w_0 \pm \alpha \frac{\Delta L}{\Delta w} \quad \text{(gradient)}$$

$$\frac{\Delta L}{\Delta w} = \left[\frac{\partial L}{\partial w_0} \cdot \frac{\partial L}{\partial w_1} \cdots \frac{\partial L}{\partial w_n} \right]$$

$$L = \frac{1}{m} \left[y \log \hat{y} + (1-y) \log(1-\hat{y}) \right]$$

$$\frac{dL}{dw} = (1)^y \cdot \frac{d}{dw} \log \hat{y} = \frac{y}{\hat{y}} \frac{d}{dL}(\hat{y})$$

$$= \frac{y}{\hat{y}} \frac{d}{dx}(\sigma(xw))$$

$$= \frac{y}{\hat{y}} \sigma(wx)\left[1 - \sigma(wx)\right] \frac{d}{dw}(wx)$$

$= \hat{y}(1-\hat{y})x$

$= \hat{y}(1-\hat{y})x$ ——— (1)

for the second term

$= (1-\hat{y})\log(1-\hat{y})$

$\frac{\partial L}{\partial w} = \frac{\partial}{\partial w}\left[(1-\hat{y})\log(1-\hat{y})\right]$

$= (1-\hat{y})\frac{\partial}{\partial w}\left[\log(1-\hat{y})\right]$

$= \frac{(1-\hat{y})}{(1-\hat{y})}\frac{\partial}{\partial w}(1-\hat{y})$

$= \frac{(1-\hat{y})}{(1-\hat{y})}\left[-\frac{\partial \hat{y}}{\partial w}\right]$

$= \frac{(1-\hat{y})}{(1-\hat{y})}\left[-\frac{\partial}{\partial w}(\sigma(xw))\right]$

$= -\left(\frac{1-\hat{y}}{1-\hat{y}}\right)\left[\sigma(xw)\cdot(1-\sigma(xw))\right]\cdot\frac{\partial}{\partial w}[xw]$

$= \frac{-(1-\hat{y})\hat{y}}{(1-\hat{y})}x$

$= -\hat{y}(1-\hat{y})x$ ——— (2)

$\frac{\partial L}{\partial w} = -\frac{1}{m}\left[\hat{y}(1-\hat{y})x * -\hat{y}(1-\hat{y})x\right]$

$= -\frac{1}{m}\left[\hat{y}(1-\hat{y}) - \hat{y}(1-\hat{y})\right]x$

$= -\frac{1}{m}\left[\hat{y} - \hat{y}\hat{y} - \hat{y} + \hat{y}\hat{y}\right]x$

$\boxed{\frac{\partial L}{\partial w} = -\frac{1}{m}\left[y-\hat{y}\right]x}$

So, update frequency can be

$$w = w_0 + \eta \frac{1}{m}(y - \hat{y})x$$

$$w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}_{(n+1)(1)}, \quad x = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{12} & x_{22} & \cdots & x_{2n} \\ 1 & x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}_{(m)\times(n+1)}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{(m)\times(1)}, \quad \hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix}_{(m)\times(1)}$$

$$w = w_0 + \frac{\eta}{m}(y - \hat{y})(x)$$

$$(n+1,1) = (n+1,1) + (m,1)(m,n+1)$$

$$(n+1,1) = (1,n+1) + (1,m)(m,n+1)$$

$$= (1,n+1)$$ proved and verified

Classification Matrix

Accuracy score

$$\frac{\text{Total no of correct predictions}}{\text{Total no of predictions}}$$

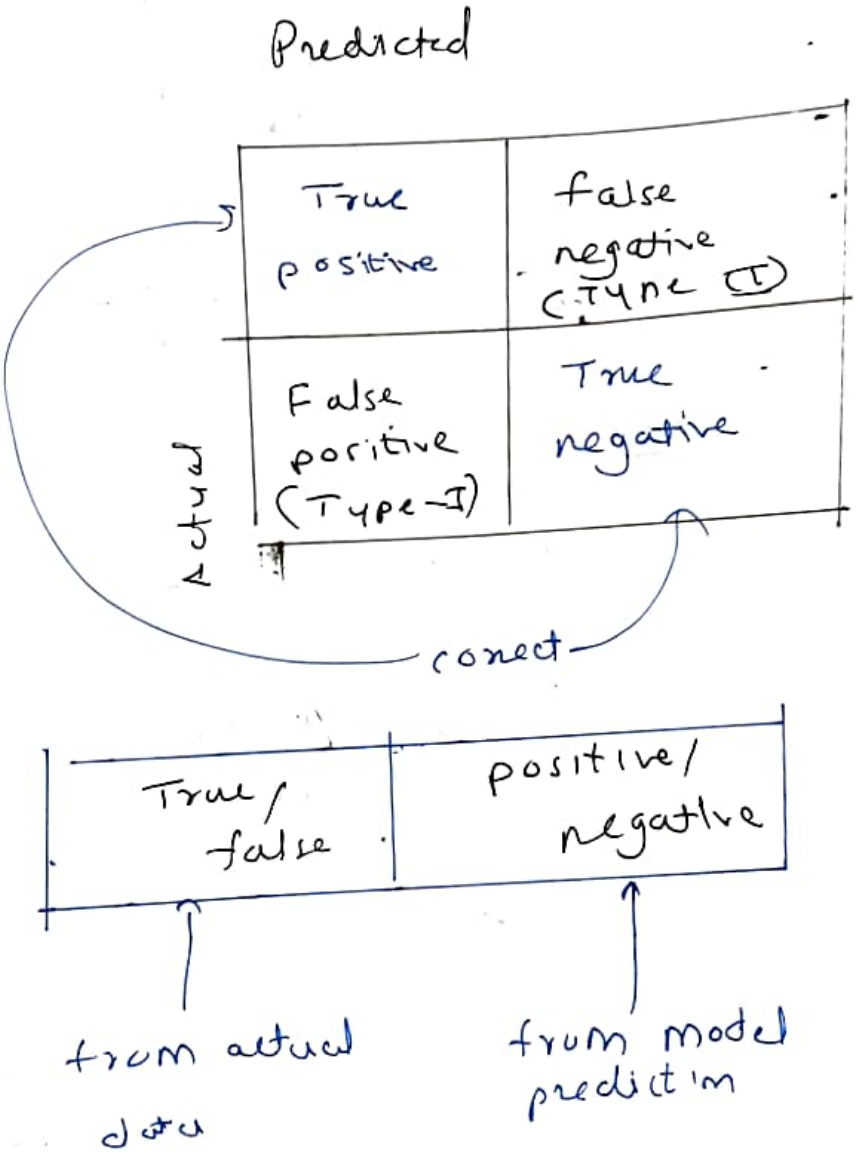
How much should accuracy score?
Depends on data & problem.

example 1:
A model predicting by 99% accuracy that a patient of cancer. We cannot deploy it due to one life.

example 2
If we want to predict that a costumer this weekend will order food or not. In case, In 80% accuracy will fine.

But this is not much accurate. matrix
It is not like best,
Because a number, But not tells type
and detail exactly.
To solve this, We can use confusion
matrix.

confusion matrix



confusion matrix → accuracy score

$$\text{Accuracy score} = \frac{\text{True pos} + \text{True neg}}{\text{True pos} + \text{True neg} + \text{False pos} + \text{false neg}}$$

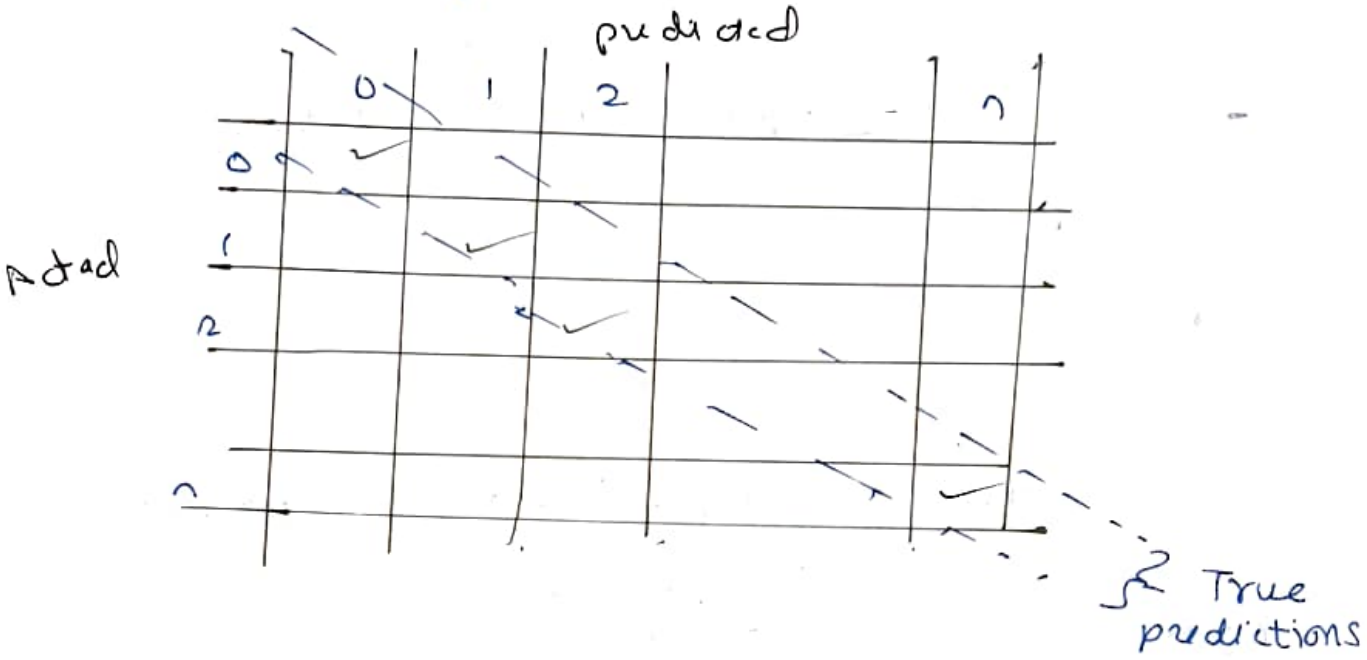
Type I - error

No of false positive

Type - II error

No of false negative

for multiple

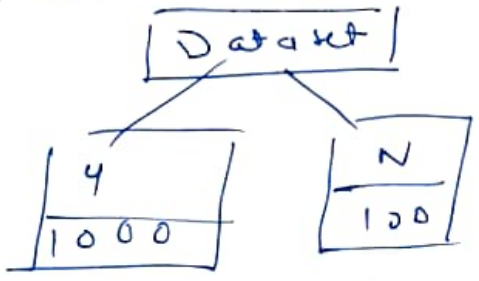


$$\text{Accuracy} = \frac{\text{Total correct predictions (Diagonal elements)}}{\text{Total number of predictions}}$$

Accuracy can missrad too!!

Due to inbalance dataset,

for example



Inbalanced

Accuracy give wrong results

In case
As a manager you have two models.

model 1	sent to spam	not sent to spam
spam	100	170
not spam	30	700

model 2	sent to spam	Not sent to spam
spam	100	190
not spam	10	700

most safe model is, model 2.
as due to false negative.

Precision
The ratio of true positives to total positive

$$TP/(TP+FP)$$

Again In case
To decide as a manager to decide

model 1	Detected Cancer	Not detected cancer
has cancer	1000	200
no cancer	800	8000

model 2	Detected cancer	Not detected cancer
has cancer	1000	500
no cancer	500	8000

most safe model is, model 1

Recall
Among of all positive, how many positive detected.

$$TP/(TP+FN)$$

In case In a problem, we cant decide a that if type I error is more problematic or type II.

example → classify cat / dog

we don't know which is more dangerous.

type I — said a cat a dog.

type II - said a dog a cat.

$$\text{precision} \propto \frac{1}{\text{recall}}$$ (harmonic mean)

So F1 score

F1 score

$$= \frac{2\,(\text{prec})\,(\text{recall})}{(\text{prec}) + (\text{recall})}$$

f1 score tends always to lower side

Because to stabalize the model

multi-class

Actual \ predicted	Dog	Cat	Rabbit	Total
Dog	25	5	10	40
Cat	20	30	4	34
Rabbit	4	10	20	34
total	29	45	34	

predicted by model

Orginal number in data

So, Let now we will calculate precision

$$\text{pre dog} = \frac{\text{Actual}}{\text{predicted}} = \frac{25}{29} = 0.86$$

$$\text{pre cat} = \frac{30}{45} = 0.66$$

$$\text{pre Rab} = \frac{20}{34} = 0.58$$

$$\text{macro - precision} = \text{Average}$$

$$= \frac{0.86 + 0.66 + 0.58}{3}$$

$$= 0.70$$

$$\text{weighted precision} = \sum_{\text{class}} \left(\frac{\text{weight of class}}{\text{class}}\right)(\text{prec})$$

$$= \sum_{\text{class}} \left(\frac{\text{class actual}}{\text{total actual}}\right)(\text{prec})$$

$$= \left(\frac{40}{106}\right)(0.86) + \left(\frac{34}{106}\right)(0.66) + \left(\frac{34}{106}\right)(0.58)$$

$$= 0.71$$

Balance classes - macro

Inbalance class - weighted

$$\text{recall} = \frac{\text{Correct dogs}}{\text{actual dogs}} = \frac{25}{40} = 0.625$$

$$\text{recall dog} = \frac{\cancel{\text{actual dog}}}{\cancel{\text{Correct dogs}}}$$

$$\text{recall cat} = \frac{30}{34} = 0.882$$

$$\text{recall rab} = \frac{20}{34} = 0.588$$

macro recall

$$\frac{0.625 + 0.882 + 0.566}{3} = 0.69$$

weighted recall

$$= \left(\frac{40}{108}\right)(0.625) + \left(\frac{34}{106}\right)(0.882) + \left(\frac{34}{106}\right)(0.566)$$

$$= (0.37)(0.625) + (0.31)(0.882) + (0.31)(0.566)$$

$$= 0.6869$$

$$F_1 \text{ dog} = \frac{2 \text{ Precdog Recdog}}{\text{Precdog} + \text{recaldog}} = \frac{2 \times 0.88 \times 0.62}{0.88 + 0.62}$$

$$= \frac{1.064}{1.48} = 0.7202$$

$$F_1 \text{ cat} = \frac{2 \times 0.88 \times 0.66}{0.88 + 0.66} = \frac{1.161}{1.54} = 0.7538$$

$$F_1 \text{ rab} = \frac{2 \times 0.58 \times 0.58}{0.58 + 0.58} = \frac{0.672}{1.16} = 0.42$$

$$\text{macro } F_1 = \frac{0.7202 + 0.7538 + 0.42}{3}$$

$$= 1.511$$

weighted F1

$$= \left(\frac{40}{108}\right)(0.753) + \left(\frac{40}{108}\right)(0.7202) + \left(\frac{34}{108}\right)(0.42)$$

$$= (0.37)(0.7538) + (0.37)(0.7202) + (0.31)(0.42)$$

$$= 0.2183$$

=

Softmax regression (multinomial logistic regression)
Going ahead from binary classification.

Sample data

CGPA	IQ	Placement
		Yes
		No
		Not opt out

softmax function

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

K = no of classes
$0 <$ Individual value < 1
add probs equals to 1.

Training part

cgpa	IQ	placement (data)
-	-	0 (Y)
-	-	1 (N)
-	-	2 (Not opt)

→ onehotencoding

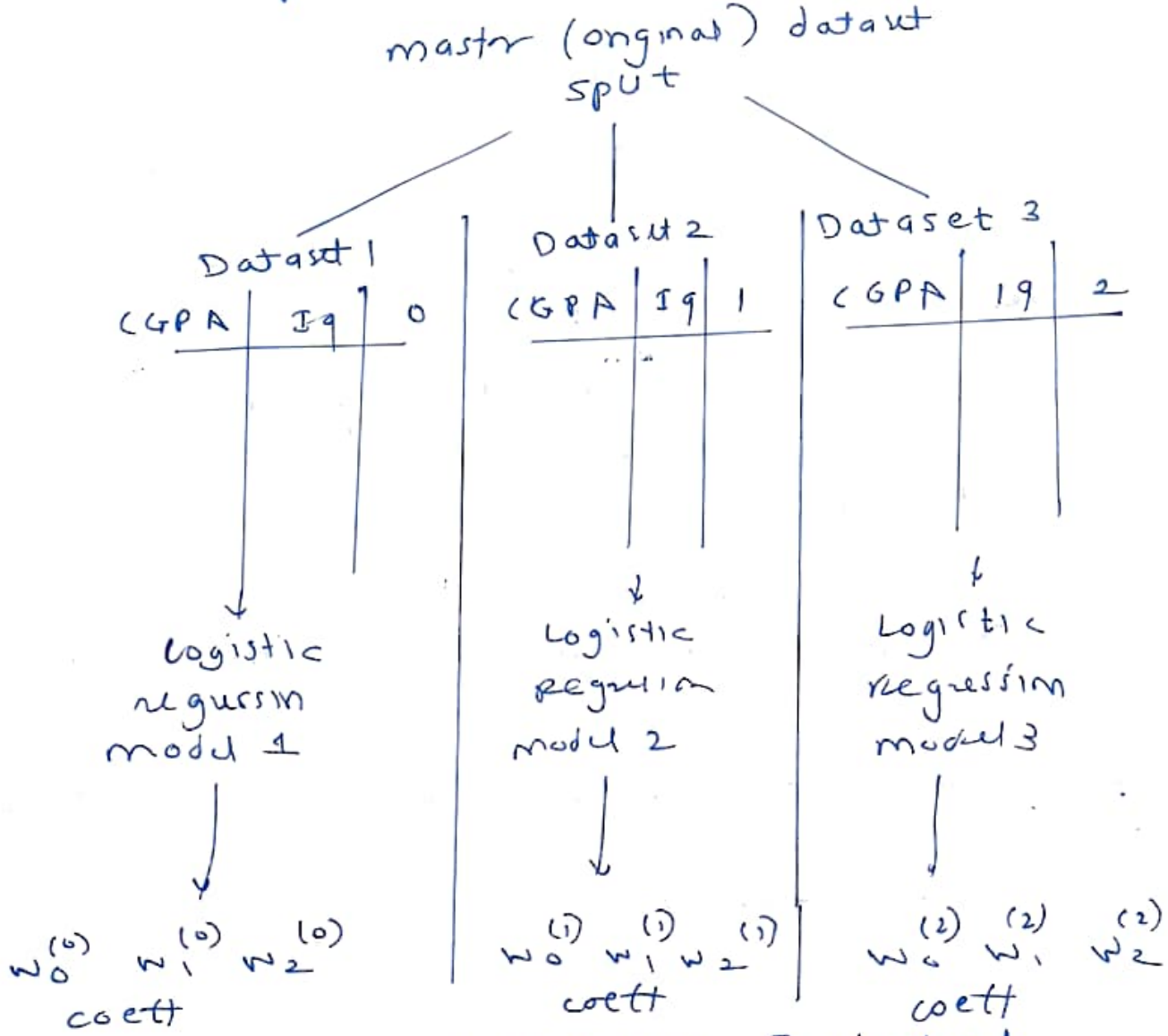
Yes - 0
No - 1
Not opt - 2

= 0	= 1	= 2
1	0	0
0	1	0
0	0	1

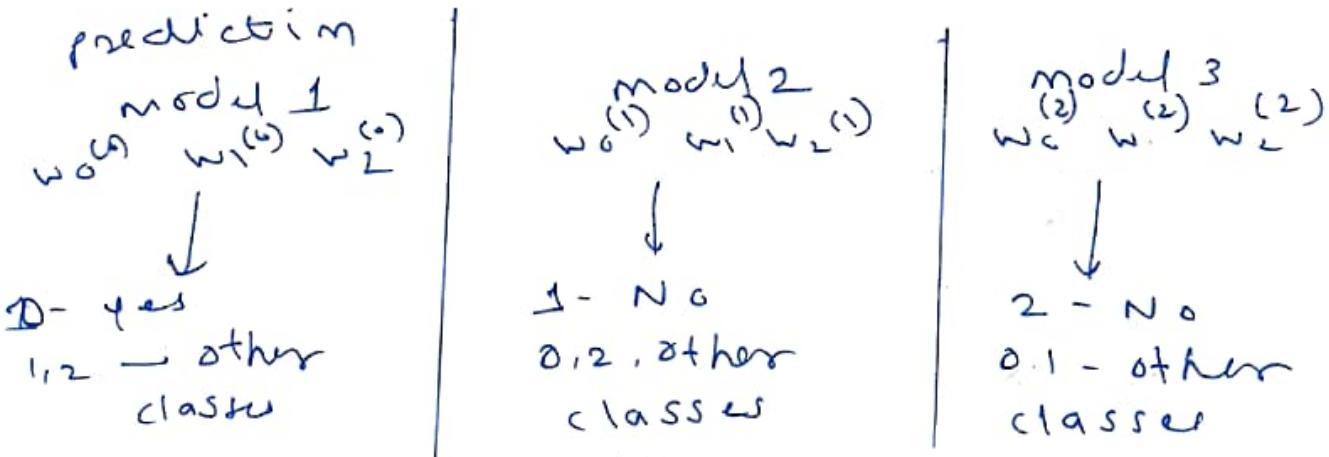
cgpa	IQ	0 (Y)	1 (N)	2 (Not opt)
-	-	1	0	0
-	-	0	1	0
-	-	0	0	1

multi to binary (0,1)

Split into 3 datasets



For just Intution, In backend process is different



prediction test, X_1, X_2.

calculating z_1

$z_1 = w_0^{(0)} + x_1 w_1^{(0)} + x_2 w_2^{(0)}$

prob for yes

calculating z_2

$z_2 = w_0^{(1)} + x_1 w_1^{(1)} + x_2 w_2^{(1)}$

prob for NO

calculating z_3

$z_3 = w_0^{(2)} + x_1 w_1^{(2)} + x_2 w_2^{(2)}$

prob for opt out

Apply sigmoid function

$$\sigma(y) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$\sigma(w) = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$\sigma(op) = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

The largest (probability) will consider (as prediction)

But THIS IS sinously time consuming!

In sicikit learn

loss function

$$L = \frac{1}{m} \sum_{i=1}^{m} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

In logistic regression,

$$L = \frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} y_k^{(i)} \log(\hat{y}_k)^{(i)}$$

In softmax.

Sample data

x_1	x_2	y	$y=0$	$y=1$	$y=2$
x_{11}	x_{12}	0	1	0	0
x_{21}	x_{22}	1	0	1	0
x_{31}	x_{32}	2	0	0	1

$$L = \frac{1}{3} \sum_{i=1}^{3} \left[\sum_{k=} \right] \left(y_k^{(i)} \right) \log \left(\hat{y}_k^{(i)} \right)$$

first row. $k = \{0, 1, 2\}$.

$$L = \frac{1}{3} \sum_{i=1}^{3} \left\{ \left[\left(y_1^{(1)} \right) \log \left(\hat{y}_1^{(1)} \right) \right] + \left[\left(y_2^{(2)} \right) \log \left(\hat{y}_2^{(2)} \right) \right] + \left[\left(y_3^{(3)} \right) \log \left(\hat{y}_3^{(3)} \right) \right] \right\}$$

$$= \frac{1}{3} \left\{ \left[\left(y_1^{(1)} \right) \log \left(\hat{y}_1^{(1)} \right) \right] + \left[\left(y_2^{(1)} \right) \log \left(\hat{y}_2^{(1)} \right) \right] + \left[\left(y_3^{(1)} \right) \log \left(\hat{y}_3^{(1)} \right) \right] \right\} + \left\{ \left[\left(y_1^{(2)} \right) \log \left(\hat{y}_1^{(2)} \right) \right] + \left[\left(y_2^{(2)} \right) \log \left(\hat{y}_2^{(2)} \right) \right] + \left[\left(y_3^{(2)} \right) \log \left(\hat{y}_3^{(2)} \right) \right] \right\} + \left\{ \left[\left(y_1^{(3)} \right) \log \left(\hat{y}_1^{(3)} \right) \right] + \left[\left(y_2^{(3)} \right) \log \left(\hat{y}_2^{(3)} \right) \right] + \left[\left(y_3^{(3)} \right) \log \left(\hat{y}_3^{(3)} \right) \right] \right\}$$

0 0 0 0 0 0

$L = (y_1^{(1)}) \log(\hat{y}_1^{(1)}) + (y_2^{(2)}) \log(\hat{y}_2^{(2)}) + (y_3^{(3)}) \log(\hat{y}_3^{(3)})$

so, this for 3 rows.

$\hat{y}_1^{(1)} = \sigma(w_1^{(1)} x_{11} + w_2^{(1)} x_{12} + w_0^{(1)})$

softmax

$\hat{y}_2^{(2)} = \sigma(w_1^{(2)} x_{21} + w_2^{(2)} x_{22} + w_0^{(2)})$

$\hat{y}_3^{(3)} = \sigma(w_1^{(3)} x_{31} + w_2^{(3)} x_{32} + w_0^{(3)})$

So, How are wts of us model.

$$\begin{bmatrix} w_1^{(1)} & w_2^{(1)} & w_0^{(1)} \\ w_1^{(2)} & w_2^{(2)} & w_0^{(2)} \\ w_1^{(3)} & w_2^{(3)} & w_0^{(3)} \end{bmatrix}$$

while applying gradient descent, I have to take derivative. wrt everyone

$\frac{\partial L}{\partial w_0^{(1)}}, \frac{\partial L}{\partial w_1^{(1)}}, \frac{\partial L}{\partial w_2^{(1)}} \dots$

9 values.

Initialize value of matrix

epochs for loop

update everyone wtt.

$w_1^{(1)} = w_1^{(1)} - \eta \frac{\partial L}{\partial w_1^{(1)}}$

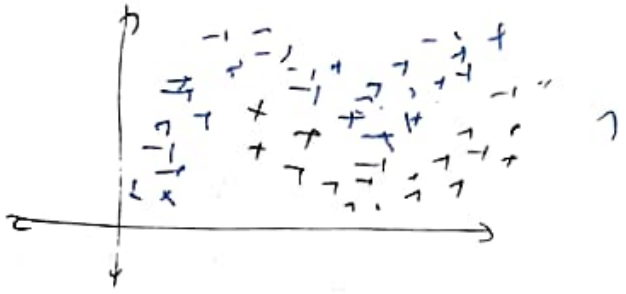
$w_2^{(1)} = w_2^{(1)} - \eta \frac{\partial L}{\partial w_2^{(1)}}$

⋮

9 classes.

polynomial Logistic regression

we can convert to polynomial features



for 2 features & 3 degree,

x_0, x_2,

we have

$x_0^0, x_2^0, x_{31}^0, x_2^1, x_1^2, x_2^2$

Output → Input columns = degree + 1

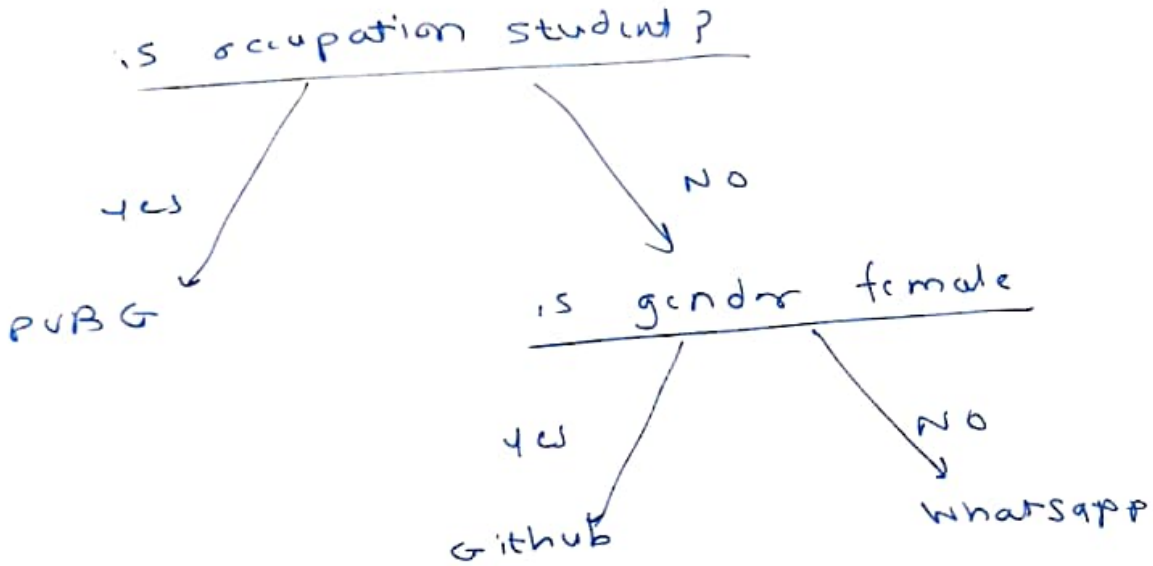
Decision tree

example

Gendr	Occupation	Suggestion
F	stu	PUBG
F	prog	Github
M	prog	whatsapp
F	prog	Github
M	stu	PUBG
M	stu	PUBG

```
if occp == student:
    print(PUBG)
else
    if gendr == female
        print(Github)
    else
        print(whatsapp)
```

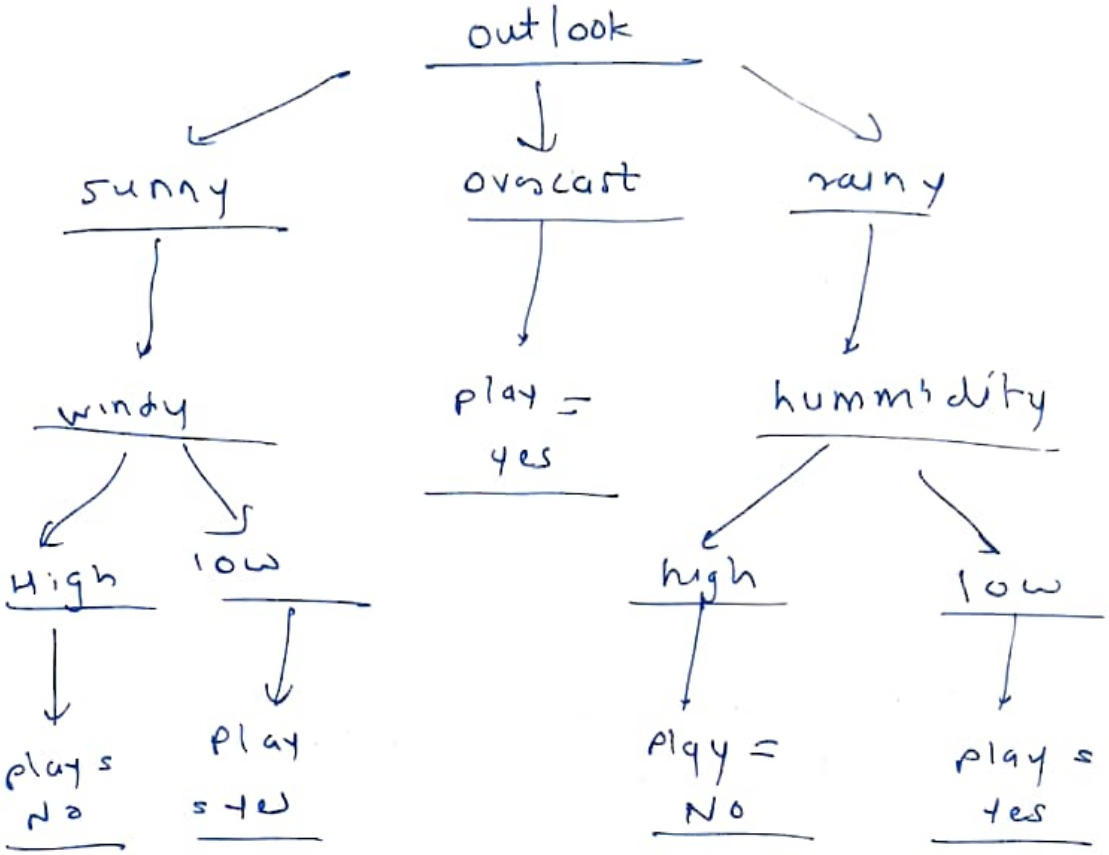
Tree



example 2

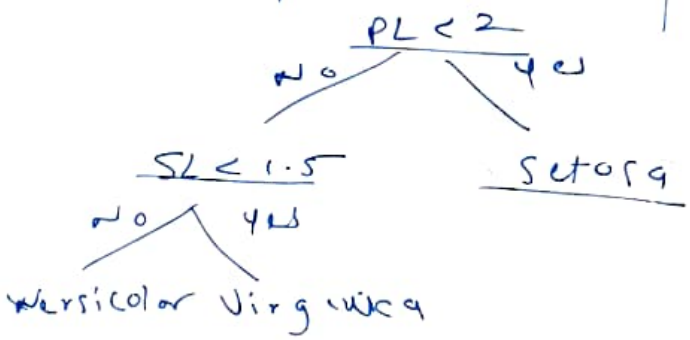
Day	Outlook	Temp	Humid	Wind	play
1	sunny	Hot	High	weak	No
2	Sunny	Hot	High	strong	No
3	Overcait	Hot	High	Wiak	Yes
4	Rain	mild	High	weak	Yes
5	Rain	cool	Nor	weak	Yes
6	Rain	Cool	Nr	strong	No
7	overcait	Cool	Nor	strong	Yes
8	Sunny	mild	High	weak	No
9	Sunny	cool	Nor	weak	yes
10	rain	mild	Nor	weak	yes
11	Sunny	mild	Nor	strong	yes
12	Overcait	mild	High	Strong	yes
13	Overcast	Hot	Nor	Weak	Yes
14	rain	Mild	high	strong	No

Tree

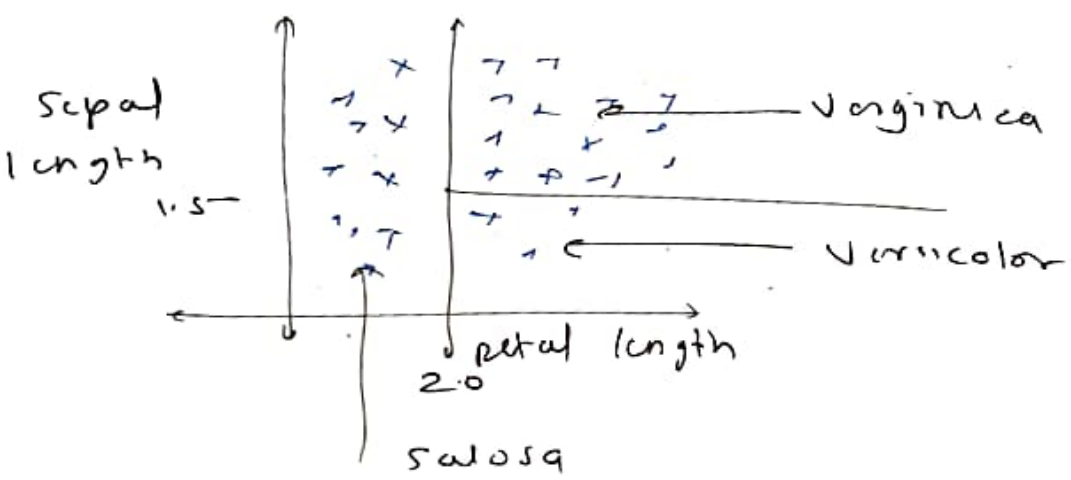


example 3

petal length	sepal length	Type
1.34	0.34	Setosa
3.45	1.85	Versicolor
1.69	0.90	Saiosa virginica
2.56	1.79	Versicolor
3.00	1.13	Setosa
1.3	0.86	



Geometric Intution



Algorithm

A training data with some feature variables and classification or regression output.

Determine 'best feature' to split data, later on also.

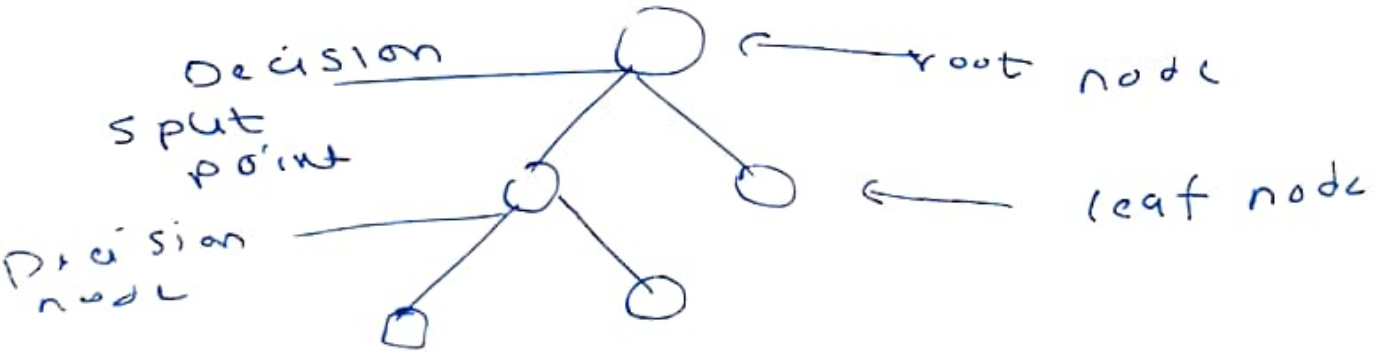
Split data in features for best subset contain correct values for feature. It is a node of tree structure.

Recursively generate nodes.

mathematically, DT's are using hyperplanes which run parallel to any axes to cut your cordinate system in hyper cuboids.

programatically, DT's are if-else statements

How to decide that which node to be a root node? (Next)



Advantages

Intuitive & easy to understand
minimal data prepration
log time complexity

Disadvantages

overfitting
prone to error for imbalance datasets

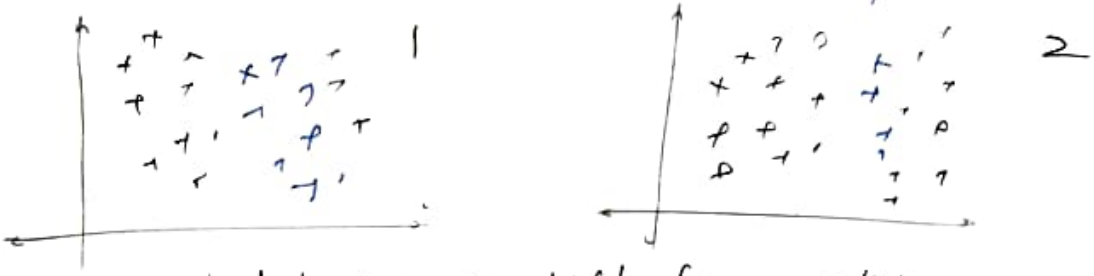
CART - Classification And Regression Tree

Entropy

measure of disorder / purity - impurity

Ice	water	vapour
less entropy		most entropy
more order		less order

more knowledge = less entropy



model 1 → more diff for confidence
dataset → high entropy

The mathematical formula for entropy is

$$E(S) = \sum_{i=1}^{c} - P_i \log_2 P_i$$

P_i - frequentist probability of an element/class in our data

eg. If our data has only two classes yes and No.

$$E(D) = - P_{yes} \log_2 (P_{yes}) - P_{No} \log_2 (P_{No})$$

Example : Dataset

salary	age	purchase
20000	21	yes
10000	45	no
60000	27	yes
15000	31	no
12000	28	no

$$H(d) = -P_y \log_2(P_y) - P_n \log_2(P_n)$$
$$= -(2/5) \log_2(2/5) - (3/5) \log_2(3/5)$$
$$= 0.97$$

salary	age	purchase
34000	31	no
15000	25	no
69000	57	yes
25000	21	no
32000	28	no

$$P(d) = (1/5) \log_2(-1/5) + (-4/5) \log_2(-4/5)$$
$$= 0.72$$

salary	age	purchase
[illegible]	<45	no
[illegible]	45-	no
[illegible]	34	no

$$f(0) = (0/5)\log_2(0/5) - (5/5)\log_2(5/5)$$
$$= 0$$

Calculate entropy for 3 classes

Salary	age	physique
20000	21	yes
10000	45	NO
60000	27	yes
15000	31	NO
30000	30	maybe
12000	18	NO
40000	40	maybe
20000	20	maybe

$$H(0) = P_y \log_2 P_y + P_N \log_2 P_N + P_{mb} \log_2 P_{mb}$$
$$= -\left(\frac{2}{8}\right)\log_2\left(\frac{2}{8}\right) - \left(\frac{3}{8}\right)\log_2\left(\frac{3}{8}\right) - \left(\frac{3}{8}\right)\log_2\left(\frac{3}{8}\right)$$
$$= 1.56$$

Observation

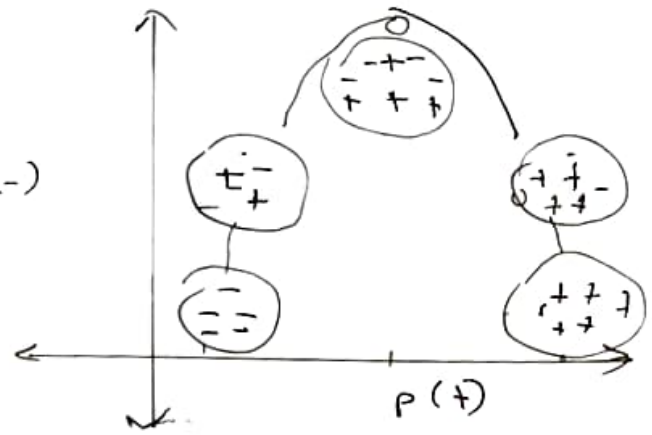
More uncertainty - more entropy

for 2 class problem, min entropy = 0
max entropy = 1.

for > 2 classes, min entropy = 0
max entropy > 1.

Can use $\log_2 a$ or $\log_e a$

Entropy v/s probabuty

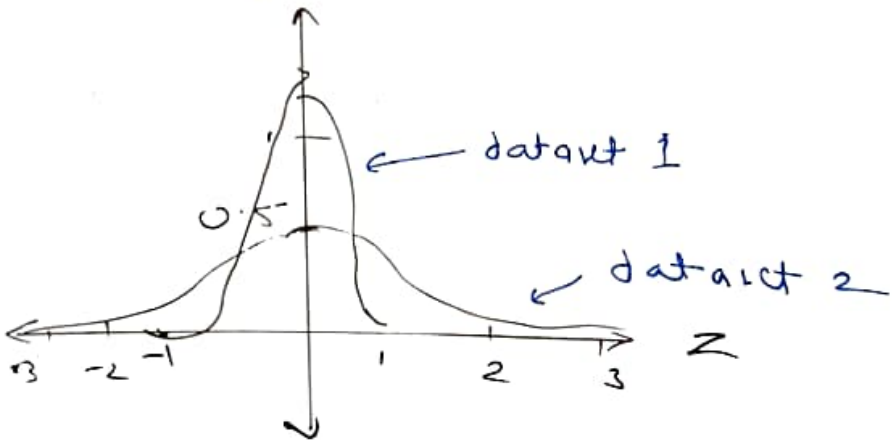


entropy for continuous variables

Area	Built-in	Price
1200	1999	3-5
1800	2011	5.6
1400	2000	7.3

Area	Builtin	Price
2200	1989	4.6
600	2018	6.5
1100	2005	12.8

find most entroped dataset.



Due to less range (-1,1), we know more about dataset 1. Hence less entropy.

entropy dataset 1 < entropy dataset 2

Information gain

Measures the quality of a split.

It's based on ~~entropy~~ the decrease in entropy after a data set is split or a column. Constructing a decision tree is all about finding attribute that returns the highest information gain.

Information gain = E(parent) − (weighted average) * E(children)

example: see tennis dataset again.

Outlook	Temperature	Humidity	Windy	Play tennis
Sunny	Hot	High	False	No
Sunny	H	H	True	N
Overcast	H	H	F	Yes
Rainy	Mid	H	F	Y
Rainy	Cool	Normal	F	Y
Rainy	C	N	T	N
Overcast	C	N	T	Y
Sunny	M	H	F	N
Sunny	C	N	F	Y
Rainy	M	N	F	Y
Sunny	M	N	T	Y
Overcast	M	H	T	Y
Overcast	H	N	F	Y
Rainy	M	H	T	N

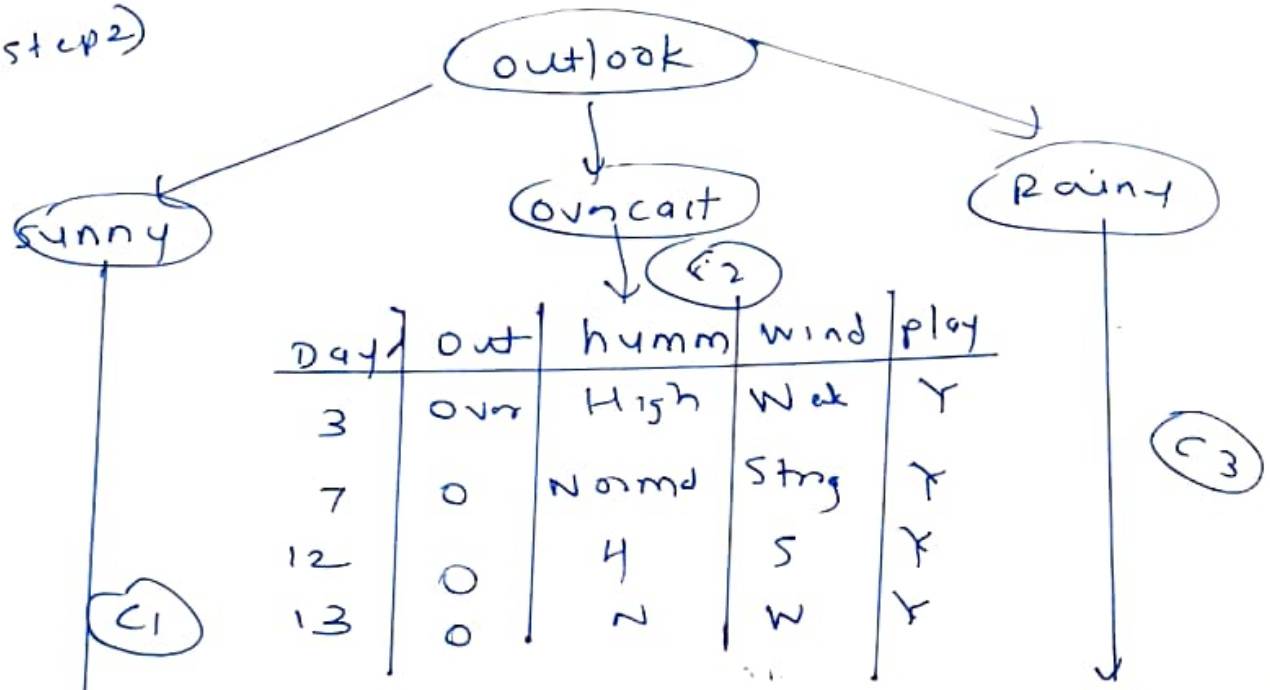
step 1) root node entropy

$E(P) = -P_y \log_2(P_y) - P_n \log_2(P_n)$

$= -\left(\frac{9}{14}\right) \log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right) \log_2\left(\frac{5}{14}\right)$

$= 0.94$

step 2)



Day	Out	humm	Wind	play
3	Ovr	High	Weak	Y
7	O	Normd	Strng	Y
12	O	H	S	Y
13	O	N	W	Y

Day	outlook	Humm	Wi	play
1	Sunny	High	Weak	No
2	S	H	Strng	No
8	S	H	W	N
9	S	Normd	W	Ye
11	S	N	S	Y

Day	Outlook	humm	Wind	Pl
4	Rainy	High	Weak	Y
5	R	Normd	W	Y
6	R	N	Strng	N
10	R	N	W	Y
14	R	H	S	N

for C1 → $(-2/5)\log(2/5) + (3/5)\log(3/5) = 0.97$

for C2 = $(-5/5)\log(5/5) + (0/5)\log(0/5) = 0$

for C3 = $(-3/5)\log(3/5) - (2/5)\log(2/5) = 0.97$

Step 3) - Calculate weighted entropy

we ent = $(5/14)(0.97) + (4/14)(0) + (5/14)(0.97)$

we (child) = 0.69

Step 4)

Info gain = E(parent) - (weight avg) $\sum_E$ (children)

= 0.97 - 0.69 = 0.28

So Information gain (~~Increase~~ / Decrease) in entropy / Impurity on the basis of outlook column is 0.28.

Step 5) Calculate Information Gain for all the columns.

Whichever columns has the highest info gain (maximum decrease in entropy) the algo will select that column to split data

Step 6)

DT then apply a greedy search algorithm in top-bottom fashion to find the inform gain at every level of the tree.

Once a Leaf node get reached. (Entropy = 0) no more splitting is done.

Gini impurity - For Sakit-learn formula:

$$GI = 1 - (P_Y^2 + P_N^2)$$

Go on next page for example for gini - impurity.

D1

salary	age	phone
20K	21	Y
10K	45	N
60K	27	Y
15K	31	N
12K	18	N

D2

salary	age	phone
34K	31	N
15K	25	N
69K	57	Y
25K	21	N
32K	28	N

$$GN = 1 - (P_{Y_1}^2 + P_{N_1}^2)$$

$$= 1 - \left(\left(\frac{2}{5}\right)^2 + \left(\frac{3}{5}\right)^2\right)$$

$$= 1 - \left(\frac{4}{25} + \frac{9}{25}\right)$$

$$= 1 - \frac{13}{25} = \frac{12}{25} = 0.48$$

$$Gi2 = 1 - \left(P_{Y_2}^2 + P_{N_2}^2\right)$$

$$= 1 - \left(\left(\frac{1}{5}\right)^2 + \left(\frac{4}{5}\right)^2\right)$$

$$= 1 - \left(\frac{1}{25} + \frac{16}{25}\right)$$

$$= 1 - \frac{17}{25}$$

$$= \frac{8}{25}$$

$$= 0.32$$

$Gi2 < Gi1$

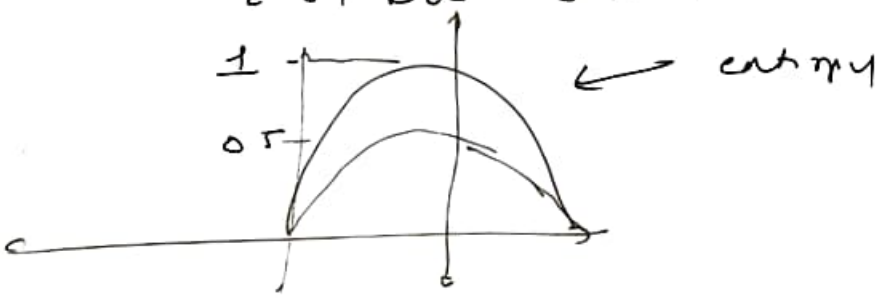
Difference between Gini and entropy

case 1 :) $P_Y = 0$ & $P_N = 1$.

$E = 1$ and $G_i = 1$.

case 2 :) $P_Y = 0.5$ & $P_N = 0.5$

$E = 1$ but $G = 0.5$



Why gini ?

fast on board computing

But in some cases, Entropy can give better splits.

Handle numerical

example

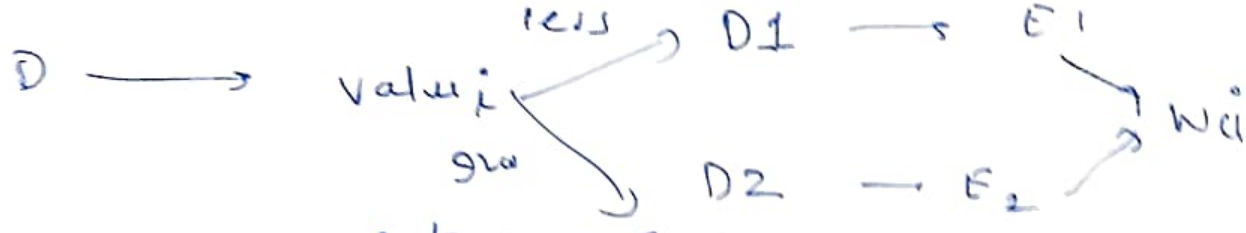
No	User rating	Download
1	3.5	Y
2	4.6	Y
3	2.2	N
4	1.6	Y
5	4.1	N
6	3.9	N
7	3.2	Y
8	2.9	Y
9	4.8	N
10	3.3	Y
11	2.5	Y
12	10	
13		

step 1)
sort data on basis of numerical column

NO	rating	Download
1	1.5	Y
2	1.9	N
3	2.2	Y
4	2.5	N
5	2.9	N
6	3.2	Y
7	3.3	N
9	3.5	Y
10	3.9	N
11	4.1	N
12	4.6	Y
13	4.8	Y

step 3) Split in parts. for every row value.
data < value < data
1 2

step 4) for every row element in column



for all values in column.
wi = weighted entropy
D → Dataset

Now we will calculate

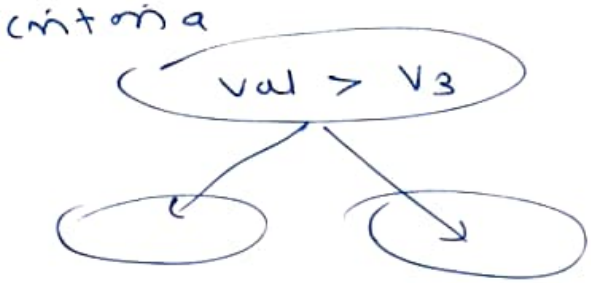
E(parent) − E(child) = Information Gain

IGi

find Max(IG1, IG2 ... IGn)

let us I assume it is IG3.

Now, use it as splitting criteria



step4:) Do it recursively until to get all leaf nodes.

This thing looks computationally toc expensive. is it right way of splitting criteria?

Yes.

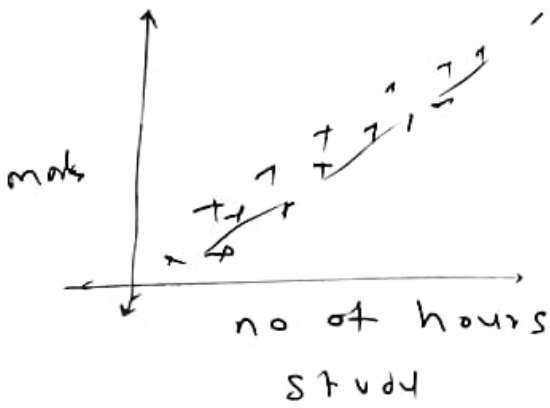
It's worth it.:)

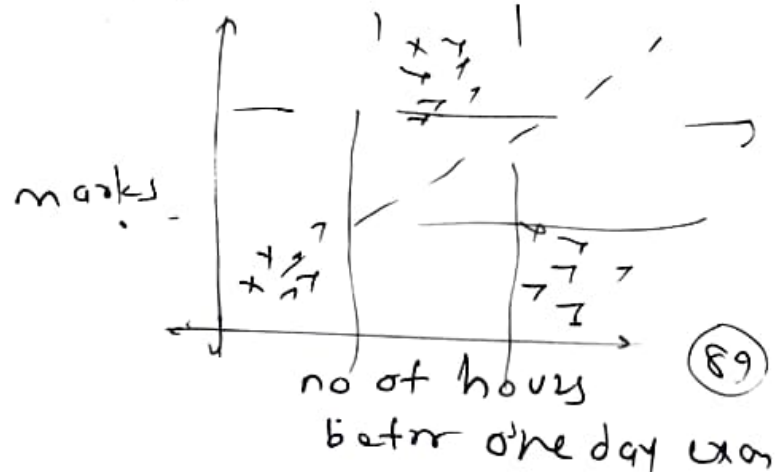
Because compute is in only training part, & not in testing part much.

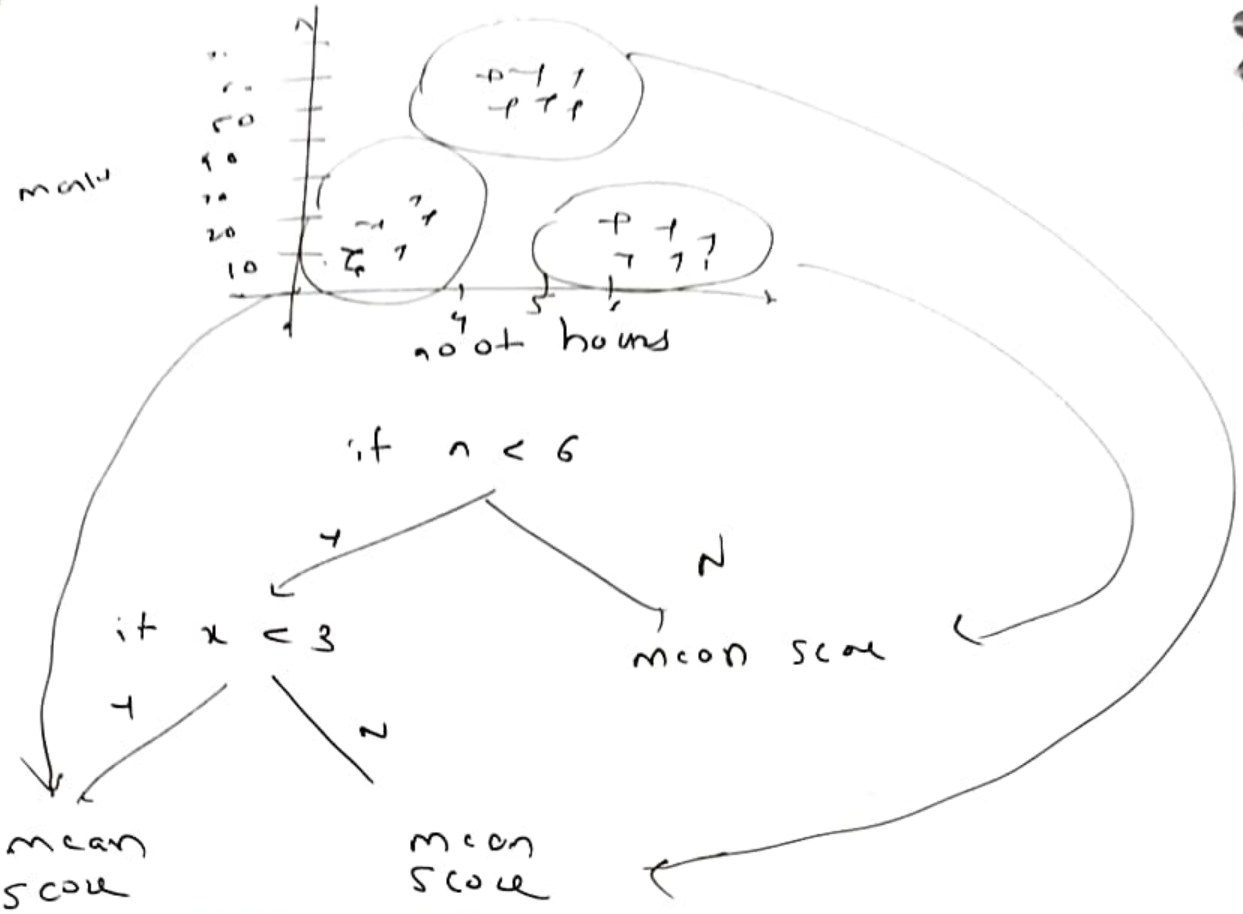
for too much numerical colum, Avoid decision trees.

Regression trees

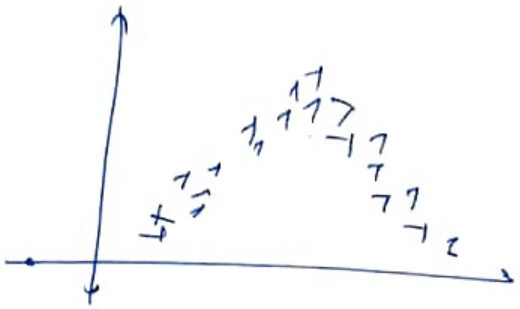
In non linear situation, regression tree gives us better results.





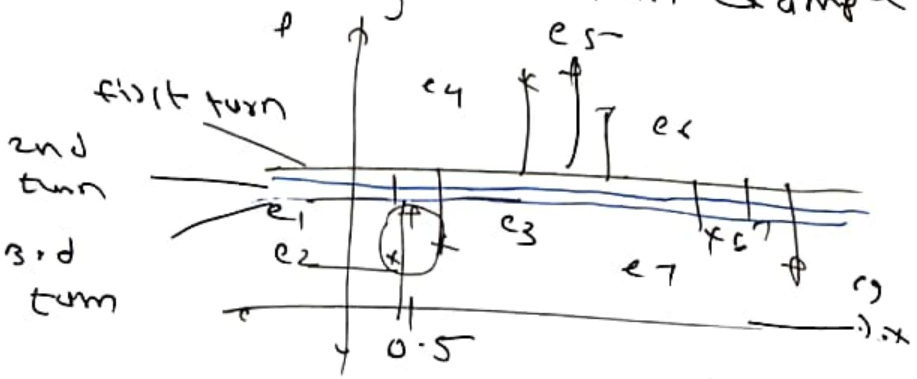


But what if the curve is like this



Spliting criteria

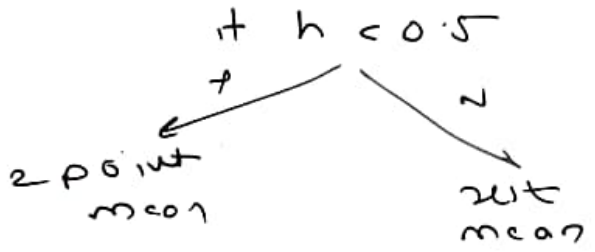
let us go with an example



consider first two points, find mean (m1)

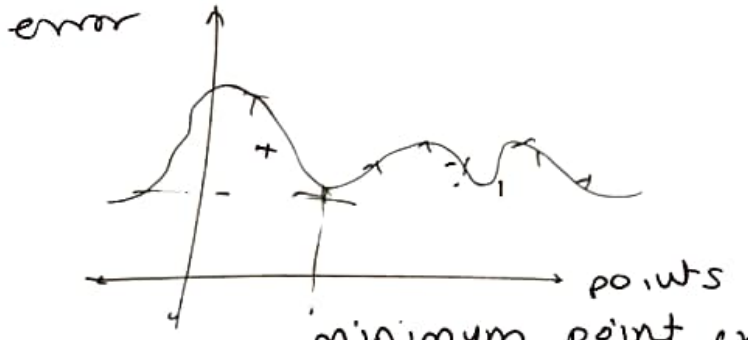
consider rest of points find mean (m2)

calculate distances.



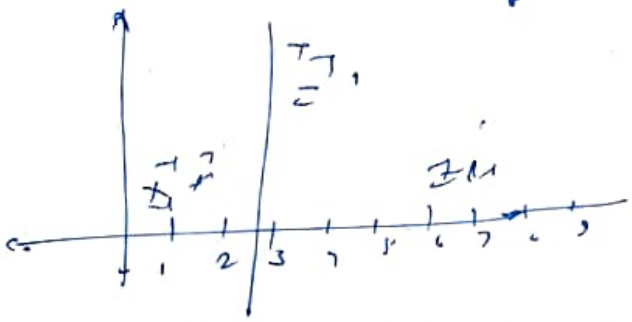
$SSE = e_1^2 + e_2^2 - - e_n^2$

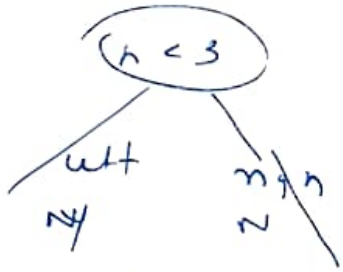
Now consider next 2 points. (w.r.t. both "left" & "right" means & so on. and plot them.



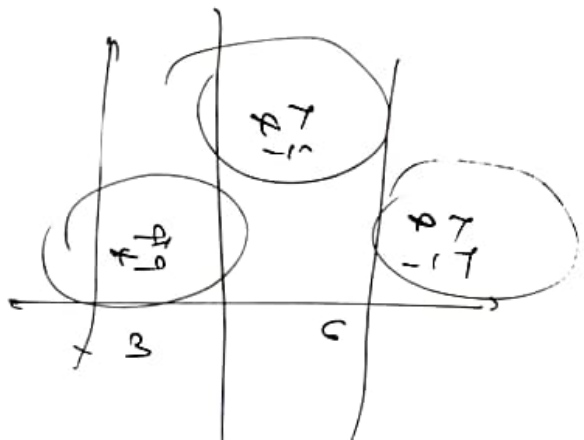
minimum point error — for the first split.

Now, first one to splitted and repeat process.





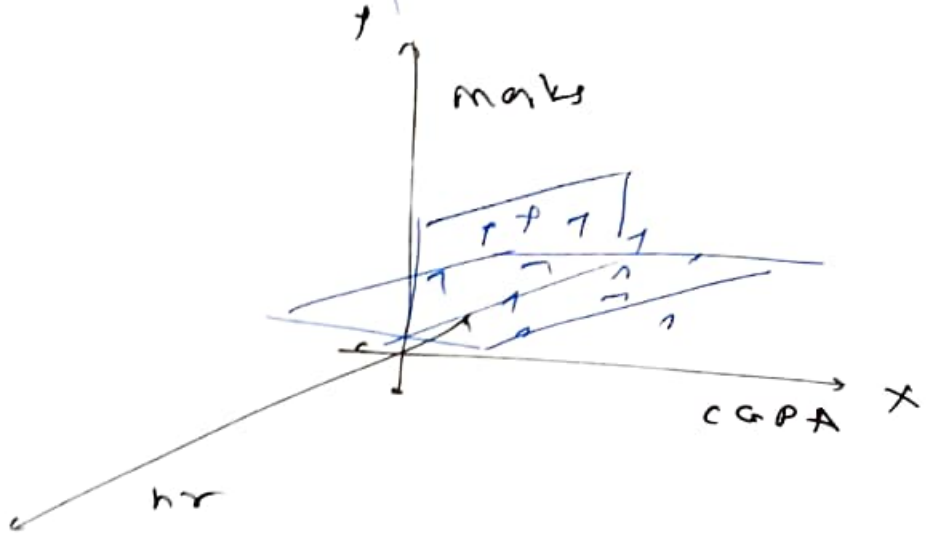
But if we split on and on, then, we seriously split on every point (overfit). So, we will set a threshold like if only points < threshold, we will not split. or continue to next points,



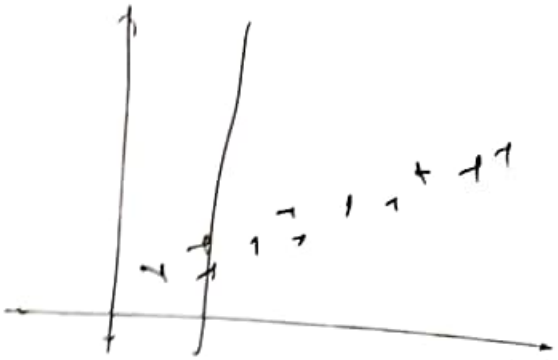
still this is in 2D what if higher dimen

sample data

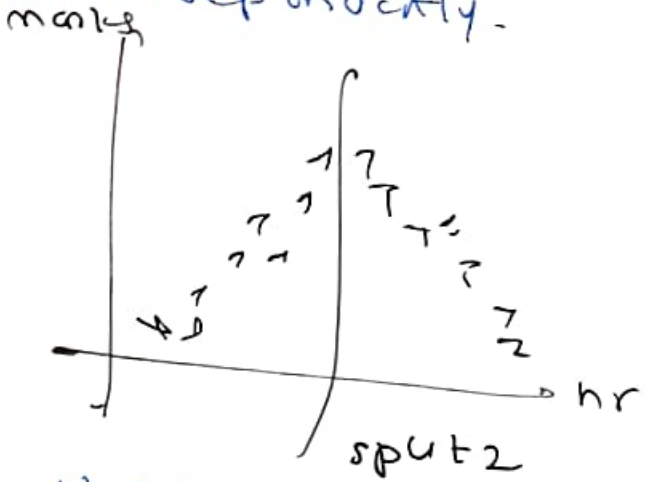
marks	cgpa	studyhr



marks



So, we will do independently.



Now, compare the errors

if 5E hr < 5E cgpa

whatever small, cut parallel to its direction., and follow on threshold.

Ensemble learning

A collection of multiple models

Wisdom of the crowd

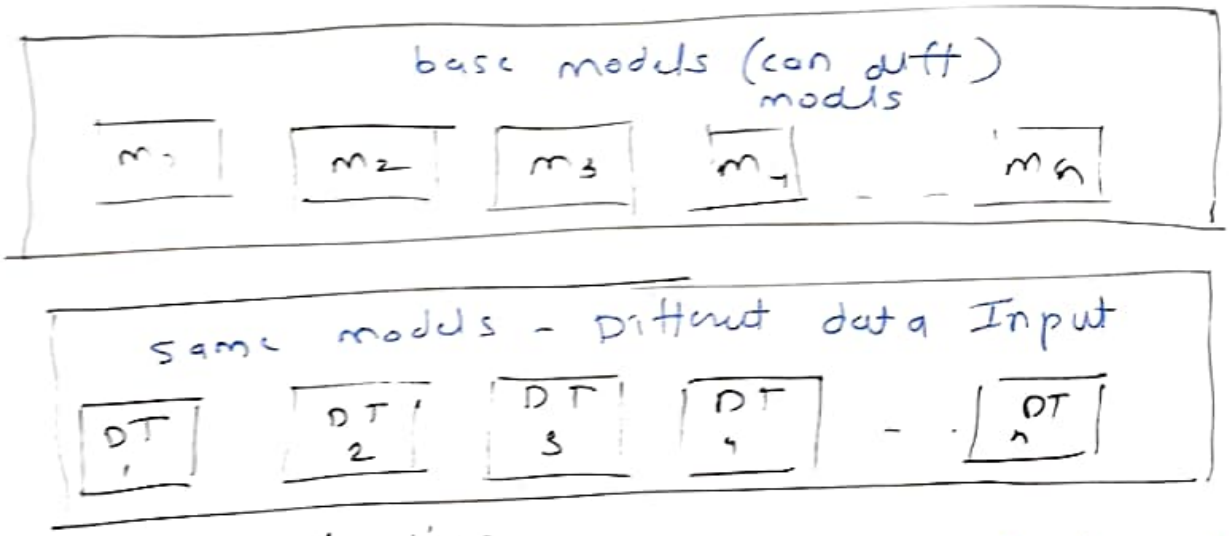
Audience poll in KBC.

No of reviews rules on amazon.

↑ trust of people on a democratic party.

Core idea

prediction



for classification

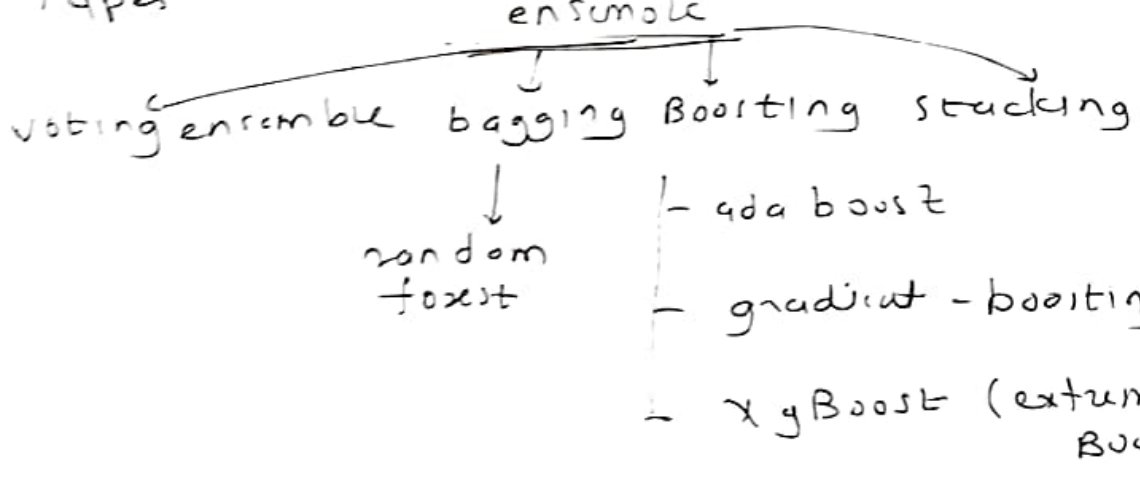
Data will be given to models. Each model will predict, and majority will be considered as a class.

for regression

Data will be given to models. Each model will predict, and we will calculate mean.

simple but effective

Types of ensimble learning



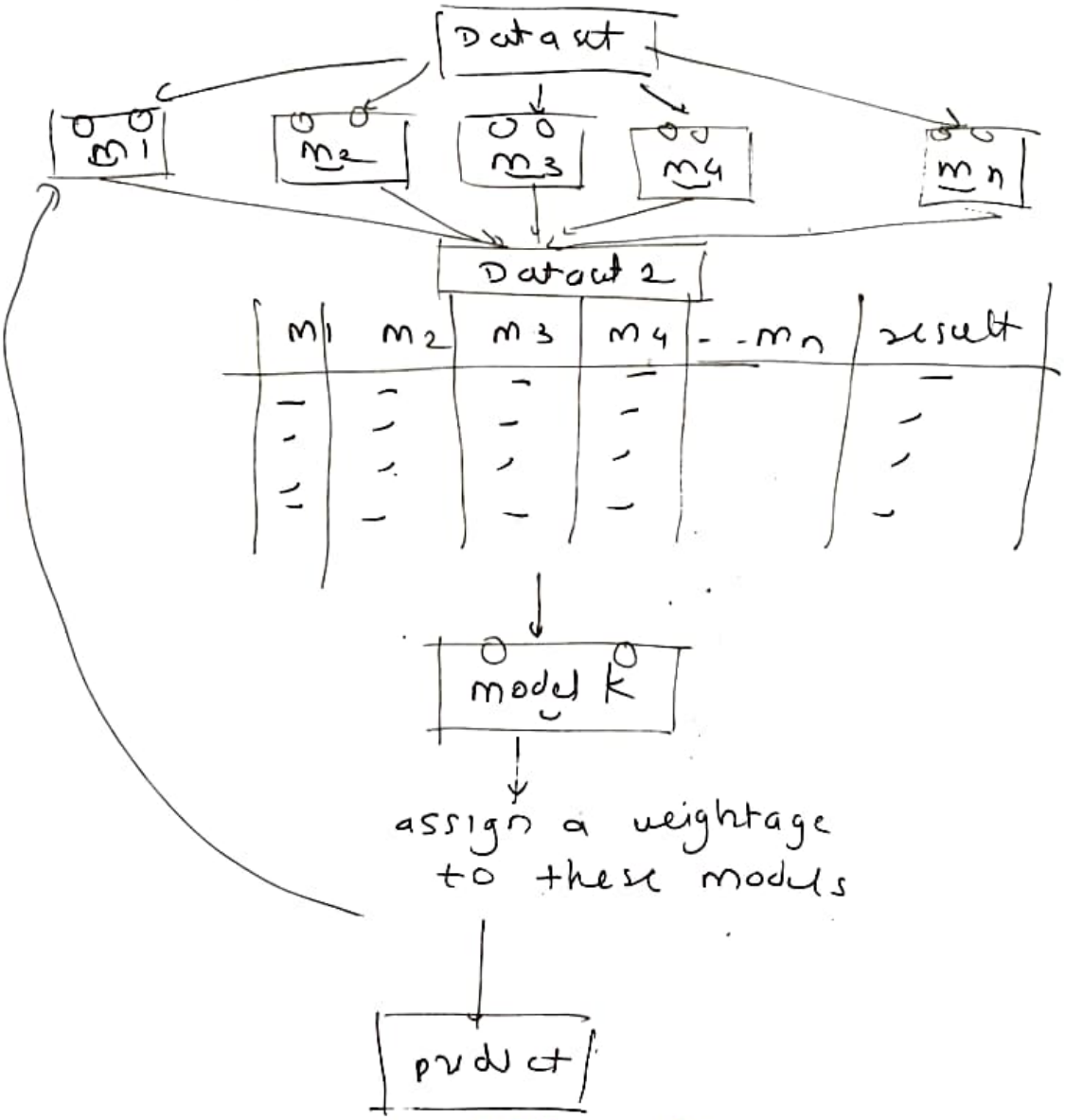
Why so many types?

We have two majors, train diff models, or train diff data. That's why difference.

Light Intio.

Voting ensimble - Base models difficult algorithms
Give inputs.
if classification,
Go with majority
if regression,
Go with mean.

Stacking -

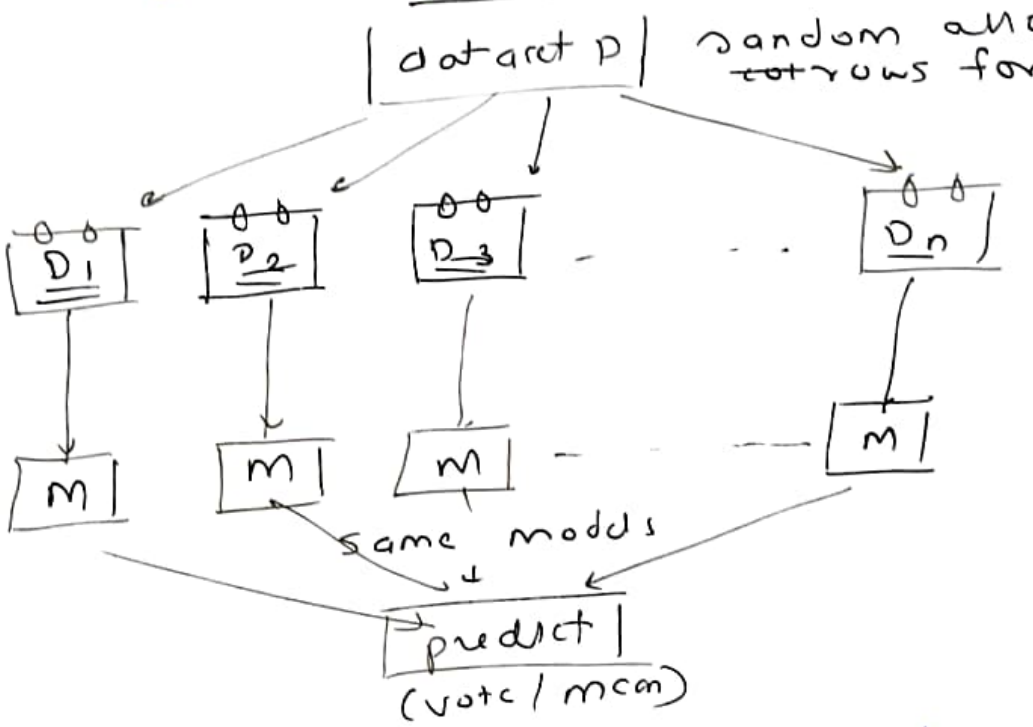


train dice diff models,
make a dataset, their model's output & actual results.
train a master model, & decide weights.
and predict.

Bagging

Bootstrapped aggregation

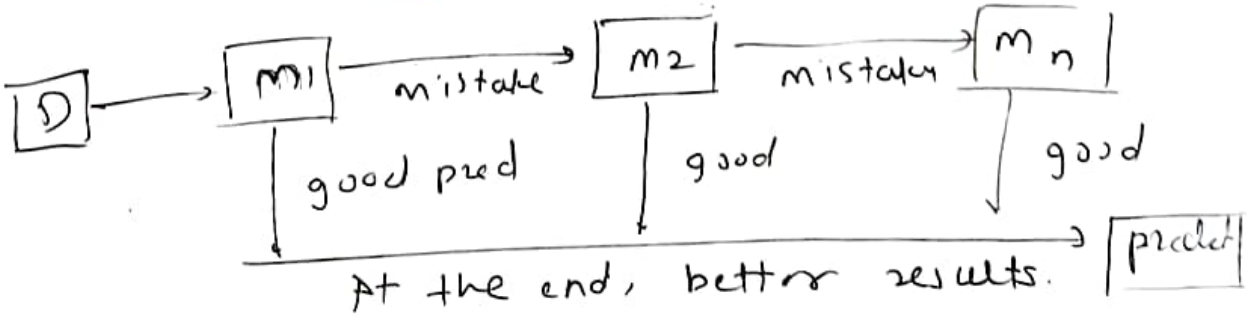
Same algorithm, Diff data Input



random forest is a subset of ensemble learning - (bagging) + (diff trees)

Boosting

Boosting results in a series!



But still why it works?

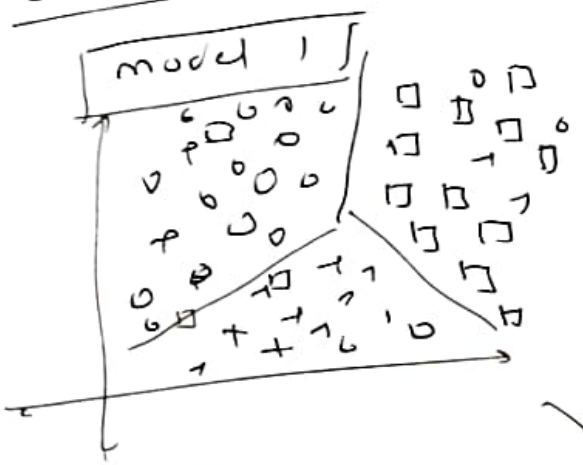
Let us see in the figure,

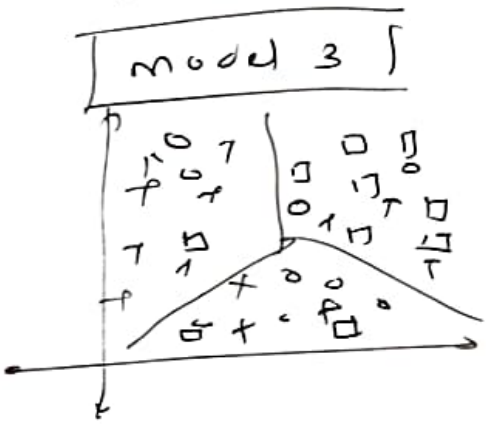
there are two different reasons.

1) Classification

2) regression

Classificatin



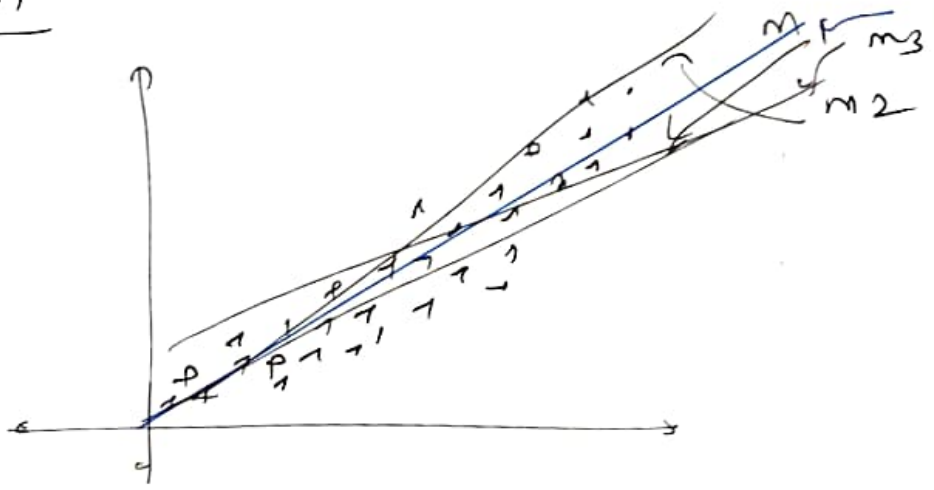


resultant model

final one model

better results

Regression



main model finally

High computational

Advantage

1) Improvement in performance
2) Reduce bias and variance
   It helps into achieve low (bias + var)
3) Robustness

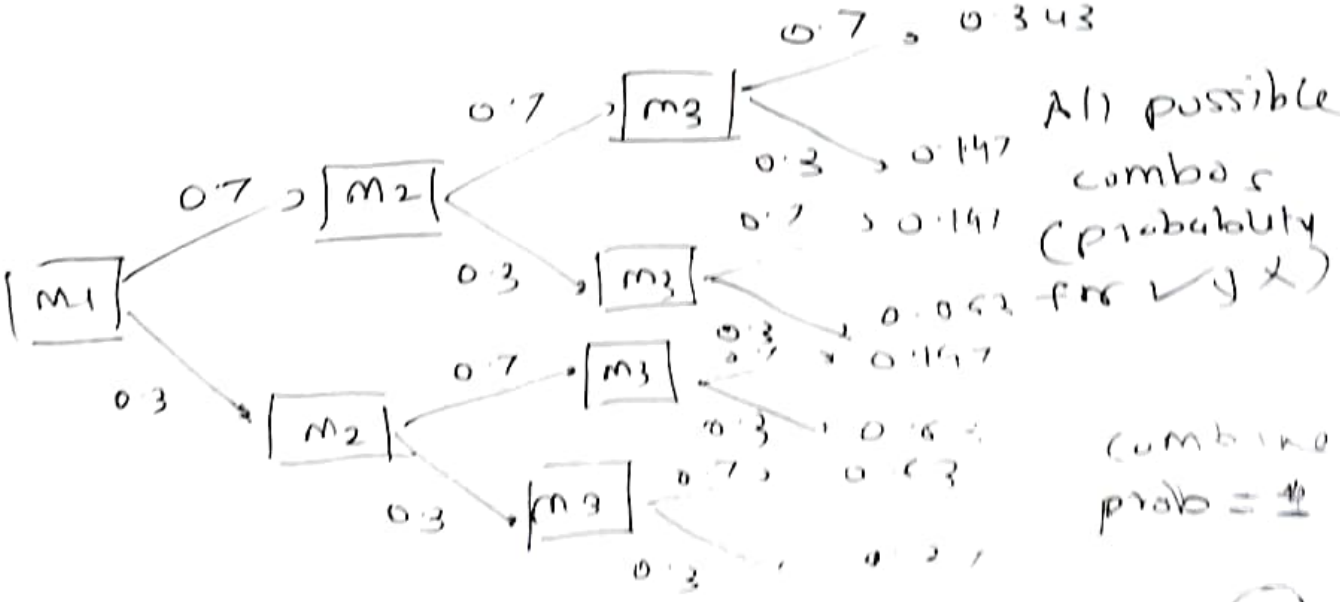
When to use ensemble learning?

Always.
non reason to dont use it!

1. Voting ensemble

please read the techniques, from intro, how to work.
But why it work?
How it is possible that it outperform from the combo of all algorithm?
Assumption
models should be independent.
for all models, min accuracy should be 51% at least, else it will be worst one ever.

example.
we have 3 models, each accuracy 0.7,



(91)

So, In the figures the out of 3, the right prob at least 2 is, there are 4 chances where accuracy (votes) are atleast 2 times same.

If we add, we get entire models right probability of prediction.

$34.3 + 14.7 + 14.7 + 14.7 = 0.78$

So, by this way, We proove that a combination work.

But we have a rule, that, accuracy at least for an individual model should be $\geq 51\%$ at least.

let us proove this, by assuming it is 51

Now, instad of 0.7, assume 03 place

Now, by combining these 03 models, we will get overallness accuracy.

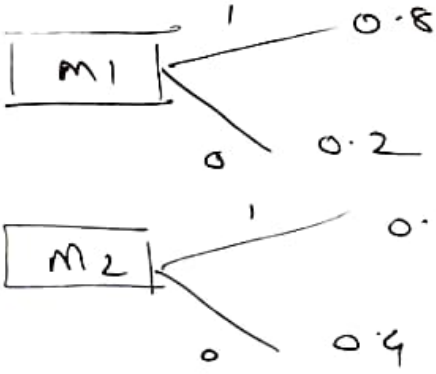
take cases whe 0.3 is at least 2 times.

So, 100 - 78%. (from the right overall).

$= 0.22$

prooved

hard voting / soft voting



hard voting

majority = 1 so 1.

soft voting

$1 = \frac{0.8 + 0.6}{2} = 0.7$ so 1.

$0 = \frac{0.2 + 0.4}{2} = 0.4$

generally soft voting better.

Regression

see part of Intro for the regression.

Bagging

bootstrapping + aggregation

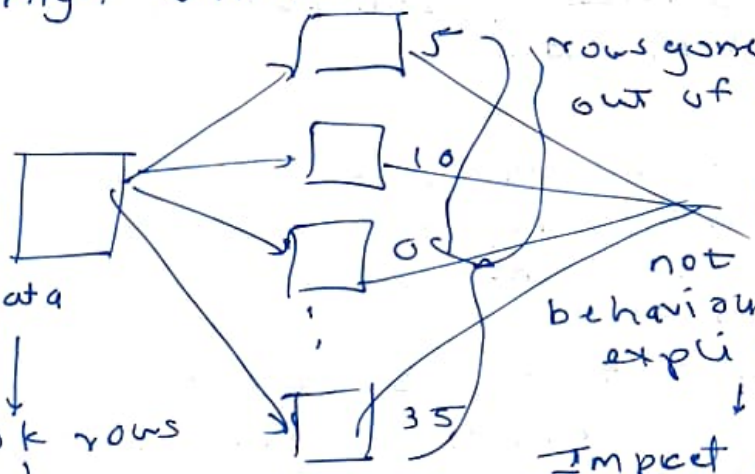
see Intro part for Intution.

But why bagging Improves score?

we can get { LB, HW / HB, LN } combo.

How?

we take models as a base for low bias and high variance



rows gone out of 100.

But it will not change behaviour of models explicatly.

Impact distributed

We replace 100 rows, which are different from last data.

NO high variance affected.

When to use?

Always give a try for it.

So, we have random forest.

Types of bagging

can do row + columns

pasting - without replacement row sampling

Random Subspace - column sampling

In bagging with replacement

random patches - rows + column sampling

OOB score

out of bag score

since we select at random, some rows go again and again, & some dont go. so they are called out of bag.

statistically it is proven that only 63% of rows go in. rest 37% never ever go.

Bagging generally perform better than pasting

good results come around 25% to 50%.

random patches & subspaces could use on high dim data.

Random forest

can solve any ML problem

good fit in any project

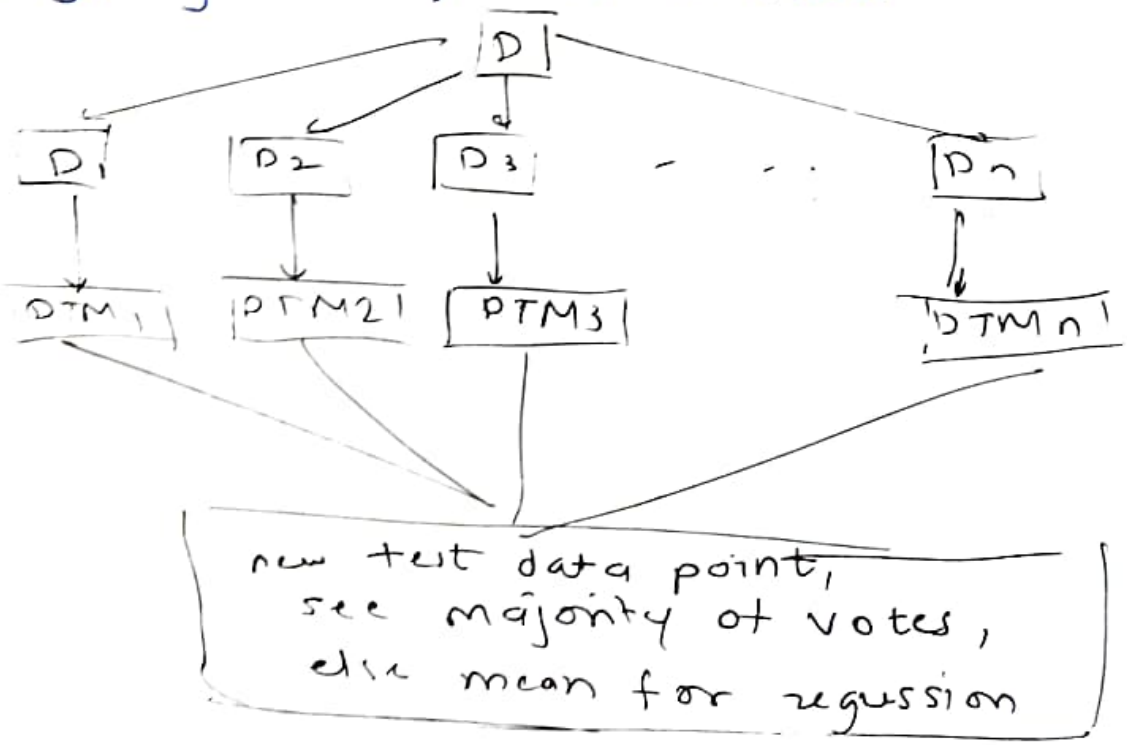
best if we dont change settings.

collection of trees & a bagging technique.

A group of trees. so forest.

random mean for random sampling via bagging.

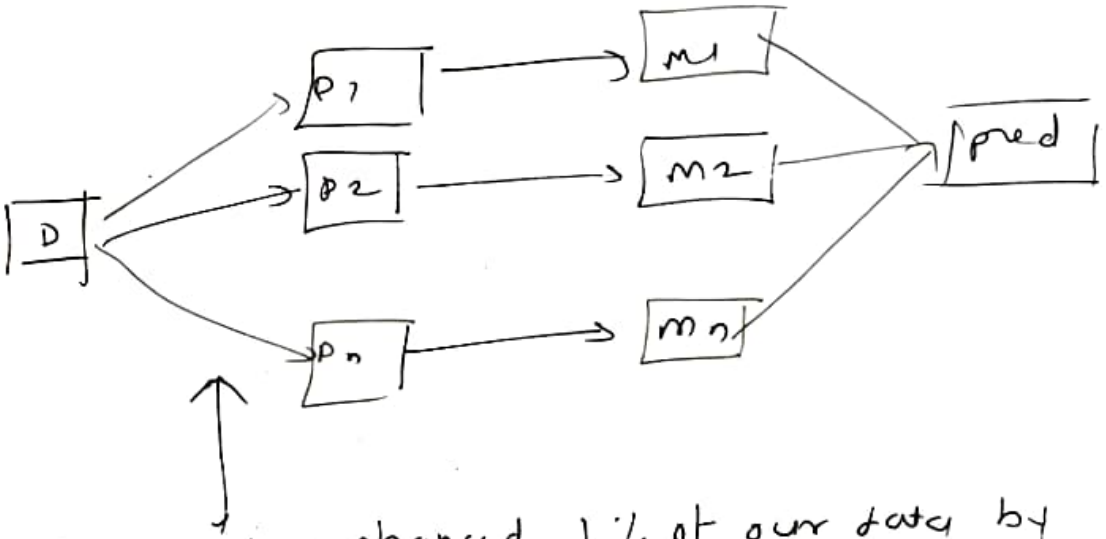
bagging technique with trees.



random forest helps us in like we have some algorithms,

Low bias high var	High bias low var
fully grown dec tree	LR
SVM	
KNN	

random forest helps us to maintain, like it convert Low Bias High var to low bias low var algorithm.



let say, we changed 1% of our data by outliers & noisy points.
It is not possible to get all noisy points to a single DT. Impact distributed.
so, NO much variance

Hyperparameters

number of estimators ~~too~~.
Trees to be made

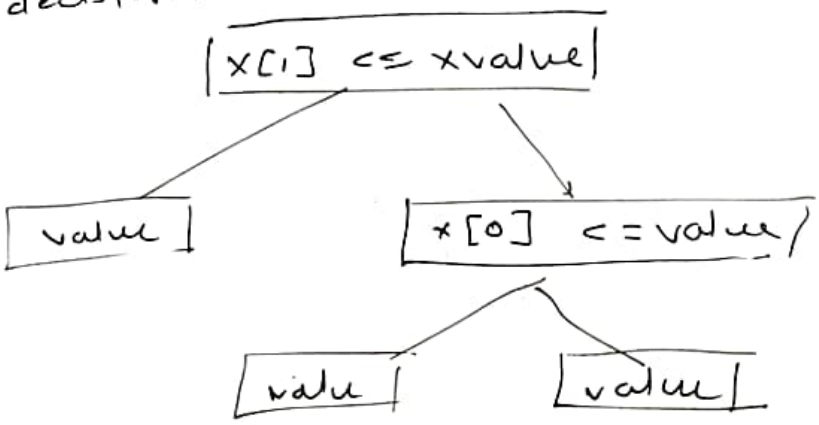
max - features
number of columns considered

Bootstrap
will you select rows / cols random

max samples
rows.

Feature importance

In decision tree



So, Here, we need to calculate feature importance.

So, for a column, we need to found how many nodes where made due to that column.

So,

$$\text{Feature imp} = \frac{\text{No of nodes made due to that column (Importance)}}{\text{Total number of Importance (all nodes)}}$$

How to be node importance

$$ni = \frac{N\_t}{N}\left[\text{impurity} - \left(\frac{N\_t\_r}{N\_t} \times \text{right\_node\_imp} - \frac{N\_t\_L}{N\_t} \times \text{left\_node\_imp}\right)\right]$$

N_t - no of rows to that node

N = total no of rows

impurity,

n_t_r = no of rows in right node

right impurity = right node impourity

n_t_L = no of rows in left node

left impurity = left node impurity

So, Decision tree will calculate importance of every feature.

In random forest

for n no of models, we have n number of feature importances for every feature. & random forest then will average it.

AdaBoost

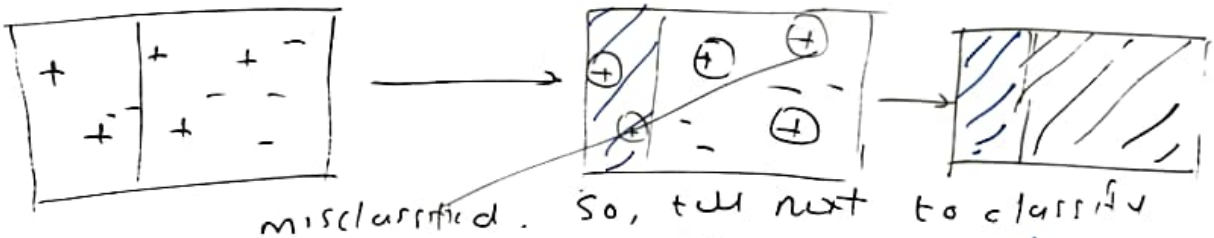
Weak learners - A model with less accuracy, around 50%.

Decision stumps - A type of weak learners which max depth 1 as a decision tree.

+1 and -1 class in binary classification instad of 1 and 0.

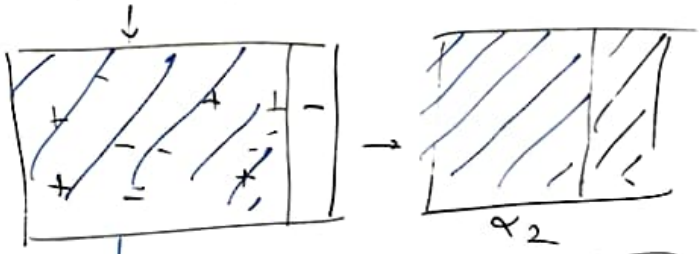
A split on any axis with minimum entropy.

Adaboost
- stage wise } add all one by one
- additive method } add multiple weak learners

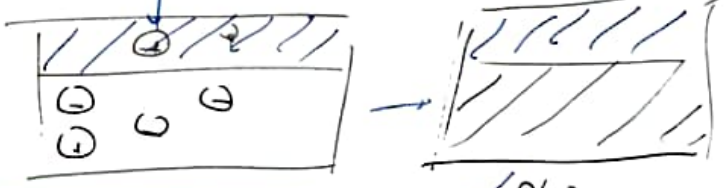


misclassified. So, tell next to classify them more property

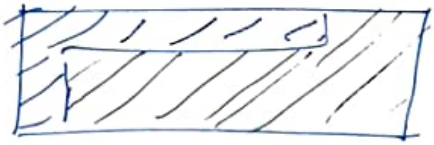
α_1



α_2



α_3



So, eqn is

$$h(x) = \text{sign}(\alpha_1 * h_1(x) + \alpha_2 * h_2(x) + \alpha_3 * h_3(x) +)$$

Hx = adaboost eqn

α_i = weights assign

H_N = Indiv eqn of models.

Core mathematics

A classification problem

$n = 5$

x_1	x_2	y	wt	y.pred
3	7	1	0.2	1
2	9	0	0.2	1 x
1	4	1	0.2	0 x
9	3	0	0.2	0
5	7	0	0.2	0

$\Rightarrow$ n starting we will assign equal weight of 1/5.

Step 1) Decide a 1-depth decision stump cut/split on any column, name it as model 1.

α_1 = for model 1 (weight)

α depend on model error rate

$$\text{error rate} \propto \frac{1}{\alpha}$$

How to calculate α?

A	B	C
0%	50%	100%

error rate

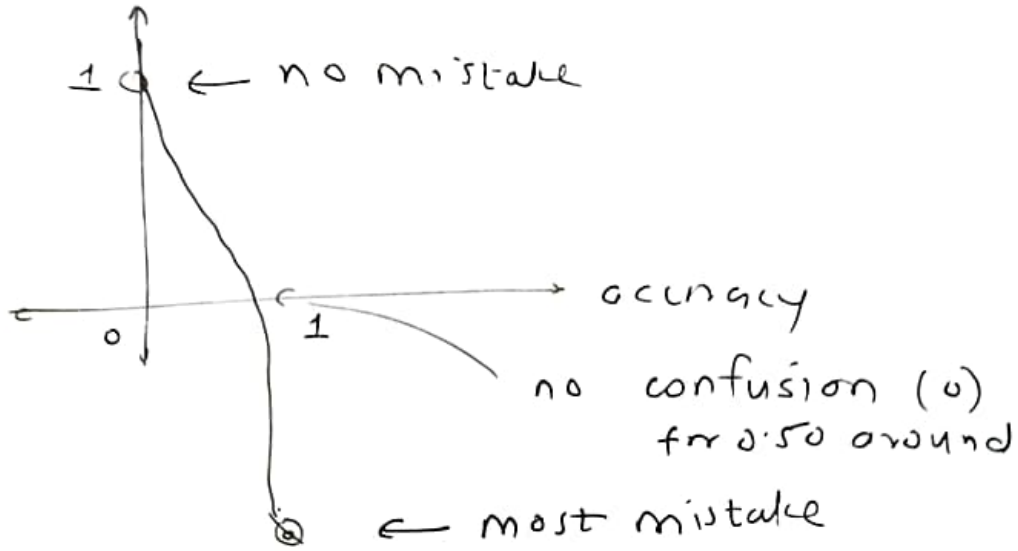
which is most reliable?

model A

except model A

model C,

because, If we invert results, we are accurate



This function exist, that is,

$$\alpha = \frac{1}{2} \log_e \left(\frac{1 - err}{err} \right)$$

So, when model M1 predicted

error was $= \frac{2}{5} = 0.4$

$$\alpha = \frac{1}{2} \log_e \left(\frac{1 - 0.4}{0.4} \right)$$

$$= \frac{1}{2} \log \left(\frac{0.6}{0.4} \right) = \frac{1}{2} \log \left(\frac{3}{2} \right)$$

$$= 0.20$$

Now, first prediction successful

Now we need to point to next model that to need to classify that wrong points properly.

But how algo will know about that misclassified points?

we will increase weightage of misclass points and decrease weightage of classif.

But in which quantity to increase & decrease?

for misclassified points

new_wt = current_wt $\times e^{\alpha i}$

for classified points

new_wt = current_wt $* e^{-\alpha i}$

og-wt	new-wt		
	1	0.166	0 - 0.166
	0.16	0.25	0.166 - 0.410
0.2	0.24	0.25	0.410 - 0.660
0.2	0.24	0.166	0.660 - 0.826
0.2	0.16	0.166	0.826 - 0.992
0.2	0.16		
	0.96	≈1.00	

boosting misclassified points via upsampling

Now, generate 5 random number, and put them in rows range

0.13	0.46	0.82	0.50	0.8
1	3	3	3	4

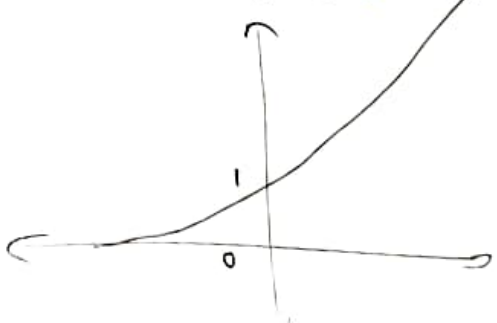
what happens here, that, the probabuty of "selating" that row, on random number generation, Increase on having higher range, mean upsampling is occred

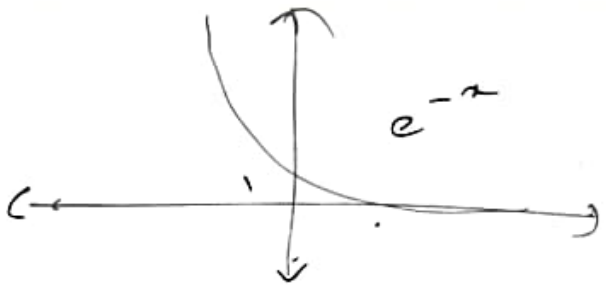
now, I will agan train model, and do stumps, again α_2, & upsampling. do until we have n no of decision stumps.

so $F = h_1(x)\alpha_1 + h_2(x)\alpha_{2x} - \_\_ h_n(x)\alpha_n$

why e^x or e^{-x} term?

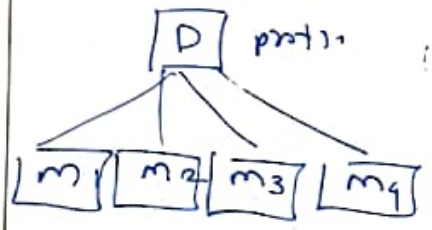
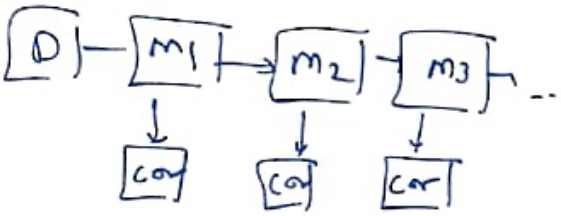
if a model has large α, mean less mistak
so, this value when model apply, it will a hig
value. A high trustable model get high valu





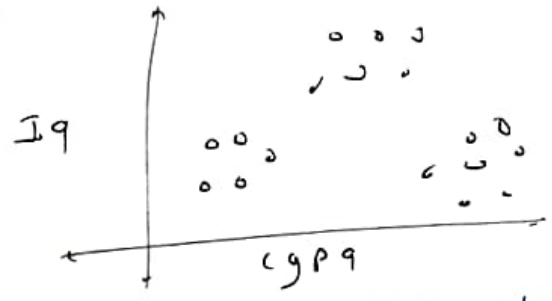
Reducing learning rate in adaboost decreased overfitting.

Bagging VS boosting

	Bagging	Boosting
Type of algorithm	Low Bias + high variance algorithm (fully grown DT)	High Bias + Low variance algorithm (Decision stumps 1-length DT)
learning	Parallel in bagging 	Sequencial in boosting 
weightage of base learner	same for all models	have some weightage to a model

K-mean clustering

I have a data of our studat, I wish to run them for exams, & I wish to classify them.



We wish to make cluster (groups).

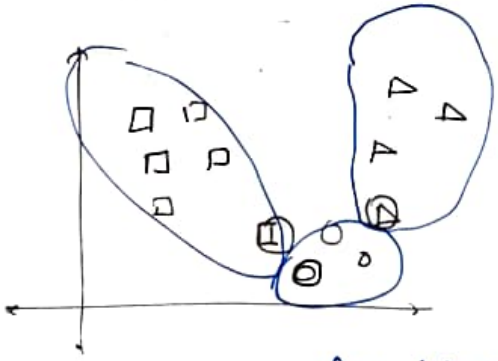
This is easy in 2D, But not in higher dimensions.

Steps working

1) Tell algo number of clusters. (k)

2) Initialize randomly, centroids =(k).

3) We will make clusters based on ~~that~~ points. So from the centroid, we will calculate every distance of every point, & will assign point to that centroid, if it is nearest one.



after round 1, we calculated the distance & classified points on least distance

Now we have these clusters.

Now for that clusters, Calculate means with respect to the points and axes, and assign a new centroid.

O → new centroid

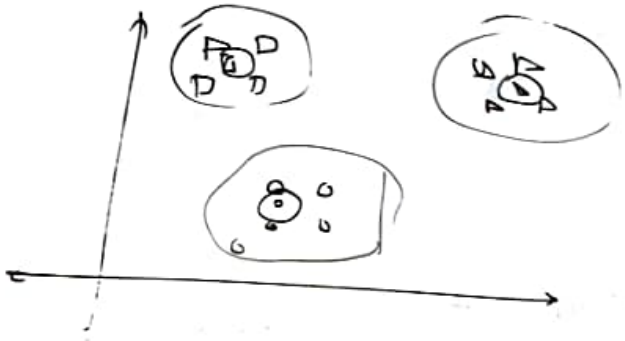
moved centroid

run algonthm until:

our centroids goes in the groups.

In last, we still don't have achieved, as centroid so far from points.

Now, assign clusters again according to centroids.



still our centroid moved & cluster change, we will again move our centroids by mean.

if centroid moved, we need to repeat again.

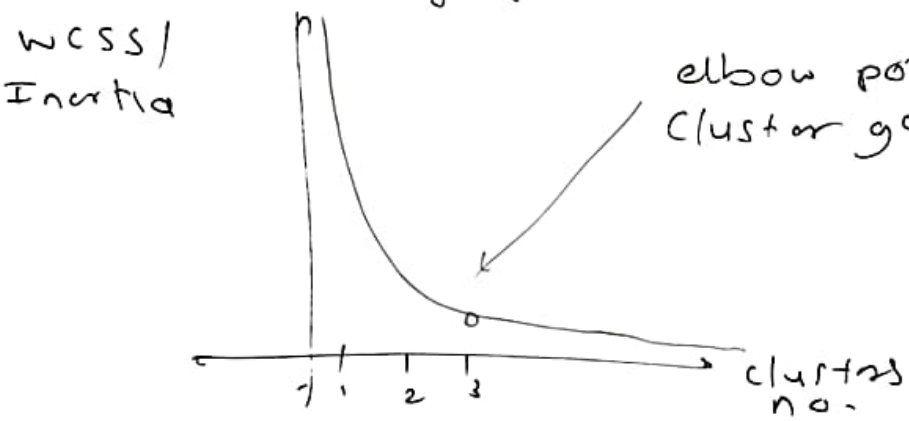
Assign clusters, Calculate centroids, and fix cathroid.

At the end, Our centroids will not move.

How I will know How many clusters to take?

by Elbow method.

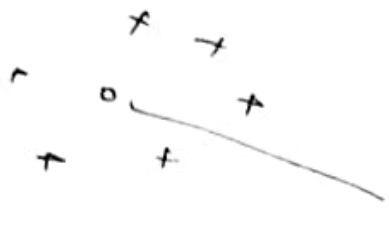
plot graph



WCSS - within cluster Sum of square dist

$WCSS_1 > WCSS_2 > WCSS_3$ - -

wcss for one cluster



Calculate, square, add distance bet^ them

for two clusters

Calculate individual clusters, add wcss

start with cluster 1, 2 ... & plot on the graph of elbow.

equlidian distance for calculating distance between centroid & points.

two points, $(x_1, y_1), (x_2, y_2)$

$$d = \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$$

but higher dim,

$$d = \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2 + (z_2-z_1)^2} \ldots$$

but we can't calculate this way

we say,

$$= (x_2-x_1)^2 + (y_2-y_1)^2$$

$$= (x_2-x_1)(x_2-x_1) + (y_2-y_1)(y_2-y_1)$$

for the vector,

$$= [(x_2-x_1)+(y_2-y_1)][(x_2-x_1)+(y_2-y_1)]$$

we say $(x_1, y_1) = a$

$(x_2, y_2) = b$

$$= (b-a) = [(x_2-x_1)(y_2-y_1)]$$

so this way, Entire thing is

```
np.sqrt(
    np.dot(a-b, a-b)
)
```

Gradient boosting

Again a boosting technique

iq	cgpa	salary	pred M1	res-1
90	8	3	4.8	-1.8
100	7	4	4.8	-0.8
140	6	8	4.8	3.2
120	9	6	4.8	1.2
80	5	3	4.8	-1.8

$|M1| = \frac{(3+4+5+6+3)}{5} = 4.8$

In regression, It's mean of output.

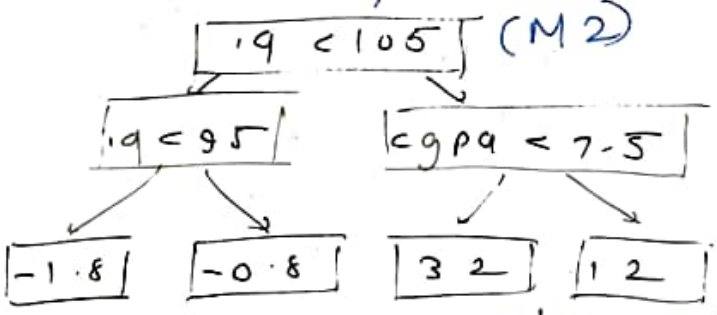
loss function = Actual - predicted (pseudo residual)

We want to tell M2, the mistakes of M1 anyway. M2 a decision tree model (assume)

Here come concept of gradient boosting for model 2,

Input - same OUTPUT - RES1.

We wish to at model 2 predict how many mistakes are done by model 1.



iq	cgpa	salary	res 1 (M1)	(pred. M2)
90	8	3	-1.8	-1.8
100	7	4	-0.8	-0.8
140	6	8	3.2	3.2
120	9	6	1.2	1.2
80	5	3	-0.8	-1.8

At this moment,
let assume we have 2 models.
Now, formula.

$$\text{pred\_ovr} = \text{out\_M}_1 + \text{out\_M}_2$$

for student 3,

$$= 4.8 + 1.2 = \text{salary}$$
$$(M_1) \quad (m_2) \quad (r)$$

But, this is simply an overfitting.
so, we use learning rate

$$\text{pred\_ovr} = \text{out\_M}_1 + (\text{learng\_rate})\ \text{out}\ M_2$$

we only take it to reduce overfitting,
and it is usually 0.1.
So, on prediction of
same student

$$4.8 + (0.2)(0.1) = 4.8 - 0.12$$
$$= 4.92$$

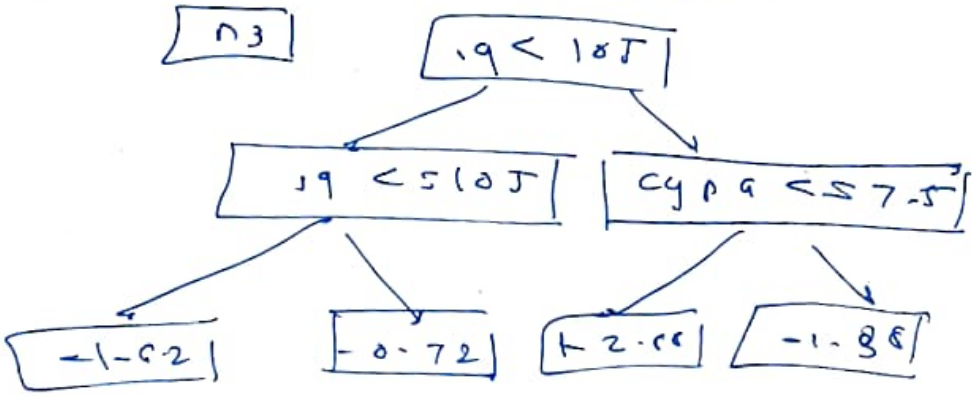
obsolutely wrong,
but if we add more models, It will add and add, and go closer to actual TO reduce overfitting.

So, res2 = actual − $(m_1 + (0.1)\, m_2)$

$$3 - (4.8 + 0.1 \times (-1.8) = -1.52$$

So, Now, Calculate res2,
you notes AS you go to add models further, you comes closer to your ideal residual (0).

Now, let us train M3, as a DT.
Input same OUTPUT res2.



so, finally,

$$m\_prd = m_1 + (0.1)\, m_2 + (0.1)\, m_3$$

ig	cgpa	salary	pred1	res1	pred2	res2	pred3
90	8	3	4.8	-1.8	-1.8	-1.62	
100	7	4	4.8	-0.8	0.8	0.72	
110	6	8	4.8	3.2	3.2	2.88	
120	9	6	4.8	1.2	1.2	1.08	
80	5	3	4.8	-1.8	-1.8	-1.62	

In gradboost, we usually take max leaf nodes between 8-32 for best results.

▶ adaboost vs gradient-boost

	adaboost	grad-boost
max-leaf	1 (stump)	(8-32)
train compt	weight for different models.	same learning rate for all models.

mathematics behind gradient-boosting algo.

Every Boosting algorithm use additive modeling.

let discuss. what ML does exactly?

what our ML models?

$y = f(x)$ — A function

look a dataset

linear ? don't work

polynomial problem

Runge's phenomenon

Basically it disturb from line in higher degree

is this any direct fun^ ?

No

a combo ? yes $5 \sin x + x$

breaking down fun^ / phasing

rigonthm?
we need input
$\{\{x, y\}\}_1^n$
we need a loss function, differentiable.
$L(y, f(x))$
we take a math's loss fun'
$L(y, \hat{y}) = \frac{1}{2}(y - y_i)^2$
$=$ ⟵ to handle diff

no of iterations M
for i in range M → step 2.
$f(x) = f_0(x) + f_1(x) + f_2(x) \ldots f_n(x))$
step 1) find $f_0(x)$
$f_0(x) = \arg\min \sum_{i=1}^{n} L(y_i, \gamma)$ $L =$ $y_i \to \gamma$

$$f_0(x) = \arg\min_{\gamma} \frac{1}{2} \sum_{i=1}^{n} (y_i - \gamma)^2$$

we need γ so the value of this expression should be minimum.

$$\frac{d}{d\gamma} f_0(x) = \frac{d}{d\gamma} \frac{1}{2} \sum_{i=1}^{n} (y_i - \gamma)^2$$

$$= \frac{1}{2} \sum_{i=1}^{n} \frac{d}{d\gamma} (y_i - \gamma)^2$$

$$= \sum_{i=1}^{n} (y_i - \gamma) \frac{d}{d\gamma} (y_i - \gamma)$$

$$= -\sum_{i=1}^{n} (y_i - \gamma) = 0$$

$$\sum_{i=1}^{n} (\gamma - y_i) = 0$$

So it's make a dataset now

nd	RND	admin str	profit	marketing
0	165	137	192	472
11	101	92	144	250
3(	29	127	91	201

so, if

$$\sum_{i=1}^{3} (r - y_i) = 0$$

$$(r - 192) + (r - 144) + (r - 91) = 0$$

$$3r = 192 + 144 + 91$$

$$3r = 426$$

$$r = 142$$

kind of this is a mean!

step 2)

$$f(x) = \underbrace{f(x_0)}_{\text{mean}} + \underbrace{f(x_1) + f(x_2) \ldots f(x_n)}_{\text{calc in loop}}$$

for i = 1 to M:

A) $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial (x_i)} \right]_{f = f_{m-1}}$$

i = row number

m = decision tree number

for first, m = 1

$$r_{i1} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f = f_0}$$

cond 0

pseudo residual for first row, r_{11}

first DT r_{21}

~~first DT~~ first DT r_3, 3rd row

Loss $f_{0'}$

No need of summ, as already calculating for row.

$$r_{11} = -\left[\frac{\partial (L(y_i, \hat{y_i}))}{\partial \hat{y_i}} \right]_{f = f_0}$$

$$= -\left[\frac{\partial}{\partial \hat{y_i}} \left(\frac{1}{2} (y_i - \hat{y})^2 \right) \right]_{f = f_0}$$

$$= [(y_i - \hat{y_i})]_{f = f_0}$$

$$= [(y_i - f(x_i)]_{f = f_0}$$

$$r_{i1} = y_i - f_0(x_i) \quad \text{——— (1)}$$

At this case, so, In

$r_{11} = y_1 - f_0(x_1) = 192 - 142 = 49$

$r_{21} = y_2 - f_0(x_2) = 144 - 142 = 1$

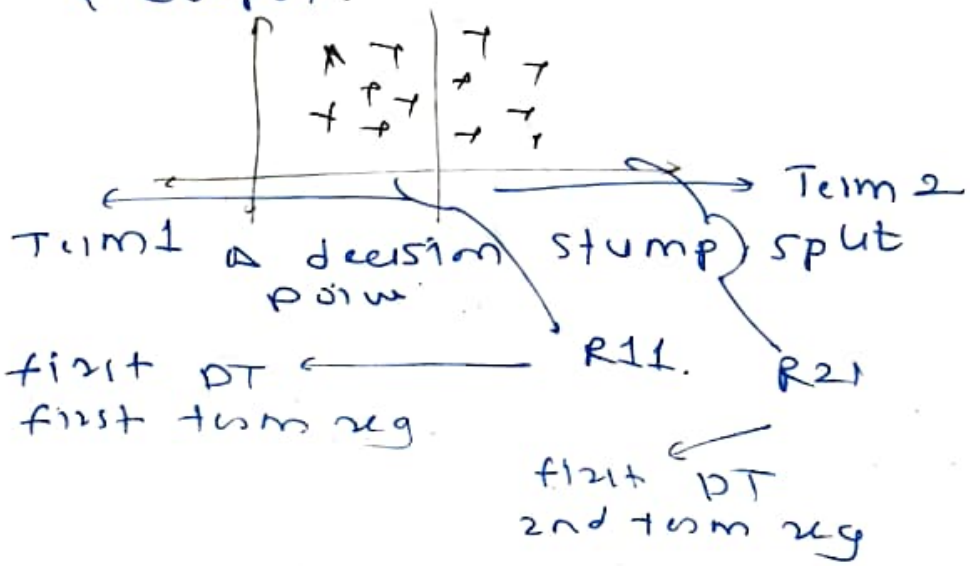
$r_{31} = y_3 - f_0(x_3) = 91 - 142 = -51$

B) fit a regression tree, to targets r_m, using terminal regions $R_{jm}, j = 1, 2, 3 \ldots m$

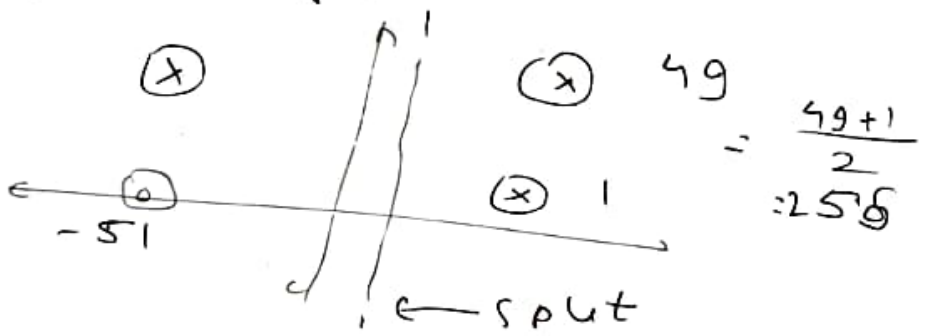
In short, fit a regression tree as data X input & r_{ti} as output

What is a terminal region? Answer in DT working.

We say, A dataset, Input, & Output.



In decision tree, there is a 'value' named value, actually it is output.



(c) calculate output for every terminal region.

$$r_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$

r_{j1} → r_{j1}, r_{j2}

$$\gamma_{11} = \arg\min_{\gamma} \sum_{x_i \in R_{11}} L(y_i, f_{(m-1)}(x_i) + \gamma)$$

so, basically $x_i \in R_{11}$ means, consider rows fall in that region/ leaf only.

$$\gamma_{11} = \arg\min_{\gamma} \left(\frac{1}{2}(y_i - (f_0(x_i) + \gamma)) \right)$$

$$\frac{dL}{d\gamma} = \frac{1}{2} \cdot 2(y_i - (f_0(x) + \gamma))$$

$$\frac{d}{d\gamma}(y_i - f_0(x) - \gamma) = 0$$

$$= -(y_i - f_0(x) - \gamma) = 0$$

$$= y_i - f_0(x) - \gamma = 0 \quad \text{——— (2)}$$

$$\gamma_{11} = 91 - 142 - \gamma = 0$$

$$\gamma = 91 - 142 = -51$$

$$\gamma_{21} = \arg\min_{\gamma} \left(\sum_{i=r_2} L(y_i - f_0(x_i + \gamma)) \right)$$

$$= \arg\min_{\gamma} \left(\sum_{i=1}^{2} (y_i - f_0(x_i) + \gamma)^2 \right)$$

$$= \sum_{i=1}^{2} (y_i - f_0(x_i) - \gamma) = 0$$

$$= \sum_{i=1}^{2} (y_i - f_0(x) - \gamma) = 0$$

$$= y_1 - f_0(x_1) - \gamma + y_2 - f_0(x_2) - \gamma = 0$$

$$= 192 - 142 + \gamma + 144 - 142 - \gamma = 0 = 26$$

you may note that, this is exactly same output as a leaf of decision tree

But But - this was due to loss function ordinary square.

like, we can consider output of DT, as output of OLS.

(1) update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

DT output

$f_1(x) = f_0(x) + DT_1$

$f_2(x) = f_1(x) + DT_2$

$f_3(x) = f_2(x) + DT_3$

Step 3) output $= \hat{f}(x) = f_M(x)$

Gradient boosting in classification

for example, let us consider 3 models

$M_1 \rightarrow M_2 \rightarrow M_3$

$f_0(x)$ $f_1(x)$ $f_2(x)$

phase 1) build M_1 - a very simple model

$$\log(\text{odds}) = \log_e\left(\frac{n_0(1)}{n_0(0)}\right)$$

This is our first model.

we need to calculate residuals. So, we do need something

But we can't use that fun directly

Instead, It is giving us probabuty.

To handle log(odds) and probabuty.

we need to calculate [illegible] from the

$$p = \frac{1}{1+e^{-\log(odd)}}$$

So for the dataset,

cgpa	IQ	pred_1 log(odds)	prob(pred)	res
6.82	95	0	0.5108	0.625 -0.625
6.34	125	1	0.5108	0.625 -0.375
5.39	99	1	0.5108	0.625 -0.375
5.50	106	1	0.5108	0.625 -0.375
6.39	148	0	0.5108	0.625 -0.625
9.13	148	1	0.5108	0.625 -0.375
7.17	147	1	0.5108	0.625 -0.375
7.72	72	0	0.5108	0.625 -0.625

So, Now us let set up a threshold of 0.5, like if $0.50 < p \rightarrow 1$, else 0.

Here by M1, all are placed.

phase 2 M_2

Calculate errors of M_1.

pseudo-residual = $y - f_0(x)$

So, res col^n is as followed in table

Now, for model 2, train a DT with same input but res1 as output.

So train a model decision tree regressor

Here, In the decision tree, we will get a value we used to put it directly in regression.

But now, This is probability. To predict,

$f_0(x) + f_1(x)$, ie

$\log(odds) + f_1(m2)$

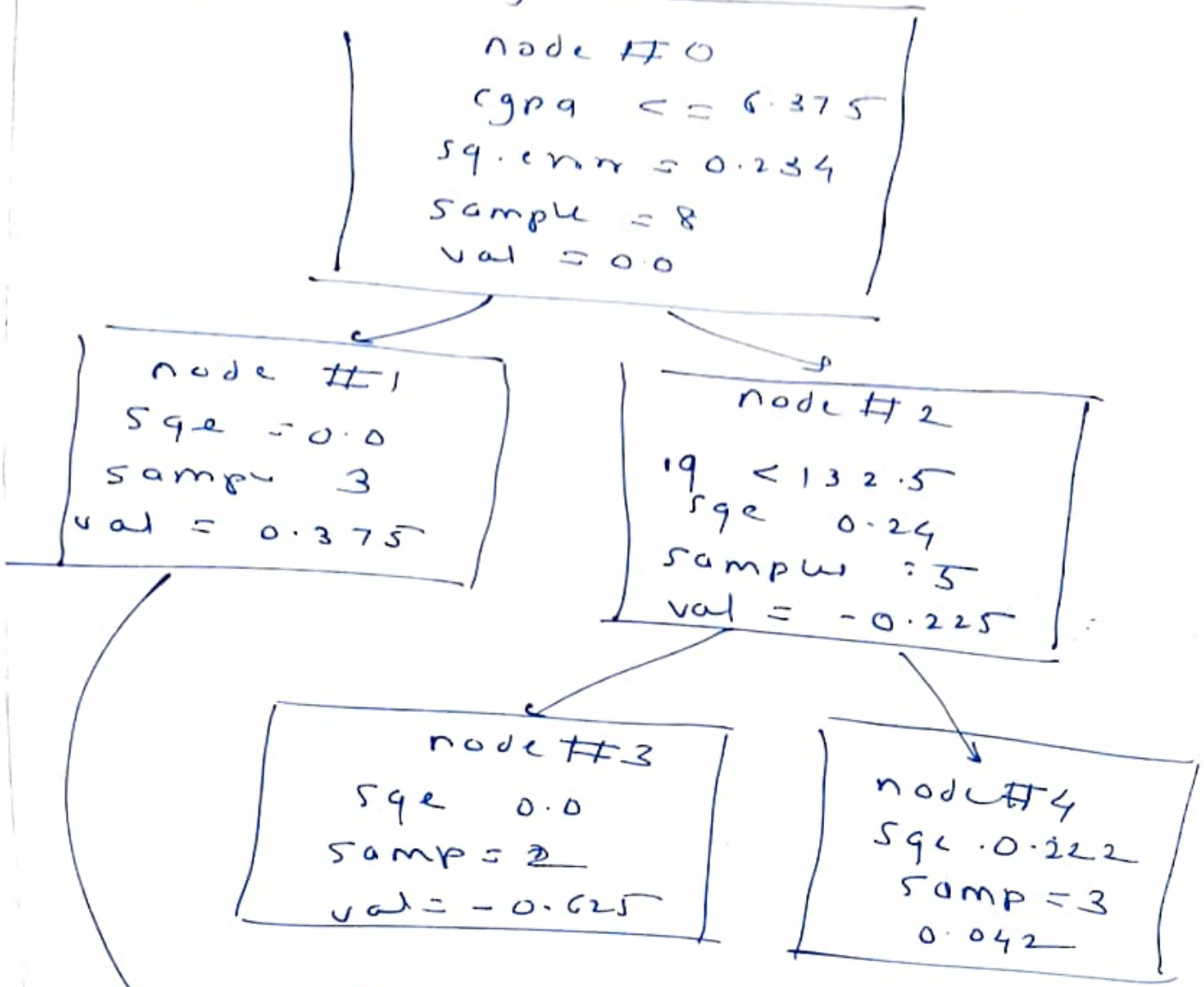
we cannot add probability directly.

So, we need to convert these probs to log(odds).

for the purple (prob → log(odds)), we use the formula for every leaf node

$$\left[\frac{\sum \text{residual}}{\sum[(\text{prev prob}) * (1 - \text{prev prob})]}\right]$$

So, for ~~right~~ now, My DT is



find for node1, find all points fall in node1.

Sklearn tells us that node for node 1, rows are [2, 3, 4].

Now, let us calculate log(odds) for node 2, rows are 1 and 8.

for node #3,
rows are 1 and 8.

Σ residual $= -0.625 - 0.625$
$= -1.250$

Σ(pair prob) $=$ for (1) $+$ for (8)
$= 0.625$

so,

$$\frac{-1.250}{(0.625)(1-0.625)+(0.625)(1-0.625)}$$

$$= \frac{(-2)(0.625)}{(0.625 \times 0.375)+(0.625 \times 0.375)}$$

$$= \frac{(-2)(0.625)}{2(0.625 \times 0.375)}$$

$$= \frac{-1}{0.375} = -2.66$$

so, log(odd for node #3) $= -2.66$

for node #1
rows are: (2, 3, 4).

so, $$\frac{\Sigma(\text{residual})}{\Sigma[(\text{pair\_prob})(1-\text{pair\_prob})]}$$

$$= \frac{0.375 + 0.375 + 0.375}{3[(0.625)(1-0.625)]}$$

$$= \frac{0.375}{(0.625)(1-0.625)}$$

$$= \frac{1}{0.625}$$

log(odds(node #3) $= 1.66$

for node #4

$$= \frac{-0.625 + 0.375 + 0.375}{3 \times 0.625 \times (1-0.625)}$$

$$= \frac{0.125}{3 \times 0.625 \times 0.375} = 0.18$$

log(odds(node #1)) = 0.1.6
log(odds(node #3)) = - 2.6
log(odds(node #4)) = 0.18

Now,

pred_until_now = M1 + M2

= log(odds(m1)) + log(odds(m2))

for leaf 3 (1,6) = 0.51 + (-2.6) = -2.16

for leaf 1 (2,3,4) = 0.51 + (1.6) = 2.11

for leaf 4 (5,6,7) = 0.51 + (0.16) = 0.67

Now, we will convert this again to probab, because we need to calculate residual

r(i).2 = 4 - [f0(x) + f1(x)]

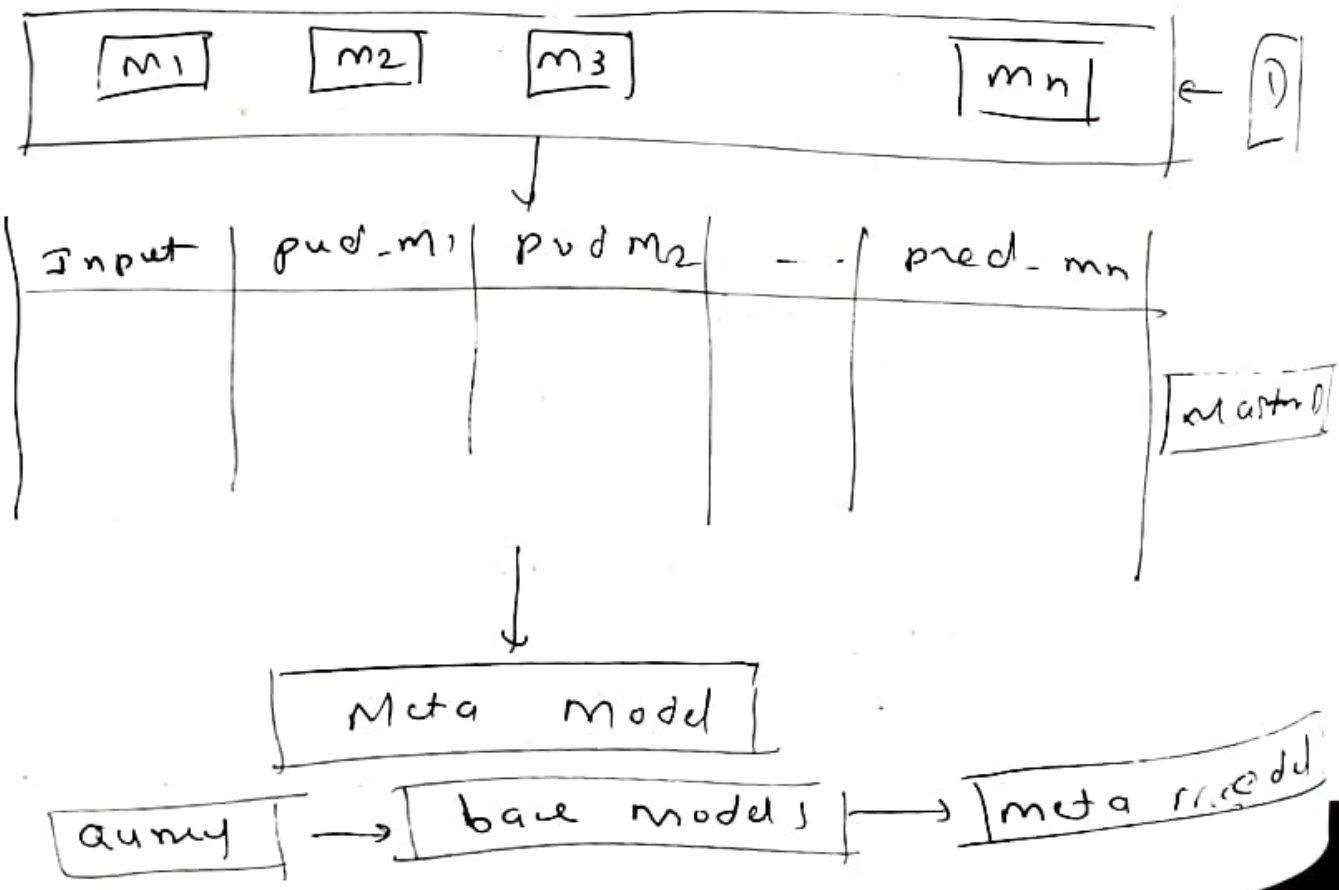
Now, again train DT2 & M3, as same input but res 2.

if these jumps are high, use learning rate

i.e, $r_{i1} = 4 - [\ f_0(x) - (0.1) f_1(x)\ ]$

$pred = \frac{f_0(x)}{\text{odd log}} + f_1(x) + f_2(x)$

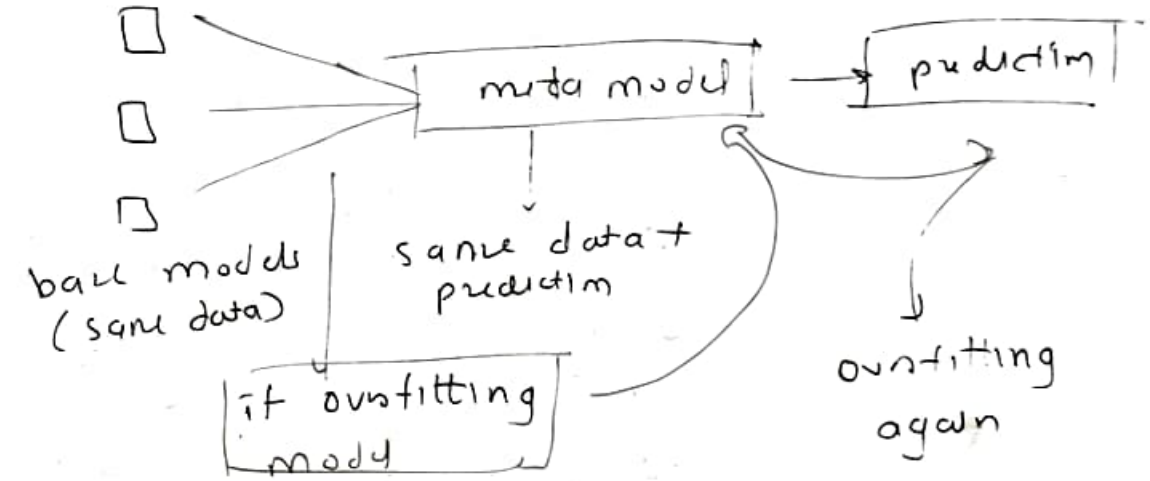
Stacking



Difference stacking V/S bagging / boosting

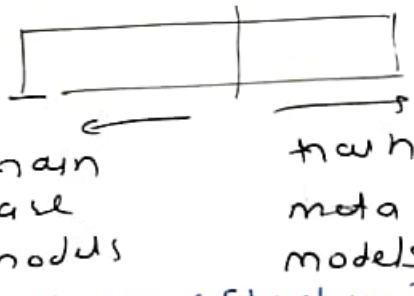
	stacking	bagging / boosting
base models	can be different	can be same. only
output use	Indirect	direct

problem ⚠

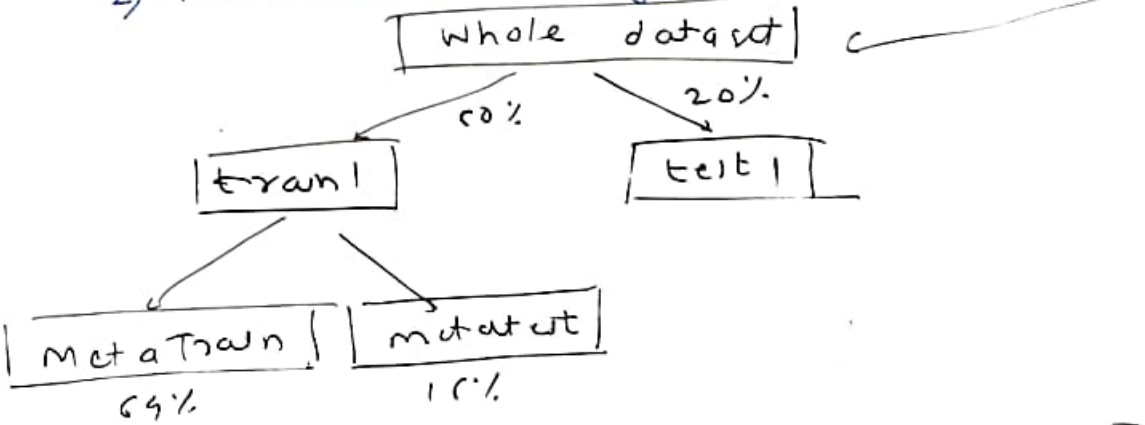


Solutions ✓

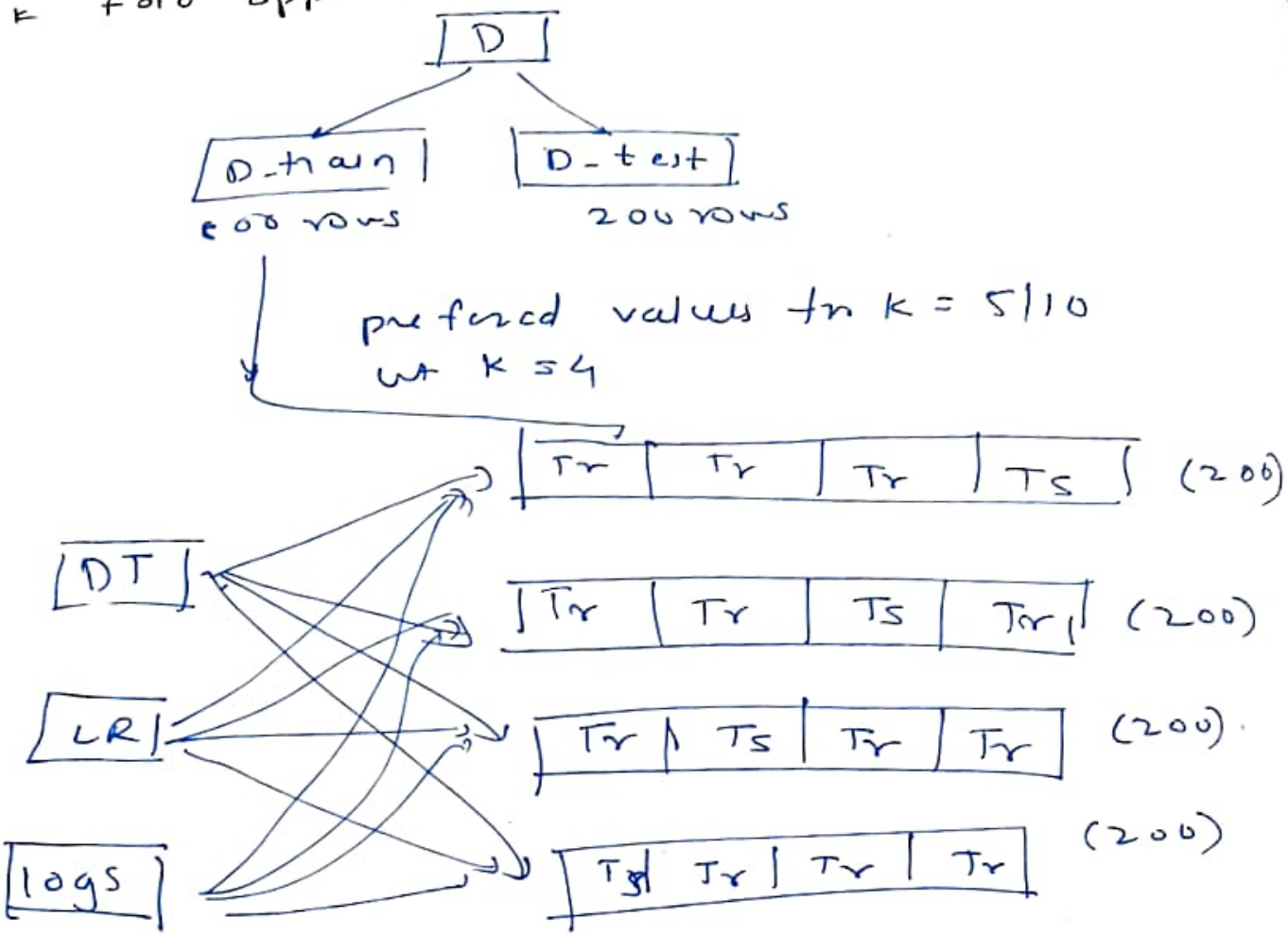
1) Hold out method (Blending)



2) k-fold (stacking) → (further)



k fold approach



DT	LR	logs	actual
200 + 200 + 200 + 200 = 600	—	—	—

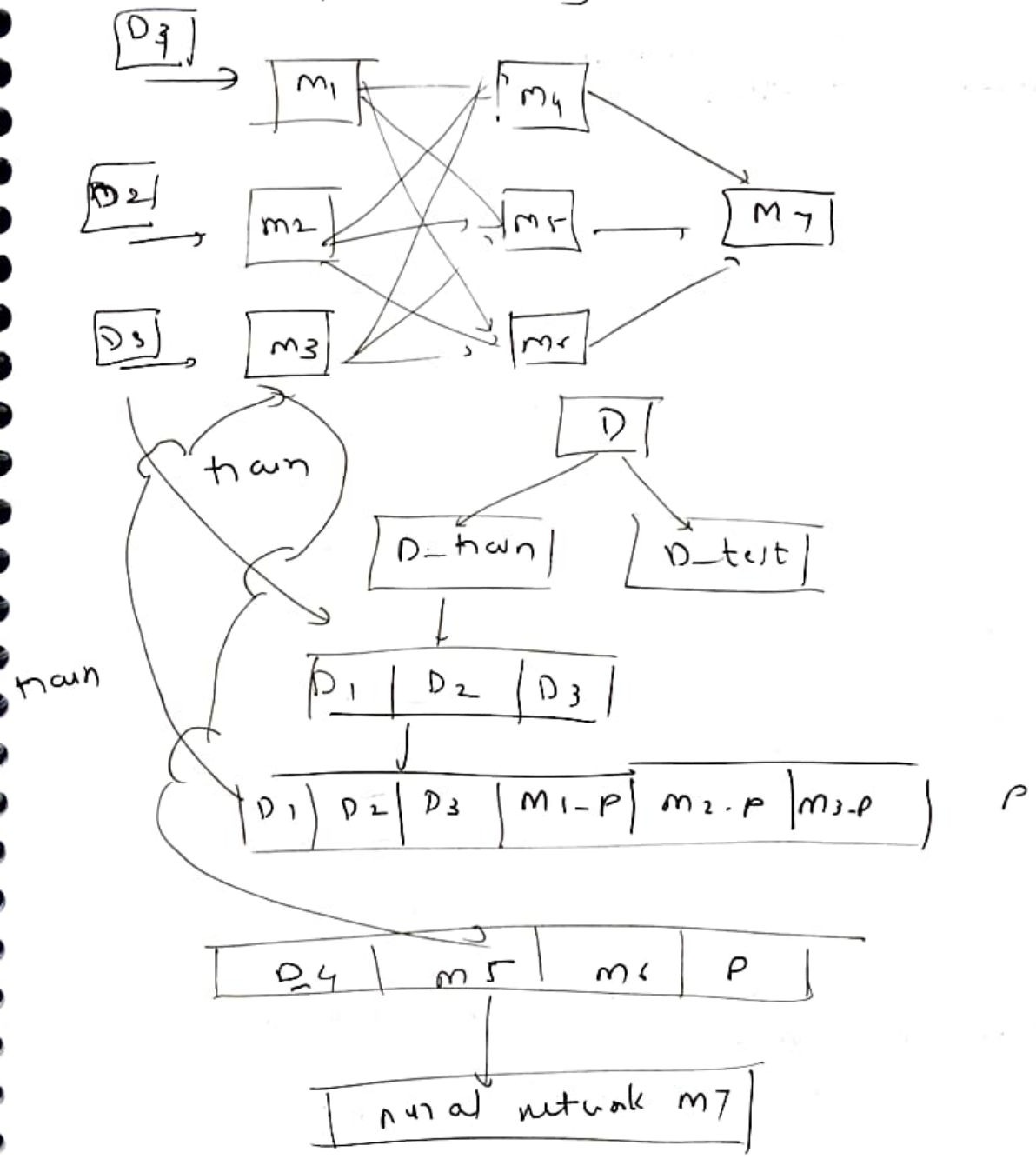
50, 12 models trained.

meta model

So) But which base model to choose?
Lift whole d_train,
train LR / DT and logs
(retrain)
So,

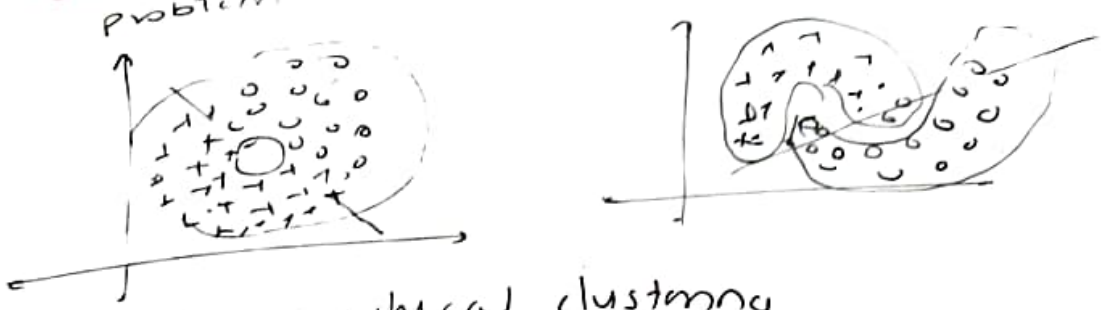
meta model → base model

Multi-layer stacking



Agglomerative hierarchical clustering

Problem is in centroid working.



Types of hierarchical clustering

① Agglomerative clustering
Divisive clustering

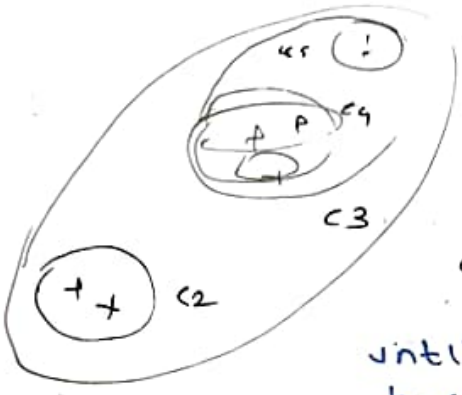
Dendogram

Initially, make assume that each point is an individual cluster.

Then, start merging them, and keep a record.

(2) Divisive clustering

exact opposite to agglomerative



until every point, become a cluster.

But, merging is easy than breaking.

Algorithm

1) Initialize proximity matrix

	P_1		P_2	P_3		P_n
P_1	0	d_1	d_2	d_3		
P_2	d_1	0				
P_4	d_2		0			
P_n	d_3					
					0	

2) make each point as a cluster.

3) Inside loop

merge the two closest clusters

update proximity matrix.

	P_1	P_2	P_3	P_5
P_1	0	-	-	-
P_2	0	-	-	-
P_1	-	0	-	-
P_5	-	-	0	-

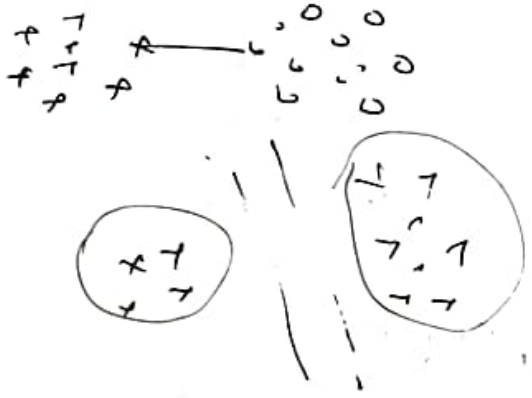
calculate these again

To find the distances can vary, it depends on what method we use to calculate distances between two clusters.

Types of agglomerative clustering

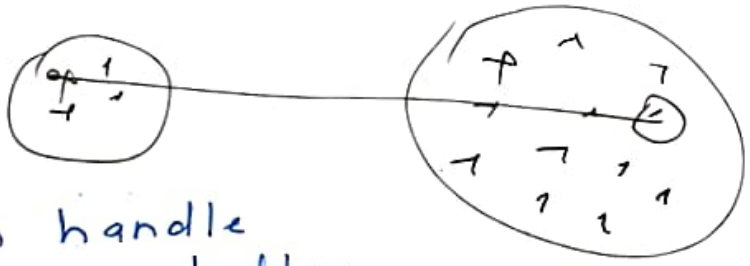
1) min (single link)
2) max (complete link)
3) average
4) ward

1) min (single link)



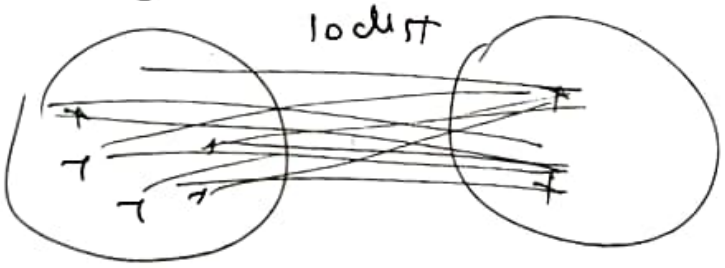
good in logic, can cluster if there is a gap.
But, fails if outliers.

2) complete link



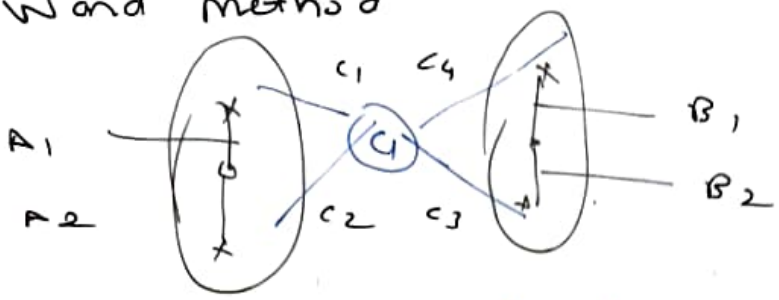
It can handle noise / outliers better.

3) group average



$$\frac{(\text{dist})\,10}{2} = \text{gap / distance bet}^{n}\text{ clusters.}$$

4) Ward method



1) plot a centroid.
2) Calculate distance between points and centroid.

So, Distance is,

$$d = c_1^2 + c_2^2 + c_3^2 + c_4^2 - (A_1^2 + A_2^2 + A_3^2 + A_4^2)$$

To minize the variance

Default in sklearn.

How to decide no of clusters?

plot a dendogram, and see longest vertical lines.

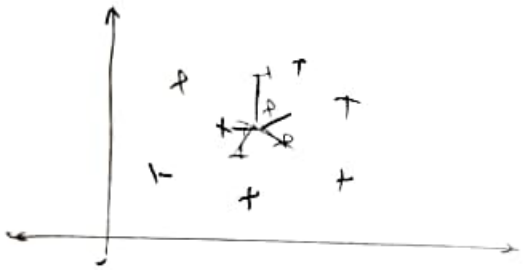
Hyperparameters (Agglomerative Clustering)

1) n_clusters
2) affinity, default - 'euclidian'
3) Linkage = ward. (different types of agglomerative clustering)
4) Distance_threshold.

Benefit
1) widely applicable
2) dendogram

disadvantage
1) cannot apply on large datasets

KNN - k nearest neighbours
you are like your k-neighbors.
see a datapoint,
calculate distances,
sort them, in ascending
take majority for first k.

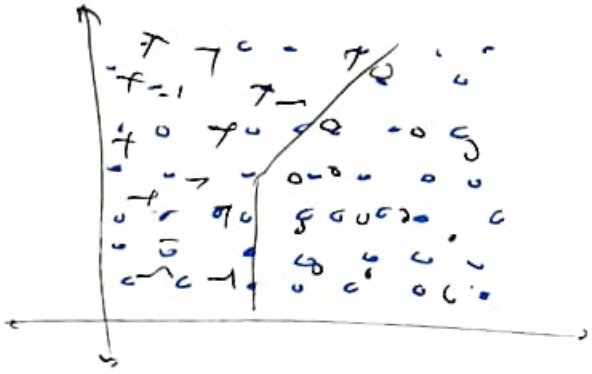


But how to decide k?
not any fixed answer.
1) hueristic approach
2) experiment

$\sqrt{n}$ → avoid even

split data, train diff KNN models for diff vals of k.

Decision Surface / Boundry



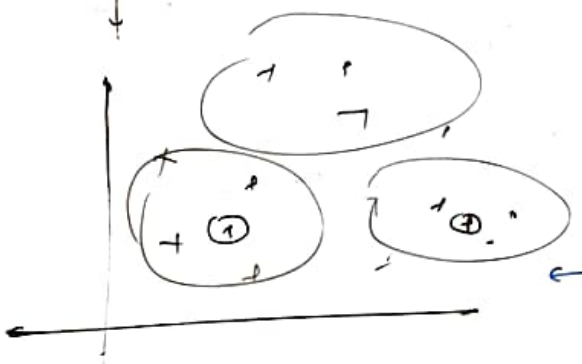
How to build?

1) plot training data
2) calculate range for axes
3) for range, generate a numpy matrix.
4) Train kNN by these points & send all points in it.
5) Encode points by the kNN response
7) Show points as a pixel on an image.

Overfitting & underfitting

overfitting starts

max K ? = N.



K = 3

overfitting

underfitting

you can't understand the data

Limitations of KNN

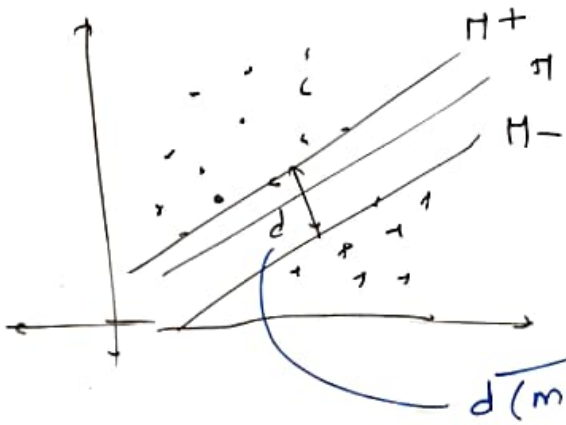
1) large datasets
2) Lazy learner,
   most of work of in predictions.
   In training, we do just store
3) In higher dimensions,
   cannot relate.
4) Imbalanced data
5) A black box model - No feature imp

SVM - support vector machine

← better?
Due to high distance
∴ less chance of overfitting

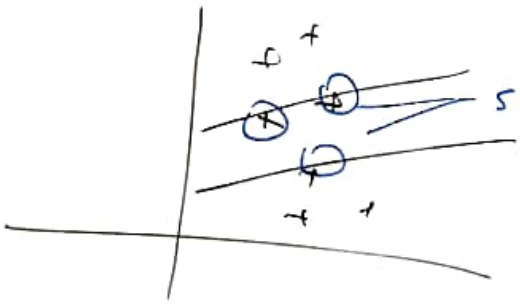
→ margin maximizing hyperplane

↓
how?



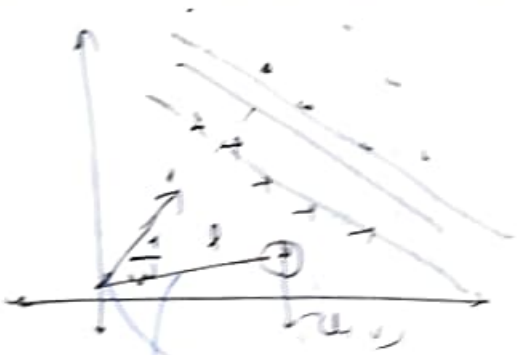
wx+b
is max.

d(max)



support vectors

1) Robust to outliers
2) non linear data handles
3) Classification & regression



$\vec{w}$ a perpendicular to all of hyperplanes

This could also be a vector.

Now, let us define us decision rule project u dot on w.

$\vec{w} \cdot \vec{u} \geq c$ if next (upon) to vector then positive
if before (down) to vector then negative

$\vec{w}\vec{u} - c \geq 0$

let replace $-c = b$

$\vec{w}\vec{u} - b = 0$

let say $u = (2, 3)$

$2x + 3y = 3$

$2(2) + 3(3) - 3 \geq 0$ (upon)

$u = (2, -4)$

$= 2(2) + 3(-4) - 3 = 0$

$= 4 - 12 - 3 = 0$

$= -11 < 0$ (down)

$w^T x + b = 1$ ← assumption

$w^T x - b = 0$

$w^T x + a = -1$ ←

for any $\vec{u}$

$(\vec{w}\vec{x}) + b > 0$

$\vec{w}\vec{x} - b < 0$

To find the distance d, we need to find the both lines equation.

As for now, we assume upper vectors equation is $w^T x + b = 1$ and lower vectors equation is $w^T x + b = -1$.

but why? why this magnitudes should 1 and -1?

because our (M) should exactly between (H^+) and (H^-). so equal magnitude (1 & -1).

Now why (1) only? Actually it doesn't matter, can be anything.

If we multiply that x, y coeff with val > 1, it will compact, shrink, & if val < 1, it will expand.

but to calculate loss, ie that boundries,

The point, either upper or lower for the hyperplane should satisfy the equation.

$$y_{i+1} \rightarrow \begin{cases} \bar{w}\bar{x} + b \geq 1 \\ \bar{w}\bar{x} + b \leq -1 \end{cases}$$

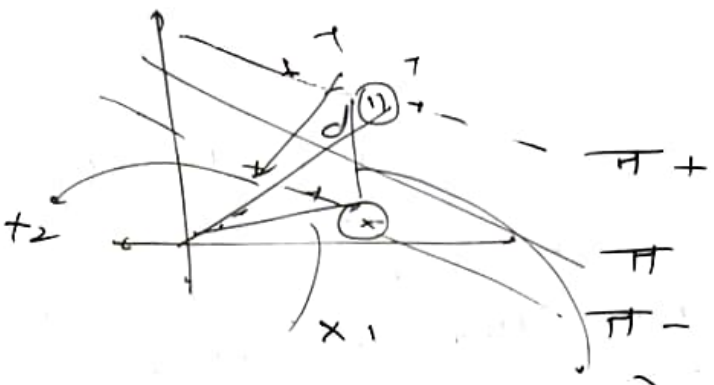
$$\hookrightarrow \begin{cases} (\bar{w}\bar{x} + b)(y_i) \geq 1 \\ (y_i)(w_i x + b) \geq 1 \end{cases}$$

y_i - neibghour

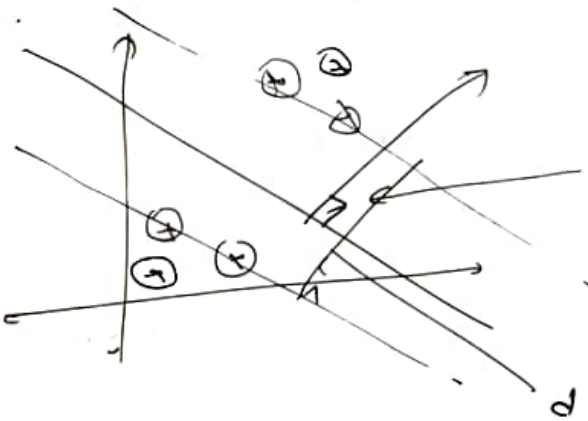
$$y_i(\bar{w}_i x_i + b) \geq 1$$

(= for support vectors $y_i(\bar{w}_i x + b) = 1$.)

↳ Calculate distance until this is true



But we cannot calculate distance this way.



$$= \frac{\bar{w}}{\|\bar{w}\|}$$

$$\therefore d = \left(\frac{\bar{w}}{\|\bar{w}\|}\right)(\bar{x}_2 - \bar{x}_1)$$

$$= \frac{\bar{x}_2 \cdot \bar{w} - \bar{x}_1 \cdot \bar{w}}{\|\bar{w}\|}$$

$$\text{if } (\bar{w}\bar{x}_1 + b) = 1$$

$$(1)(\bar{w}\bar{x}_2 + b) = 1$$

$$\bar{w}\bar{x}_1 = (1 - b)$$

also

$$(-1)(\bar{w}\bar{x}_1 + b) = 1$$

$$-b - 1 = -b - 1$$

$$\frac{(1-b) - (-b-1)}{\|\bar{w}\|}$$

$$= \frac{1 + b - b + 1}{\|\bar{w}\|} = \boxed{d = \frac{2}{\|\bar{w}\|}}$$

optimization function

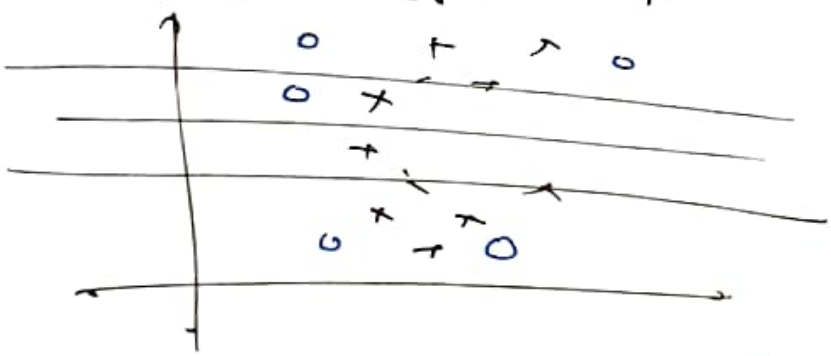
$$\underset{(w^*, b^*)}{\operatorname{argmax}} \left(\frac{2}{\|w\|} \text{ such that } y_i(\bar{w}\bar{x}_i + b) \geq 1 \right)$$

So, All this kind of clear linear separable data, was a hard margin SVM.

But it is possible in real world, we get non-linear data. For that, we need ~~hard~~ (soft) margin SVM.

Soft margin SVM

Make space for outliers!



$$\max(f(x)) = \min\left(\frac{1}{f(x)}\right)$$

$$= \frac{\|w\|}{2} + \sum_{i=1}^{n} \beta_i$$

for all every correctly classified point, β should be equal to 0. (0) error points.

$\beta \rightarrow$ distance from their respective hyperplanes.

Now, I wish to minimize their sum.

$$\text{SVM error} = \text{margin error} + \text{classification error}$$

$$\frac{\|w\|}{2} + C \sum_{i=1}^{n} \beta_i$$

if C high $\rightarrow$ Don't worry about width worry about misclassif.

algo should not misclassify any point

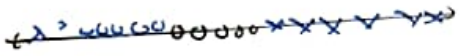
if C is very small, Ignore classification error, focus on margin error, & vice versa for higher C.

In logistic also

logistic loss + $\lambda \|v\|$

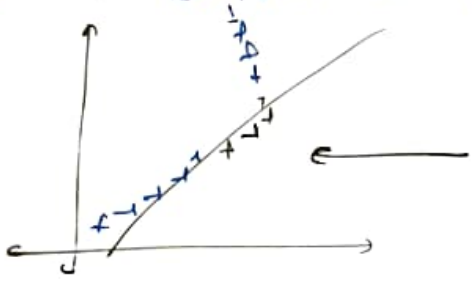
$C \propto \frac{1}{\lambda}$

Kernel trick in SVM



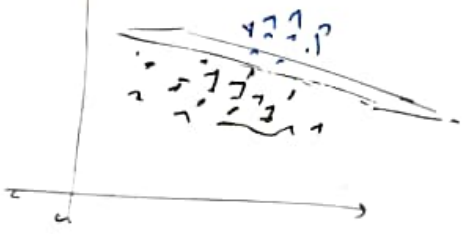
We cannot classify this, as it is a non linear data

But by some mathematical eqns, we can convert it to higher dimensionals



← kernel
RBF
polynomial
signoid

$\longrightarrow f(x) = e^{-\gamma L}$



Naive Bayes

Conditional probabuty

A, B

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \quad \text{given} \quad P(B) \neq 0$$

Let us consider a dice's problem.

two dices

Sample space $S = \{(1,1), (1,2), \ldots (1,6),$
$(2,1), (2,2) \ldots (2,6)$
$\ldots \ldots (6,6)\}$

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

1) $P(D_1 = 5) = \frac{6}{36} = \frac{1}{6}$

2) $P(D_1 + D_2 \leq 10) = \frac{33}{36} = \frac{11}{12}$

3) $D_1 = 5$ given $D_1 + D_2 \leq 10$

A B_1

$$P(D_1 = 5 \mid D_1 + D_2 \leq 10) =$$

$$P(A|B) = \frac{5}{33}$$

$$P(A \cap B) = \frac{5}{36} \Big/ \frac{33}{36} = \frac{5}{33}$$

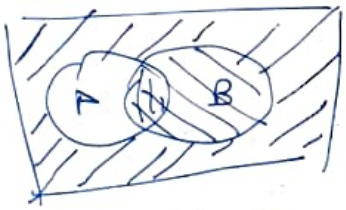
Independent evnts

ex.
Two dies rolled, If die 1 comes 3, what is prob for die 2 to come 6? They are indepint, No matter to other.

$$P(A \cap B) = P(A) + P(B)$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \quad \text{—— (1)}$$

this way, P(B) is already happin. We are looking for existance of A in that.



$$P(A) = \frac{P(A \cap B)}{P(B)}$$

$$= \frac{n(A \cap B)}{\cancel{n(S)} \times \frac{n(B)}{\cancel{n(S)}}}$$

$$= P(A) \frac{n(A \cap B)}{n(B)} \quad \text{—— (2)}$$

$$\boxed{P(A) = P(A|B)}$$

B doesn't matter. No contribution

Mutually exclusive evnt

for A, B

$$P(A \cap B) = 0$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$P(A|B) = 0$$

example,
you are driving car. what is probability of turning right if we took left already?
like 2 dices, In first dices, come 3, what is prob to come 6 in first itself?

Bayes theorm

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

likelihood → $P(B|A)$; prior → $P(A)$; posterior ← $P(A|B)$; evidence ← $P(B)$

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \quad \text{—— (1)}$$

$A \cap B = B \cap A$

$$P(B|A) = \frac{P(B \cap A)}{P(A)} = \frac{P(A \cap B)}{P(A)}$$

$$P(A \cap B) = P(B|A) \cdot P(A) \quad \text{—— (2)}$$

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

example by bayes theorm

factory

m_1	m_2	m_3	
(20%)	(30%)	(50%)	manufacturing
(5%)	(3%)	(1%)	manufacturing defect

Now, let say, I picked on object, which was defective from random bag. Now, tell probabuty it was from m_3.

$P(m_1) = \frac{1}{5}$ $P(m_2) = \frac{3}{10}$ $P(m_3) = \frac{1}{2}$

$P(D|m_1) = \frac{1}{20}$ $P(D|m_2) = \frac{3}{100}$ $P(D|m_3) = \frac{1}{100}$

$P(M_3|D)$ = prob of m_3, given it is defective

$= P(D|M_3) \times \frac{P(m_3)}{P(D)}$

$P(D|M_3) = \frac{1}{100}$

$P(m_3) = \frac{1}{2}$

$P(D) = P(D \cap M_1) + P(D \cap M_2) + P(D \cap m_3)$

obj Being def & come from M_1

$P(D) = P(D|M_1)(P(m_2)) + P(D|M_2)\,P(m_2) + P(D|M_3)\,P(m_3)$

$= \left(\frac{1}{20}\right)\left(\frac{1}{5}\right) + \left(\frac{3}{10}\right)\left(\frac{3}{100}\right) + \left(\frac{1}{2}\right)\left(\frac{1}{100}\right)$

$= \frac{1}{100} + \frac{9}{1000} + \frac{1}{200}$

$= \frac{10 + 9 + 5}{1000} = \frac{24}{1000}$ — ①

Now,

$= \frac{\left(\frac{1}{100}\right)\left(\frac{1}{2}\right)}{\left(\frac{24}{1000}\right)} = \left(\frac{1}{200}\right) / \left(\frac{24}{1000}\right) = \frac{5}{24}$

Sample example and mathematics

toss	CSK venue	outlook	result
won	mumbai	Overcast	won
lost	chennai	Sunny	won
won	kolkata	Sunny	won
won	C	Sunny	won
lost	M	Sunny	lost
won	C	Overcast	lost
won	K	Overcast	lost
won	M	Sunny	won

Now, predict win loss by datapoint {lost, Mumbai, sunny}

Prob(lost ∩ mumbai ∩ sunny) = ?
by Naive bayes.

In more clearity

P(winning | (lost ∩ mumbai ∩ sunny))
P(lost | (lost ∩ mumbai ∩ sunny))

lost: find probab
(lost ∩ mumbai ∩ sunny): Given condition

will find prob, greater one will answer.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

$$P(W \mid \text{lost, mumbai, sunny}) = \frac{P(\text{lost, mumbai, sunny} \mid W)\,P(W)}{P(\text{lost, mumbai, sunny})}$$

$$P(L \mid \text{lost, mumbai, sunny}) = \frac{P(\text{lost, mumbai, sunny} \mid L)\,P(L)}{P(\text{lost, mumbai, sunny})}$$

At the end, we will divide greater one, so, we don't need denominator, which is same in both cases.

So, remove that

$P(W \mid \text{lost, mumbai, sunny}) =$

$P(\text{lost, mumbai, sunny} \mid W)\, P(W)$

$P(L \mid \text{lost, mumbai, sunny}) =$

$P(\text{lost, mumbai, sunny} \mid L)\, P(L)$ ——①

"Naive bayes", because, it applies a very simplistic assumption on bayes theorm.

$P(L \mid \text{lost, mumbai, sunny}) =$

$P(\text{lost} \mid L)\, P(\text{mumbai} \mid L)\, P(\text{sunny} \mid L)\, P(L)$

Just did, because [——①] is so specific seens to be occur very unlikely,

win $= \left(\frac{1}{8}\right)\left(\frac{2}{5}\right)\left(\frac{4}{5}\right) \times \left(\frac{5}{8}\right) = \frac{1}{25} = 0.040$

loss $= \left(\frac{1}{3}\right)\left(\frac{1}{3}\right)\left(\frac{1}{3}\right)\left(\frac{3}{5}\right) = \frac{1}{72} = 0.013$

Core mathematics

$X = \{x_1, x_2 \ldots x_n\}$

$C = \{C_1, C_2 \ldots C_n\}$

$P(C_k \mid x) = \dfrac{P(X \mid C_k)\, P(C_k)}{P(X)}$ ← no contrib

$\propto P(X \mid C_k)\, P(C_k)$

$P(C_k \mid x) \propto P(X \cap C_k)$

$P(A \mid B) = \dfrac{P(A \cap B)}{P(B)}$

$P(A \mid B)\, P(B) = P(A \cap B)$

$$P(C_k|\vec{x}) = P(x \cap C_k) \quad \text{——— (1)}$$

$$P(C_k|x) = P(x_1, C_k)$$

$$P(\underbrace{x_1, x_2, x_3 \ldots x_n}_{A \cap B}, C_k)$$

Now, assume

$$P(A|B)$$

$$P(x_1 | x_2 x_3 \ldots x_n, C_k)\, P(x_1, x_2, x_3 \ldots x_n, C_k)$$

$$P(C_k|x) = a \cdot P(x_2, x_3 \ldots x_n, C_k)$$

Now,

$$P(x_2, x_3, x_n, C_k)$$

$$= a \cdot P(x_2 | x_3 x_4 \ldots x_n, C_k)\, P(x_3 x_4 \ldots x_n, C_k)$$

$$= a \cdot b\, P(x_3, x_4 \ldots x_n, C_k)$$

chain rule for conditional probability

↓ break down

↓ break down

last term =

$$P(x_n | C_k)\, P(C_k)$$

$$= P(x_1 | x_2 x_3 x_4 \ldots x_n, C_k) \cdot P(x_2 | x_3 x_4 x_5 \ldots x_n, C_k) \cdot P(x_3 | x_4 x_5 x_6 \ldots x_n, C_k) \cdot \ldots \cdot P(x_{n-1} | x_n, C_k) \cdot P(x_n | C_k) \cdot P(C_k)$$

this only depends on C_k. — Assumption

↓

[conditional independence] ——— (2)

we are taking the assumption because, finding that very specific event probability is very very less or even zero.

Independent event

$$P(A|\bar{B}) = P(A)$$

So, further,

$$P(A|B,C) = P(A|C)$$

↑ A is independent on B conditionally

So, further simplifying eqn (2)

$$= P(x_1|C_k)\,P(X_2|C_k)\ldots P(X_n|C_k)\,P(C_k)$$

$$\therefore P(C_k|x) = P(C_k)\prod_{i=1}^{n} P(x_i/C_k)$$

↓ actually

$$\therefore P(C_k|x) \propto P(C_k)\prod_{i=1}^{n} P(x_i/C_k) \quad \text{— (3)}$$

↓

$$= P(C_k|x) = \frac{1}{z}\left(P(C_k)\prod_{i=1}^{n} P(x_i/C_k)\right)$$

$z = P(x)$

Ignore coz it will come same

$$\hat{y} = \underset{k\in\{1,2\ldots k\}}{\arg\max}\left\{P(C_k))\prod_{i=1}^{n} P(x_i/C_k)\right.$$

⇐

---

maximum a posterior rule

map

Horse numerical data

Height	weight	gender
172	150	M
180	170	M
165	140	M
190	200	M
139	100	f
145	120	f
160	140	f
172	150	f

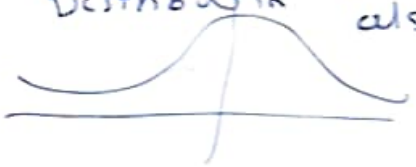
find { H = 185, W = 170. G }

$P(M \mid H = 165, W = 170)$

$= P(H = 165 \mid M)\, P(W = 170 \mid M)\, P(M)$

Assumption :

Height is a gaussion Distribution also weight



find out mean and σ.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Gaussian Naive Bayes

use histplot, and find best one fit for your case

use GaussianNV,
BinomialNV,
PoissonNV

XGBoost - A rockstar guy

To understand, first, let revise ML history

(70-80) → simple algorithm, that can only work on a specific kind of data

(80,90) → some complex SVM, random forest powerful But lead to overfitting and scalability is not much

Introducing xgBoost (power, performance, Scalability, speed).

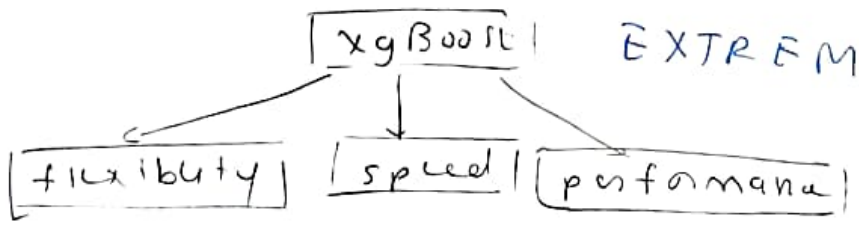
fact: It is a library!

Tinqi chain, made xgBoost by SE concepts and gradient boosting.

History → kaggle → open source power

early days (2014) grad boost
- flexibility
- loss fun
- performance
- kaggle
- robust — performance big data

kaggle work →
- features
- optimization
- multiple features kaggle



1) flexibuty
   1) cross platform (windows, unux)
   2) multiple language support
   3) Integration with other ubs and tools
   4) confestabie with any ML problem.

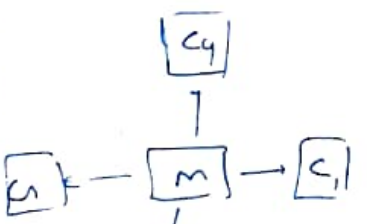
2) Speced
   1) parallel processing
      It is a sequintial boosting tnck.
      To build a model indivisually,
      we usl parallel processing.
      like In a dataset, whw multiple
      features, & we will In that
      column assume a column, so.
      In case of DT, we will found
      bit sputs, by gini and enthopy,
      and thus computation will
      individual for any column so
      parallel computing.

   2) Optimized data structure
      others intorpute row-wise
      It (Xgb) is columnwise

   3) Cache awareness
      on numrical features, Xgboost
      calculate bins, We need it regularly,
      so we store it in cache memory.
      and ull thus, many things.

   4) Out of core computing
      splitting data / making chunks
      that cannot fit in computers
      main memory.

   5) Distributed computing
      perform computing parallel on
      diffrent nodes (machines)



      1) split data
      2) local each data on a single
         node
      3) one master node perforns
         agressim & communicatim
         (dask,
         kubernetes)

   6) GPU support

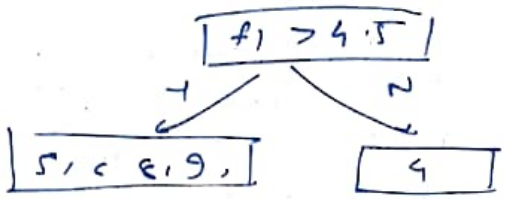
3) Performance

1) Regularization learning objective, uk in Uncor modus, to reduce overfitting
2) handling missing values
3) sparsity aware split finding
4) efficient split finding (weighted quantile search + Approximate tree learning)
5) Tree pruning (cut down depth of tree) regularization term in loss fun? itself.

xgb detects sparsity (very large 0 values) and handles it. But how?

f

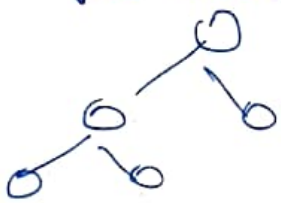
4 — 4.5
5
6 — 5.5
7
8 — 6.5
9



Now, send missing value in both node & calculate gain., and go with max gain node

Trying to make sence of direction of missing value

We train a lot of trees, likely,



f1
3
5
7
9
11
13

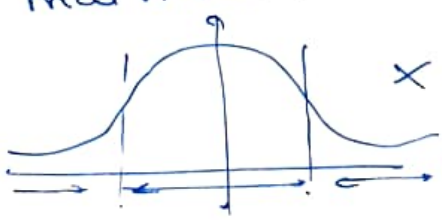
→ exact greedy search, and splitting criteria

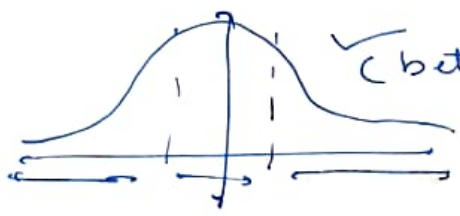
↓

But, it is slow approach

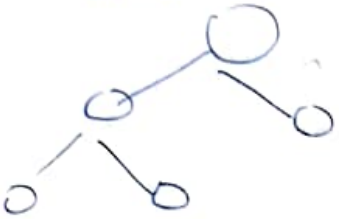
← Instad for the numerical column, we divide it in bins. (continuous → descrete), This is approximate tree learning. (Histogram based) (weightage quantile sketch)

We use quantile binning instad of uniform, to maintain distribution, & decide bins.





f) tree pruning
To cut down tree for reducing complexity and overfitting.



→ reduce complexation and overfitting

post pruning

pre pruning

xgboost

prepruning (Hyperparameters relevance)

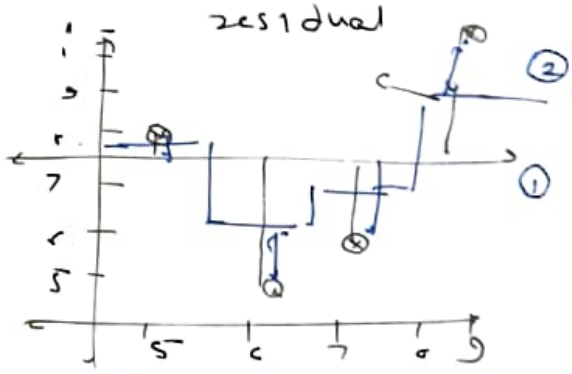
postpruning

gamma value
↓
To decide to make a new branch or not

Intro, related to gradient boosting
1) ~~Light GBM~~ Light GBM
2) cat Boost → (catagorical)

for regression

cgpa	package
6.7	4.5
9	11
7.5	6
5	6



like gradient boosting, start first model as mean. calculate residuals, Now, train a DT to predict residuals. calculate ① and, then predict DT, build model ②, predict errors of M2. train M3, same input, & res of M2 & it will starting, goes to accuracy
xGB will work same as GB
But one of the major difference is they train DT differently. unlike gini & entropy (vanila).
So, this way, DT train differently.

Algorithm

P1) Take mean as base estimator
calculate prediction & error.

cgpa	package	model1	res 1	model2	res 2
6.7	4.5	7.3	-2.8	6.69	-2.19
9	11	7.3	3.7	8.41	2.59
7.5	6	7.3	-1.3	7.29	-0.69
5	8	7.3	0.7	7.51	0.49

Now train a DT on cgpa and res 1.

before that,

form a leaf node for residual

LN = -2.8, 3.7, -1.3, 0.7

calculate similarity score.

$$SSc = \frac{(\text{sum of residuals})^2}{(\text{no of res}) + \lambda}$$

λ → regularization parameter

for right now, $\lambda = 0$

$$SSc = \frac{(-2.8 + 3.7 - 1.3 + 0.7)^2}{4 + 0}$$

$$= \frac{(0.3)^2}{4} = 0.02$$

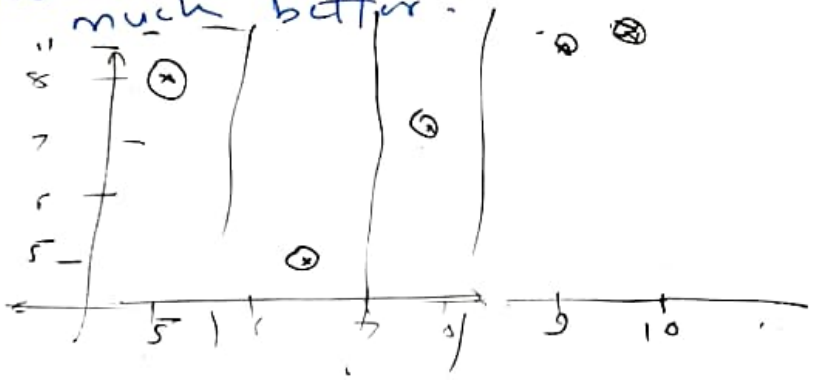
we will now try to split residuals and see if we can increase similarity score

sort them & calc mean,

5.0 6.7 7.5 9.0

5.85 7.1 8.25

Now see which one can improve similarity score much better.



cgpa < 5.45

0.7 | -2.6, -1.3, 3.7

$SSC_L = \frac{(0.7)^2}{1+0} = 0.49$

$SSR = \frac{(-2.6 - 1.3 + 3.7)^2}{3+0} = \frac{0.16}{3} = 0.05$

gain = (SSL + SSR) − SSroot
= (0.49 + 0.05) − 0.02
= 0.52

cgpa < 7.4

0.7, 2.6 | -1.3, 3.7

$SSL = \frac{(0.7 - 2.6)^2}{2} = \frac{4.41}{2} = 2.20$

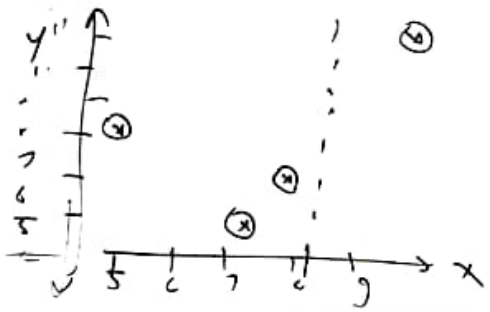
$SSR = \frac{(-1.3 + 3.7)^2}{2} = \frac{5.76}{2} = 2.88$

gain = (SSL + SSR) − SSroot
= (2.20 + 2.88) − 0.02
= 5.06

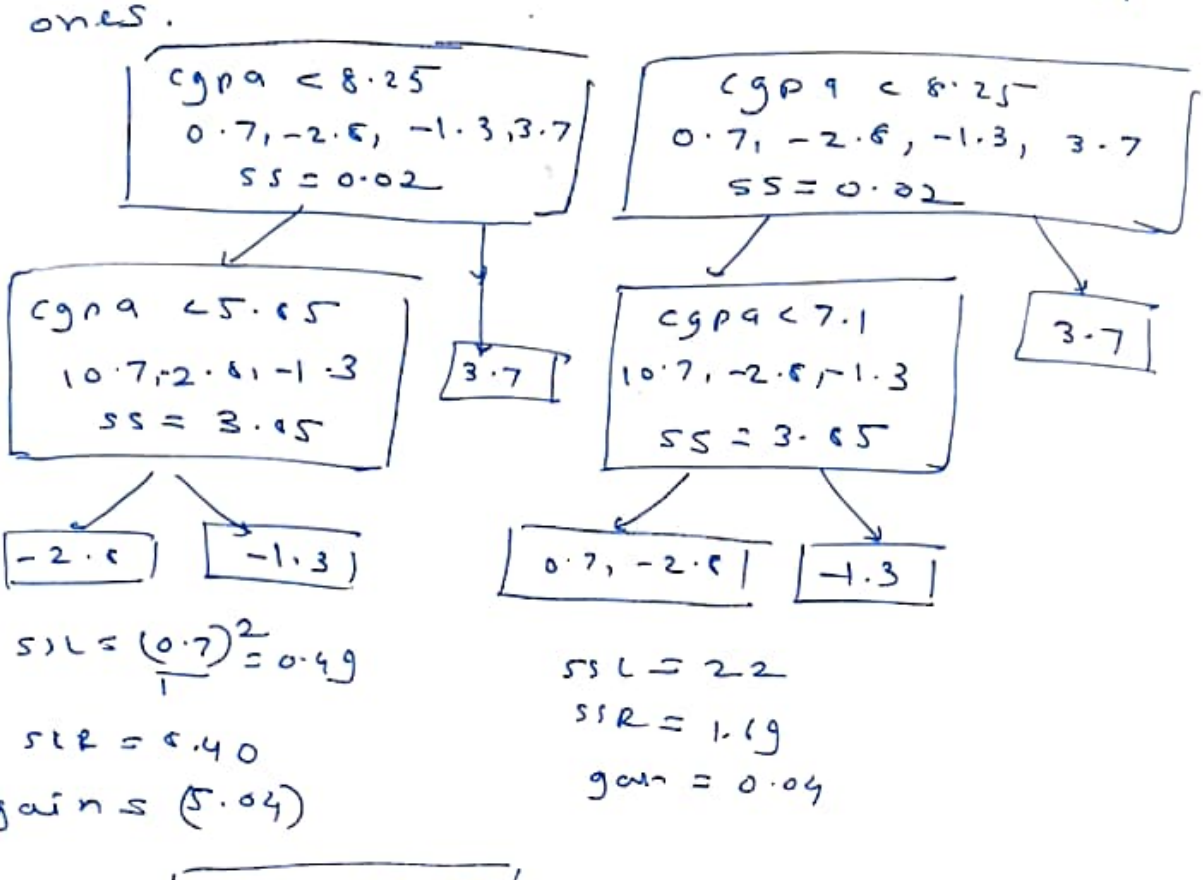
cgpa < 8.25

0.7, -2.6, 1.3 | 3.7

SSL = 3.55
SSR = 13.69
gain = 17.52 → choose this



Now, we can have two possibilities on left ones.



$SSL = \frac{(0.7)^2}{1} = 0.49$

$SSR = 5.40$

gain = (5.04)

$SSL = 22$

$SSR = 1.69$

gain = 0.04

3.7

0.7

$\frac{(-2.6 - 1.3)}{2} = -2.05$

for first value

package & cgpa 6.7 ← leaf

$(7.3) + (0.3)(+(-2.05)) = 6.69$

m1 ↑ (regularization) η

In ~~multiclass~~ multiple features
scan all, do all possible splits, calculate gain & SSC and ~~eat~~ consider max gain.
In categorical
Consider all uniques In categorical, and consider both as a potential split point, calculate gain & similarity with max gain.
go

multiclass categorical variables

Identify unique & consider everyone as a potential split point, calculate SSc & maximum gain.

Two approach:

1) one single node + rest in a group

{small} : { big, large tiny}

2) Non empty non ovalaping subset

{small, big} . { large, tiny}

Now, for every possible split calculate similarity scores, gain and go with max gain.

By the ~~west~~ way,

+ve is not only one single algorithm, The criteria we were tryout every criteria, this is exact greedy algorithm. in small dataset

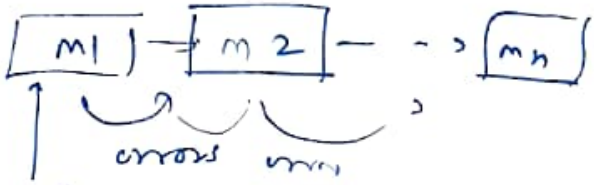
But it is not only way, we can use approximation in large datasets.

Xgboost in classification

We need to pudict a students placement.

cgpa	placement
5.70	0
6.25	1
7.10	0
8.15	1
9.00	1

Overview



DTs we train are not vanilla like DT. they are based on similarity score rather then gini/entropy

so, let's start problem solving.

phase 1) base model

$$\log(\text{odds}) = \left(\frac{p}{1-p}\right) = \log\left(\frac{p}{1-p}\right) = \log\left(\frac{315}{215}\right)$$

$$= \log\left(\frac{3}{2}\right) = \log(1.5)$$

$$= 0.405$$

But, we need to convert this to probability

$$p = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}} = \frac{e^{(0.405)}}{1 + e^{(0.405)}} = 0.60$$

with output to phase 1 model, we need to calculate pseudo residual. so, it is placed - prob

cgpa	placed	model	prob	resi
5.70	0	0.405	0.6	-0.6
6.25	1	0.405	0.6	0.4
7.10	0	0.405	0.6	-0.6
6.15	1	0.405	0.6	0.4
9.0	1	0.405	0.6	0.4

phase 2) DT

Sort data on basis of input

for a leaf node

-0.6, 0.4, -0.6, 0.4, 0.4

$$SS = \frac{(\Sigma \text{ residuals})^2}{\Sigma \text{ prev-prob}\,(1 - \text{prev-prob}) + \lambda}$$

$$= \frac{(-0.6 + 0.4 + 0.4 - 0.6 + 0.4)^2}{5(0.6[0.4])^2 + \lambda} = 0$$

$\lambda = 0$

now, split on splitting criteria (goal is to maximise similarity score, on basis of cgpa

splitting criteria is simple

5.70 6.25 7.10 6.15 9.60

5.97 6.68 7.52 8.67

calculate gain max on splitters.

Splitting criteria 5.97

cgpa < 5.97

-0.6	0.4, 0.6, 0.4, 0.4

SSL

$$\frac{(-0.6)^2}{(0.6)(0.4)}$$

$$= 1.5$$

SSR

$$\frac{(0.4+0.4-0.6+0.4)^2}{(4\times 0.6\times 0.4)}$$

$$= 0.37$$

gain

$$= (SSL - SSR) - SSroot$$

$$= (1.5 + 0.37) - 0$$

$$= 1.87$$

Splitting criteria 6.17

cgpa < 6.17

-0.6, 0.4	-0.6, 0.4, 0.4

SSL

$$\frac{(-0.2)^2}{2(0.6)(0.4)+0}$$

$$= \frac{0.04}{2\times 0.24}$$

$$= \frac{0.04}{0.24}$$

$$= 0.16$$

$$\frac{0.04}{3\times 0.24}$$

$$= 0.05$$

$$= 0.13$$

on 7.625

SSL = 0.88

SSR = 1.33

gain = 2.22

for 8.675

SSL = 0.17

SSR = 0.67

gain = 0.64

So DT is

cgpa < 0.713

-0.6, 0.4, -0.6	0.4, 0.4

output for a particular leaf for classification

$$\text{output} = \frac{\Sigma(\text{residual})}{\Sigma(\text{prev. prob})(1-\text{prev. prob}) + \lambda}$$

$$\frac{-0.6+0.4-0.6}{3(+0.6\times 0.4)}$$

$$= \frac{-0.8}{3\times(0.24)} = -1.11$$

$$\frac{(0.4+0.4)}{(2\times 0.24)}$$

$$= 1.66$$

so basically this model says,

cgpa < 7.66

-1.11 | 1.66

as an output

DT1

phase 2 model prediction

(M1) + (0.3)(M2)

log odds | DT1

(0.405 + (0.3) (DT1))

(2)

= 0.405

0.405 + (0.3) (-1.11) = 0.072

0.405 (+0.3) * 1.66 = 0.903 → log of odds format.

← we need it in probabilities

so we will do,

prob = $\frac{e^{\text{log odds for the residuals}}}{1 + e^{\text{log odds}}}$

ie = $\frac{e^{(0.072)}}{1 + e^{(0.072)}} = 0.518$

= $\frac{e^{(0.903)}}{1 + e^{(0.903)}} = 0.712$

mathematics behind xgboost

	Regression	Classification
SSC	$\frac{\Sigma(\text{residuals})^2}{N + \lambda}$	$\frac{\Sigma(\text{residuals})^2}{\Sigma(P(1-P)) + \lambda}$
OUT	$\frac{\Sigma(\text{residuals})}{N + \lambda}$	$\frac{\Sigma(\text{residuals})}{\Sigma(P(1-P)) + \lambda}$

for a combinatim chain of models,

$m_1 \rightarrow m_2 \rightarrow m_3 \rightarrow \dots \rightarrow m_n$

This way it is a chain of functions.

So, let us say,

$$\hat{y} = f_1(x_1) + f_2(x_2) \dots f_n(x_n)$$

$$\hat{y} = \sum_{i=1}^{n} f_i(x_i)$$

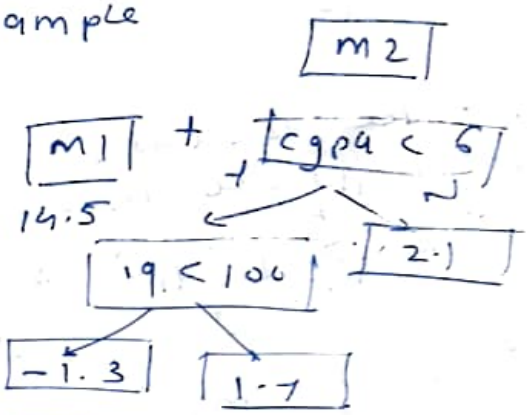
So, let us say our model is a xgb tree.

Now, for a output leaf weight (node) the formula is,

$$\text{out} = \frac{\text{sum of resid for node leaf}}{N + \lambda}$$

Let us take an example

cgpa	iq	percentage
6.5	71	13
5	91	10
9.1	120	16



$\hat{y} = m_1 + m_2$ ←

$(f_1 x_1) + (f_2 x_2)$ → Calculate 3 different values for output. (for 3 leaf nodes)

14.5

$$J = \sum_{i=p}^{2} (f_i x_i)$$

So we go closer to reality, i.e. y_i.

So the relation between y_i and $\hat{y_i}$, and this is called loss function, we need to minimize it.

In gradient boosting, there is no fixed loss function,

$L[(y, \hat{y})]$ for gradient boosting.

we can put mse, log, etc. but it should be differentiable

In xgb, there is a to parameter λ.

according to xgb paper,

$$L = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \Omega(f_2 x_i)$$

$\Omega(f_2 x_i)$ ← only on our last one DT

↓ RP ↘

regularization parameter → $\gamma T + \frac{1}{2}\lambda \|w\|^2$ ← weight of node

T ↑ no of leaf nodes in DT

So basically, We need such a values of $w_1, w_2 \ldots w_i$ such that it should minimise the loss function.

derivation

$$L = \sum_{i=1}^{n} L(y_i, \hat{y}) + \Omega(f_k)$$

loss fun^n regularization fun^n

objective fun^n

at as take 2 cases.

case 1) only one model (mean / log odds)

phase 1) $L^{(1)} = \sum_{i=1}^{n} L(y_i, \hat{y}) + 0$ → coz no DT so no consideration

m_1 m_2

case 2)

phase 2 $\boxed{m_1}$ $f_1(x_i)$ $\boxed{m_2}$ $f_2(x_i)$ DT

so, $\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i)$

$$L = \sum_{i=1}^{n} \left[y_i, f_1(x_i) + f_2(x_i) \right] + \underbrace{\Omega(f_2(x_i))}_{DT}$$

case 3)

phase 3 $\boxed{m_1}$ $f_1(x)$ $\boxed{m_2}$ $f_2(x_i)$ DT $\boxed{\quad}$ $f_3(x_i)$ DT

(3)

$$L = \sum_{i=1}^{n} \left(y_i, f_1(x_i) + f_2(x_i) + f_3(x_i) \right) + \Omega\left[f_3(x_i)\right]$$

why only $f_3(x)$ in regularization in f_3? because for f_2 we already handled $f_2 x_i$ regularization in phase 2.

stage t)

$$L^{(t)} = \sum_{i=1}^{n} L\left(y_i, f_1(x_i) + f_2(x_i) + f_3(x_i) \ldots\right) + \Omega(f_t x_i)$$

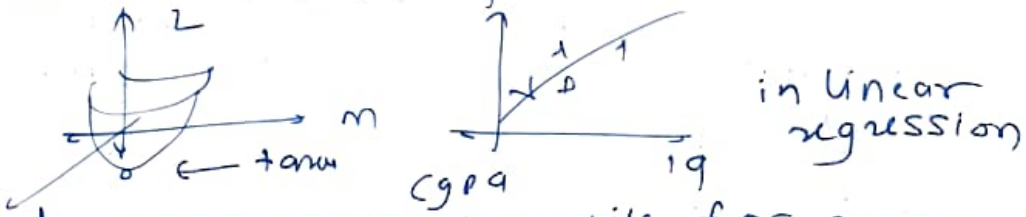
$$L^{(t)} = \sum_{i=1}^{m} L\left(y_i, y_i^{\wedge(t-1)} + f_t(x_i)\right) + \Omega(f_t)$$

at any stage t,

$$L^{(t)} = \sum_{i=1}^{n} L\left(y_i, \hat{y}_i^{<t-1>} + f_t(x_i)\right) + \Omega(f_t(x_i))$$

So, our goal is to find such a value of $f_t(x_i)$ that should minimise all objective function.

But this optimization problem is not as simple as linear regression, like in linear regression



in Decision tree, like for now,

$$\hat{y}_i = \hat{y}_i^{<t-1>} + f_t(x)$$

constant

↓ Decision tree

due to decision tree because it has sudden jumps in values.

So, basically, if we imgine it in a 3D, it will not smooth as linear regression, It can be like this due to DT.

this way, It is not differentiable.

But for us, in optimization, It is our requirement that this fun? should differentiable so we unable to find slope.

but what, if I ... smoothly
like to replace this equation with a simple
equation. so that in this process of approxi
-mation (closer to curve), we make it
smooth and differentiable?
we can do it by taylor series.

Taylor series
I ⇒ powerful mathematical approximation
technique you can give it a complex equation
and then it will convert to simple by a
polynomial equation.
as we add more polynomial terms, x^2, x^3, x^5.
our approximation getting better & accurate
It use derivatives (first, second, third)
to convert a complex function to a simple
polynomial.

To approximate a fun^n around point a,

$$f(x) \simeq f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 \cdots \frac{f^{iv}}{4!}(x-a)^4$$

for example

$f(x) = e^x \quad a = 0$

$f(a) = e^a = 1$

$f'(a)(x-a) = e^a(x-a) = x$

$f''(a)\frac{(x-a)^2}{2!} = e^0\frac{x^2}{2} = \frac{x^2}{2}$

$f^n(a)\frac{(x-a)^n}{n!} = \frac{x^n}{n}$

approx e^x using taylor series

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} \cdots \frac{x^n}{n!}$$

this way we need to apply taylor series
on our function.

$$f(x) \simeq f(a) + f'(a)(x-a) + \frac{f''(a)(x-a)^2}{2!}$$

$$\simeq \left[\sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right] + \Omega(f_t(x_i))$$

$x = \hat{y}_i^{(t-1)} + f_t(x)$

a = changing point.

Basically, we will take a as $\hat{y}_i^{<t+1>}$.

this after will make the change.

$$f(a) = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{<t+1>} + f_t(x_i))$$

$$f(a) = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{<t+1>})$$

$$f'(a)(x-a) = \sum_{i=1}^{n} \frac{\partial L(y_i, \hat{y}_i^{<t+1>})}{\partial \hat{y}_i^{<t+1>}} f_t(x_i)$$

gradient

$$f''(a)(x-a)^2 = \frac{1}{2}\sum_{i=1}^{n} \frac{\partial^2 L(y_i, \hat{y}_i^{<t+1>})}{\partial^2 y_i^{<t-1>2}} f_t^2(x_i)$$

hessian

now, the equation will become,

$$L^{(t)} = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{<t-1>}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) + \Omega(f_t(x_i))$$

$L(y_i, \hat{y}_i^{<t-1>}) \to 0$

$$L^{(t)} = \sum_{i=1}^{n} \left[g_i f(x_i) + \frac{1}{2} h_i f_i^2(x_i)\right] + \Omega(f_t(x_i))$$

$$L^{(t)} = \sum_{i=1}^{n} \left[g_i f(x_i) + \frac{1}{2} h_i f_i^2(x)\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

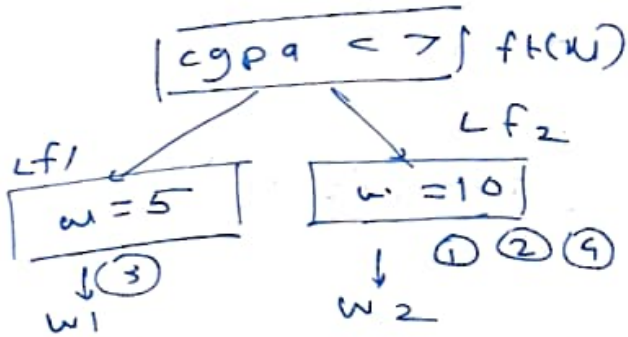
$1 \to n$ (nodes) ; $(1 \to T)$ no of leafs.

basically these two summation are different.

we are going to do

$w_j \to f_t(x_i)$ | $\sum_{i=1}^{n} \to \sum_{j=1}^{T} \sum_{i \in j}$

So, let us take a simple example again.



cgpa	package
7.1	4.5
8.2	6.9
6.5	5.1
9.1	10

so)

$$\left(g_1 f_t(x_1) + \frac{1}{2} h_1 f_t^2(x_1)\right) + \left(g_2 f_t(x_2) + h_2 f_t^2(x_2)\right) + \cdots \left(g_4 f_t(x_4) + \frac{1}{2} h_4 f_t^2(x_4)\right)$$

what being $i \in I_j$ → Instance set of leaf j

for Lf1, we have only one row.
for Lf2 we have 3 rows.

$j = 1$

$$g_3 f_t(x_3) + \frac{1}{2} h_3 f_t^2(x_3) + g_1 f_t(x) + \frac{1}{2} h_1 f_t^2(x_1)$$

same →

$$+ g_2 f_t(x_2) + \frac{1}{2} h_2 f_t^2(x_2)$$
$$+ g_4 f_t(x_4) + \frac{1}{2} h_4 f_t^2(x_3)$$

w_j → output of tree for a particular leaf node

$$L^{(t)} = \sum_{j=1}^{t} \left[\sum_{i \in I_j} g_i w_j + \frac{1}{2} \sum_{i \in I_j} h_i w_j^2 \right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2$$

$$L^{(t)} = \sum_{j=1}^{t} \left[\sum_{i \in I_j} g_i w_j + \frac{1}{2} \sum_{i \in I_j} h_i w_j^2 + \frac{1}{2} \lambda w_j^2 \right] + \gamma T$$

$$L^{(t)} = \sum_{j=1}^{T} \left[\sum_{i \in I_j} g_i w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T$$

now we are going to differentiate it.

So, Now, for the differentiation,

$$L^{(t)} = \sum_{j=1}^{T} \left[\sum_{i \in I_j} g_i w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T$$

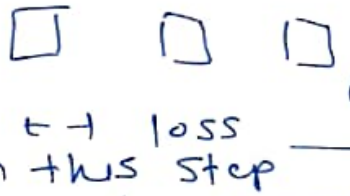
$$\frac{\partial L}{\partial w_j} = \sum_{i \in I_j} g_i + \left(\sum_{i \in I_j} h_i + \lambda \right) w_j = 0$$

$$\boxed{w_j = - \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}}$$

Now, the

$$g = \frac{\partial L(y_i, \hat{y}_i^{<t+1>})}{\partial \hat{y}_i^{<t+1>}}, \quad h = \frac{\partial^2 L(y_i - \hat{y}_i^{<t+1>})}{\partial y^{i<t+1>2}}$$

$t+1$ mean →
on iteration, we say, on model training, we are on $t+1$ th model



$t+1$ loss on this step

In case of regression, mse

$$L = \frac{1}{2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$g_i = \frac{\partial L}{\partial \hat{y}_i} = - \sum_{i=1}^{n} (y_i - \hat{y}_i) = \sum_{i=1}^{n} (\hat{y}_i - y_i)$$

(pred step)

So, basically

$$\therefore \frac{\sum_{i \in I_j} (y_i - \hat{y}_i^{(t+1)})}{\text{sum of residuals}}$$ ← residuals for every point

$$h_i = \frac{\partial L}{\partial \hat{y}_i} = -(\hat{y}_i - y_i)$$

$$\frac{\partial^2 L}{\partial \hat{y}_i^2} = \frac{\partial}{\partial y_i} (\hat{y}_i - y_i) = 1$$

so moving forward

$$= \frac{\sum_{i \in I_j} (y_i - \hat{y}_i^{(t-1)})}{\sum_{i=1} 1 + \lambda}$$

$$= \frac{}{\text{no of residuals in a point}}$$

$$\text{output } w_j = \frac{\text{sum of residuals}}{\text{no of residuals} + \lambda} \quad \text{---(1)}$$

classification problem

loss f_{ui} = log loss

$$w_j = \frac{\text{sum of residuals}}{\sum_{i \in I_j} p_i(1-p_i) + \lambda} \quad \text{---(2)}$$

p_i – predicted probability of previous iteration

so input w_j in loss function, Basically calculate minimum loss on minimum w_j.

$$L^{(t)}(q) = \sum_{j=1}^{T} \left[\left(\sum_{i=1}^{} g_i \right) \left(\frac{-\sum_{i=1} g_i}{\sum_{i=1} h_i + \lambda} + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) \left(\frac{\left(\sum_{i \in I_j} g_i \right)^2}{\left(\sum_{i \in I_j} h_i + \lambda \right)^2} \right) \right] + \gamma T \right.$$

$$L^{(t)}(q) = \sum_{j=1}^{T} \left[\frac{-\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} + \frac{1}{2} \frac{\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} \right] + \lambda T$$

$$= \sum_{j=1}^{T} \left[-\frac{1}{2} \frac{\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} \right] + \lambda T$$

$$= -\frac{1}{2} \sum_{j=1}^{T} \frac{\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T$$

SSC

minimum loss for last model added

So, basically, This equation can be used as a scoring function, basically to measure the quality of last one tree we made.

$$\text{similarity score} = -\frac{1}{2} \frac{\sum_{j=1}^{+} \left(\sum_{i \in I_j} g_i \right)^2}{\sum_{j \in I_j} h_i + \lambda} + \gamma T$$

and this is exactly.

regression

$$SIC = \frac{\left(\sum \text{residuals} \right)^2}{N + \lambda}$$

classification

$$SIC = \frac{\left(\sum \text{residuals} \right)^2}{\sum (P \cdot (1-P)) + \lambda}$$

How to build the tree using similarity score?

Let us say, we are on $t+1$ th step in models.

← Decision tree

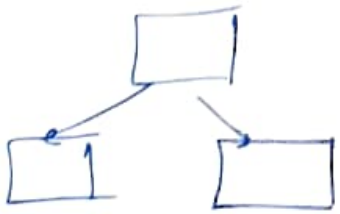
for a leaf node

$$L^{(1)}(q_P) = -\frac{1}{2} \frac{\left(\sum_{i \in I_P} g_i \right)^2}{\left(\sum_{i \in I_P} h_i + \lambda \right)} + \gamma T$$

(parent)

for one leaf node.

Then we will do split for the leaf



structure

we splitted it and made new nodes, because we wish to reduce the loss too more.

Then we will now calculate overall loss for tree structure

$$L^{(t)}(q_c) = \frac{1}{2}\left\{\left[\frac{\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i+\lambda}\right] + \left[\frac{\sum_{i\in I_R} (g_i)^2}{\sum_{i\in I_R} h_i+\lambda}\right]\right\}$$

(child)

$$+ 2\lambda \quad \because (T=2)$$

so now I have both parent & child loss.

I wish to know how much loss reduced after the split.

$$L^{(t)}(q_P) - L^{(t)}(q_c) = \left\{\left[\frac{-\frac{1}{2}\left(\sum_{i\in I_P} g_i\right)^2}{\sum_{i\in I_P} h_i+\lambda} + \gamma\right]\right.$$

$$\left. - \left(\frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i+\lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i+\lambda}\right] + 2\gamma\right)\right\}$$

$$= \left[\frac{-\frac{1}{2}\left(\sum_{i\in I_P} g_i\right)^2}{\sum_{i\in I_P} h_i+\lambda} + \frac{\frac{1}{2}\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i+\lambda} + \frac{\frac{1}{2}\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_R} h_i+\lambda}\right]$$

$$- \gamma$$

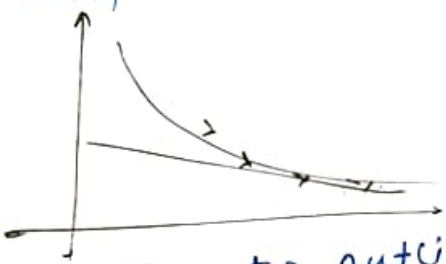
$$= \frac{1}{2}\left[\frac{\left(\sum_{i \in I_L} g_i\right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i\right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I_P} g_i\right)^2}{\sum_{i \in I_P} h_i + \lambda}\right] - \gamma$$

This is exactly formula for when we did $(SSL + SSR) - SSP$ to calculate loss.

DBScan (Density Based Spatial Clustering of Application with noise)

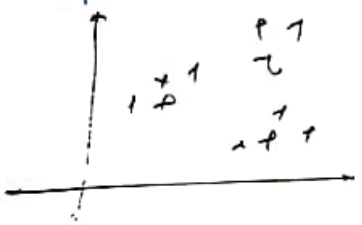
Why dbscan? Why to change some flaws for kmeans

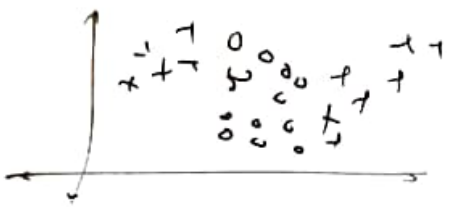
1) need to tell how many clusters, should be done. So we use elbow method. but many time it is difficult to tell.



2) very sensitive to outliers.

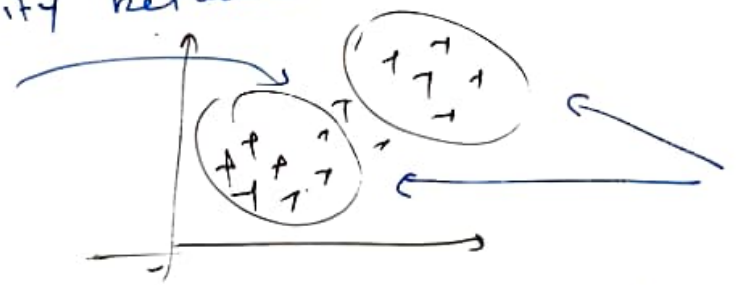
3) kmeans try where can't form spherical datapoints.





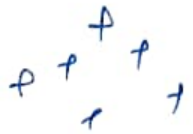
Density clustering

We try to cluster the data on the basis of density between two datapoints.



basically, DBscan see data, separate them out form sparse region, and separate dense region.

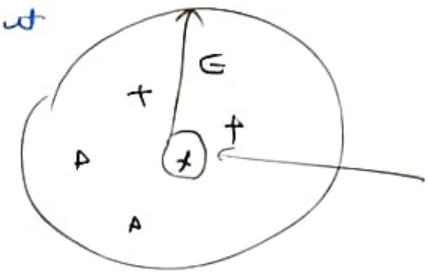
basically let's say,



we need to define a threshold, and It will work radious for the targeted point. and it some k number of ~~clusters~~ points come under the radious of that spcific threshold, we need to consider it dense if they are less, then sparse

some concepts

1) Core point

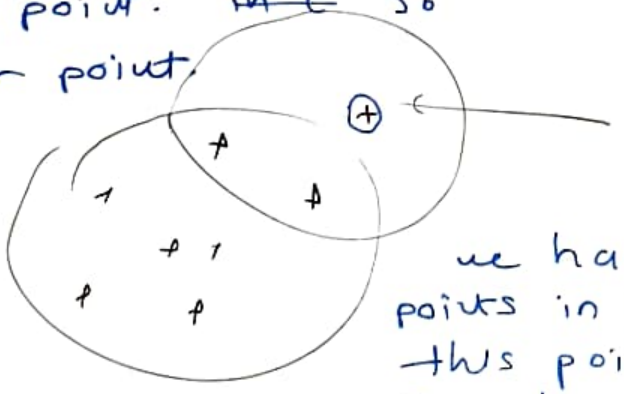


core points

MIN point = 4

in ϵ neighborhood, we have greater or equal to datapoints, that is called as core point. ~~IN ϵ~~ so

2) Border point.

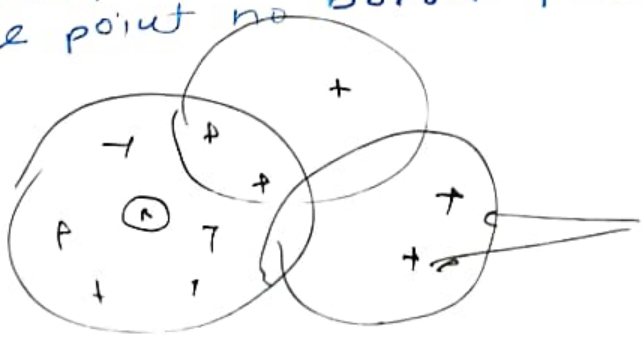


bndr point.

we have less no of points in neighborhood of this point. so, this is a bordr point. BUT it have also a core point in ϵ region.

3) noise point.

NO core point no bordr point



noise points

density connected points.

if A and B are two points, and if they are density connected, then we put both them in a single cluster.

$d < \epsilon$

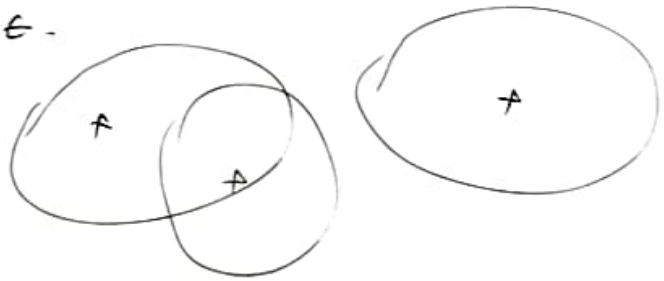
core points.

basically under the influence of core points. Basically connected indirectly.

this fails when the main cause of failure can be

no core point.

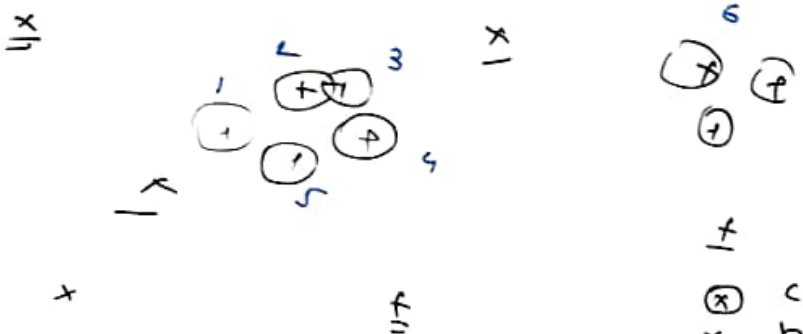
if distance between two core points is greater than ϵ.



Algorithm

step 0) define ϵ and no points.

step 1) classify core, border, noise



core point

border point

noise point

add all points that are unclustered and density connected to the current point to this cluster.

Let start with ① and check conditions are points density connected?

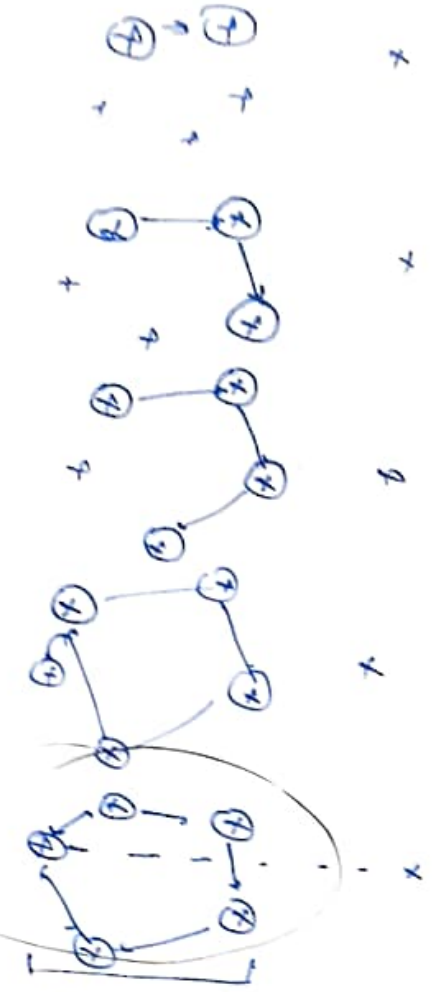
① → ② ✓

④ → ⑨ ✓

① → ④ ✓

④ → ⑦ ✓

① → ⑥ ×



a cluster formed

do now for all unclustered points.

Step 3) for each unstructured border point, assign it to the nearest cluster point.

Step 4) leave all noise points as it is.

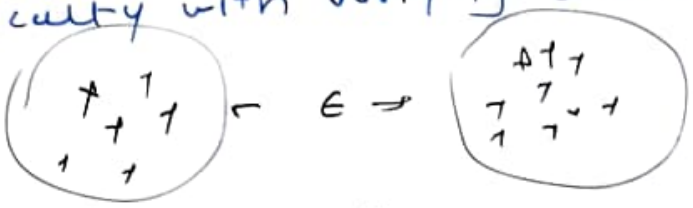
* Advantages

1) Robust to outliers
2) no need to specify clusters.
3) can find arbitary shaped clusters
4) 2 hyp to tune (m_{in}, ϵ).

disadvantage

1) sensitive to hyperparameters

2) difficulty with varying density clusters



3) cannot do prediction

compiled playlist!

Credit goes to CampusX!