15.02.23

CSE 475 : Machine Learning

Books

1. The hundred-page Machine Learning Book, 2019
* 2. Python Machine Learning – Sebastian Raschka 2015
3.

AI

AI was born in 1956.
John Mc Carthy – Assistant Professor of Math

↳ to find how to make machines use language from abstractions, and concepts, solve kinds of problems now reserved for humans.

- The effort to automate intellectual tasks normally performed by humans.

- Not necessarily related with ML

- Hard coded rules crafted by programers
  (Deep Blue computer made for playing chess had beaten a champion of chess at that time)

- Was at peak in 1980s.

→ Image classification
→ Speech Recognition

Not possible with hardcoded rules

So, then comes

Machine Learning

ML

ML looks at the input data, the corresponding answers and figures out what the rules should be

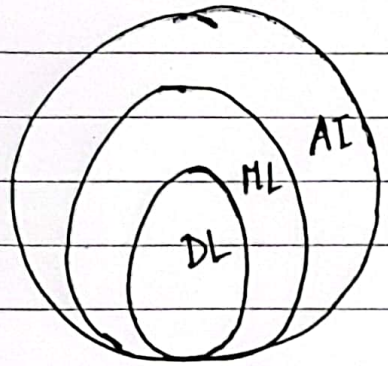
- an ML system is trained rather than explicitly programmed.

22·02·23

3 parametres which have been improved over decades

Hardware
Data
Algorithm

} ML → DL (huge neural networks
huge trained data)



→ Image Classification
→ Speech Recognition
→ Handwriting Recognition
→ TTS
→ Self ~~driving~~ driven car
→ Google Assistant
→ Smart Home

ML

Field of study that gives computers the ability to learn without being explicitely programmed

Real world unseen input → output
ML

Tom Michel (1999)

A computer program is said to learn from experience E with ~~rig~~ respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by

P, improves with experience E.

E - data for detecting spam or not
T - filtering
P - measured (Accuracy)

Classification

- Supervised Learning
- Unsupervised Learning
- Reinforcement

Supervised Learning

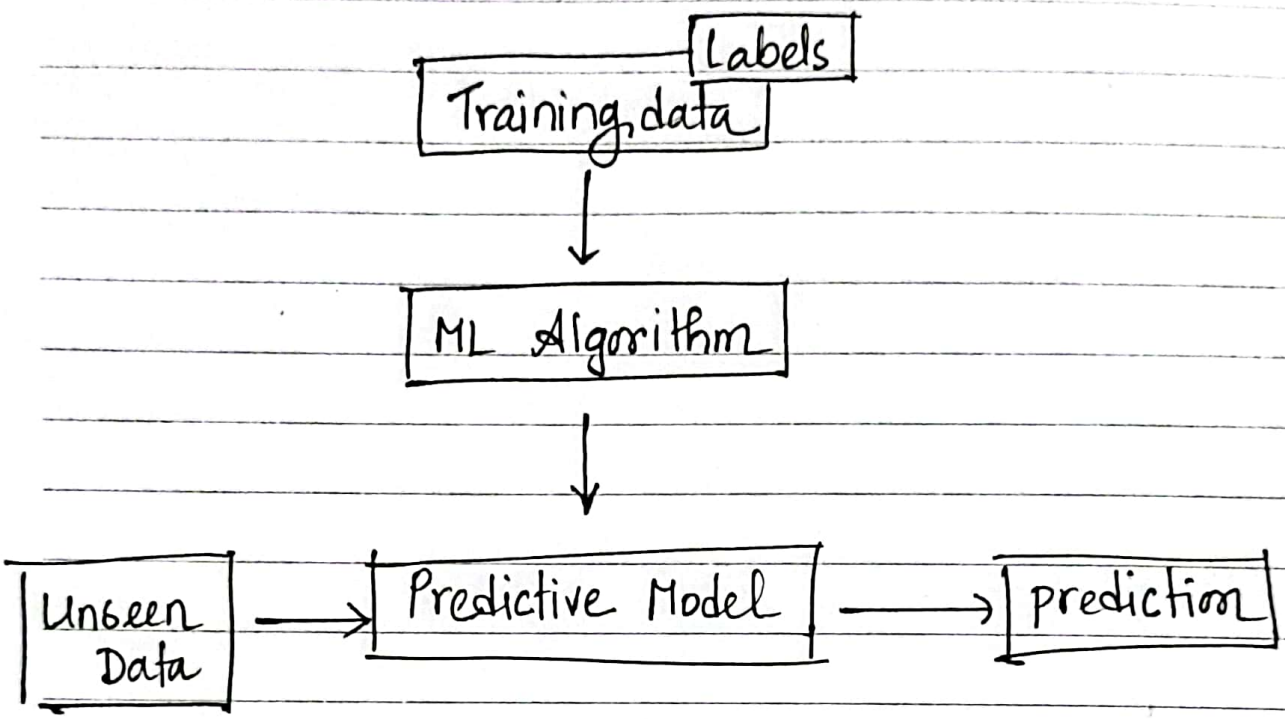
A set of samples where the desired output labels are known

e.g - spam filtering

emails	labels
e1 - - - -	spam
e2 - - - -	not spam

Unseen data

→ predict.



2 types of supervised ML:

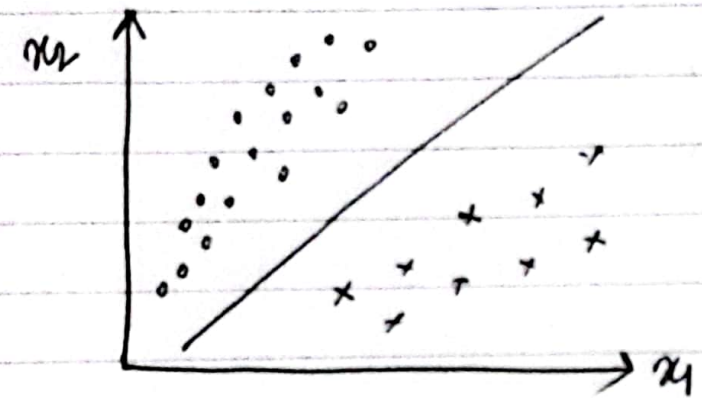
- Classification (Spam filtering) discrete value prediction
- Regression Analysis continuous value prediction (weather forecasting)

Classification

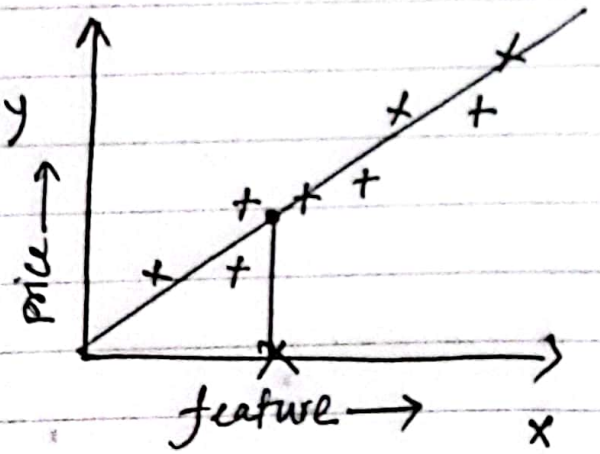
- Binary (0, 1) No Yes
- Multiclass — OCR (optical character recognition)

Regression : Prediction of continuous values/ outcomes

e.g: Housing Price Prediction

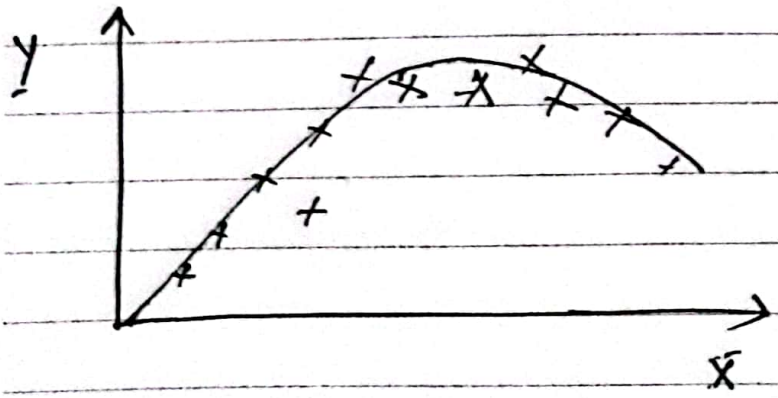


Binary Classification



Linear Regression

Feature space



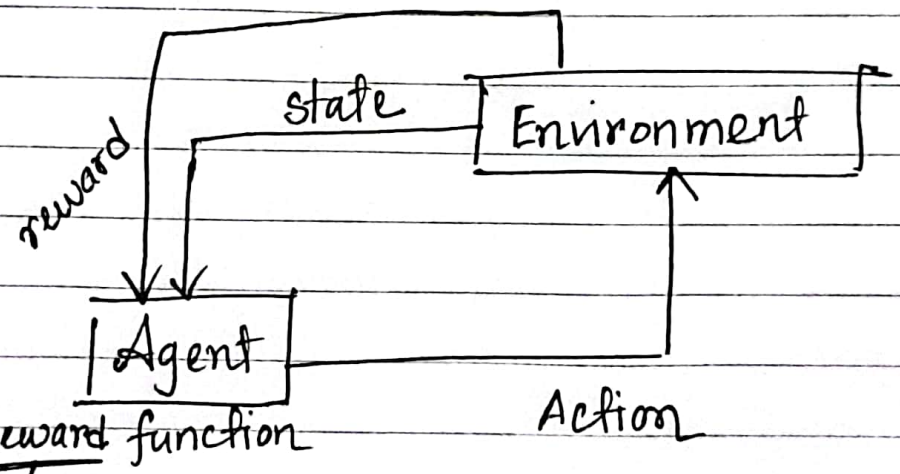
polynomial regression

Reinforcement Learning

- interactive learning

Improves the performance based on interactions and environment.

e.g: Chess Agent



Reward function

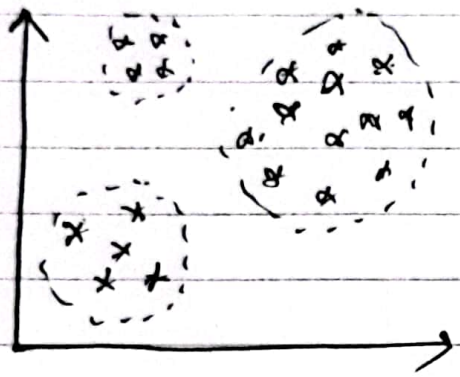
sequence of moves makes an agent → WIN

negative reward → LOSE

Unsupervised Learning

- Discovering the hidden structure

e.g: Market Analysis. (groups customers)

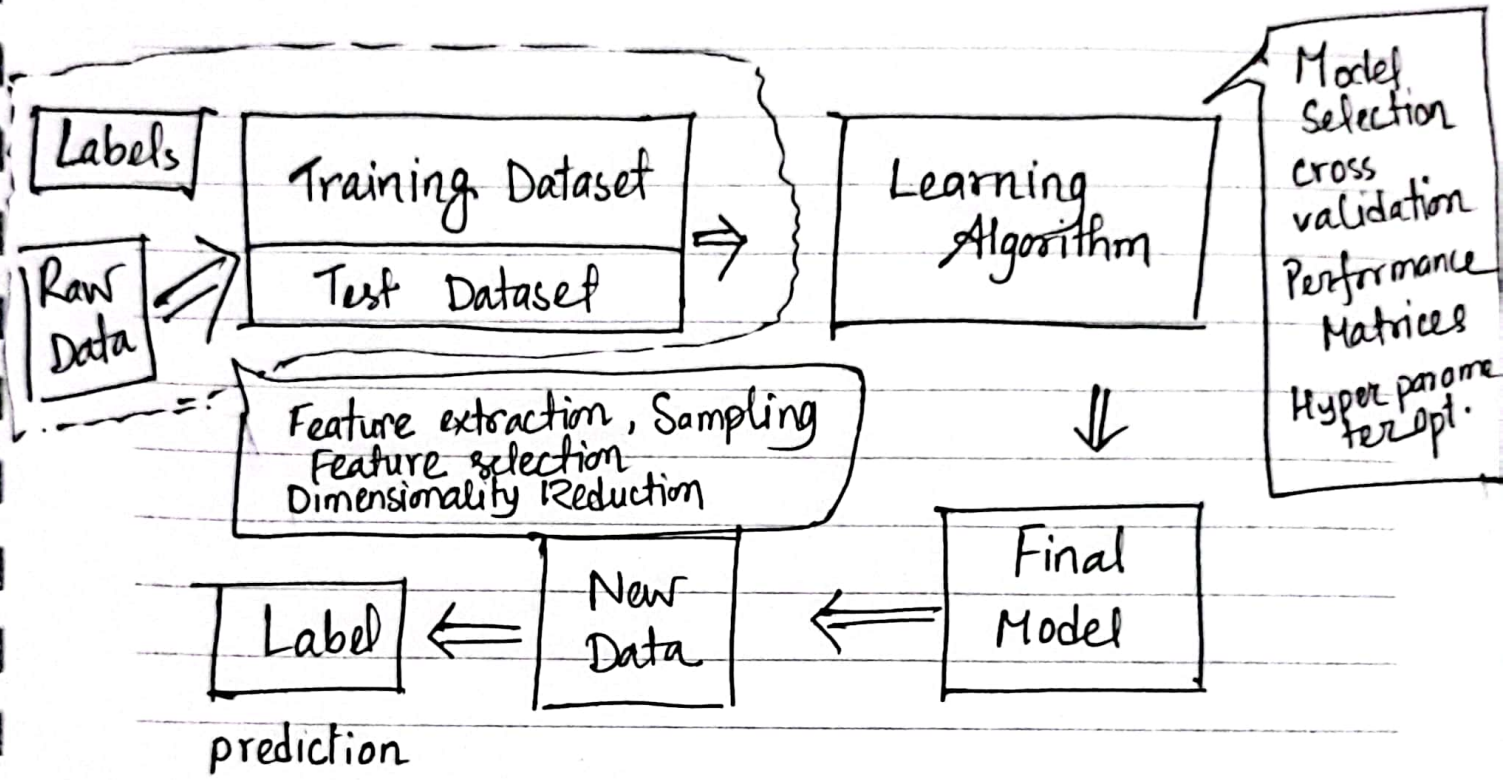


Clustering

- to explore the structure of our data to extract meaningful information without the guidance of a known outcome variable or reward function

26·02·23

A roadmap for building a ML Systems

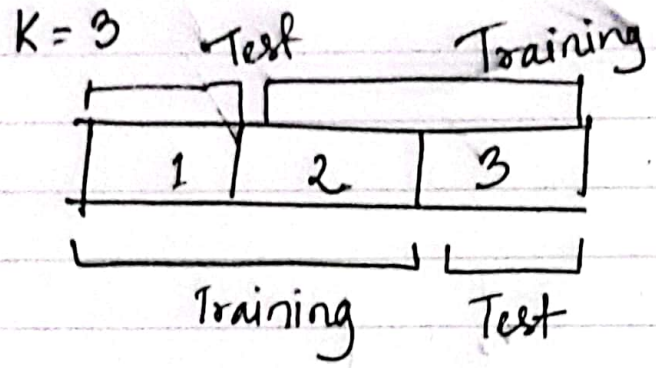


Improved ML system can be ~~done~~ made initially by quality data.

↳ First criteria for improvement.

Training Dataset — 70%-80%
Test Dataset — 20%-30%

cross validation / K fold cross validation
divide dataset into k parts



Test data is changed everytime.

New data → 100% unseen data

Held out validation → one time validation.

After cross validation, the full data become training dataset

The unused test data → 100% unseen data

Chapter- 02

McCullock - Pits (1943) published a paper -

MCP neuron

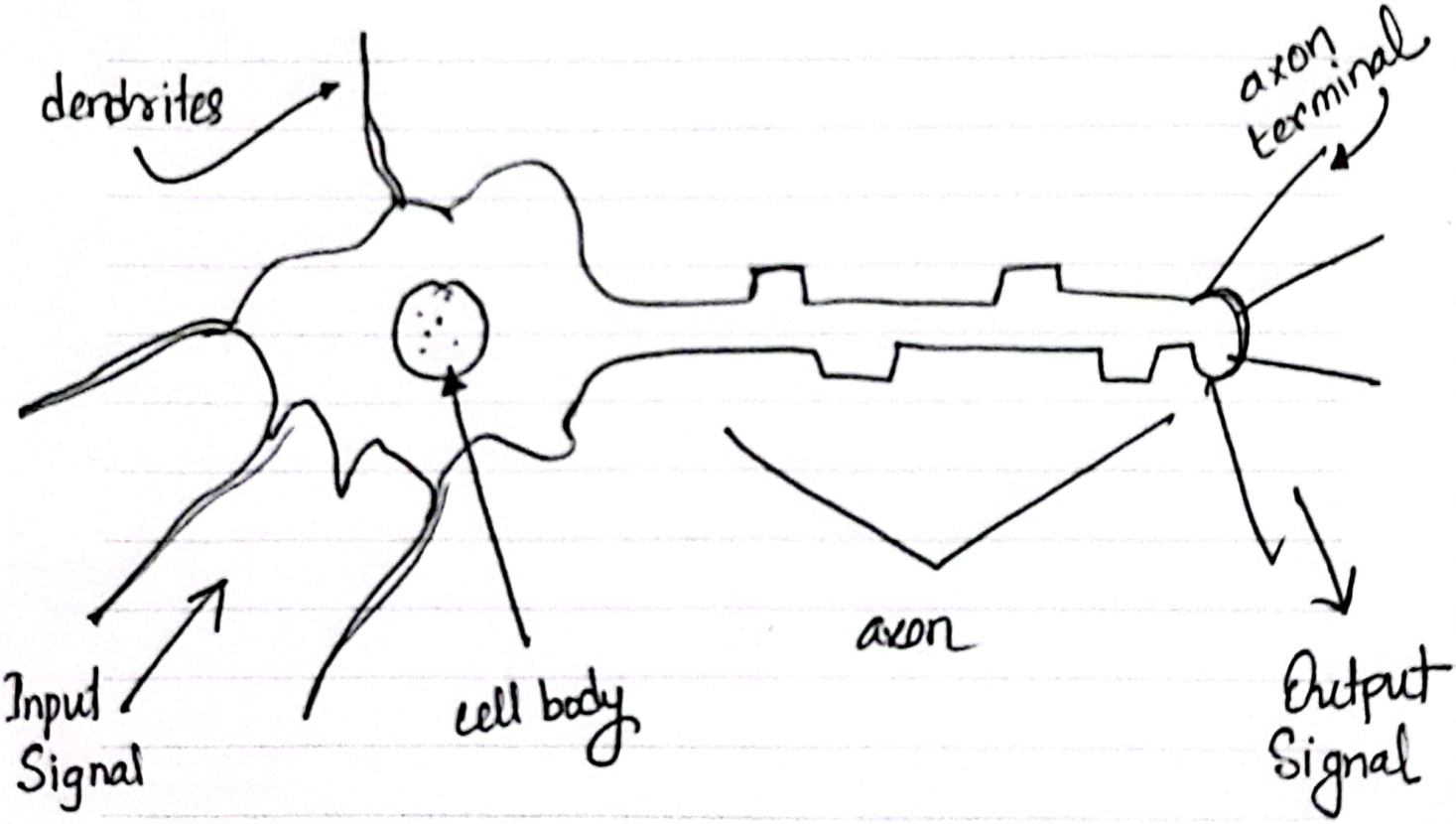


Fig: Neural Body

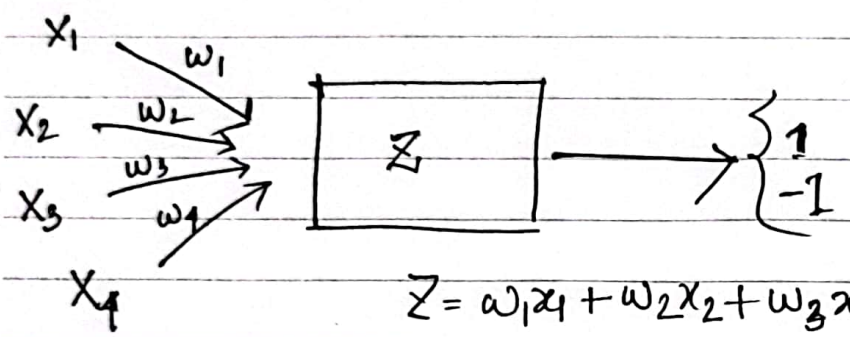
fires (high/low).

Rosenblatt, 1957 works on MCP neuron

MCP neuron

$\Downarrow$

perceptron



$$\phi(z) = \begin{cases} 1, & \text{if } z > \theta \\ -1, & \text{otherwise} \end{cases}$$

$$z = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$$

$\theta \rightarrow$ threshold.

07.03.23

Logic Gate

→ Simple logic gate with binary outputs
→ Single Signals arrive at dendrites
→ Integrated at cell body
→

Binary Classification task
activation function $\phi(z)$

$$z = w_1 x_1 + \dots + w_m x_m$$

$$W = \begin{bmatrix} w^{(1)} \\ w^{(2)} \\ \vdots \\ w^{(3)} \end{bmatrix}, \quad x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(3)} \end{bmatrix}$$

Heaviside Step Function

$$\phi(z) = \begin{cases} 1 & \text{if } z \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

weight zero as $w_0 = -\theta$, $x_0 = 1$ and bring the threshold θ to the left side

$$z = w_0 x_0 + w_1 x_1 + \cdots w_m x_m = w^T x$$

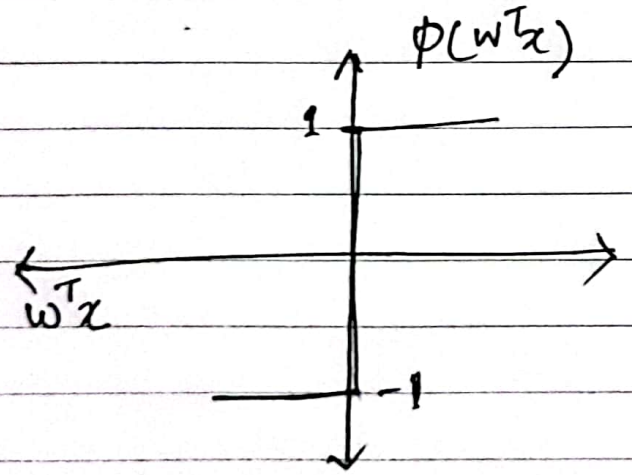
and

$$\phi(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$w_1 x_1 + \cdots w_m x_m = \theta.$$

$$\Rightarrow w_0 x_0 + w_1 x_1 + \cdots w_m x_m = 0$$

$$z = w^T x = \sum_{j=0}^{m} w_j x_j$$



Algorithm (Rosenblatt)

1. Initialize the weights to 0 or small random numbers.

② For $x^{(i)}$

→ compute $\hat{y}^{(i)}$

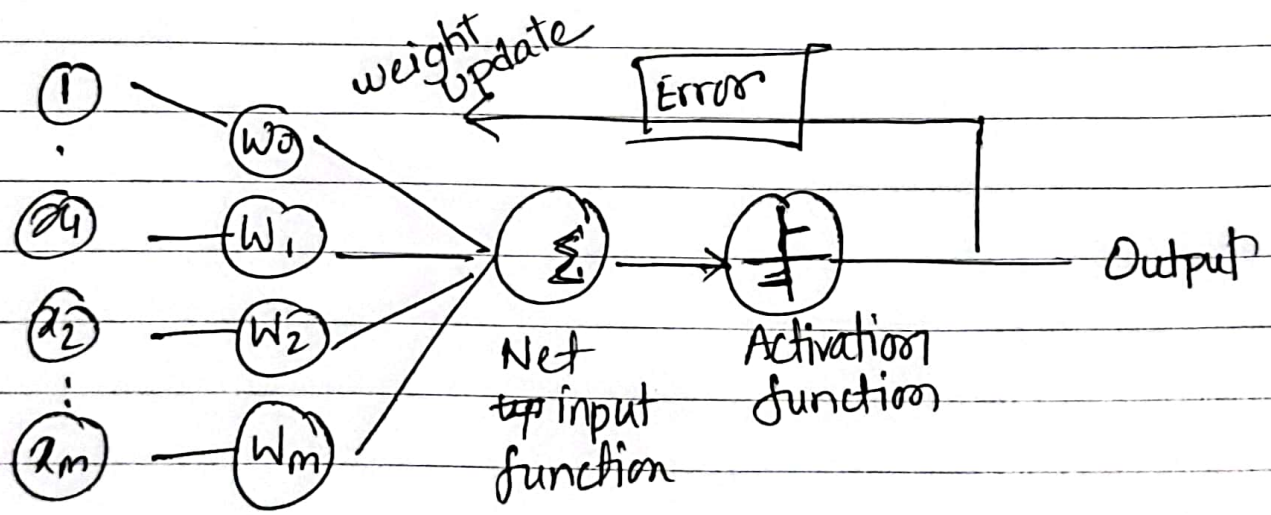
→ update $w^{(i)}$

$$w_j := w_j + \Delta w_j$$

Perceptron learning rule:

$$\Delta w_j = \eta \left(y^{(i)} - \hat{y}^{(i)} \right) x_j^{(i)}$$

η (learning rate 0.0 – 0.1)



<u>Correct predication</u>

$$\Delta w_j = \eta (1-1) x_j^{(i)} = 0$$

<u>Wrong predication</u>

$$\Delta w_j = \eta (1 - -1) x_j^{(i)} = \eta (2) x_j^{(i)}$$

$$\Delta w_j = \eta(-1-1)x_j^{(i)} = \eta(-2)\, x_j^{(i)}$$

Convergence guaranteed if → 2 classes linearly separation

→ learning rate is sufficiently small.

If not,

→ set max number of passes (epochs)

→ Set a threshold.

Adaptive Linear Neuron (Adaline)

→ Linear ~~actior~~ activation function

→ perceptron used a unit step function

→ $\phi(w^T x) = w^T x$

→ Quantizer is used.

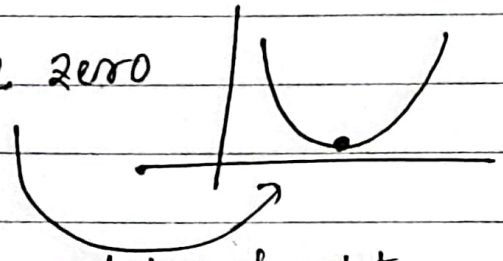
Cost Functions

ML algorithms often define an objective function optimized during learning

$$J(w) = \frac{1}{2} \sum_i \left(y^{(i)} - \phi(z^{(i)})\right)^2$$

sum of squared errors (SSE)

derivative zero



at lowest point

$$\frac{\partial J(w)}{\partial w_i} = 0$$; then cost fuction is minimized

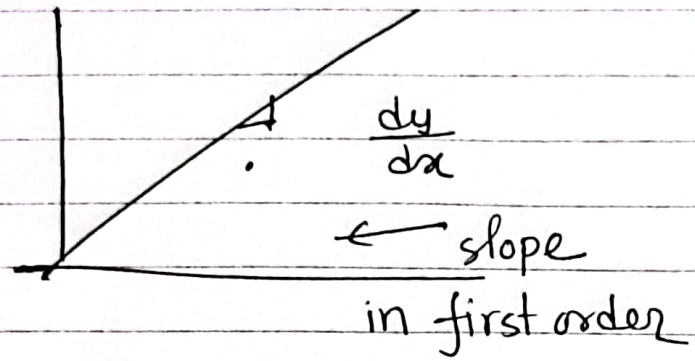
Adv:

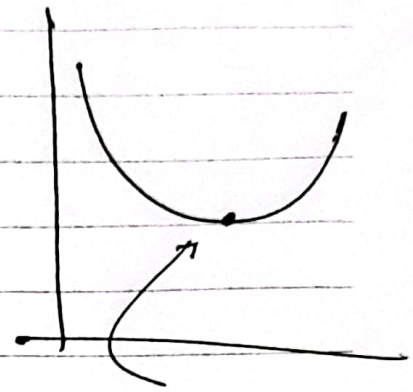
→ linear activation is differentiable (but not ~~int~~ step function

→ convex

→ use gradient descent

it is discrete





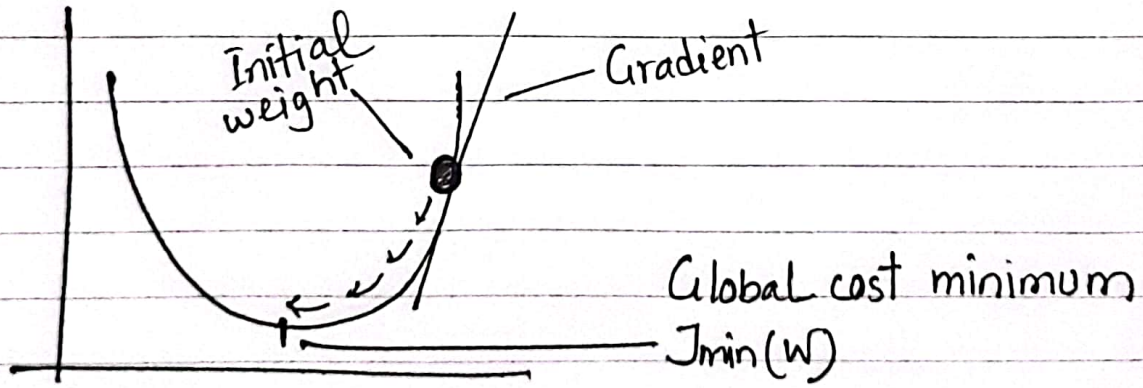
Slope: Linear function

Derivative: Convex function — derivative in polynomial

Gradient: Multivariable derivative

function of several variables → gradient is used.

→ represents the slope of the tangent of the graph of the function



$$w_j = w_j - \Delta w_j$$

Take a step away from gradient $\nabla J(w)$ of the cost function:

$$w := w + \Delta w$$

weight change —

$$\Delta w = -\eta \nabla J(w)$$

gradient of cost function:

$$\frac{\delta J}{\delta w_j} = -\sum_i \left(y^{(i)} - \phi(z^{(i)})\right) x_j^{(i)}$$

$$\Delta w_j = -\eta \frac{\partial J}{\delta w_j} = \eta \sum_i \left(y^{(i)} - \phi(z^{(i)})\right) x_j^{(i)}$$

$$\frac{\delta J}{\delta w_j} = \frac{\delta}{\delta w_j} \frac{1}{2} \sum_i \left(y^{(i)} - \phi(z^{(i)})\right)^2$$

$$= \frac{1}{2} \frac{\delta}{\delta w_j} \sum_i \left(y^{(i)} - \phi(z^{(i)})\right)^2$$

$$= \frac{1}{2} \sum_i 2\left(y^{(i)} - \phi(z^{(i)})\right) \frac{\delta}{\delta w_j} \left(y^{(i)} - \phi(z^{(i)})\right)$$

$$= \sum_i \left(y^{(i)} - \phi(z^{(i)})\right) \frac{\delta}{\delta w_j} \left(y^{(i)} - \sum_i (w_j^{(i)} x_j^{(i)})\right)$$

$$= \sum_i \left(y^{(i)} - \phi(z^{(i)})\right) \left(0 - x_j^{(i)}\right)$$

$$= -\sum_i \left(y^{(i)} - \phi(z^{(i)})\right) x_j^{(i)}$$

Adaline learning rule Vs Perceptron

→ $\phi(z^{(i)})$ with $z^{(i)} = w^T x^{(i)}$ is a real number

~~But~~ Integar class label ~~a~~ in Perceptron

→ Weight update on all samples in training set
Batch gradient descent

But perceptron updates weights incrementally after each sample

Learning Rate

Too high : error becomes largers (overshoots global min)

Too low : many epochs

Stochastic gradient descent

updates weights incrementally for each training sample

$$\boxed{\Delta w = \eta \left(y^{(i)} - \phi(z^{(i)})\right) x^{(i)}}$$

Stochastic $\longrightarrow$ Random

$$\boxed{\Delta w = \eta \sum_i \left(y^{(i)} - \phi(z^{(i)})\right) x^{(i)}}$$

$\hookrightarrow$ Batch gradient descent

Middle ground between SGD and batch GD

$\hookrightarrow$ Mini Batch Learning

$\rightarrow$ vector / matrix operations can be used.

Stochastic $\rightarrow$ too slow

Batch $\rightarrow$ computationally expensive

12.03.23

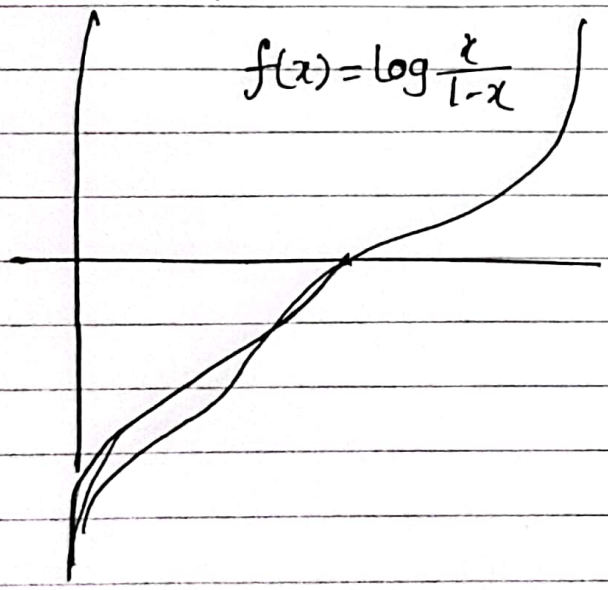
Logistic Regression

Probability in classification

Odds ratio $= \frac{p}{1-p}$

Logit function: $\text{logit}(p) = \log \frac{p}{1-p}$

natural log of odds



logit function is the inverse of the logistic function

probability more than 0.5 → Yes
probability less than 0.5 → No

Binary classification

$$\text{logit}\left(p(y=1|x)\right) = w_0 x_0 + w_1 x_1 + \dots w_m x_m$$

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

$$= w^T x$$

$$p(y=1|x) = \frac{1}{1+e^{-w^T x}}$$

<u>Derivation:</u>

$$\log(\text{odds}) = \log\left(\frac{p}{1+p}\right)$$

$$\Rightarrow e^{\log(\text{odds})} = \frac{p}{1+p}$$

$$\Rightarrow p = (1+p)\, e^{\log(\text{odds})}$$

$$= e^{\log(\text{odds})} - p\, e^{\log(\text{odds})}$$

$$\Rightarrow p\left(1+e^{\log(\text{odds})}\right) = e^{\log(\text{odds})}$$

$$\Rightarrow p = \frac{e^{\log(\text{odds})}}{1+e^{\log(\text{odds})}}$$

$$= \frac{1}{1+\frac{1}{e^{\log(\text{odds})}}}$$

$$= \frac{1}{1+e^{-\log(\text{odds})}}$$

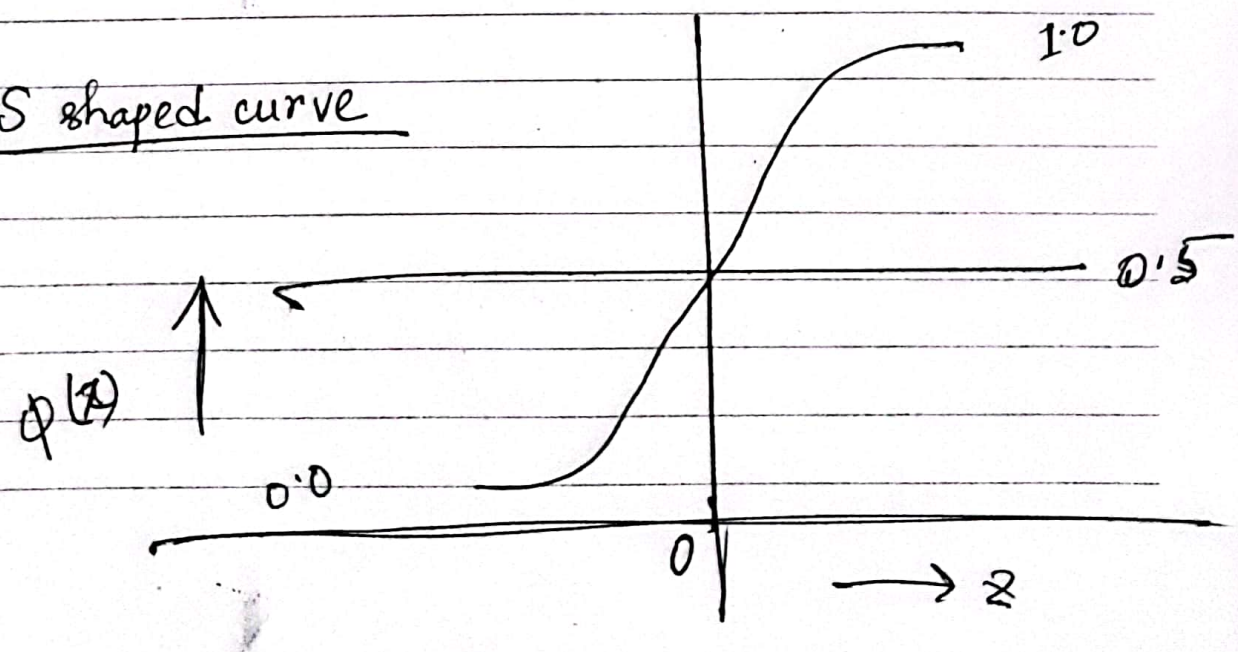
$$\boxed{p = \frac{1}{1+e^{-w^T x}}}$$

sigmoid function

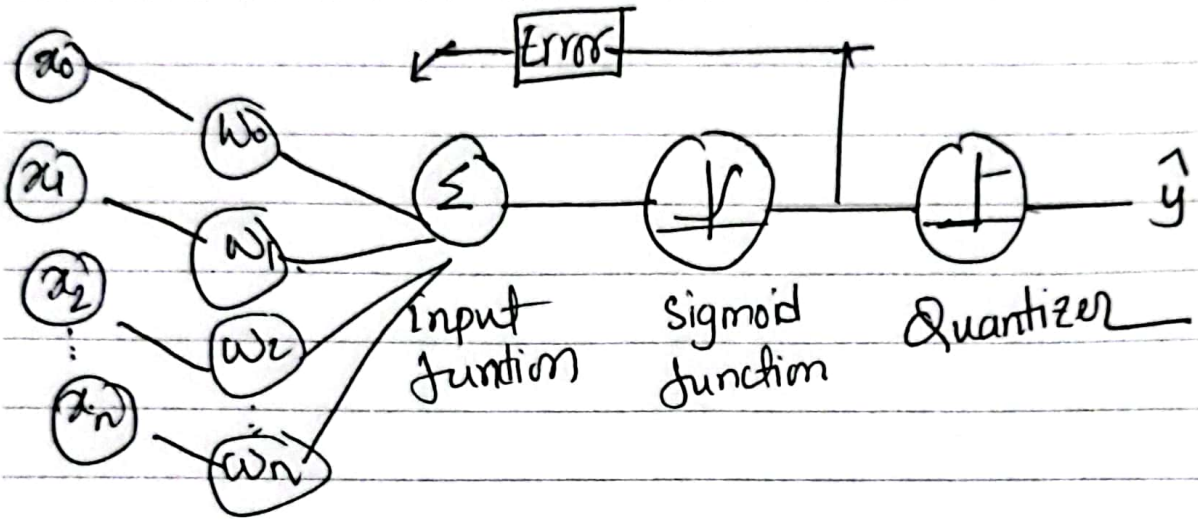
Logistic sigmoid

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

S shaped curve



Logistic — Probability
— Classification



$$\hat{y} = \begin{cases} 1 & \text{if } \phi(z) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Likelihood L

$$L(w) = P(y|x;w) = \prod_{i=1}^{n} P(y^{(i)} | x^{(i)}; w)$$

$$L(w) = \prod_{i=1}^{n} \left(\phi(z^{(i)})\right)^{y^{(i)}} \left(1 - \phi(z^{(i)})\right)^{1-y^{(i)}}$$

$$l(w) = \log L(w)$$

$$l(w) = \sum_{i=1}^{n} \left[y^{(i)} \log\left(\phi(z^{(i)})\right) + \left(1-y^{(i)}\right) \log\left(1-\phi(z^{(i)})\right) \right]$$

Cost Function

$$J(w) = \sum_{i=1}^{n} \left[-y^{(i)} \log\left(\phi(z^{(i)})\right) - \left(1-y^{(i)}\right) \log\left(1-\phi(z^{(i)})\right) \right]$$

Partial derivative of the log-likelihood function

gradient ascent

$$\frac{\delta}{\delta w_j} l(w) = \left(y \frac{1}{\phi(z)} - (1-y) \frac{1}{1-\phi(z)} \right) \frac{\delta}{\delta w_j} \phi(z)$$

we maximizes

gradient descent

we minimizes

$$\frac{\delta}{\delta z}\varphi(z) = \frac{\delta}{\delta z}\frac{1}{1-e^{-z}}$$

$$= \frac{1}{(1-e^{-z})^2} \cdot e^{-z}$$

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

$$= \varphi(z)\,(1-\varphi(z))$$

Resubstitute $\frac{\delta}{\delta z}\varphi(z) = \varphi(z)(1-\varphi(z))$

$$\left(y\frac{1}{\varphi(z)} - (1-y)\frac{1}{1-\varphi(z)}\right)\frac{\delta}{\delta w_j}\varphi(z)$$

$$= \left(y\frac{1}{\varphi(z)} - (1-y)\frac{1}{1-\varphi(z)}\,\varphi(z)\,(1-\varphi(z))\right)\frac{\delta}{\delta w_j}z$$

$$= \left(y\,(1-\varphi(z)) - (1-y)\,\varphi(z)\right)x_j$$

$$= (y - \varphi(z))\,x_j$$

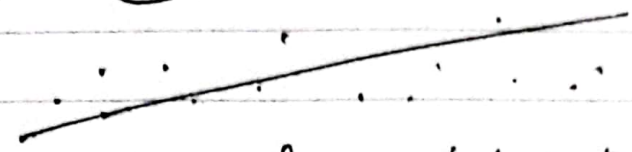
$$w = w + \Delta w$$

$$\Delta w = -\eta \frac{\delta J}{\delta w_c}$$

$$= -\eta \sum_{i=1}^{n} \left(y^{(i)} - \varphi(z^{(i)}) \right) x_j^{(i)}$$

15·03·23

Overfitting →

Underfitting → 

polynomial expected but
st. line is used

In overfitting, the model performs well on training dataset but not in test dataset

→ It is called high variance.
↳ variability
(the model fits whatever data is put into)

Underfitting has high bias

Bias: measures how far of the predictions are from the correct values

Variance: measures the variability/consistancy of the model on different subset of the training data

Regularization

adding information in order to solve an ill-posed problem or to prevent overfitting.

→ tune complexity of the model

→ filter out noise from training data

↓

irregular samples

L2 regularization (L2 shrinkage / weight decay)

$$\frac{\lambda}{2}\|w\|^2 = \frac{\lambda}{2}\sum_{j=1}^{m} w_j^2$$

$$J(w) = \sum_{i=1}^{n}\left[-y^{(i)}\log(\phi(z^{(i)})) - (1-y^{(i)})\log(1-\phi(z^{(i)}))\right] + \frac{\lambda}{2}\|w\|^2$$

$$w_i = w_i - \eta\frac{\partial J}{\partial w_i} - \eta\lambda w_i$$

SVM terminology, $C = \frac{1}{\lambda}$

Support Vector Machines

* optimization objective – maximize margin

Margin: distance between the separating hyperplane and training samples that are closest to this hyperplane

↑ margin, ↓ generalization error

⇒ non linear samples are classified

⇒ support vectors are to be taken in such a way that margin is maximized.

Positive hyperplane
Negative hyperplane

$$w_0 + w^T x_{pos} = 1$$
$$w_0 + w^T x_{neg} = -1$$

distance between 2 planes = $\frac{2}{\|w\|}$; $\|w\| = \sqrt{\sum_{j=1}^{m} w_j^2}$

21.03.23

$w_0 + w^T x_{pos} = 1$
$w_0 + w^T x_{neg} = -1$

Subtracting,

$$w^T(x_{pos} - x_{neg}) = 2 \qquad ; \|w\| = \sqrt{\sum_{j=1}^{m} w_j^2}$$

$$\Rightarrow \frac{w^T(x_{pos} - x_{neg})}{\|w\|} = \frac{2}{\|w\|}$$

We will minimize : $\frac{1}{2}\|w\|$

Slack variable : ξ_i

Allow some ~~variables~~ instances to fall off the margin, but penalize them

Bias / Variance Tradeoff

Main aim of ML is generalization. Not overfitting. Some misclassifaction will be allowed.

Slack variable

→ Soft margin SVM.
→ Helps in regularization
→ For handling outliers and penalize them

New objective to minimize

$$\frac{1}{2}\|W\|^2 + C\left(\sum_i \xi^{(i)}\right)$$

↳ regularization term

C → large value, overfitting occurs
→ small value, reduces overfitting

Non Linear Data (Non separable Classes)

→ Projection into higher dimension to separate classes

$$\phi(x_1, x_2) = z_1, z_2, z_3 = x_1, x_2, x_1^2 + x_2^2$$

Kernel Trick → Similarity Function

It will give the same result of classification by doing dimension shifting.

$$K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi x^{(j)}$$

<u>RBF kernel</u>

Radial Basis Function / Gaussian kernel

$$k(x^{(i)}, x^{(j)}) = \exp\left(-\frac{\|x^{(i)} - x^{(j)}\|^2}{2\sigma^2}\right)$$

$$k(x^{(i)}, x^{(j)}) = \exp\left(-\gamma\|x^{(i)} - x^{(j)}\|^2\right)$$

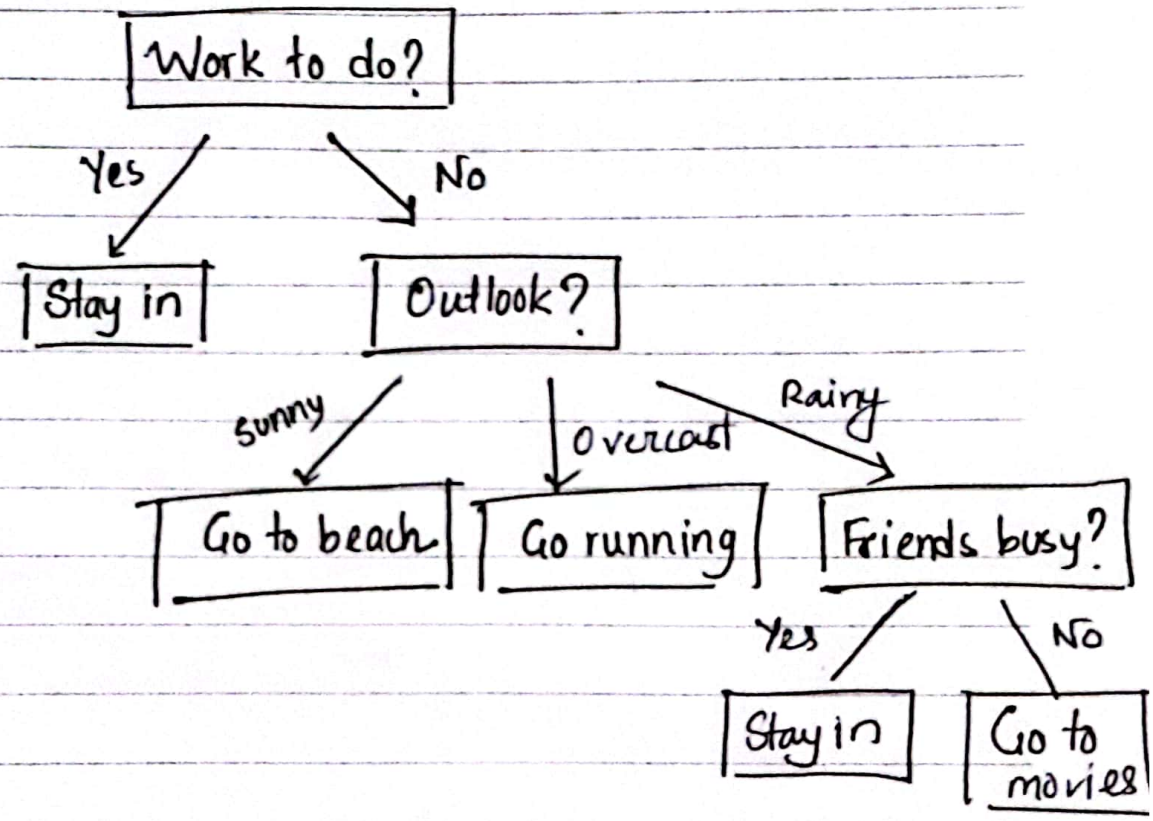
needs to be optimized $\longrightarrow$ $\boxed{\gamma = \frac{1}{2\sigma^2}}$

H.W Comparison of Logistic Regression and SVM

25.03.23

Decision Tree Learning

- works by making decision based on asking a series of questions.



2 types:

① Categorical variable (Yes, No)

② Continuous variable (Sepal width > 2.8 cm)

Algo:

① Start from the tree root [main dataset] split the data based on feature that results in the largest information gain (IG)

② In an iterative process, we can then repeat this splitting procedure at each node until the leaves are pure.

→ belongs to the same class

Information Gain Maximizing

Inorder to split the nodes at the most informative features, we define an objective function:

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

N_j → No. of samples in j^{th} child node

N_p → No. of samples at parent node

D_p → parent dataset

f → feature

$I(D_p)$ → Impurity measure

- Implements binary decision tree

$$IG(D_p, f) = I(D_p) - \frac{N_l}{N_p} I(D_l) - \frac{N_r}{N_p} I(D_r)$$

3 impurity measures / Splitting criteria

1. GINI impurity (I_G)
2. Entropy (I_H)
3. Classification Error (I_E)

Entropy

$$\boxed{I_H(t) = -\sum_{i=1}^{c} P(i|t) \log_2 P(i|t)}$$

→ For a binary tree, $I_H(t) = -p \log_2 p - (1-p) \log_2 (1-p)$
(2 classes)

→ Entropy will be zero if all samples belong to one class, $I_H = 0$.

→ If distribution is uniformly (half-half)

$I_H = 1$

Gini Index

$$\boxed{I_G(t) = \sum_{i=1}^{c} p(i|t) \left(1 - p(i|t)\right)}$$

→ For 2 classes, $I_G(t) = P_+ * (1 - P_+) + (1 - P_-) * (P_-)$

$$\boxed{I_G(t) = 1 - \sum_{i=1}^{c} \left(p(i|t)\right)^2}$$

→ For evenly distribution. $I_G(t) = 1 - \sum_{i=1}^{2} 0.5^2 = 0.5$

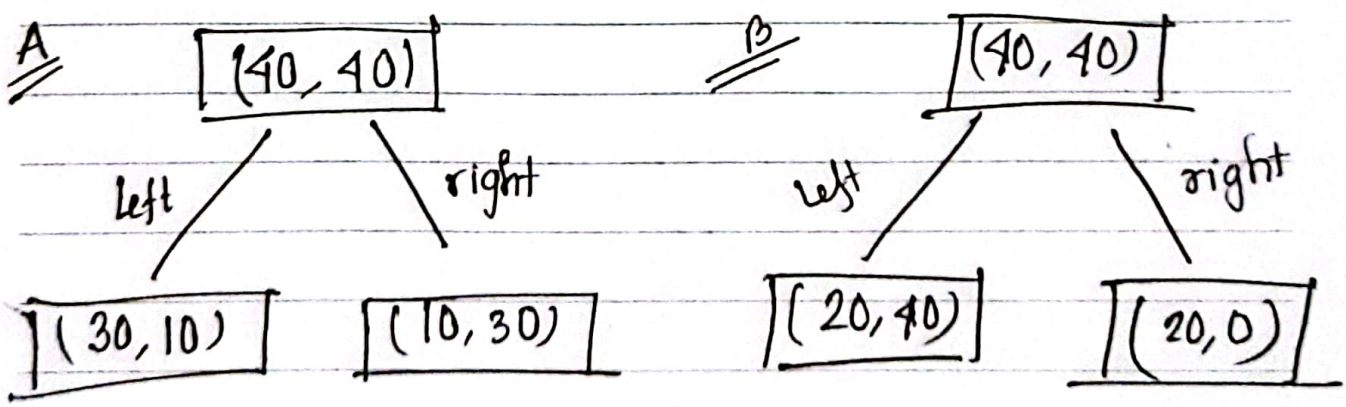
Classification Error

$$I_E = 1 - \max\{p(i|t)\}$$

↳ from all the child nodes, max is taken

For 2 classes : $1 - \max(p, (1-p))$

Example:



A

$$I_G(D_P) = 1 - (0.5^2 + 0.5)^2$$
$$= 0.5$$

$$I_G(D_L) = 1 - \left\{\left(\tfrac{3}{4}\right)^2 + \left(\tfrac{1}{4}\right)^2\right\} = 0.375$$

$$I_G(D_R) = 1 - \left\{\left(\tfrac{1}{4}\right)^2 + \left(\tfrac{3}{4}\right)^2\right\} = 0.375$$

$$IG(D_P, f) = I(D_P) - \frac{N_L}{N_P} I(D_L) - \frac{N_r}{N_P} I(D_r)$$

$$= 0.5 - \frac{4}{8} \times 0.375 - \frac{4}{8} \times 0.375$$

$$= 0.125$$

B

$$IG(D, f) = 0.5 - \frac{6}{8} \times 0.4 - 0 = 0.16$$

A

$$I_H(D_P) = -(0.5 \log 0.5 + 0.5 \log 0.5)$$

$$= 1$$

$$I_H(D_L) = -\left(\frac{3}{4} \log \frac{3}{4} + \frac{1}{4} \log \frac{1}{4}\right)$$

$$= 0.81$$

$$I_H(D_R) = -\left(\frac{3}{4} \log \frac{3}{4} + \frac{1}{4} \log \frac{1}{4}\right)$$

$$= 0.81$$

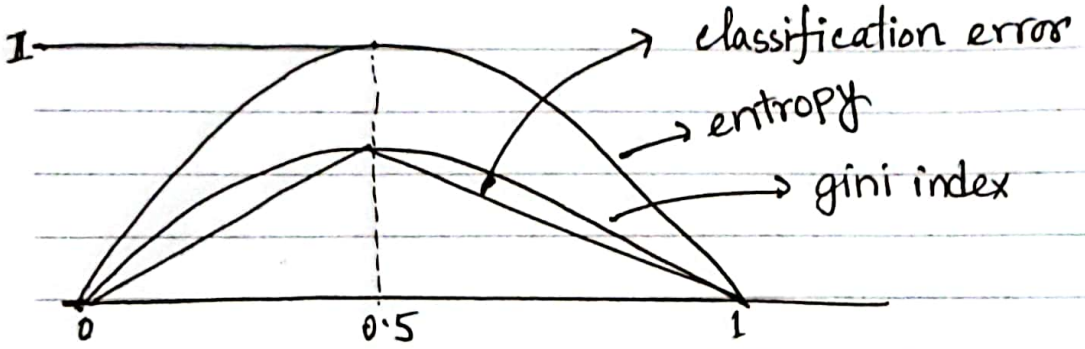
$$IG_H(D_P, f) = 1 - \frac{4}{8} \times 0.81 - \frac{4}{8} \times 0.81$$

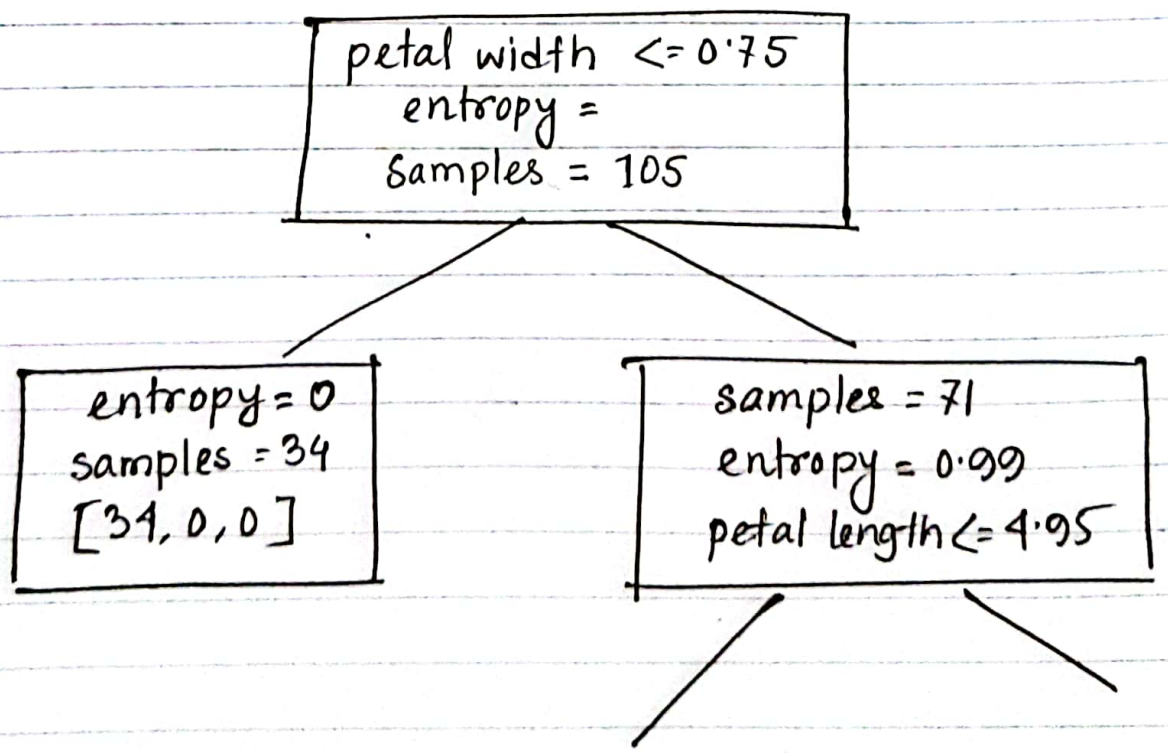
$$= 0.19$$

B

$$IG_H = 1 - \frac{6}{8} \times 0.92 - 0$$
$$= 0.31$$

Split will occur like the B type.





Disadvantage: → the tree can ~~become~~ result in a deep tree.

↳ overfitting

solution

Pruning

→ set a limit for the maximum depth of the tree.

→ allow some misclassification

26.03.23

Random Forest

→ Combining weak to strong learners via random forest
→ gained huge popularity
→ scalability
→ considered as an ~~as~~ ensemble of decision tree
→ combines weak learners to a robust,

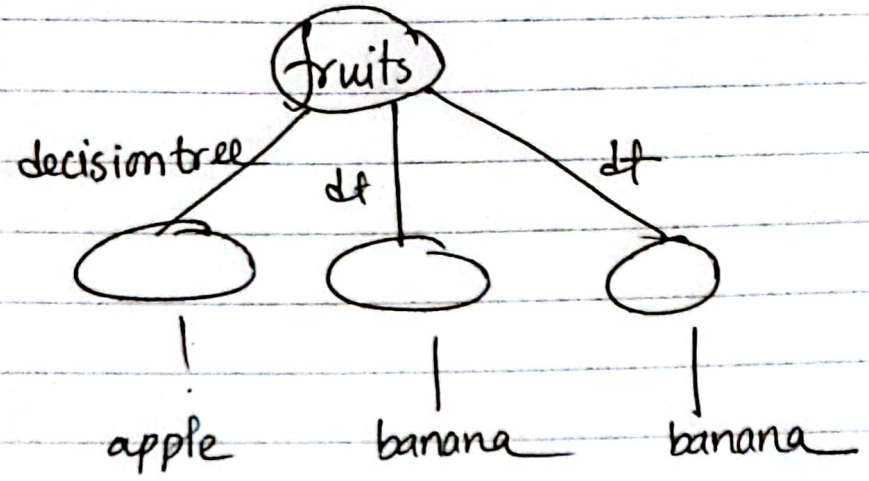
Algo:

① Draw a random bootstrap sample of size n
(. randomly choose n samples from the training set with replacement)
↳ same samples can be used more than once

② Grow a decision tree at
i) Randomly select d features without replacement.
ii) Split the node using the features that provides the best split

③ Repeat the steps 1 to 2 k times
④ Aggregate the prediction by each tree based on majority voting.

e.g



majority → banna

- We don't have to worry so much about choosing good hyper meter values
- Need to fix only k
- We can control the bias-variance trade-off via the sample size n
- Standard value of $d = \sqrt{m}$, m is the no. of features in the training set
- larger the no. of trees, the better the performance at increased computation cost.

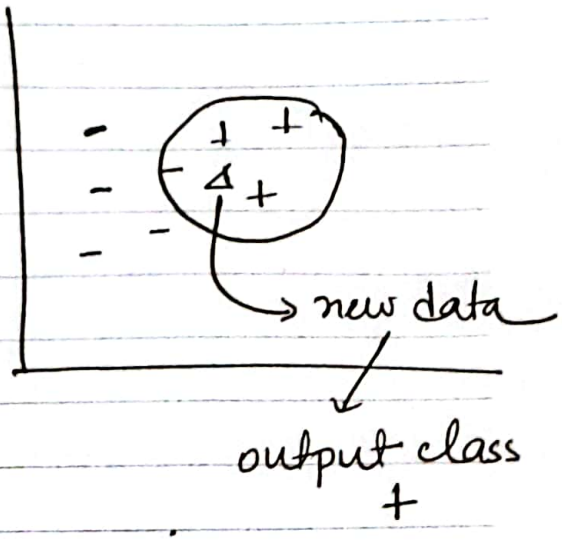
KNN (K-nearest neighbours)

- fundamentally different from the algorithms discussed so far.
- it does not learn a discriminative function
- memorises the training dataset

① Choose the number of k and a distance metric

② Find the k nearest neighbours of the sample that we want to classify

③ Assign the class label by majority voting.



→ The right choice of k is crucial to find a good balance between over and underfitting

→ Affected by outliers

→ needs to apply scaling

Distance Metric

$$d(x^{(i)}, x^{(j)}) = \sqrt{\sum (x_k^{(i)} - x_k^{(j)})^2}$$

↳ Euclidian distance

Minkowski:

$$d(x^{(i)}, x^{(j)}) = \sqrt[p]{\sum |x_k^{(i)} - x_k^{(j)}|^p}$$

$p = 2$, Euclidian
$p = 1$, Manhattam

Parametric Vs Non Parametric model

$\boxed{\text{estimate parameter}}$

ex: Perceptron, SVM, Logistic regression

ex: KNN, decision tree, random forest, kernel SVM

28.03.23

Chapter-04
Building good training sets - Data Preprocessing

Dealing with missing data

NaN - Not a number
↳ error we face often

⇒ Removing the ~~columns~~ rows with missing data
- df.dropna()

⇒ Removing the columns with missing data
- df.dropna(axis=1)

Imputting ~~Imputation~~ of missing values

⇒ Replacing the missing value with mean of the entire column

⇒ Imputter(..)
axis = 0 → row
axis = 1 → column

- Nominal features
- Ordinal features

e.g: XL, L, M, S

Labelencoder() ← for handling the above features.

one-hot encoding | simpliest encoding system 0/1

□ Partitioning dataset into training and test set

train_test_split()

Train - Test :- 70 : 30, 80 : 20, 60 : 40] for small dataset

Large Dataset: 90:10, 99:1

□ After training and evaluation

retrain on the whole dataset

□ Bringing features onto same scale

1. Normalization : data comes into [0, 1] range

- Minmax Scaler ( )

$$x_{norm}^{(i)} = \frac{x^{(i)} - x_{min}}{x_{max} - x_{min}}$$

2. Standardization : convert the feature column at mean 0 with standard deviation 1

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_{mean}}{\sigma_x}$$

- Standard Scaler ( )

29·03·23

Filling out missing values with

- mean
- most frequent
- median

Selecting Meaningful features

Curse of dimensionality - large amount of computations for a large number of features

To combat with overfitting :

- Collect new data
- Introduce a penalty via regularization
- Choose a simpler model with fewer features

Regularization is of 2 types:

- Lasso (L1 regularization)
- Ridge (L2 regularization)

$$L1 = \frac{\lambda}{2} \sum w_i$$

$$L2 = \frac{\lambda}{2} \sum w_i^2$$

In lasso regularization, some w_i becomes zero
↳ reduced dimensionality is achieved.
↳ acts as a feature selection method

Sequential Backward Selection

Sequentially removes features from the full feature set untill the new feature set contains the desired no. of features

- objective function is ~~defined~~ followed
  ↓
  defines whether there is any impact when the feature is removed

04.04.23

Chap – 06

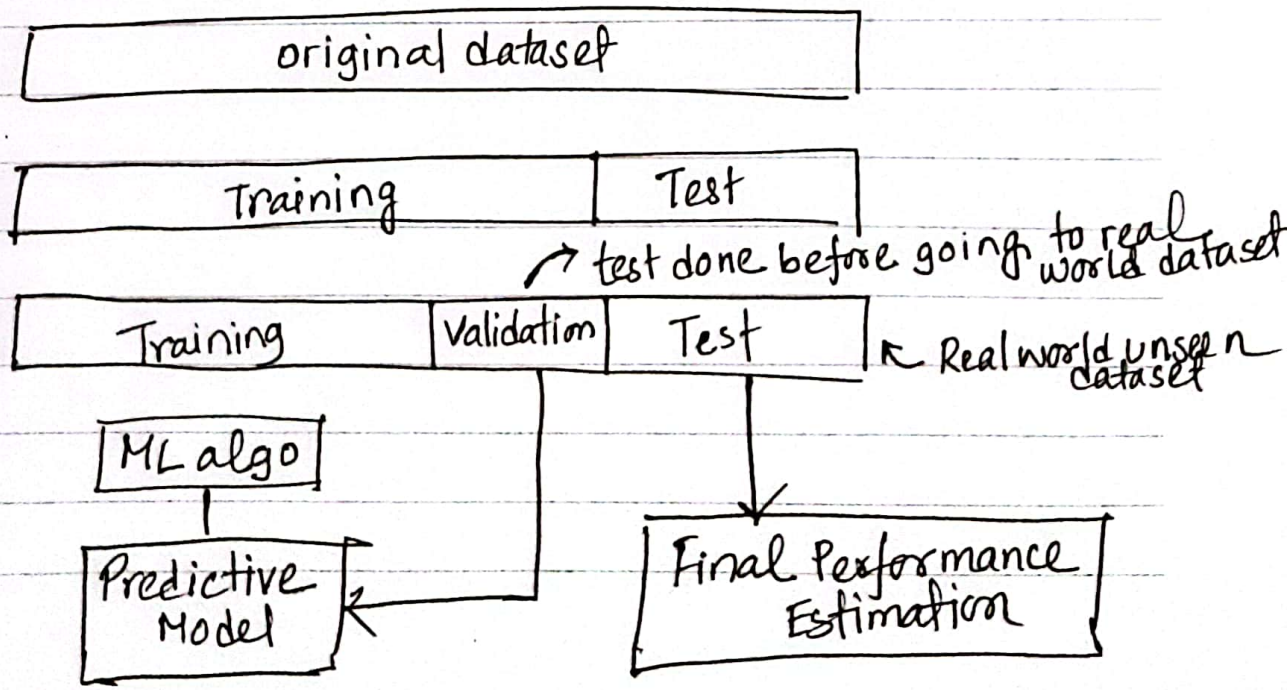
Learning best practices for Model Evaluation and Hyperparameter Tuning.

Using cross validation

↳ Hold out cross validation
↳ k-fold cross validation

Model parameters : w
Hyper parameter : Learning rate, C, γ, depth of decision tree
↳ Tuning parameter

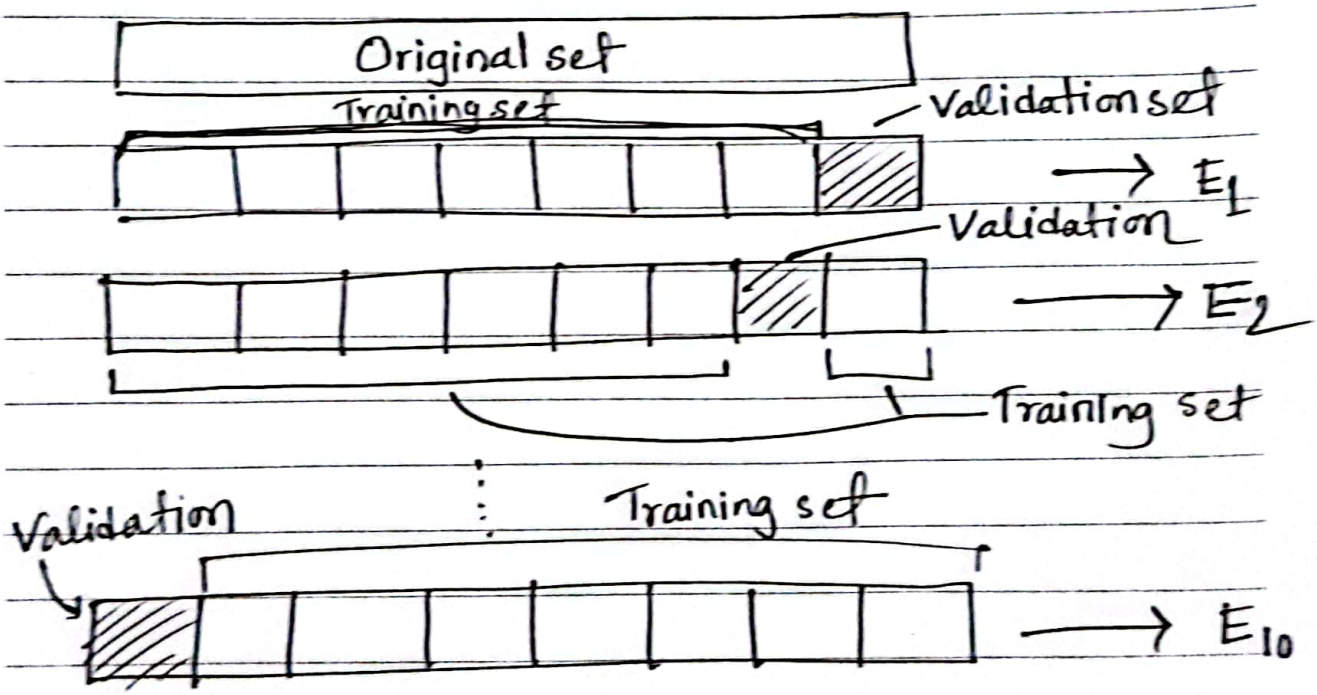
Hold out cross validation



1 fold validation. When tuning is needed. Using the validation set 1 repeatedly gives it a feeling of training set

- k fold cross validation is used.

k fold cross validation



$$\boxed{E = \frac{1}{10} \sum_{k=1}^{10} E_k}$$

For large dataset: $k = 5$

Leave-one-out (LOO)

$k = n$: Each fold contains one training data

→ too expensive

Stratified k-fold cross validation

dataset imbalance — in digit recognition, if a single number has lesser data than other numbers.
0 → 2 data
1-9 → 1000 data

⇒ Training data should have ~~represtan~~ representation of every classes.

→ - Class proportions are preserved in each fold to ensure that each fold is representative of the class proportion in the training set

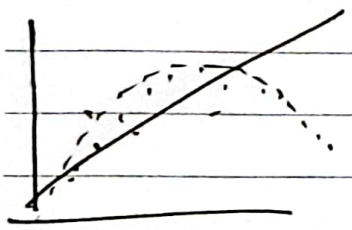
03·05·23

Validation is done to measure the appropiateness of the training dataset (whether the model is ready for real ~~datas~~ world dataset or not)

Bias and Variance Problems

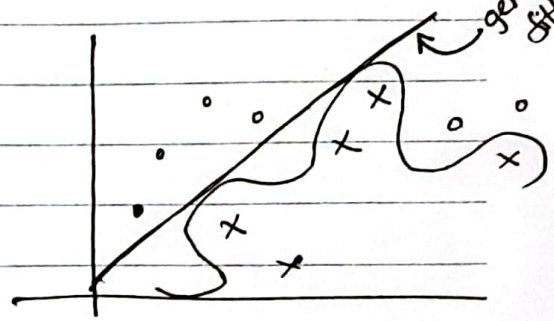
error due to underfit

e.g.: if 2nd order polynomial is fitted using 1st order straigh line



High bias → underfit

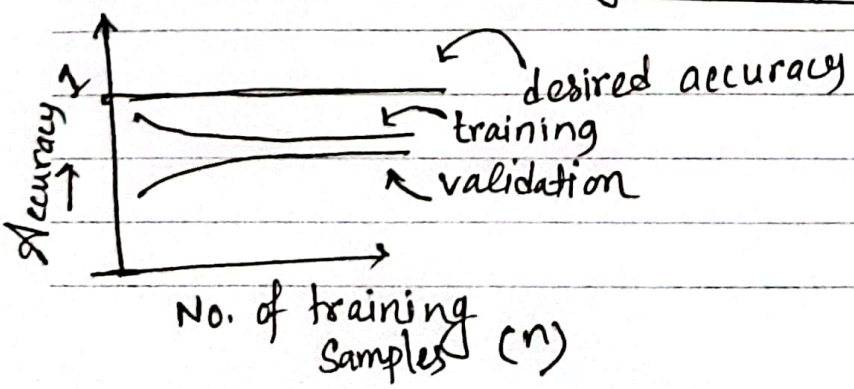
error due to overfit



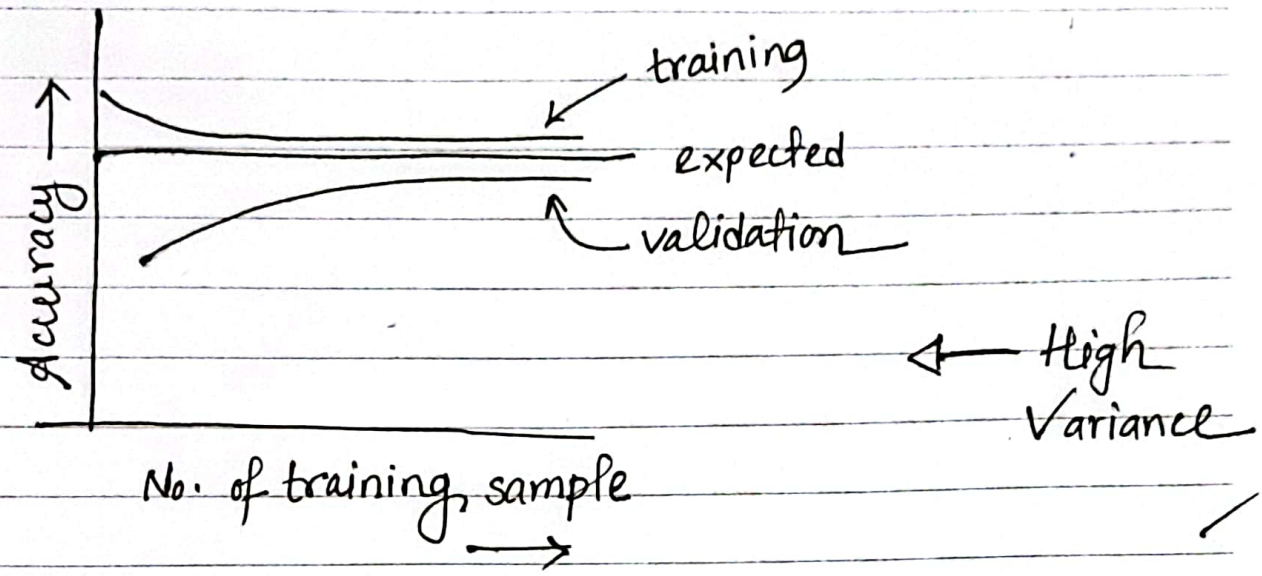
High variance

very complex fitting.

Bias and Variance Using Learning Curve



← High Bias.



To remove overfitting:

→ training samples need to be increased

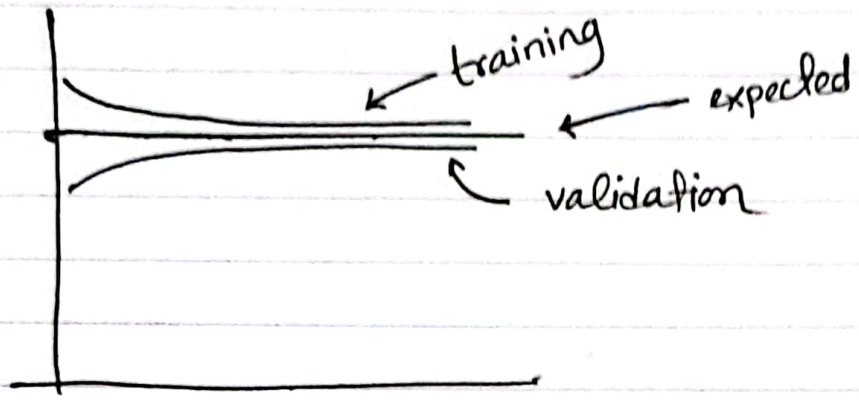
→ Increase regularization

To remove underfitting:

→ include more features

→ decrease regularization

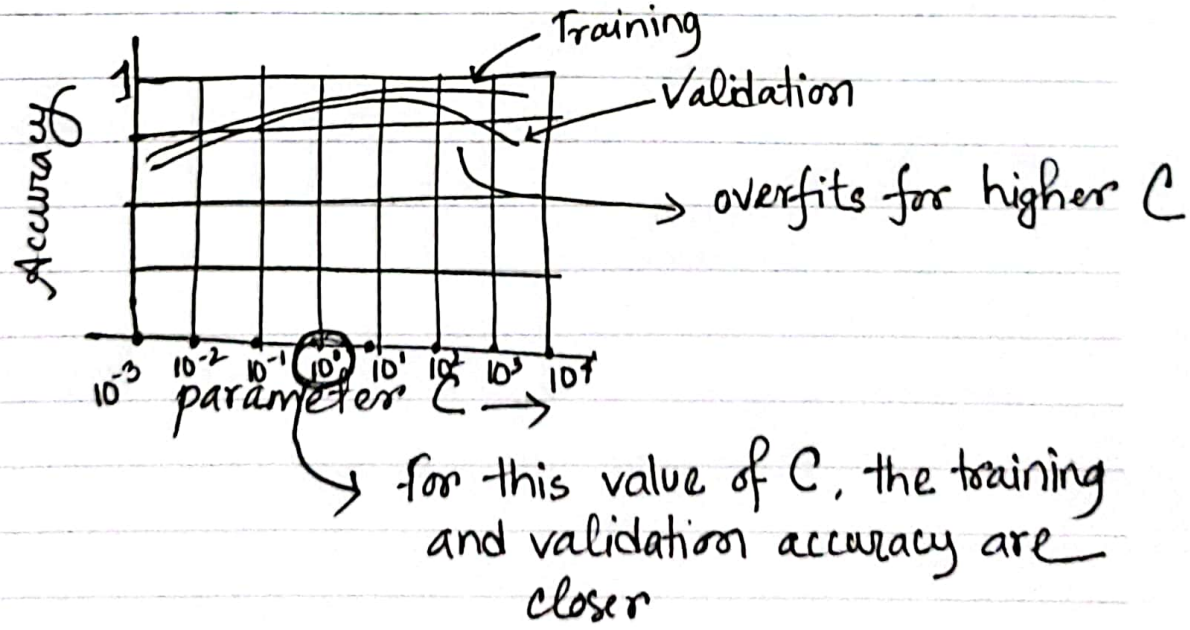
"Machine Learning is more an art than science."



Ideal Situation

Addressing overfitting and underfitting with validation curves

Ex. Logistic Regression : C ← parameter used in regularization



SVM

Linear
RBF
gamma

} select which value is better

weight → W (Learned from training data fitting Normal Parameter.)

regularization parameter → C (Got from tunning → Hyper Parameter → Tuning Parameter)

Fine Tuning ML models via grid search

Hyperparameter search

grid search → brute force exhaustive search

parameter C : 0.0001, 0.001, 0.01, 0.1, 1, 10

best score : Accuracy

07.05.23

Different Performance Evaluation Metrics

Confusion Matrix

A matrix that lays out the performance of a learning algorithm.

$$\text{Accuracy} = \frac{TP + TN}{FP + FN + TP + TN}$$

$TP + TN \rightarrow$ accurate prediction

$FP + FN + TP + TN \rightarrow$ All predictions

Actual \ Predicted	T	F
T	True Positive	False Negative
F	False Positive	True Negative

$$\text{Error} = 1 - \text{Accuracy}$$

$$= \frac{FP + FN}{FP + FN + TP + TN}$$

Precision

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

Recall

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

1) Heart disease detection

2) Bank loan defaulter Prediction

F1 Score

Harmonic average of precision and recall

$$F1 = 2 * \frac{Pre * Rec}{Pre + Rec}$$

ROC (Receiver Operating Characteristics)

$$FPR = \frac{FP}{N} = \frac{FP}{TN+FP}$$ (False Positive Rate)

Want it to be zero

$$TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$$ (True Positive Rate)

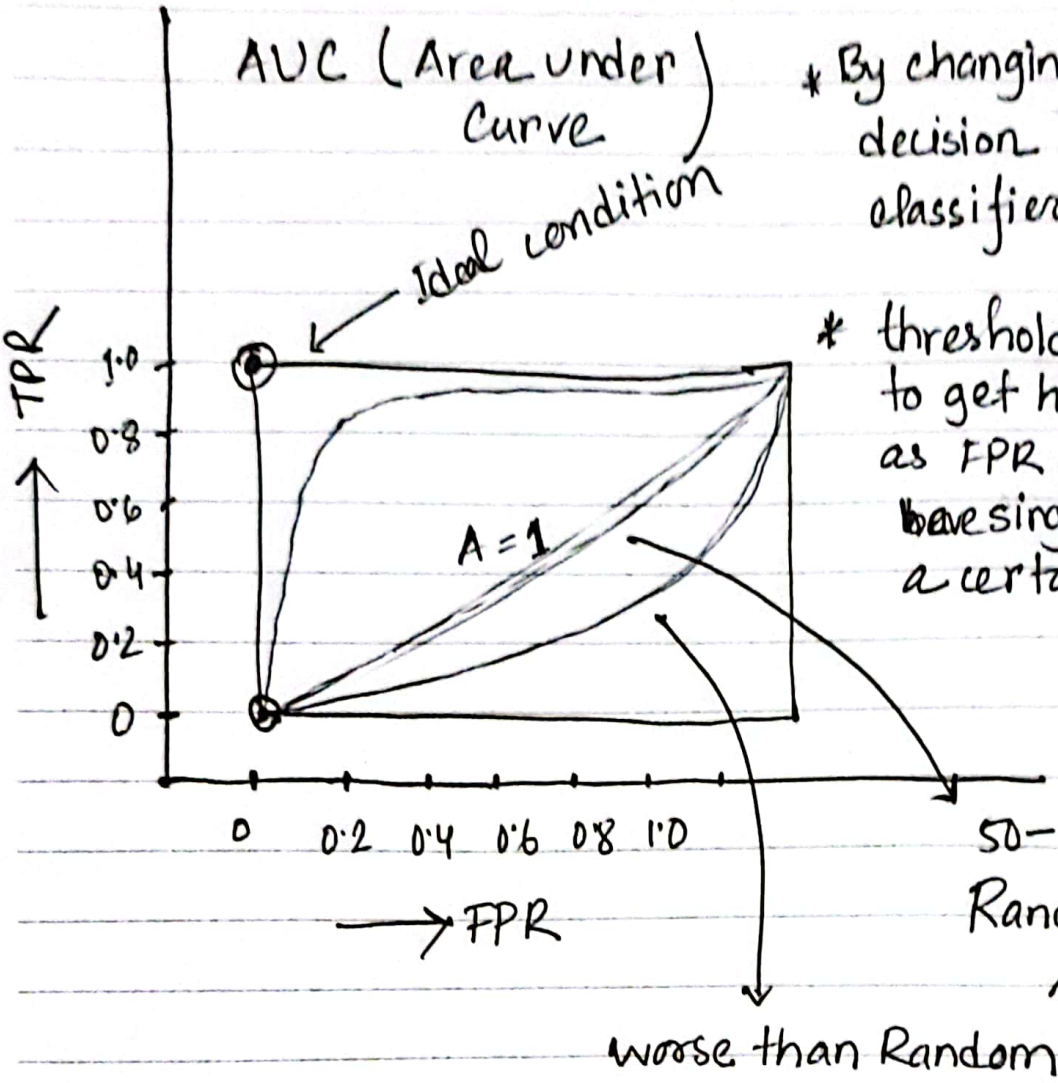
same
- Recall
- Sensitivity

Want it to be 1

Graph is drawn using FPR and TPR

AUC (Area under Curve)

* By changing the decision of the classifier



* threshold is changed to get his graph as FPR and TPR will have single value of a certain model

Value of AUC goes similar to accuracy

Multi class classification

One vs All

Many binary classifications are done.

If there are 3 classes.

First one binary classification is done class-0 vs others.

Secondly → class-1 vs others

Thirdly → class-2 vs others

In this way, multi class classification is done.

$$\text{Precision}_{micro} = \frac{TP_1 + \dots TP_K}{TP_1 + \dots TP_K + FP_1 + \dots FP_K}$$

$$\text{Precision}_{macro} = \frac{Pre_1 + Pre_2 + \dots Pre_K}{K}$$

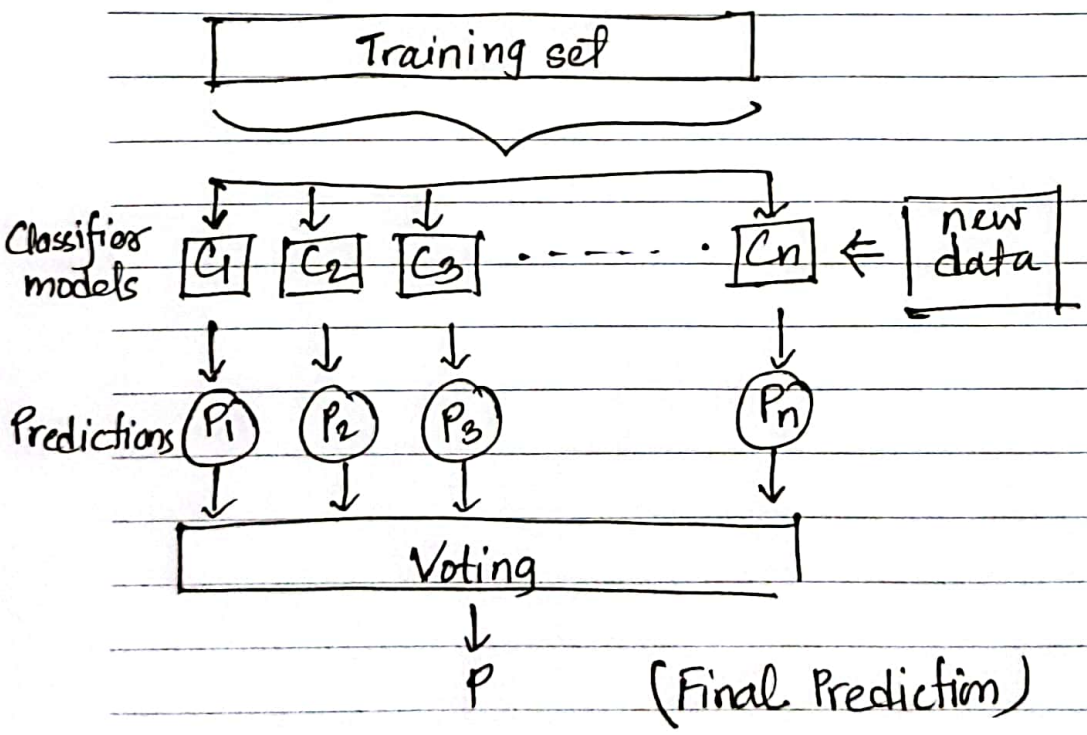
10·05·23

Chapter 07 : Ensemble Learning

The goal of Ensemble methods is to combine different classifiers into a metaclassifier that has a better generalization than each individual classifier alone.

- Majority voting
- Bagging
- Boosting

Majority Voting



Binary classification → majority voting
000000 Δ Δ Δ Δ.

Multi-class classification → plurality voting

sklearn — votingClassifier

$\hat{y} = \text{mode}\{C_1(x), C_2(x), \ldots C_n(x)\}$

$$C(x) = \text{Sign}\left[\sum C_j(x)\right] = \begin{cases} 1, & \text{if } \sum C(x) \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

Iris Dataset

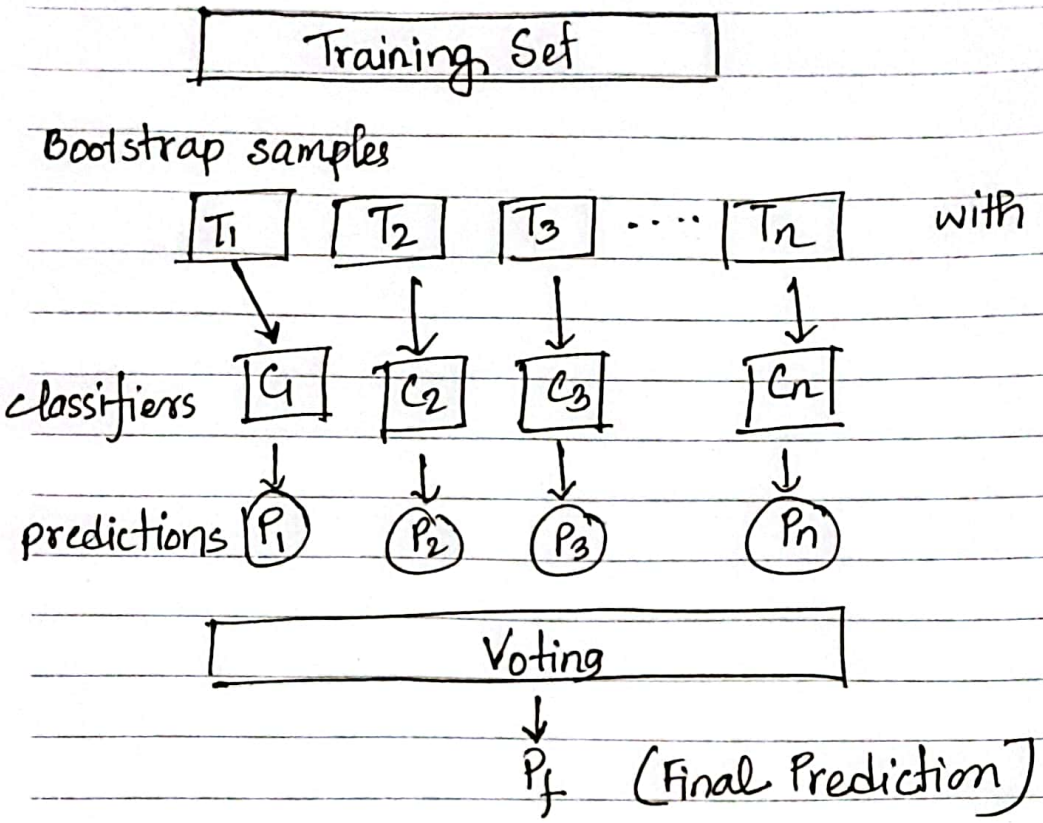
Decision Tree : AUC : 0.89
Logistic Regression : AUC : 0.92
KNN : AUC : 0.86

} 0.95

Bagging

Building an ensemble of classifiers from bootstrap samples.

(1994)



Sample indices	Bagging Round 1	Bagging Round 2	.....
1	2	5	
2	2	3	
3	1	2	
4	3	1	
5	4	1	

Sklearn

Bagging Classifier

→ fights against overfitting

→ Generalization improves

→ Variations in training set

<u>Boosting</u>

→ weak learners are used intentionally.

AdaBoost <u>1996</u>

Adaptive Boosting

15.05.23

Boosting

Adaptive Boosting
— ICML 1996
by Schapire & Freund.

— Goedel Prize, 2003
most significant prize in computer science

Boosting Procedure

1. Draw a random subset of training samples d_1 without replacement from the training set D to train a weak learner C_1

2. Draw a second random training subset d_2 without replacement from the training set and 50% of the samples that were previously misclassified to train a weak learner C_2

3. Find the training sample d_3 in the training set D on which C_1 & C_2 disagree to train a third weak learner C_3

4. Combine the weak learners: C_1, C_2, C_3

via majority voting

e.g.:

$C_1 \rightarrow 1$, $C_2 \rightarrow -1$, $C_3 \rightarrow 1$

$\downarrow$

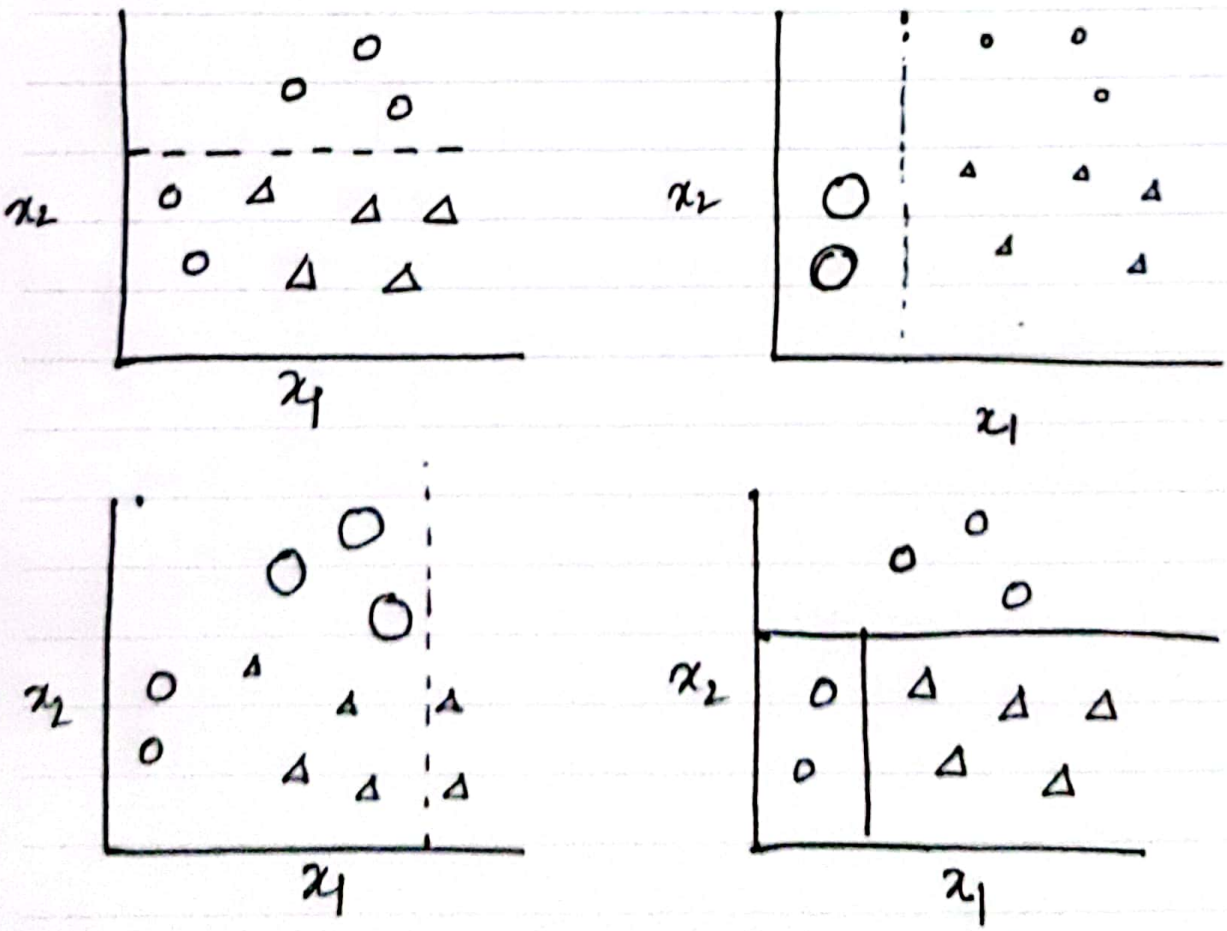
1

$D \rightarrow d_1 \rightarrow C_1$

$D \rightarrow d_2 \rightarrow +50\% \rightarrow C_2$

weak classifer $\rightarrow$ better than random classifier

AdaBoost



final prediction by majority voting

AdaBoost

1. Set weight vector w to uniform weights where $\sum_i W_i = 1$

2. For j in m boost rounds, do the following:

   3. Train a weighted weak learner : $C_j = \text{train}(x, y, w)$
   4. Predict class labels : $\hat{y} = \text{predict}(C_j, x)$
   5. Compute weighted error rate :
      $\epsilon = w \cdot (\hat{y} == y)$
      $0 \rightarrow$ if true
      $1 \rightarrow$ if false
   6. Compute coefficient : $\alpha_j = 0.5 \log \frac{1-\epsilon}{\epsilon}$
   7. Update weights : $w = w \times \exp(1 - \alpha_j \times \hat{y} \times y)$
   8. Normalize weights to sum to 1 :
      $$w = \frac{w}{\sum_i w_i}$$

9. Compute final predication :
$$\hat{y} = \left(\sum_{j=1}^{m} (\alpha_j \times \text{predict}(C_j, x)) > 0 \right)$$

X	Y	W	$\hat{y}$ $x <= 3.0$	Correct?
1.0	1	0.1	1	Y
2.0	1	0.1	1	Y
3.0	1	0.1	1	Y
4.0	-1	0.1	-1	Y
5.0	-1	0.1	-1	Y
6.0	-1	0.1	-1	Y
7.0	1	0.1	-1	N
8.0	1	0.1	-1	N
9.0	1	0.1	-1	N
10.0	-1	0.1	-1	Y

$\varepsilon = W \cdot (\hat{y} == y)$ ↳ 1 happen 3 times

$= 0.3$

$\alpha_j = 0.5 \log \frac{1-\varepsilon}{\varepsilon} \approx 0.424$

$W = W \times \exp(-\alpha_j \times \hat{y} \times y)$

vector multiplication

for correctly classification $= 1$

will have much small value as exp of negative number gives small value

$= 0.1 \times \exp(-0.424 \times 1)$

≈ 0.065

(reduced in case of correctly classification)

For incorrect classification.

$W = 0.1 \times \exp(-0.424 \times -1)$

≈ 0.153

Increased for misclassified samples.

↳ AdaBoost works in a similar way to Bagging

↳ AdaBoost also suffers from overfitting

- Enessemble learning increases computational complexity.
- An often modest improvement of predictive performance

2009, Netflix → competition
won by Ensemble teachnique.
But Netflix never applied this technique.

21.05.23

Chapter - 08

Applying ML in Sentiment Analysis

NLP : Natural Language processing

① Text processing

Cleaning and Preparing text data

IMDb dataset (Review of movies)

Bag of words (for converting text into number)

Input : "The sun is shining"
"The weather is sweet"
" The sun is shinning, the weather is sweet and one and one is true."

Token Vectorizer

dictionary.

'and' : 0	'sun' : 4
'two' : 7	'weather' : 8
'shining' : 3	'the' : 6
'one' : 2	'sweet' : 5
	'is' : 1

$$\begin{bmatrix} [0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0] \\ [0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1] \\ [2 & 3 & 2 & 1 & 1 & 1 & 2 & 1 & 1] \end{bmatrix}$$ ⇒ feature vector

$\boxed{0}$ $\boxed{\text{and}}$ is present twice in 3rd document

number of words present in the documents according to the index of the dictionary.

- Use the most frequently used ~~ES~~ words

para = 20000

N. gram : Unigram | The cat sat on the mat

2-gram | "The cat", "cat sat", "sat on", "on the", "the mat"

Disadvantage:

→ Sequence is Lost

Advantage:

→ Easy

local sequence info exists in N-gram approx

One hot encoding

For categorical data, encoding is done

Male → 0

Female → 1

	Green	Red	Blue
Green	1	0	0
Red	0	1	0
Blue	0	0	1

Label encoding

Green — 0
Blue — 1
Red — 2
⋮ ⋮

Assessing Word Relevancy

Term frequency-inverse document frequency

$\boxed{tf \cdot idf(\ )} = tf \times idf$

$tf(t, d)$ - the number of times term t is present in document d

$tf(\text{"is"}, d_1) = 1$

$$idf(t, d) = \log \frac{n_d}{1 - df(d,t)}$$

$n_d \rightarrow$ no. of documents

$df(d,t)$ = document frequency = no. of document in which term t present.

unimportant word will have $tf \cdot idf$ close to zero.

In scikit learn,

$$tf \cdot idf(t,d) = tf(t,d) \times (idf(1 + idf(t,d)))$$

$$V_{norm} = \frac{V}{\|V\|_2} = \frac{V}{\sqrt{V_1^2 + V_2^2 + \dots V_n^2}}$$

② <u>Cleaning the text</u>

① is, seven,

 Title ..

② stop word removal; and

③ Removing capitalization

④ Stemming : run running

⑤ Lemmatizing ;
grammatically correct form ⟹ আচ্ছি → আছয়া

finding the root word.

discard

emoticons can not be discarded

21.05.23

Chapter 10

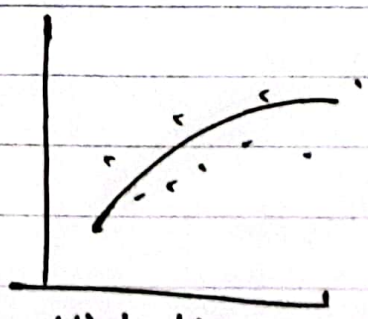
Supervised Learning

- Classification: Logistic Regression, D.T. SVM
- Regression Analysis
  - Regression models are used to predict target variable on a continuous scale ⇒ for example: Given data temp for 1 year, temp prediction for 1 month

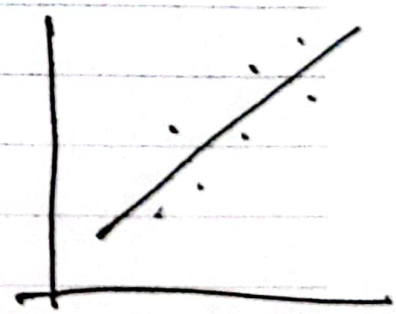
Regression :

Linear regression : Fitting best line by measuring distance from all points

regression
↓
curve fitting



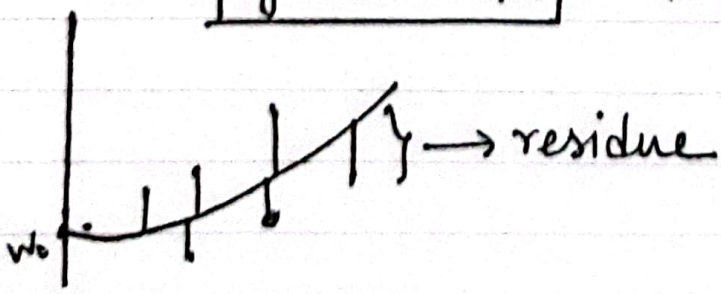
multiple linear regression



simple linear regression

Eq^n of linear model with one variable :

$$\boxed{y = w_0 + w_1 x} \rightarrow y = mx + c$$



$$w_1 = \frac{\Delta y}{\Delta x} \text{ (slope)}$$

Best fit: The line which has minimum distance from all points in total.

Regression line

Residual → error; distance from main line to actual point.

Goal → to learn the weights (w_0, w_1)

Multiple Linear Regression:

$$y = w_0 x_0 + w_1 x_1 + \dots w_m x_m = \sum_{i=0}^{m} w_i x_i = w^T x$$

$x_0 = 1$, w_0 = intercept

Exploring the housing dataset:

sample 506

Feature:

CRIM: per capita crime rate

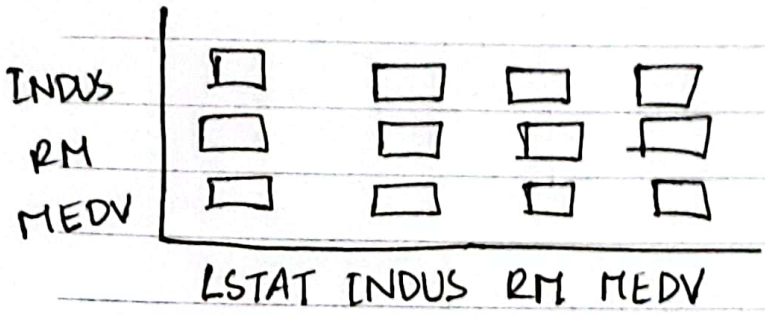
INDUS: proportion of non retail business acres/town

RM: No. of room per dwelling acres/town

⋮

MEDV : Median value in \$1000

Visualize the important characteristics:



import seaborn as sns
sns.pairplot(df[cols])

* most linear relation: RM and MEDV
* pick the feature which is ~~cho~~ close to linear

2nd rule : Correlation Matrix

cross correlation, correlation and covarience. heatmap

Heatmap

- RM and LSAT
- RM and MEDV

} good relation

It means RM is linear.

To quantify linear relationship

correlation matrix can be used.

$$P = \frac{\sum (x^{(i)} - \mu_x)(y^{(i)} - \mu_y)}{\sqrt{\sum (x^{(i)} - \mu_x)^2 (y^{(i)} - \mu_y)^2}}$$

From the correlation matrix and scatter plot:
RM – MEDV (linearly related)

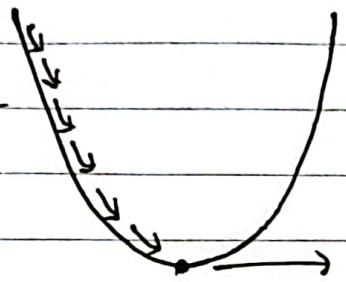
Solving regression for parameters

$J(w) = \frac{1}{2} \sum (y^{(i)} - y^{n(i)})^2$

we need to minimize difference betn predicted and actual value

Gradient descent is used to minimize error

~~int~~ iterative method



→ J(w) → error

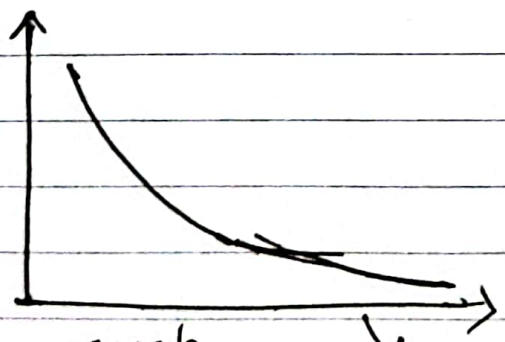
→ expected w ; derivative 0

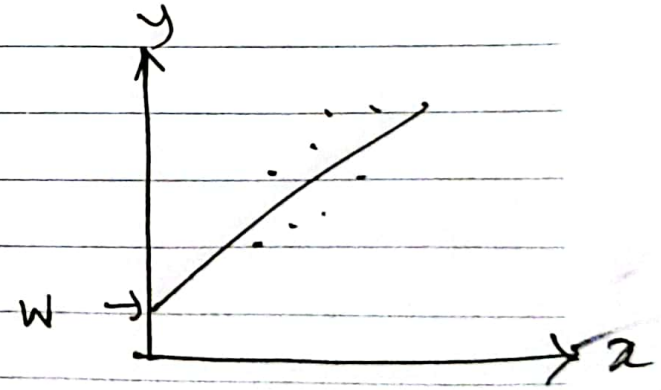
Cost minimization

using gradient descent

$w_1 = w_0 - \text{derivative } [J(w)]$

updated prev





an alternate sol^n without gradient descent

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

↳ closed form sol^n

converged means almost minimum. More minimum not possible

31.05.23

Linear Regression

Normal equation

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

Adj: Guranteed to get an optimal solution

Disadj: 1. For singular matrix, no solution is obtained

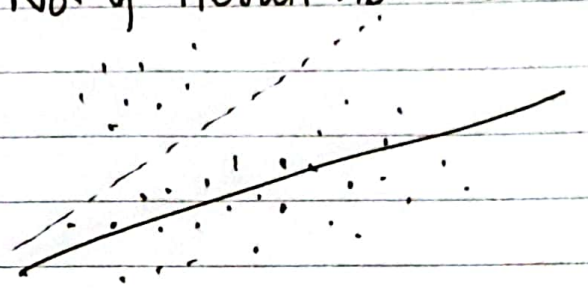
2. For large dataset, it is computationally very expensive

RANSAC (Random Sample Concensus)

Outliers

1. Select a random number (50) of samples to be inliners and fit the model.
2. Test all other data points against the fitted model and add those points that fall within a user-given tolerance to the inlines.
3. Refit the model using all inliners
4. Estimate the error of the model versus the inliners

5. Terminate the algo if the performance meets a certain user-defined threshold or go back to step 1 → No. of iterations = 100



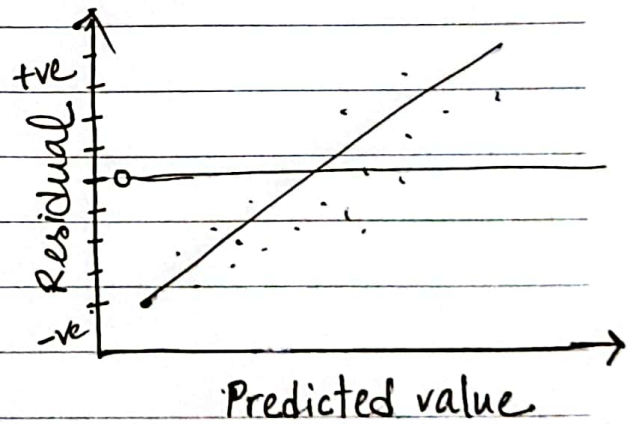
Evaluating the performance

Multiple ~~linear~~ Regression

$$y = w_0 x_0 + w_1 x_1 + \ldots\ldots w_m x_m$$

MSE (Mean Square Error)

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



Regularization for Linear Regression

L2 → Ridge Regression

L1 → LASSO (Least Absolute Shrinkage and

Selection Operator)

$$J(w)_{Ridge} = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2 + \lambda \|w\|_2^2$$

$$L2: \lambda \|w\|_2^2 = \lambda \sum_{j=1}^{m} w_j^2$$

some weights will be reduced by weakening the coefficients of some orders of polynomial

$$J(w)_{LASSO} = \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda \|w\|_1$$

$$L1: \lambda \|w\|_1 = \lambda \sum_{j=1}^{m} w_j$$

effect much in zeroing weights.
- sparse will be created.

$$\boxed{J(w)_{Elasticnet} = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2 + \underbrace{\lambda \|w\|_1}_{\text{reduce weights to zero}} + \underbrace{\lambda \|w\|_2^2}_{\text{recover some}}}$$

↳ Combining ridge and lasso.

Elasticnet

Polynomial Regression

$$y = w_0 + w_1 x + w_2 x^2 + \ldots\ldots w_d x^d$$

04.06.23

Chapter - 11

Working with Unlabled data - Clustering Analysis

- to discover the hidden structure in data where we donot know the right answer upfront.

e.g.: K-means technique

- easy to compute
- difficult to find (K)

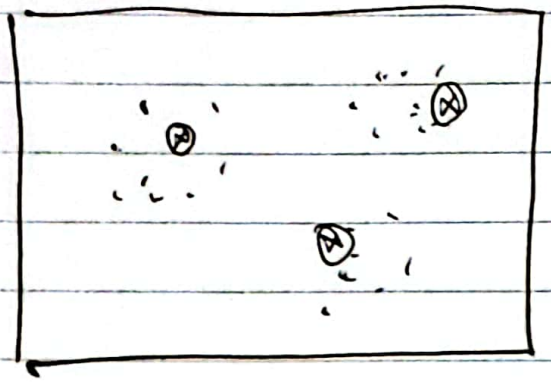
3 types of clustering:

⇒ Prototype based clustering
⇒ Hierarichal clustering
⇒ Density based clustering

Prototype Based Clustering

cluster, centroid are taken before start of computation.

Then mean distance is found in every ~~int~~ iterations



and the centroid converges.

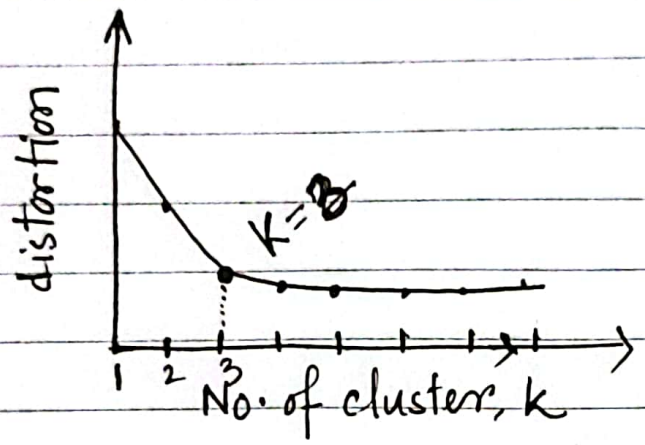
K-means algorithm

1. Randomly pick k centroids from the sample-points as initial cluster centers.
2. Assign each sample to the nearest centroid $\mu^{(j)}$ $j \in \{1, \ldots k\}$
3. Move the centroids to the center of the samples that were assigned to it.
4. Repeat steps 2 and 3 until the cluster assignments do not change or a user defined tolerance or a max number of iterations is reached.

$$SSE = \sum_{i=1}^{n} \sum_{j=1}^{k} w^{(i,j)} \| x^{(i)} - \mu^{(j)} \|_2^2$$

Disadj: Finding the value of K, the number of clusters.

Elbow Method (to identify the value of k where the distortion begins to decrease more rapidly)



If points are closer, then convergence becomes slower

K- means ++

- Run k-means multiple times on a data and select the best performing model.
- Initialize the centroids far away from one another.

07.06.23

Hard Vs Soft Clustering

1 sample in 1 cluster
n clusters.

sample may be in one or more clusters

Hard clustering describes a family of algorithm where each sample in a dataset is assigned to exactly one cluster. (like in k-means)

Algorithms for soft clustering assign a sample to one or more cluster

- Fuzzy C means (FCM)

sample x → cluster:

$\mu^{(1)} \to 0.1$

$\mu^{(2)} \to 0.85$

$\mu^{(3)} \to 0.05$

$\mu^{(1)} \to 0$

$\mu^{(2)} \to 1$

$\mu^{(3)} \to 0$

k-means

Fuzzy C means

Algorithm

① Specify the number of k centroids and randomly assign the cluster memberships for each point.

② Compute the cluster centroids $\mu^{(j)}$, $j \in \{1, \dots k\}$

③ ~~Repeat~~ Update the class memberships for each point

④ Repeat steps 2 and 3

$x^{(i)} \rightarrow \mu^{(j)}$

$w^{(i,j)}, \in (0,1)$ | 1 if it is its own cluster, 0 otherwise

$$w^{(i,j)} = \left[\sum_{p=1}^{k} \left(\frac{\|x^{(i)} - \mu^{(j)}\|_2}{\|x^{(i)} - \mu^{(p)}\|_2} \right)^{\frac{2}{m-1}} \right]^{-1} \qquad m = 2 \sim \infty$$

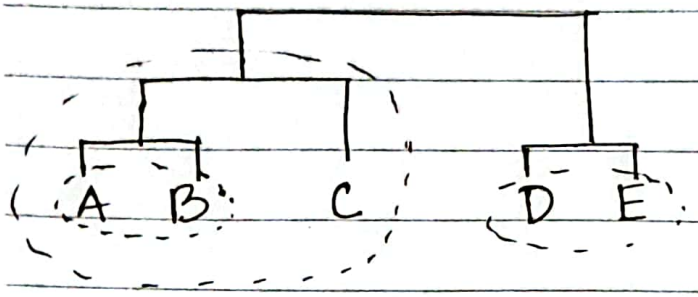
$$\mu^{(j)} = \frac{\sum_{i=1}^{n} w^{m(i,j)} x^{(i)}}{\sum_{i=1} w^{m(i,j)}}$$

Hierarchical Cluster

Dendrogram

Hierarchical Structure

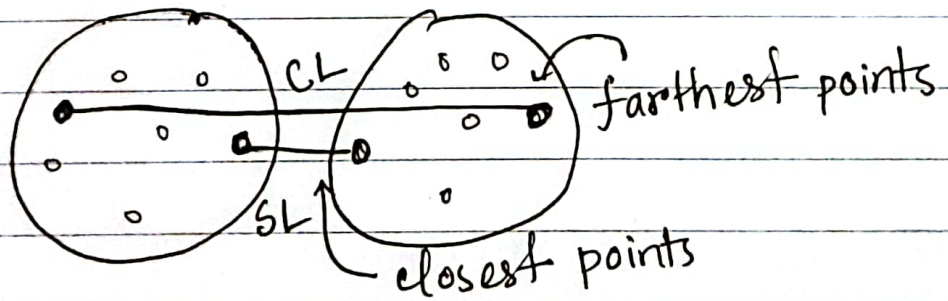
- Leaf
- root



Hierarchical Cluster

- → Agglomerative cluster
- → Divisive
  - Single Link
  - Complete Link

Using Single Linkage, we compute the distances between the most similar members for each pair of clusters and merge the clusters for which the distance between the most similar member is the smallest



Algo:

1. Compute the distance matrix of all samples.
2. Represent each data point as a singleton cluster.

③ Merge the two closest clusters based on the distance of the most similar (close) members / most disimilar (far) members.

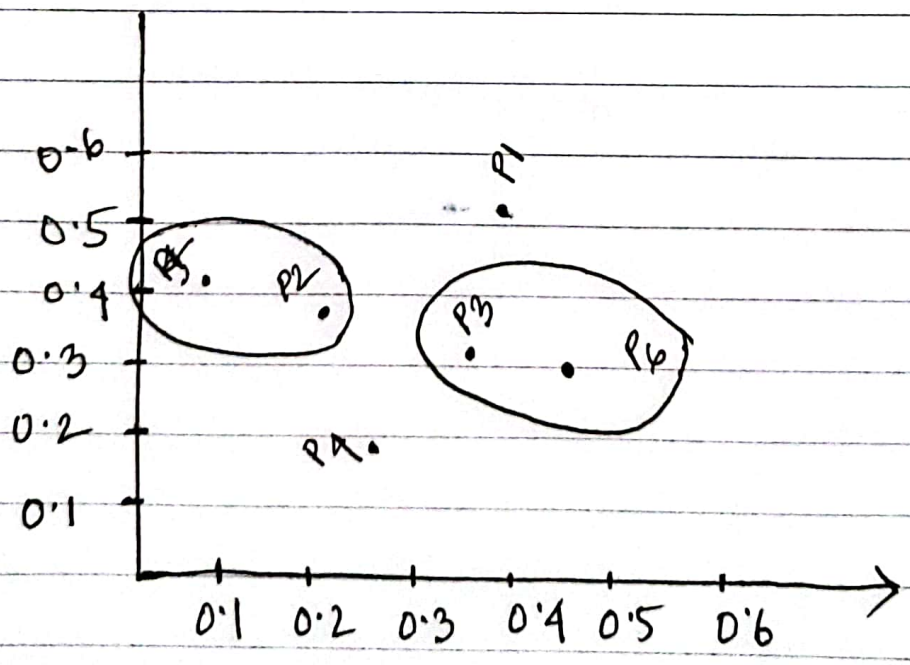
④ Update the distance matrix

⑤ Repeat steps 2 to 4

Example

	X	Y
P_1	0.40	0.53
P_2	0.22	0.38
P_3	0.35	0.32
P_4	0.26	0.19
P_5	0.08	0.41
P_6	0.45	0.30

	P1	P2	P3	P4	P5	P6
P1	0					
P2	0.23	0				
P3	0.22	0.15	0			
P4	0.37	0.20	0.15	0		
P5	0.34	0.14	0.28	0.29	0	
P6	0.23	0.25	$\boxed{0.11}$	0.28		0



$$D(P_1, P2) = \sqrt{(0.40-0.22)^2 - (0.53-0.38)^2}$$
$$= 0.23$$

	P1	P2	P3P6	P4	P5
P1	0				
P2	0.23	0			
P3 P6	0.22	0.15	0		
P4	0.37	0.20	0.15	0	
P5	0.34	$\boxed{0.14}$	0.28	0.29	0

min distance Point ↔ (P3 P6) are taken

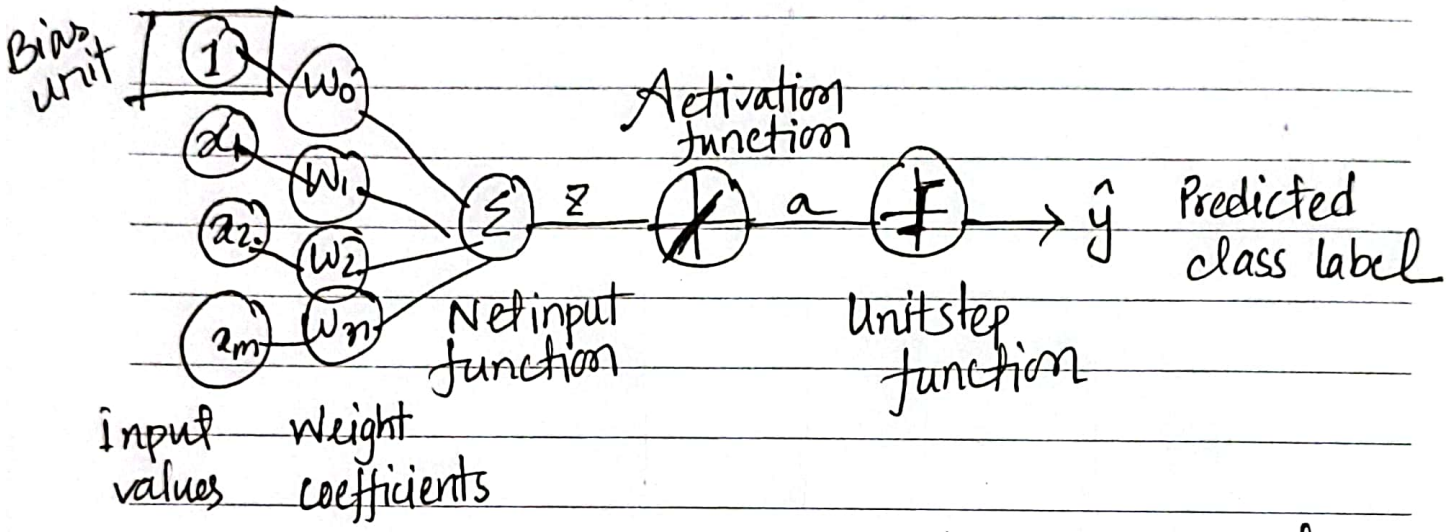
	P1	P2 P5	P3 P6	P4
P1	0			
P2 P5	0.23	0		
P3 P6	0.22	0.15	0	
P4	0.37	0.20	0.15	0

P3 P6 P2 P5

11·06·23

Chap 12

Artificial Neural Network



learning parameter → w

Perceptron

$$\Delta w_j = \eta\left(y^{(i)} - \hat{y}^{(i)}\right)x_j^{(i)}$$

Adaline

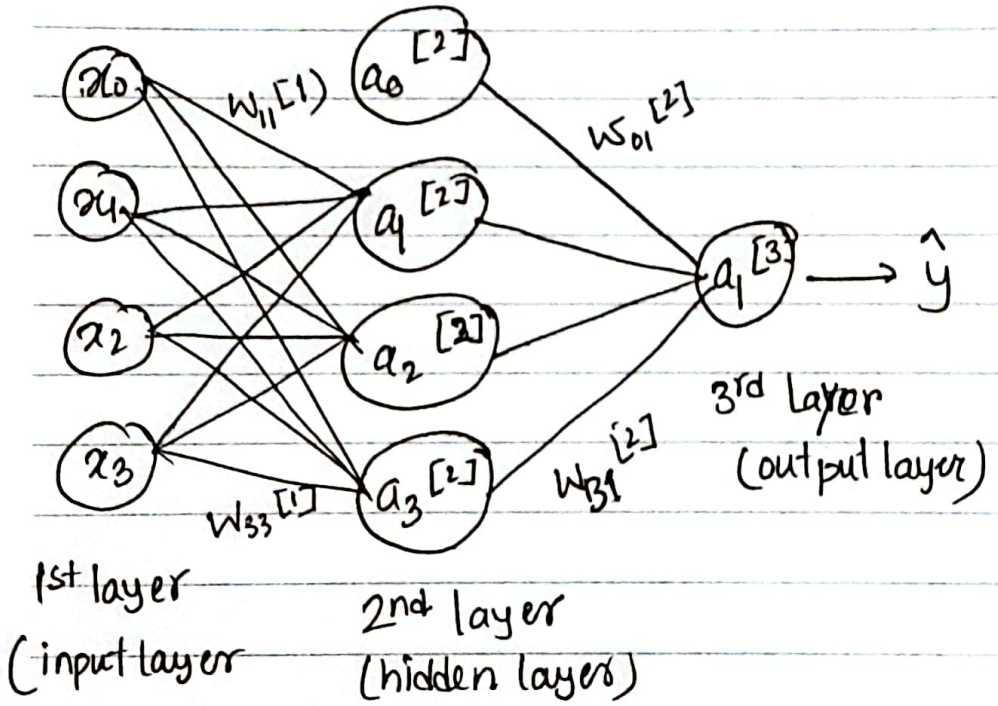
$$w := w + \Delta w \qquad \Delta w = -\eta \nabla J(w)$$

$$\frac{\delta}{\delta w_j} J(w) = \sum_i \left(y^{(i)} - a^{(i)}\right) x_j^{(i)}$$

$$\phi(z) = z = a$$

step function is not differential
So, we need a continuous curve for allowing 'learning' to happen.

Multi layer Feedforward neural network



No. of hidden layer $> 1 \longrightarrow$ deep learning
otherwise $\longrightarrow$ shallow learning

Fully connected : dense network. Every input is connected with the hidden layer.

MLP → Multi layer Perceptron

epoch → no. of iterations over dataset.

$$z_1^{(2)} = a_0^{(1)} \cdot w_{1,0}^{(1)} + a_1^{(1)} w_{1,1}^{(1)} + \cdots + a_m^{(1)} w_{1,m}^{(1)}$$

$$a_1^{(2)} = \varphi(z_1^{(2)})$$

$$J(W) = -\left[\sum_{i=1}^{n} y^{(i)} \log(a^{(i)}) + (1-y^{(i)}) \log(1-a^{(i)})\right] + \frac{\lambda}{2} \|W\|_2^2$$

14·06·23

Back propagation

$$\frac{\delta J(W)}{\delta w_{j,i}^{(l)}}$$

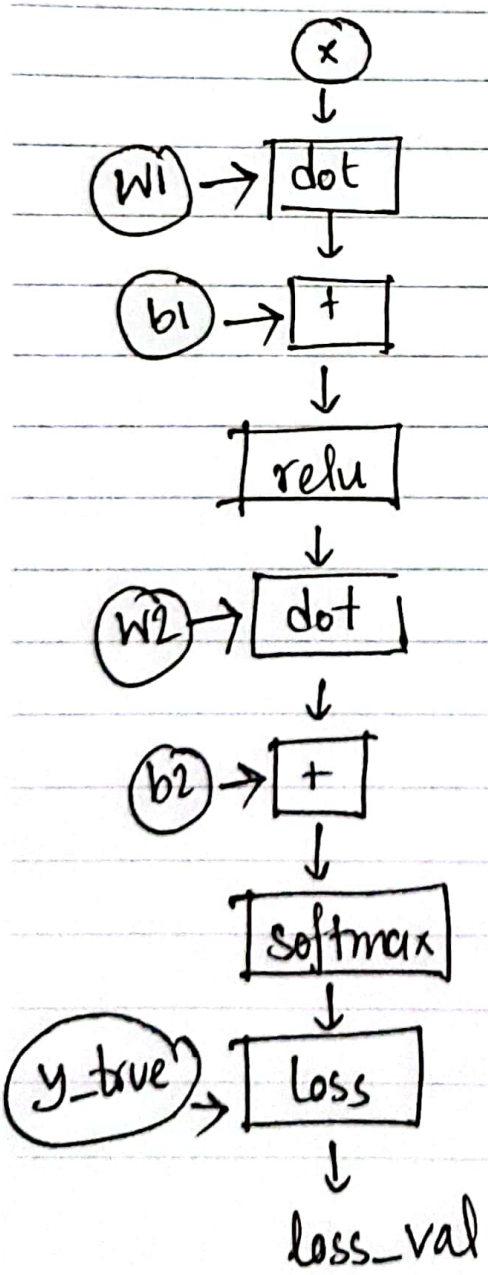
$\delta J(W)$ ← vector/matrix

$\delta w_{j,i}^{(l)}$ ← scalar

individual weights

change of cost function with respect of weights

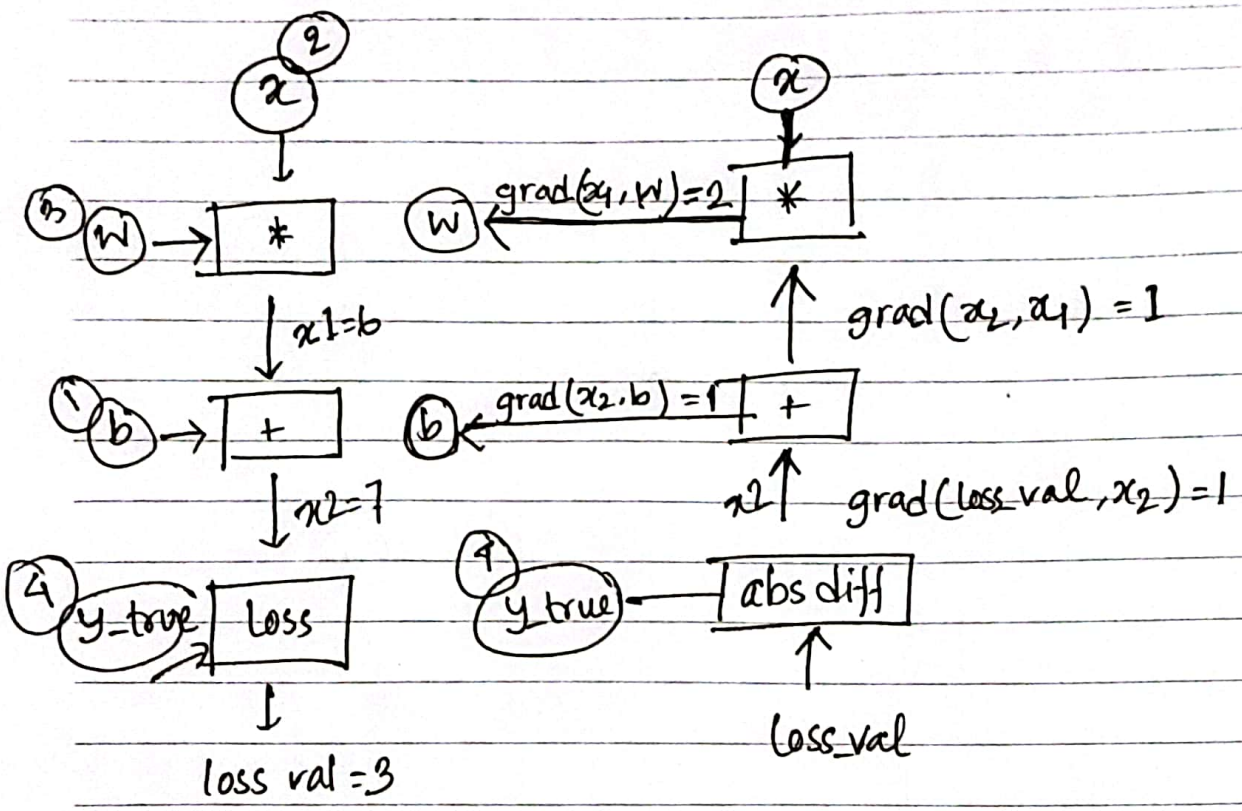
Computation Graph ~~for backpropagation~~



softmax → activation function used for multiclass

Forward Pass

Backward Pass



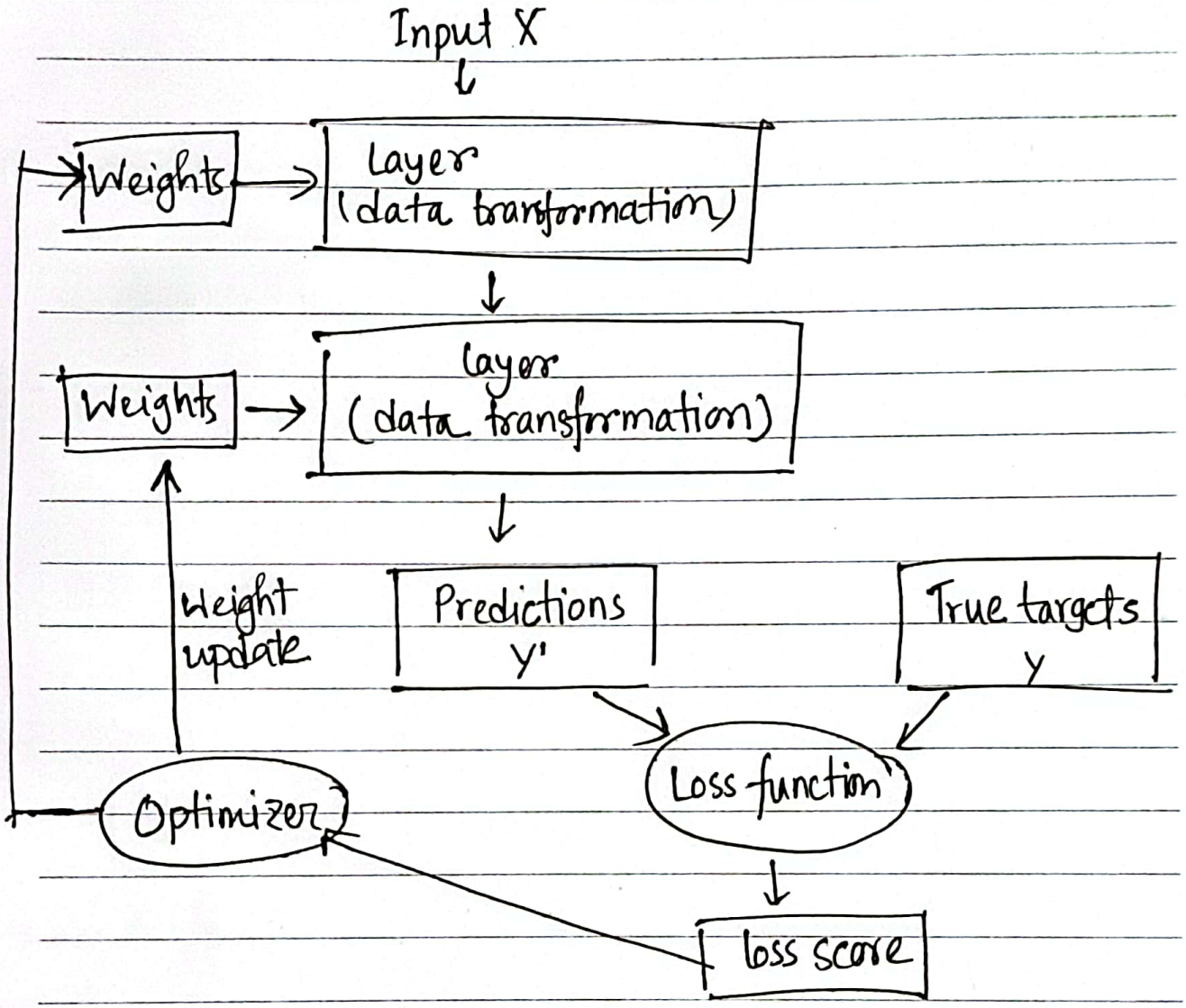
Forward pass

Backward pass

chain rule of differentiation

$$\frac{\delta Loss}{\delta w} = \frac{\delta Loss}{\delta x_2} \cdot \frac{\delta x_2}{\delta x_1} \cdot \frac{\delta x_1}{\delta w}$$

grad(loss_val, w) = grad(loss_val, x2) * grad(x2, x1) * grad(x1, w)



Convergence in neural network

↳ global minimum

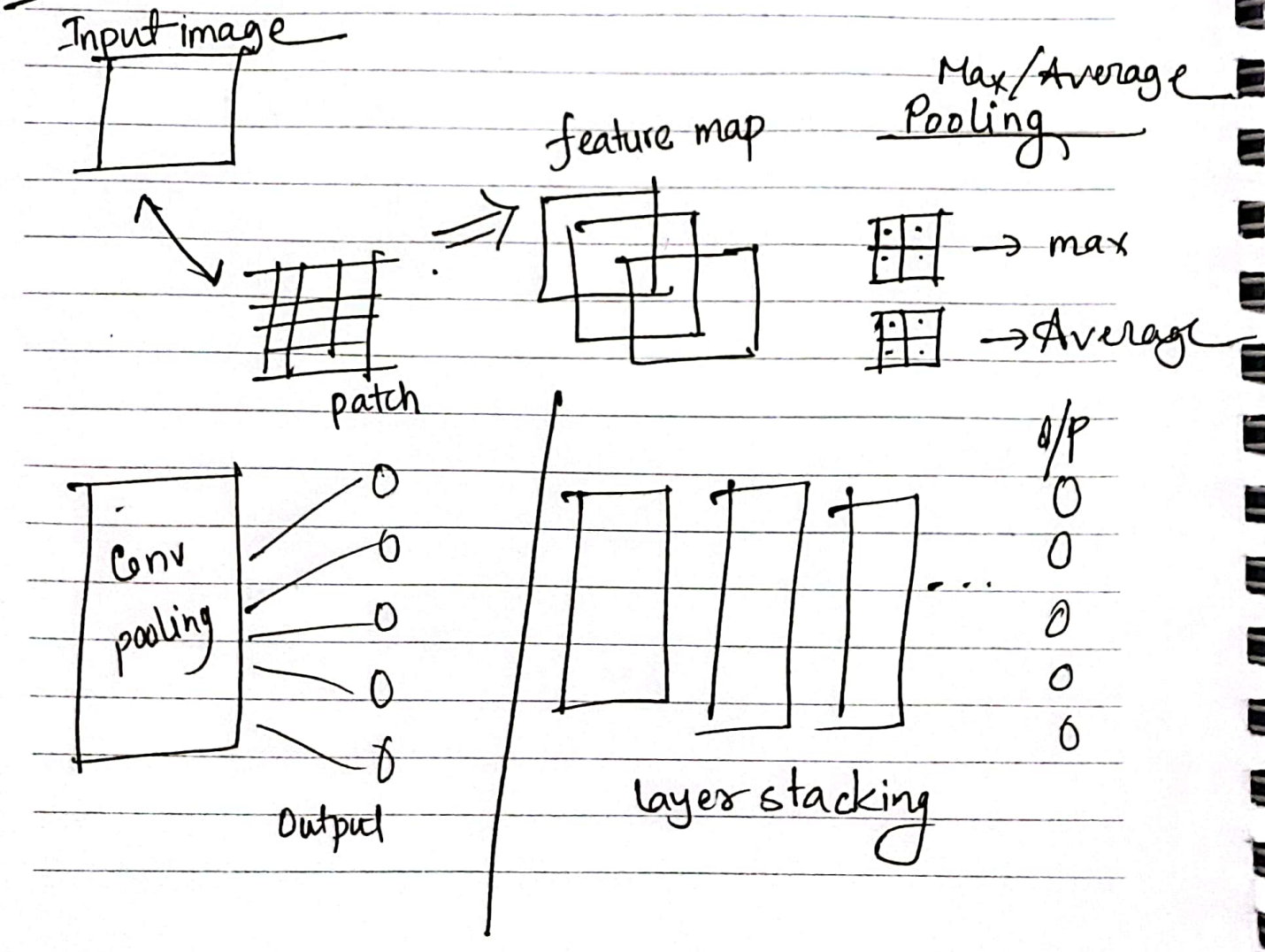
learning rate
can be ~~man~~ changed
if small → will stuck at local min
if large → overshoots.

Popular Neural Networks:

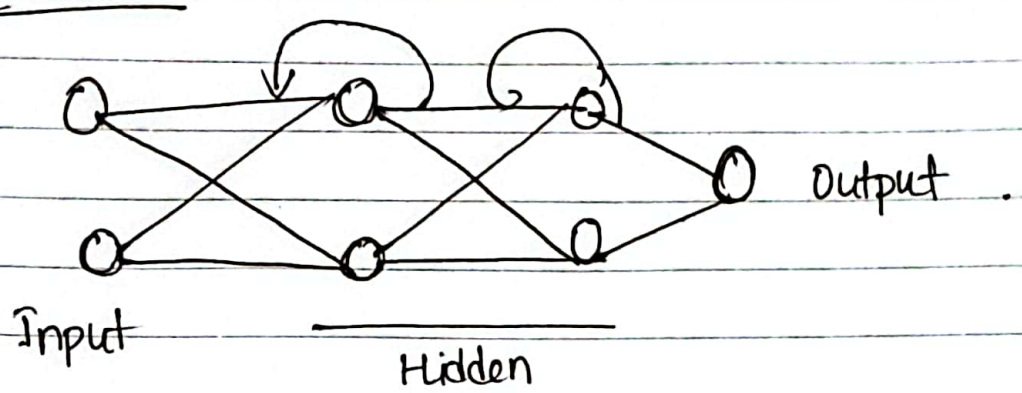
CNN → Convolutional NN → Image classification

RNN → Recurrent NN → Time series
Sequential data
Machine Translation
Video processing
Speech processing
Weather prediction

CNN



RNN

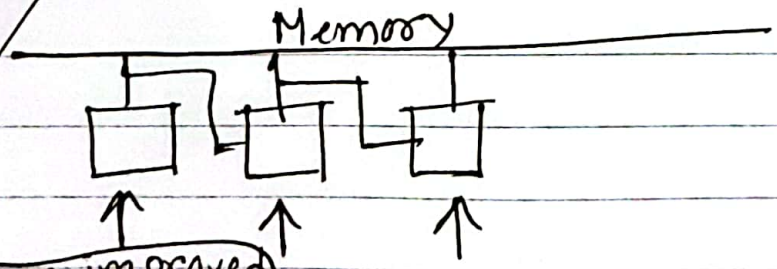


Disadvantages:

→ Vanishing gradient : value of gradient approaches zero. No update of w occurs. Dead network

Soln: LSTM to solve vanishing gradient

Long Short Term Memory



more improved

Transformer

self attention