SL 1 Day 1 Sum up

What is machine learning?

→ is about How machine learning Certain things.

→ Thru. phase (two phases)

→ i.e. patterns (e.g. ↑ no. of cus leads to ↑ sales)

1 Training phase : learns patterns

2 Testing phase : replicate the learned patterns (i.e. Predict)

⇓

Then that machine (that learned some pattern) is called model. *Imp

Model

→ what?

1 learn pattern [train]

2 replicate patterns [test]
(optional : e.g. USL)

↳ check the performance (i.e. % of patterns it replicate successfully).

⇓

error = 1 − performance.

→ How? (learns patterns)

large amount of data *Imp.

⇓ need to

learn patterns.

Data

→ X, y

→ X : Independent variable / feature / cause.

→ y : Target variable / dependent variable / outcome / effect.

↳ y : is pain area of the client / problem of client.

[is our objective ; get from directly from client]

↳ i.e. requirement.

↳ y / target variable / pain area

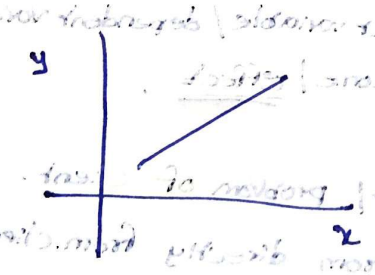
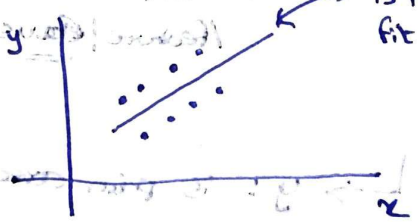
two types

1. continous → model used in such case is "Regression Model"
2. Discrete → model used in such case is "Classification Model"

↳ Model

- based on y (nature of y)
  - Regression model
  - Classification Model
- based on y (given / not) in training
  - Supervised learning Model
    - * x, y given.
    - * learn pattern & predict
  - unsupervised learning model
    - * x given; y not given
    - * learn pattern & no prediction

Estimation vs prediction

estimation	prediction
$y = mx + c$	$\hat{y} = mx + c + e$
$y = b_0 + b_1 x$	$\hat{y} = b_0 + b_1 x + e$ (e for error)
	 is the best fit line. ↳ i.e. linear regression line / prediction line.
* It is not prediction.	* It is not estimation.

Error

- error = Actual − predicted
- So, Error (e) = $\sum_{i=1}^{\text{no. of observation/row}} (\text{Actual} - \text{predicted}) = \sum_{i=1}^{n} (y - \hat{y})$
  - in $\hat{y} = b_0 + b_1 x + e$.
    - but in reality, it is Sum of Squared error (SSE)
      - i.e. $\sum_{i=1}^{n} (y_i - \hat{y})^2$
- Error/SSE can be -ve, 0, +ve.

How to find the best fit line? (Build the model)

- using: minimum SSE (min. Sum of Squared error)
  - i.e. the line with min. SSE.

⇓

method used (Build the model)

1. Statistical way
   - use OLS (Ordinary least squared)
   - $b_0 = \bar{y} - b_1 \bar{x}$
   - $b_1 = \frac{cov(x,y)}{cov(x)} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$

   Purpose/when to use:
   - ~~To test~~
   - 1 To find significant variable ~~imp~~
   - 2 To test ~~signif~~ linear regression assumptions (~~90~~ % of times not met)

2. Machine learning way
   - use Gradient descent.

   Purpose/when to use:
   - To know performance of the model.
   - (Most widely used)

Test performance ⟹ Main use by R^2.

→ **baseline / benchMark**: Average model.
↳ the model with no independent variables
↳ so it is the worst model you can have ever.

→ Using Metric
↳ 3 variations
→ ¹ Sum of squared errors (SSE)
→ ² Sum of squared Regression (SSR)
→ ³ Sum of squared total (SST)

↳ Now, we use Stats way / formulaic already derived:

$b_0 = \bar{y} - b_1 \bar{x}$ ($\bar{x}, \bar{y}$ → are means)

$b_1 = \frac{\text{cov}(x,y)}{\text{var}(y)}$

@ Average model ⟶ $b_1 = 0$ ⟹

$\hat{y} = b_0 + b_1 x$

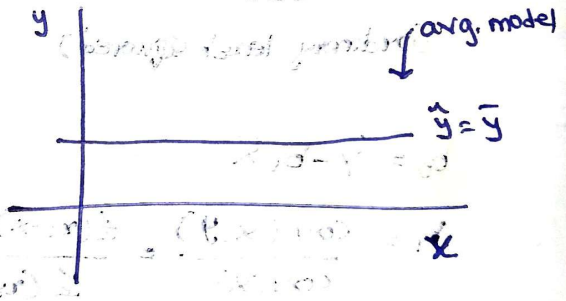
$\hat{y} = b_0 + (0)x$

$\hat{y} = b_0$

$\hat{y} = \bar{y} - b_1 \bar{x}$

$\hat{y} = \bar{y} - (0)\bar{x}$

$\boxed{\hat{y} = \bar{y}}$



v.v.v. ↳ based on $y, \hat{y}, \bar{y}$.

SSE = $\sum_{i=1}^{\text{no. of observations}} (\text{Actual} - \text{predicted})^2$ [Model fails to explain] due to randomness, noise, influence beyond model scope.

SSR = $\sum_{i=1}^{\text{no. of observation}} (\text{predicted} - \text{Avg. model})^2$ [how the independent variables contribute to Predicted dependent variable]

SST = $\sum_{i=1}^{\text{no. of observation}} (\text{Actual} - \text{Avg. Model})^2$ [variation of y]

↳ $SST = SSR + SSE$ — total variance → explainable variance by model (known factor) + unexplainable variance (unknown factor)

$R^2 = \frac{SSR}{SST}$ (* imp)

$R^2 = 1 - \frac{SSE}{SST}$

↳ of total variation explainable by explainable variation caused by regression.

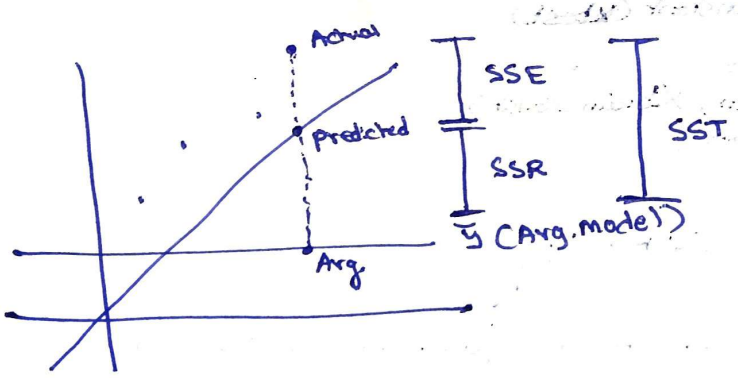
R^2 → square of Pearson Coeff. (r)
↳ i.e. using $\hat{y}$ and y.

↳ $R^2 \Rightarrow [0,1]$ as $-1 \le r \le 1$

① $R^2 = 0$ when $SSR = 0$ Then → $SST = SSE$.
⇓
worst model [also note: $\hat{y} = \bar{y}$ (Avg. model) as predicted = Actual for $SST = SSE$]

② $R^2 = 1$ when $SSR = SST$ Then → $SSE = 0$
⇓
'best' model
↳ is that so? i.e. $SSE = 0$
↳ impossible in reality as error $\neq 0$.
[comes concept of overfit, Goodfit, underfit]
⇓
* use R^2_{train}, R^2_{test} & comparision blw them.
* other way also exists using RMSE.

One of assumption of linear regression → Strong correlation blw X (any feature) & Y (target) (else bringdown the model performance)
⇓
inline with $r^2 = 0$ as worst model and $r^2 = 1$ as 'best' model (actually overfit)



Recap:

R^2_{train}

- if < 0.5 ⇒ ~~worst model~~ Underfit model (not desired)
- if ≥ 0.99 ⇒ Overfit model (not desired) (way 1)
- if b/w 0.5 and 0.99

Also another way (way 2)

① R^2_{train} b/w 0.5 & 0.99 [Perform well in train]

② $R^2_{test} < 0.5$ [Perform poorly in test]

↳ esp. useful in case of eg.

$R^2_{train} = 0.51$

$R^2_{test} = 0.49$

use 0.5, 0.99 & also definite

↓

& $R^2_{train} \times 10\% \pm R^2_{train}$

↳ is range of R^2_{test}

↓ then

Goodfit model

SL-1 Day 2 Sum up

Model building - Six Step Process

1. Split X (independent variable) and y (dependent variable)
   ↓ then
2. Split each X and y as train and test
   ↓ then
3. Initialize Model / Algorithm (e.g. Lr, Randomforest, etc)
   ↓ then
4. Train Model / learn patterns
   ↓ then
5. Test Model / predict (replicate patterns learned)
   ↓ then
6. Check performance [use RMSE and/or MAPE] (Actual - predicted)
   → only in case of regression

Performance checking

1. Error = (Actual - predicted)
2. Squared Error = $(\text{Actual} - \text{predicted})^2$
3. Mean squared Error = $\frac{1}{n} \cdot \left(\sum_{i=1}^{n} (\text{Actual} - \text{predicted})^2\right)$ [n → no. of observations]
4. Root mean squared error [RMSE] = $\sqrt{\frac{1}{n}\left(\sum_{i=1}^{n} (\text{Actual} - \text{predicted})^2\right)}$

Note:

R^2 → range [0,1] => so gives Strength (i.e. in bbb range [0,1]) useful in saying good fit / overfit / underfit.

RMSE → range [0,∞) => so not gives strength (as any arbitary value), so, hardly used in saying goodfit / overfit / underfit.

R^2 & RMSE =>

RMSE $\xrightarrow{\text{tend to}}$ 0 $\Rightarrow$ error $\xrightarrow{\text{tend to}}$ 0 $\Rightarrow$ R^2 $\xrightarrow{\text{tends to}}$ 1

$$R^2 = \frac{SSR}{SST}$$

Goodfit vs overfit vs underfit

R^2_{train} & R^2_{test}

R^2 value	fit type of model	additional criteria /Guideline (not rule)
$R^2_{train} = 1$	Overfit	—
$R^2_{train} = 0$	Underfit	—
$R^2_{train} \in [0, 0.5]$	Underfit	—
$R^2_{train} \in [0.5, 0.99]$	Goodfit	Difference b/w R^2 value of train prediction and test prediction is ~~test~~ not more than 10%.

Model fit type	Definition
Overfit	* Model perform very well in training data but very poor in testing data. * 2 conditions → R^2_{train} close to 1. → Model perform very well in training data but very poor in testing data.
Goodfit	* Model perform very well in training data and also perform very well in testing data.
Underfit	* Model perform very poor in training data and also perform very poor in testing data.

R^2_{train} condition check → if only for $R^2_{train} \in [0.5, 0.99]$ → R^2_{test} condition of 10% check

Assumptions of linear Regression

Types of Assumptions

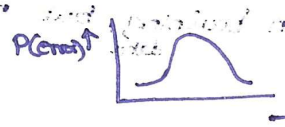
- based on Errors
  1. Normality of Errors
  2. Homoscedasticity
  3. auto correlation
- based on X and y
  4. Linearity
  5. Multi collinearity

NHA LM

	X_1	X_2	....	y_{actual} (A)	y_{pred} (CP)	error (A-P)
R_1						e_1
R_2						e_2
R_3						e_3
⋮						⋮

① Normality of Errors

↳ i.e. distribution Errors Should follow Normal distribution.



↳ Test:

Jarque-bera (preferred)
Shapiro (only if no. of datapoints < 5000)
Q-Q plot
Anderson-darling

② Homoscedasticity

↳ i.e. Equal variance for errors. [i.e. no pattern forming]

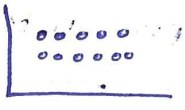


fig: constant variance

↳ Test:

Brusche pagan test
Levene test

* Heteroscedasticity

↳ i.e. unequal variance for errors. [i.e. pattern forming]

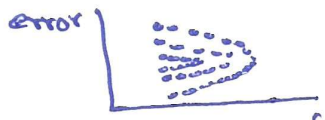


Fig. unequal variance of convergent pattern

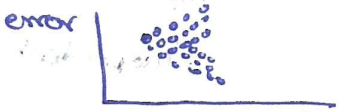


Fig. unequal variance of divergent pattern

③ Auto correlation

↳ i.e. dependency / correlation b/w errors.
[i.e. follows certain patterns]

e_1 100
e_2 50
e_3 0
e_4 -50

→ so, pattern $e_i = e_{(i-1)} - 50$

↳ Test:
~~Rainbow test~~ Durbin, watson

Key idea for assumptions based on error

How Model learn to predict?

↳ based on pattern in data

↓↓

so, then If pattern in error, it learns it

↓↓ lead to

reduce error, (i.e. most of time Actual = predicted)

↓↓ case of

Overfit [not learn pattern, instead just by hards]

So, pattern in error(s) is not preferred or must be avoided

④ Linearity

↳ i.e. High relationship / correlation / linearly related b/w X_1 (i.e. feature 1 (Foreg.)) and Y.
(X_1: Independent variable; Y: dependent variable)

↳ Test:
Rainbow test

⑤ Multicollinearity

↳ i.e. High corerelated / dependency / related b/w Independent variable(s).

e.g.

$X_3 = X_1 + X_2$

$X_3 = X_4 - X_5$

$X_3 = X_6 / X_7$

↳ Test:

COND NO. (condition number)

VIF test

COND NO.	
>1000	High Multicollinearity
b/w 1000 and 100	Moderate Multicollinearity
<100	Low multicollinearity

→ for this only using (N-1) dummies encoding

* If High Multicollinearity (or in general collinearity), then β value of the dependent variable (in form of 'independent variable') [e.g. X_3] is very high than of the actual independent variable [e.g. X_1].

⇓

it means X_3 is given more importance (as contributor to predict) but it is not the actual case, X_1 & X_2 need to be in reality more importance (instead as least importance by the model built with Multicollinearity)

extra points

* $R^2_{test} \geq R^2_{train}$ (at least possible cases by chance)

↳ when the test data pattern is the most learned by the model in training phase from train data.

* 1st check Good fit / not → then only 2nd check Model performance

* OLS given linear regression with min. SSE can be worst (as it's just min SSE but not may good model), so compare with Avg. model using R^2.

* Null hypothesis (usual thing / Status quo) is Mutually Exclusive to Alternate hypothesis.

SL-1 Day 3 Sum UP

1. Durbin watson
2. VIF test,
3. Q-Q plot
4. Error
5. Bias error
6. Variance error
7. error≠0 ?
8. BE & VE
9. BE, VE, overfit, underfit.
10. Regulizn.
11. Types of Reg.

Durbin watson

- test for autocorrelation.
- Range [0,4]
- If in b/w 1.5 and 2.5 ⇒ Auto correlation exists.

VIF test

1. Multi Collinearity

- If exist in dataset ⇒ check with help of COND. NO.
- If exists in which variables / features ⇒ check with help of VIF test.

VIF	Status	Next step
<5	low multicollinearity	Keep & good to go
5-10	Moderate Multicollinearity	remove that variable/ feature & again check VIF test.
>10	Severe Multicollinearity	

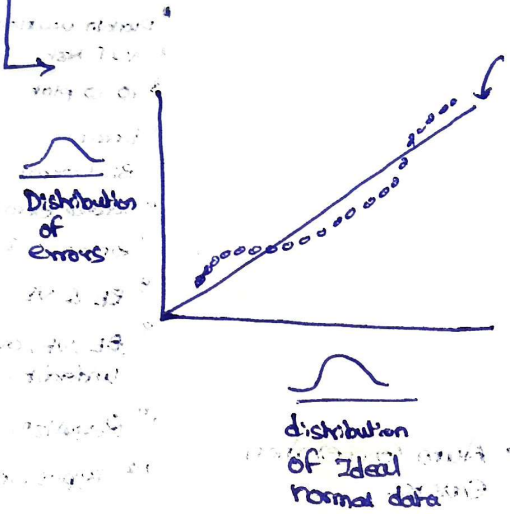
- $VIF = \frac{1}{1-R^2}$

High Multicollinearity $\xrightarrow{\text{then}}$ $R^2 \xrightarrow{\text{tends to}} 1$ $\xrightarrow{\text{then}}$ VIF $\xrightarrow{\text{tends to}}$ inf.

- VIF is "Variation Inflation Factor"
  - ↳ by it use, we can reduce the Inflated βs.
- only for numeric Columns.

Q-Q plot

→ Tests the normality of errors/residuals.



This line represents both are same. And any deviation from it indicates the dist. of errors is not normally dist.

→ Quantile - Quantile plot.

Errors

→ is (Actual - predicted)

→ 3 types (Mainly)

1. Bias Errors
2. Variance Errors
3. Random Errors

1 & 2 → Controllable

3 → uncontrolled in nature

Bias Errors

→ is the inability of the model to predict correctly/right data

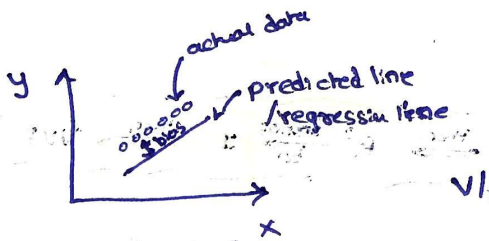


fig: low bias

V/s

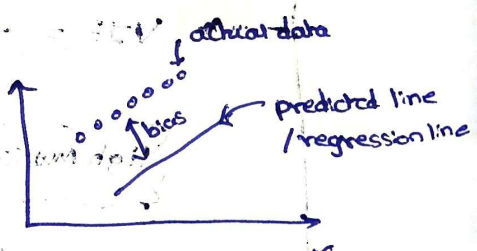


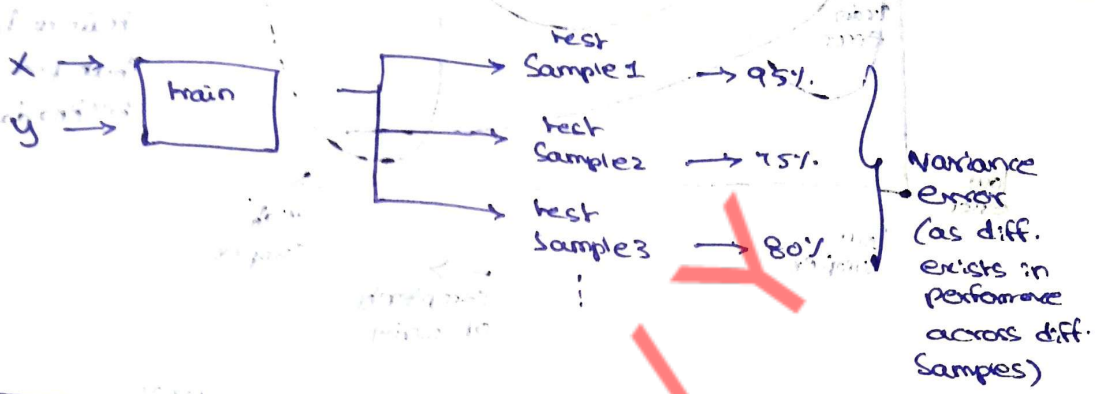
fig: High bias

→ Sources of bias:

1. Collection of data (e.g. Sample, may miss values)
2. Feature selection (e.g. remove w/o domain knowledge)
3. Feature engineering (e.g. intro. noise may be)
4. Introduced by model (if more data is on one category → learn pattern well only to that category)
5. Introduced by Data Scientist (EDA) eg. missing values, outliers.

Variance Error

- is the difference in the performance of the model for different samples.
- so, inconsistency in performance.
- model with low variance is preferred. * (bias comes in pic.)



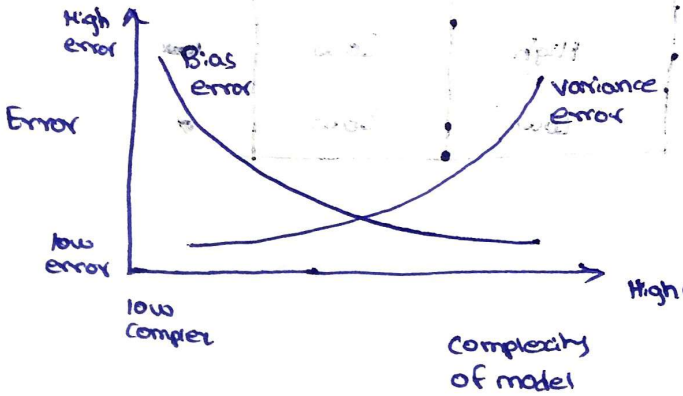
Random Error

- is of random in nature. [so, we are not known about it]

Can Errors = 0?

- Errors = Bias Error + Variance Error + Random Error
  - Bias Error → can be reduce, but not 0.
  - Variance Error → can be reduce, but not 0.
  - Random Error → uncontrolled.
- so Errors ≠ 0 Imp.

Bias Error and Variance Error



Inference 1:

↑ complexity ⇒ ↑ Variance error

↑ complexity ⇒ ↓ Bias error

Imp.

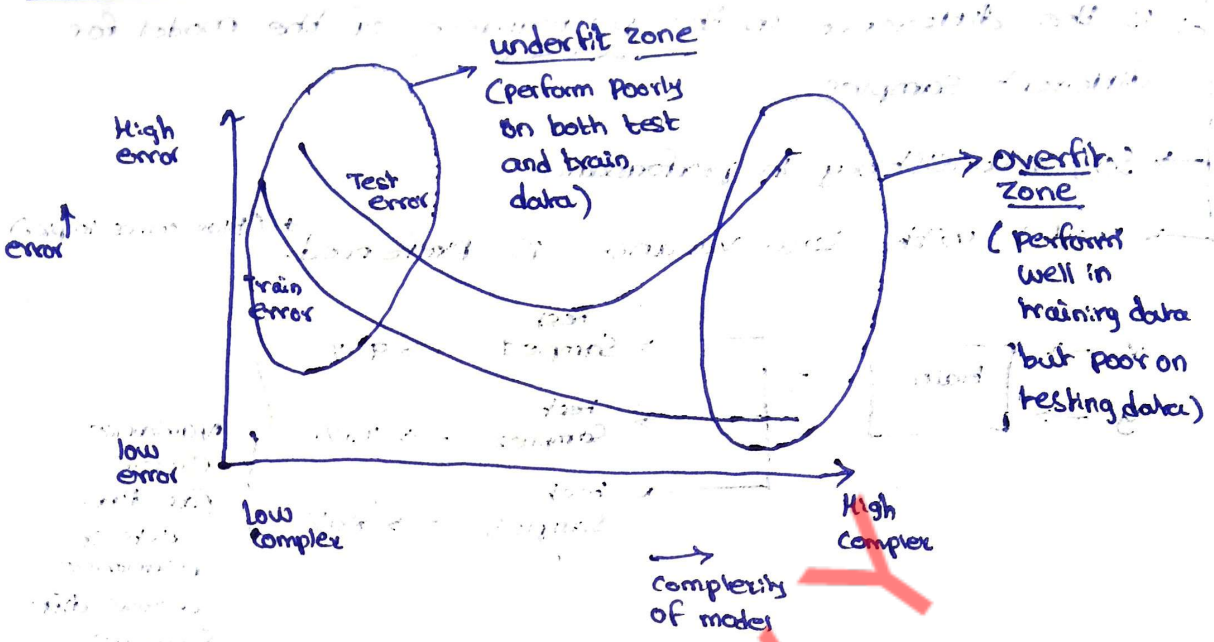
Inference 2: seesaw y/n.

↑ Bias error ⇒ ↓ Variance error

↓ Bias error ⇒ ↑ Variance error

Overfit and underfit

(based on 'Train error & 'Test error)



Bias Error, Variance Error, Overfit, underfit

→ is base / foundation for 'Regularization'.

fig 1

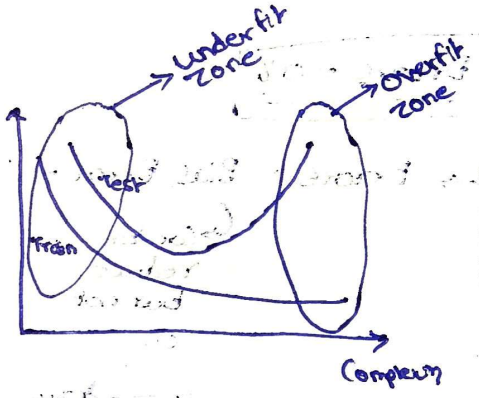


fig 2

From fig 1 & fig 2,

fit zone	Bias	Variance
Overfit zone	Low	High
underfit zone	High	Low
Good fit zone	Low	Low

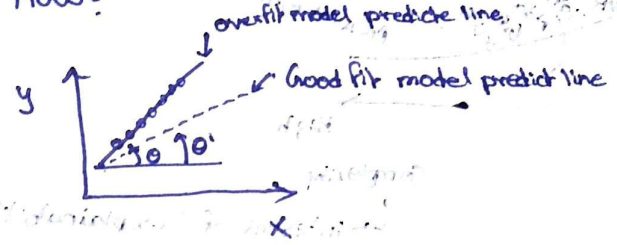
Regularization

→ is overfitted model (Low Bias, High Variance)

↓ to

Good fit model

→ How?



key thing of Reg. is

↑bias error & ↓variance error

by ↓ β

* So, it is done by introducing some bias (i.e. The inability of model to predict correctly/right data)

⇓ How to introduce bias, then?

By Shrinking / reducing the coefficient of the Independent Variables.

→ Flow :

↑bias error —automatically lead to (seesaw rln)→ ↓variance error

⇑

By Shrinking / reducing coefficients

→ Regularization → is all about Shrinking coefficients.

Overfit —become by Regz.→ Good fit

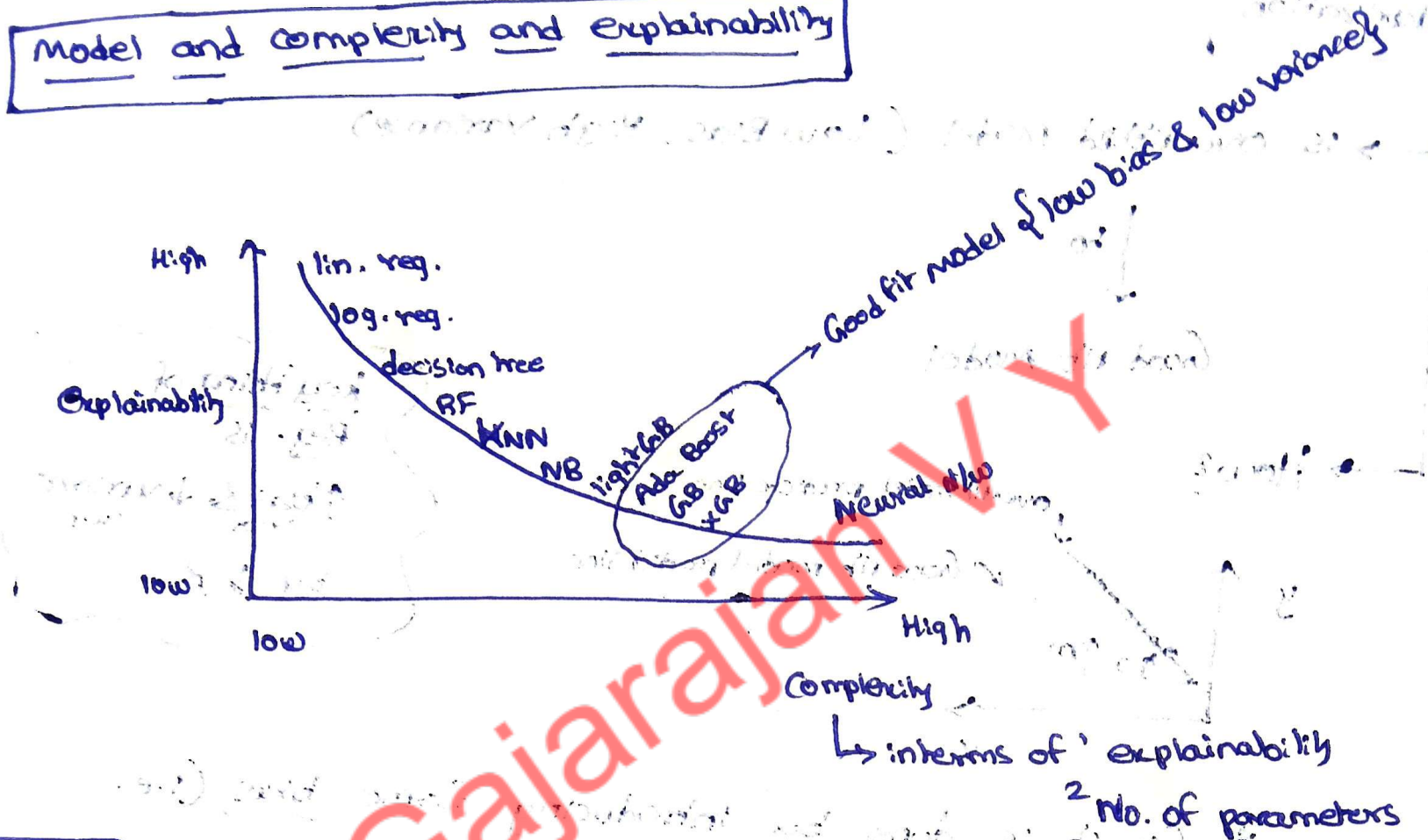
{ Low bias, High variance } { Low bias, Low variance }

Types of Regularization

→ 3 Main type

- → Ridge (some β becomes near to 0)
- → Lasso (some β becomes 0)
- → Elastic Net (some β becomes 0 and some β becomes near to 0)

Model and complexity and explainability



↳ interms of ' explainability
² No. of parameters

Extra ~~Xfzy~~

① Scaling only for model based on ' weights, & ² distance

② ~~Scaling~~ ~~good~~ Transformation can be done, of target variable.

③ DON'T SCALE
DON'T Impute Missing value } for target variable

SL-1 Day 4 Sum up

Regularization

→ By shrinking β using penalty factor.
→ also trade off b/w 'SSE & 'Bs balanced'.

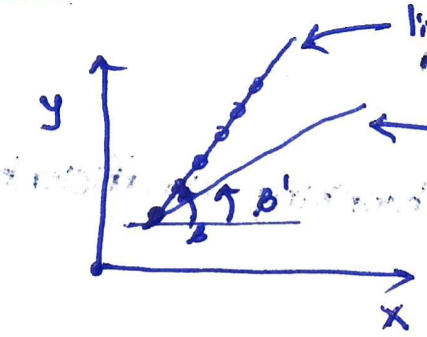


fig:

* Regularization :: why? Aim: overfit model to Goodfit (if done excessively then it may become underfit)]

what?
Shrink/reduce β
⇓ so
↑ Bias (indirectly ↓ Variance)
⇓ so
Goodfit line

key
Prerequisite:
Scaling of independent variables.

How?

Cost function	Type of regularization	
$\sum (y_i-\hat{y}_i)^2 + \lambda(b_0^2+b_1^2+\dots)$	Ridge regularization	near zero and never zero
$\sum (y_i-\hat{y}_i)^2 + \lambda(b_0+b_1+\dots)$	Lasso regularization	zero / not zero
$\sum (y_i-\hat{y}_i)^2 + \lambda_R(b_0^2+b_1^2+\dots) + \lambda_L(b_0+b_1+\dots)$	Elastic Net regularization	L_1 ratio = $\frac{\lambda_{lasso}}{\lambda_{lasso}+\lambda_{ridge}}$; 1 Some zero, 2 Some near to zero

* So, any regularization,

Trade off b/w ① SSE (Sum of squar error) [need to be min.]
② β/b (coeff) [need to be reduced]

⇓ so

Most imp:

min(SSE) + min(penalty) → so, min. cost function. [aim of Regularization]

$(y-\hat{y})^2 + \lambda$ (betas)

- $(y-\hat{y})^2$ → Sum of square errors
- λ → penalty factor / regularization factor / alpha

why trade off exists?

↑ alpha / penalty factor ⟹ ↓ Beta(s)

⇓

- effect $\hat{y}$ (as $\hat{y} = \sum$ betas × indep. vari.) ⇓ effect SSE ⇓ so balance b/w 1 SSE & 2 penalty
- ↑ Bias (indirectly ↓ Variance) ⇓ Overfit model to → goodfit / underfit model

Tradeoff in SL-1
① 1 Bias & 2 variance
② 1 SSE & ~~penalty~~ 2 β

	Bias	Variance
Overfit	L	H
Good fit	L	L
underfit	H	L

(by regulation: 1 → 2 → 3 ↓ ; 3 → 2 → 1 ↑)

So, Select the linear regression line with min. cost function [i.e SSE + $\lambda(\beta s)$] (so, optimal λ/α with optimal β)

↓

then How?

1 Grid 2 Search 3 CV

why? → O/p: optimal parameter among given possible combination of parameters.

- Grid = set of values combinations (of parameter)
- Search thru. the Grid (all combinations of parameters)
- use CV (cross validation) as estimation

→ To find the optimal parameter / alpha / penalty factor.

- → so, used with regularization to find the optimal value for alpha / penalty factor (such that trade off b/w SSE and β) is balanced.
- → Let, $\alpha = \{v_1, v_2, v_3 \dots, v_n\}$

$\alpha = v_1$ CV get → mean r_1^2

$\alpha = v_2$ CV get → mean r_2^2

$\alpha = v_3$ CV get → mean r_3^2

$\alpha = v_n$ CV get → mean r_n^2

→ Select the α_i with best mean r_i^2

('is the optimal value of α)

x among the possible combinations

(may not be the best).

1 Randomised 2 Search 3 CV

why? → O/p: optimal parameter among Random combinations of given parameter.

- → 1 Grid
- → 2 Search for Random no. of times among Random combination of parameters.
- → 3 CV use as estimator

→ To find the optimal parameter / alpha / penalty factor

→ so, 1 not all combinations of parameters

2 but a random no. of b random combination of parameters

⇓

get

Cross validation

why? → o/p: usually, Average performance of the model.
↳ e.g. r^2 score.

other? → a. Train on all samples
b. Test on all samples
→ To find the best performer.

why? → Usually,

Data
- Train (70) → model built on it may miss to learn some patterns (due to not much data on it) → mislead in performance measure.
- Test (30)

cross validation

Data
- Train (70) → let, cv = n
  - itr1 → r_1^2
  - itr2 → r_2^2
  - ⋮
  - itrn → r_n^2
- Test (no action on it) (30)

mean of r^2 scores → as performance measure (is better to use as mean)

↳ Similarly, Cross_val_score / k fold cross validation.
↳ i.e. cv = K
↳ i.e. split data into k folds.

↳ For each iteration, if Shuffle = True,
Shuffle → then → Split → then

Regularization use cases

→ 1 Overfit model → to → Goodfit/underfit model.
→ 2 reduce multicollinearity as reducing $\beta(s)$ values.
→ 3 use Lasso regularization for feature selection (as $\beta_{(s)}$ can be zero)

Gradient descent

↳ what?

- Gradient → Slope
- descent → downwards

* It finds the optimal value for β(s)/ coeff. such that it also has min (SSE).

↳ If cost func. = min(SSE) [w/o regularization]

↳ $\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \dots$

⇓

optimal βs → Such that → min (SSE) → o/p → Best fit line

↳ OLS v/s Gradient descent

	OLS	GD
	1. find optimal β(s) such that min (SSE)	1. find optimal β(s) such that min (SSE) [Both OLS, GD same function]
why GD over OLS?	1. works best only for small dataset.	1. works best even for large dataset. (as converges fast)
	2. Slower in processing	2. faster in processing.
	3. Ancient / old statistical way. (before GD invention)	

$b_0 =$

$$b_1 = \frac{cov(x,y)}{var(x)} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

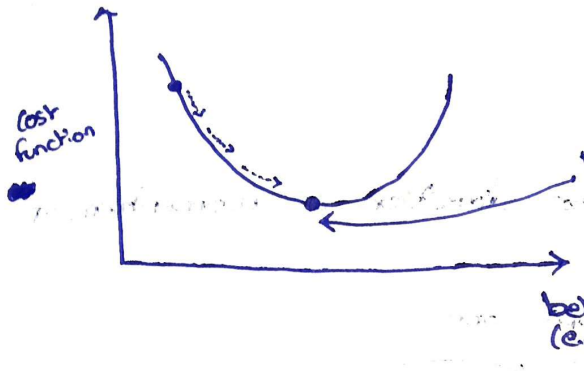
↳ ↑ cost of processing time, other resources for larger dataset with more no. of rows & columns.

↳ How it works?

Initialize random β(s) → Δ β(s) → Δ SSE

(i.e. changes β(s)) (i.e. changes SSE)

(Still find min SSE)



Note:

1. cost function can be $\sum(y-\hat{y})^2$ [i.e SSE] or $^2\sum(y-\hat{y})^2 +$ λ (betas...) [i.e. SSE + penalty for betα] → linear regression with regularisation (lasso/Ridge/elasticnet)

→ linear regression

2. cost function never zero.

↳ as Error = Bias error + variance error + Random error

Error ≠ 0.

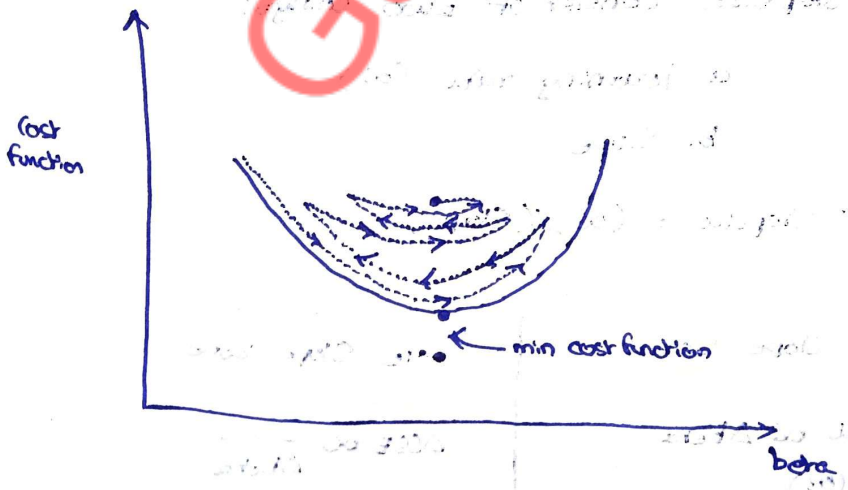


fig: process of working of GD

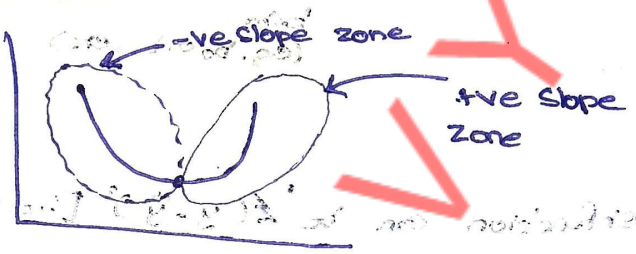
↳ is similar to the ball oscillate in a 'V' shape back & front and finally settles.

Factors responsible for working of GD

1. Direction of momentum of GD → by → Slope
2. Magnitude of change of beta → by → Stepsize

Direction & Slope

1. Slope decides the direction of momentum of Gradient descent.



2. How?

-ve slope → indicates → direction of reducing error/cost func.

+ve slope → indicates → direction of increasing error/cost func.

Stepsize

1. Stepsize consist of two things:
   a. learning rate (α)
   b. Slope
2. Stepsize = (α) X (Slope)

+ve slope zone	-ve Slope zone
$\Delta SSE \propto \Delta beta$ (or) $\Delta costfunc. \propto \Delta beta$	$\Delta SSE \propto \frac{1}{\Delta beta}$ (or) $\Delta SSE \propto -\Delta beta$ (or) $\Delta costfunc. \propto -\Delta beta$

↳ so, to find optimal beta value,

new beta = old beta − $\alpha \times$ Slope

e.g. new b_0 = old b_0 − $\alpha \times$ Slope

new b_1 = old b_1 − $\alpha \times$ Slope

logic behind it working: ★ Imp.

① −ve slope zone ⟹ slope is −ve so ⟹ ↑ new beta

② +ve slope zone ⟹ slope is +ve so ⟹ ↓ new beta

★★ Imp v.v.Imp

Formula:

$b_0 = -(y - \hat{y})$

$b_1 = -(x)(y - \hat{y})$

↳ learning rate (α)

Early 2014	After 2014
① fixed α	① dynamic α based on min. cost function / min. SSE ⇑ i.e., Adam's algorithm
② Need manual intervention for precision.	

↳ what does 'back and front' glide indicates?

↳ precision is the Answer.

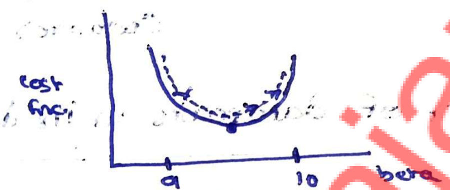


fig. front

find e.g. min cost func. in [9, 10]

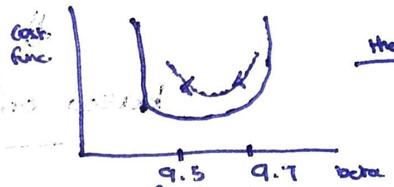


fig. back

find e.g. min cost func. in [9.5, 9.7]

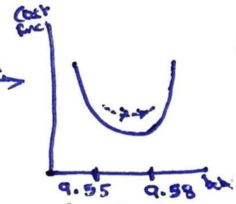
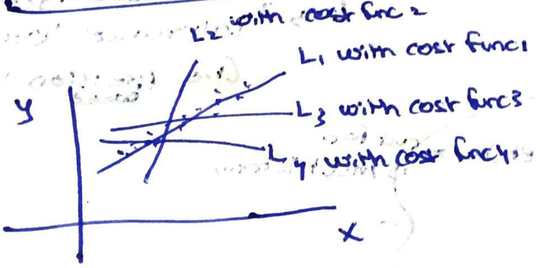


fig. front

find e.g. min cost func. in [9.55, 9.58]

So, from each glide, there is increasing level of precision for finding optimal β(s) @ min. cost function.



⟶ Select the line (L_i) with min. cost func (CF_i)

Created by GD.

Types of Gradient descent (Mainly)

① Batch Gradient descent

↳ 1 Compute Gradient (Slope) and 2 update parameters (beta, etc)

⇓

based on entire dataset

↳ useful in small dataset & convex problem.

global min. only in convex problem

② Stochastic Gradient descent

↳ 1 compute Gradient (Slope) and 2 update parameters (beta, etc)

⇓

based on each datapoint in the dataset.

↳ useful in large dataset, & non-convex problem.

escape local mini. in non-convex

③ Mini-Batch Gradient descent

↳ 1 compute Gradient (Slope) and 2 update parameters (beta, etc)

⇓

based on each subset of datapoints in the dataset

↳ useful in 1 stability & 2 speed balancing

[leverage parallel computing]

When to Stop?

↳ based on any one of two criteria achieving

↳ 1 max. no. of iterations (usually, 1000)

↳ 2 tolerance level. (i.e. precision based)

Where GD is used

1 linear regression → cost fnc. is MSE

2 Regularizn (lasso, ridge, elasticnet) → cost fnc is MSE + L term

3 Ensembling tech. (XGB, etc)

4 Neural network (β(s) as → weights)

β, Pvalue

β or Explainatory power

↳ it says for 1 unit of change in X_i (feature) brings β units of change in y (dependent variable)

P-Value

↳ it tells about whether variable is significant or not.

↳ H_0: insignificant variable

H_1: Significant variable

↳ Significance is the contribution by that independent variable to target variable.

Interaction effect

↳ is independent variable / features (x_1, x_2) interact with each other (i.e. multiplication operator) to bring change in target variable (y).

↳ H_0: No interaction effect
H_1: interaction effect
} → i.e. based on Pvalue of the new created interaction variable [$x_1 * x_2$]

↳ ≠ Multicollinearity (i.e. High correlation b/w independent variables)

Measures

MAPE

↳ Mean Absolute percentage error (MAPE)

Error = A-P

percentage Error (PE) = $\left(\frac{A-P}{A}\right) \times 100$

Absolute percentage Error = $\left|\frac{A-P}{A}\right| \times 100$

Mean absolute percentage error = $\frac{1}{n} \sum \left|\frac{A-P}{A}\right| \times 100$ //

↳ RMSE is preffered over MAPE (as MAPE get inflated with min. A and max. P)

MAE

↳ Mean absolute error (MAE) $= \frac{1}{n}\Sigma|A-P|$.

↳ less preffered to use.

Adjusted R^2

↳ Adj. $R^2 = 1 - \frac{(1-R)(n-1)}{(n-k-1)}$

n → no. of observation / rows

k → no. of features

Adj. $R^2 \leq R^2$

↳ why Adj. R^2 ?

↳ because of drawback of R^2.

↳ i.e. As ↑k (no. of features), irrespective of feature/variable is important R^2 value increases telling that variable is important (but in reality may not be the case)

⇓

So, Adj. R^2 penalizes the value by $(n-k-1)$ for increasing k.