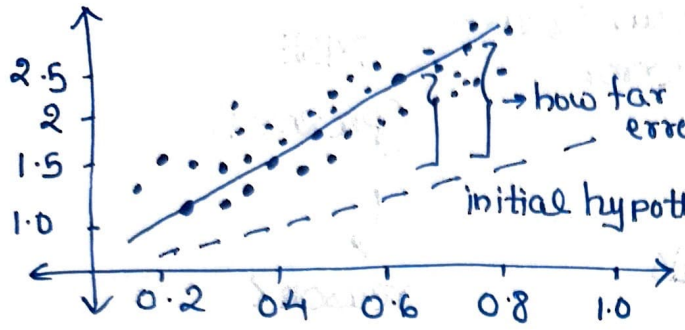
*Parametric Learning, Linear Regression



$y = ax + b$ } parametric learning

modelling { hypothesis
models → specific a & b

at one point we find the best possible hypothesis

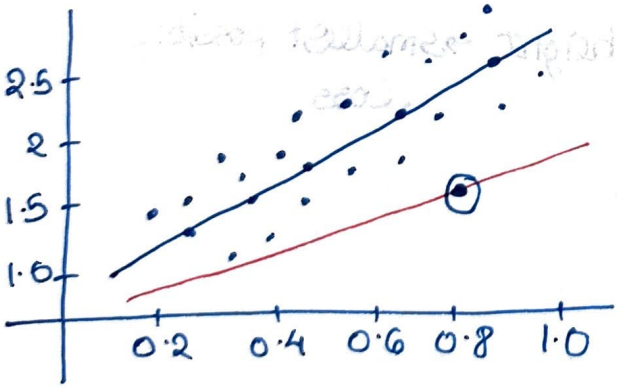
(x, y) ↳ numerical label

based on errors we correct it. ↳ multiple iteration → practice makes a man perfect.

linear regression

$y = f(x)$ → possible hypothesis bigger not just one

*Loss function for Linear Regression:-



$y = ax + b$

way to quantify how good a model is?

$\hat{y} = a(0.8) + b$ — we maybe know the correct value is $y = 26$

$x_i : \hat{y}_i = a x_i + b$ ↓ predicted

y_L ↳ correct label

Error in Correct & predicted

$$e_i = |\hat{y}_i - y_L| = |axi + b - y_L|$$

error depends on a & b

Loss fn $L_{MAE}(a, b)$

$$\sum_{i=1}^{n} e_i = \frac{e_1 + \dots + e_n}{n}$$

Mean absolute error

Other type of loss function

$$L_{MSE}(a,b) = \frac{1}{n}\sum_{i=1}^{n} e_i^2$$ | mean squared error

MSE (preferred)

↓

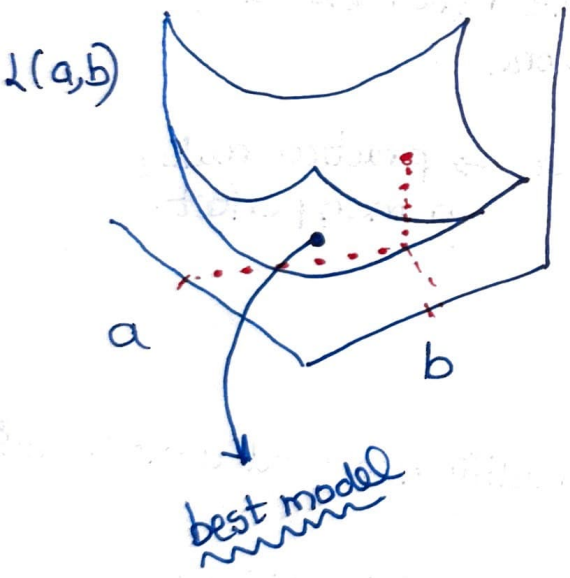
reason?

↓

mathematical mainly

Which one is better?

We can maybe make this hyperparameter.

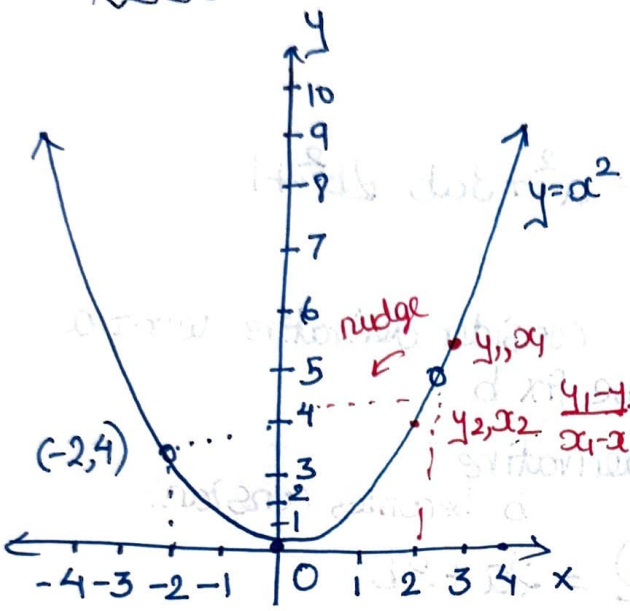


Landscape → convex shape

Smoothly more around.

Smallest possible height → Smallest possible loss

* Gradient Descent (one variable)



$x=2, \quad y=x^2=4$

derivative

$y' = f'(x) = 2x \quad \Big| \quad f'(2)=2\cdot 2=4$
$f'(2)=2.$

derivative : speed (how fast the it moves)

Sign → shows speed / things

if $x>0 \rightarrow +1$

$x=0 \rightarrow 0$

$x<0 \rightarrow -1$

↳ value of f(x) like f(2)

Sign $(f'(2)) = +$

its increasing $f'(2) = 2\cdot 2 = 2$

$f'(3) = 2\cdot 3 = 6$

$f'(4) = 2\cdot 4 = 8$

$f'(-2) = 2(-2) = -4$

Sign $(f'(-2)) = -$

↳ value is 4 but in negative

nudge — derivative — step 0.1

$$X := X - \text{step\_size} \cdot f'(x)$$

new X — old X

$f'(x) = 2x$

$X := 2 - 0.1(2\cdot 2)$

$X := 2 - 0.1(4) = 1.6$

descented towards minimum

→ This is called gradient descent

$X := 1.6 - 0.1(2\times 1.6)$ → descent towards minimum.

$X = 1.6 - 0.32$

$X = 1.28$

we can do iterations and go towards minimum.

We need to be careful

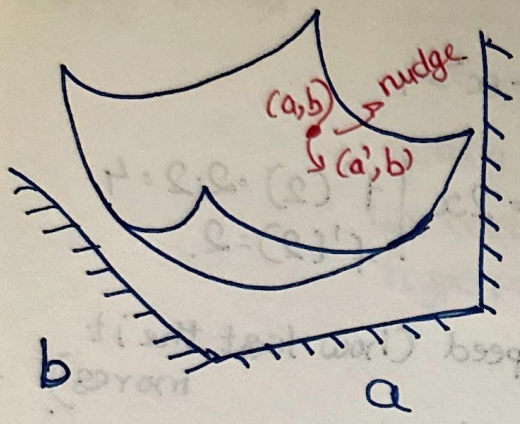
if we take step size = 2

$X = 2 - (2\cdot 4) = -6$

if we overnudge the we overshoot, correct direction but we overshot.

* Gradient Descent (Multiple Variables)

$$L(a,b) = a^2 + 3ab + 2b^2 + 1$$



we can consider derivative w.r.t a where we fix b

Partial derivative

b becomes constant

$$\frac{dL(a,b)}{da} = 2a + 3b$$

$$\frac{dL(a,b)}{db} = 3a + 2b(2)$$

$$= 3a + 4b$$

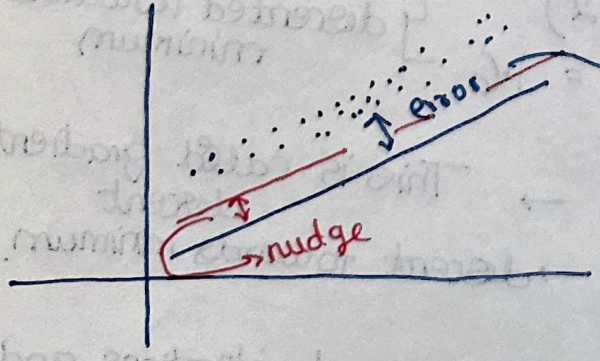
Now we gotta nudge it

new a old a

$$a := a - \text{step-size} \cdot \frac{dL(a,b)}{da}$$ → how fast a mores

$$b := b - \text{step-size} \cdot \frac{dL(a,b)}{db}$$ → we have one for b too

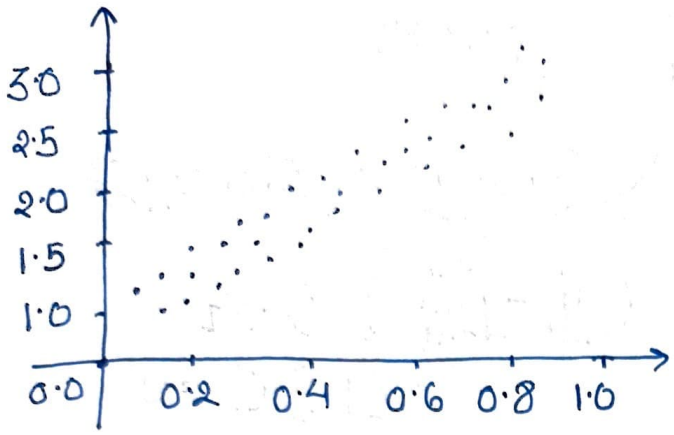
we do two updates.



So we nudge it. and it mores closer to our set of points.

Gradient descent fits it better to the data.

* Gradient Descent for mSE



$y = ax + b \rightarrow$ intercept

↓ Slope coefficient.

$$L(a,b) = \sum_{i=1}^{n} e_i^2$$

→ predicted

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

→ correct y_i → labelled

$$= \sum_{i=1}^{n} (ax_i + b - y_i)^2$$

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

$$\frac{1}{n}\frac{dL}{da} = \frac{1}{n}\frac{de_i^2}{da} + \cdots\cdots + \frac{de_n^2}{da} = \frac{2}{n}(\hat{y}_1 - y_1)^2 \cdot x_1 + \cdots\cdots + 2(\hat{y}_n - y_n)x_n$$

Gradient Descent for mSE

$$\frac{dL}{da} = \frac{2}{n}\sum_{i=1}^{n} (\hat{y}_i - y_i) \cdot x_i$$

$$\frac{dL}{db} = \frac{2}{n}\sum_{i=1}^{n} (\hat{y} - y_i) \cdot 1$$

$$a = a - \text{step\_size}\frac{dL}{da} \qquad b = b - \text{step\_size}\frac{dL}{db}$$

Example:-

$$X = \begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix} \quad y = \begin{pmatrix} 1 \\ 2 \\ 5 \end{pmatrix}$$

$y = x + 2$

~~$(\hat{y}_1 - y_1) = (ax + b - y_1)$~~

$a = 1, b = 2$

no of points $(n) = 3$

$(\hat{y}_1 - y_1) x_1 = (1 - 1) \cdot x_1$

$= (1-1)(1) = 0$

$(\hat{y}_2 - y_2) x_2 = (-1 - 2)(-1)$

$= 3$

$(\hat{y}_3 - y_3) x_3 = (2 - 5)(2)$

$= -3(2) = -6$

$$\frac{dL}{da} = \frac{2}{n} \sum_{j=1}^{n} (\hat{y}_j - y_j) \cdot x_j$$

$$\frac{dL}{da} = \frac{2}{3} (0 + 3 + (-6))$$

$$\frac{dL}{da} = \frac{2}{3}(-3) = -2$$

$$\frac{dL}{db} = \ldots\ldots$$

$a :=$ stepsize (rho).

$$a := a - \rho \frac{dL}{da}$$

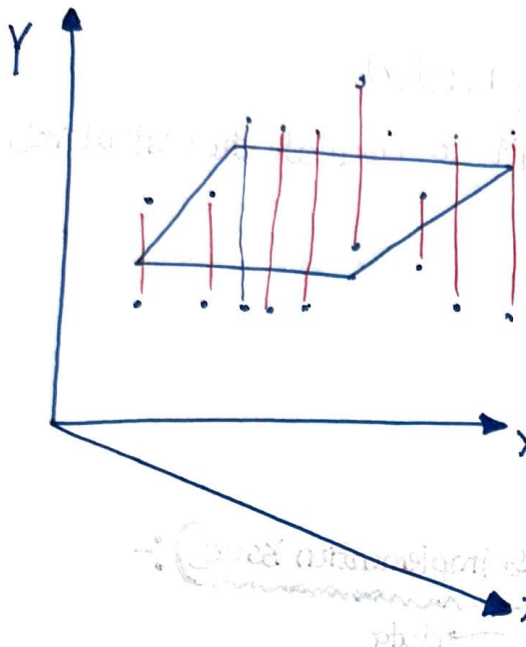
$$a = a - \rho(-2)$$

$$a = a - \rho(-2)$$

$$a = 1 - \rho(-2)$$

whatever step size (ρ) is we get new a.

* Generalization in higher Dimensions



dimension

$d=2$

$y = \underline{w_1} x_1 + \underline{w_2} x_2 + b$

coeffs — intercept

$\times$ $n \times d$

hyperplane (dimension 3 or higher)

$y = w_1 x_1 + w_2 x_2 + \dots + w_d x_d + b$

$y = w_1 x_1 + w_2 x_2 + \dots + w_d x_d + b$

$X = [x_1 \dots x_d]$

$w = [w_1 \dots w_d]$

$y = X \cdot w^T + b$

dot product. matrix

$$[x_1 \dots x_d] \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}$$

Update Rules?

update coeff w

for Ax

$$w_i^{new} = w_i^{old} - \rho \sum_{j=1}^{n} (\hat{y}_j - y_j) x_{j,i}$$

$ax+b$

i: coordinate column i

independently nudge

bias

$$b := b - \rho \sum_{j=1}^{n} (\hat{y}_j - y_j) \cdot 1$$

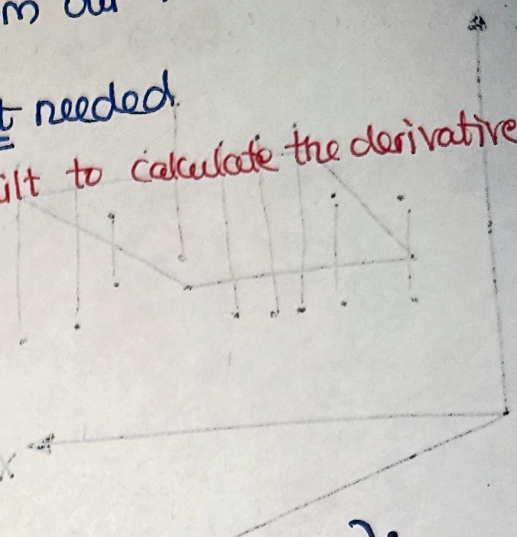
$+b$

ρ: learning rate (step size)

ρ → every step we learn from our mistakes

Gradient Calculation is not needed

There are programs built to calculate the derivatives



* Gradient Descent: The algorithm (efficiency & implementation issues):-

function w = Linear Regression-fit (X, y) → labels; data

Initialize randomly a d-dimensional vector w and a scalar b

for $j = 1$ to n {epochs:} — every iteration all data points

$$b = b - \rho \sum_{j=1}^{n} (b + X_j w^T) - y_i$$

predicted label; intercept

for $i = 1$ to d

$$w_i = w_i - \rho \sum_{j=1}^{n} (b + x_j w^T - y_j) \cdot x_{j,i}$$

} one update per loop; sum of individual error

return w, b

function w = Linear Regression-Predict (w, b, x) → unseen point

return $xw^T + b$

Epoch:- entire dataset →

① initialization

random state

issue → reproducibility (get back to same values or reproduce a good fit)

pseudo random number

Hyperparameter ② Learning Rate ρ can change by iteration

③ In every epoch we consider entire dataset, individual error calculation, a lot work $O(n \cdot d)$

Intuitively doesn't make sense, do we need this much?

Scaling features for faster learning:

1) For each attribute i, calculate average

$$\mu_i = \sum_{j=1}^{n} x_{j,i}/n$$

2) for each attribute, calculate Std Der

$$\sigma_i = \sqrt{\sum_{j=1}^{n} (x_{j,i} - \mu_i)^2/n}$$

3) For each point j and attribut i compute a new attribute.

$$x'_{j,i} = \frac{x_{j,i} - \mu_i}{\sigma_i}$$

(μ_i → mean; σ_i → Std der)

$\text{mean}(x'_i) = 0$

$\text{Std}(x_i) = 1$

$L(w_1, w_2, b)$: Speed towards bottom / convergence

So we do scaling of features.

$X_{n \times d}$ = (| | | | |) (i)

in every column

X' = [| | |]

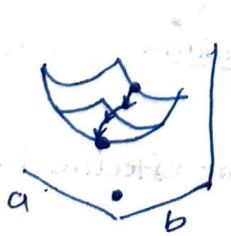
for this column.

→ better landscape of the ~~radius~~ error

easier for gradient descent.

If we don't have info a column is not that important then scale it

make it more fair.



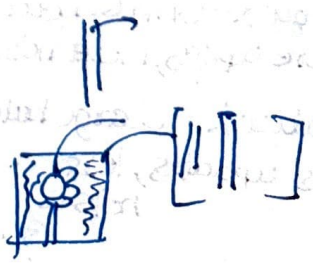
$L(a,b)$

err

$\frac{dL}{da}$ $\frac{dL}{db}$ ( )

$X w^T + b$

(,)



Evaluating a regression model

$y = wx + b$

$$L(w,b) = \sum_{i=1}^{n} e_i^2$$

} dependent on scale of data

X, y

$X' = 2x$, $y' = 2y$ blow all points by factor of 2

$L_{new}(w,b) = 4L(w,b)$

---

* MSE

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

Baseline model : $\bar{y} = mean(y_i)$

average of existing points

no new calculations

X, y

$$MSE_{base} = \sum_{i=1}^{n} (y_i - mean(y_i))^2$$

Coefficient of Determination : $R^2 = 1 - \frac{MSE_{model}}{MSE_{base}}$ (new model) / (already existing) (mean of values)

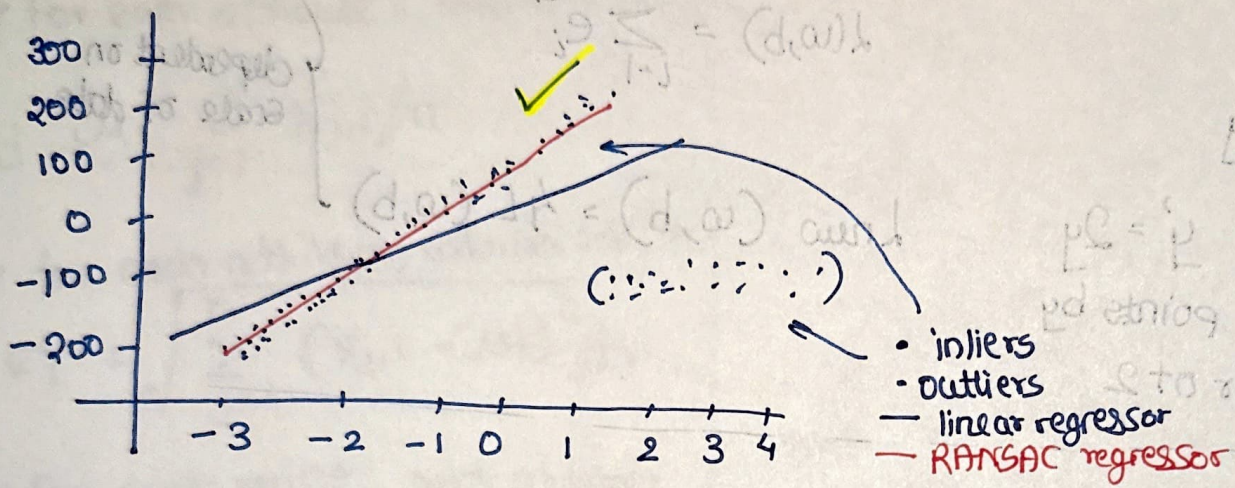
$R^2 = 1$ } Consider $MSE_{model} = 0$ Perfect case.

$R^2 = 0$ } if model sucks model = base

mse model = MSE_{base} worst-case

Dealing with outliers RANSAC

* random sample consensus



Hyperparameters

n' → number of sampled points

Θ → inlier threshold

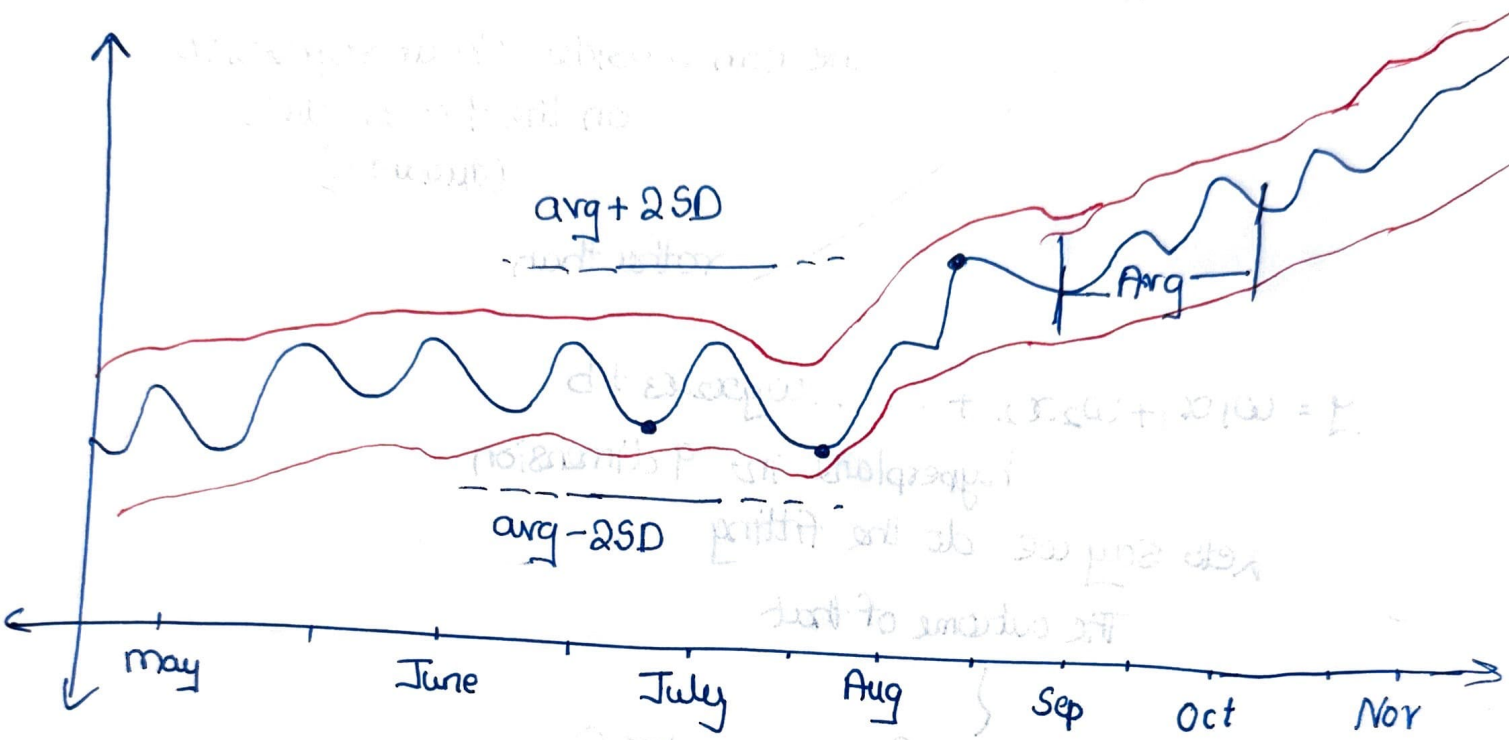
n'' → Sufficient no of inliers

- Sample randomly n' points (X', y') from dataset (X, y)
- fit a model f to (X', y')
- Evaluate all points in (X, y) with f and keep only those that give an loss less than a threshold Θ. These are inliers
- If number of inliers exceed than n'' → model is good.
- If model is good then return new model fit on the inliers.

→ Start with a model
→ figure out threshold
→ if there are enough below threshold
→ add to inliers
→ fit this new model.

* Feature Engineering

SSP 500



$X = [X_1 \ldots \ldots X_{30}]$

mean (X)

Std dev (X)

2 x Std dev.

for every point plot $\xrightarrow{2SD}$ + avg

& avg - 2SD

2 Std dev(X) $\sqrt{\frac{\sum (x_2 - x_{mean})^2}{n}}$

at various point when it hits low

it seems to have tendency to bounce up as per SD

Polynomial Regression :-

$X = [X_1, X_2, X_3] \longrightarrow [X_1, X_2, X_3, X_1^2, X_2^2, X_3^2, X_1X_2, X_2X_3, X_1X_3]$

raw features — quadratic

we can consider linear regression on the 9 coeff class (quadratic) rather than

$y = w_1x_1 + w_2x_2 + \ldots . w_9x_2x_3 + b$

hyperplane in 9 dimension

Lets say we do the fitting

The outcome of that

{

make a few parameters 0

$y = X_1^2 + X_2^2$ (other coeff too small / 0)

but this eqn is not linear → its quadratic

3D (raw space)

not linear

not hyperplan }

its just a surface.

We can also define cubic

Possible Applications :- amount of COVID cases

$X_1, X_2 \ldots \ldots X_{20}$ (following a increase) → $\log X_1, \log X_2 \ldots \log X_n$

log regression

Covid Data $\rho^2 : 0.99$ (not necessarily true)

actual for COVID from [redacted] sources

So analysts thought it was made up.