Computational Intelligence & Machine Learning.

Basics :-

→ The ability of machine to categorize correctly new Examples that differ from those used for training is known as Generalization.

→ Machine learning is done in 2 phases

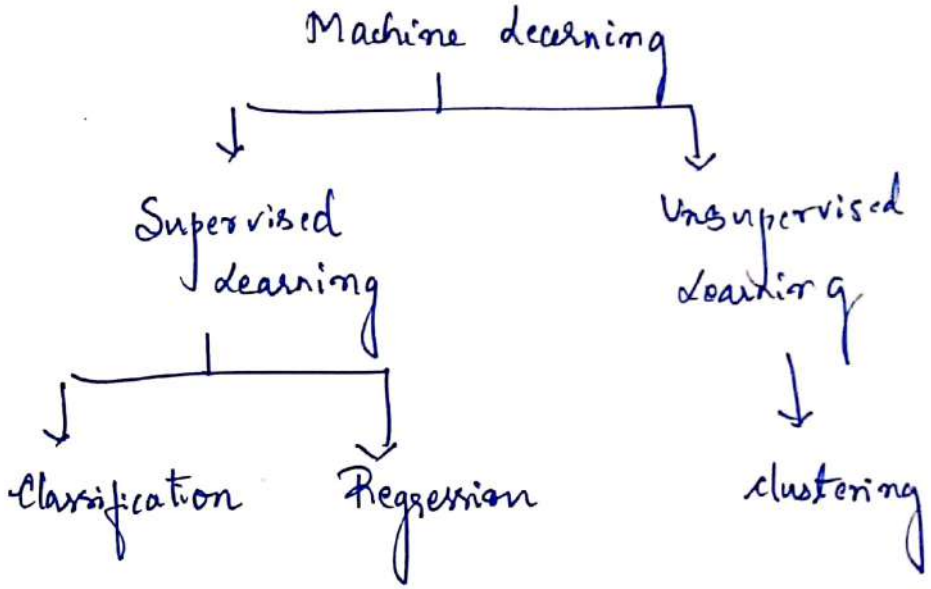
- Training phase
- Testing phase

* In training phase we train the machine with N variants of a random continuously varying data say 'X'

* In testing phase a new varient of the continously varying quantity x is given to test the prediction of the Machine learning Algorithm.

→ There can be 2 ways of machine learning

- Supervised
- un-Supervised.



Linear Basis function Models :- [Linear Regression]

→ Regression analysis is a form of predictive modeling technique which gives relationship between dependent (target) & Independent variables (predictor)

→ Linear Regression is a statistical Model which gives linear relationship between two or more variables.

Note :- Regression algorithms are used to predict the continuous values such as price, salary, age etc.

→ Simple Linear Model of regression is given as

$$y(x, w) = w_0 + w_1 x_1 + \ldots + w_D x_D$$

where $x = (x_1, \ldots, x_D)^T$

$x \rightarrow$ Independent variable

$y \rightarrow$ dependent variable.

→ To Extend the class of models to Include non-linearity of o/p variables Equation is modified as

$$y(x, w) = w_0 + \sum_{j=1}^{M-1} w_j \underbrace{\phi_j(x)}_{\text{Basis function}}$$

$\downarrow$ Total M parameters

* w_0 → Bias parameter

$$\cdot y(x, w) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x)$$

where $w = (w_0 \ldots, w_{M-1})^T$ & $\phi = (\phi_0, \ldots \phi_{M-1})^T$

Note :- Here original variables are vector x, features of x is denoted as $\{\phi_j(x)\}$

→ By using non-linear Basis fn we allow $y(x, w)$ to non-linear fn of o/p vector x.

→ Basis function → polynomial
→ Gaussian
→ Sigmoidal

* polynomial $\phi_j(x) = x^j$

* Gaussian $\phi_j(x) = \exp\left\{-\frac{(x-\mu)^2}{2s^2}\right\}$

* Sigmoidal $\phi_j(x) = \sigma\left(\frac{x - \mu}{s}\right)$

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

Note :- Other basis for is Fourier Basis, wavelets.
We can also assume $\phi(x)$ to simply equal to x.
$\phi(x) = x$.

Maximum likelihood & Least Squares :-

* Basics to be known
  - Polynomial curve fitting
  - Gaussian function
  - Bayes theorem.

→ Applications in which training data comprises Examples of input vectors along with their corresponding target vectors are known as Supervised learning problems. If the desired output consists of one or more continuous variables, then task is called regression.

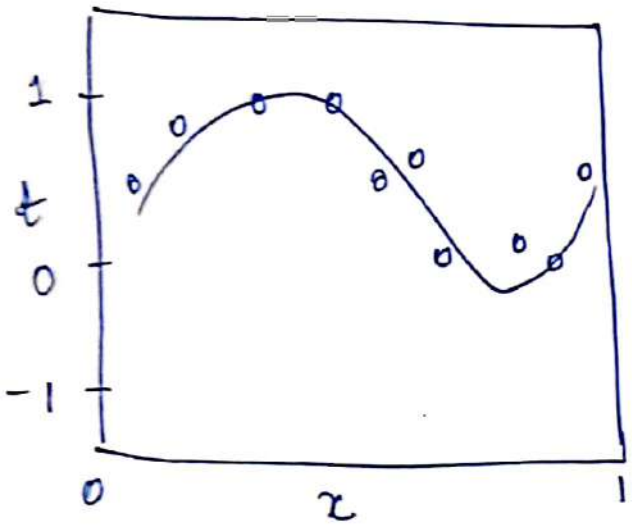
→ If training data consists of a set of input vector x without any corresponding target values. This is unsupervised learning where we need to discover groups of similar examples in the data. called as clustering.

→ The technique of reinforcement learning is concerned with the problem of suitable actions to taken in a given situation in order to maximize a reward. Here learning is on a trail & Error process.

Ex:- polynomial curve fitting

→ given a training set with N observations of x

$x = (x_1, \ldots x_N)^T$ with $t = (t_1, \ldots, t_N)^T$



* x is uniformly spaced in range [0, 1]

* t is calculated by taking an Example function $\sin 2\pi x$ with small level of random noise. ← o/p

→ Our goal is to Exploit this training set in order to make predictions of $\hat{t}$ of the target for some new value $\hat{x}$. of the Input variable. & discover the underlying function $\sin(2\pi x)$ through Machine learning Algorithm.

→ Polynomial function is given as

$$y(x, w) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M$$

$$= \sum_{j=0}^{M} w_j x^j$$

* M → order of polynomial
* polynomial co-efficients $w_0, \ldots, w_M$ are denoted as vector w.
* polynomial function $y(x, w)$ is a non-linear function of x, and linear function of w.

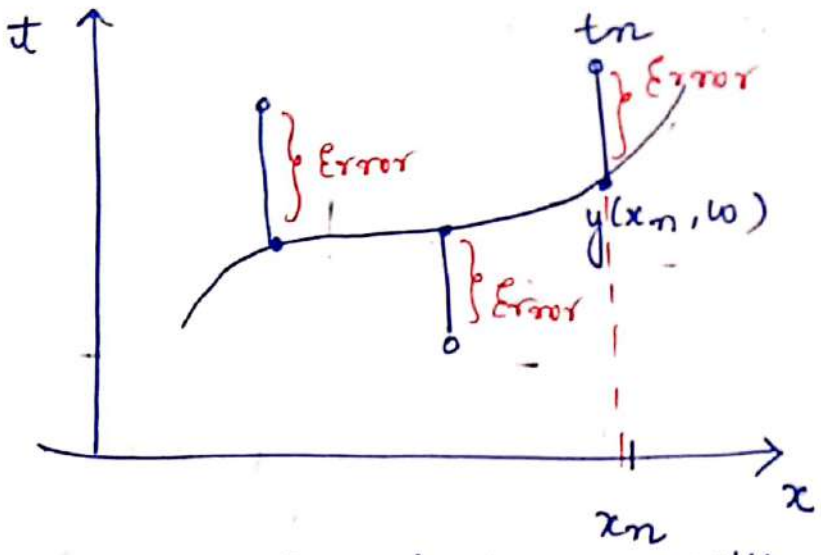
→ The values of co-efficients will be determined by fitting the polynomial to the training data

→ On fitting the polynomial curve to the training data we occur with Error or misfit ~~isx~~ between $y(x, w)$ for any given w & x.

→ To minimize errors widely used Error function for is Square of errors between $y(x_n, w)$ & data point x_n & corresponding target t_n.

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2$$

→ Fig below shows the geometrical interpretation of sum of square Errors.

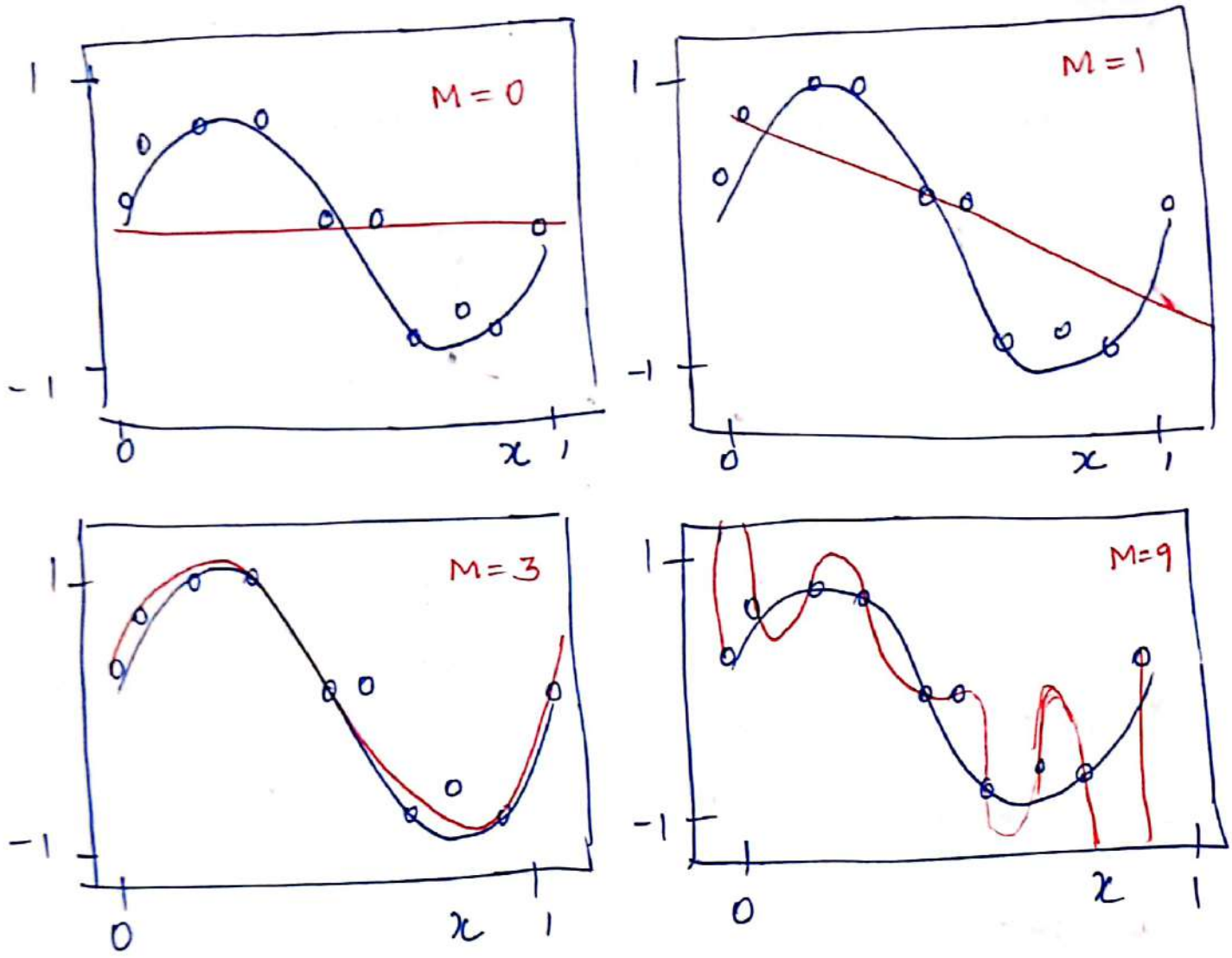


* Here we can solve the curve fitting problem by choosing the value of w $E(w)$ is as small as possible.

* Resulting polynomial with changed w is given as $y(x, w^*)$

→ Other than reducing $E(w)$, there remains the problem of choosing order of M of the polynomial.

→ .



→ we notice that with M = 0 & M = 1 polynomials we get poor fits of data [we need a sine ($\sin 2\pi x$) function fit]

Note :- Blue colour curve → Actual fit needed
Red colour curve → fit got with the order of polynomial.

→ with M = 3 we get Best fit to the function $\sin(2\pi x)$

→ with M = 9 we get Excellent fit to the training data.

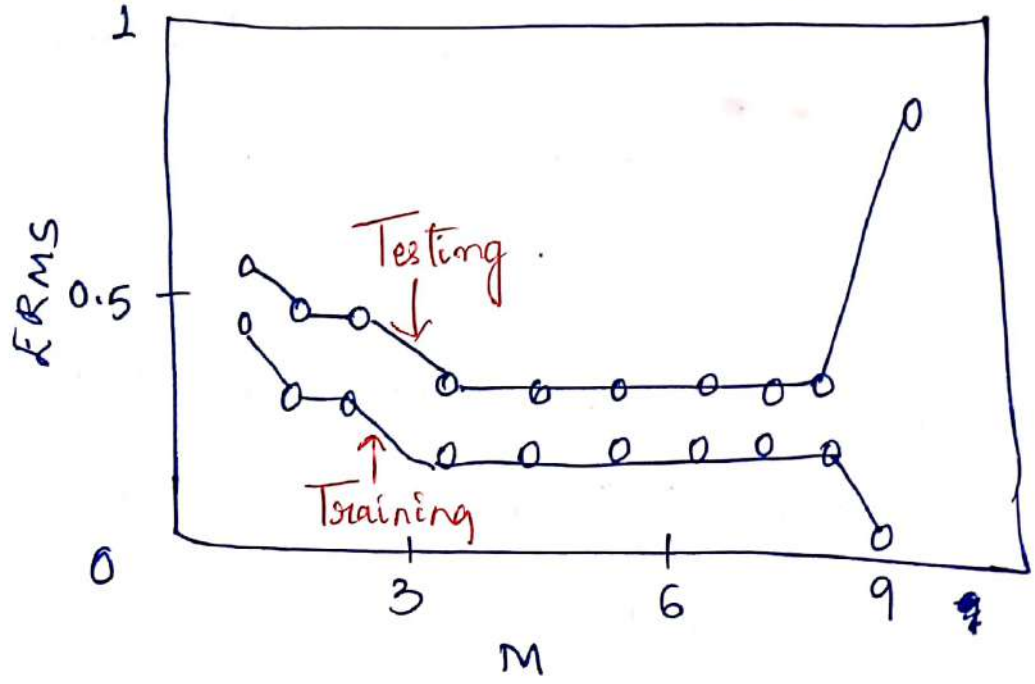
→ In fact polynomial passes Exactly through Each data point & $E(w^*) = 0$; But curve Oscillates Wildely & gives poor representation of the function $\sin(2\pi x)$. This is called over-fitting.

→ It is sometimes more convenient to use root-mean-square of Error function

$$E_{RMS} = \sqrt{2E(w^*)/N}$$

↓

different sizes of data sets.



* We note in above fig that for $3 \leq M \leq 8$ we get small values of E_{RMS} (Error).

* For $M = 9$, training set error goes to zero

but test-set error becomes very large, ∴ giving wild oscillations in $y(x, w^*)$ & the predictions.

→ The over-fitting problem becomes less severe as the size of data-set increases.

→ Over-fitting problem is a general property of maximum likelihood, By adapting Bayesian approch, overfitting problem can be avoided.

note :- In Bayesian Model the effectave number of parameters adapts automatically to the size of the data set.

→ One more approch to control the over-fitting phenomenon is regularization. Regularization involves adding a penalty term to the Error function.

$$\tilde{E}(w) = \frac{1}{2}\sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2}\|w\|^2$$

where $\|w\|^2 = w^T w = w_0^2 + w_1^2 + \dots + w_M^2$ & the co-efficient λ governs the relative importance of the regularization term.

Probability theory :-

→ Probability theory provides a consistent framework for the quantification & manipulation of uncertainity, When combined with decision theory. will allow us to make optimal predictions given all the Information available to us, even though that information may be incomplete or ambiguos.

Rules of Probability :-

Sum rule :- $P(x) = \sum_{y} P(x, y)$

Product rule :- $P(x, y) = P(y|x) P(x)$

note :- $P(x, y)$ → joint probability {Probability of x & y}

$P(y|x)$ → Probability of y given x.

$P(x)$ → Marginal probability / probability of x

→ From product rule, together with the symmetry property $P(x, y) = P(y, x)$

we obtain the following relationship between Conditional probabilities.

$$P(Y|X) = \frac{P(X|Y)\,P(Y)}{P(X)}$$

Baye's theorem

* Baye's theorem plays a central role in pattern recognition & machine learning.

* Denominator of Baye's theorem can be written as

$$P(X) = \sum_{Y} P(X, Y) \rightarrow \text{sum rule}$$

or

$$P(X) = \sum_{Y} P(X|Y)\,P(Y)$$

$$\left(\because P(X,Y) = P(Y|X)\,P(X)\right)$$

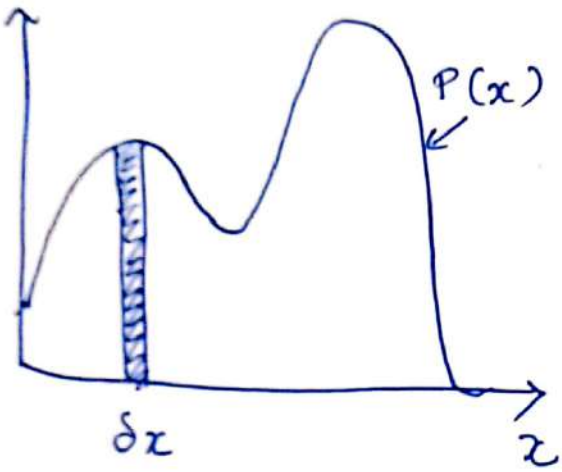
$$\therefore P(X) = \sum_{Y} P(X|Y)\,P(Y)$$

↓ normalization constant-

Probability densities :-

→ we wish to consider probabilities with continu-os variables rather than just discrete set of Events.

→ If we have to take probability of continuously varying real-valued variable x falling in the interval $(x, x+\delta x)$ is given as $P(x)\,\delta(x)$ for $\delta x \to 0$, then $P(x)$ is called Probability density function over x.



→ The probability that x will lie in an interval (a, b) is given by

$$P(x \in (a,b)) = \int_a^b P(x)\,dx$$

→ Probabilities are nonnegative

$$\therefore P(x) \geq 0$$

$$\int_{-\infty}^{\infty} P(x)\,dx = 1$$

→ $P(x)$ must satisfy these two conditions.

→ Most important operations involves probabilities is that of finding weighted averages of functions

→ The average value of some function $f(x)$ under a probability distribution $p(x)$ is called Expectation of $f(x)$. & is denoted by $\mathbb{E}[f]$.

Discrete Distribution

$$E[f] = \sum_x P(x) f(x)$$

Continuous variables

$$E[f] = \int P(x) f(x) \, dx$$

→ Applying / Re-writing concept of Baye's theorem with respect to linear Basis function model

likelihood

$$y(w|t) = \frac{P(t|w) \; P(w)}{P(t)}$$

↓ (posterior Probability)

Prior Probability Info.

* w → weights associated with input

t → target achieved when right weights are set

→ Baye's theorem is used to convert a prior probability into a posterior probability by incorporating the Evidence provided by the observed data.

$$\text{Posterior} \propto \text{likelihood} \times \text{prior}$$

note :- all of the quantities in Baye's probability theorem are viewed as function of w.

* The denominator is normalization Constant which

Ensures that posterior distribution on d·H·S is a valid probability density & Integrates to one.

→ In Baye's theorom, the likelihood function plays a central role.

→ A widely used Estimator of w is maximum likelihood in which w is set to a value that maximizes likelihood function ~~P(t|x,w)~~ $P(t|w)$.

The Gaussian Distribution :-

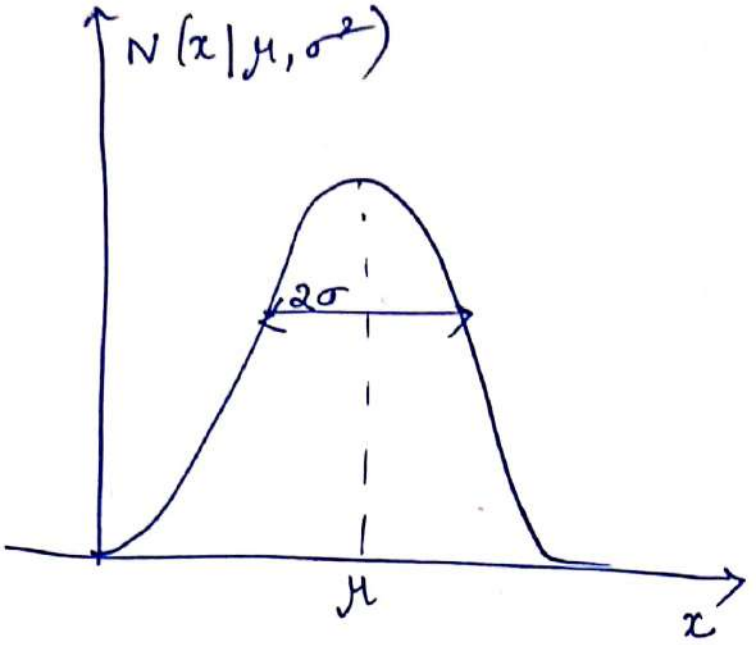
→ Let us assume that the random variable which is continuously varying & is used to train the Machine has a gaussian distribution Instead of polynomial distribution, discussed Earlier

→ For a real-valued variable x, Gaussian distribution is defined as.

$$N(x \mid \mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

↓

Gaussian is also called normal distribution.



$\mu \rightarrow$ mean

$\sigma^2 \rightarrow$ variance

$\sqrt{\sigma^2}$ / $\sqrt{\text{variance}} = \sigma =$ Std deviation

$\frac{1}{\text{Variance}} = \frac{1}{\sigma^2} = \beta$ [precision]

* Above Gaussian distribution satisfies the below Condition

$$N(x|\mu, \sigma^2) > 0$$

$$\int_{-\infty}^{\infty} N(x|\mu, \sigma^2)\, dx = 1$$

* Expectation of x under this gaussian distribution is given as

$$\mathbb{E}[x] = \int_{-\infty}^{\infty} N(x|\mu, \sigma^2)\, x\, dx = \mu$$

μ → average value of x

→ We can obtain or Maximize likelihood function by writing the probability of data set, given μ, σ^2

Gaussian distribution is given as

$N(x|\mu, \sigma^2)$ → taking probability of this distribution wrt μ, σ^2

wrt μ

$$P(x|\mu, \sigma^2) = \prod_{N=1}^{N} N(x_n|\mu, \sigma^2)$$

* It is more convenient to maximize log of likelihood function.

∴ Above Equation becomes

$$\ln p(x|\mu, \sigma^2) = -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (x_n - \mu)^2 - \frac{N}{2} \ln \sigma^2 - \frac{N}{2} \ln(2\pi)$$

∴ likelihood fn wrt μ is maximizing the above Expression. we get

$$\mu_{ML} = \frac{1}{N} \sum_{n=1}^{N} x_n$$

μ_{ML} → max likelihood

wrt β

$$p(t|x, w, \beta) = N(t|y(x, w), \beta^{-1})$$

$$p(t|x, w, \beta) = \prod_{n=1}^{N} N(t_n|y(x_n, w), \beta^{-1})$$

$$\beta = \frac{1}{\sigma^2} \quad \therefore \beta^{-1} = \sigma^2$$

111ly taking log for above Expression we get

$$\ln p(t|x, w, \beta) = -\frac{\beta}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2 + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi)$$

$$\frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{n=1}^{N} \{y(x_n, w_{ML}) - t_n\}^2$$

→ Now returning back to maximum likelihood & least square concept.

→ Let us consider target t with Gaussian noise model

$$\therefore t = y(x, w) + \epsilon \leftarrow \text{Gaussian noise}$$

$$\downarrow$$

$$y(x, w) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x)$$ [Linear Basis for model]

→ Taking probability of the above Expression with precision parameter from Gaussian distribution

$$p(t|x, w, \beta) = N(t|y(x, w), \beta^{-1})$$

Normal distribution · Inverse variance

β ↓ precision

$\left[\frac{1}{\sigma^2} = \beta \Big/ \frac{1}{\beta} = \sigma^2, \quad \beta^{-1} = \sigma^2\right]$ ↓ Variance

* Here Expectation of new x to a target t is given as

$$\mathbb{E}[t|x] = \int t \cdot p(t|x)\, dt = y(x, w)$$

→ Now consider a data set of inputs

$X = \{x_1, \ldots, x_N\}$ with $t_1 \ldots t_N$

→ We can obtain the Expression for the likelihood function as a function of Adjustable parameter w, β

$$\therefore p(t|x, w, \beta) = \prod_{n=1}^{N} N(t_n | w^T \phi(x_n), \beta^{-1})$$

$$y(x, w) = w^T \phi(x_n)$$

→ To maximize this likelihood function

$$p(w|t) = \frac{p(t|w) \cdot p(w)}{p(t)}$$

(likelihood fn) → Bayes theorem

& keeping notation of LHS dropping x we get :

$$p(t|w, \beta) = \prod_{n=1}^{N} N(t_n | w^T \phi(x_n), \beta^{-1})$$

→ Taking log on both sides to maximize the above function $p(t|w, \beta)$ [refer gaussian distribution]

we get

$$\ln p(t|w, \beta) = \sum_{n=1}^{N} \ln N\{t_n | w^T \phi(x_n), \beta^{-1}\}$$

$$= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(w) \quad \rightarrow (1)$$

↓ likelihood ↓ Error to likelihood

where where $E_D \rightarrow$ Sum of Square Error. for n

$$E_D(w) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}^2$$

→ We can maximize the likelihood function under a conditional Gaussian noise distribution by minimizing a sum-of squares Error given by $E_D(w)$.

→ ~~Maxmize~~ Minimize by taking gradient of $E_D(w)$

$$\nabla \ln p(t|w,\beta) = \beta \frac{d}{dw}\left[\frac{1}{2}\sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}^2\right]$$

(const) (const)

* [Simply differentiate by w]

βE_D.

$$0 = \frac{d}{dw}\left[\frac{1}{2}\sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}^2\right]$$

Assume this x [const's $= 0$]

note :- above Expression is w.r.t to

$$\frac{d}{dx}x^2 = 2x\frac{d}{dx}.x$$

$$\therefore \quad 0 = \frac{d}{d}\sum_{n=1}^{N}\frac{1}{2}\cdot 2(t_n - w^T\phi(x_n)).\frac{d}{dw}[t_n - w^T\phi(x_n)]$$

$$\therefore 0 = \sum_{n=1}^{N} \cancel{\alpha} \cdot \frac{1}{\cancel{\alpha}} \left(t_n - w^T \phi(x_n) \cdot \left[0 - \phi^T(x_n)\right]\right)$$

$$0 = \sum_{n=1}^{N} \left(t_n - w^T \phi(x_n) \cdot \phi^T(x_n)\right)$$

$$0 = \sum_{n=1}^{N} t_n \phi(x_n)^T - \sum_{n=1}^{N} w^T \phi(x_n) \cdot \phi^T(x_n)$$

* For making the Equation in generalized form we will write the above Equation as

$$\rightarrow \quad 0 = t \cdot \phi(x_n)^T - w^T \phi(x_n) \phi(x_n)^T$$

$$\rightarrow \quad t \cdot \phi(x_n)^T = w^T \phi(x_n) \phi(x_n)^T$$

Or

$$\rightarrow \quad w \phi(x_n)^T \phi(x_n) = t \cdot \phi(x_n)^T$$

$$\rightarrow \quad \therefore w \phi(x_n)^T \phi(x_n) \cdot \left[\phi^T(x_n) \cdot \phi(x_n)\right]^{-1} = \left[\phi^T(x_n) \cdot \phi(x_n)\right]^{-1} \cdot t \cdot \phi(x_n)^T$$

$$\left[\because A \cdot A^{-1} = 1 \;/\; \phi^T(x) \cdot \phi(x)^{-1} = 1\right]$$

$$\therefore w \underbrace{\phi(x_n)^T \phi(x_n) \cdot \left[\phi^T(x_n) \cdot \phi(x_n)\right]^{-1}}_{w_{ML}} = \left[\phi^T(x_n) \cdot \phi(x_n)\right]^{-1} \cdot t \cdot \phi(x_n)^T$$

$$\boxed{\therefore \; w_{ML} = \left[\phi^T \phi\right]^{-1} \cdot \phi^T \cdot t}$$

→ The quantity $w_{ML} = (\phi^T \phi)^{-1} \phi^T t$

↓

normal Equations for least Squares problem

$\phi \rightarrow N \times M$ matrix called Design matrix

$$\phi = \begin{bmatrix} \phi_0(x_1) & \phi(x_1) & \cdots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_{M-1}(x_2) \\ \vdots & & & \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_{M-1}(x_N) \end{bmatrix}$$

The quantity $\phi = (\phi^T \cdot \phi)^{-1} \phi^T$ is known as Moore – Penrose pseudo Inverse of matrix ϕ. [ϕ is a $N \times M$ matrix]

note :- $w_{ML} = [\phi^T \phi]^{-1} \cdot \phi^T \cdot t$

$(M \times N)(N \times M)$ $\quad$ $N \times M$ $\quad$ $N \times 1$

$M \times M$

$w_{ML} = (M \times 1)$ → Row vector as a

→ Now we can maximize the log likelihood function

$$\ln p(t|w,\beta) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(w)$$

wrt to precision parameter β.

$$\frac{d}{d\beta}\left[\ln p(t|w,\beta)\right] = \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(w)$$

$$= \frac{N}{2}\cdot\frac{1}{\beta} - 0 - E_D(w)$$

Equating above Equation to zero in order to minimize it

$$\frac{N}{2\beta} - E_D(w) = 0$$

$$\frac{N}{2\beta} = E_D(w)$$

$$\frac{1}{\beta} = \frac{2}{N}\cdot E_D(w)$$

Subst $E_D(w)$

$$\therefore \frac{1}{\beta_{ML}} = \frac{2}{N}\cdot\frac{1}{2}\sum_{n=1}^{N}\left(t_n - w^T\phi(x)\right)^2$$

$$\therefore \frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\left(t_n - w^T\phi(x)\right)^2$$

∴ we can conclude that to maximize the likelihood function by 2 parameters w, β.

wrt w :-

$$w_{ML} = (\Phi^T \Phi)^{-1} \Phi^T t$$

wrt β :-

$$\frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{n=1}^{N} \{t_n - w_{ML}^T \phi(x_n)\}^2$$

∴ We can conclude that to maximize the likelihood function by 2 parameters w, β.

wrt w :-

$$w_{ML} = (\phi^T \phi)^{-1} \phi^T t$$

wrt β :-

$$\frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{n=1}^{N} \{t_n - w_{ML}^T \phi(x_n)\}^2$$

Sequential learning :-

→ When we have Input data set sufficiently Large & we use the Complete set or Entire Batch for training the machine [Batch processing] the computationally costly

→ ∴ we can use sequential algorithms as an alternative. which is also known as on-line algorithms.

→ Here data sets of the Input are considered one at a time & parameters are updated Each time.

→ We can obtain a Sequential learning algorithm by applying the technique of Stochastic gradient descent / Sequential gradient descent.

→ Error function is given for all data points as Summation function.

$$E = \sum_n E_n$$

→ The stochastic gradient descent algorithm updates the parameter w using

$$w^{(\tau+1)} = w^{\tau} - \eta \nabla E_n$$

τ → Iteration number

η → Learning rate

→ we Initialize value of 'w' to some starting vector $w^{(0)}$

→ For the case of Sum of square of Error function $\left[E(w) = \frac{1}{2} \sum_{n=1}^{N} \{t_n - w^T \phi(x_n)\}^2 \right.$

$$w^{(\tau+1)} = w^{\tau} + \eta (t_n - w^{(\tau)T} \phi_n) \phi_n$$

where $\phi_n = \phi(x_n)$. This is known as least-mean-squares or LMS algorithm.

Regularized least Squares :-

→ earlier concept of adding a regularization term to an Error function in order to control over-fitting to control total Error function & minimize it.

$$E_D(w) + \lambda E_W(w)$$

↓ regularization term

$E_D(w)$ → data dependent Error

$E_W(w)$ → regularization term

→ simple regularizer is given by sum-of-squares of weight vector Elements.

$$E_W(w) = \frac{1}{2} w^T w$$

also $$E(w) = \frac{1}{2} \sum_{n=1}^{N} \{t_n - w^T \phi(x_n)\}^2$$

[Subst done for $E_D(w) + \lambda E_W(w)$]

∴ Total Error fn is

$$\frac{1}{2} \sum_{n=1}^{N} \{t_n - w^T \phi(x_n)\}^2 + \frac{\lambda}{2} w^T w$$

$$E(w) + \lambda E_W(w) \rightarrow (1)$$

→ This choice of regularizer is known in machine learning as weight decay as in Sequential learning algorithms, it Encourages weight-vectors to decay towards zero, unless supported by the data.

→ Setting gradient of Eqn (1) wrt to 'w' to zero & solving for 'w' we get

Eqn (1) is

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}^2 + \frac{\lambda}{2}w^T.w$$

$$\therefore \frac{1}{2}\sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}^2 + \frac{\lambda}{2}w^2$$

differentiating above Expression wrt 'w'

$$= \frac{1}{2}\cdot 2\cdot\{t_n - w^T\phi\}\phi^T + \frac{\lambda}{2}\cdot 2w$$

$$= t\phi^T - w^T\phi\phi^T + \lambda.w$$

$$= t\phi^T - w^T(\phi\phi^T - (-)\lambda I)$$

$$0 = t\phi^T - w^T(\phi\phi^T + \lambda I)$$

$$\therefore w(\phi^T\phi + \lambda I) = t\phi^T$$

or $$\boxed{w = (\lambda I + \phi^T \phi)^{-1} \phi^T . t .}$$

↑ regularization factor

Simple Extension of least square with regularization factor.

$$w_{ML} = (\phi^T \phi)^{-1} \phi^T . t .$$

Multiple outputs :-

→ If we wish to predict more target variables 't' denoted collectively by target vector 't', then

$$y(x, w) = w^T \phi(x)$$

↓

K → dimensional column vector

$w \rightarrow M \times K$ matrix

$\phi(x) \rightarrow M$ dimensional column vector

* Gaussian distribution

$$p(t|x, w, \beta) = N(t | w^T \phi(x), \beta^{-1} I)$$

$$\therefore \ln p(T|X, w, \beta) = \sum_{n=1}^{N} \ln N(t_n | w^T \phi(x_n), \beta^{-1} I)$$

$\{t_1 \cdots t_n\}$ ↑ , $\{x_1 \cdots x_n\}$ ↓

$$= \frac{NK}{2} \ln\left(\frac{\beta}{2\pi}\right) - \frac{\beta}{2} \sum_{n=1}^{N} \| t_n - w^T \phi(x_n) \|^2$$

↑ III term

[III° to $\frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(w)$ derived Earlier, above Expression can also be written as $\frac{N}{2} \ln\left[\frac{\beta}{2\pi}\right] - \beta E_D(w)$ $\left[\log A - \log B = \log A/B\right]$]

∴ maximizing above fn wrt w

$$\boxed{w_{ML} = (\phi^T \phi)^{-1} \phi^T \cdot T}$$

→ Target vector

↑ III° to $w_{ML} = (\phi^T \phi)^{-1} \phi^T \cdot t$

↑ Single target

→ If we Examine result for target variable t_K

$$w_K = (\phi^T \phi)^{-1} \phi^T t_K = \phi\, t_K.$$

The Bias-variance Decomposition :-

↔ Use of maximum likelihood or least Squares can lead to Severe over-fitting if complex data sets are taken.

→ We can limit the no of basis function in order to avoid over-fitting but this will limit the flexibility of the model to capture important trends in data.

→ Here adding a regularization term (λ) can control over-fitting for models with many parameters.

→ Bayesian treatment of linear regression will avoid the over-fitting problem of maximum likelihood.

→ Note :-> Bias is used to simplify the assumptions made by the model to make the target function easier to approximate.

2) Variance is the amount that the Estimate of the target will change given different training data.

3) Bias - variance tradeoff is the property of a model that the variance of the parameter Estimated across samples by increasing the bias in the Estimated parameters.

⇒ Bias - variance decomposition or tradeoff is the conflict in trying to simultaneously minimize the two sources of Error that prevent supervised learning algorithms from generalizing beyond the training set.

Bias Error - erroneous assumptions in the learning algorithm.

Variance - Error from sensitivity to small fluctuations in the training set.

** * Bias - Variance decomposition is a way of analyzing a learning algorithm's Expected generalization Error with respect to three terms bias, variance, irreducible Error. resulting from noise itself.

→ Assuming $g(x)$ as generalization Error which is taken for predicting a new target t for a new input x

$$E\left[(t - g(x)\right]^2$$

→ Error in Estimation of t can be written as

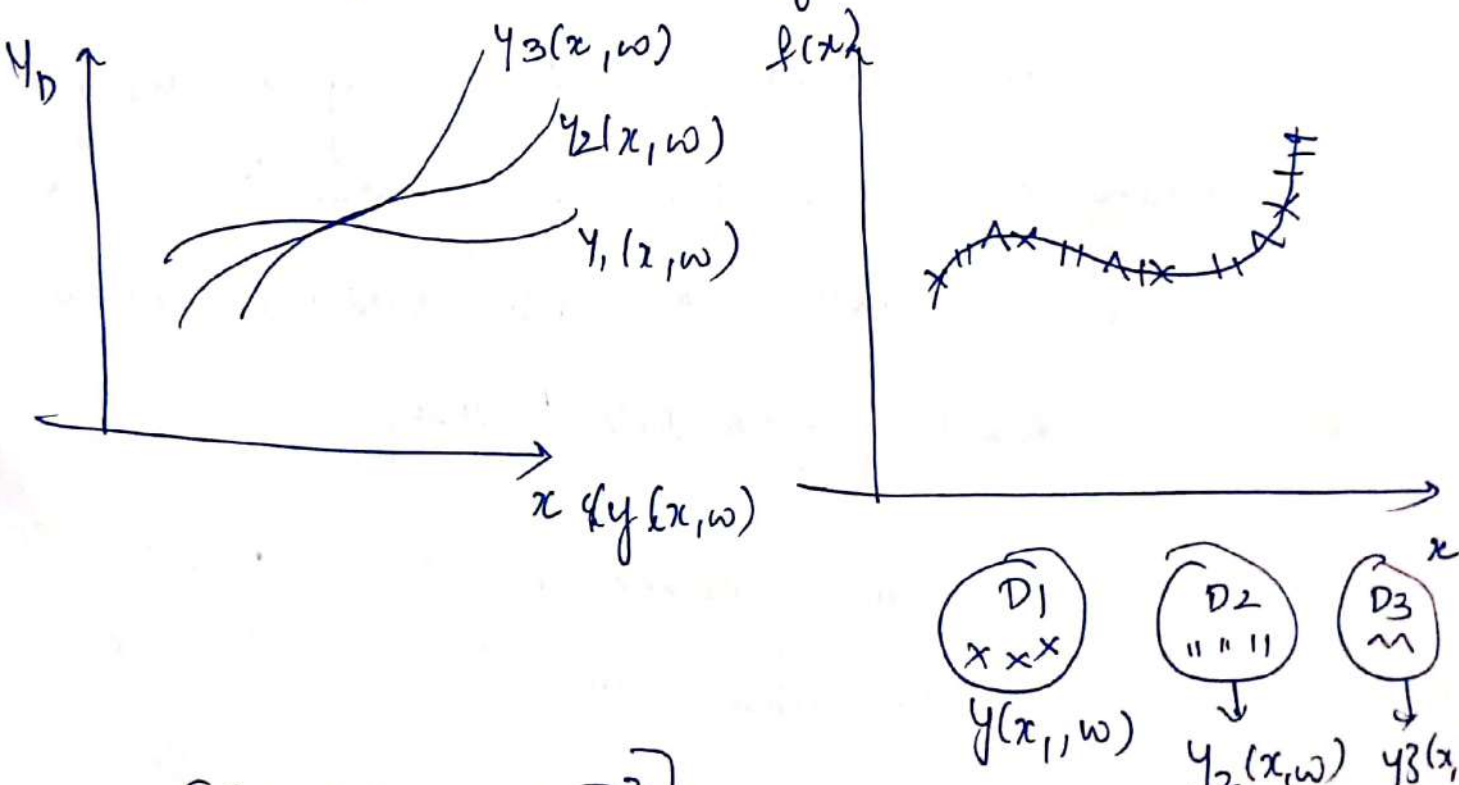
$$E\left[(t - y(x)\right]^2 = E\left[(t - g(x)\right]^2$$

$$= E\left[(t - g(x) + g(x) - y(x)\right]^2$$

$$= E\left[(t - g(x))^2\right] + E\left[(g(x) - y(x))^2\right]$$

$$+ 2 \cdot E\left[(t - g(x))\, g(x) - y(x)\right]$$

→ Here $y(x)$ is obtained from data set-



→ $E\left[\left[(y(x) - g(x)\right]^2\right]$ → (1)

we can write the above eqn (1) as

→ $E_D\left[(g(x) - y(x,D))^2\right]$.

$$= E_D\left[\left(\underbrace{y(x,D) - E_D[y(x,D)]}_{p} + \underbrace{E_D[y(x,D)] - g(x)}_{q}\right)^2\right]$$

$$= E_D\left[p^2 + q^2 + 2pq\right]$$

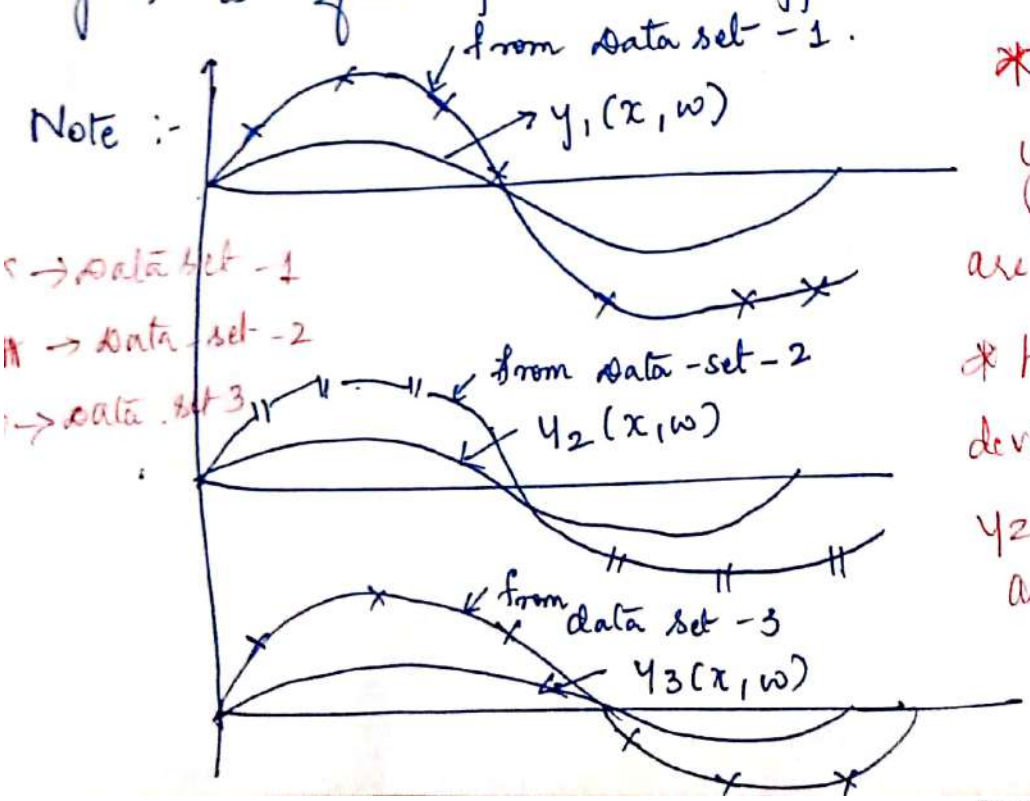
$$= \underbrace{E_D\left[\left(y(x,D) - E_D[y(x,D)]\right)^2\right]}_{\text{Variance}} + \underbrace{E_D\left[E_D[y(x,D) - g(x)]^2\right]}_{\text{Bias}^2}$$

$$+ 2\,\underbrace{E_D\left[\left(y(x,D) - E_D[y(x,D)]\right)\left[E_D(y(x,D) - g(x))\right]\right]}_{\text{noise}}$$

$$E^{(\text{loss})} = \text{Variance} + \text{Bias}^2 + \text{~~loss~~ Noise}$$

* $g(x)$ is from Expectation

$y(x)$ is from parametric approch.



Note :-

→ data set - 1

→ data set - 2

→ data set 3

* $g(x)$ is reference

$y_1(x,w)$, $y_2(x,w)$ are different solution

* how far $g(x)$ is deviated from $y_1(x,w)$

$y_2(x,w)$ is known as Bias.

* Estimation within $y_1(x,w)$, $y_2(x,w)$, $y_3(x,w)$ or deviation within $y_1(x,w)$, $y_2(x,w)$, $y_3(x,w)$ is called as Variance.

-: Module 2 :-

Introduction :-

→ The goal in classification is to take an input vector x & to assign it to one of K discrete classes C_k.

→ The input space is divided into decision regions whose boundaries are called as decision boundaries.

→ As we are discussing linear classification, it means that the decision surfaces are linear functions of input vector x & are defined by (D-1) dimensional hyperplanes within the D-dimensional input space.

→ There are 3 distinct approches to the classification problems & the simplest involves constructing a discriminant function which directly assigns each vector 'x' to a specific class.

Discriminant functions :-

→ A discriminant function takes an i/p vector 'x' and assigns it to one of K classes denoted as C_k.

→ Here we anume dinear discriminations in which decision surfaces are hyperplanes.

Two classes :-

Simplest Heprerentation of a dinear discriminant function is

$$y(x) = w^T x + w_0$$

$w \rightarrow$ weight vector

$w_0 \rightarrow$ Bias

→ I/p vector x is assigned to class C_1 if $y(x) \geq 0$ & C_2 if otherwise.

→ The decision boundary is corresponding to a $(D-1)$ dimensional hyperplane within the D-dimensional Onput space.

→ Consider two points x_A & x_B both lie on the decision surfaces. Because $y(x_A) = y(x_B) = 0$ we have $w^T(x_A - x_B) = 0$ & hence 'w' vector is orthogonal to every vector lying on the decision surface, ∴ 'w' determines the Orienta-tion of the decision surface.

→ The normal distance from the Origin to the decision surface is given by

$$\frac{w^T x}{\|w\|} = -\frac{w_0}{\|w\|}$$

→ Consider the diagram shown below.

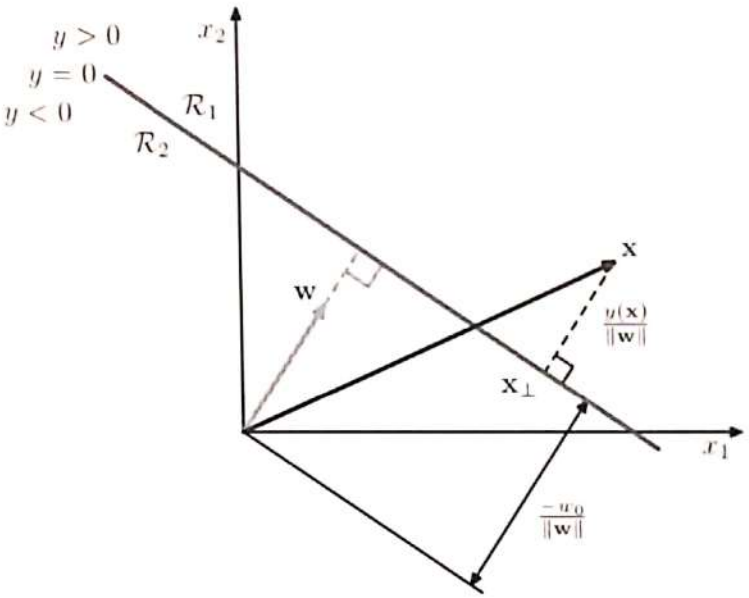


Figure 4.1 Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to $\mathbf{w}$, and its displacement from the origin is controlled by the bias parameter w_0. Also, the signed orthogonal distance of a general point $\mathbf{x}$ from the decision surface is given by $y(\mathbf{x})/\|\mathbf{w}\|$.

→ Here $y(x)$ gives a signed measure of the perpendicular distance r of the point x from the decision surface.

→ For Example let x be the arbitary point & $x_\perp$ be its orthogonal projection onto the decision surface.

$$x = x_\perp + r\frac{w}{\|w\|}$$

→ multiplying both sides with w^T & adding w_0 we get-

$$x \cdot w^T + w_0 = \left[x_\perp + r \frac{w}{\|w\|} \right] w^T + w_0$$

* using $y(x) = w^T x + w_0$

$$y(x_\perp) = w^T x_\perp + w_0 = 0$$

we get

$$r = \frac{y(x)}{\|w\|}$$

$$y(x) = \tilde{w}^T \tilde{x}$$

ii) Multiple classes :-

→ Consider Extension of linear discriminations to $K > 2$ classes.

→ Consider the use of $K-1$ classifiers, each of which solves a two class problem of seperating points in a particular class C_k from points not in that class.

→ This is known as a one-versus-the-rest classifier.

→ Fig below shows an example of 3 classes with ambiguous classification.

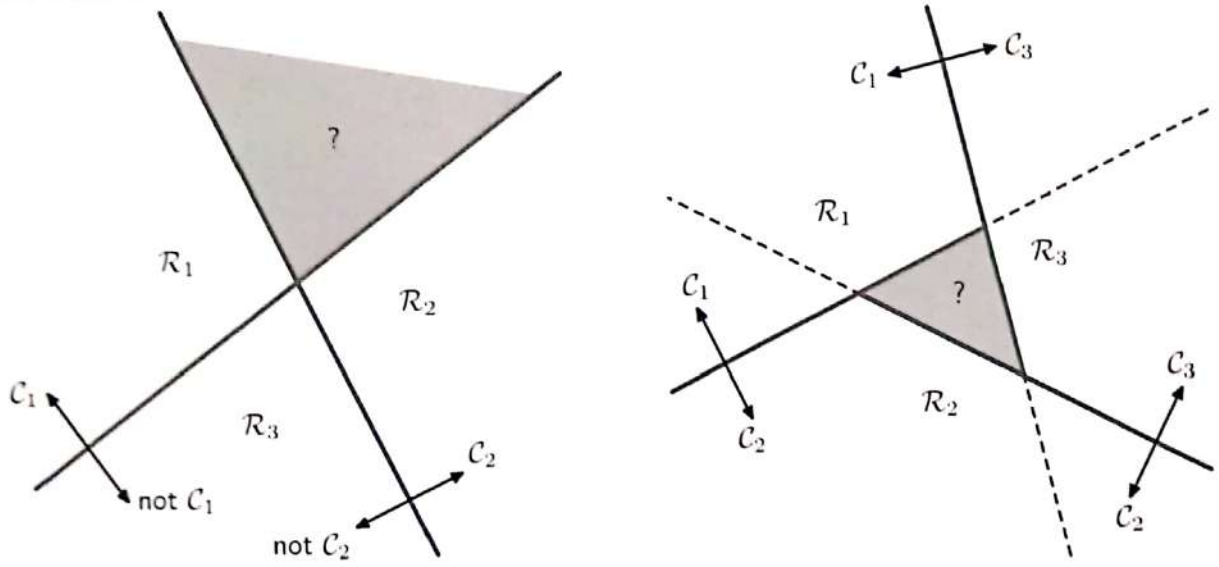


Figure 4.2 Attempting to construct a K class discriminant from a set of two class discriminants leads to ambiguous regions, shown in green. On the left is an example involving the use of two discriminants designed to distinguish points in class $\mathcal{C}_k$ from points not in class $\mathcal{C}_k$. On the right is an example involving three discriminant functions each of which is used to separate a pair of classes $\mathcal{C}_k$ and $\mathcal{C}_j$.

→ An alternate is to introduce $\frac{K(K-1)}{2}$ Binary discriminant functions, one for every possible pair of classes. This is known as a one-versus-one classifies.

→ Further Illustrating in Equations.

$$y_K(x) = w_K^T x + w_K^0$$

$$\& \ (w_K - w_j)^T x + (w_{K0} - w_{j0}) = 0$$

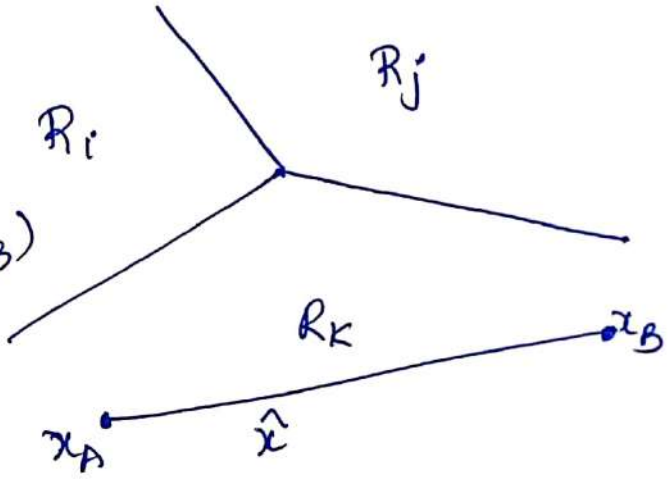
were K & j are two classes C_K & C_j.

→ Consider two points x_A & x_B both of which lie inside decision region R_K. Any point $\hat{x}$ that lies on the line connecting x_A & x_B can be expressed in the form.

$$\hat{x} = \lambda x_A + (1-\lambda) x_B.$$

where $0 \leq \lambda \leq 1$

$$y_K(\hat{x}) = \lambda y_K(x_A) + (1-\lambda) y_K(x_B)$$



note :- Both x_A & x_B lie inside R_K. R_K is singly connected & convex.

Least Squares for classification :-

→ We know that minimization of sum-of-squares error function led to a simple closed-form solution for the parameter values. in regression, so that can be applied for classification.

→ Each class C_K described by its linear model

$$y_K(x) = w_K^T x + w_{K0}$$

where $K = 1, \cdots, K$, grouping these together using vector notation.

$$y(x) = \tilde{w}^T \tilde{x}$$

→ we now determine parameter matrix $\tilde{w}$ by minimizing sum-of-squares error function.

Error function is defined as

$$E_D(\tilde{w}) = \frac{1}{2} Tr\{(\tilde{x}\tilde{w} - T)^T (\tilde{x}\tilde{w} - T)\}$$

* Setting the derivative with respect to $\tilde{w}$ to zero & rearranging we obtain

$$\tilde{w} = (\tilde{x}^T \tilde{x})^{-1} \tilde{x}^T T = \tilde{x}^+ T$$

↓

Pseudo-inverse of the matrix $\tilde{x}$.

note :- Sum-of-squares error function penalizes predictions that are 'too correct' & which lie along the decision boundary.

Principle Component Analysis :-

→ It is a way of identifying pattern in data and Expressing the data in such a way to highlight their similarities & differences

→ It is also way to reduce the dimension [Dimensionality Reduction] to reduce the complexity of the Analysis.

Example problem :-

1. Given the following data use PCA to reduce the dimension from 2 to 1.

Feature	1	2	3	4
x	4	8	13	7
y	11	4	5	14

Step 1 :- from the data set find n & N

n = no. of features = 2

N = no. of samples = 4

Step 2 :- Computation of mean of variables

$$\bar{x} = \frac{4+8+13+7}{4} = 8$$

$$\bar{y} = \frac{11+4+5+14}{4} = 8.5$$

Step 3 :- Computation of covariance matrix

i) find ordered pairs → x, y

(x, x) (x, y) (y, x) (y, y)

ii) find covariance of all ordered pairs

$$Cov(x,x) = \frac{1}{N-1} \sum_{K=1}^{N} (x_{iK} - \bar{x}_i)(x_{jK} - \bar{x}_j)$$

or

a) $$Cov(x,x) = \frac{1}{N-1} \sum_{K=1}^{N} (x_i - \bar{x})^2$$

$$= \frac{1}{4-1} \left[(4-8)^2 + (8-8)^2 + (13-8)^2 + (7-8)^2 \right]$$

$$= 14$$

b) $$Cov(x,y) = \frac{1}{4-1} \left[(4-8)(11-8.5) + (8-8)(4-8.5) + (18-8)(5-8.5) + (7-8)(14-8.5) \right]$$

$$= -11$$

$$cov(y,x) = cov(x,y) = -11$$

$$Cov(y,y) = \frac{1}{4-1}\left[(11-8.5)^2+(4-8.5)^2+(5-8.5)^2 + (14-8.5)^2\right]$$

$$= 23$$

Covariance matrix :-

$$S = \begin{bmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{bmatrix}$$

$$= \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

Step 4 :- Eigen value, Eigen vector, normalized eigen vector.

i) Eigen value :-

$$\det(S - \lambda I) = 0$$

$$\text{Identity matrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \times \lambda = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$

$$S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

$$\therefore \det(S - \lambda I) = 0$$

$$\det\left[\begin{pmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{pmatrix}\right] = 0$$

$$(14-\lambda)(23-\lambda) - (-11 \times -11) = 0$$

$$\lambda^2 - 37\lambda + 201 = 0$$

$$\therefore \lambda_1 = 30.3849$$

$$\lambda_2 = 6.6151 \qquad [\lambda_1 > \lambda_2]$$

* pick largest value which becomes first principal component.

ii) Eigen vector of λ_1

$$(S - \lambda_1 I)\, U_1 = 0$$

$(S - \lambda_1 I)$ → Eigen vector of λ_1

$$\begin{bmatrix} 14-\lambda_1 & -11 \\ -11 & 23\lambda_1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = 0$$

$$\begin{bmatrix} (14-\lambda_1)u_1 - 11u_2 \\ -11u_1 + (23-\lambda_1)u_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$(14-\lambda_1)u_1 - 11u_2 = 0 \longrightarrow (1)$$

$$-11u_1 + (23-\lambda_1)u_2 = 0 \longrightarrow (2)$$

* use any one linear Equation to find values of eigen vector

$\therefore$ Using Eqn (1)

$$(14-\lambda_1)u_1 - 11u_2 = 0$$

$$\frac{u_1}{11} = \frac{u_2}{14-\lambda_1} = t$$

when $t = 1$

$$u_1 = 11, \quad u_2 = 14-\lambda_1$$

Eigen vector u_1 of $\lambda_1 = \begin{bmatrix} 11 \\ 14-\lambda_1 \end{bmatrix}$

$$= \begin{bmatrix} 11 \\ 14-30.3849 \end{bmatrix} = \begin{bmatrix} 11 \\ -16.3849 \end{bmatrix}$$

* Normalizing Eigen vector u_1

$$e_1 = \begin{bmatrix} \frac{11}{\sqrt{11^2+(-16.838)^2}} \\ \frac{-16.3849}{\sqrt{11^2+(-16.38)^2}} \end{bmatrix} = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$$

III^{rly} $e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$

Step 5 :- Derive new dataset-

	1	2	3	4
first princi component	P_{11}	P_{12}	P_{13}	P_{14}

$$P_{11} = e_1^T \begin{bmatrix} x_i - \bar{x} \\ y_i - \bar{y} \end{bmatrix} = e_1^T \begin{bmatrix} 4 - 8 \\ 11 - 8.5 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} -4 \\ 2.5 \end{bmatrix} = -4.3052$$

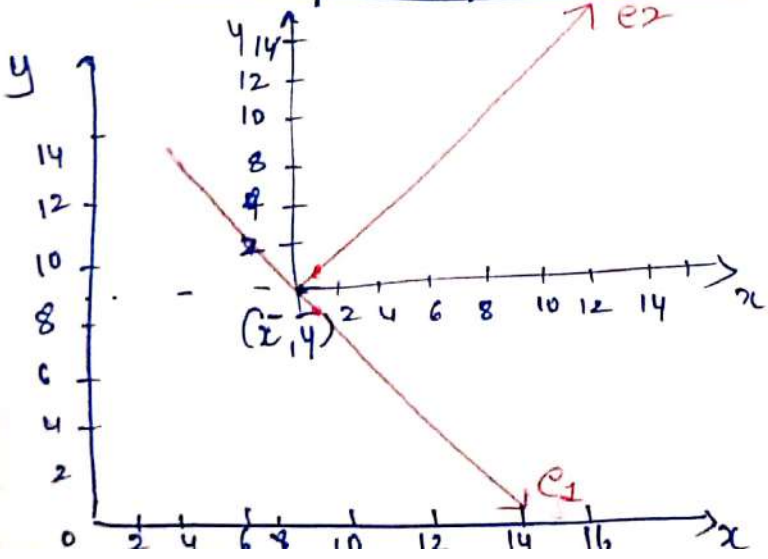
$$P_{12} = \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} 8 - 8 \\ 4 - 8.5 \end{bmatrix} = 3.73$$

$P_{13} = 5.69$

$P_{14} = -5.12$

	1	2	3	4
PC1	-4.3	3.7	5.6	-5.12

$\Rightarrow$ new data set with dimension 1



Linear Discriminant Analysis :-

→ LDA is a dimensionality reduction technique used as a pre-processing step for pattern-classification & Machine learning applications.

→ LDA is similar to PCA but LDA in addition finds the axes that maximizes the seperation between multiple classes.

Goal :- To project a feature space (N-dimensional data onto a smaller subspace k ($K \leq n-1$) while maintaining the class discriminating information

Example :- Lets take a 2-D dataset-

$C_1 \Rightarrow x_1 = (x_1, x_2) = \{(4,1), (2,4), (2,3), (3,6), (4,4)\}$

$C_2 \Rightarrow x_2 = (x_1, x_2) = \{(9,10)(6,8)(9,5)(8,7)(10,8)\}$

Step 1 :- Compute within-class scatter matrix (S_W)

$$S_W = S_1 + S_2$$

$S_1 \Rightarrow$ Covariance matrix for class C_1

$S_2 \Rightarrow$ Covariance matrix for class C_2.

i) finding S_1

$$S_1 = \sum_{x \in C_1} (x - \mu_1)(x - \mu_1)^T$$

$\mu_1 \rightarrow$ mean of class C_1

$$\mu_1 = \left\{ \frac{4+2+2+3+4}{5} , \frac{1+4+3+6+4}{5} \right\}$$

$$\mu_1 = [3.00 \quad 3.60]$$

III^{ly} $\mu_2 = [8.4 \quad 7.60]$

$$\therefore (x - \mu_1) = \begin{bmatrix} 1 & -1 & -1 & 0 & 1 \\ -2.6 & 0.4 & -0.6 & 2.4 & 0.4 \end{bmatrix}$$

Now, For Each x we calculate $(x - \mu_1)(x - \mu_1)^T$.

$$\therefore \begin{bmatrix} 1 \\ -2.6 \end{bmatrix} \begin{bmatrix} 1 & -2.6 \end{bmatrix} = \begin{bmatrix} 1 & -2.6 \\ -2.6 & 6.76 \end{bmatrix} \rightarrow (1)$$

$$\begin{bmatrix} -1 \\ 0.4 \end{bmatrix} \begin{bmatrix} -1 & 0.4 \end{bmatrix} = \begin{bmatrix} 1 & -0.4 \\ -0.4 & 0.16 \end{bmatrix} \rightarrow (2)$$

$$\begin{bmatrix} -1 \\ -0.6 \end{bmatrix} \begin{bmatrix} -1 & -0.6 \end{bmatrix} = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 0.36 \end{bmatrix} \rightarrow (3)$$

$$\begin{bmatrix} 0 \\ 2.4 \end{bmatrix} \begin{bmatrix} 0 & 2.4 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 5.76 \end{bmatrix} \rightarrow (4)$$

$$\begin{bmatrix} 1 \\ 0.4 \end{bmatrix} \begin{bmatrix} 1 & 0.4 \end{bmatrix} = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 0.16 \end{bmatrix} \rightarrow (5)$$

Adding (1) + (2) + (3) + (4) + (5) & taking average we get covariance matrix S_1

$$S_1 = \begin{bmatrix} \frac{4}{5} & -2/5 \\ -2/5 & (13 \cdot 2/5) \end{bmatrix} = \begin{bmatrix} 0.8 & -0.4 \\ -0.4 & 2.6 \end{bmatrix}$$

Similary for the class 2, the co-variance matrix is given by

$$S_2 = \begin{bmatrix} 1.84 & -0.04 \\ -0.04 & 2.64 \end{bmatrix} \quad \& \quad \mu_2 = [8.4 \quad 7.6]$$

$$S_W = S_1 + S_2$$

$$S_W = \begin{bmatrix} 2.64 & -0.44 \\ -0.44 & 5.28 \end{bmatrix}$$

Step 2:- Compute Between class scatter matrix (S_B)

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

$$\mu_1 = [3 \quad 3.6) \quad \mu_2 = [8.4 \quad 7.6]$$

$$S_B = \begin{bmatrix} -5.4 \\ -4 \end{bmatrix} \begin{bmatrix} -5.4 & -4 \end{bmatrix} = \begin{bmatrix} 29.16 & 21.6 \\ 21.6 & 16.00 \end{bmatrix}$$

Step 3:- Find the best LDA projection vector

* Similar to principal component Analysis we find this using eigen vectors using Largest Eigen value.

$$S_W^{-1} S_B \cdot W = \lambda \cdot W \longrightarrow (1)$$

(W → Projection vector)

i> Solving the eigen value

$$|S_W^{-1} S_B - \lambda I| = 0$$

$$S_W^{-1} = \begin{bmatrix} 2.64 & -0.44 \\ -0.44 & 5.28 \end{bmatrix}^{-1} = \begin{bmatrix} 0.38 & 0.03 \\ 0.03 & 0.19 \end{bmatrix}$$

$$S_B = \begin{bmatrix} 29.16 & 21.6 \\ 21.6 & 16.00 \end{bmatrix}$$

$$\lambda I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \times \lambda = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$

$$S_W^{-1} \cdot S_B = \begin{bmatrix} 11.72 & 8.6 \\ 4.9 & 3.6 \end{bmatrix}$$

$$\therefore |S_W^{-1} S_B - \lambda I| = \begin{vmatrix} 11.72-\lambda & 8.6 \\ 4.9 & 3.6-\lambda \end{vmatrix} = 0$$

Solving for λ

$$(11.72-\lambda)(3.6-\lambda)-(4.9\times 8.6)=0$$

$$42.19-11.72\lambda-3.6\lambda+\lambda^2-42.14=0$$

$$\lambda^2-15.32\lambda+0.05=0.$$

$$\therefore \lambda_1=15.31 \quad , \lambda_2=3.26\times10^{-3}$$

Substituting highest Eigen value in Eqn (1)

$$\begin{bmatrix} 11.72 & 8.6 \\ 4.9 & 3.6 \end{bmatrix}\begin{bmatrix} w1 \\ w2 \end{bmatrix} = 15.31\begin{bmatrix} w1 \\ w2 \end{bmatrix}$$

$$11.72\,w1+8.6\,w2=15.31\,w1 \longrightarrow (2)$$

$$4.9w1+3.6w2=15.31\,w2 \longrightarrow (3)$$

$$11.72w1-15.31w1+8.6w2=0$$

$$-3.59w1+8.6w2=0 \longrightarrow (4)$$

$$4.9w1+3.6w2-15.3w2=0$$

$$4.9w1-11.7w2=0$$

$$\begin{bmatrix} -3.59 & 8.6 \\ 4.9 & -11.7 \end{bmatrix}\begin{bmatrix} w1 \\ w2 \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$-3.59w1+8.6w2=0$$

$$w1=\frac{8.6}{3.59}w2=2.39w2$$

$$4.9w1-11.7w2=0$$

Let $w2=1$

$$\therefore w1=\frac{11.7}{4.9}=2.38$$

$$\begin{Bmatrix} w1 \\ w2 \end{Bmatrix}=\begin{bmatrix} 2.38 \\ 1 \end{bmatrix}$$

normalizing

$$\begin{Bmatrix} w1 \\ w2 \end{Bmatrix}=\begin{bmatrix} \frac{2.38}{\sqrt{1^2+2.38^2}} \\ \frac{1}{\sqrt{1^2+2.38^2}} \end{bmatrix}$$

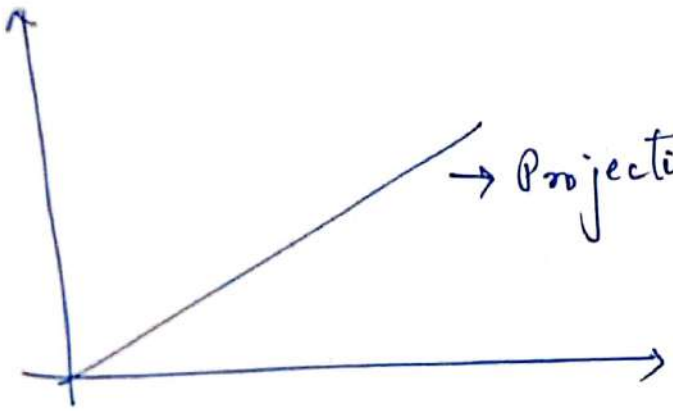
$\therefore$ we get $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0.91 \\ 0.39 \end{bmatrix}$

Step 4 :- Dimension Reduction

$y = w^T x \rightarrow$ Input data sample.

$\downarrow$

Projection vector



→ Projection vector corresponding to highest eigen value

The perceptron algorithm :-

→ Here let us consider two-class model in which input vector x is first-transformed using a fixed nonlinear transformation to give a feature vector $\phi(x)$ & then construct a generalized linear model

$$y(x) = f(w^T \phi(x)).$$

where the non-linear activation function $f(\cdot)$ is a step function of form

$$f(a) = \begin{cases} +1 & a \geq 0 \\ -1 & a < 0. \end{cases}$$

→ In Perceptron algorithm it convinent to use $t = +1$ for C_1 & $t = -1$ for C_2 as target values matching the choice of activation function.

→ To determine the parameter w correctly we can minimize error function but alternatively we can also use perceptron criterion.

→ To derive perceptron criterion based Error we 1st consider

pattern $x_n \rightarrow C_1 \Rightarrow$ when $w^T \phi(x_n) > 0$

$x_n \rightarrow C_2 \Rightarrow$ when $w^T \phi(x_n) < 0$

$t \in \{-1, +1\}$

note :- v Satisfy condition $w^T \phi(x_n) t_n > 0$

All patterns must.

→ perception criterion is given by

$$E_p(w) = -\sum_{n \in M} w^T \phi_n t_n$$

M → denotes the set of misclassified patterns

→ we apply gradient descent algorithm to this error function. The change in weight is given by

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_p(w)$$
$$= w^{(\tau)} + \eta \phi_n t_n .$$

where η → learning rate

T → Integer values of steps of algorithm.

note !- As the weight are change during training the misclassified pattern will change.
set of

Probabilistic Discriminative Models :-

→ There are 2 Models

i) Generative

ii) discriminative.

Analogy :-

Task → Determine the language that someone is speaking.

i) Generative approach :- Learn each language & determine as to which language the speech belongs to.

ii) Discriminative approch :- Determine the linguistic differences without learning any language

→ We wish to lear $f : x \rightarrow y$ eg.., $P(y|x)$

i) In generative we model the joint distribution of all data

ii) In discriminative we model only points at the boundary.

note :- Discriminative models make predictions on the unseen data based on conditional probability

→ Discriminative classification is also called as "informative".

Probability Basics :-

* Prior, conditional & joint probability for random variables.

1) Prior Probability - $P(x)$

2) Conditional probability - $P(x_1|x_2)$, $P(x_2|x_1)$

3) Joint probability :- $x = (x_1, x_2)$ $P(x) = P(x_1, x_2)$

4) Relationship $P(x_1, x_2) = P(x_2|x_1) P(x_1)$
$= P(x_1|x_2) P(x_2)$

5) Independence

$P(x_2|x_1) = P(x_2)$, $P(x_1|x_2) = P(x_1)$, $P(x_1, x_2) = P(x_1) P(x_2)$

Bayes rule :-

Generative.

$$P(c|x) = \frac{P(x|c) P(c)}{P(x)}$$

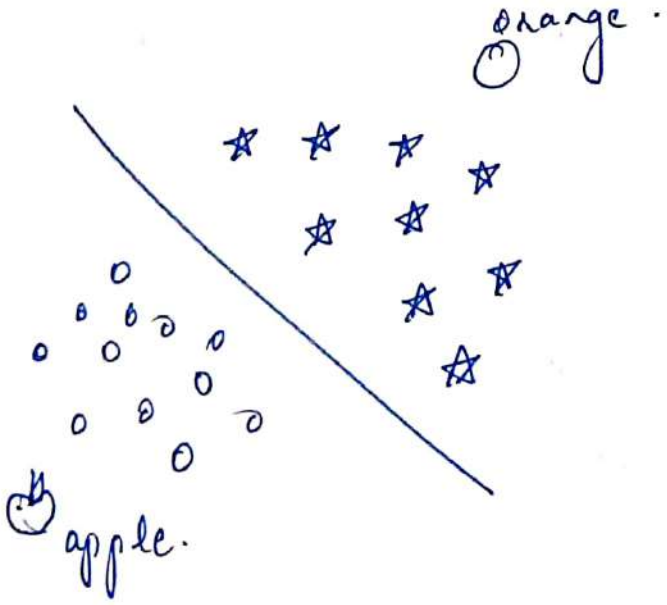
$P(c|x)$ ← discriminative

$$\text{Posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{Evidence}}.$$

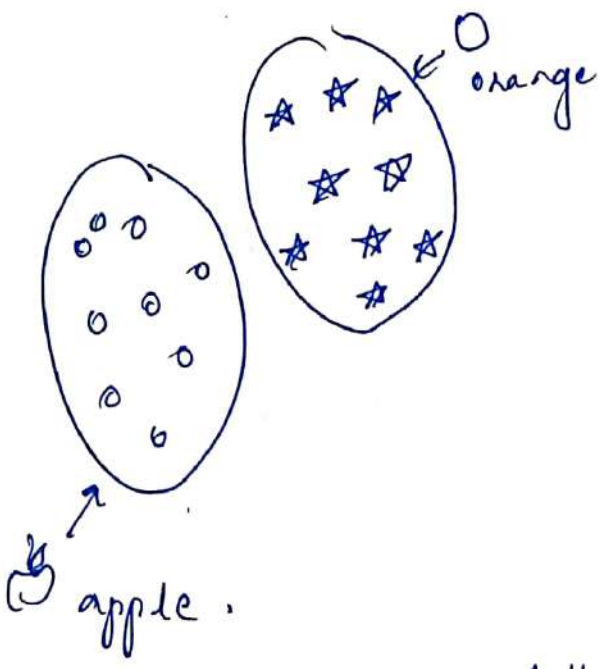
→ Given some data points, the discriminative model learns to classify the data points into their respective classes by learning the decision boundary that separate the classes.

→ The generative models can also classify the given data points, but instead of learning the decision boundary, they learn the characteristics of the classes.

Discriminative model



Generative Model



* Above shows Example of image classification, whether an Input Image is an apple/orange.

i) Discriminative Model learns the optimal

boundary that seperates the apple & orange classes

ii) Generative model learn their distribution by learning the characteristics of apple & orange classes.

→ while both types of models are used to predict $P(y|x)$. Calculation methods for both Models are different.

i) Generative Model → first find joint probability distribution $P(x,y)$, then use Bayes theorem to calculate $P(y|x)$

ii) discriminative Model → Will learn the conditional probability $P(y|x)$ directly.

Ex :- Data set (0,1) (0,1) (1,0) (1,1)

$P(x,y)$	$Y=0$	$Y=1$	
$x=0$		$1/2$	$\Rightarrow P(x_1)$
$x=1$	$1/4$	$1/4$	$\Rightarrow P(x_2)$
	$P(Y)=1/4$	$P(Y)=1/8$	

and $P(y|x) = \frac{P(x,y)}{P(x)}$ or $\frac{P(x,y)}{P(y)}$

$P(x_1) = 1/2 \qquad P(x_2) = 1/2$

$\therefore$ $P(y|x)$ $\quad y=0 \quad y=1$

$x=0$ $\quad$ 1

$x=1$ $\quad 1/2 \quad 1/2$

$\frac{P(x,y)}{P(y)} = \frac{1/2}{1/2}$

Summary :-

Generative Model	Discriminative Model
→ Works good for small data & delivers high accuracy.	→ Accuracy is high → Resource saving → Fewer parameters to be determined.

Probabilistic Models → Discriminative → Tools used for

- Fixed BASIS function
- Logistic regression
- Iterative reweighted least-Square
- multiclass logistic regression

Logistic Regression :-

→ Here we model the posterior probabilities directly assuming that they have a sigmoid-shaped distribution [without Modeling class priors & class conditional densities]

→ The sigmoid-shaped function (σ) is a model function logistic regression.

→ 1st non-linear transformation of inputs using a vector of basis function $\phi(x)$

$$P(C_1|\phi) = y(\phi) = \sigma(w^T\phi)$$

$$P(C_2|\phi) = 1 - P(C_1|\phi)$$

∴ If we have M-dimensional feature space ϕ, we use M+1 adjustable parameters to represent it directly.

→ Here $\sigma(\cdot)$ → logistic sigmoid function or logistic regression.

→ We can use maximum likelihood to determine the parameters of the logistic regression model.

$$\frac{d\sigma}{da} = \sigma(1-\sigma)$$

→ For a data set $\{\phi_n, t_n\}$, $t_n \in \{0, 1\}$ &

$\phi_n = \phi(x_n)$ with $n = 1 \ldots N$.

likelihood function can be written as

$$P(t|w) = \prod_{n=1}^{N} y_n^{t_n} \{1 - y_n\}^{1-t_n}$$

where $t = \{t_1, \ldots t_N\}^T$

$y_n = P(c_1|\phi_n)$ & Error Equation

$$\nabla E(w) = \sum_{n=1}^{N} (y_n - t_n)\phi_n$$

Iterative reweighted least squares :-

→ Logistic regression has no closed-form solution of $\nabla E(w) = 0$. due to non-linearity of logistic sigmoid function.

→ ∴ Error function can be minimized by an efficient iterative technique.

i.e., $w^{(new)} = w^{(old)} - H^{-1} \nabla E(w)$

where $H \rightarrow$ Hessian matrix $\rightarrow$ 2nd derivatives of $E(w)$ wrt 'w'

$$H = \nabla^2 E(w)$$

& $\nabla E(w) = \phi^T \phi w - \phi^T t$.

$$\therefore H = \nabla \nabla E(w) = \sum_{n=1}^{N} \phi_n \phi_n^T = \phi^T \phi$$

$\downarrow$

design matrix

Module 3 :-

K- Means algorithm :-

Clustering :-

→ clustering is the classification of objects into different groups or partitioning of a data set into subsets or clusters. Each subset of data share some common trait

Types of clustering are

1) Hierarchical algorithms
2) Partitional clustering.

* K means algorithm comes under partitional clustering type of clustering.

Common distance Measures :-

→ Distance measure will determine how the similarity of two Elements is calculated & will influence the shape of the clusters.

Euclidean distance (2-norm distance) is given by

$$d(x,y) = \sum_{i=1}^{p} |x_i - y_i|$$

→ K means clustering is an algorithm to cluster n objects based on attributes into K partitions. where $K < n$.

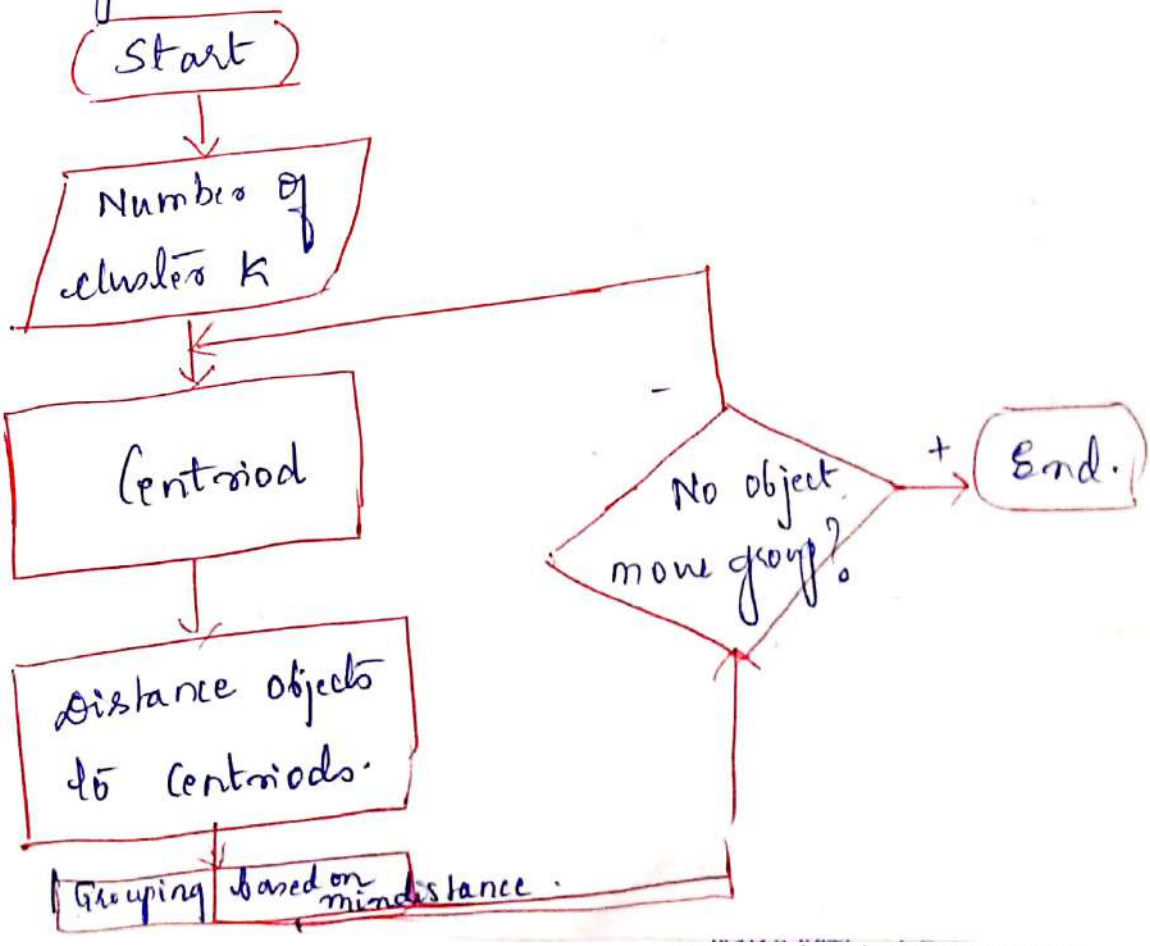
→ It assumes that the object attributes form a vector space.

note :- Supervised algorithms → Target variables are known

Unsupervised Algorithms → set of features are known, target- or label are not known

clustering ⟷ find clusters of data

How the Algorithm works :-



Step 1 :- For each data point, place it in a cluster whose current centroid to which it is nearest.

Step 2 :- After all data points are assigned, update the locations of centroid of the K clusters.

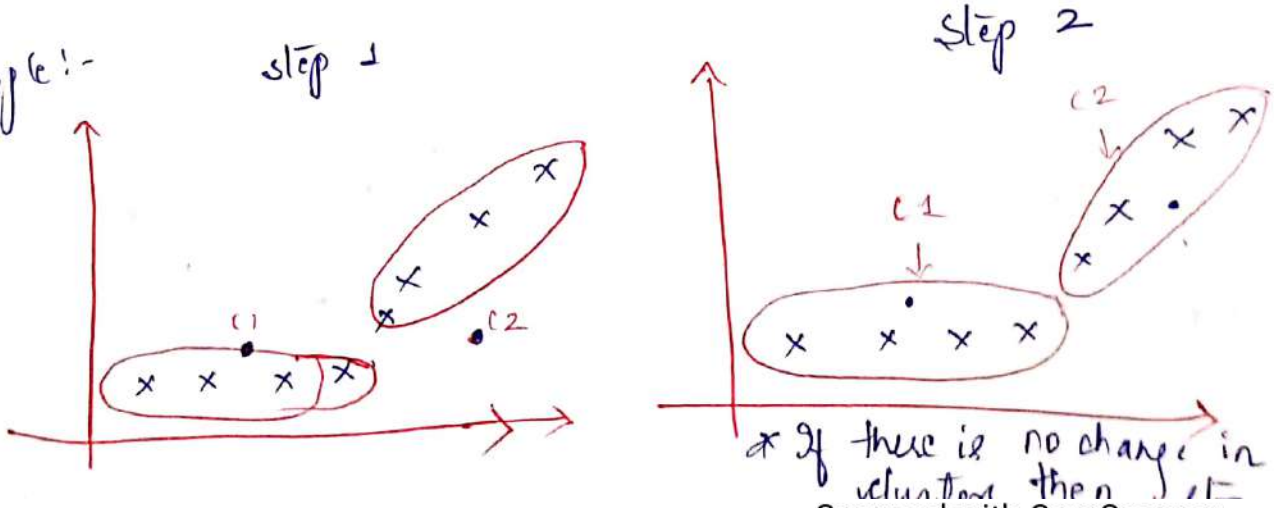
Step 3 :- Rearrign all points to their closest Centriod

note :- Sometimes. data points changes its clusters centriods.

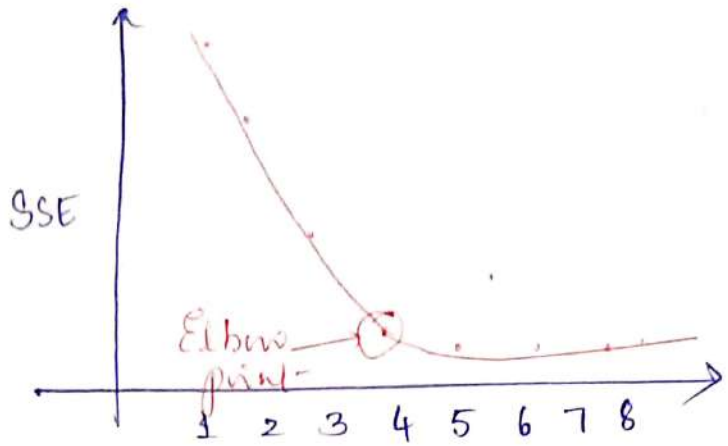
Step 4 :- Repeat step 2 & Step 3 untill no data points change clusters & untill all converge.

Con note :- Convergence - Points do not- move between clusters & centroids are stable.

Example :-



→ How to determine correct number of clusters (K)



* Elbow Technique.

So value of $K = 4$ for a set of data points.

note :- SSE [Sum of Square Error]

$$SSE = \sum_{i=0}^{n} dist(x_i - C_i)^2$$

C_i → cluster number ← C1, C2, C3 ....

$$\therefore SSE = SSE_1 + SSE_2 + \ldots + SSE_K$$

SSE_1 → for cluster 1
SSE_2 → cluster 2
SSE_K → for cluster K.

Conclusion :-

→ K-means algorithm is useful for undirected Knowledge discovery & is relatively simple.

Fuzzy K-means Algorithm :-

clustering

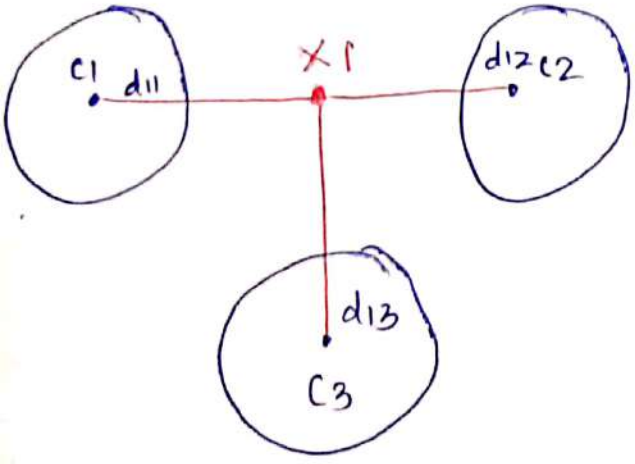
- Hard clustering
- Soft clustering

note :- K-means is a hard-clustering approach.

→ In soft or fuzzy clustering, instead of putting each data points into seperate clusters, a probability of that point to be in that cluster assigned.

→ In soft clustering or fuzzy clustering, each data point can belong to multiple clusters along with its probability score or likelihood.

→ Here fuzzy set theory is used to obtain the optimal values of the centroid.



→ In this technique the particular vector (x_1) belongs to all the clusters with different membership value.

note :- Sum of membership value is '1'.

→ Consider vector x_1 belongs to the cluster 1 whose centroid is c_1 then $(x_1 - c_1)^2$ must be minimized. But if x_1 belongs to cluster 1 with membership value m_1 then $m_1 (x_1 - c_1)^2$ has to be minimized.

→ The fuzzy K-means problem is treated as the optimization technique for obtaining the membership values along with the centroids of the individual clusters such that

$$\sum_{i=1}^{N} \sum_{K=1}^{K} M_{ik}^2 (x_i - c_K)^2 \text{ is minimized}$$

M_{ik} → Membership value of vector x_i belongs to Cluster K

x_i → i^{th} vector

c_k → c_K is K^{th} Centroid.

note :- Here we use converging the membership values to find an optimal solution.

Algorithm :-

Step 1 :- Initialize the membership values (M_{ij}) randomly.

Step 2 :- Compute the Centroids of the clusters

$$C_1 = \frac{[\text{vector 1} * (\text{membership values of vector 1 belonging to cluster 1})]^2 \text{ i.e., } M11^2 + M_{12} + \ldots}{\text{Sum of the squared values of membership values belonging to the cluster 1.}}$$

III^{rd} all Centroids (Eg :- $C_2, C_3 \ldots C_6$) are obtained.

Step 3 :- Update the membership values M using new Centroids.

$$M_{11} = \frac{1}{\left[\frac{(x_1 - c_1)^2}{(x_1 - c_1)^2}\right]^2 + \left[\frac{(x_1 - c_1)^2}{(x_1 - c_2)^2}\right]^2 + \ldots + \left[\frac{(x_1 - c_1)^2}{(x_1 - c_6)^2}\right]^2}$$

M_{11} : vector, Centroid.

Step 4 :- Compute the sum of squared difference between the previous membership value & current membership value.

$$\left[m_{11}^{old} - m_{11}^{new}\right]^2 + \left[m_{12}^{old} - M_{12}^{new}\right]^2 + \ldots \quad (6)$$

* If computed value is not less than the threshold then goto Step 2.

* If computed value is less than the threshold then stop the iteration.

Self Organizing Maps :-

→ Supervised training techniques, where there is a target output for each input pattern & the network learns to produce the required output.

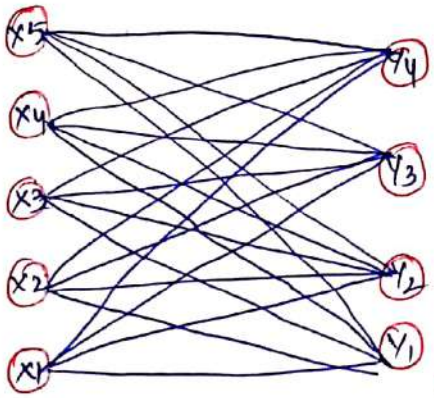
→ Unsupervised training is where the network learn to form their own classifications of the training data without external help.

→ One particularly interesting class of unsupervised based on competitive learning in which the o/p neurons compete amongst themselves to be activated & this activated neuron is called a winner-takes-all neuron or winning neuron.

→ Such competation can be implemented by having lateral inhibition connections between the neurons. Resulting in neurons organizing themselves. Such a network is called as self Organizing Maps (SOM)

→ The goal of SOM is to transform an incoming signal pattern of arbitrary dimension into 1 or 2 dimensional discrete map, ∴ setting up a SOM.

→ We shall concentrate on a particular Kind of SOM known as Kohonen network. This is a feed forward structure with a single computational layer arrange in rows & columns. Each neuron is fully connected to all source nodes in the input layer.



→ The aim is to learn a "feature map" from the spacially "continuous Input Space" to a low dimensionality spatially discrete output space. which is formed by arranging the computational neurons into a grid.

Algorithm :-

Step 1 :- Each node weights are Initialized

Step 2 :- Every node in the network is Examined to Calculate which one's weight are most like the Input vector. The winning node is commonly known as the "Best matching unit"

Step 3 :- Update the weight after finding the winning neuron

Step 4 :- Training will stop if Input & weight vectors are Identical otherwise goto step - 2

$$w^{new} = w^{old} + \alpha\eta(\text{I/p} - \text{weight vector})$$

K-means algorithm :-

*) Assume Euclidean space / distance

*) start by picking K, the number of clusters

*) Initialize cluster by picking one point per cluster

note: for this moment, assume we pick the K points at Random.

step 1: for each point, place it in the cluster whose current centroid it is nearest

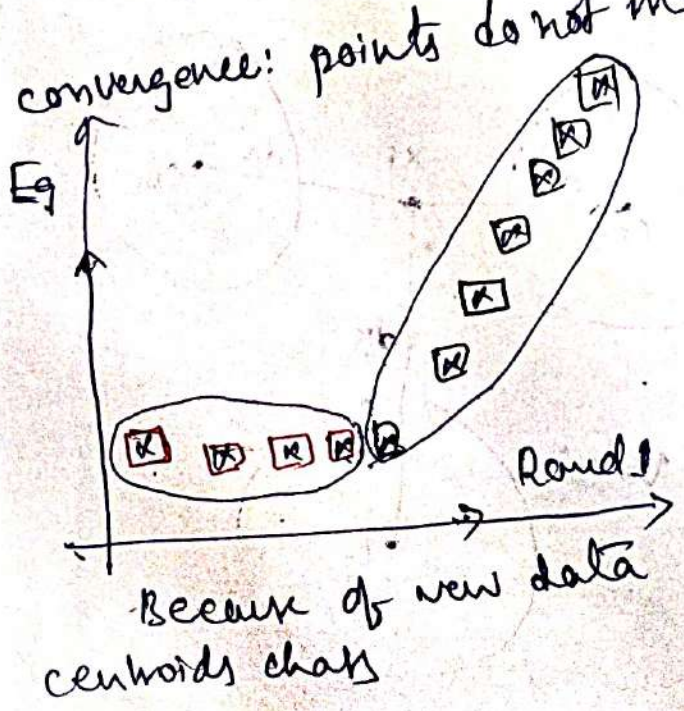
step 2: After all points are assigned, update the locations of centroid of the K clusters

step 3: Reassigns all points to their closest centroid

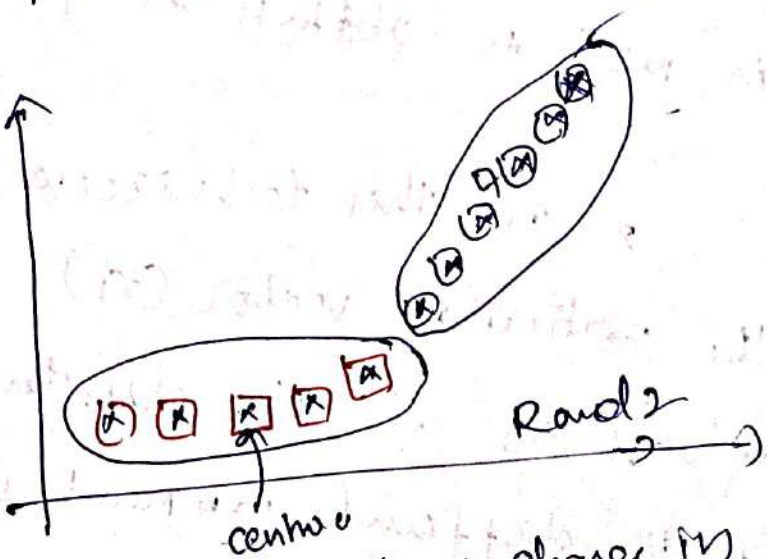
note: some times moves points b/w clusters

step 4: Repeat step 2 and step 3 until convergence

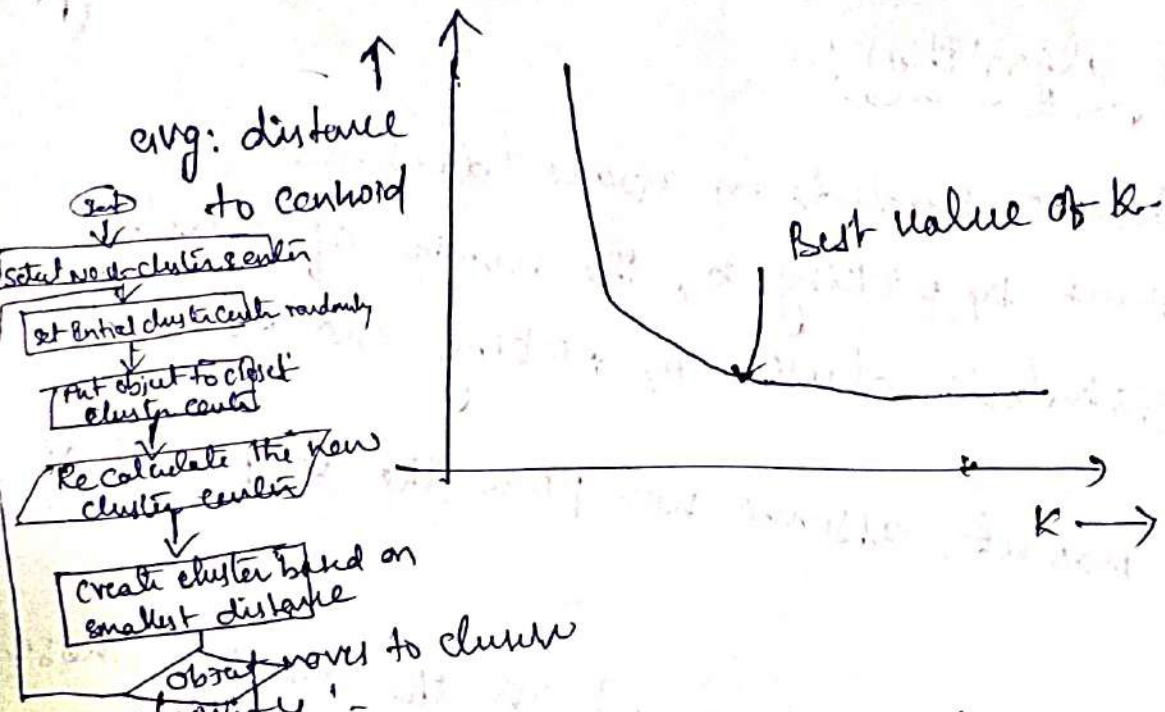
convergence: points do not move b/w clusters and centroids stable



Because of new data centroids chats



In round-'3' no change in centroid position.



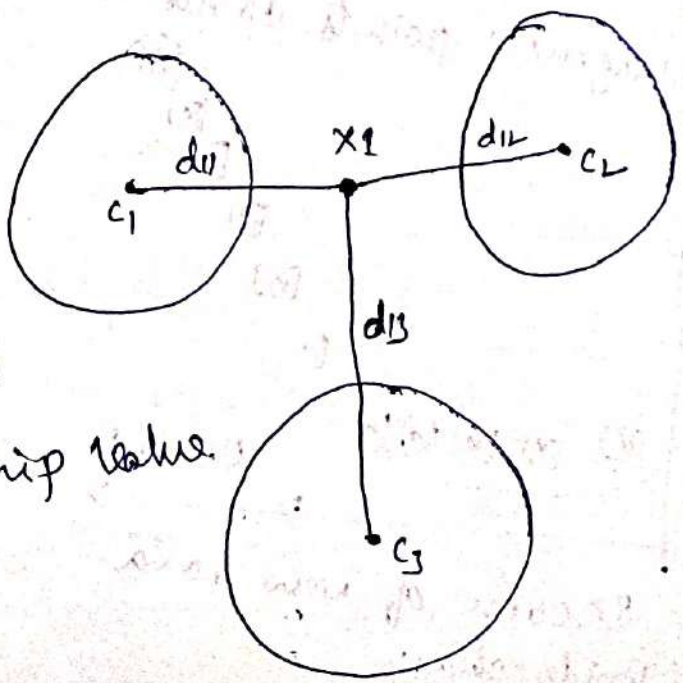
1) In each ~~used~~ round, we have to examine each input point exactly once to find closest centroid

2) If data is too large then computation or ~~If~~ iterations are increased.

2) Fuzzy K-means algorithm

1. In fuzzy-K means techniques, Fuzzy set theory is used to obtain the optimal values of the centroid

2. In this techniques the particular vector (x_1) belongs to all the clusters with different ~~is~~ membership value.



Note: Sum of membership value is '1'

3. consider the vector x_1 belongs to the cluster 1 whose centroid is c_1, then $(x_1 - c_1)^2$ must be minimized. But if x_1 belongs to cluster one with membership value m_1 then $m_1 (x_1 - c_1)^2$ has to be minimized. Similarly x_1 belongs to cluster-2 with membership value m_2 then $(x_1 - c_2)^2 . m_2$ has to be minimized.

The fuzzy k-means problem is treated as the optimization technique for obtaing the membership values along with the centroids of the individual clusters such that

$$\sum_{i=1}^{N} \sum_{k=1}^{K} M_{ik}^2 . (x_i - c_k)^2 \text{ is minimized}$$

M_{ik} → membership value of vector x_i belongs to the cluster K.

x_i → i^{th} vector

c_k → c_k is k^{th} centroid.

Note: Here we must converging the membership values.

fuzzy K-means Algorithm:

step 1: Initialize the membership values (M_{ij}) randomly

step 2: Compute the centroids of the clusters (Eg: 6 clusters)

hint $C_1 = \frac{\sum x_i M_i}{\sum M_i}$

hint

C_1 = [vector 1 * (membership value of vector 1 belongs to cluster 1) i.e M_{11} + vector 2 * (membership value of vector 2 belongs cluster 1) i.e M_{12} + ... vector 100 * (membership value of vector 100 belongs to cluster 1) i.e M_{1100}] / sum of the squared values of the membership values belongs to the cluster-1.

By all centroids (Eg: $C_2, C_3 \ldots C_6$) are obtained

step 3: Update the membership values M using new (current) values of centroids.

$$M_{11} = \frac{1}{\left[\frac{(x_1-c_1)^2}{(x_1-c_1)^2}\right]^2 + \left[\frac{(x_1-c_1)^2}{(x_1-c_2)^2}\right]^2 + \ldots + \left[\frac{(x_1-c_1)^2}{(x_1-c_6)^2}\right]}$$

Similarly other membership values are computed.

$$M_{45} = \frac{1}{\left[\frac{(x_4-c_5)^2}{(x_4-c_1)^2}\right]^2 + \left[\frac{(x_4-c_5)^2}{(x_4-c_2)^2}\right]^2 + \ldots + \left[\frac{(x_4-c_5)^2}{(x_4-c_6)^2}\right]}$$

M_{45}: vector 4, centroid 5

step 4 : compute the sum of squared difference between the previous membership value and the current membership value.

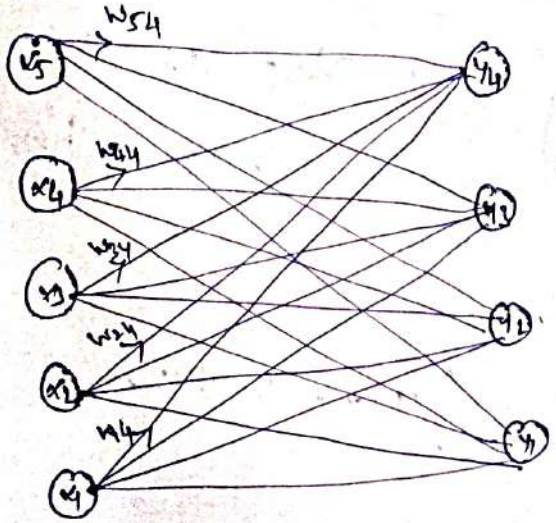
If computed value is not less than the threshold then go to step-2.

If computed value is less than the threshold then stop the iteration.

Eg : $(M_{11}^{old} - M_{11}^{new})^2 + (M_{12}^{old} - M_{12}^{new})^2 + \ldots\ldots$

KSOM (Kohonen self-organizing Maps)

This has a feed-forward structure with a single computational layer of neurons. Each neuron is fully connected to all the source units in the I/p layers.



Input units

cluster unit

The aim is to learn a "feature map" from the spacially "continuous input space", in which our I/p vectors live, to the low dimensional spatially discrete output space which is formed by arranging the computational neurons into a grid.

KSOM (Algorithm)

step 1: Each node's weights are initialized
(all I/p & o/p are normalized)

step 2: Every node in the network is examined to calculate which ones' weights are most like the I/p vector.
The winning node is commonly known as the "Best matching unit".

step 3: Update the weight after find the winner neuron
(more importance to the neuron which is closer to winner, less importance to the neuron which is far from winner)

step 4: Training will stop if I/p & weight vectors are identical. otherwise go to step-2.

$$W^{new} = W^{old} + \alpha\eta\,(\text{I/p} - \text{weight vector})$$

Particle Swarm Algorithm :-

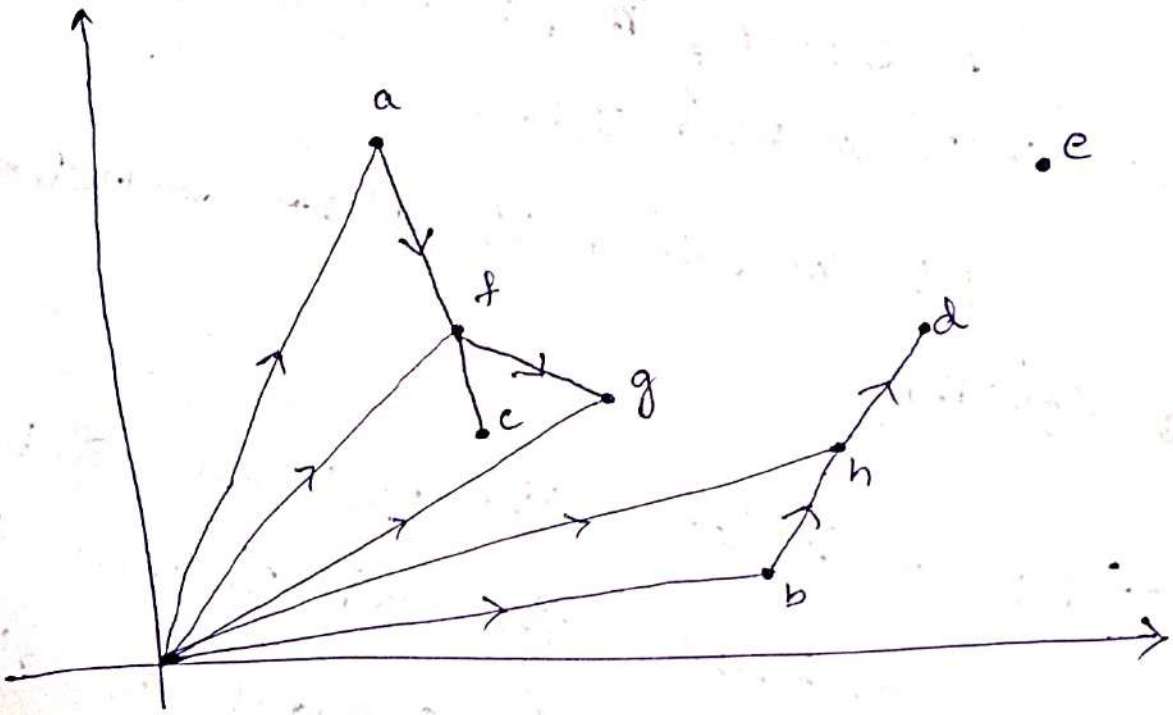


fig: vector Representation of PSO Algorithm.

1) Let the position of the swarms be at 'a' and 'b'. Both are trying to reach 'e'.

2) Let 'a' decided to move 'c' and 'b' is decided to move 'd'.

3) The distance b/w position 'c' and 'e' is greater than the distance b/w d and 'e'.

4) So, 'd' is a common position decided by the a and b

5) The position 'c' is the Individual decision taken by 'a', The position 'd' is the " " a by 'b'. the position 'd' is common position to take by both a & b.

Now. let us consider three swarms (A, B, C) are ~~hoping~~ trying to reach the particular destination point 'D'.

A decides 'A'', B decides B', C decides C' "
~~which is the~~ as next position.

~~Let the distance b/w. A' &~~ ... where

PSO algorithm

step 1 : Initialize the positions a, b, c, d, e

step 2 : Initialize the next positions decided by the Individual swarms as a', b', c', d', e'

step 3 : Global decision regarding the next position is computed as follows.

compute f(a', b, c, d, e), f(a, b', c, d, e), f(a, b, c', d, e) f(a, b, c, d', e) and f(a, b, c, d, e'). find the minimum among the computed values.

If f(a', b, c, d, e) is minimum among all, the global position decided regarding the next position is a'.

If f(a, b', c, d, e) is minimum the global position is b' and so on.

Let the selected global positions is represented as "global"

step 4 : Net value for 'a' is computed as the linear combination of 'a', (a'-a), (global - a). i.e

weight of individual decision ↑ weight of random decision

$$next\,a = a + C1 * RAND * (a'-a) + C2 * RAND * (global - a)$$

$$next\,b = b + C1 * RAND * (b'-b) + C2 * RAND * (global - b)$$

$$\vdots$$

$$next\,e = e + C1 * RAND * (e'-e) + C_2 * RAND * (global-e)$$

step 5 : change the current value for a, b, c, d, e as next a, next b, next c, next d, ~~next~~ next e.

step 6 : If f(next a, b, c, d, e) is less than f(a, b, c, d, e) then update the value for 'a' as "next a" otherwise 'a' is not change (do this for b', c', d', e' also.)

step 7 : Repeat the steps 3 to 6 for much iterations to reach the final decision.

The values C1 & C2 are decided based on the weightage given to individual decision and global decisions respectively.

Let $\Delta a(t)$ is the change in the value for updating the value for 'a' in 't'th iteration. Then next a at (t+1)th iteration can be computed using the following formula.

This is considered as the velocity for updating the position of the swarm in every iteration

$$next\,a(t+1) = a(t) + \Delta a(t+1)$$

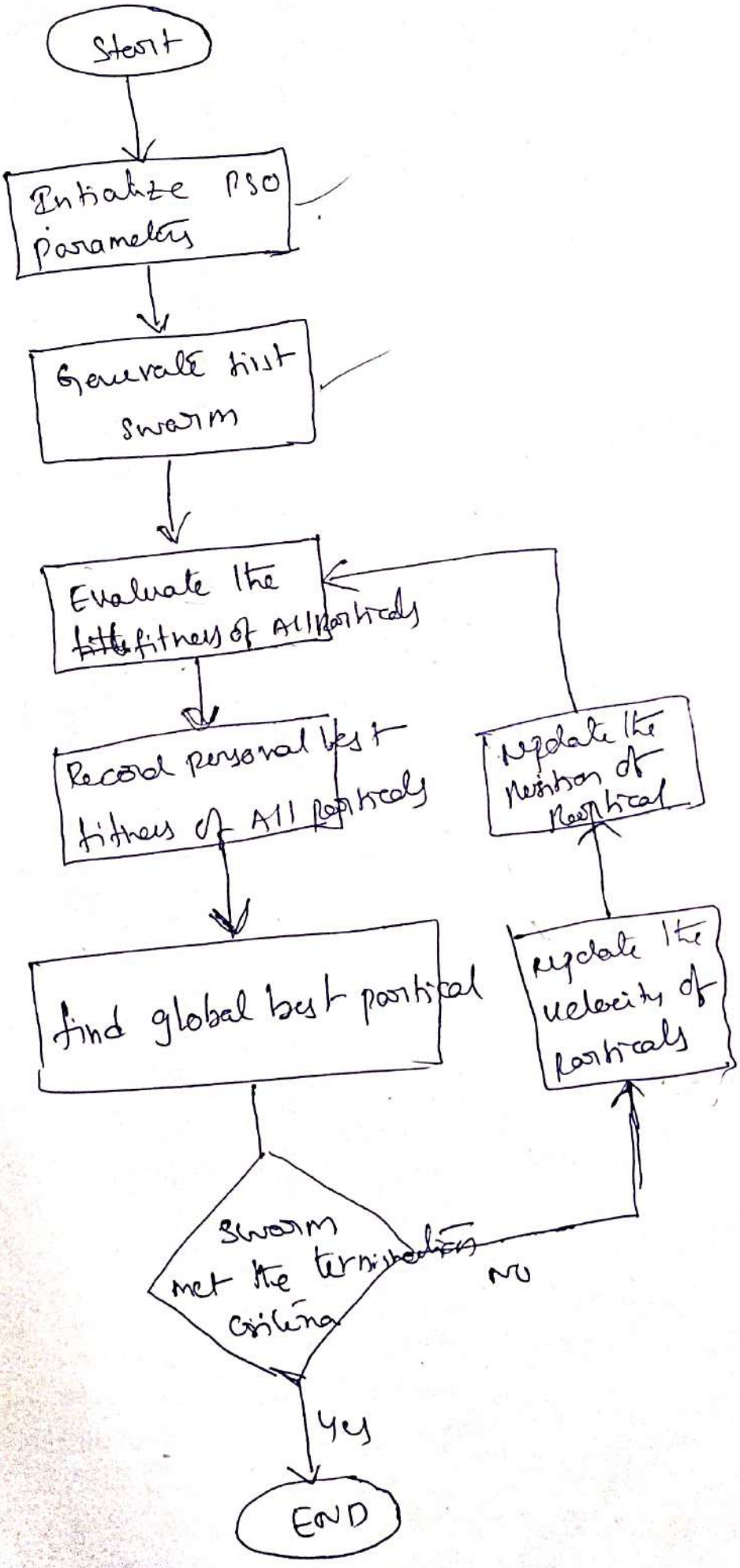
where

$$\Delta a(t+1) = c1 * rand * (a' - a) + c2 * rand * (Global\,a) + w(t) * \Delta a(t)$$

w(t) is the weight at t^{th} iteration. The value for 'w' is adjusted at every iteration as given bellow, where 'iter' is total no. of iterations used.

$$w(t+1) = \frac{w(t) - t * w(t)}{iter}$$

PSO Algorithm flow chart



Ant colony optimization

Ant colony optimization technique exploit the natural behaviour of ants in finding the shortest paths to reach destination from the source

Algorithm

consider the problem of finding the optimum order in which the numbers from 1 to 8 are arranged so that the cost of the order is maximized. cost of the particular order is computed using the two ~~matrix~~ matrices A and B as described below

a_1							
a_2	a_3						
a_4	a_5	a_6					
a_7	a_8	a_9	a_{10}				
a_{11}	a_{12}	a_{13}	a_{14}	a_{15}			
a_{16}	a_{17}	a_{18}	a_{19}	a_{20}	a_{21}		
a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}	a_{28}	
a_{29}	a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}

Matrix A

b_1							
b_2	b_3						
b_4	b_5	b_6					
b_7	b_8	b_9	b_{10}				
b_{11}	b_{12}	b_{13}	b_{14}	b_{15}			
b_{16}	b_{17}	b_{18}	b_{19}	b_{20}	b_{21}		
b_{22}	b_{23}	b_{24}	b_{25}	b_{26}	b_{27}	b_{28}	
b_{29}	b_{30}	b_{31}	b_{32}	b_{33}	b_{34}	b_{35}	b_{36}

Matrix B

The cost of the order [3 2 1 5 7 2 4 6 8] is computed as the sum of the products of values obtained from the positions (3,1) of (1,3), (2,2), (3,1), (4,5) of (5,4), (5,7) of (7,5), (6,2), (7,4), (8,6) of matrix A & B

$(a_6 \times b_6) + (a_3 \times b_3) + (a_6 \times b_6) + (a_{14} \times b_{14}) + (a_{26} \times b_{26}) + (a_{27} \times b_{27}) + (a_{17} \times b_{17}) + (a_{36} \times b_{36})$

The problem of finding the optimal order is achieved using Ant colony optimization technique as described below.

step 1: Generate the five sets of orders randomly which are treated as the initial values selected by the 5 ants respectively.

The following are the orders selected by the 5 ants along with the corresponding cost as given below -

ANT	ORDERS	COST
ANT 1	7 1 3 4 6 2 5 8	C1
ANT 2	4 2 8 5 7 1 3 6	C2
ANT 3	5 1 3 2 4 7 6 8	C3
ANT 4	3 7 4 1 8 6 5 2	C4
ANT 5	8 1 2 4 3 5 7 6	C5

step 2: Pheromone is the value assigned for its number for the ith position of the sequence selected by all the ants. Initially all the values of matrix are completely filled up with ones. The values of pheromone matrix are updated in every iteration.

pheromone matrix (n+1) = pheromone matrix (n) + updating pheromone matrix.

	1	2	3	4	5	6	7	8
1	0	$\frac{1}{c_1}+\frac{1}{c_3}+\frac{1}{c_5}$	0	$\frac{1}{c_4}$	0	$\frac{1}{c_2}$	0	0
2	0	$\frac{1}{c_2}$	$\frac{1}{c_5}$	$\frac{1}{c_3}$	0	$1/c_4$	0	$\frac{1}{c_4}$
3	$\frac{1}{c_4}$	0	$\frac{1}{c_1}+\frac{1}{c_3}$	0	$\frac{1}{c_5}$	0	$\frac{1}{c_2}$	0
4	$\frac{1}{c_2}$	0	$\frac{1}{c_4}$	$\frac{1}{c_5}+\frac{1}{c_1}$	$\frac{1}{c_3}$	0	0	0
5	$\frac{1}{c_3}$	0	0	$1/c_2$	0	$\frac{1}{c_5}$	$\frac{1}{c_4}+\frac{1}{c_1}$	0
6	0	0	0	0	$\frac{1}{c_1}$	$\frac{1}{c_4}$	$\frac{1}{c_3}$	$\frac{1}{c_2}+\frac{1}{c_5}$
7	$\frac{1}{c_1}$	$\frac{1}{c_4}$	0	0	$1/c_2$	$\frac{1}{c_3}$	$\frac{1}{c_5}$	0
8	$1/c_5$	0	$\frac{1}{c_2}$	0	$\frac{1}{c_4}$	0	0	$\frac{1}{c_1}+\frac{1}{c_3}$

Step 3: Next set of orders selected by the ants is as described below.

Eg: The random positions 7 8 2 3 6 4 1 5 (say)

Let the order decided by the ant 1 be represented as [u1, u2 u3 u4 u5 u6 (u7) u8] (selected)

* The value for the u7 in the order as mentioned above is selected first as the first number of the ~~ran~~ randomly generated position sequence is 7.

* To select the value for ant 7, 7th column of the pheromone matrix is used as bellow,

$$\text{column 7} = [0 \quad 0 \quad \tfrac{1}{c_2} \quad 0 \quad \tfrac{1}{c_4}+\tfrac{1}{c_4} \quad \tfrac{1}{c_3} \quad \tfrac{1}{c_5} \quad 0]$$

* Now calculate the normalized value of column 7. This is also called as the probability vector used for selecting the value for ant 7 in the order

$$S = [\tfrac{1}{c_2}+\tfrac{1}{c_1}+\tfrac{1}{c_4}+\tfrac{1}{c_3}+\tfrac{1}{c_5}] = \text{value}$$

$$\text{Normalized vector } 7 = \frac{\text{column 7}}{S} = \left[\frac{0}{S} \quad \frac{0}{S} \quad \frac{1/c_2}{S} \quad \frac{0}{S} \quad \frac{c_4+\frac{1}{c_4}}{S} \quad \frac{\frac{1}{c_3}}{S} + \frac{1/c_5}{S} \quad \frac{0}{S}\right]$$

$$= [P_1 \; P_2 \; P_3 \; P_4 \; P_5 \; P_6 \; P_7 \; P_8]$$

* find the lowest value position, for Example P_5 is the lowest value the 5 is assigned to ant 7.

Note: position no. '5' is not used again.

* The next value in position sequence is 8, so column '8' is selected from pheromone matrix, and select the normalized value

$$\text{column 8} = [0 \quad 1/c_4 \quad 0 \quad 0 \quad \tfrac{1}{c_2}+\tfrac{1}{c_1} \quad 0 \quad \tfrac{1}{c_1}+\tfrac{1}{c_3}]$$

$$\therefore \quad S = \left[\frac{1}{C_4} + \left(\frac{1}{C_2} + \frac{1}{C_7}\right) + \left(\frac{1}{C_1} + \frac{1}{C_3}\right) \right]$$

normalized vector $8 = \dfrac{\text{column } 8}{S} = (a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7)$

find the lowest value in normalized vector '8' and replaced with V8. Exam If a_3 is lowest then number '3' is assigned to V8.

* This process is repeated to obtain the order selected by the ~~ant~~ Ant 2, Ant 3, Ant 4, Ant 5.

* This is called one Iteration.

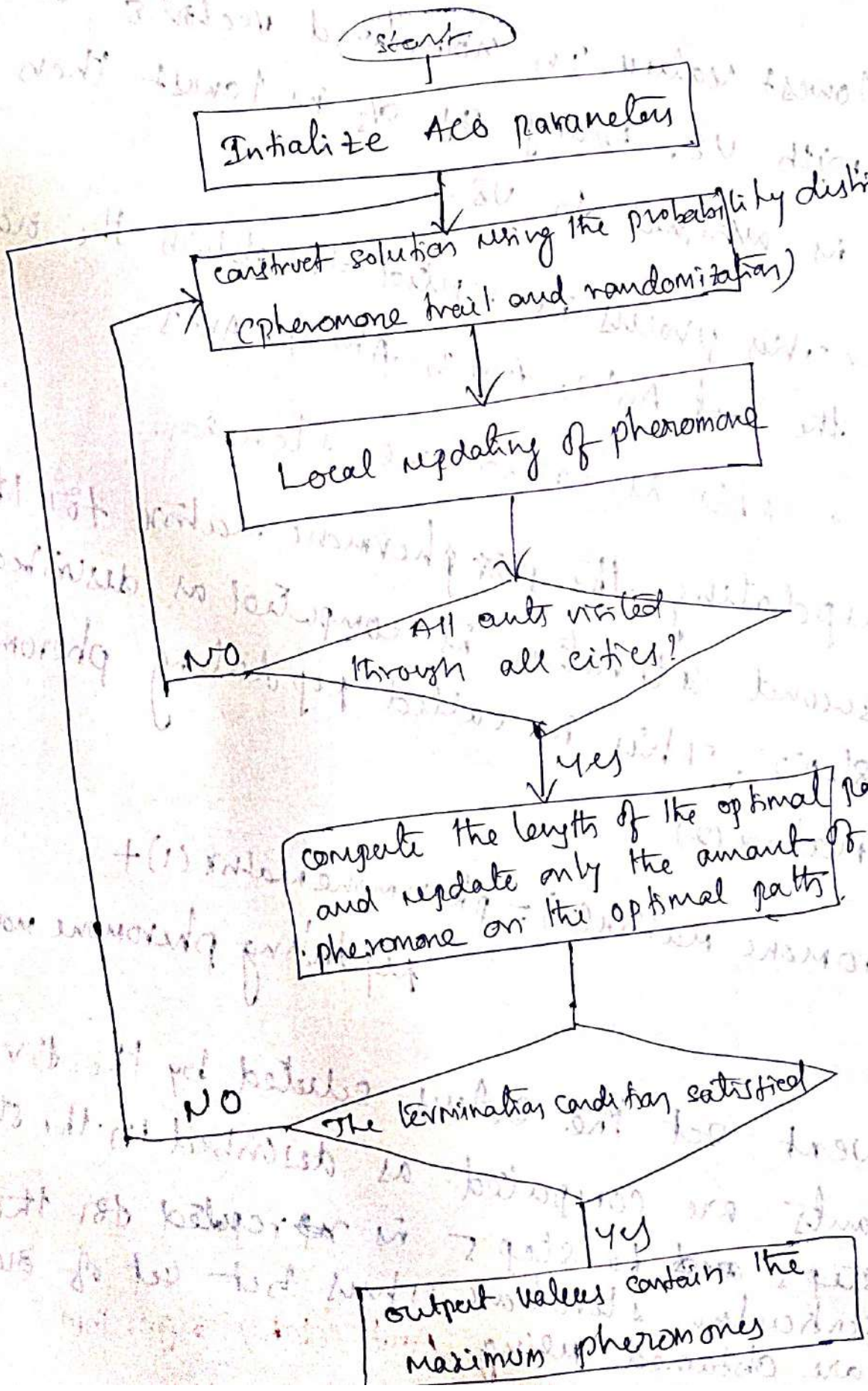
step 4 : Updating the ~~pre~~ pheromone matrix for the second Iteration is computed as described in step '2'. This is called updating pheromone matrix (2)

$\therefore$ pheromone matrix(2) = pheromone matrix (1) + updating pheromone matrix (2)

step 5 : Next set the orders selected by the five ants are computed as described in the step '3'

step 6 : step 3 ~~and~~ to step 5 is ~~re~~ repeated for the particular Iterations. Thus best set of orders are obtained using Ant colony algorithm

Note: Best among the best set of orders selected by the five ~~ants~~ ants in the last iterations is selected as the gobel best order.



...al Foraging

1- Bacterial tries to move towards the region where more nutrients are available (i.e) moving towards Increasing concentrations of nutrients

$J(\theta) \downarrow\downarrow \rightarrow$ objective functions

θ = the position of the Bacteria

$J(\theta)$ = is the nutrient profile

$J(\theta) < 0 \longrightarrow$ presence of Nutrients

$J(\theta) = 0 \longrightarrow$ Neutral

$J(\theta) > 0 \longrightarrow$ noxious substance

1. Chemotaxis: Tendency to move towards distant sources of nutrients

a) Tumbling [change in direction]

$\theta^{new} = \theta^{i} + C\phi \rightarrow$ unit walk in random direction

$\phi = \frac{\Delta}{\sqrt{\Delta^T \Delta}}$

C → chemotactic step size

b) Swimming:

movement in same direction as the previous tumbling.

Note: Bacterium will tend to keep moving if it is headed in the direction of an increasingly favorable environment

If at θ^{new}, J^{new} is lower (better) than the cost at θ_i, another swimming step is taken and is continued as long as it continues maximum no of. swimming steps Ns.

2. Reproduction

After Ns chemotactic steps, Reproduction steps is taken

a) Least healthy bacteria die

b) "S_r" healthiest bacteria split into two bacteria which are then placed in the same location

Note: Health of the bacteria are measured by objective function J.

Higher cost represents that the bacterium did not get many nutrients (work as healthy) and hence unlikely to reproduce

3. Elimination - Dispersion

In real word process, bacteria can be dispersed to new locations, for example wind dispersal.

Individual bacterium is stochastically selected for elimination from the population and is replaced by a new Bacterium located at a random new location within the optimization domine, according to the present probability Ped.

4. social communication

Bacterium that has uncovered good source of nutrients can release a "chemical attract signal" which attracts the other bacteria to converge to its current location. (This help in making all the bacterium to come closer)

Also "chemical Repellent signal" is released to ensure that the bacteria do not get too close to each other.

Let Θ^i be the position of the i^{th} bacteria then the objective function J is modified as,

$$J_{SW}^i = J^i + J_{cc}(\Theta^i, \Theta)$$

$J_{cc}(\theta^i, \theta)$ is chosen such that the distance b/w ith bacteria and all other bacterium are maximized (for Repellent signal) and distance b/w "ith" bacteria and all other bacterium are minimized (for attractan signal).

Note: The width of the attractant and repellent signals are used based on the requirent.

$$J_{cc}(\theta^i, \theta) = -M\left(\sum_{k=1}^{s} e^{-w_a \|\theta^k - \theta^i\|^2}\right) - \sum_{k=1}^{s} e^{-w_r \|\theta^k - \theta^i\|^2}$$

width of the attractan

width of the repellent (a)

let

P = Dimensions of the search space

S = Total no. of bacteria in the population

N_c = The no. of chemotactic steps

N_s = The swimming lengths

N_{re} = No. of Reproduction steps

N_{ed} = No. of elimination-dispersal events

P_{ed} = Elimination dispersal probability

$C(i)$ = The size of the step taken in the random direction specified by the tumble.

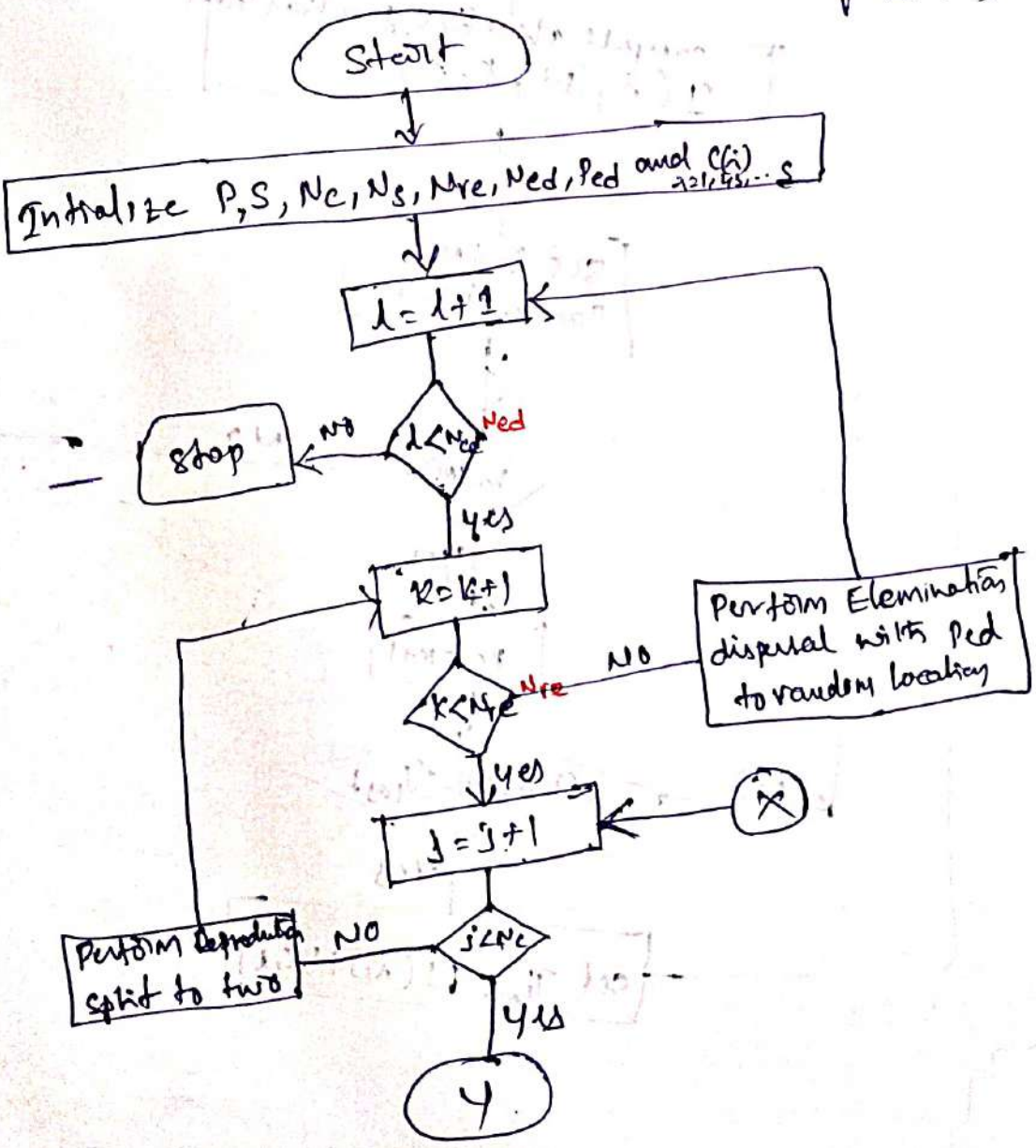
let : $\theta^i(j,k,l)$ = i^{th} bacterium at j^{th} chemotactic k^{th} reproductive and l^{th} elimination dispersion step.

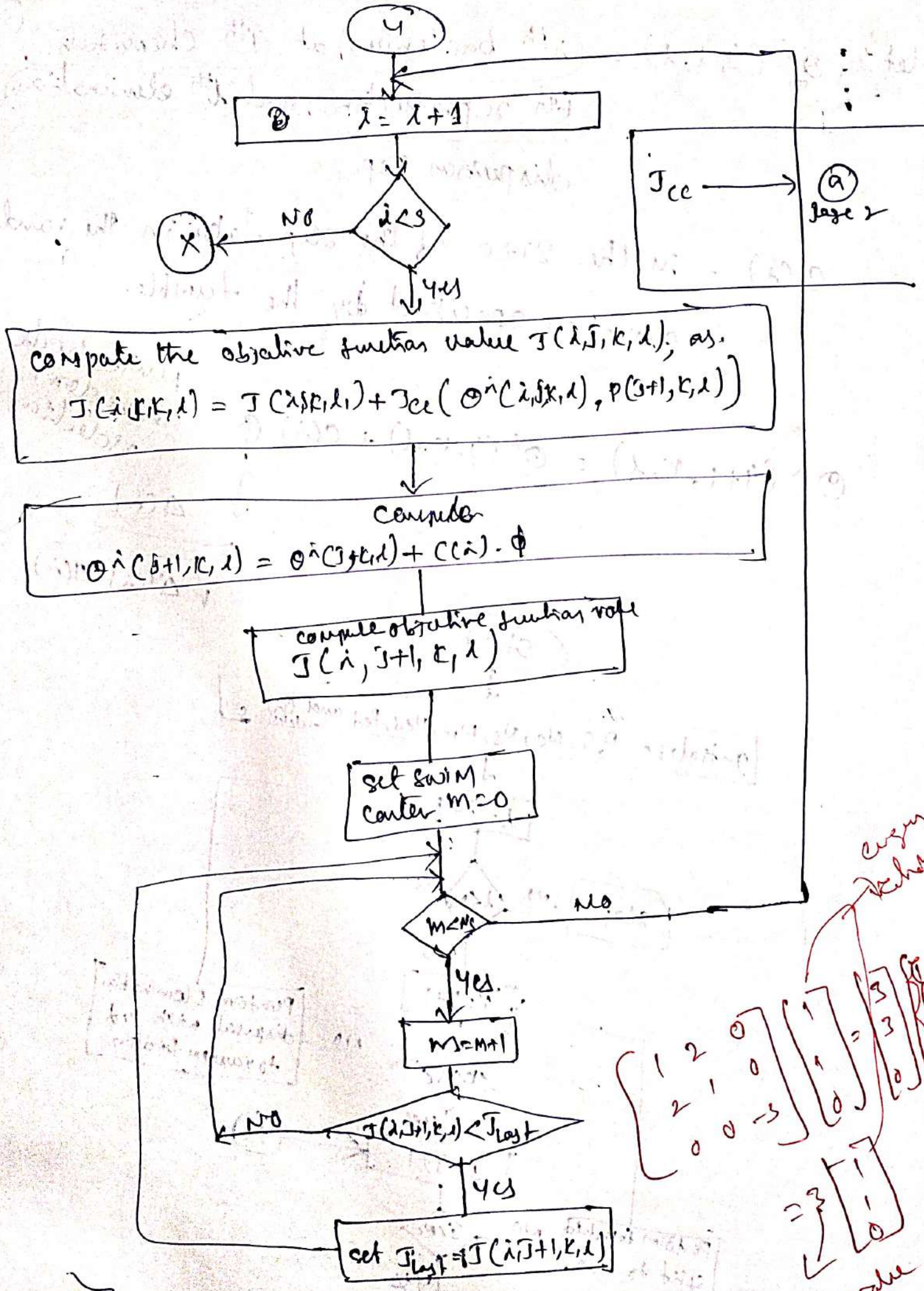
$C(i)$ = is the size of the step taken in the random direction specified by the tumble.

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\,\phi$$

ϕ → Unit walk in random direction

$$\phi = \frac{\Delta(i)}{\sqrt{\Delta^T(i)\cdot\Delta(i)}}$$




4
2 λ = λ + 1
λ < S
NO
X
YES
J_{cc}
9
page 2
compute the objective function value $J(\lambda, j, k, l)$, as.
$J(\lambda, j, k, l) = J(\lambda, k, l) + J_{cc}(\theta\hat{}(\lambda, j, k, l), p(j+1, k, l))$
Compute
$\theta\hat{}(j+1, k, l) = \theta\hat{}(j, k, l) + c(\lambda) \cdot \phi$
compute objective function value
$J(\lambda, j+1, k, l)$
set & with M
counter m=0
m < M
NO
YES
m = m+1
$J(\lambda, j+1, k, l) < J_{last}$
NO
YES
set $J_{last} = J(\lambda, j+1, k, l)$

Q.1 Use k-means clustering.
To divide data into 2 clusters

x_1	1	2	2	3	4	5
x_2	1	1	3	2	3	5

Step 1 :- Choose 2 random cluster centres.

Say $v_1 = (2,1)$ $v_2 = (2,3)$

Data point	Distance from v_1	Distance from v_2	Ans
a_1 (1,1)	1	2.24	
a_2 (2,1)	0	2	
a_3 (2,3)	2	0	
a_4 (3,2)	1.41	1.41	
a_5 (4,3)	2.83	2	
a_6 (5,5)	5	3.61	

(x_1, x_2) (y_1, y_2)

(1,1) (2,1)

$$\sqrt{(1-2)^2 + (1-1)^2} = \sqrt{1+0}$$

$$= 1$$

Step 3 :- cluster 1 of v_1

: $\{a_1, a_2, a_4,\}$

cluster 2 of v_2 : $\{a_3, a_5, a_6\}$

Step 4 :- Recalculate cluster centers.

$$v_1 = \frac{1}{3}\left[(1,1) + (2,1) + (3,2)\right]$$

$$= \frac{1}{3}\left[6, 4\right]$$

$$= (2, 1.33)$$

$$v_2 = \frac{1}{3}\left[a_3 + a_5 + a_6\right)$$

$$= \frac{1}{3}\left[(2,3)(4,3)(5,5)\right]$$

$$= \frac{1}{3}\left[(11, 11)\right]$$

$$= (3.67, 3.67).$$

Step 5 :- Repeat from S2 until we get same cluster centers as same cluster elements.

Distance table :-

Data point	Dis from $V_1(2, 1.33)$	$V_2(3.67, 3.67)$	Assigned cen
a_1	1.05	3.78	V_1
a_2	0.33	3.15	V_1
a_3	1.67	1.8	V_1
a_4	1.204	1.8	V_1
a_5	2.605	0.75	V_2
a_6	4.74	1.88	V_2

cluster 1 of V_1 : $\{a_1, a_2, a_3, a_4\}$

V_2 : $\{a_5, a_6\}$

Recalculating cluster centers

$$V_1 = \frac{1}{4}[a_1 + a_2 + a_3 + a_4]$$

$$= \frac{1}{4}[(1,1) + (2,1) + (2,3) + (3,2)]$$

$$= \frac{1}{4}(8, 7) = (2, 1.75)$$

$$v_2 = \frac{1}{2}[a5 + a6]$$

$$= \frac{1}{2}[(4,3) + (5,5)]$$

$$= \frac{1}{2}(9,8) = 4.5, 4$$

* centroid not same.

Again Distance table.

	(2, 1.75) v_1	(4.5, 4) v_2	Assigned centroid
a1	1.25	4.61	v_1
a2	0.75	3.9	v_1
a3	1.25	2.69	v_1
a4	1.03	2.5	v_1
a5	2.36	1.12	v_2
a6	4.42	1.12	v_2 .

cluster 1 of $v_1 = \{a_1, a_2, a_3, a_4\}$

of $v_2 = \{a_5, a_6\}$

* cluster elements are same as in previous iteration.

cluster 1 : { (1, 1), (2, 1), (2, 3)
(3, 2)}

cluster 2 : { (4, 3), (5, 5)}.

$$\left[m_{11}^{old} - m_{11}^{new}\right]^2 + \left[m_{12}^{old} - M_{12}^{new}\right]^2 + \ldots$$

- * If computed value is not less than the threshold then goto step 2.

* If computed value is less than the threshold then stop the iteration.

Self Organizing Maps :-

→ Supervised training techniques, where there is a target output for each input pattern & the network learns to produce the required output.

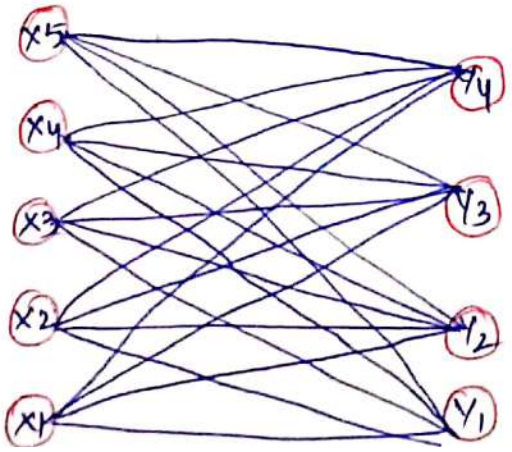
→ Unsupervised training - is where the network learn to form their own classifications of the training data without external help.

→ One particularly interesting class of unsupervised based on competitive learning in which the o/p neurons compete amongst themselves to be activated & this activated neuron is called a winner-takes.

→ Such competation can be implemented by having lateral inhibition connections between the neurons. resulting in neurons organizing themselves. Such a network is called as self Organizing Maps (SOM)

→ The goal of SOM is to transform an incoming signal pattern of arbitrary dimension into 1 or 2 dimensional discrete map, ∴ setting up a SOM.

→ We shall concentrate on a particular kind of SOM known as Kohonen network. This is a feed forward structure with a single computational layer arrange in rows & columns. Each neuro is fully connected to all source nodes in the input layer.



→ The aim is to learn a "feature map" from the spacially "continuous input- space" to a low dimensionality spatially discrete output space. which is formed by arranging the computational neurons into a grid.

Algorithm :-

Step 1 :- Each node weights are initialized

Step 2 :- Every node in the network is Examined to Calculate which one's weight are most like the input vector. The winning node is commonly known as the "Best matching unit"

Step 3 :- Update the weight after finding the winning neuron

Step 4 :- Training will stop if input & weight vectors are identical otherwise goto step - 2

$$w^{new} = w^{old} + \alpha\eta(\text{ip} - \text{weight vector}).$$

Genetic Algorithm :-

→ The Genetic Algorithm was developed by John holland & is a model or abstraction of biological Evolution based on charle's Darwin theory of natural selection.

→ charle's Darwin theory of natural selection Explains survival of fittest overtime, change within species leads to replacement of old species by new species as less successful species becomes Extinct.

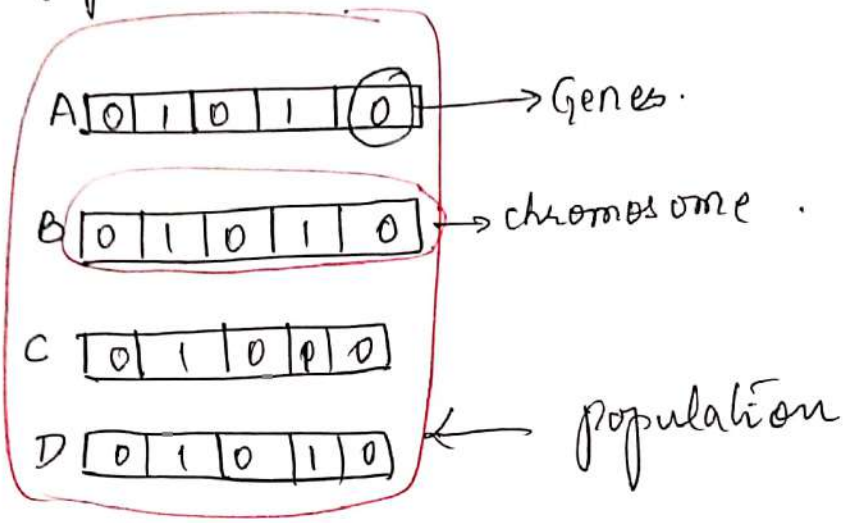
→ Genetic Algorithms are a class of probabilistic optimization algorithms Inspired by biological Evolution process. These algorithms use "natural selection" & "genetic Inheritance".

→ Phases of Genetic Algorithm :-

GE Begins with a large population & gradually reduces it using some heuristics until only the best is left.

⇒ Initial population :-

→ Process starts with a set of individuals known as a population.



note :- Genes → chromosomes are divided into several parts called genes.

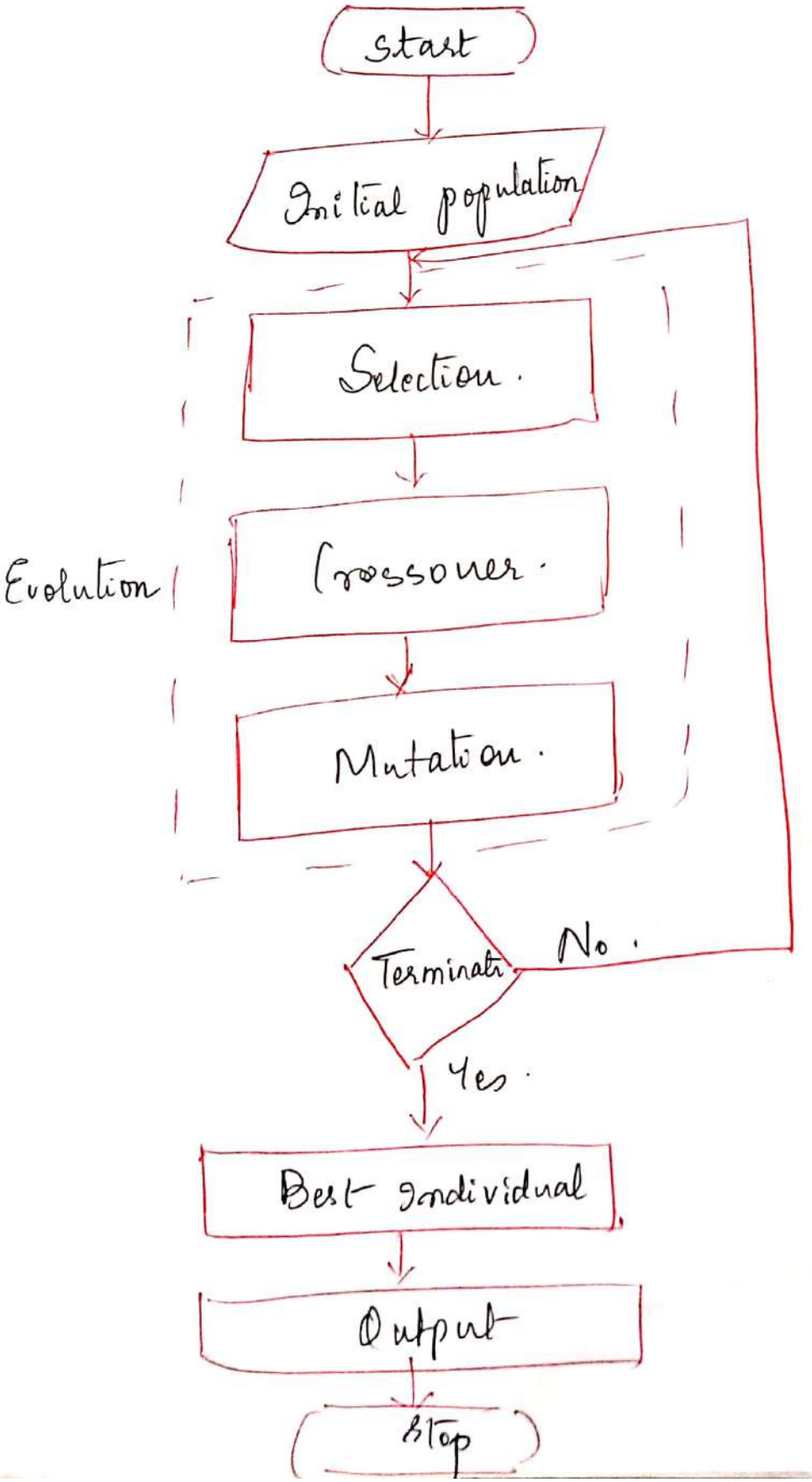
chromosome - All genetic information of a species is stored in chromosomes.

Natural Evolution	Genetic Algorithm
1) Gene	Feature
2) Chromosome	string.
3) Genotype	coded string
4) Phenotype	Decoded structure

Applications of Genetic Algorithm :-

1) Traveling & shipment routing.

2) Robotics.



2) Fitness function :-

→ Determines how fit an individual member of the population is.

→ A fitness score is associated with each individual & higher the fitness score, higher the probability that individual will be selected for reproduction to next generation.

3) Selection :-

→ This phase main goal is to find the region where getting the best solution is more.

4) Crossover :-

→ Also called as recombination, where genetic merging takes place to produce new offspring.

5) Convergence.

→ When there is no improvement in solution quality.

Bacterial Foraging Optimization Algorithm (BFO):-

→ Foraging means searching for food & Exploiting food resources. It affects an animal's fitness because it plays an important role in an animal's ability to survive & reproduce.

→ Natural selection tends to Eliminate those poor "foraging strategies" & favor the reproduction of those animals that have successful foraging strategies

→ Such Evolutionary principles have led scientists in the field of "foraging theory" to hypothesize.

→ Here the algorithm is based a bacteria E.coli. ∴ this optimization is named as Bacterial Foraging optimization.

→ "E.coli" [Escherichia coli] is a type of bacteria that normally lives in our Intestines of healthy people & Animals.

It helps digest the food we eat.

→ During foraging a real bacteria does two basic Operations.

1) Swim
2) Tumble.

Algorithm :-

1) CHEMOTAXIS

→ a) When a bacterium meets a favorable Environment (rich in nutrients)

i) It will continuesly swim in the same direction

tumble → run → run

b) when it meets an unfavorable Environment

i) It will tumble ; change the direction of swim.

tumble → run → tumble.

2) SWARMING

→ E.coli bacterium has a specific sensing actuation & decision making mechanism.

→ On each move it releases attracts to signal

other bacteria towards it.

3) Reproduction :-

→ After calculating fitness value of each bacteria Reproduction allows the bacteria to survive & Reproduce.

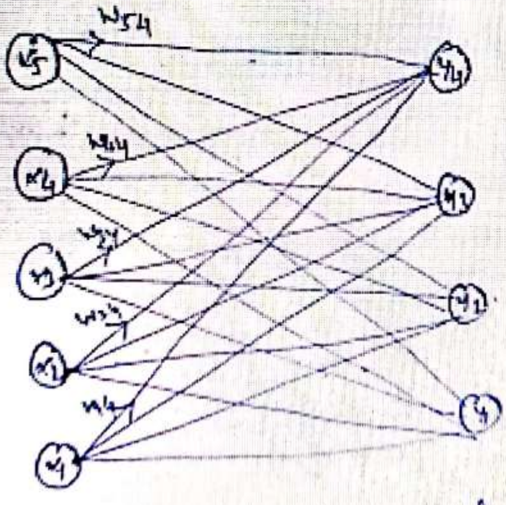
4) Elimination & dispersal

→ Unfit bacterias are Eliminated

Eg: (M,1 −M,1 ...

KSOM (Kohonen self-organizing Maps)

This has a feed-forword structure with a single computational layer of neurons. ~~arranged~~ Each neuron is fully connected to all the source units in the i/p layers.



Input units

cluster unit

step 4: compute the sum of squared difference between the previous membership values and the current membership values.

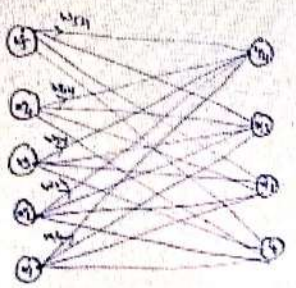
If computed values is not less than the threshold then go to step-1.

If computed value is less than the threshold then stop the iteration.

Eg: $(m_{11}^{old} - m_{11}^{new})^2 + (m_{12}^{old} - m_{12}^{new})^2 + \ldots$

KSOM (Kohonen self-Organizing Maps)

This has a feed-forward structure with a single computational layer of neurons. Each neuron is fully connected to all the source units in the i/p layers.



i/p units cluster unit

The aim is to learn a "feature map" from the spatially "continuous input space", in which our i/p vectors live, to the low dimensional spatially "discrete output space" which is formed by arranging the computational neurons into a grid.

KSOM (Algorithm)

step 1: Each node's weights are initialized (all i/p & o/p are normalized).

step 2: Every node in the network is examined to calculate which ones' weights are most like the i/p vector. The winning node is commonly known as the "Best matching unit".

step 3: Update the weight after find the winner neuron (more importance to the neuron which is closer to winner, less importance to the more neuron which is far from winner)

step 4: Training will stop if i/p & weight vectors are identical. otherwise go to step-2.

$$w^{new} = w^{old} + \alpha(I/p - \text{weight vector})$$

Particle swarm Algorithm :-

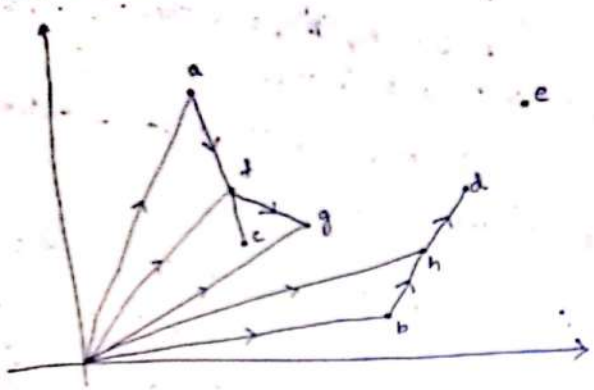


Fig: vector representation of pso Algorithm

1) Let the position of the swarms be at 'a' and 'b'. Both are trying to reach 'e'.

2) Let 'a' decided to move 'c' and 'b' is decided to move 'd'.

3) The distance b/w position 'c' and 'e' is greater than the distance b/w d and 'e'.

4) So, 'd' is a common position decided by the a and b

5) The position 'c' is the individual decision taken by 'a'; The position 'd' is the " " " by 'b'; the position 'd' is common position to move by both

Now. Let us consider three swarms (a, b, c) are trying to reach the particular destination point 'D'.

A decides 'd', B decides b', c decides c' as next position.

pso algorithm

step 1 : Initialize the positions a, b, c, d, e

step 2 : Initialize the next positions decided by the individual swarms as a', b', c', d', e'

step 3 : Global decision regarding the next position is computed as follows.

compute $f(a', b, c, d, e)$, $f(a, b', c, d, e)$, $f(a, b, c', d, e)$ $f(a, b, c, d', e)$ and $f(a, b, c, d, e')$. find the minimum among the computed values.

If $f(a', b, c, d, e)$ is minimum among all, the global position decided regarding the next position is 'a'

If $f(a, b', c, d, e)$ is minimum the global position is 'b' and so on.

Let the selected global position is represented as "global"

step 4: Net value for 'a' is computed as the linear combination of 'a', (a'-a), (global - a). i.e

weight + randomness

next a = a + C1 * RAND * (a'-a) + C2 * RAND * (global - a)

next b = b + C1 * RAND * (b'-b) + C2 * RAND * (global - b)

next e = e + C1 * RAND * (e'-e) + C2 * RAND * (global - e)

step 5 : change the current value for a, b, c, d, e as next a, next b, next c, next d, next e.

step 6 : If f(next a, b, c, d, e) is less than f(a, b, c, d, e) then update the value for 'a' as "next a" otherwise 'a' is not change (do this for b, c', d', e' also)

step 7 : Repeat the steps 3 to 6 for much Iteration to reach the final decision.

The values C1 & C2 are decided based on the weightage given to individual decision and global decision respectively.

Let Δa(t) is the change in the value for updating the value for 'a' in 't'th Iteration. Then next a of (t+1)th Iteration, can be computed using the following formula.

This is considered as the velocity for updating the position of the swarm in every Iteration

next a(t+1) = a(t) + Δa(t+1)

where

Δa(t+1) = C1 * rand * (a'-a) + C2 * rand * (Global + w(t) * Δa(t)

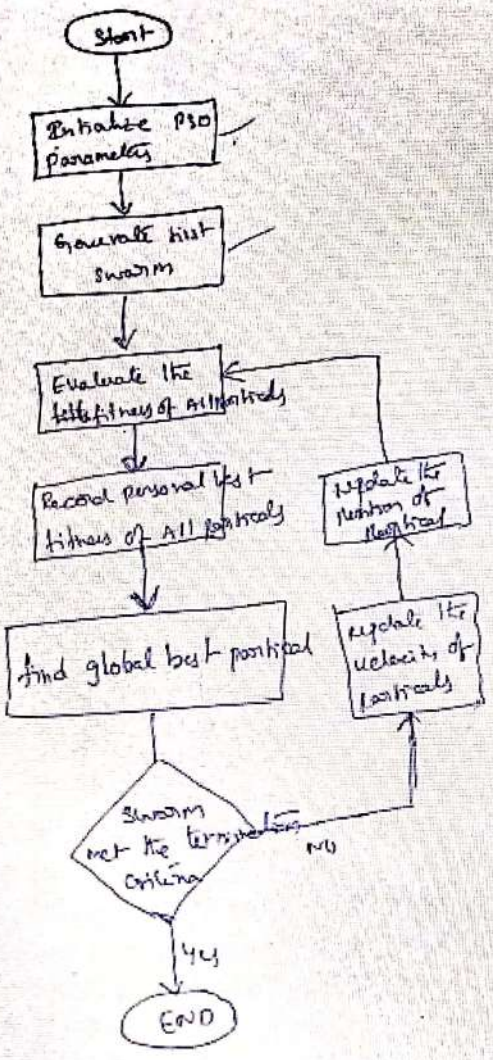
w(t) is the weight at t'th Iteration. The value for 'w' is adjusted at every Iteration as given bellow, where 'Iter' is total no. of Iteration used.

$$w(t+1) = \frac{w(t) - t * w(t)}{Iter}$$

PSO algorithm flow chart

Ant colony optimization

PSO algorithm flow chart



Ant colony optimization

Ant colony optimization techniques exploit the natural behavior of ants, in finding the shortest paths to reach destination from the source

Algorithm

consider the problem of finding the optimum order in which the numbers from 1 to 8 are arranged so that the cost of the order is maximized. cost of the particular particular order is computed using the two ~~matrix~~ matrices A and B as described below

a_1							
a_2	a_3						
a_4	a_5	a_6					
a_7	a_8	a_9	a_{10}				
a_{11}	a_{12}	a_{13}	a_{14}	a_{15}			
a_{16}	a_{17}	a_{18}	a_{19}	a_{20}	a_{21}		
a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}	a_{28}	
a_{29}	a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}

Matrix A

b_1							
b_2	b_3						
b_4	b_5	b_6					
b_7	b_8	b_9	b_{10}				
b_{11}	b_{12}	b_{13}	b_{14}	b_{15}			
b_{16}	b_{17}	b_{18}	b_{19}	b_{20}	b_{21}		
b_{22}	b_{23}	b_{24}	b_{25}	b_{26}	b_{27}	b_{28}	
b_{29}	b_{30}	b_{31}	b_{32}	b_{33}	b_{34}	b_{35}	b_{36}

Matrix B

The cost of the order [3 2 1 5 7 2 4 6 8] is computed as the sum of the products of values obtained from the positions (3,1) of matrix A & B (3,1), (2,2), (1,3), (5,4), (7,5), (2,2), (4,7), (6,4), (8,6) of matrix A & B

$(a_6 \times b_{14}) + (a_3 \times b_{21}) + (a_5 \times b_{36}) + (a_{14} \times b_{14}) + (a_{21} \times b_{16}) + (a_{15} \times b_{15}) + (a_{20} \times b_{11}) + (a_{36} \times b_{16})$

...evaluated using Ant colony optimization technique as described below.

Step 1: Generate the five sets of orders randomly which are treated as the initial values selected by the 5 ants respectively.

The following are the orders selected by the 5 ants along with the corresponding cost as given below

ANT	ORDERS	COST
ANT 1	4 1 3 4 6 2 5 8	C1
ANT 2	4 2 8 5 7 1 3 6	C2
ANT 3	5 1 3 2 4 7 6 8	C3
ANT 4	2 7 4 1 8 6 3 2	C4
ANT 5	8 1 2 4 3 5 7 6	C5

Step 2: Pheromone is the value assigned for ith number in the jth position of the sequence selected by all the ants. Initially all the values of matrix are completely filled up with ones. The values of pheromone matrix are updated in every iteration.

The pheromone matrix(n+1) = pheromone matrix(n) + pheromone matrix(n) updating

* To select the value for row, jth column of

	1	2	3	4	5	6	7	8
1	0	$\frac{1}{C_1}+\frac{1}{C_2}+\frac{1}{C_3}+\frac{1}{C_4}$	0	$\frac{1}{C_4}$	0	$\frac{1}{C_2}$	0	0
2	0	$\frac{1}{C_1}$	$\frac{1}{C_1}$	$\frac{1}{C_1}$	0	$\frac{1}{C_4}$	0	$\frac{1}{C_4}$
3	$\frac{1}{C_4}$	0	$\frac{1}{C_1}+\frac{1}{C_3}$	0	$\frac{1}{C_5}$	0	$\frac{1}{C_2}$	0
4	$\frac{1}{C_3}$	0	$\frac{1}{C_4}$	$\frac{1}{C_5}+\frac{1}{C_1}$	$\frac{1}{C_3}$	0	0	0
5	$\frac{1}{C_5}$	0	0	$\frac{1}{C_1}$	0	$\frac{1}{C_8}$	$\frac{1}{C_4}+\frac{1}{C_5}$	0
6	0	0	0	0	$\frac{1}{C_1}$	$\frac{1}{C_4}$	$\frac{1}{C_3}$	$\frac{1}{C_1}+\frac{1}{C_2}$
7	$\frac{1}{C_1}$	$\frac{1}{C_4}$	0	0	$\frac{1}{C_2}$	$\frac{1}{C_3}$	$\frac{1}{C_5}$	0
8	$\frac{1}{C_5}$	0	$\frac{1}{C_2}$	0	$\frac{1}{C_4}$	0	0	$\frac{1}{C_2}+\frac{1}{C_3}$

Step 3: Next set of orders selected by the ants will be as described below

eg: The random position is 8 2 3 6 4 1 5 (say)

Let the order decided by the ant 1 be represented as [x1, x2, x3, x4, x5, x6, (x7) x8]

* The value for the x7 in the order as mentioned above is selected such as the first number of the random generated position sequence is 7.

* To select the value for v17, 7th column of the pheromone matrix is used as bellow,

column 7 = $[0 \;\; 0 \;\; \frac{1}{c_2} \;\; 0 \;\; \frac{1}{c_4}+\frac{1}{c_4} \;\; \frac{1}{c_3} \;\; \frac{1}{c_5} \;\; 0]$

* now calculate the normalized value of column 7 in the order

This is also called as the probability vector and for selecting the value of v17

$S = [\frac{1}{c_2}+\frac{1}{c_1}+\frac{1}{c_4}+\frac{1}{c_3}+\frac{1}{c_5}]$ = value

Normalized vector 7 = $\frac{\text{column 7}}{S} = [\frac{0}{S} \;\; \frac{0}{S} \;\; \frac{1/c_2}{S} \;\; \frac{0}{S} \;\; \frac{c_4+\frac{1}{c_4}}{S}$

$= [p_1 \; p_2 \; p_3 \; p_4 \; p_5 \; p_6 \; p_7 \; p_8] \;\; \frac{1}{c_2} \;\; \frac{1/c_5}{S} \;\; \frac{0}{S}]$

* find the lowest value position, for Example p_5 is the lowest value the 5 is assigned to v17.

Note: position no. '5' is not used again.

* The next value in position sequence is 8, so colum '8' is selected from pheromone matrix, and select the normalized value

column 8 = $[0 \;\; 1/c_4 \;\; 0 \;\; 0 \;\; \frac{1}{c_2}+\frac{1}{c_1} \;\; 0 \;\; \frac{1}{c_1}+\frac{1}{c_2}]$

$\therefore 8 = [\frac{1}{c_4}+(\frac{1}{c_2}+\frac{1}{c_1})+(\frac{1}{c_4}+\frac{1}{c_3})]$

Normalized vector 8 = $\frac{\text{column 8}}{S} = (a_1, a_2, a_3, a_4, a_5, a_6, a_8)$

find the lowest value in normalized vector '8' and replaced with v8. Example if a_3 is lowest then number '3' is assigned to v8.

* This process is repeated to obtain the order selected by the Ant 2, Ant 3, Ant 4, Ant 5.

* This is called one Iteration.

step 4: updating the pheromone matrix for the second iteration is computed as described in step 2. This is called updating pheromone matrix (2)

∴ pheromone matrix(2) = pheromone matrix (1) + updating pheromone matrix

step 5: Next set the orders selected by the five ants are computed as described in the step 3

step 6: step 3 to step 5 is repeated for the particular Iterations. Thus best set of orders are obtained using Ant colony algorithm

Note: Best among the best set of orders selected by the five ants in the last iteration is selected as the global best order.