Machine Learning 101

Lecture ① :

a) get data
b) Problem objective
c) decide algo
d) run
e) validate result.

Estimation.

generalization.

•) Supervised learning :

$$D_n = \{(x^1, y^1), (x^2, y^2) \dots (x^n, y^n)\}$$

input → x^i , output → y^i

$x^i \in \mathbb{R}^d$ → dimension.

$y^i \in \{+1, -1\}$ → Binary classification

+1 → Positive example
-1 → Negative example.

$\varphi(x)$: feature representation $\in \mathbb{R}^d$

•) Regression → Supervised with $\{y \in \mathbb{R}^D\}$

•) Unsupervised learning:

Density estimation:

given: $x^1, x^2 \dots x^n \in \mathbb{R}^D$ } → Drawn from $P_r(X)$ Distribution.

Predict

→ $P_r(x^{n+1})$ of an element drawn from same distribution.

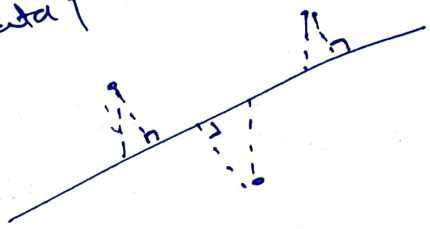
Setup

L1 ML intro

L2 Regression:

pip install Quandl → get data { Quandl.get(' ') }

pip install pandas



(Features, Labels) → Supervised..

Input

output

Scaling → Feature → (-1, +1)

Cross_Validation → shuffles the data & breaks it into training & testing.

Sklearn → preprocessing
Sklearn . Cross-validation
Sklearn . LinearRegression
Sklearn . SVM

X_train, X_test, Y_train, Y_test = Cross_Validation.train_split(x, y, test_size = 0.2)

n_jobs → used to thread the regression process parallelly.

Classifier = LinearRegression();
clf.fit(X_train, Y_train); → training.
clf.score(X_test, Y_test);
↓
accuracy

classifier = SVM.SVR()
↓
support vector regression

kernel = 'polynomial' / 'linear'

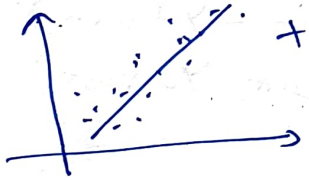
L-5

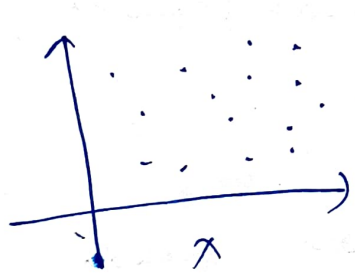
Regression for future prediction

Pickling : (Pickle file) lets us not to run the classifier for the whole data again and again. if the new data is entered.

L-6 :

Regression.





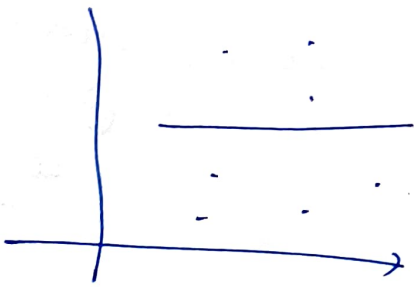
→ X is not very well Co-related with Y, therefore doing something like OLR will not be a good idea.

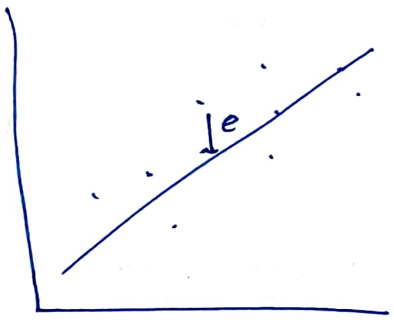
$Y = mx + b$ → best fit

$$\left.\begin{array}{l} m = \dfrac{\bar{x}\cdot\bar{y} - \overline{xy}}{(\bar{x})^2 - \overline{(x^2)}} \\ b = \bar{y} - m\bar{x} \end{array}\right\} \text{L.R.}$$

2.10 : R-squared (Coefficient of Determination)

↑ accuracy





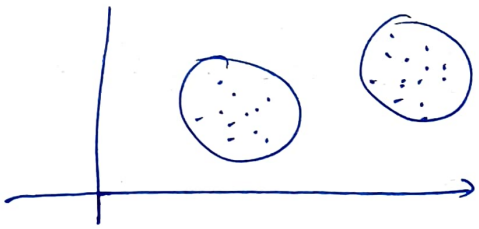
$$r^2 = 1 - \frac{SE(\hat{y})}{SE(\bar{y})} \quad (0-1)$$

r^2 should be high

L11 : Testing Assumptions :

Creating your own dataset

L13 & 14 : Classification :



k = odd

} To break tie }

k > no. of groups

Clustering

k = 3

↳ (- - +) ⇒ 66% Confidence

Not accuracy

Euclid dist

Classifier = KNeighborsClassifier()

sklearn.neighbors

L15 : Euclid dist = $\sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$

q_i → self

p_i → neighbour.

L17

To inc. accuracy ⇒ we can shuffle the data

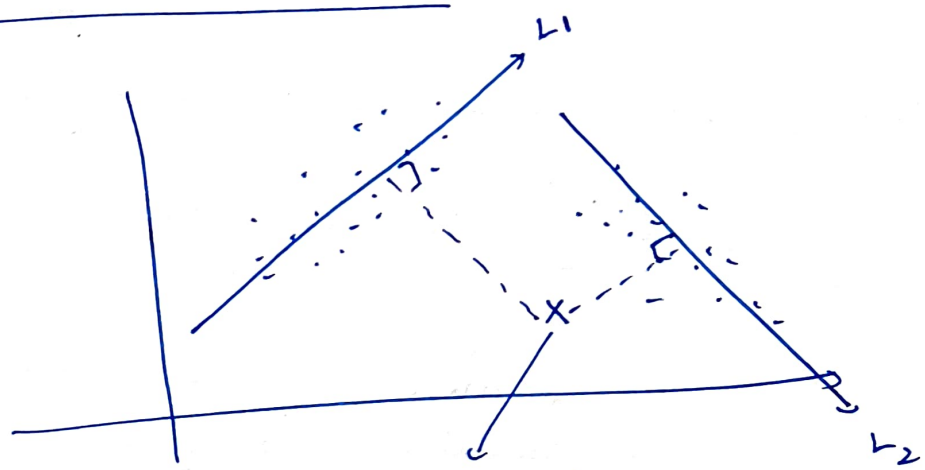
L19

K, accuracy & Predictions

There is two diff^ b/w Confidence & Accuracy. *

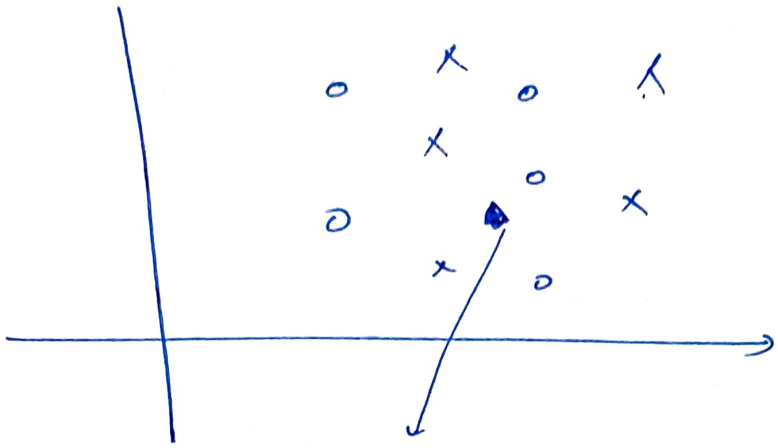
we → can thread, to ~~incr the~~ + Make it fast.
→ we can use Radius Concept.

KNN ——→ Line regressi.



we can Measure
the SE of the point from
the 2 regression line and
put it the there. and in case
Error.

But for



to be predicted

Here we can't use LR

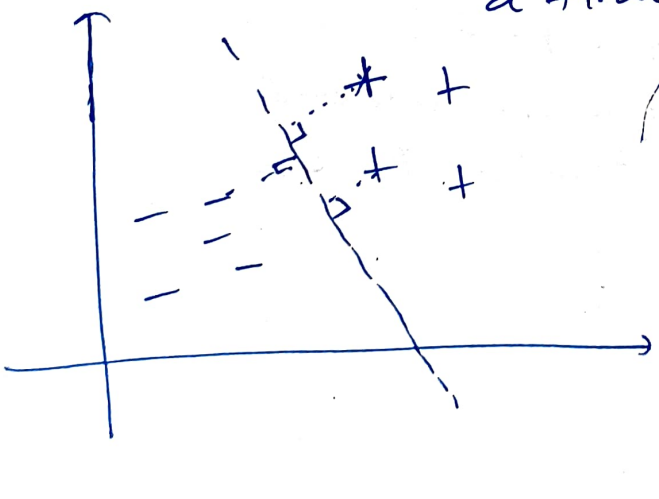
Means we can use but the accuracy will be very bad due to very bad co-relation

But we can use KNN here.

L20 SVM : Supervised Machine Learning classifier

Support Vector Machine (Is a Binary classifier)

6 means it can only separate two groups at a time.

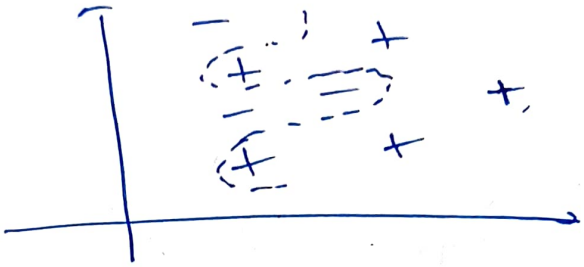


SVM finds the best hyperplane (best separating boundaries).

Maximum

Hyperplane → has the ~~best~~ possible separation distance between points on the two sides within it.

As it uses hyperplane ∴ we can ~~do it on a line~~ use it on a linear data & Not some data like



⇒ Is Perfectly not exist

Sklearn

→ svm.SVC( )

Support Vector Classifier

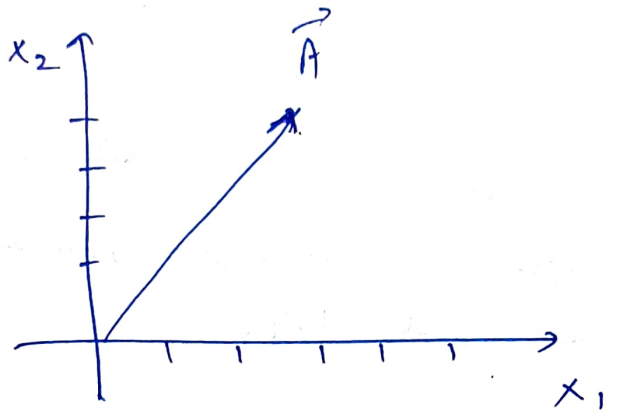
{# Including id Gamma only reduced the accuracy of our classifier.}

L21

SUM

$\vec{A} = [3, 4]$

$||\vec{A}|| \Rightarrow$ Norm / Mag.



L22:

$\vec{x} \cdot \vec{w} + b \geq 0 \Rightarrow$ '+' Sample

$\leq 0 \Rightarrow$ '−' Sample.

i) $\vec{x} \cdot \vec{w} + b = 0 \Rightarrow$ on the Decision boundary.

$\vec{x}_{-sv} \cdot \vec{w} + b = -1$

& $\vec{x}_{+sv} \cdot \vec{w} + b = +1$

y (Class) $\rightarrow$ +ve $= +1$

$\rightarrow$ −ve $= -1$

+ Class : $(\bar{x}_i \bar{w} + b = 1) \times y_i$

$(y_i = 1)$

- Class : $(\bar{x}_i \bar{w} + b = -1) \times y_i$

$(y_i = -1)$

y_i

$y_i(\bar{x}_i \bar{w} + b) = 1 \cdot y_i$

$y_i(\bar{x}_i \bar{w} + b) - 1 = 0$

$(\bar{x}_i \bar{w} + b) y_i = -1 \times (-1)$

$= 1$

$y_i(\bar{x}_i \bar{w} + b) - 1 = 0$

Same Equation:

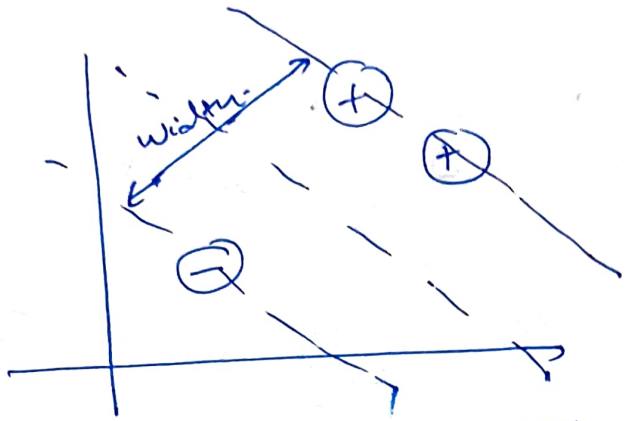
SVM use

$$y_i(\bar{x}_i \bar{w} + b) - 1 = 0$$

Maximizing with b/w the support vectors.

L23°

$$y_i(\bar{x}_i \bar{w} + b) - 1 = 0$$



$$\text{width} = (\bar{x}_+ - \bar{x}_-) \cdot \frac{\bar{w}}{\|w\|}$$

$$y_i(\bar{x}_i w + b) - 1 = 0$$

expand

$$\bar{x}_+ = 1 - b$$
$$\bar{x}_- = 1 + b$$

on solving we find

$$\text{width} = \frac{2}{\|w\|}$$

Maximize

$\therefore$ we minimize $\|w\| \Rightarrow$

$\therefore$ Minimize $= \boxed{\frac{1}{2}\|w\|^2}$ for mathematical convenience

$\left(\frac{1}{2}\|w\|^2\right)$

↳ Maximize with constraint $\left(y_i(\bar{x}_i \bar{w} + b) - 1 = 0\right)$

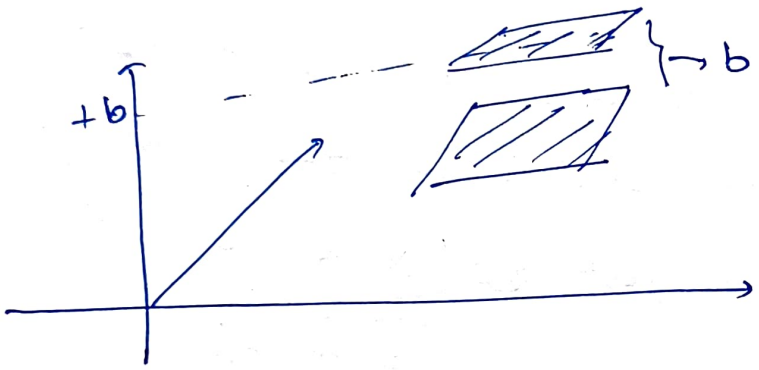
Lagrange multiplier

Lagrange,

$$L(w,b) = \frac{1}{2}\|\bar{w}\|^2 - \sum \alpha_i \left[y_i(\bar{x}_i \bar{w} + b) - 1 \right]$$

we want to minimize 'w' and maximize 'b'

'b' → biase



$$\frac{\partial L}{\partial w} = 0 \;,\; \frac{\partial L}{\partial b} = 0$$

$$\frac{\partial L}{\partial w} = \bar{w} - \sum \alpha_i y_i \bar{x}_i = 0$$

$$\therefore \left\{ \bar{w} = \sum \alpha_i y_i \bar{x}_i \right\} \rightarrow (1)$$

$$\left.\frac{\partial L}{\partial b} = -\sum \alpha_i y_i = 0 \right\} \rightarrow (2)$$

$$L = \sum_j \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \cdot (\bar{x}_i \cdot \bar{x}_j)$$

↑ Quadratic

downside

SVM ⇒ has same problem of speed or in KNN, but it can be done by taking batches at a time.
Mini batches

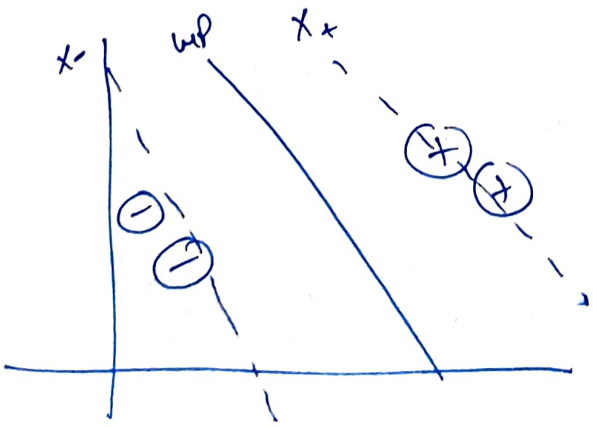
SMO ⇒ Sequential Minimal optimization.
Some use tips for making it fast.

Upside of SVM:

once we have trained the data set we don't need old feature & the classifier for the new point is just $\underbrace{(\vec{x} \cdot \vec{w} + b)}_{\text{Sign of}}$

L24:

My perplane ⇒ $(X \cdot W + b) = 1$ (+ class) SV
$= -1$ (− class) SV



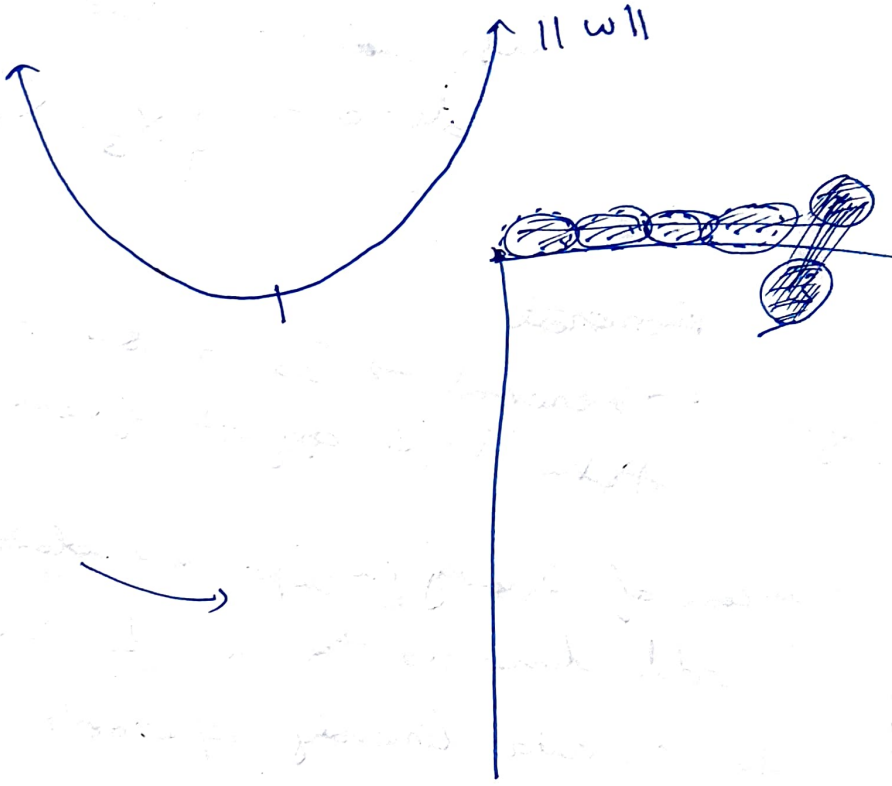
Find $\bar{w}$ & b ?

optimization objective:

To min $\|\bar{w}\|$ & max 'b'

Constraint to $\Rightarrow$ $y_i(\bar{x}_i \bar{w} + b) \geq 1$

$\therefore$ a Convex problem



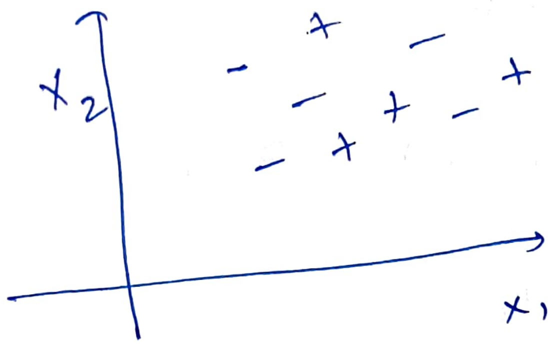
$\bar{x}_i \Rightarrow$ known.

$w, b \Rightarrow$ unknown.

L25: Kernels: $\rightarrow$

Kernel

Non linear data



Lets ~~add~~ in a 3rd dimension: $\{X_3 = X_1 * X_2\}$

~~Kerenal~~

(use to argument) Kerenal → Is a similarity function take input arg. and given an output.

⇒ In case of linearly in seperable data what we do is add dimensions to our features set to make our data linearly seperable.

(dot = inner) product

X space ⟶ Z space

Unknown feature set $\bar{x}'$,

classification of $\bar{x} \Rightarrow y = \text{sign}(\bar{w} \cdot \bar{x} + b)$

$\bar{w} \cdot \bar{x}$ → scalar

∴ we can replace $\bar{x}$ to $\bar{z}$ (any dimension)

Constraint

① $\{ y_i(\bar{x}_i \cdot \bar{w} + b) - 1 \geq 0 \}$

$\bar{z}_i$

②

$$w = \sum \alpha_i \cdot y_i \cdot x_i$$

$$L = \sum \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j \cdot y_i \cdot y_j \cdot (\bar{x}_i \cdot \bar{x}_j)$$

kernel (ϕ)

$K(x, x') = \bar{z} \cdot \bar{z}'$ → scalar

$\bar{z} = \text{func}(x)$

$\bar{z}' = \text{func}(x')$ → same function

$\{ y = w \cdot \phi(x) + b \}$ ⇒ using kernel.

Feature set $X=[x_1, x_2]$ ⇒ 2nd order polynomial

$$z = [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2]$$

(2nd order; z-space)

$$K(x, x') = z \cdot z'$$

$$z' = [1, x_1', x_2', x_1'^2, x_2'^2, x_1' x_2']$$

$$K(x, x') = z \cdot z' =$$

$$\hookrightarrow 1 + x_1 x_1' + x_2 x_2' + x_1^2 x_1'^2 + x_2^2 x_2'^2 + x_1 x_1' x_2 x_2'$$

we can actually use kernel without using the z space above:

$$K(x, x') = (1 + x \cdot x')^p \equiv \text{polynomial kernel}$$

p → degree of poly

x, x' = has dimension 'n'

$$K(x, x') (1 + x_1 x_1' + x_2 x_2' - - - + x_n x_n')^p$$

⇒ if we choose $n = 15$ & $p = 100$

how can we do it.

RBF ⇒ Radial basis Kernel.

Default kernel.

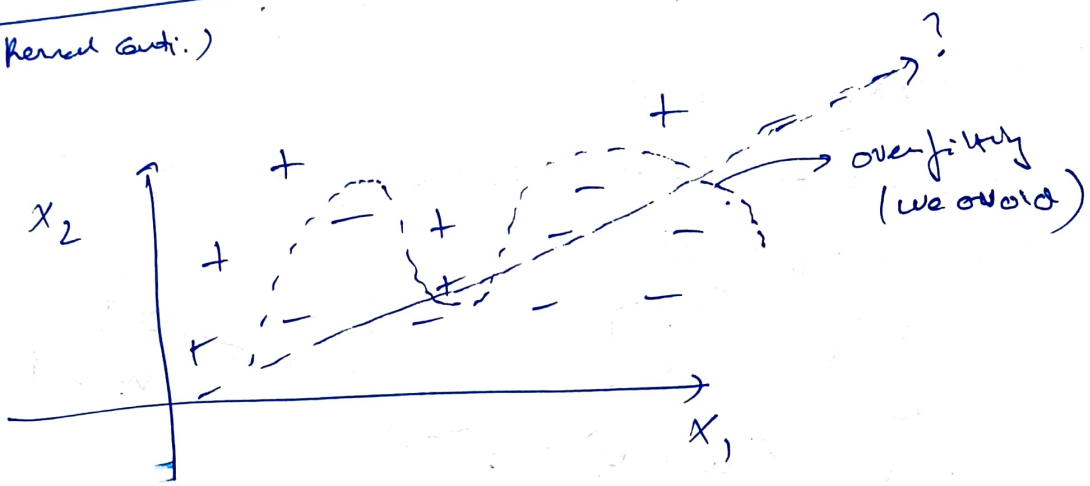
~~$K(x, x') = \exp(-\gamma$~~

$K(x, x') = \exp\left(-\gamma \|x - x'\|^2\right)$

can take us to ∞ dimension

L31 Soft Margins in SVM

(Kernel contd.)



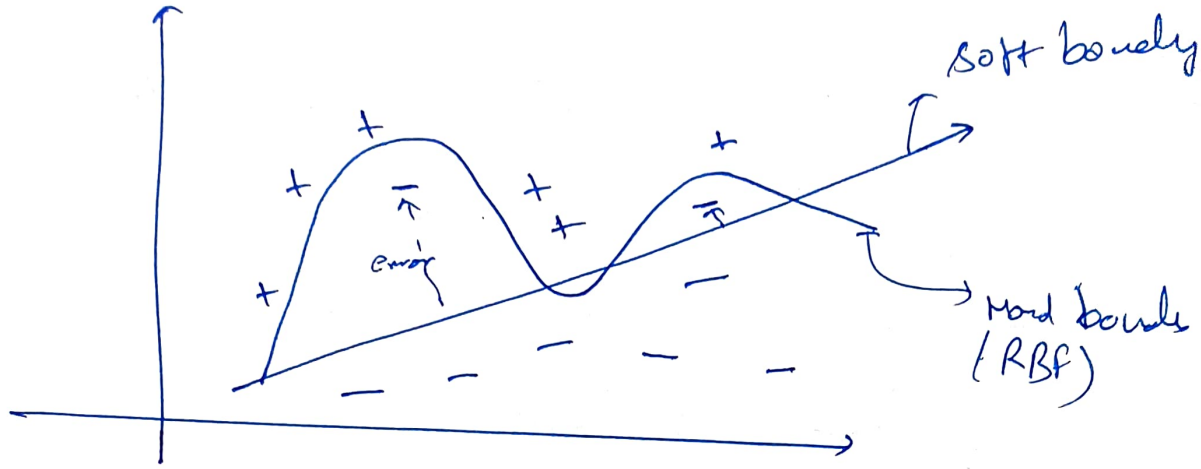
if $\frac{\#\text{ points on SV}}{\#\text{ samples}} > 10\%$ overfitting

if we have [52% accuracy & 80% SVs] → overfitting.

if we have [52% accuracy & 8% S.V.s]

then the data is probably not good. work out & ∴ try a different Kernel.

Soft Margin classifier SVM



Soft Margin Classifier → allow some degree of Error in the hyperplane. we allow do this using Slack.

Slack = ξ_i $\{\xi_i \geq 0$ → Hard Margin

$y_i(\bar{x}_i \bar{w} + b) - 1 \geq 0$

$y_i(\bar{x}_i \bar{w} + b) \geq 1 - \xi_i$

Total Slack = $\sum_{\forall i} \xi_i$

~~Now we want~~

Now we want to Minimize the Slack,

∴ Now we will Minimize = $\min\left(\frac{1}{2}\|w\|^2 + C\sum_{\forall i} \xi_i\right)$

Controls the Voilation Margin.

i.e is $C\uparrow$ ⇒ we allow Hard Less Voilations.

& if we $C\downarrow$ ⇒ we allow Voilation.

L32

"Cvx opt": Pattern recognization.

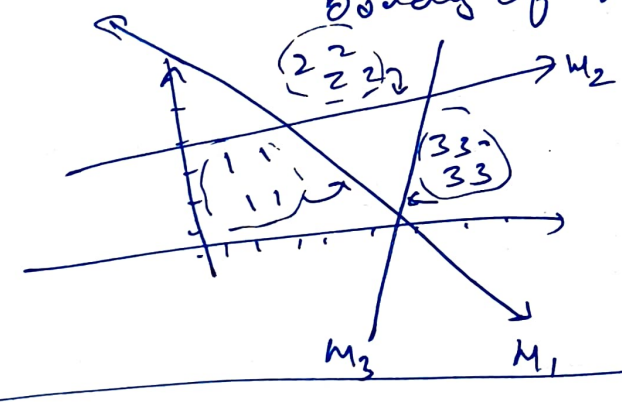
→ linear kernel
→ polynomial kernel
→ Gaussian kernel
→ Radial Basis function.

L33: Classifying More than 2 groups using SVM:

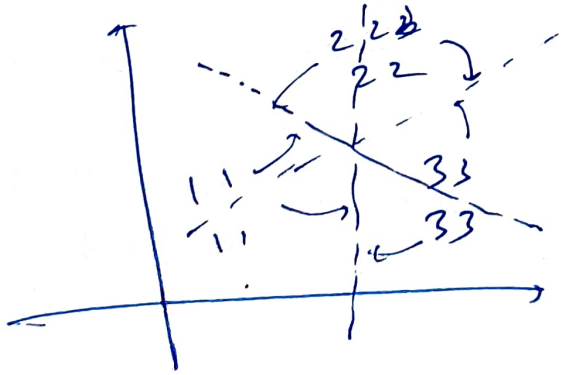
OVR → one vs Rest
OVS → one vs one

OVR

Problem :→ have sort of weighting issue i.e imbalance b/w the two # of data points on two side of the boundry of hyperplane.



OVO



SKlearn

SVM.SVC

kernel
→ 'sigmoid'
→ 'precomputed'

"probability" → just like "confidence" in the K-NN

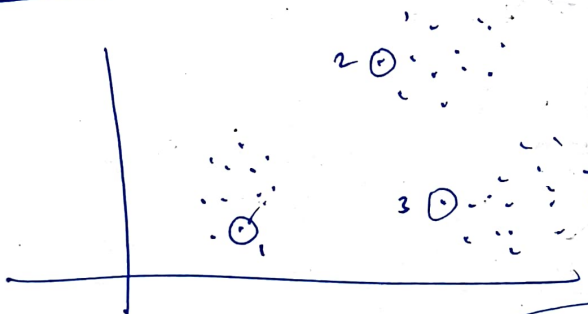
"shrinking" uses SMO if True, to ignore the non important features while fitting the data.

"verbose" !

L34 Unsupervised Learning

Clustering → Flat → we give no. of clusters
→ Hierarchal → Machine figures out (Meanshift) the number of clusters.

K-Means :



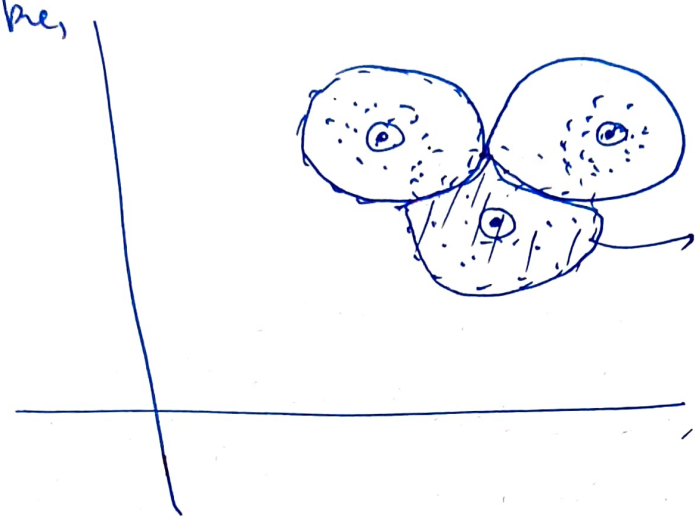
find dist of all points from (1), (2), (3) → random chosen centroids.
& assign to it the cluster closest to it.
Now find new centroid of the each cluster and repeat the process until the new centroid becomes equal to the old one.

~~Less with house Confidence~~

We will have → "tolerance" & "max_iter" parameter in this.

Downside of K-mean

→ In like, house data



This must not be like this problem can be resolved by use of kernels. due to eculidian dist^n

Up side

→ we don't have to train again & again just like SVM.

{ Convert non-Numerical data column → Numeric data column }

Titanic dataset : for k=2 (survival / non-survival)

can also use SVM,

* # Scaling the features make a great changes in accuracy. (like in titanic data).

L-37

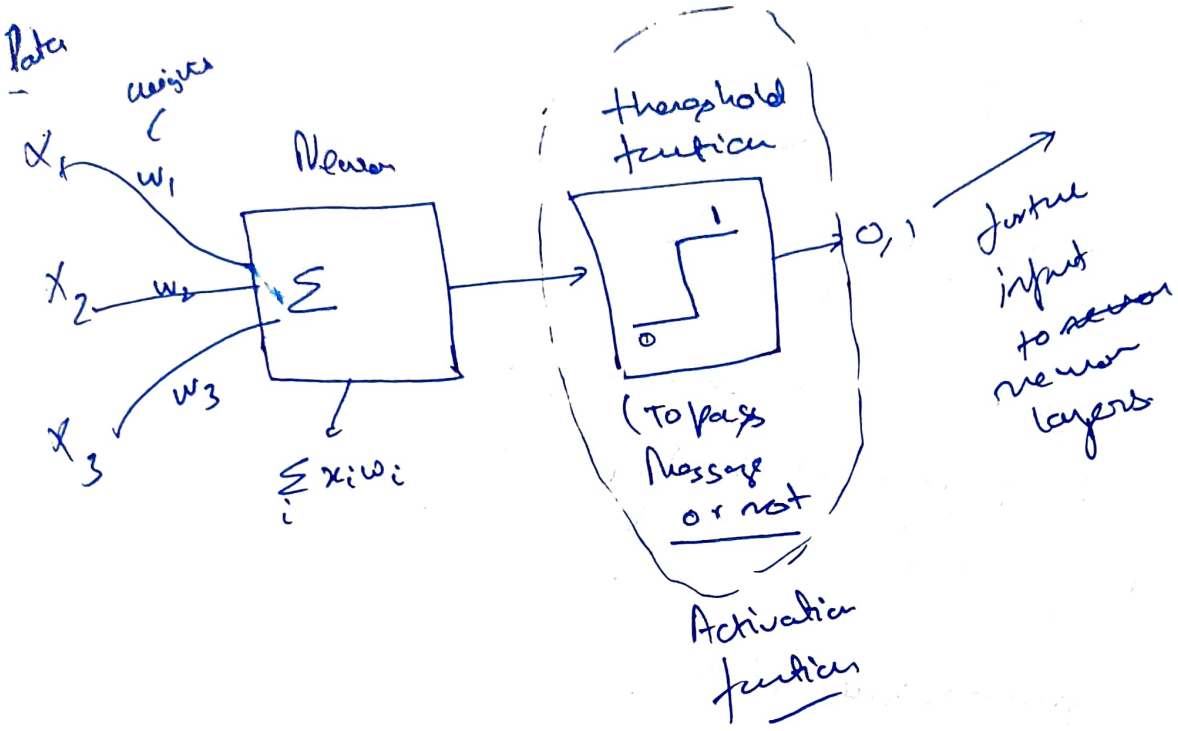
Predicting : in K-Mean we predict. here new data by seeing the dist" from the previous centroids.

L38 : Heirarcial Clustering :

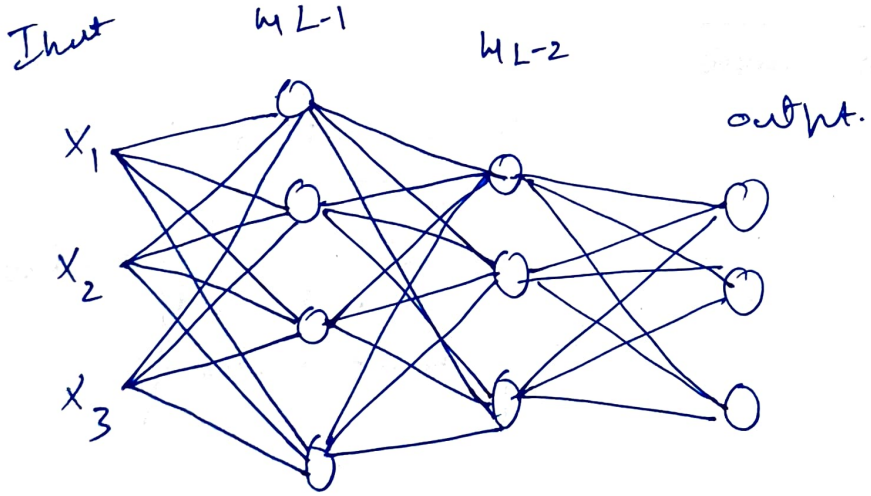
→ Choose every data point one by one
→ find the centroid of all the points lying in the radius/bandwidth of 'r' with the chosen point as center. (takes default value if not passed)
→ after finding the new centroid, do the same until the centroid converges //
→ Do this for all the data points
Ban!! we did heirarcial clustering.

Automating Radius/bandwidth value :

L43: Neural Networks



$$Y = f(\bar{x}\bar{w})$$



1 HL → Regular N.N

2
> 1 HL → Deep N.N.

~~Huggingface~~

Neural Networks works for doing really good Modelling analysis.
(Modelling logics).

N.N → very impressive.

Image Data → Imagenet
Text Data → Wikipedia,
Speech Data → Tatoba.

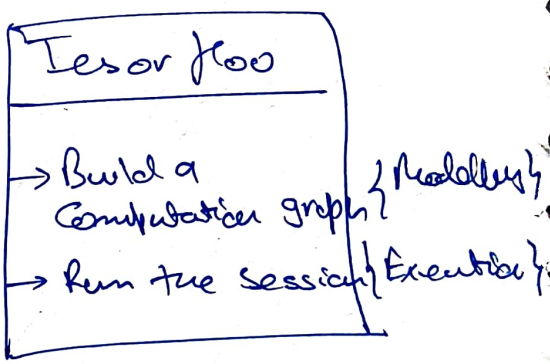
Commoncrawl data

L 44:

Tensorflow (Framework) → Interactive
→ works in chunks.

(Deep learning lib), like numpy.

tf.Session()



LMS:

MNIST: 60,000 training examples of handwritten digits. (28×28) pixels & 10,000 testing samples.

Feed forward Neural Network → Pass the data right through the last hidden layer to the output layer

⇒ Data in the output layer is compared to the Intended output then we create a cost/loss function. (Cross entropy).

⇒ Then we use optimizer to minimize the cost.

AdamOptimizer, StochasticGradientDescent (SGD) AdaGrad etc.

⇒ what the optimizer does that it goes backwards and Manipulates weights {Back propagation}

⇒ feed forward + back prop = epoch {epic}

Input data $\xrightarrow{\text{weights}}$ $\vec{X}\vec{\omega}$ $\xrightarrow{\text{bias}}$ $(\vec{X}\vec{\omega}+b)$

Why bias?

Eg $(\vec{X}\vec{\omega})$ → if all of the input data is zero i.e $\vec{X} = \vec{0}$ then $(\vec{X}\vec{\omega} = 0)$ ∴ Input will be zero ∴ ~~we add a bias 'b'~~. No neuron will ever fire/activate. ∴ we add bias 'b' to this to making a neuron active. some.

for every neuron Node.

# of nodes in the output layer = No. of classes.

relu ⇒ Rectified linear Activation function

Init: def test NN(x):

prediction = n_neuron_Model(x)

Cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(prediction, y));

optimizer

Generally,

1 Hidden layer → Linear data
2 Hidden layer → Non linear data.
3 or

(Buffering → for huge data)

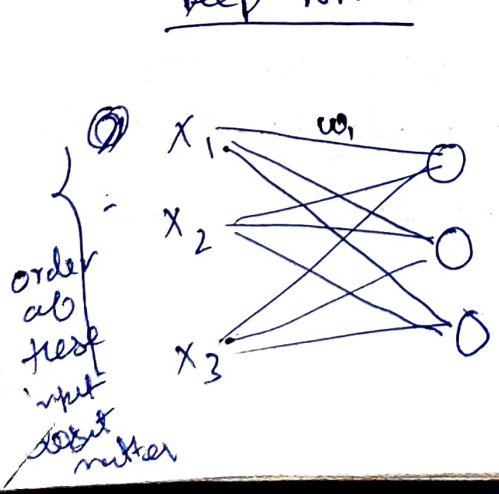
LSTM : RNN (Recurrent Neural Network)

Traditional FF.NN (Feed forward neural network / Deep neural network)
do not account in the parameter ab. time or the order in which the things are happening.
Ex: (Aball money away / towards)

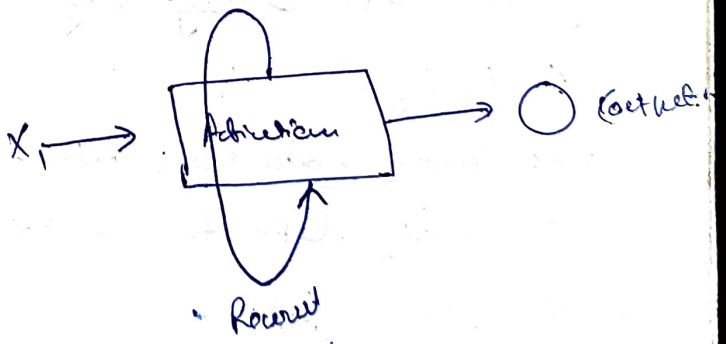
RNN .. is mostly used for language data.

LSTM cell
Gated Recurrent cell

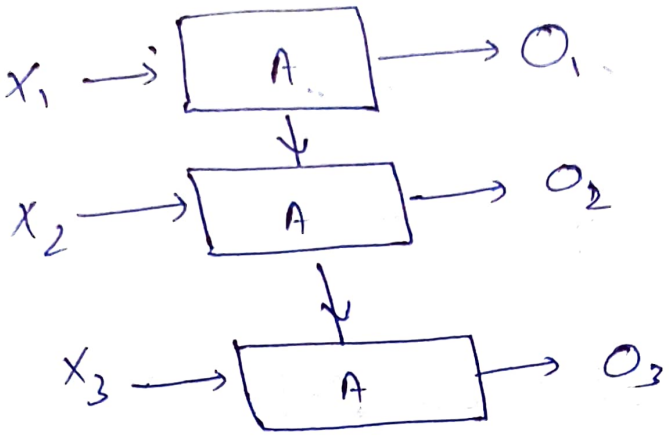
Deep NN



Recurrent NN



RNN can also be visualised like this -



So, this is how it is different from Deep NN,
So, if we have a sentence:

"Harrison Drove the Car"

for deep Neural Network the order will not matter & it may be same as -

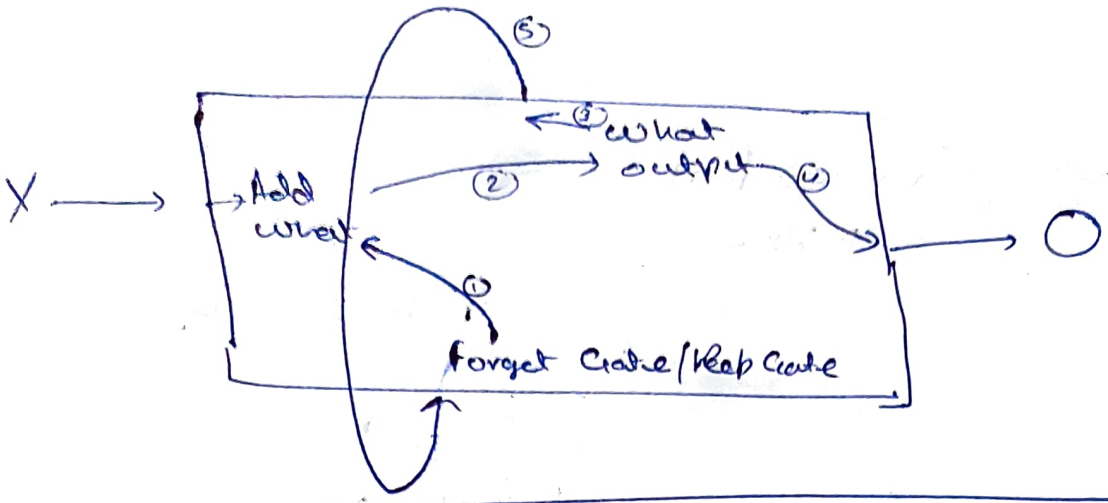
"The Car drove Harrison".

⇒ But in RNN the order/time of occurence really matters.

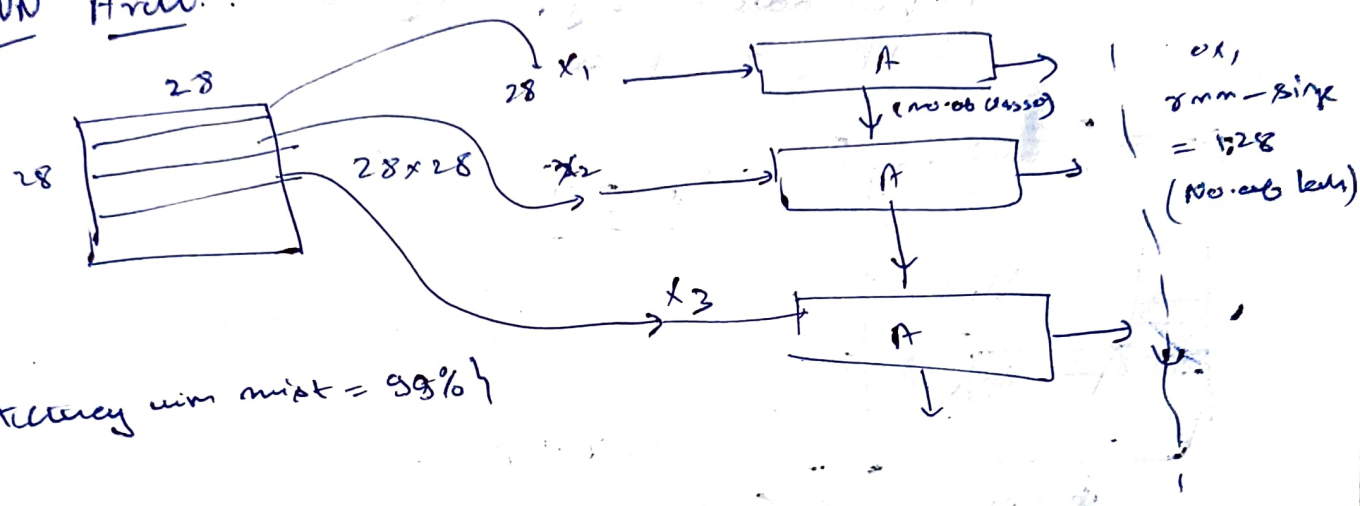
LSTM cells: It is an RNN cell, which do not

If we have a very long sequence of input like $x_1, x_2, x_3 \cdots x_n$, from this series we might have many data terms which doesn't really even matter & we only like to memorise few terms ∴ we have LSTM Cell

LSTM Cell



RNN Arch. :



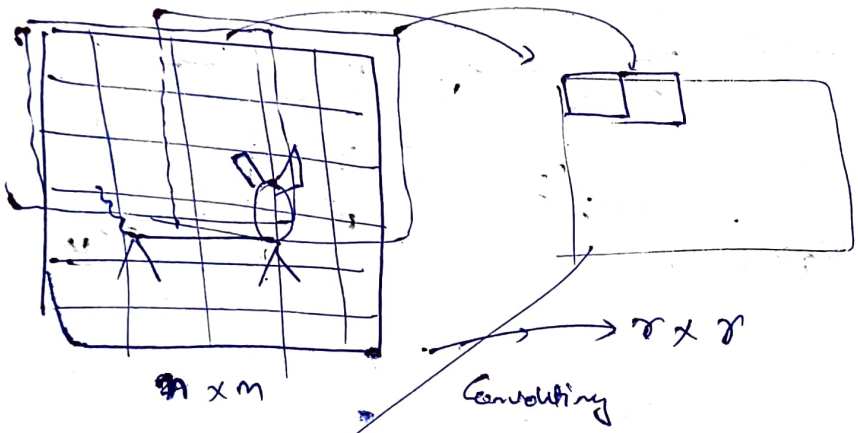
Accuracy on mnist = 99%

L 56 CNN : Conv Net

input ⇒ (Conv ⇒ Pool) ⇒ Fully connected layer. ⇒ output.

hidden layer

Convolution → Generating feature map from the input data. Using sliding window, without skipping any pixel.



Pooling :
(No overlap)

Suppose the new feature map is

1	2	2	1	1
1	0	3	1	
1	4	1	2	2
3	2	2	0	2

Pooling

4	3
4	3

Pooling is Simplifying

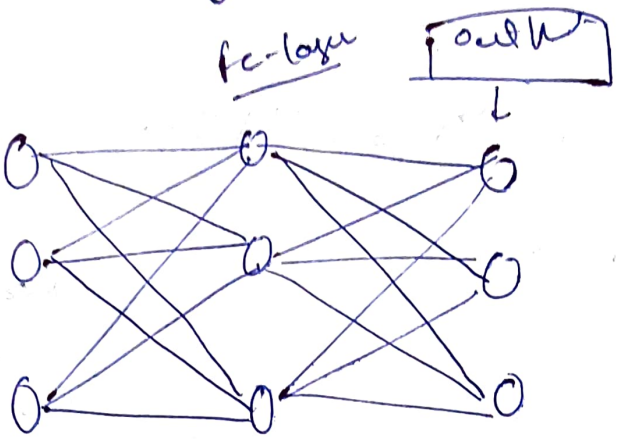
→ Let's take a 3x3 pool

Ex: Max pool ⇒ take max value in the window

(Conv + Pool) = Hidden layer

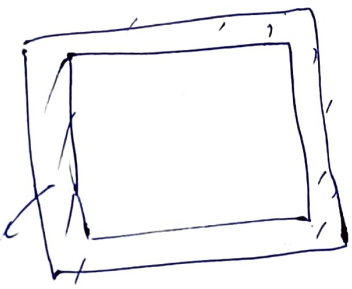
Fully Connected layer:

⇒ Every neuron in the previous layer is connected to every neuron in the next layer.



[5, 5, (1), (32)]

- 5×5 convolution
- take 1 input
- produce 32 outputs.



For small dataset RNN works better than CNN.

Dropout in CNN: de-activates some unnecessary neurons during the fc layer.

LS8: TfLearn :-

High level abstraction layer library on top of tensorflow.

" Tflearn, Keras, SkFlow, TfSlim "

↳ less error prone.

Read:
Convolution Neural Network

AlexNet => for image data.

Overlapping pooling :-

↳ was used in AlexNet as a reduction of 0.5% error was obtained

Batch Normalization: Technique used before giving input to a convolution layer to migrate the effect of unstable gradients within deep neural Network.

input $X = \{x_1 \dots x_m\}$

parameters: w, b {weights & biases}

$$y_i = BN_{w,b}(x_i)$$

Normalize

$$\therefore \mu = \frac{1}{m}\sum_{i=1}^{m} x_i \quad \{m = \text{mini batch size}\}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x-\mu)^2 \quad \{\text{mini batch variance}\}$$

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \quad \text{// Normalize.}$$

$\{ y_i = w\hat{x}_i + b \} \Rightarrow$ Scaling & Shifting.

Optimizer: used to change attributes of NN such as weights & learning rate in order to reduce the loss

#GradientDescent: 1st order optimization.

$$\theta = \theta - \alpha \cdot \nabla J(\theta)$$

- Easy to reach local Minima.
- Fast computation
- High Memory

weights are changed after calculating gradient on the whole dataset. So takes very long time.

Stochastic Gradient Descent:

Requirement: Randomized training dataset.

↳ as the weights are updated at each training set. : there can be very much noise & cost function can keep jumping.

⇒ Mini-Batch Gradient Descent: optimize weights after every batch.

⇒ Momentum: ↗ hyperparameter 'γ' (≈0.9)

Momentum: ⇒ Converge fast optimizers

⇒ Adam: (see)

⇒ Adagrad: ⇒ change learning rate 'η' for each parameter and at every time step ('t')

⇒ Second order optimization

Model parameters ⇒ (w, b) → in ANN

(Support vectors) → SVM

(Coefficient) → Linear Regression & Logistic Regression

Model Hyperparameters: Can be tuned for a predictive modelling problem.

⇒ Inputs to the function which we can tune.

Ex → learning rate in neural nets.

→ C & γ in SVM

→ K in KNN

Random Search Vs grid Search for Hyperparameter tuning.

Grid Search.

Regularization:

Cross-validation:

avoid noise

- → Validation
- → Hold out Method.
- → K-fold Cross validation.
- → Stratified //

Non-exhaustive

⇒ Train loss → alter Model parameters.

⇒ Validation loss → used to tune hyperparameters, and not to alter weights

Exhaustive Methods

⇒ leave-p-out cross validation.

n-p , p

all combinations & then loss is averaged.

→ Avoid overfit & underfit of data

Hyperparameter Tuning:

① Randomised Grid search Cross-validation

→ Create a dict of all the hyper parameters & all their possible/likely values as the keys.

→ our Model choose any one combination of the above hyperparameter grid and evaluate the Model.

Regularization: Discourages learning from a very complex and very flexible Model to avoid overfitting.

Ex.

Residual Sum of Square (RSS) in Linear regression.

$$RSS = \sum_{i=1}^{n} \left(y_i - \left(\beta_0 + \beta_1 x_i \right) \right)^2$$

↳ if x has only one feature

while training we will adjust β_0 & β_1 using Gradient Descent.

But if 'x' has noise then it can't generalize well.

Here comes regularization and shrinks these learned parameters at noisy points to zero by giving a penalty in the cost function.

Therefore we update the cost function.

① Ridge Regression (L2)

no. of parameters

$$RSS' = RSS + \lambda \sum_{i=0}^{m} (\beta_i)^2$$

slopes of ~~curve~~ on different β_i's

Penalises for high value of coefficients ∴ reduce the Model complexity & Multi-Collinearity.

Underfitting →

- If we have very few features on data set & both training & test loss is high than its underfitting case.

overfitting (over-generalization)

- Large no. of features
- Low training loss & high test loss.

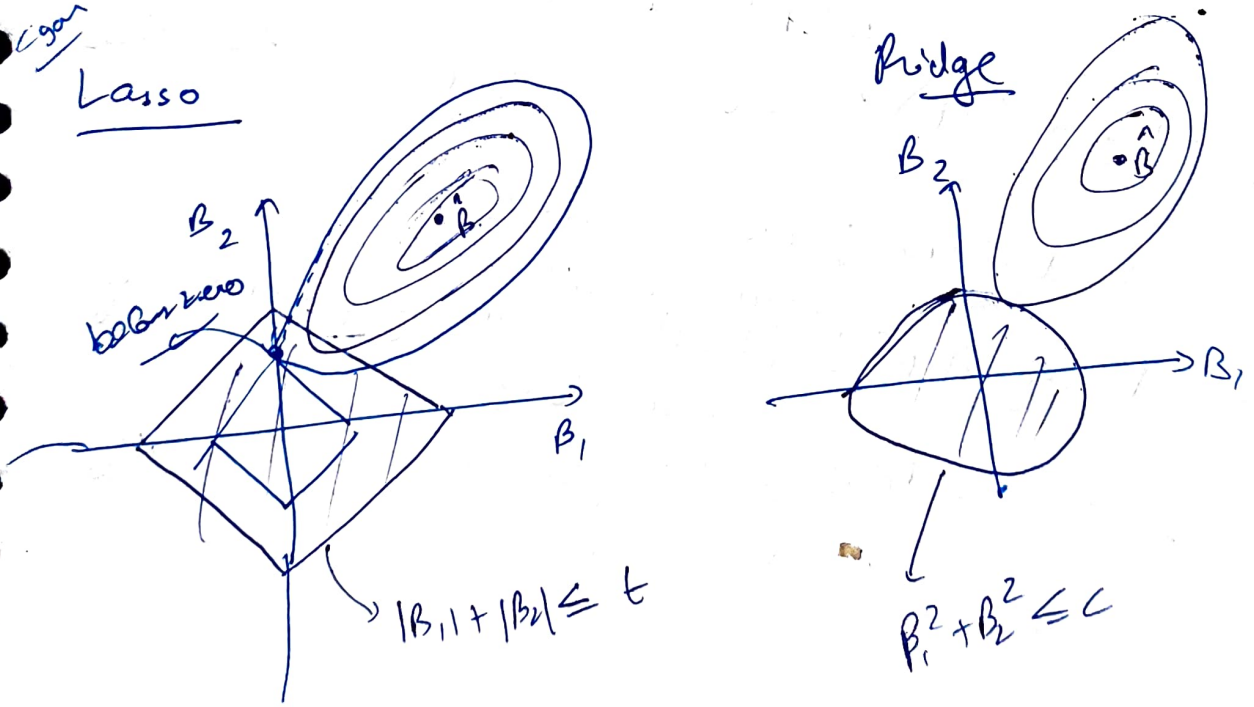
② Lasso Regression (L1 Regularization)

LAsso (least Abs. Shrinkage and selection operator)

$$RSS' = RSS + \lambda \sum_{i=0}^{n} |\beta_i|$$

→ Lasso can lead to Zeroing some of the coefficients i.e totally neglecting them.

∴ Lasso regression not only helps in over-fitting issue but also feature selection



Sem 7 08.04.2021

Performance Metrics :

Accuracy → can be decisive when dealing with unbalanced data.

Confusion Matrix, Precision, recall, f1 score gives better intution of Prediction as compared to accuracy.

Confusion Matrix :

Matrix of size 2×2 for binary classification with actual data on one axis & predicted on other.

∴ In a binary classification if class are 0, 1

Predicted \ Actual	0 Neg	1 Pos
0 Neg	True Negative	False Negative
Pos 1	False +ve	True +ve

True Positive (TP) : Model correctly predicts +ve class.

True Negative (TN) : Model correctly predicts -ve class.

False Positive (FP) : (Type (I) Error) Model gives wrong prediction of a -ve class

False Negative (FN) : Model gives wrong prediction of a +ve class.

↓

Type II error

With the help of above Matrix we can calculate True Positive rates (TPR), False Negative Rates (FNR), True Negative Rate (TNR) and false Positive Rates (FNR).

$$TPR = \frac{TP}{\text{Actual Positive}} = \frac{TP}{TP + FN}$$

$$FNR = \frac{FN}{\text{Actual Positive}} = \frac{FN}{TP + FN}$$

$$TNR = \frac{TN}{\text{Actual Negative}} = \frac{TN}{TN + FP}$$

$$FPR = \frac{FP}{\text{Actual Negative}} = \frac{FP}{TN + FP}$$

⇒ Even if our data is imbalanced we can figure out that our Model is working well or not. For that, value of TPR, TNR should be High & FNR, FPR should be as low as possible.

• Precision & Recall:

Both Precision & Recall are important for information retrival; where positive class matter the most as compared to Negative class.

→ while searching on web the model does not care about something irrelevant and not retrived (this is the TN case).

Therefore only TP, FP, FN are used in Precision & Recall

Precision: out of all +ve "predicted", what % are Truly +ve

$$\therefore \text{Precision} = \frac{TP}{TP + FP}$$

Recall: out of the total Positive, what % are Predicted Positive.

$$\text{Recall} = \frac{TP}{TP + FN}$$

same as TPR

Example: Credit fraud detection (+ve)
(when Recall is important)

Predicted \ Actual	Fair Txn	Fraud Txn
Fair Txn	TN	FN
Fraud Txn	FP	TP

In case of transaction,
- we do not want to miss any fraud transaction;

∴ we want FN to be as low as possible.

∴ Recall should be high, but we can compromise with low precision.

Example: Spam detection
(when precision is important.)

	Not Spam	Spam
Not Spam	TN	FN
Spam	FP	TP

In Spam if we get any spam as non-spam its ok, but if we miss any important msg due to spam it not at all good

→ In such case FP should be as low as possible.

∴ Precision is more used than recall.

⇒ F1-Score: When comparing two models it is hard to compare which is more precise and low recall or vice versa). ~~F beta~~

It is the harmonic Mean of Precision & recall. thereby taking both FP, FN into account ∴ performs well on imbalanced data.

$$F_1 \text{ Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

∴ gives same weightage to Precision & Recall.

Weighted F1-Score: Here we can give different weights to Precision & Recall. ~~As discussed~~

$$F_\beta = (1+\beta^2) \times \frac{\text{Precision} * \text{Recall}}{(\beta^2 * \text{Precision}) + \text{Recall}}$$

↳ weights can be given to Precision & Recall as per the Demand of Problem.

$\beta \equiv$ Represent how many times recall is more important than Precision.

⇒ Support: ~~The no. of samples of True responses that lies in that class.~~ The no. of actual occurrences of the class in the specified dataset.

$\beta \Rightarrow 1$, Precision & Recall have Equal weight

$\beta \Rightarrow 0.5 \Rightarrow$ Precision ↓

$\beta > 1 \Rightarrow$ Recall ↑

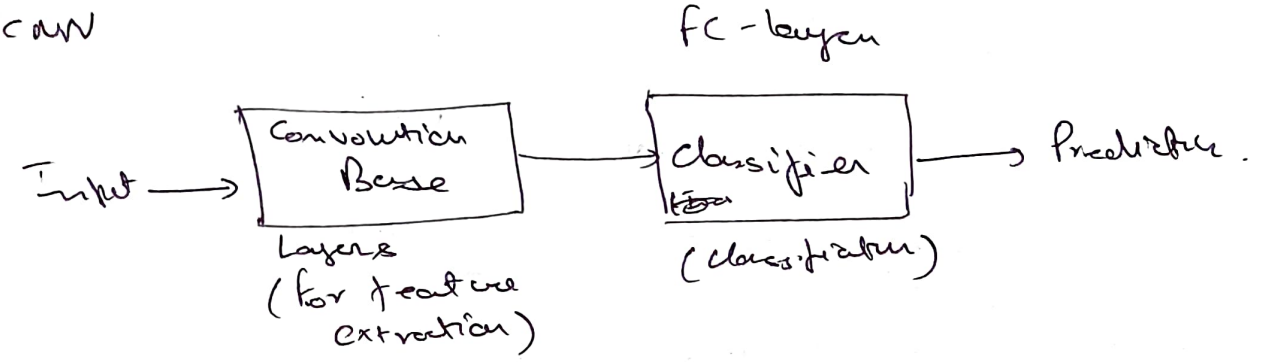
$\beta \in (0-10)$

Transfer learning from pre-trained Models

① Transfer learning → To build accurate Models in a time-saving way.

pre-trained Models : VGG, Inception, Mobile Net.

② CNN



③ Repurposing a pre-trained Model :

When we are repurposing a pre-trained model for our need, we start by removing "the original classifier". Then we add a new classifier that fits our purpose, and finally we fine tune our model according to one of the 3 strategies.

④ Train the entire Model : Train from scratch.

•) Train some layers and leave others frozen:

As we know lower layers → Problem independent ∴ they are left frozen, while the higher layers are trained also in this case we keep the learning rate is low, because for high learning rates may lead to the risk of loosing previous knowledge.

→ Also if we have small data set, & large no. of parameters we freeze more no. of lower layers to avoid over fitting.

→ If we have large data set, and small no. of model parameters we can train more layers as overfitting is not an issue

c) Freeze the Convolution base: train/freeze trade off.

Trained Model is used as a fixed feature extraction machine.

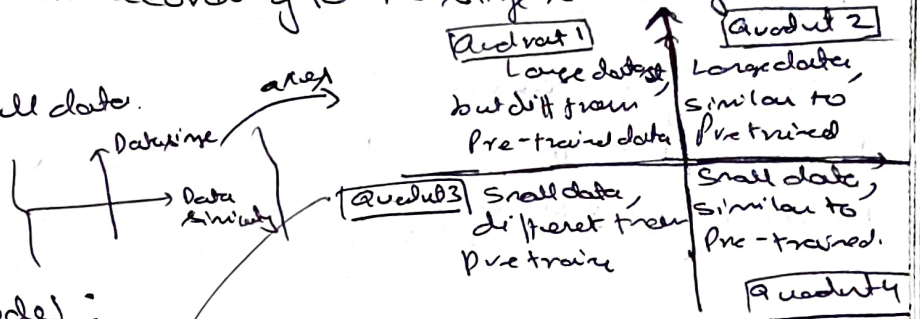
④ Transfer learning Process:

(a) Select a Pre-trained Model ⇒ here we have → VGG4, InceptionV3, ResNet5.

(b) Classify your problem according to the size-similarity Matrix".

Rough outline":

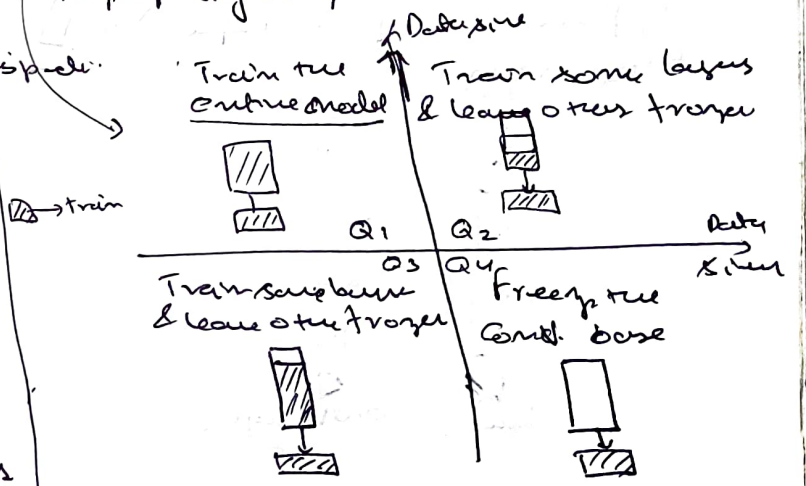
if $n < 1000$ → small data.
per class



ⓒ Fine tune the Model:

Here we can use the size similarity Matrix to guide our choice and then refer to one of the 3 options we saw in Representing a pre-trained Model.

→ ~~Quadrant~~: Quadrant



Q1: Train whole, but we can initialize the parameters by using a pre trained Model using its architecture & weight.

Q2: Here Bingo. We can Save Computation (Training) by freezing More no. of layers

Q3: "2-7 off Suite Hard of Computer with" problem. In this case we can't do ~~any~~ anything as if we "train" more layers then there will be overfitty & if we "train" less layers then our model won't learn anything useful.
(we can use data augmentation techniques in this case)
After seeing all this we must go deeper than Quadrant 2.
i.e train more than in Quadrant 2.

Q4: use strategy 3°. Freeze the Convolution base.

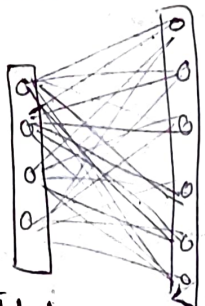
Data Augmentation:

Rocket Engine → Deep learning Models
Fuel → Amount of Data.

Ex. of Recent Models:

	VGG Net	DeepVideo	GNMT
Used for	Image classification	Video classification	Translation
Input	Image	video	English text
output	1000 classes	47 classes	French text
Parameters	140 M	100M	380M
Data size	1.2 M images with class	1.1 M videos & class	6M Source Pair 340M words
Dataset	ILSVRC-2012	Sports-1M	WMT'14

No. of parameters to train = [size(inputs) + 1] * [size(outputs) + 1]



A CNN, that can robustly classify objects even if it's placed in different orientation is said to have the property called Invariance. More specifically a CNN can be invariant to translation, viewpoint, size or illumination.

When to augment in our ML Pipline?

① Before feeding data into Model. {offline augmentation}

or

② Augment the mini-batches, just before feeding it to the machine learning Model. { online augmentation / Augmentation on fly }

Offline Augmentation	Online Augmentation
→ for small datasets	→ large datasets.

Types
- ⇒ flip (Horizontally, Vertically)
- ⇒ Rotation.
- ⇒ Scale
- ⇒ Crop
- ⇒ Translation. {Moving image in x, y direction}
- ⇒ Gaussian Noise / {Salt Noise, pepper Noise} | over-fitting usually happen when Net tries to learn high frequency feature (ND) -: Gaussian Noise reduces the quality by adding Noise.

Advanced Augmentation: {Frozen lake, glacier ex}

Adding effect of different season on the pic, day/night etc.

→ Conditional GANs | Can transform image from one domain to other} ex: Summer → winter, Night → day
↳ {high computation}

⇒ Neural Style transfer: It grabs Style Style (Texture, ambiance, appearance) of one image & flow it with Content of other. {This provides effect & variation to our Conditional GAN}

Downside ⇒ output look more artistic than realistic

↳ can be overcomed by using: Deep photo Style transfer.

Transfer learning

(5) Classifiers on top of Deep NN

(a) Fully connected layer : Mainly uses Softmax activation function } used for multiclass regression classification

(b) Global average pooling : Instead of adding fully connected layers on top of the Conv. base, we add a global avg. pooling layer & feed its output directly into softmax

(c) Linear SVMs : We can use SVM to classify the features output by the Conv. base, it can give the accuracy.

Activation function :
(or Transfer function)

⇒ used to determine the output of neuron like yes/No.

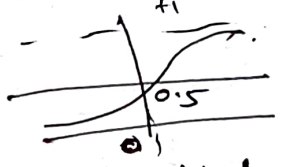
Types :

(1) Linear Activation fn
⇒ output not confined b/w layer $(-\infty, +\infty)$

(2) non-linear Activ.

(a) Sigmoid / logistic Act. fn

$$\phi = \frac{1}{1+e^{-z}}$$



hoff (0, 1)

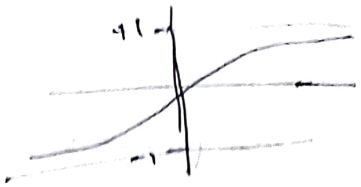
∴ gives probability output.

Softmax ⇒ Multiclass classification Activation.

(b) Tanh / hyperbolic tangent Activation fn

⇒ Tanh:

- Same like sigmoid but a bit better.
- Range (-1, +1)
- negative input → strongly -ve
  near zero → strongly near zero
  +ve input → " positive.

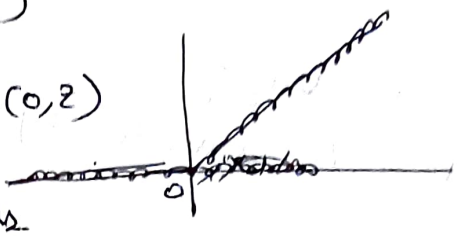


⇒ ReLU (Rectified Linear Unit)

$$R(z) = \max(0, z)$$

Range = $(0, \infty)$

- Not able to map -ve values properly.

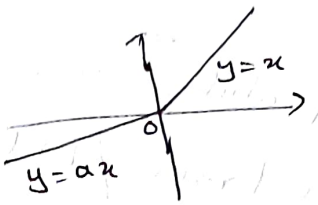


⇒ Leaky ReLU:

$a \approx 0.01$

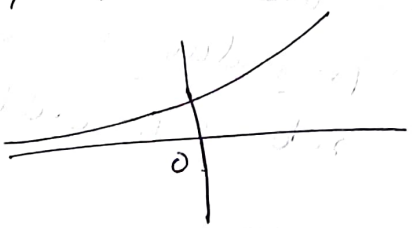
when $a != 0.01$ ⇒ Randomized ReLU.

Range = $(-\infty, +\infty)$



⇒ Soft plus

$$f = \ln(1 + e^x)$$



Sigmoid vs Softmax

→ output probabilities are independent

$\sum_i p_i \neq 1$

→ because of sigmoid looks at each new output value

⇒ Can be used for multi label classification as every classification has their own probability

$\sum_i p_i = 1$

→ inc. likelihood of one class dec. that of the other.

→ Trained with log loss or cross entropy regime.

→ Multiclass classification

Losses:

→ Loss functions

Types
① Regression losses
② classification losses.

① Regression loss:

(a) MSE / quadratic loss / L2 loss:

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

→ do not consider error direction

→ Squaring penalizes the outliers.

(b) Mean Absolute Error / L1 loss:

$$MAE = \frac{\sum |y_i - \hat{y}_i|}{n}$$

→ do not consider error direction
→ use linear programming to calculate gradients.
→ less robust to outliers.

② Mean bias Error:

$$MBE = \frac{\sum (y_i - \hat{y}_i)}{n}$$

→ Can be used to determine if model has an overall +ve or -ve bias.

② Classification loss:

(a) Hinge loss / Multi class SVM loss

→ The score of correct category must be greater than the sum of all the incorrect category by some safety margin (usually one)

→ Hinge loss is used for maximum-margin classification, for SVMs.

→ Convex function ∴ can be optimized easily.

$$\text{SVM loss} = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

(b) Cross entropy loss / Negative log likelihood:

→ inc. as predicted probability diverges from actual label.

$$\text{Cross Entropy Loss} = \underbrace{-(y_i \log(\hat{y}_i))}_{①} + \underbrace{(1-y_0)(\log(1-\hat{y}_i))}_{②}$$

(when we use 2 classes (0,1)

∴ when actual $y_i = 0 \rightarrow$ ① half disappears

& when $y_i = 1 \rightarrow$ ② half disappears.

→ Penalizes heavily for predictions that are confident but wrong. or if it is $y_i = 0$ & $\hat{y}_i = 1$

$L = 1 \times \log 0 = \infty$ or,

or

$y_i = 1, \hat{y}_0 = 0 \Rightarrow L = 1 \times \log 0 = -\infty$

Dropout:

Keras: Sequential API → Feed forward Functional API → Recurrent.

Handling Multicollinearity in data:

Explanatory variable → Independent variable (X)

Response → Dependent variable (Y)

Model interpretability is affected by several problems such as:

1. Multi-Collinearity
2. Correlation of variance of errors terms.
3. Non-linearity impact the model's interpretability.

1. Multi-collinearity:
   - → When two & or more explanatory variables are found to be closely co-related to each other this co-relation is refered to as Multi co-linearity.
   - → It affects the interpretability of linear regression models
   - → Due to this it is difficult to isolate the individual effect of explantory variable on the response variable.

It results in following:

① uncertainty of coffi (weight) estimate or unstable variance: small change in data can change the coefficient.

② Increased Standard Error: Reduce accuracy of the estimate & inc. the chances of detection.

③ Dec. Statistical Significance: Due to increased standard error, t-stat. decline which negatively impact the capability of detecting statistical significance in coffi leading to type II error.

④ Reduce coffi & p-value: The impact of the correlated explanatory variable is masked due to collinearity

⑤ overfitting: Leads to over fitting as is indicated by high variance.

Multi-collinearity detection:

→ Correlation Matrix,
→ heatmaps are definitely more intuitive & used. However. it helps to identify correlation b/w 2 variables strictly and fails to identify collinearity which exist b/w 3 or more variables, for which 'Variance Inflation factor' can be used.

Variance Inflation factor (VIF): VIF is the ratio of variance of coefficient estimate when fitting the full model divided by the variance of coefficient estimate if fit on its own. Minimum possible value is '1' which mean no-collinearity. if VIF > 5 then collinearity must be addressed.

Dealing with Multicollinearity:

① Introduce penalization or remove highly correlated variables: By Lasso / Ridge (L1/L2 Regularization) can be achieved by observing VIF values.

② Combine highly correlated variables: Using methods like PCA

~~KD Trees~~:

KD-Trees: efficient alternative of KNN

→ K-dimensional trees,

Another Alternative of KNN
→ ball tree

Decision Trees:

→ A decision tree can be used to visually and explicitly represent decisions and decision making.

→ A decision tree is drawn upside down. Internal node contains the conditions according to which the node splits into edges.

→ Non-parametric supervised learning used for both regression & classification.

Decision Tree algo Refered to CART (Classification & Regression Trees)

1) What's going on in the background?

→ It involves deciding on which feature to choose

→ What condition to use for splitting.

→ Knowing when to stop.

→ As a tree generally grows arbitrarily, we will need to trim it down for it to work beautiful.

Techniques used for splitting:

(1) Recursive Binary Splitting: In this process:

→ All the features are considered

→ Different split points are tried and tested using a cost function.

→ The split with the best (least) cost is selected.

→ Greedy Algorithm as we have an excessive desire of lowering the cost.

→ This makes the root node as best predictor / classifier

② Cost of a Split :

In both classification & regression, ~~we try to find~~ The Cost function try to find "most "homogenous branches", or branches having groups with "similar "responses" (outputs) that

Regression : $\text{Sum}(y - \text{prediction}(\hat{y}))^2$

Decision Trees → moderate : for nonlinear decision making with simple linear decision surface.

Classification : $\text{Entropy}(E) = \sum_{\forall i=\text{class}} -p_i \log_2(p_i)$ → probab. of i^{th} class in the split.
(Gini Index)

$\text{Information Gain} = E(\text{Parent}) - \sum w_i E(\text{Child}_i)$

Model Compares every possible split & ~~Trees~~ Choose those that Maximise the IG (Info. Gain)

Regression : To check impurity use "Variance Reduction" technique.
"Normalized Need YT"

$$\text{Var} = \frac{1}{n} \sum (y_i - \bar{y})^2$$

$\text{Variance Reduction} = \text{Var}(\text{Parent}) - \sum w_i \text{Var}(\text{child})$

↳ Higher is preferred :) → ∴ dec. impurity more.

Random forest classifier:

(Bi-ary classifier)

⇒ Why we use Random forest when we have D-trees.?

•) If there is a slight change in data then the whole decision tree needs to be rebuilt.

∴ Highly sensitive to training data, which shows high variance & overfitting. And we need to generalize our model.

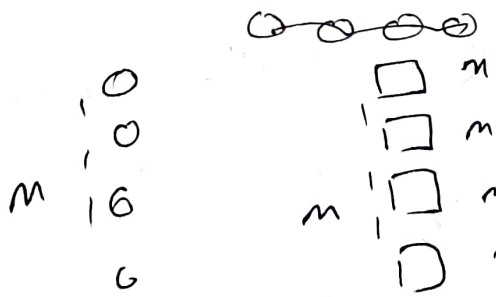
⇒ Random forest ⇒ Is a collection of multiple Random Decision trees & therefore is much less sensitive to the training data.

Step ① (Bootstrapping) ⇒ Bild new data sets from our original data. randomly selecting rows ~~from~~ with replacement, to make a new Dataset.

Ex: If our original data has n rows. then we will create random data with n rows & select rows with replace.

∴ No. of such random trees $= n \times n \times n \times n \cdots n$

$= n^n$



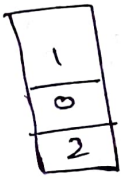
for building tree Colums we do not choose all → Random feature selection
the Colums we take different set of Colum at a time

Ex: original data

New data:

x_1, x_2

or



$x_2 x_2$ - - -

∴ Total possible such data $= (n^m \times 2^c)$

n^m → no. of rows, 2^c → no. of Colum

Step 2°: Make tree for all such data.

Step 3°: we pass the data to be predicted through each tree one by one and make the prediction.

Step 4°: we combine all the prediction by a Majority voting.

↳ Aggregation step.

Result = (BootStrapping + Aggregation) = Bagging

Random feature selection step°: Helps to decrease effect of Co-relation b/w the features.

Ideal size to select the no. of features (Colus) to take into account = log (Total features) OR $\sqrt{\text{Total features}}$

Now Random forest for regression problem:

In Step 4: while combining the prediction just take average.

Dealing with Imbalanced data:

(1) Change Performance Metric: Accuracy can be very misleading as in such cases dummy classifier can also give a high accuracy.

(2) Change the algo: Use Decision tree

(3) Resampling Techniques (a) over sample Minority class.

(b) undersample Majority class

(4) Generating Synthetic Samples:

imblearn's SMOTE (Synthetic Minority Oversampling Technique)

Use Nearest neighbour to generate new and Synthetic data we can use for training our model.

Categorical Encoding: Changing non-numerical data to numeric

(1) Label Encoding

(2) One Hot Encoding.

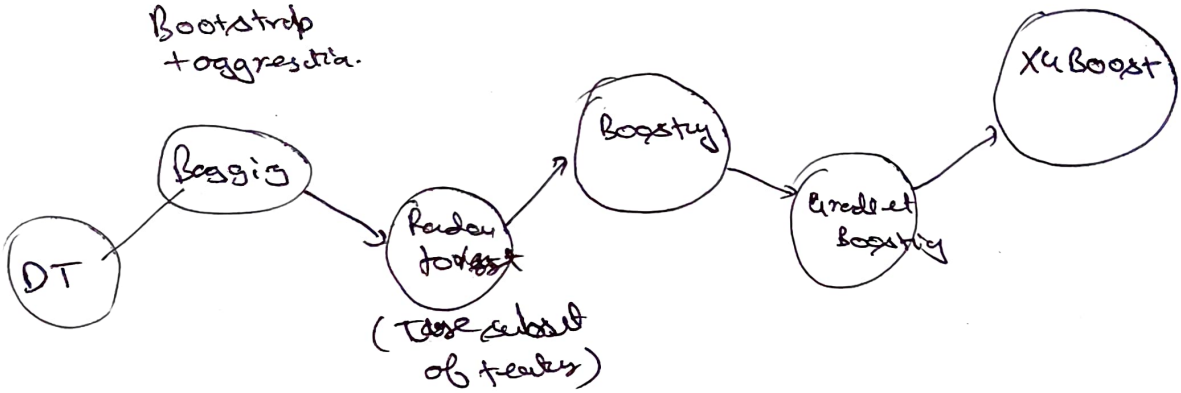
XGBoost Algorithm:

(Extreme Gradient Boosting)

} others are → LightGBM, CatBoost

→ Decision tree-based ML algo. that uses gradient boosting framework.

Evolution of Decision Trees:



Boosting: Models are built sequentially by minimizing the errors from previous model while increasing (or boosting) influence of high performing models.

Gradient Boosting: Gradient Boosting employs gradient descent algo to minimize errors in sequential models.

XGBoost: Optimized Gradient Boosting algo through parallel processing, tree pruning, handling missing values and regularization to avoid overfitting/bias.

Bias-Variance Trade off:

→ Model interpretability (long assumption) vs Model flexibility (few assumptions)

ex: linear

ex: polynomial.

Bias: Error introduced due to assumptions made to model a complex real-life problem by an approximately simple model.

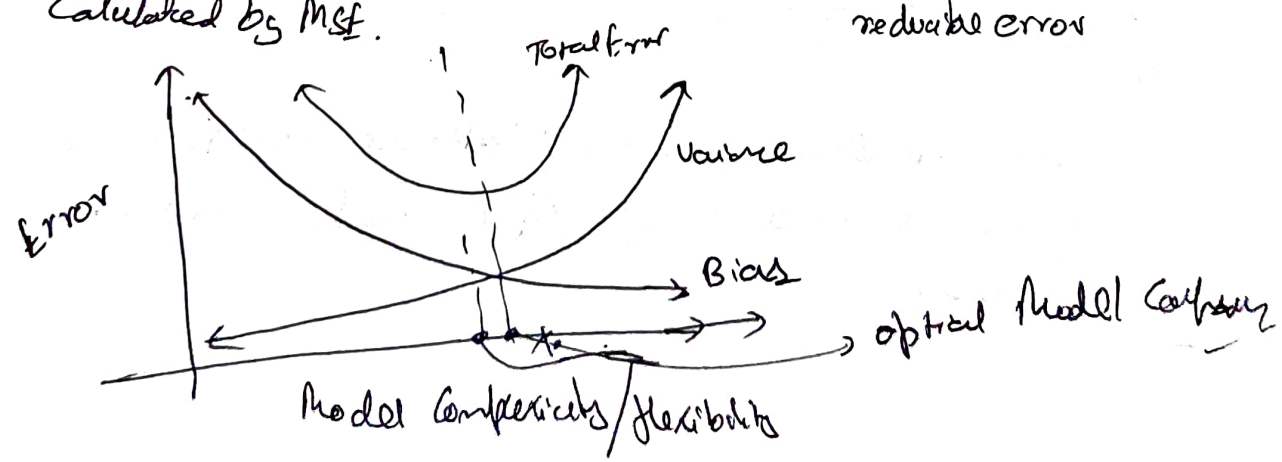
→ More flexible models take less assumption about the data result is less bias. However they might lead to ~~underfitting~~ overfitting.?

Variance: Amount by which the estimates or results of the ML model change with a different training dataset. More flexible model has more variance. → leads to overfitting.

Trade off

Expected Test Error = (Irreducible Error due to noise) + (Bias^2 + Variance) ← reducible error

↳ Calculated by MSE.



Ways to reduce Bias & Variance:

- Bias can be reduced by inc. Model complexity by
  - → Boosting
  - → Add more features or introduce polynomial feature.

Boosting:

→ unlike many ML models which focus on high quality prediction done by a single model, boosting algo seeks to improve the prediction power by training a sequence of weak models, each compensating the weakness of its predecessors.

→ It is generic

Types: Based on different definition of weakness and their correspond algorithm. Some are →

① Adaptive Boosting (AdaBoost) ② Gradient Boosting.

① Adaptive Boosting (AdaBoost):

→ for classification Problems & Regression.

→ Weakness is identified by the weak estimator's error rate.

for a weak classifier C:

$X: n \times d$; $Y: n \times k$, Sample weights $W: n \times 1$

(d = no. of features; k = no. of classes)

$$\text{Error} = \frac{\sum_{j=1}^{n} w_j I(C(X_j) \neq Y_j)}{\sum_{j=1}^{n} w_j}$$

$$I(x) = \begin{cases} 1, & \text{if } x = \text{True} \\ 0, & \text{if } x = \text{false} \end{cases}$$

→ The weak learners in AdaBoost are Decision Trees with a single split, called decision Stumps.

→ Adaboost works by putting more weights on difficult to classify instances (features) & less on those already handled well.

[*Ensemble : ML techniques that combines several base models in order to produce one optimal predictive model]

→ Does same splits like that in Decision Trees.

Each stump choose a feature X_i & a threshold T_{ij} & checks checks for the condition on (X_i with T_{ij}) → Node

→ This can be tried out for all the features & all possible Threshold look for the best accuracy or the Information Gain.

→ While it naively seems like there are ∞ no. of checks for threshold, two diff. Thresholds are meaningful if it divides the data in diff. side of split.

→ To try every possibility, then we can sort the examples by feature in the question & try one threshold halfway b/w each adjacent pair of examples.

(Although the Algo is fast if it tries all possible T's).

⇒ Adaboost → forest of Stumps.

D-Trees	Random Forest	Adaboost
Takes all features to make the best possible tree	→ Manipulates data & selects features to be used to make many trees	→ only Decision Stumps are used ie (one feature per tree)
→ Tree is Made	→ All trees are given equal weight with calculate output.	→ diff weights are given to different stumps. → all the Stumps are combined with diff weights for output

→ In Adaboost order of the trees (stumps) matter. because the error that the prev. stump make influence how the next stump is made.

→ all stumps get the same weight at the start i.e $\left(\frac{1}{m}\right)$, & changes as the other stumps gets created.
↳ dataset

→ Total error of stump. = Sum of weights associated with the incorrectly classified sample.
(lies in [0, 1])

$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$

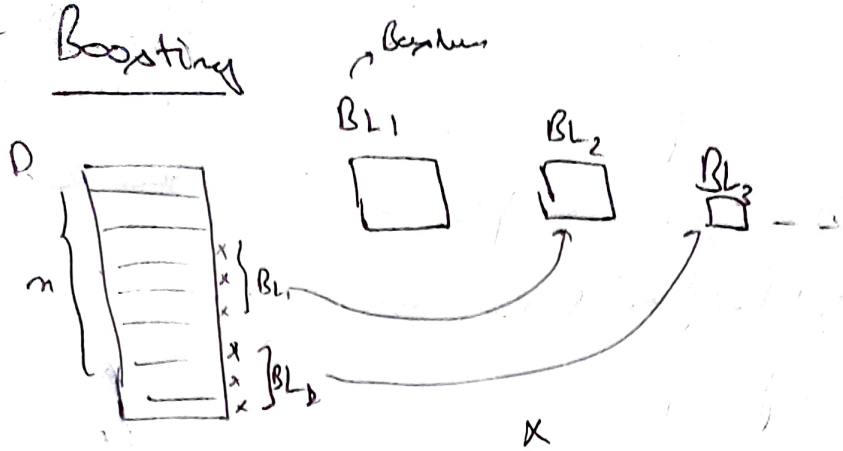
→ New sample weight for the sample that incorrectly classified = sample weight $* e^{(\text{amt of say})}$

→ New sample weight (for the sample that correctly classified) = sample weight $* e^{-\text{amt of say}}$
~~to classify~~ of samples that correctly classified)

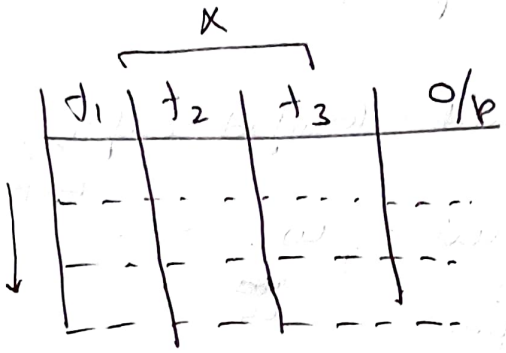
→ Now Normalize the new sample weights.

See StatQuest Adaboost

Boosting



AdaBoost :



Sample wt.

y_n

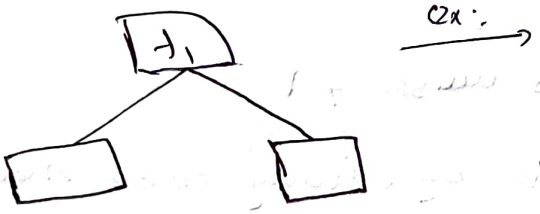
y_n

y_n

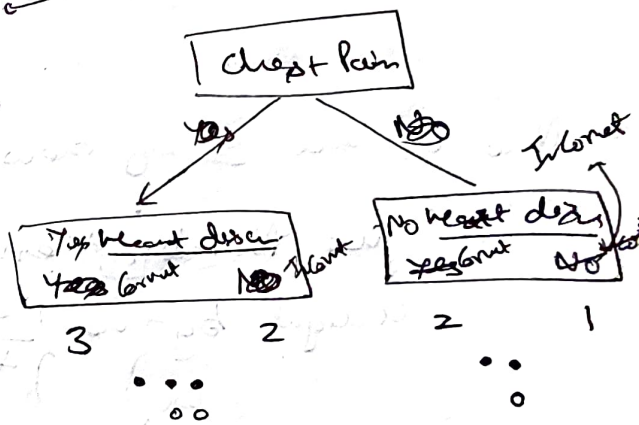
initially $\Rightarrow w = y_n$

Low Entropy stump is selected.

Let it be 'f_1'



ex:



Total Error = Sum of weights associated with incorrectly classified samples

(TE) were = $\left\{2 \times \frac{1}{n} + 1 \times \frac{1}{n}\right\}$

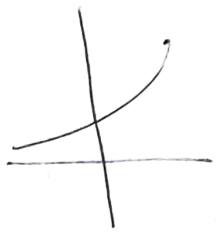
Amount of say :

Performance of stump = $\frac{1}{2} \log\left(\frac{1-TE}{TE}\right)$

Now as the wrongly classified records pass to the next ~~base~~ Stump. ∴ we inc. the wts of wrongly classified sample & dec. the wt. for Correctly classified ones. using the "Performance of Stump" = P

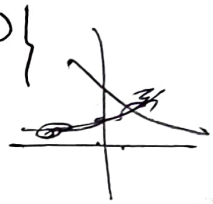
① Inc. wt of InCorrect classified samples,

$$\text{New sample wt. for InCorrectly classified} = \{ \text{wt. added wt.} \times e^{\text{performance}(P)} \}$$



② Dec. the wt of Correctly classified samples:

$$\text{New Sample wt. of Correctly classified} = \{ \text{wt.} \times e^{-\text{Perform}(P)} \}$$



⇒ Now as the Σ of new weights $\neq 1$

∴ we Normalize them by dividing each new weight by the $\left\{\frac{\Sigma}{t} \text{wt}\right\}$

$\{t = \Sigma wt\}$

i.e $\left\{ w_t = \frac{wt}{\Sigma t} \right\}$

⇒ Now new dataset with n record (with replacement) is created for next stump randomly based on the weights of the sample (the wrong records have higher wts)

⇒ & Continue the same steps till be achieve purity or, tolerance.

Done.

⇒ Aggregate the result of all the Stumps. : Majority Voted.

Gradient Boosting

Ex:

Exp.	Deg.	Salary	$\hat{y}^2$	R_1	R_2
2	BE	50	75		
3	MS	70	75		
5	MT	80	75		
6	PhD	100	75		

Base Model

1.) for a regression & classification:

1) Regression:

Step 1: Base Model 1 → Avg. of the output

Step 2: we Compute Residuals Errors, Pseudo Residuals.

Calculate loss or Residuals. (R_1)

Step 3: Construct DecisionTree { x_i, R_1 }

x_i → feature name

R_1 → output

will

The decision tree outputs the Residual from the given input features.

∴ If we again pass all the data sample features in the Decision tree it will give Residuals R_2 {which will be near equal to R_1}

Step 4: Final prediction value is calculated by doing.

$\hat{y}$ ~~=~~ $\hat{y}$.

$\{\hat{y} - R_2\}$

$\hat{y}$ → Base Model value

R_2 → DT-Residuals.

* But this is an overfitting problem

which will have low bias (because of overfitting)
& high variance (Bcoz of overfitting).

→ what we do is that we add a learning rate to this

∴ we do. $\left[\hat{y} + \alpha(R_2)\right]$ learning rate

$\{\alpha \in (0,1)\}$

∴ in our example. α

$= 75 + 0.1(-23)$

$R_2 = 73.7$

⇒ Now we create a new Decision Tree based on the new R_2 value keeping features same.

∴ what we can do:

$$f(x) = \underbrace{h_0(x)}_{\text{base}} + \overbrace{\alpha_1 h_1(x)}^{DT_1} + \overbrace{\alpha_2 h_2(x)}^{DT_2} - - + \alpha_m h_m(x)$$

$$\Rightarrow \boxed{f(x) = h_0(x) + \sum_{i=1}^{m} \alpha_i h_i(x)}$$

XGboost : Extreme Gradient Boosting

Regression -

Ex: Get loan or not → Bad, Good, None.

Salary	Credit	Approved Loan (y)	$\hat{y}$	R_1
<=50K	B	0	0.5	-0.5
<=50K	G	1	0.5	0.5
<=50K	G	1	0.5	0.5
>50K	B	0	0.5	-0.5
>50K	N	1	0.5	0.5

→ used for both binary & multi-class classification.

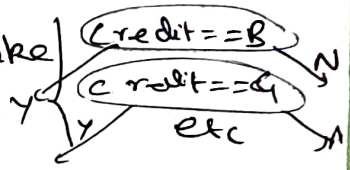
Step 1: Base Model → Binary classification ∴ take $\hat{y} = \frac{0+1}{2} = 0.5$

$Pr = 0.5$ (Most of y)

$$R_1 (\text{Residual}) = y - \hat{y}$$

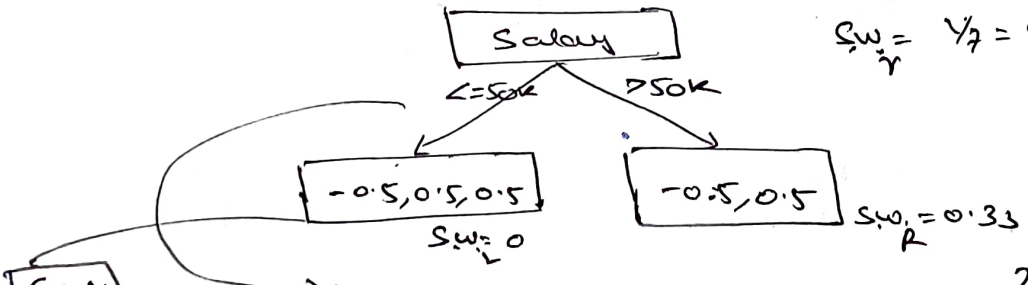
Now Construct a tree with Residual R_1

⇒ In XGboost the node always gets divided into two classes (even if there are 3 classes) like



Ex:

$$R_1 \rightarrow [-0.5, 0.5, 0.5, -0.5, 0.5]$$



$S.W_L = 0$

$S.W_R = 0.33$

$S.W_{root} = 0.14$

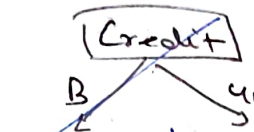
$$\text{Similarity weight} := \frac{\left[\sum \text{Residuals}\right]^2}{\sum (Pr(1-Pr)) + \lambda}$$

λ → hyperparameter

$$\text{Gain} = S.W_{Left} + S.W_{Right} - S.W_{root.} = 0 + 0.33 - 0.14 = 0.12$$

→ do the above recursively every time taking the of highest gain condition.

*) Post Pruning:

In case of 

where had more than types of values. we have to decide after the we should divide or not.

Calculated by using 'Cover value' => $P_r(1-P_r)$

Now we check if gain of Node,

Gain < Cover value, then we just cut the branch.

to calculate,

Base Model output w.r.t the Initial Probability (P_0)

$$\log(odds) = \log\left(\frac{P_{i-1}}{1-P_{i-1}}\right)$$

Now for New probability predictions, we traverse the Binary tree & find the similarity wt. or coincided to it sw.

then, the our New probability will be.

$$P_i = \sigma\left[\log(odds) + \sum_{i} \alpha_i (sw)\right]$$

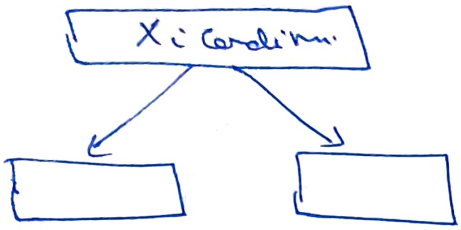
Sigmoid $\left\{\frac{1}{1+e^{-x}}\right\}$ ← learning rate

→ Now repeat the whole process again with these New probabilities, with

XGBoost Regressor:

Base Model → $R_i = y_i - Avg(y_i)$

Make Tree (Binary only)



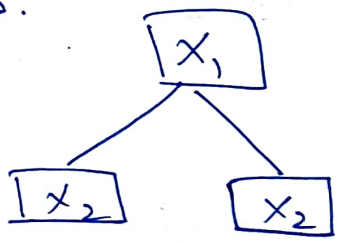
X_1	X_2	Y	R_1

$$\text{Fitness wt.} = \frac{[\sum (\text{Residual})]^2}{\text{No. of residuals} + (\lambda)}$$ → hyperparameter

$$\{\text{Gain} = S.W_{Left} + S.W._{Right} - S.W._{root}\}$$

Take "Maximum Gain" Condition

→ Keep doing this recursively to fit all the maximum gain features.



To Predict the Y,

we traverse the tree and reach the right leaf.

The T_i = Avg of all the values present in that leaf.

$$\text{output } (\hat{y}_i) = \left(\text{Base Model Value}\right) + \sum_{\text{if trees}} (\alpha_i T_i)$$

→ Repeate the same process until $\{Y = \hat{y}_i\}$

⇒ γ (gamma parameter) { Reduce over fitting (hyper parameter) }

↳ used for post pruning.

Cover value = (~~γ - Gain~~) Gain − γ

if {Cover value < 0} → then we can post prune the tree branch i.e cut it.

CNN revisited :

Layer 1 : Vertical Edge detector ⇒ (3×3)

1	0	−1
2	0	−2
1	0	−1

without padding : $\left(\frac{n-f}{s}\right)+1 = L$ (size) → next layer

(s: stride, f: filter size)

with padding : $n' - f + 1 = n$

~~what~~ we want → $n' = n + f - 1$

→ zero padding

→ ~~mean~~ nearest value padding.

P = padding dimension

∴ ~~no~~ next layer size = $\{n + 2p - f + 1\}$

$\left(\frac{n-f}{s} + 2p + 1\right)$ (s: stride)

MaxPooling : Based on "location Invariant"

↳ captures higher pixel values to refine the image.

Other Pools ⇒ MeanPooling, Min/AvgPooling

CNN → Do dimensionality reduction & keeping important features.

F_β Metrics :

i) $F_\beta = \frac{(1+\beta^2)\ Precision * Recall}{\beta^2 * Precision + Recall}$

i) $\beta = 1$ F_1 score $= \frac{2\ Precision * Recall}{Precision + Recall}$ {Harmonic Mean}

$\beta = 0.5$

$\beta = 1$ { when FP & FN both are equally important

i) FP important $>$ FN → $\beta \in [0, 1]$ ⇒ reduce β ⇒ $\beta = 0.5$

i) FP $<$ FN important ⇒ inc β ⇒ $\beta = 2$

↳ $\beta \in [1, 10]$

ROC & AUC Curves

$TPR = \frac{TP}{TP+FN}$

FPR

→ Binary classification.

→ used for deciding the threshold probability { Can alter the threshold other than 0.5 as per requirement }

Threshold value = [0, 0.2, 0.4, 0.6, 0.8, 1]

y	$\hat{y}$ (probability)	$\hat{y}(0)$ i.e. $P \geq 0 \rightarrow 1$	$\hat{y}(0.2)$
1	0.8	1	1
0	0.96	1	1
1	0.4	1	1
1	0.3	1	1
0	0.2	1	0
1	0.7	1	1

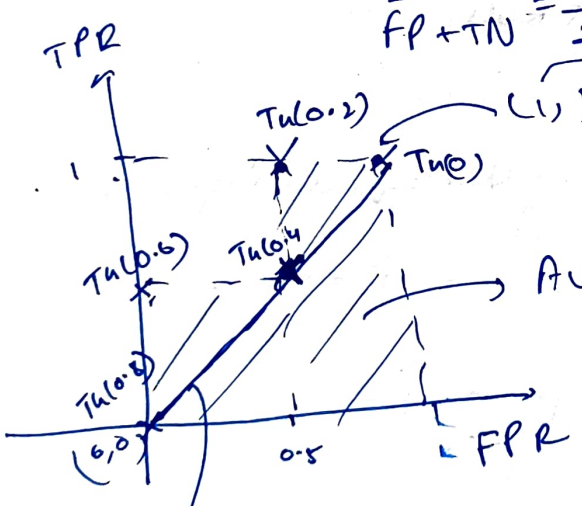
$TPR = \frac{TP}{TP+FN} = \frac{4}{4+0} = 1$

$FPR = \frac{FP}{FP+TN} = \frac{2}{2+0} = 1$

(1, 1)

$TPR = 1$

$FPR = 0.5$



AUC = Area under the curve

More the area better our model.

→ good models area must be atleast area under this line because coin tossing would be better than this. if it Prob < 0.5

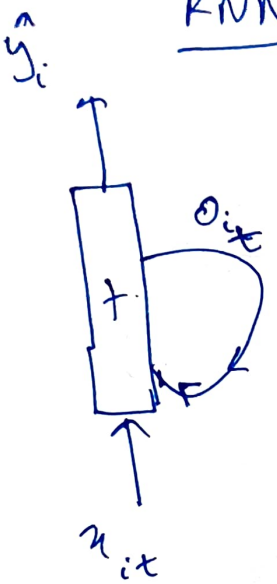
Th(0.6) → 0 FPR,

⇒ Now if we want Low FPR & high (TPR) we can select Th(0.6) in graph.

⇒ If we don't care about FPR & just want high TPR we can select Th(0.2)

⇒ If we want to focus on TPR & FPR both then we can select Th(0.4)

RNN:

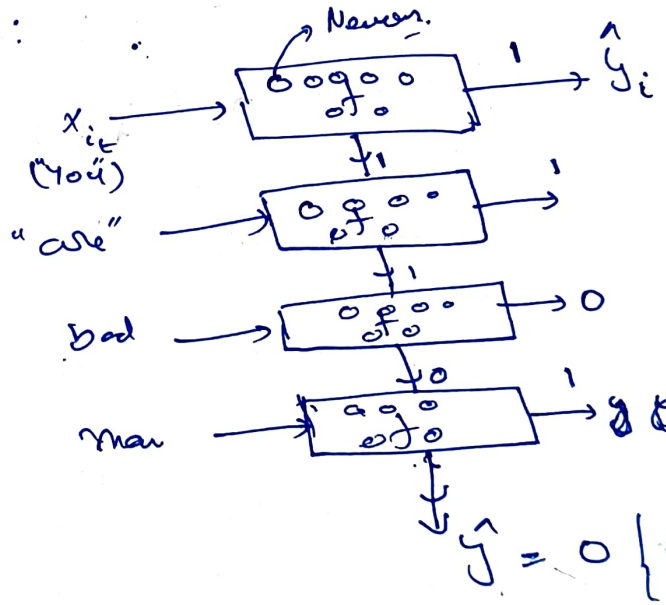


$\therefore O_i = f(x_i \times w)$ weight

"You are bad man"

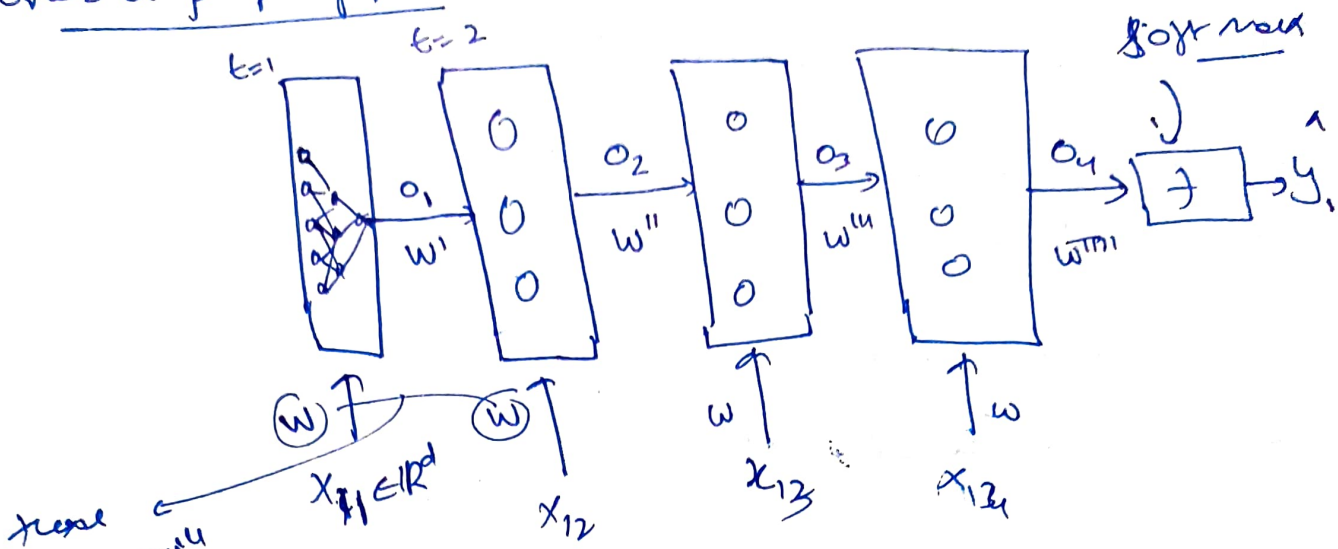
Sentiment Analysis: 0 → Neg, 1 → +ve

ex:



Let say we start with weights "W"

forward prop. of RNN:



these weights will be same and only change during "Back propagation".

activation func.

$\therefore \quad o_1 = f(x_{11} * w)$

~~$o_2 = f(x_{12} * (w + w'))$~~

$o_2 = f(x_{12} * w + o_1 * w')$

$o_3 = f(x_{13} * w + o_2 * w'')$

$o_4 = f(x_{14} * w + o_3 * w''')$

$\hat{y}_1 = f(o_4 * w'''')$

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

Next reduce the loss in back prop

Back Prop. of RNN:

$L = \text{loss} = (\hat{y} - y)$

$$\frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial L}{\partial w''''} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial w''''}$$

$$\left\{ w''''_{new} = w'''' - \frac{\partial L}{\partial w''''} \right\} \rightarrow \text{updation}$$

$$\left\{ \frac{\partial L}{\partial w} = \frac{\partial L}{\partial w''''} \cdot \frac{\partial w''''}{\partial o_4 \cdot \partial o_2} \right\}$$

!

Problem in simple RNN:

* # Vanishing & Exploding Gradient Problem -

⇒ Vanishing Gradient Problem: when during back propogation we find the derivative of functions like "Sigmoid" the derivative lies b/w [0, 1] but as we go from last layer to the first layer & back propogating the value of the derivate keeps calculated again and again becomes very small: Ex: $L_1 \; L_2 \; L_3 \rightarrow$ loss $= (\hat{y}_i - y)$

$w_1 \; w_2 \; w_3$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L_3}{\partial w_3} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial w_3}$$

$$\frac{\partial L_3}{\partial w_2} = \frac{\partial L_3}{\partial w_3} \cdot \frac{\partial w_3}{\partial (L_2)}$$

∴ due to this our step size will continuously dec. ∴ we will not be able to reach global minimum. This is called Vanishing gradient Problem.

Exploding Gradient Problem:

if we use activation function like 'Relu'
then derivative (Relu) > 1

∴ The inc. the ∂ gradient at every step back which creates an exploding gradient problem

To avoid this we use LSTM (Long Short term Memory)

LSTM:

(Gated RNN)

① Memory cell
② Forget gate
③ I/P Gate
④ O/P Gate


Neural Network layer (MLP)

○ Point wise operation

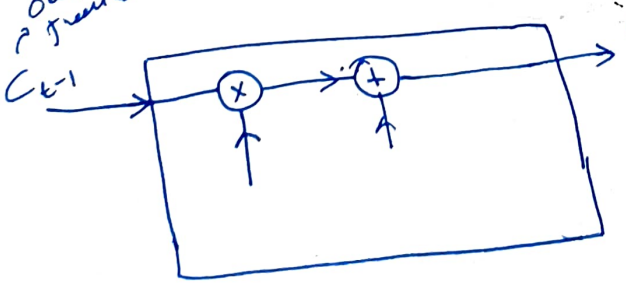
→ Vector transfer

→ Concatenate

→ Copy

① **Memory cell**: Used to remember & forget things based on content of Input.

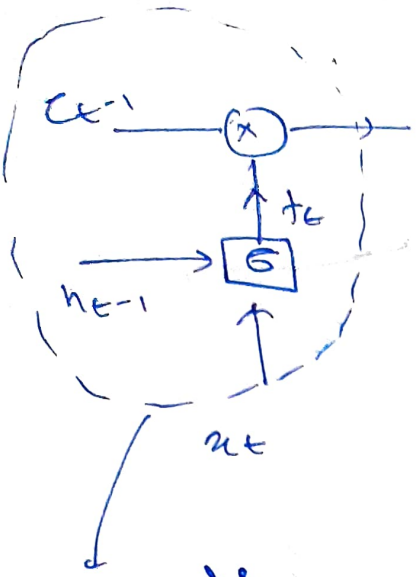
output of previous layer Also add New Info. i.e Memory.



$$\begin{bmatrix}1\\2\\3\end{bmatrix} \otimes \begin{bmatrix}1\\0\\1\end{bmatrix} = \begin{bmatrix}1\\0\\3\end{bmatrix}$$ → forgot '2'

∴ ⊗ → forget gate

⊕ → Add new information.



"forget gate"

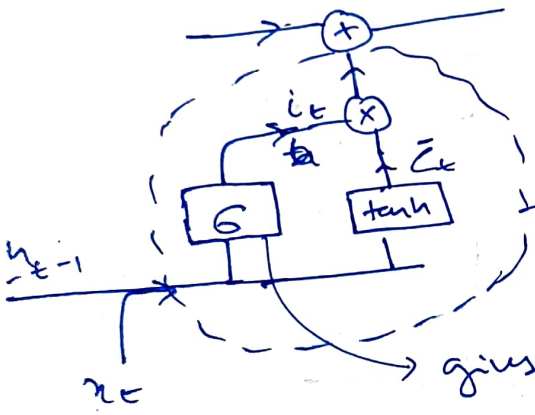
sigmoid gives (0 to 1)

bias

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

if h_{t-1} & x_t have change in context then f_t will have more zeros so that the previous info or the (x) point gets forgotten & vice versa something

---



tanh → gives (−1 to +1)

→ "Input layer"

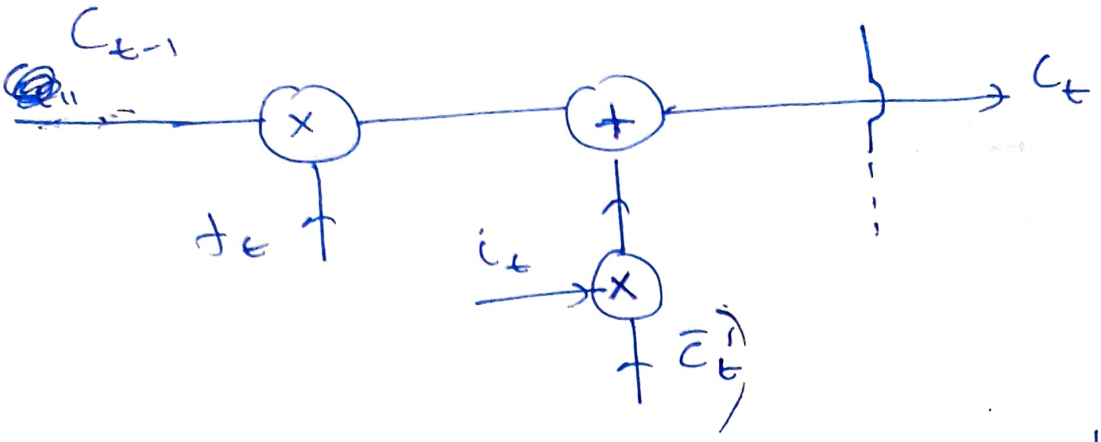
gives same things as above,

~~but what tanh does is that~~ it here 'tanh' tries to add some more information.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

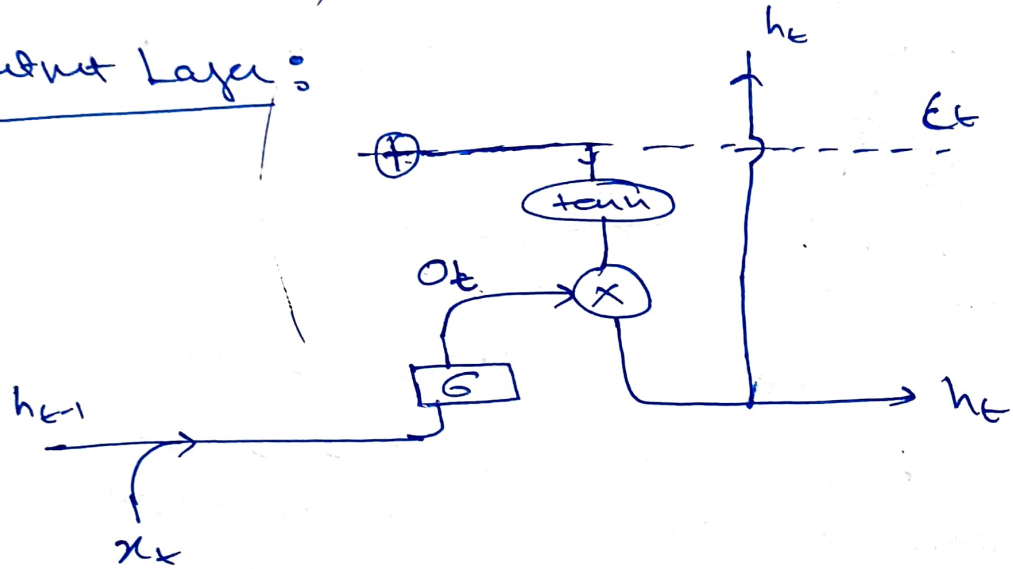
$$\bar{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t) + b_C)$$

"tanh" here acts like the Maxpooling layer in CNN.



$$\{ C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \}$$

Output Layer:



$$O_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

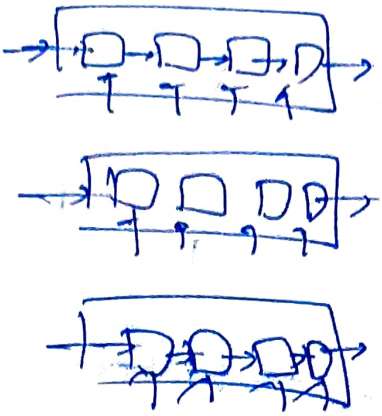
$$h_t = O_t * \tanh(C_t)$$

$\therefore h_t \rightarrow$ will have more meaningful content than that of C_t but to one more point wise operation.

LSTM Channels in NN

#Types of RNN

1. One to one
2. One to Many
3. Many to one
4. Many to Many.



Algorithmic Trading

M ① NN-based Regressor model.

M ② RNN Model Regression. (LSTM)

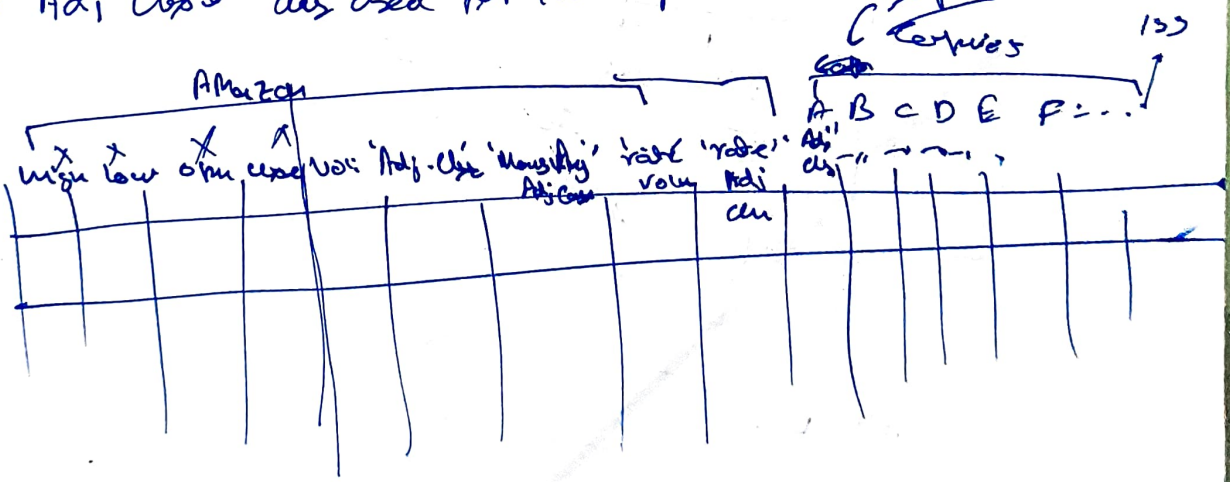
M ③ Takes both the above predictors & gives the output.

S&P 500 Companies: Data to predict the stocks of Amazon & how it affects it. Other companies affect it.

→ BeautifulSoup (Data Scraping)

Using 199 Company stock data to predict the 200th stock

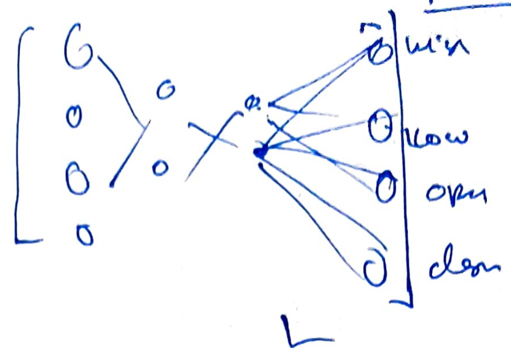
→ 'Adj Close' was used for 199 Company



~~Amazon~~ Amazon

Model ①

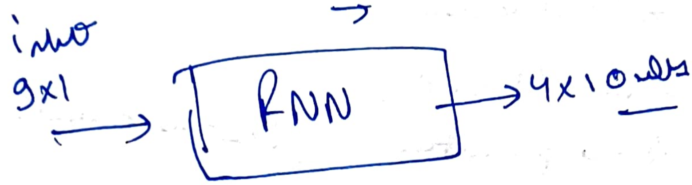
200 → Adj close of 200 days



Amazon

Model ②: RNN:

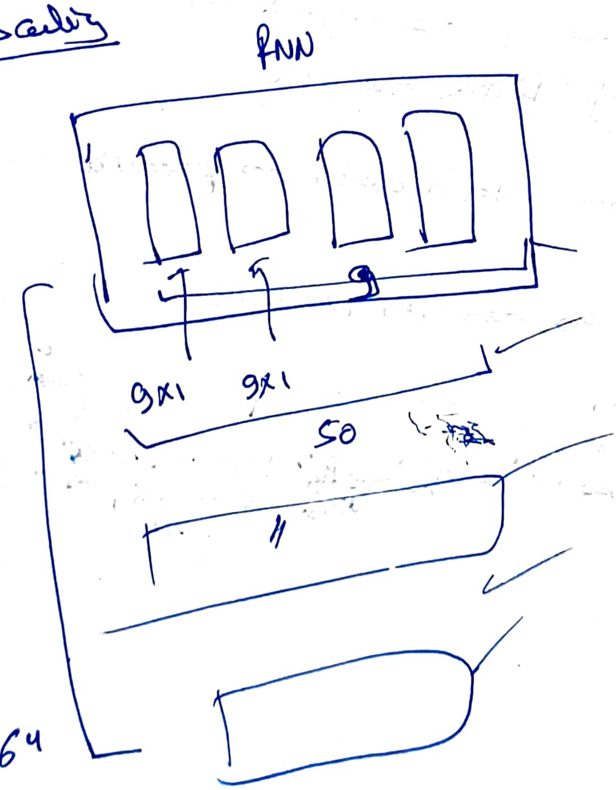
max low open close vol. Apple ... Many to Many



Many to Many RNN.

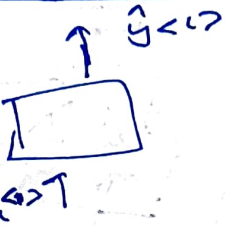
Make sets of 50 data entries.

Do scaling

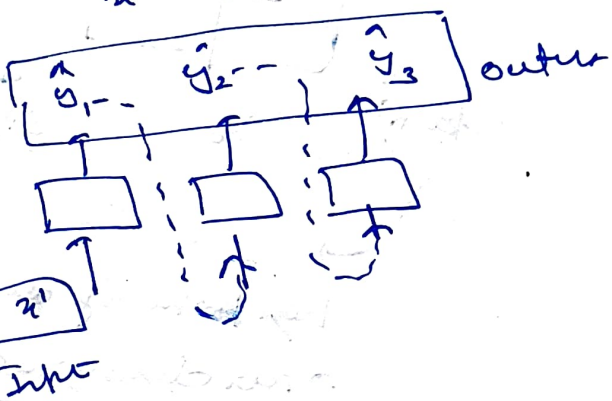


RNN:

① One to one:

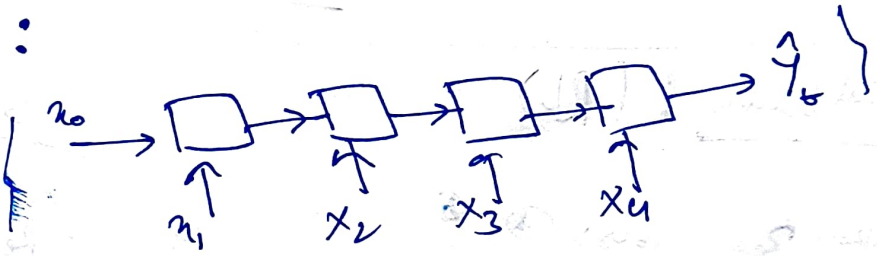


② One to Many:

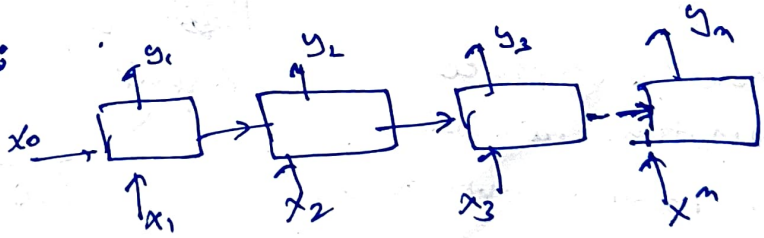


Ex:: Music generation from single note

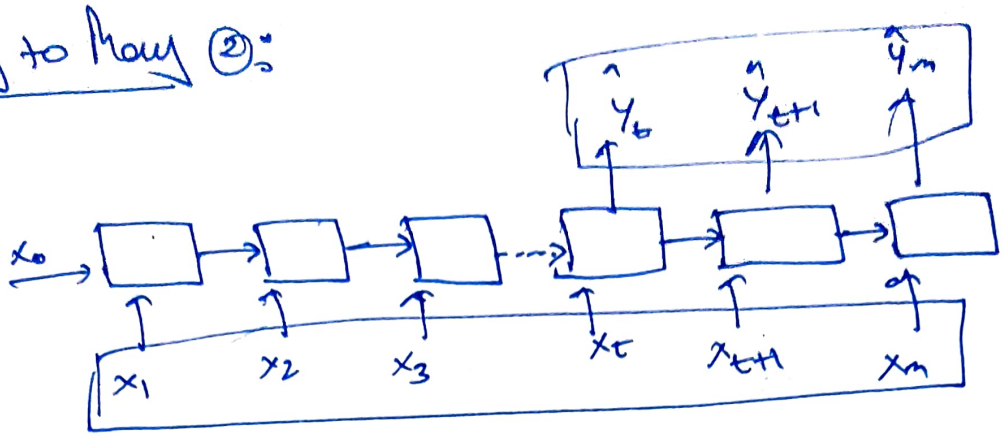
③ Many to one:



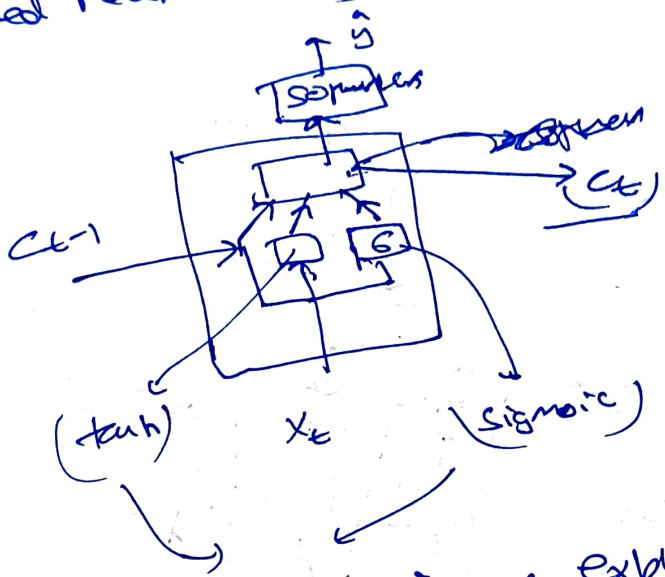
④ Many to Many ①:



Many to Many ②:

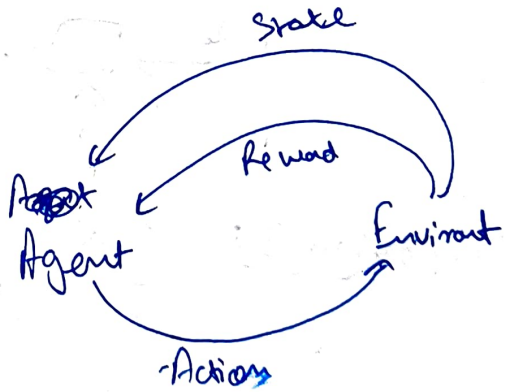


GRU (Gated Recurrent Unit)



May both if one explodes then other diminishes.

Reinforcement learning: (RL)



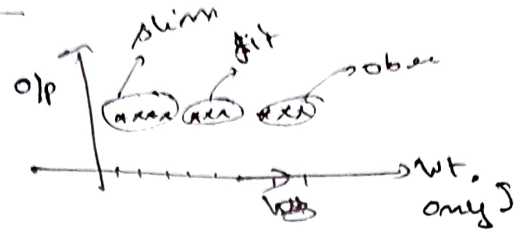
→ Markov Decision Process

→ learns from its experience and over time will be able to identify correct action that lead to the best reward

Fresh Week

Univariate, Bivariate, Multivariate analysis :

Univariate → Involve one feature



| Height | wt. | O/P | (obese, slim, fit)

Bivariate → Both height, wt.

Non linear type

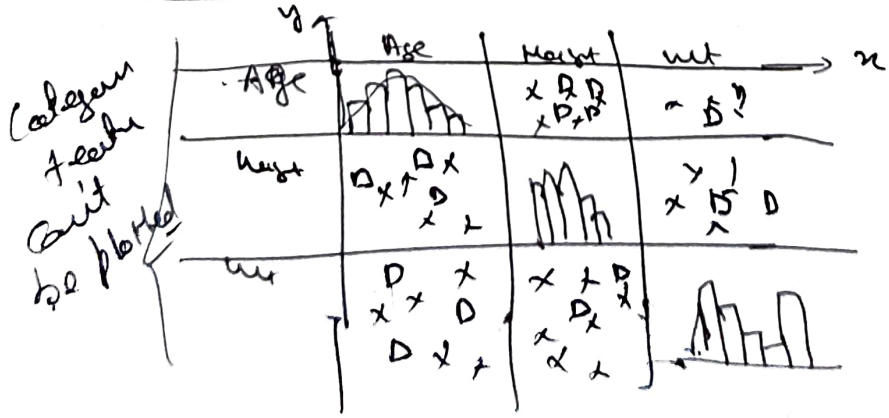
∴ Logistic can be used.

If we have 4 features : Multivariate analysis.

| DOB | Age | Height | wt | O/P |

4 feat → DOB, Age, Height, wt.

Seaborn 'pair plot' ⇒ Can be used for multivariate analysis.

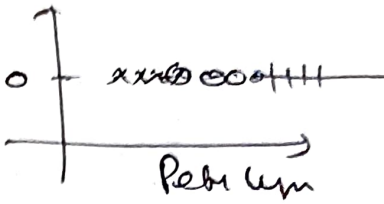


Categorical feature can't be plotted

P ⇒ Correlation

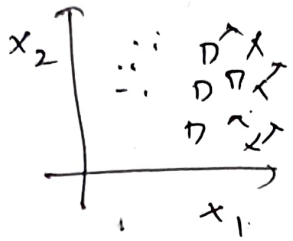
Ex: Sepal_length, Sepal_width, Petal_length, Petal_width, Species

In Classification → Univariate analysis can do $y=0$ for all & change color for diff. classes.



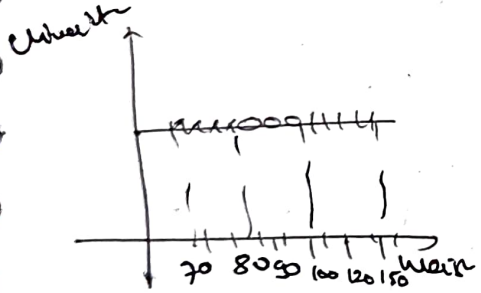
Bivariate Analysis

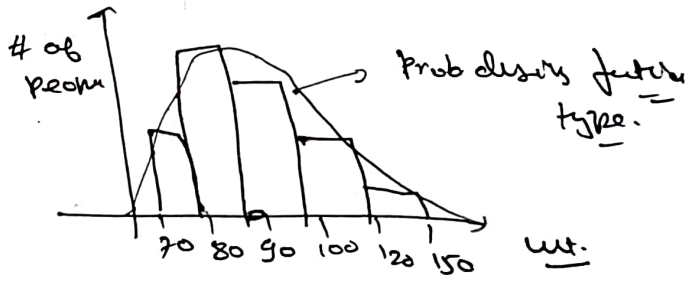
Seaborn → FacetGrid (data, hue = "species")



Multivariate Analysis: Pair plot → x_1, x_1 → plot

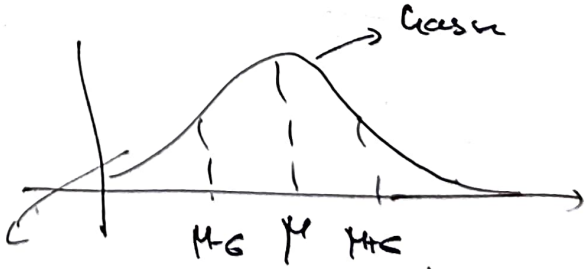
Histogram: No. of points in particular category.



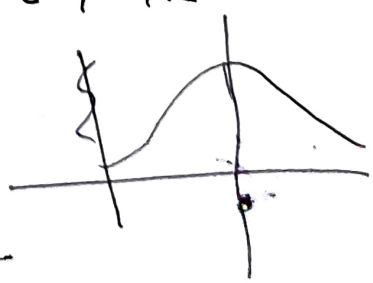


2. Score Statistics:

$\sigma \rightarrow 68\%$
$2\sigma \rightarrow 95\%$
$3\sigma \rightarrow 99.7\%$

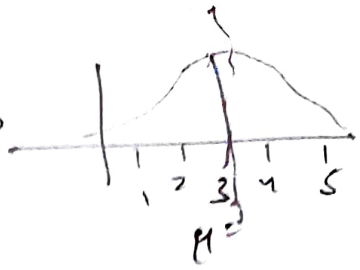


$SN. = \frac{X-\mu}{\sigma}$



next page

$$Z\text{-Score} = \frac{X_i - \mu}{\sigma}$$

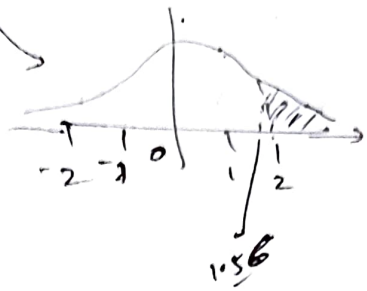


Eg. $\{1, 2, 3, 4, 5\} \Rightarrow \underset{(\mu)}{\text{mean}} = 3$

$\sigma = 1$ (given)

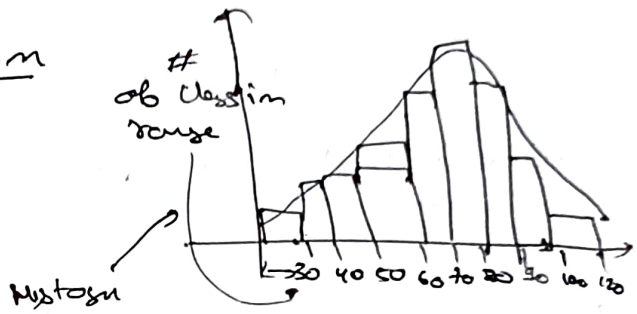
$Z\text{-Score} = \{-2, -1, 0, 1, 2\}$

$\frac{X_i - \mu}{\sigma}$

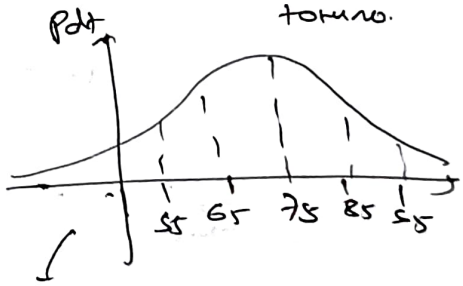


Eg: X: Population of classroom

Class No.	Population (X)
1	50
2	71
3	70
4	72
5	80
⋮	⋮
1	10
	⋮
	1



Pdf ⇒ $\frac{\#}{\text{total no.}}$



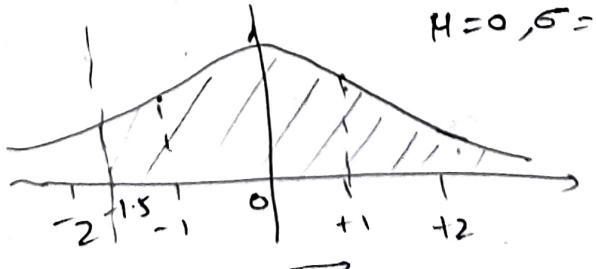
$\mu = \frac{\sum x_i}{N}$

$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$

$\mu = 75, \sigma = 10$

S.N.

$\mu = 0, \sigma = 1$



μ, σ

$P(X > 60)$

$P\left(x' > \frac{60 - \mu}{\sigma}\right)$

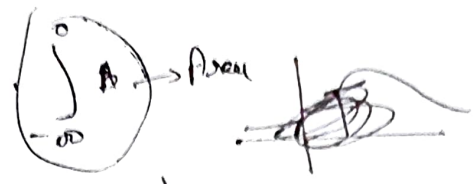
$P(x' > -1.5)$

Z-score table ⇒ $P(x' \geq 1.5) = 1 - P(x' < -1.5)$

$= 1 - 0.0668$

≈ 0.94

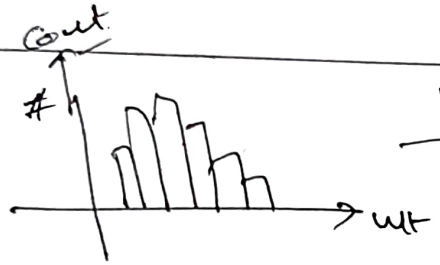
Z-score table

Z-score table ⇒ 

∴ Z score for +1.55 is 1.55

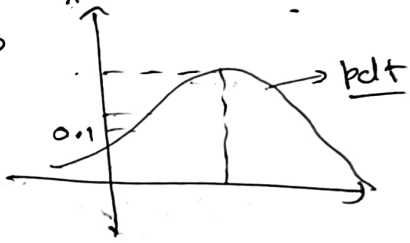
~~+1.5~~ ~~+1.55~~

PDF



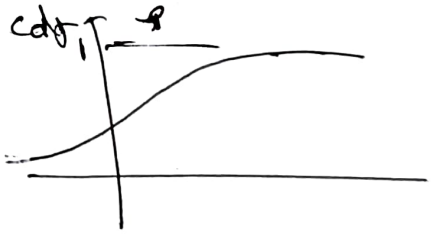
pdf

%



Regression:

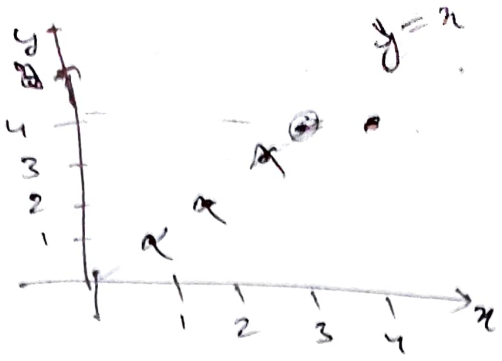
$\hat{y} = mx + c$



~~Cost fun = $\sum e_i^2$ → minimize~~

Cost function = ~~$\frac{1}{2}$~~ $\frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2$

e



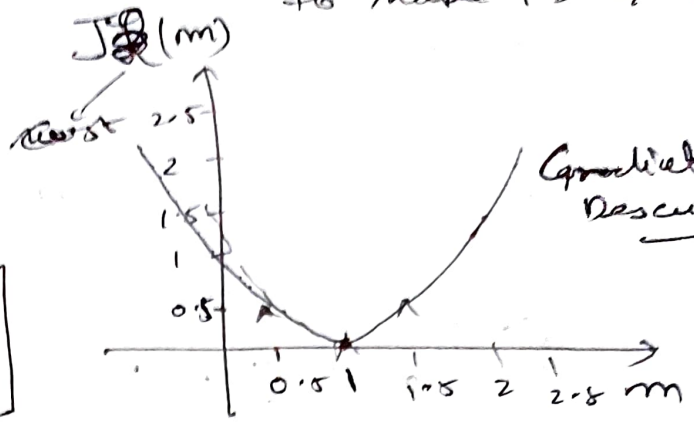
$\hat{y} = mx + c$

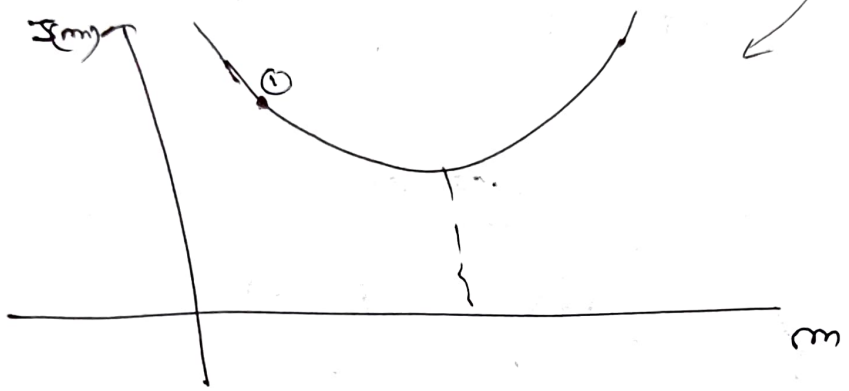
Assum $c = 0$,

to make 1D Cost fun

$\hat{y} = mx$

$J(m) = \left[J(m) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \right]$





Conv. Convergence Theorem: for convex optimization.

Learning Rate.

$m = m - \frac{\partial J}{\partial m} \times \alpha$

slope.

α (Learning Rate) = small

Generalized Model → low Bias & low Variance

Overfitting → high variance

~~Underfit~~ → ~~high bias~~ ::

Ridge Regression

$$J = \sum_{i=1}^{m} (y - \hat{y}) + \lambda (\text{Slope})^2$$

High Bias → for training data we have some amount of error

Inc. bias but reduces variance

why?

high slope → high penalization

(Large changes with x) ∴ over fitted.

do not go to zero

LASSO:

$$J = \sum_{i=1}^{m} (y - \hat{y}) + \lambda |\text{Slope}|$$

↳ Helps in feature selection.

Eg:

$$Y = m_1x_1 + m_2x_2 + m_3x_3 + c$$

$$\lambda (|m_1| + |m_2| + |m_3| + |m_4|)$$

↳ These move towards zero

Multicollinearity

Pearson Correlation Coefficient:

Co-variance: Random variable X_1 (Size of room), X_2 (Price of house)

Size	Price
1200	100
1500	200
1800	300

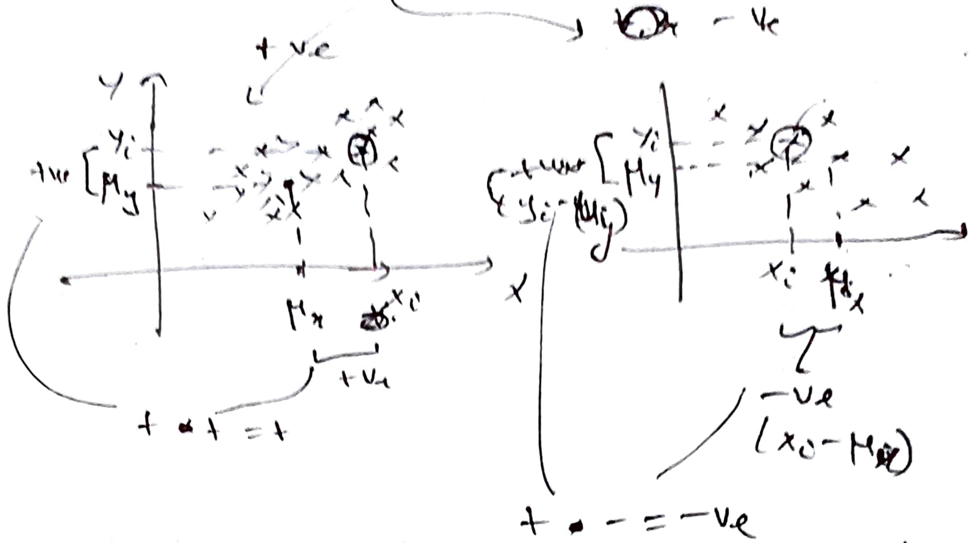
Size ⟷ Price

Can we quantify relation between them?

$$\text{Covariance}(\underset{x}{\text{Size}}, \underset{y}{\text{Price}}) = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_x) * (y_i - \mu_y)$$ Variance

$Cov(x,y)$ $\{\sigma_{xx} = (\sigma_x^2)\}$ Variance $\{Cov(x,x) = var(x)\}$

$$Cov(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu_x)(y_i-\mu_y)$$



Pearson Correlation Coefficient:

$$PCC\left(\rho_{(x,y)}\right) = \frac{Cov(x,y)}{\sigma_x \cdot \sigma_y}$$

$(-1 \text{ to } +1)$ — σ_x: s.d., σ_y: s.d

X	Y
Height	Weight

ρ → Strength / Direction of Relationship

σ_x & σ_y = S.d

σ_x^2, σ_y^2 = Variance → Consider

$$\sigma_x^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu_x)^2$$

$$Cov(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu_x)(y_i-\mu_y)$$
→ for 2

we get to know the Strength by $\rho_{(x,y)}$ & not Cov(x,y) because here we divide by sd(x) & s.d(y).

Spearman's Rank Correlation Coefficient

(Better than $\rho_{(x,y)}$ as Pearson checks for linear relationship but this checks for non-linear)

- $r_s(x,y) \in [-1, +1]$ → Pearson of Rank(x) & Rank(y)

$$r_s = \rho_{R(x),R(y)} = \frac{Cov(R(x),R(y))}{\sigma_{R(x)} \cdot \sigma_{R(y)}}$$

$$r_s = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

Acc. to 'y'

x	y	x_{score}	R(x)	y_{score}	d_i	d_i^2
10	500	5	1	2	-1	
5	100	10	2	1	1	
50	400	20	3	3	0	
20	300	50	4	4	0	
			5			

$\sum d_i^2 = 2$

$r_s = 1 - \frac{6 \times 2}{5(25-1)}$

$r_s \Rightarrow \{4/5\}^2$

Variance & S.D:

Bias & Variance:

Overfitting → understood like with degree of polynomial example

$P_n = 1$ $P_n = 2$ $P_n = n$ → Overfitted.

Regression:

Underfitting

- High Bias
- High Variance

error of training data

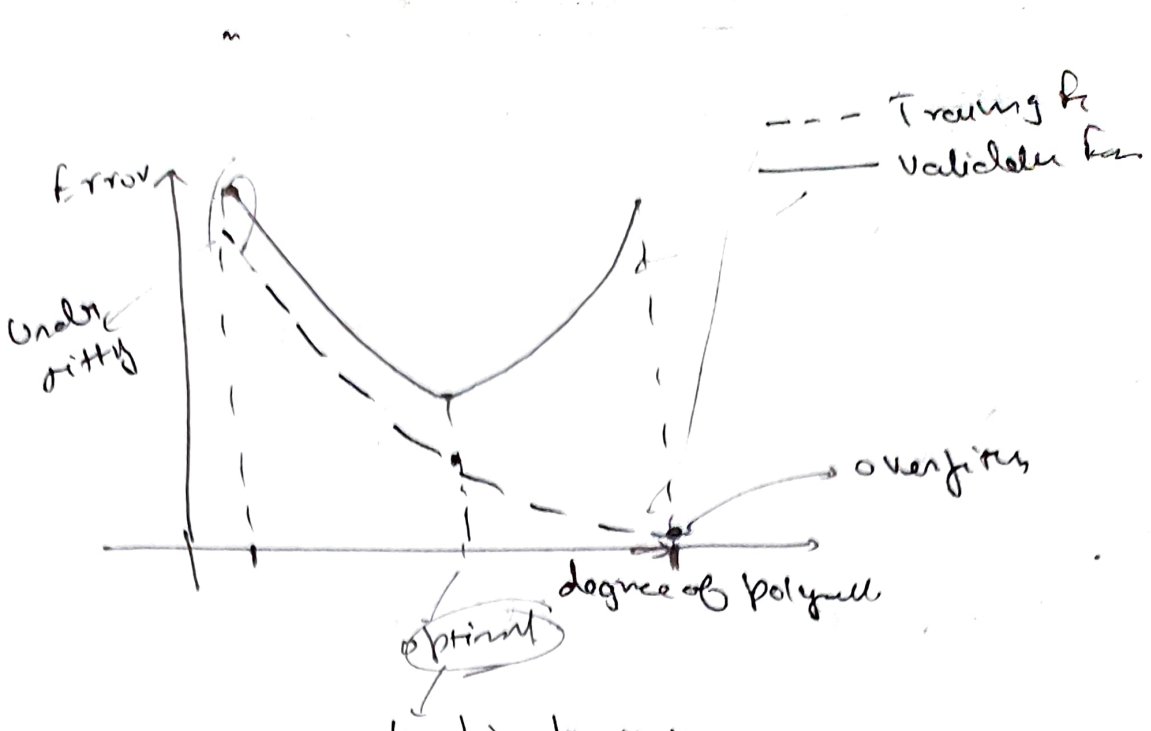
error of test data

overfitting

- low Bias
- High Variance

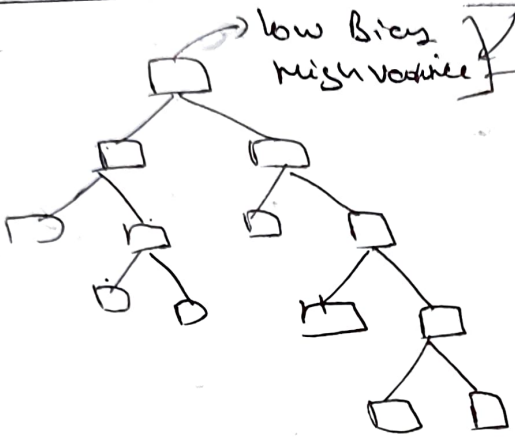
classification

similar



overfit & underfit in Decision Trees & Random Forest.

(1) DT



~~to avoid~~ as the Tree is made in accordance with the training data.

⇒ To Avoid High variance we do (pruning)

we do not make DT for whole depth.

(2) RF → Low Bias

→ high variance but due to Boosting (Bootstrap & Aggregation) → low variance

Inputs

as the training is splitted.. Different DT's are formed.

Standardization & Normalization

Feature Scaling → techniques

Normalization → Scales the feature b/w (0-1)

Standardization → Scales features based on standard Normal distribution. (M=0, S=1)

Ex: Normalization

↳ Min max normalization

$$\left(X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \right)$$

Standardization → (Standard Scaler) → Sklearn.

⇒ Scaling can be Ignored in tree based models.

When to use Normalization & Standardization?

Normalization	Standardization
→ Deep, ANN : we scale down b/w (0-1)	→ In Models where we use Euclidian distance
→ This helps them to learn weights quickly	→ when we need some type of optimization
	→ Mostly performs well.

Hypothesis testing:

→ we Evaluate 2 mutually exclusive statements (either one is true) → on the population → using SAMPLE DATA

Ex: Defendant (in court) → Guilty / Innocent

Steps: ① Make initial Assumption (H_0) → Null Hypothesis

Ex: H_0 = Defendant is Innocent →

Step (2): Called Data { Evidences } Ex: Evidences → fingerprints → DNA test etc

Step (3): Gather evidences to Either REJECT / ACCEPT NULL hypothesis (H_0).
or Status QUO hypothesis (hypothesis of No effect)

Alternate hypothesis: H_1 (opposite of NULL hypothesis H_0)
(Research hypothesis)

Type I Error:
If I know, by some way that the NULL hypothesis H_0 is true but I don't have enough evidences to prove that, then H_0 will get rejected.

Confusion Matrix:

	(H_0 Truth)	(H_1 Truth)
ACCEPT	OK	Type 2 Error
REJECT	Type 1 Error	OK

Type 2 Error: H_0 ⇒ Market will crash
H_1 ⇒ Not crash
Evidences ⇒ Not enough
Evidences ⇒ Enough

But true H_0 gets accepted in reality i.e. Market crashes → due to lack of evidences
is Type 2 Error

Hypothesis is always on parameters & NEVER ON STATISTICS (e.g. μ, σ etc)

T Test, CHI Squared Test, ANOVA Test

(One categorical feature)

Gender	Age Grp	Weight (Kg)	Height (cm)
M	Elderly	70	1.4
F	Adult	65	1.2
M	Adult	65	1.4
M	Child	20	1
F	Adult	75	1.3
M	Elderly	80	1.3

Sample.

* One Sample Proportion Test

→ choose 1 categorical feature 'Gender'

Q: Whether there is a diff. b/w proportion of M & F.? in the population

$H_0 \Rightarrow \#M = \#F$

$H_1 \Rightarrow \#M \neq \#F$

Test: Considering H_0 is true what is the prob. that H_1 is true?

i) $P \leq 0.05 \rightarrow$ significance value (α)

→ Reject H_0 & Accept H_1.

* Two Categorical features: {Gender, Age grp} → Chi Sq. test

Q: Is there diff. b/w #M & #F based on Age Group

$H_0 \Rightarrow M = F$ acc to Age grp

$H_1 \Rightarrow M \neq F$ "

Test → Chi Sq. → $P \leq 0.05 \Rightarrow$ Reject H_0

* One Continuous Variable {Weight}

Two → T-test → Correlation test → H_0: No relation $\rho = 0$, H_1: $\rho \neq 0$

→ we only talk about sample statistics.

Q: Based on some pre sample is there a difference mean weight of two samples

H_0: $\mu_A = \mu_B$ H_1: $\mu_A \neq \mu_B$

Test: T-test

⇒ [one categorical with two categories + one Numerical Variable]

→ T-test [Ex: Gender, weight]

⇒ [one or more categorical features with categories > 2 + Numerical features]

→ ANOVA TEST

#. T-test : * Type of Inferential statistics which is used to determine if there is a significant difference b/w the means of two groups which may be related in certain features

T-test has 2 types
- ① one-sampled t-test
- ② Two sampled t-test

population

① One-Sample T-test : Tells us weter means of the sample and the population are different.

$$t = \frac{\bar{x} - \mu}{s_{\bar{x}}} \quad \text{where} \quad s_{\bar{x}} = \frac{s}{\sqrt{n}}$$

H_0 ⇒ No differ.

H_1 ⇒ ∃ Differ

if $P \leq 0.05$ → Reject H_0 Accept H_1

μ ⇒ Proposed Constant for population mean.
$\bar{x}$ ⇒ Sample mean
n ⇒ Sample size (# of obsrvs)
s ⇒ Sample standard deviation
$s_{\bar{x}}$ ⇒ estimated s.d of the mean

⇒ **Two Sample T-test**: The independent samples t-test or 2-sample t-test compares the means of two independent groups in order to determine whether there is statistical evidence that the associated population means are significantly different. The independent samples t-test is a parametric test. This test is also known as: independent t-test.

$$t = \frac{\bar{x}_2 - \bar{x}_1}{\sqrt{s^2\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}} \quad ; \quad s^2 = \frac{\sum_{i=1}^{n_1}(x_i - \bar{x}_1)^2 + \sum_{j=1}^{n_2}(x_j - \bar{x}_2)^2}{n_1 + n_2 - 2}$$

⇒ **Paired T-test**: To check how different samples from the same group are.

Wt_1 = { ↳ weight of students this year

Wt_2 = Wt_1 + { ↳ added distances from a N(1.5, σ) ↳ mean = 1.5 m

{ Wt. after 3 yrs.

Now. Are the Wt_1 & Wt_2 ⇒ statically different.

H_0 ⇒ same

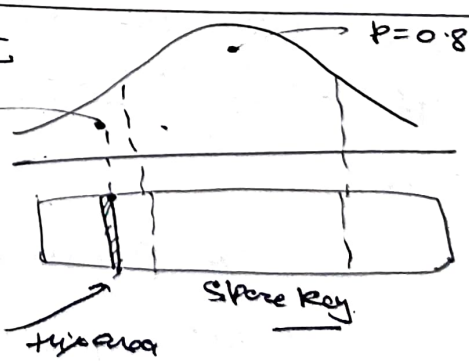
H_1 ⇒ diffrt.

Correlation Test:

---

p-value:

2 tailed test



{ p = 0.01

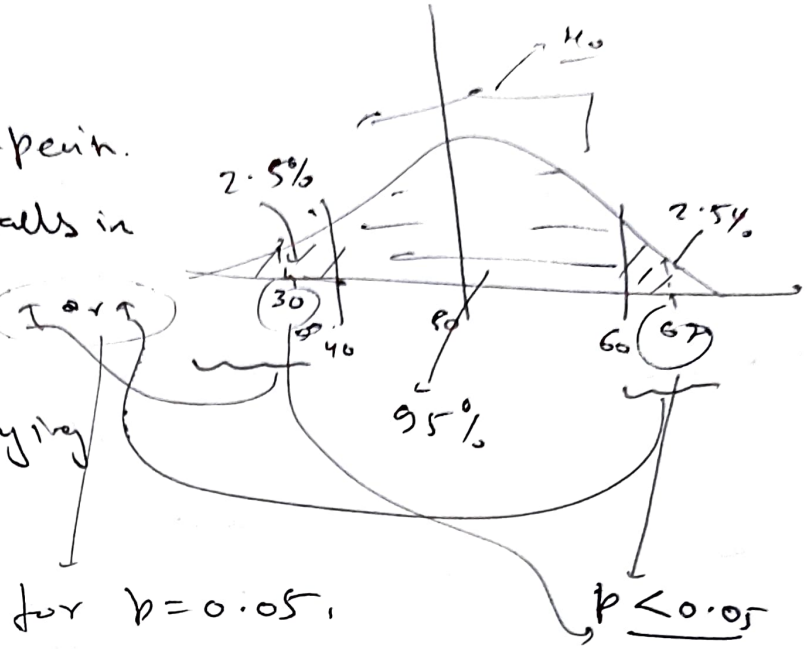
↳ if we repeat this experiment of pressing the space key the no of times we will be pressing is 1.

P-Value ⇒ It is the probability for the NULL hypothesis (Ho) to be true.

Ho (NULL hypothesis) → Treats Everything same or Equal.

Ex!

If I do an experiment & my outcome falls in this region (I or II) then I reject my Ho (NULL) saying that



for $p = 0.05$,

$p < 0.05$

∴ rejected

Multicollinearity:

in Linear regression

⇒ If our data has multicollinearity issue then we will have a Huge "Standard Error" for the variables

→ check the p-value is > 0.05 then see co-relation plot, or drop the feature with higher P-value

OLS → ordinary least sq.

↳ Take with Intercept c = 1 if taken Intercept

$y = c + \beta_1 x_1 + \beta_2 x_2$

R-square & Adjusted R-squared

SS_{res} = Sum of residuals or error.

$$SS_{res} = \sum (y_i - \hat{y}_i)^2$$

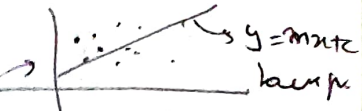
SS_{total} = Sum of average total.

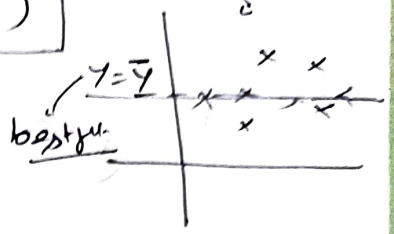
$$SS_{total} = \sum (y_i - \bar{y})^2$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{total}}$$

$R^2 \in [0-1]$

$R^2 \to 1$ {Good Model}





Q: Can the $R^2 < 0$?

Yes

$$R^2 = 1 - \frac{SS_{res}}{SS_{total}}$$

$\therefore SS_{res} > SS_{total}$

↳ Can happen when the line fitted is worse than the $y = \bar{y}$ line.

R^2 = checks Goodness of best fit line.

Adjusted R^2

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 - -$$

Penalizes features that are uncorrelated as the no. of feates inc. the ($R^2 \to$ inc.) why?

Intuitive as $(y_i - \hat{y})$ ↓, $\frac{SS_{res}}{SS_{tot}}$ ↓ ∴ R^2 ↑

Here's a problem that is the new feature added has No. Corr with the y (output) then also due to inc. in features the R^2 inc.

due to 6 feat optimized by LR

to avoid this we use Adjusted R^2

$$R^2_{adj} = 1 - \frac{(1-R^2)(N-1)}{(N-p-1)}$$

R^2 ⇒ sample R^2

p = # of Predictors

N = Total sample size

(inc. stays or doesn't change) if p ↑, R^2_{adj} ↓ due to this when feature are not correlated.

If feature that is co-related is added. then R^2 will also increase (inc.) the $(1-R^2)$ is low ∴ $(R^2_{adj}$ ↑) more

Diff. b/w R^2 & R^2_{adj}

→ Every time we add an indepdt variable to a model, the R-squared inc., even if indepdt variable is insignificant. It never decreases. Whereas Adjusted R-sq. inc. only when indepdt variable significantly affects depdt variable.

→ $R^2_{adj} \leq R^2$

Chi-squared test → Applied when we have two categorical variables from a single population.

→ It is used to determine whether there is a significant association b/w the two variables.

{sometimes like co-relation of categorical variables}

Ex: $X_1 \equiv$ Gender {M, F}
$X_2 \equiv$ Smoker {YES, NO}

Crosstab ⇒

Gen ↓ / Smoke →	YES	NO
M	60	97
F	33	54

} ⇒ given data (observed)

DoF = (# rows − 1) × (# col − 1)
(Degrees)

Not completely understood:

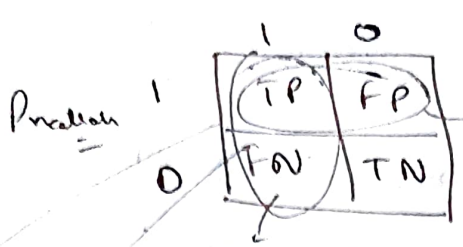
Metric: {Chi-Sqr. statistic OR P-value}

Expected

$$X^2 = \sum \frac{(O-E)^2}{E}$$

Imbalanced ⇒ {70 - 30%}

Confusion



$FPR = \frac{FP}{FP + TN}$ → Total Predicted

$FNR = \frac{FN}{TP + FN}$ → Total

Recall ⇒ $\frac{TP}{TP + FN}$

or

(TPR)

or

Sensitivity

Precision ⇒ $\left(\frac{TP}{TP + FP}\right)$ } out of the total +ve Predicted how many were correct

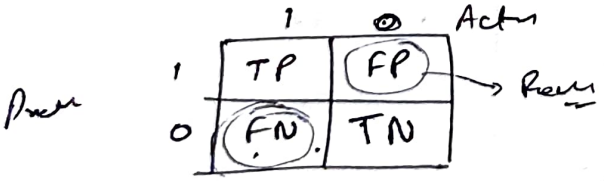
{Positive Predictive Value}

Ex: Spam 0 → Not Spam
1 → Spam

Precision ⇒ $\frac{TP}{TP + FP}$ → reduce this.

{Not a spam but classified as spam}

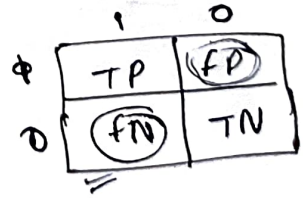
Ex: 1 → Spam (+ve)
0 → Not Spam (-ve)



Spam → Not.
1 → 0

$\frac{TP}{TP + FP}$ → Precision

1 → Cancer (+ve)
0 → No Cancer (-ve)



C → N
1 → 0

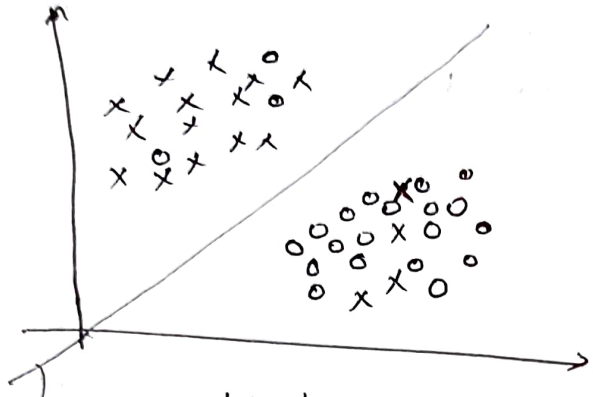
1 → No
0 → C

0 → 1

Logistic Regression:

(Binary classification)

→ for linearly separable classes.



$\{ y = w^T x + b \}$ ← slope / weight

Let's take only one feature

$y = mx + c$

Coefficient is not calculated like that in Linear Regression case

Assumption: $\{ x \rightarrow +1 \quad o \rightarrow -1 \}$ ← y's

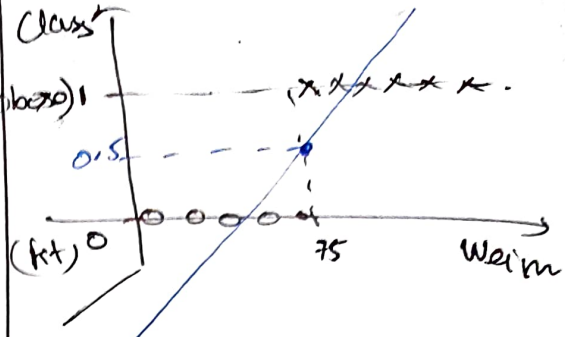
take $c = 0$

$y = w^T x$ or mx

$\{ y_x = 1 \quad y_o = -1 \}$ $\{ d_i = w^T x_i \}$

$d = \sum d_i = \sum_{i=1}^{n} w^T x_i$

Why Not do Binary classification using Linear Regression?



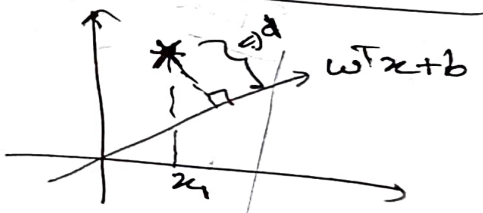
Regression

if $\hat{y} > 0.5 \rightarrow$ obese

$\hat{y} < 0.5 \rightarrow$ fit

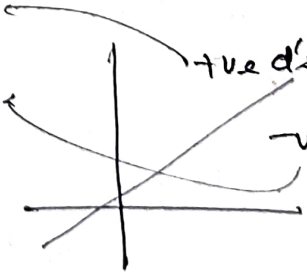
→ But due to one huge outlier like wt = 1000kg the regression line may be affected badly

∴ we do not use Linear in classification



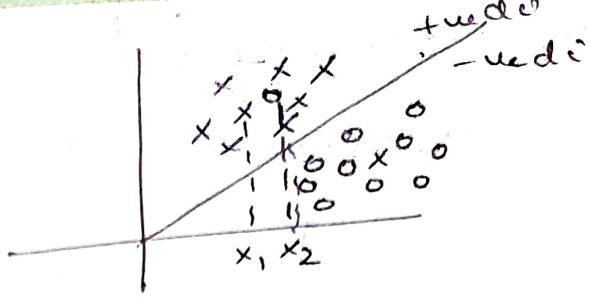
$\{ d = \frac{w^T x_i + b}{\|w^T\|} \}$

Assume w^T vector ∴ $\|w\| = 1$



$y_* = +1$

$y_o = -1$



d_i above $\Rightarrow +$

d_i below $\Rightarrow -$

Case ① → Correct classification

if we take above '*' at x_1 then $\Rightarrow y_i = +1$

$d_i = +ve$

$\therefore y_i d_i > 0$

$\{ y_i * w^T x > 0 \}$

Case ② → wrong classification

if we take 'o' at x_2,

$y_i = -1$

$d_i 2 = +1$

$y_i \cdot d_i = y_i \cdot w^T x_2 < 0$

$\Downarrow$

wrong y classified

Cost function: Max. $\sum_{i=1}^{n} (y_i (w^T x_i))$ ⇒ This should be Maximum

in logistic regression

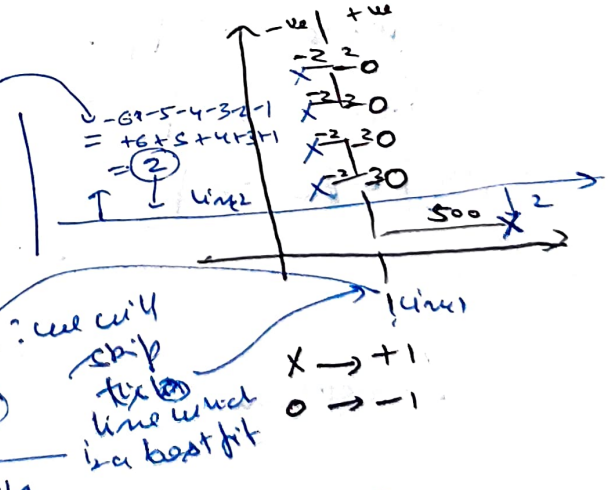
This can be optimized to get maximum Cost, for best fit

Dealing with outliers in Logistic Regression:

$\{ \sum_{\forall i} \Rightarrow y_i w^T x_i \}$ max

$\Rightarrow 2+2+2+2 - (-(2+2+2+2)) - 500$

$\Rightarrow 20 - 500 = (-480)$



∴ we will skip this line what is a best fit

∴ line(2) gets selected with value (+2) rather than line1 with value (-480).

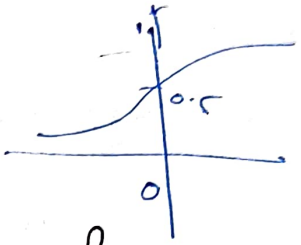
How to overcome this?

we apply sigmoid function.

$$\text{Cost function} = \max \sum_{i=1}^{n} \text{Sig}\left(y_i * w^T x_i\right)$$

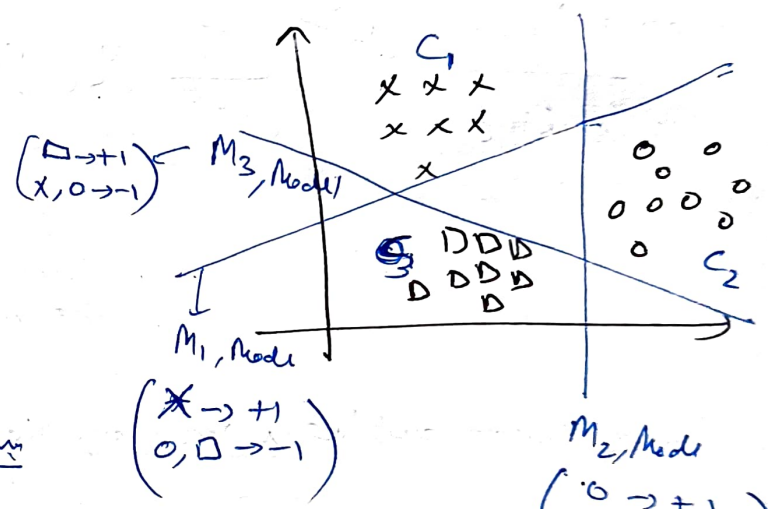
removes the effect of outlier by squashing the value b/w 0 to 1

$$\frac{1}{1+e^{-z}}$$



Multiclass classification using "Logistic Regression"

(One Vs All)



Now if I take a new point (x_{new})

Probability

M_1 Model → p_1

M_2 Model → p_2

M_3 Model → p_3

take highest probability i.e My point belongs to C_3 category i.e □

("ovr") → sklearn

ROC & AUC :

Threshold Problem → $T_p = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$

Y	Y_{prob}	T_{p_1}	T_{p_2}
1	0.8	1	1
0	0.56	1	1
1	0.4	1	1
1	0.3	1	1
0	0.2	1	0
1	0.7	1	1

(by Model)

Actual

	1	0
1	TP	FP
0	FN	TN

$$TPR = \frac{TP}{TP+FN}$$

$$FPR = \frac{FP}{FP+TN}$$

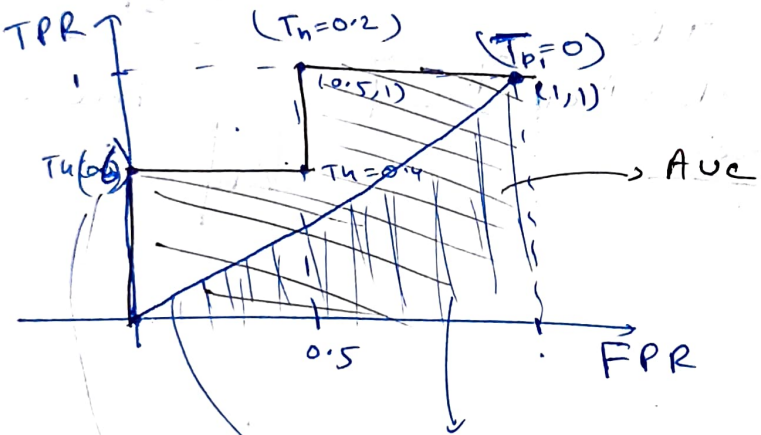
TPR, FPR

$$TPR = \frac{TP}{TP+FN} = \frac{4}{4+0} = 1$$

$$FPR = \frac{FP}{FP+TN} = \frac{2}{2+0} = 1$$

$TPR = 4/4+0 = 1$

$FPR = 1/1+1 = 0.5$



Model Must have AUC > Area under the line else it's a dumb model with p=0.5

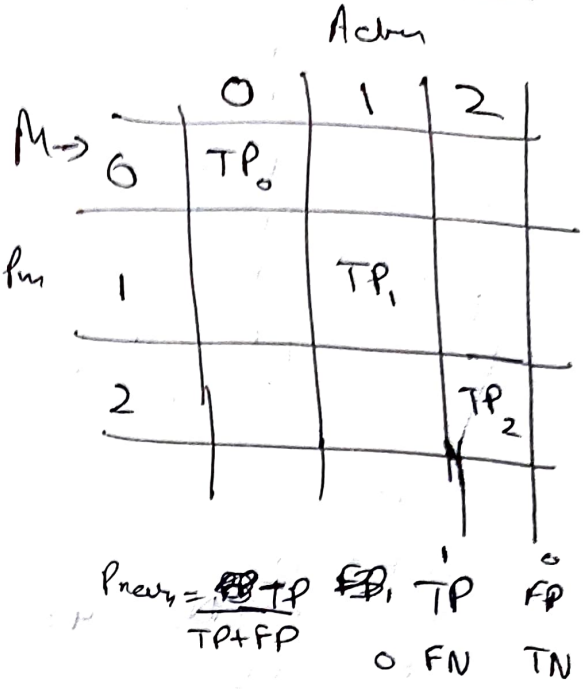
① Now, if Problem wants higher TPR ∴ ~~due to~~ ~~high~~ to lower the FN (like in cancer case) ∴ we can only focus of TPR and can take $T_h = 0.6$ as there FPR=0

② But if I want with TPR & dont care about FPR, Then we can take $T_h = 0.2$ (from TPR)

Ex:

Performance Metric for multiclass classification

$$\text{Accuracy} \Rightarrow \frac{TP_0 + TP_1 + TP_2}{\text{Total \# test cases}}$$



Precision & Recall

0 → cat
1 → Dog
2 → Fox

Precision:

→ Precision for class '0' - cat is the correctly predicted cat out of all Predicted cat

$$\text{Precision(cat)} = \frac{TP_0}{TP_0 + M_{01} + M_{02}}$$

Similar for other.

Similar for Recall.

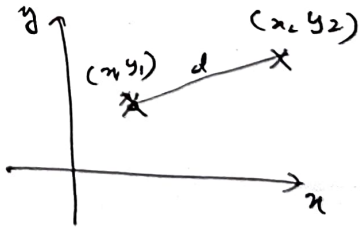
$$\text{Precision} = \frac{TP}{TP+FP} \qquad \begin{matrix} TP & FP \\ FN & TN \end{matrix}$$

Inside Precision

Actual

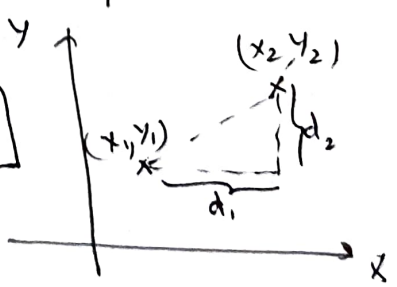
Euclidean & Manhattan distance:

Euclidean dist $\Rightarrow \sqrt{(y_2-y_1)^2 + (x_2-x_1)^2}$
(by Pythagoras)

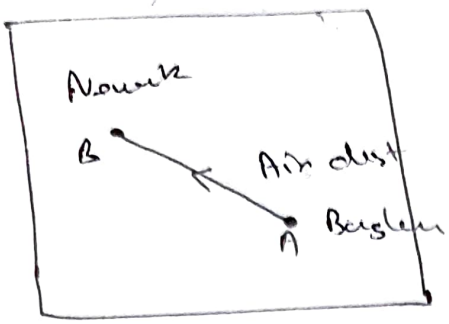


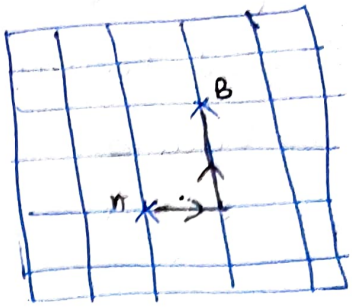
Manhattan dist $= d_1 + d_2$

$$= \left[|x_2 - x_1| + |y_2 - y_1| \right]$$



When to use Euclidian & Manhattan distance?





dist Bw A & B use Manhattan dist.

(Euclidian & Manhattan)

↳ are hyperparameter ∴ while tuning we can use both and see which work well.

Types of Cross Validation

① Leave one out Cross Validation (LOOCV)

1000 records

Test 1 → 1 Test rest Train

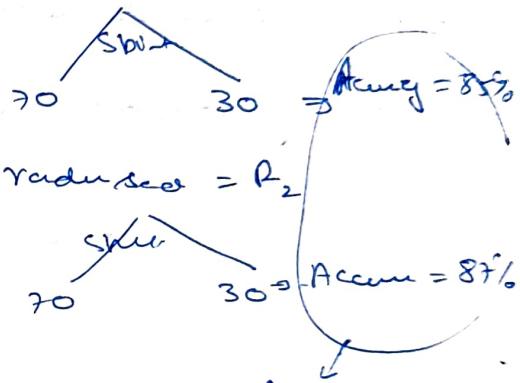
Test 2 → 1 Test, rest train

→ High Computation required.

→ Lead to Low Bias

1000 records

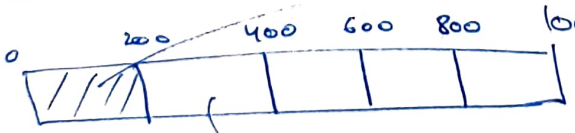
Use random state = R_1



Accuracy changes with random seed

∴ we split into Train test validation

② K-fold Cross Validation:



K=5 → 5 experiments → 1000/5 = 200

Mean Accuracy = $\frac{A_1 + A_2 + A_3 + A_4 + A_5}{5}$

Min Acc = min($A_1, A_2 \dots A_5$)

Max acc = max($A_1 \dots A_5$)

Accuracy → A_1, A_2, A_3, A_4, A_5

A_1 ← Model 1 → 0-200 Test, 200-1000 Train

Model 2 → 200-400 Test, Rest Train

Model 3 → 400-600 Train

Model 4 → 600-800

Model 5 → 800-1000

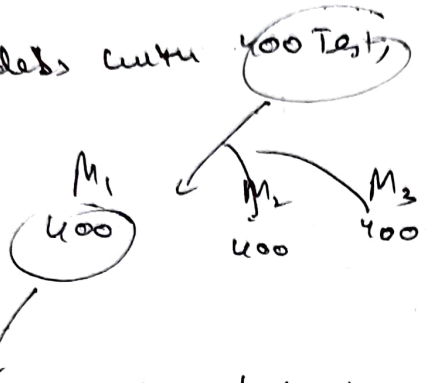
→ In K-fold Cross Validation. If the data is Imbalanced then there can be some probs of biased learning.

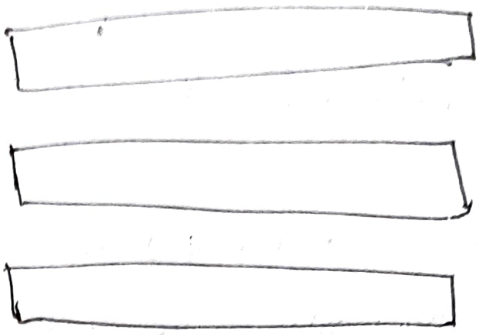
∴ we have → (Imbalanced data)

(3) Stratified Cross Validation

if K=3, 1200 record.

3 Models with 400 Test;





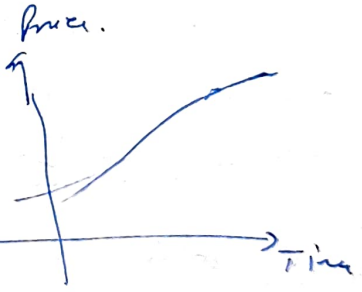
out of the 400 test stratified CV makes sure that both train & Test splits contain same proportion

(like if dataset as 60% 40% → class imbalance.

then Test → 60% 40% class

Train → //

(4) Time Series Cross Validation



- Day 1
  D2
  D3
  D4
  D5

Predict [D6, D7]

D1	D2	D3	D4	D5	D6
D2	D3	D4	D5	D6	D7

BAYE's Theorm:

$$P(A/B) = \frac{P(B/A) \cdot P(A)}{P(B)}$$

Prior

$P(A/B)$ → Posterior Probability

$P(B/A)$ → Likelihood Prob

$P(B)$ → Marginal

Naive Bayes classifier:

→

Input				output
x_1	x_2	x_3	x_4	y

$$P(y/x_1,x_2,x_3,x_4) = \frac{P(y)\prod_{i=1}^{4} P(x_i/y)}{P(x_1)\,P(x_2)\,P(x_3)\,P(x_4)}$$

(0-1) ⇒

$$P(y/x_1,x_2,x_3,x_4) \propto P(y)\prod_{i=1}^{4} P(x_i/y)$$

$$P(y/x_1,x_2,x_3,x_4) = \arg\max\left(P(y)\prod_{i=1}^{n} P(x_i/y)\right)$$

Ex: Naive Bias → Yes/No ; Good/Bad Classification

↳ Baseline for Text Classification (NLP)

Sentence → Good / Bad

After preprocess

we get Vector of (words) ⇒ words acts features

d_1	t_2	t_3	t_4	o/p (Good ① or Bad 0)
The	food	Delicious	Bad	
1	1	1	0	1
1	1	0	1	0
0	1	0	1	0

ex: Restaurant Review

$$P(y=\text{Good}/\text{Sentence}) = P(y=\text{Good}/[x_1 \cdots x_m])$$

$$\Rightarrow \frac{P(y)\cdot P(x_1/y)\cdot P(x_2/y)\cdot P(x_3/y)\cdots P(x_m/y)}{P(x_1)\cdot P(x_2)\cdot P(x_3)\cdots P(x_m)} \Rightarrow$$

$$= \text{argmax}\left(P(y) \cdot \prod_{i=1}^{n} P(x_i / y=yes) \right)$$

last table → $P(y_{yes}) = \frac{1}{3}$

$P(x_1 = \text{The} / y=yes) = 1/2$

$P(x_2 = \text{food} / y=yes) = 1/3$

$P(x_3 = \text{Delicious} / y=yes) = 1/1$

$P(x_4 = \text{bad} / y=yes) = 0/2 = 0$

Sentence → The food is Delicious

$$P(y=yes / \text{sent1}) \propto \frac{P(y=yes) \cdot P(\text{The}/yes) \cdot P(\text{food}/yes) \cdot P(\text{Delicious}/yes)}{P(\text{The}) \cdot P(\text{food}) \cdot P(\text{Delicious})}$$

$$= \frac{\frac{1}{3} \times \frac{1}{2} \times \frac{1}{3} \times 1}{\frac{2}{3} \times \frac{2}{3} \times 1} = \left(\frac{1}{18}\right)$$

$P(y=No / \text{sent1}) =$ Calculate similarly

$\neq (1 - P(yes/\text{sent1}))$

⇒ Naive Bayes fails in case when the feature vector has words that are not in feature

∴ Here $P(\text{word}/y) = 0$

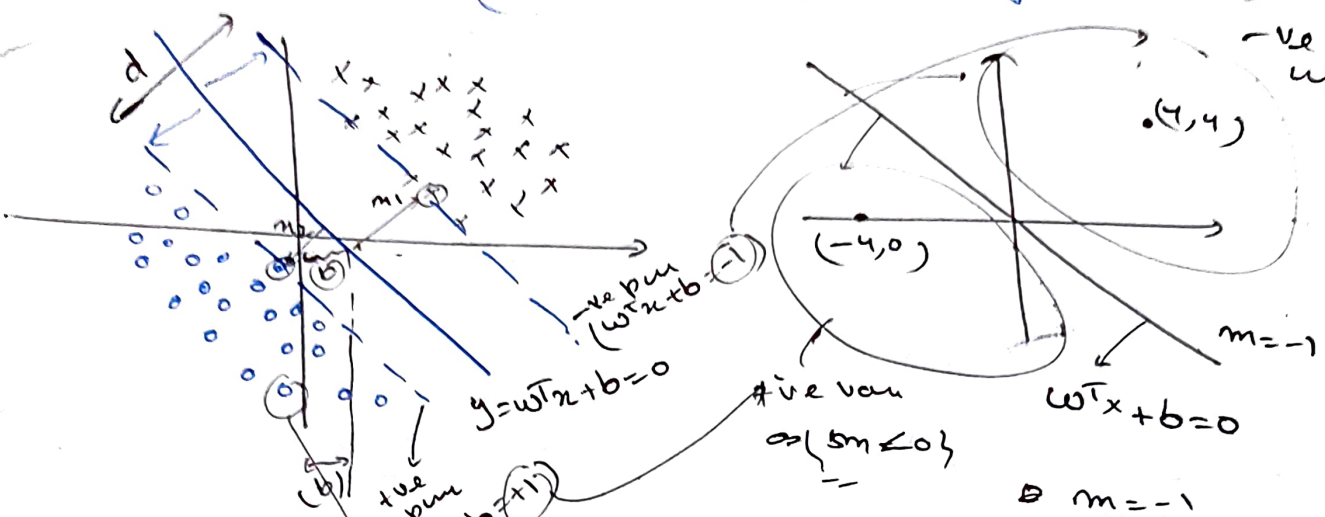
∴ giving true $\left(\frac{-ve}{No}\right)$ output as resp

Support Vector Machines (SVM)

→ Both for Regression & Classification

Maximize → Marginal distance.

SVM kernel → (Low dimension to high dimension)



$d=(x_2-x_1)$ → -ve dist

we want pnt

$w^T x_1+b=-1$

$w^T x_2+b=1$

$\{w^T(x_2-x_1)=2\}$

$m=-1$

$c=0$

$y=w^T x$

$y=\begin{bmatrix}-1\\0\end{bmatrix}[-4\ 0]$

$y=4 \geq 0$

$$|d|=\frac{w^T}{\|w\|}(x_2-x_1)=\boxed{\frac{2}{\|w\|}} \Rightarrow \text{Maximize}$$

$$(w,b)\ \max\left(\frac{2}{\|w\|}\right) \quad \text{s.t.} \quad y_i=\begin{cases}1, & w^T x+b\geq 1\\ -1 & w^T x+b\leq -1\end{cases}$$

Constraint → $y_i \cdot (w^T x_i + b_i) \geq 1$

Correctly classified.

Finally

$$(w^*, b^*) = \min \frac{\|w\|}{2} + C \sum_{i=1}^{m} \xi_i$$

$\max \frac{2}{\|w\|}$

How many error my model can take

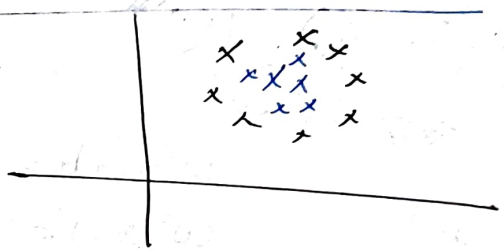
error sum (dist - from line)

Regularization

① Soft margin
② Hard margin

regularization

Non linear separable



① Polynomial kernel
② RBF kernel
③ Sigmoid kernel.

ROC curves:

∴

Box & Whisker Plot: