WJMK ☺ ♡M.

WJMK ☺ ♡M.

30/09/2024
Monday

Lec- :-

ANN :- → artificial Neural n/w.

Binary class → {0, 1} → $\frac{1}{1+e^{-x}}$ → sigmoid

for multiclass
↳ we/we to modify sigmoid, can't use as it is.

How can change classification to regression model?

→ why classification can not be applied bt regression ??

we want to normalize it, so that "scaling factor" can remain same!

⇓

ONE-HOT-CODING

⇓

say 10-class
↳ {1, 2, 3, . . , 10}

↓

(1, 0, 0, . . 0)

(0, 1, 0, . . , 0)

So y_i = (0, 0, . . , 1, 0, 0, . . 0)

↑ i^{th} position.

WORK ON:

⇒ if i^{th} position = 1 → then belongs to i^{th} class.

*. Sigmoid in this case known as "SOFT MAX"

$$z = (\underset{\text{4 in class-1}}{4}, \underset{\text{6 in class-2}}{6}, 7, 8, \ldots)$$

In Soft max, for each position, compute:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{\forall i} e^{z_i}}$$

$$e(z_1) = \frac{e^{4}}{e^{4}+e^{6}+e^{7}+e^{8}+\ldots}$$

formula not changing, representation we are changing, so that same formula can be used.

99% Data in 1 class → Data Imbalance

↳ then model with accuracy = 99%

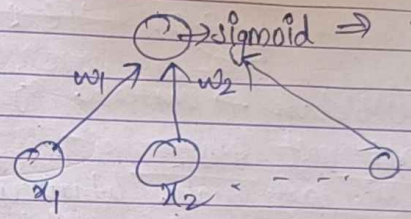
↳ saare isi class ke lelo

↓

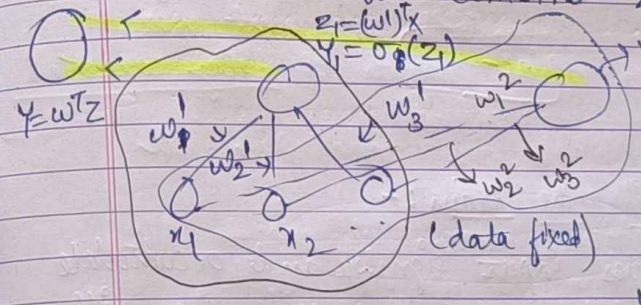
but glt hai model

WORK ON:

If data biased then model will always be biased chahe kuchh bhi karlo.



ANN → also known as multi-perceptron, as we combine so many.



↳ we go to z_2 ... can go till z_{100} etc.

↓

single hidden layer and 2 nodes in it

↳ can ↑ hidden layers as well nodes in each hidden layer.

→ what change in ANN classification to that ANN regression ban jaaye?

↳ sigmoid hatado!

(saare batayein ?, yaa sirf last wala?
↳ Done kr skte hain,
as then $y \in \mathbb{R}$.

If multiclass → instead of sigmoid, put soft max.

→ better ki last wala batayein! → as andar say w, jyada, then uska weightage jyada ho jaayega, so andar sigmoid dekho!

7/10/24
Monday

lec-

* Neurons in the brain :-
↳ receives input signals and accumulate voltage. After some threshhold, they will fire spiking responses.

* A simpler Neuron :- for neural nets, we us simpler model for neuron or unit :-

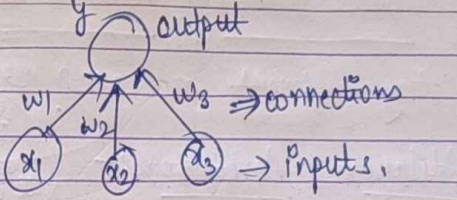
bias

$$y = \phi(w^T x + b)$$

output · activation function · weights · inputs

similar to logistic regression ⇒

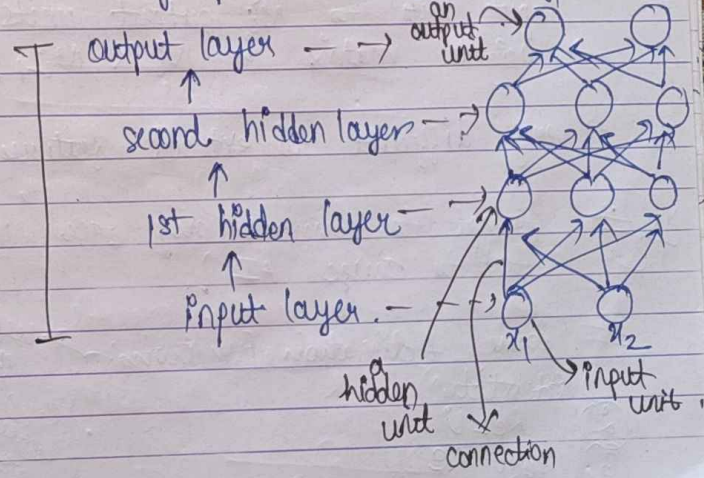
$$y = \sigma(w^T x + b)$$



⇒ By throwing together lot of these simple neuron-like processing units, we can do some powerful computations!

→ sigmoid function is non-linear.

* A Feed-forward Neural N/w
→ a DAG (directed acyclic graph)
→ Units are grouped into layers.



layers kitni chahi hain → similarly no. of nodes in each layer
↳ I don't know → Hyperparameter, baaki sab → have to learn it through algo.

* layers ↑ hain and no. of nodes same, kya difference in model?
one model with 'n' nodes and less layers,
2nd ——— (n) ——— more ↓

H/w Are they different or same? → one model with 3 nodes in each layer (2 layers), one model with all 6 nodes in 1 layer

Multilayer Perceptrons:-

→ outputs are function of input units :-
$$y = f(x) = \phi_w(Wx + b)$$
(ϕ_w: applied component wise)

→ multilayer N/w consists of fully connected layers.
↳ all inputs (units) are connected to all output units.
→ Each hidden layer i connects N_{i-1} input units to N_i output units.
→ weight matrix is $N_i \times N_{i-1}$.

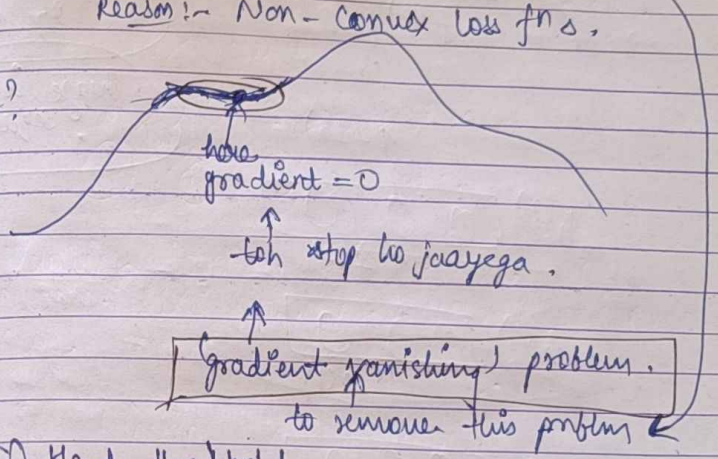
→ very difficult to extract features.

H/w # Prompt to generate my face, using "text to image" AI?

Some Activation Functions:- → H/w, why so many fns? ↳ why not only sigmoid?
① Identity
② ReLU
③ soft ReLU
④ sigmoid

→ as each layer can have different no. of nodes,
w ka kya size ho.

→ when sigmoid is powerful, then why so many?
↓ Reason:- Non-Convex loss fns.
→ difference?

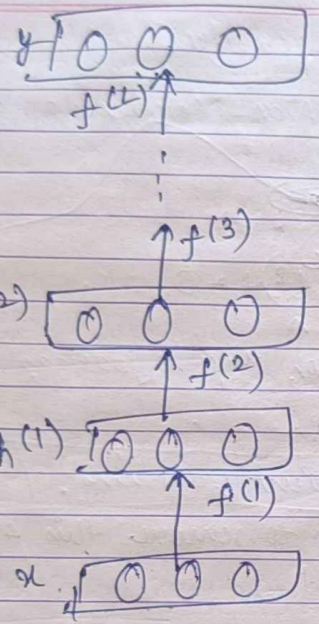


① Hard threshhold
② logistic
③ ——— ?

Computation in Each Layer:-

Each layer computes a fn.

$$h^{(1)} = f^{(1)}(x) = \phi(w^{(1)}x + b^{(1)})$$
$$h^{(2)} = f^{(2)}(h^{(1)}) = \phi(w^{(2)}h^{(1)} + b^{(2)})$$
$$y = f^{(L)}(h^{(L-1)})$$
If regression ⇒ $y = (w^{(L)})^T(h^{(L-1)}) + b^{(L)}$
If classification ⇒ $y = \sigma((w^{(L)})^T(h^{(L-1)}) + b^{(L)})$
(sigmoid (if binary), soft max (if multiclass))



audio is sequence of image

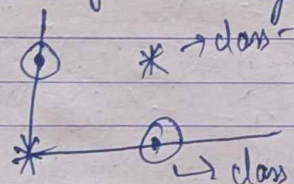
image is not sequence

Q. # Truth table for XOR :-

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Now classify this! ① using linear regression ② using ANN

→ will they be linearly separable?



→ not linearly separable, so go to higher domain / dimension → where they are linearly separable.

feature learning

Note :-

→ using ANN, can learn any fn.

sigmoid / activation fn is making it so much powerful.

Expressive power of linear N/ws.

Designing a N/W to classify XOR

N/W → one or two hidden layer, 2-3 nodes in that, hard threshold activation fn / w/o activation fn.

agar koi saying weight define kiya hoga?

① manually ② code

do 20, 50, 100 iterations to get accuracy

Back propagation of error

9/10/24

Back Propagation :-

getting 'ψ' in feature learning is a difficult task.

WTMK

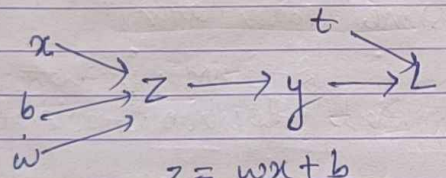
Expressivity of logistic Activation function :-

→ at last layer → we use only sigmoid or soft max f^n as activation f^n, don't use ReLU etc.
(They can be used in intermediate layers only,
↳ WHY?)

gradient descent → $w_{i+1} = w_i - \alpha\left(\frac{\partial L}{\partial w_i}\right)$

α → Hyperparameter (we've to fix)
usually km small rkhi lete hain

Logistic least squares :-



$z = wx + b$
$y = \sigma(z)$
$L = \frac{1}{2}(y-t)^2$

$\frac{dz}{dx} = \left(\frac{dz}{dy}\right)\left(\frac{dy}{dx}\right)$ [Univariate chain Rule]

as: $L = \frac{1}{2}(\sigma(wx+b) - t)^2$

$\frac{\partial L}{\partial w} = \frac{\partial}{\partial w}\left(\frac{1}{2}(\sigma(wx+b) - t)^2\right)$

WTMK

gradient for b

$= (\sigma(wx+b) - t)\,\sigma'(wx+b)\,x$

$\frac{\partial L}{\partial b} = (\sigma(wx+b) - t)\,\sigma'(wx+b)$

→ Disadvantages of this approach?

Now gradient for 'w' ⇒

$\frac{\partial L}{\partial w} = \left(\frac{\partial L}{\partial y}\right)\left(\frac{\partial y}{\partial w}\right)$

$= \left(\frac{\partial L}{\partial y}\right)\left(\frac{\partial y}{\partial z}\right)\left(\frac{\partial z}{\partial w}\right)$

$= (y-t)\,\sigma'(z)\,(x)$

$= (\sigma(wx+b) - t)\,\sigma'(wx+b)\,x$

Computing loss →

$z = wx + b$
$y = \sigma(z)$
$L = \frac{1}{2}(y-t)^2$

gradient for b :-

$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial b}$

$= (y-t)\,\sigma'(z)\,(1)$

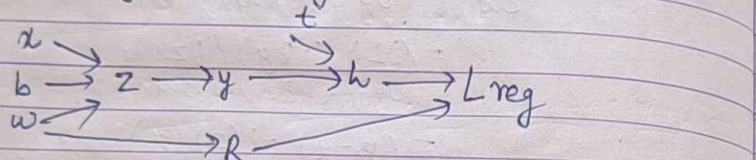
→ If layer ↑s → z, z', z'' ... so on, derivative will also change accordingly. (abhi jitni computation ki, that is only for one hidden layer)

Computing derivation :- $\bar{y} = (y - t) = \frac{\partial L}{\partial y}$

$\bar{z} = \bar{y}(\sigma'(z)) = \frac{\partial L}{\partial z} = \left(\frac{\partial L}{\partial y}\right)\left(\frac{\partial y}{\partial z}\right)$

$\left(\frac{\partial L}{\partial y}\right)\left(\frac{\partial y}{\partial z}\right)\left(\frac{\partial z}{\partial w}\right) = \frac{\partial L}{\partial w} = \bar{w} = \bar{z}x$, $\bar{b} = \bar{z}$

L_2 – Regularized Regression:-



$R = \frac{1}{2}w^2$

$\boxed{L_{reg} = L + \partial R}$

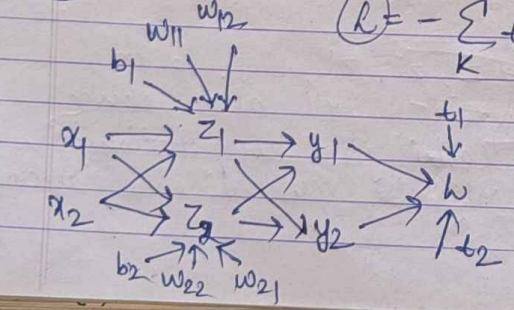
Softmax Regression:-

$z_e = \sum_j (w_{ej} x_j + b_e)$

$y_k = \frac{e^{z_k}}{\sum_e e^{z_e}}$

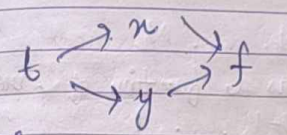
Cross-Entropy loss

$L = -\sum_k t_k \log(y_k)$



Multi-Variate chain Rule:-

$\frac{d}{dt}(f(x(t), y(t))) = \frac{\partial f}{\partial x}\left(\frac{dx}{dt}\right) + \frac{\partial f}{\partial y}\left(\frac{dy}{dt}\right)$



① eg:- $f(x, y) = y + e^{xy}$, $x(t) = \cos(t)$, $y(t) = t^2$

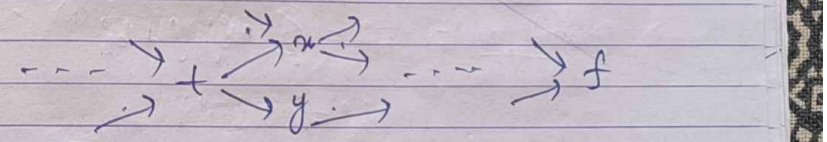
$\frac{d}{dt}(f(x,y)) = \frac{\partial f}{\partial x}\left(\frac{dx}{dt}\right) + \frac{\partial f}{\partial y}\left(\frac{dy}{dt}\right)$

$= (y e^{xy})(-\sin t) + (1 + x e^{xy})(2t)$

$\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$ → values already computed by our program.

$\frac{dx}{dt}$, $\frac{dy}{dt}$ → expressions to be evaluated.

Notation :→ $\bar{t} = \bar{x}\frac{dx}{dt} + \bar{y}\frac{dy}{dt}$



black box → as hidden layer k har node ka kitna-2 contribution in final output?

kitne sleeping nodes? → jinka no contribution

→ jo node active thi wahi hai!
→ kya unko kisi aur kaam pe lga skte hain, w/o affecting others.
↳ VERY DIFFICULT (:|)

Backpropagation Algorithm :-

① forward pass → for $i = 1, \ldots N$
compute v_i as a function of parents (v_i)
② backward pass → for $i = N-1, \ldots, 1$

Backpropagation for Regularized Logistic Least Square :-

Forward pass :- $z = wx + b$, $y = \sigma(z)$

$$L = \frac{1}{2}(y-t)^2, \quad R = \frac{1}{2}w^2$$

$$L_{reg} = L + \partial R$$

backward pass :- $\bar{L}_{reg} = 1$

$$\bar{R} = \bar{L}_{reg}\left(\frac{\partial L_{reg}}{\partial R}\right) = (\bar{L}_{reg})(\partial)$$

Backpropagation for Two-Layer Neural N/w :-

Forward pass :- $z_i = \sum_j w_{ij}^{(1)} x_j + b_i^{(1)}$

$$h_i = \sigma(z_i)$$

$$L = \frac{1}{2}\sum_K (y_K - t_K)^2$$

$$y_K = \sum_i w_{Ki}^{(2)} h_i + b_K^{(2)}$$

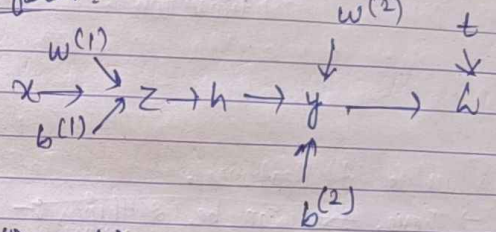
Backward pass :- $\bar{L} = 1$, $\bar{y}_K = \bar{L}(y_K - t_K)$

$$\bar{w}_{Ki}^{(2)} = \bar{y}_K h_i$$

$$b_K^{(2)} = \bar{y}_K, \quad \bar{h}_i = \sum_K (\bar{y}_K)(w_{Ki}^{(2)})$$

$$\bar{z}_i = \bar{h}_i (\sigma'(z_i))$$

① In vectorised form :-



Forward pass :-

$$z = w^{(1)}x + b^{(1)}, \quad h = \sigma(z), \quad y = w^{(2)}h + b^{(2)}$$

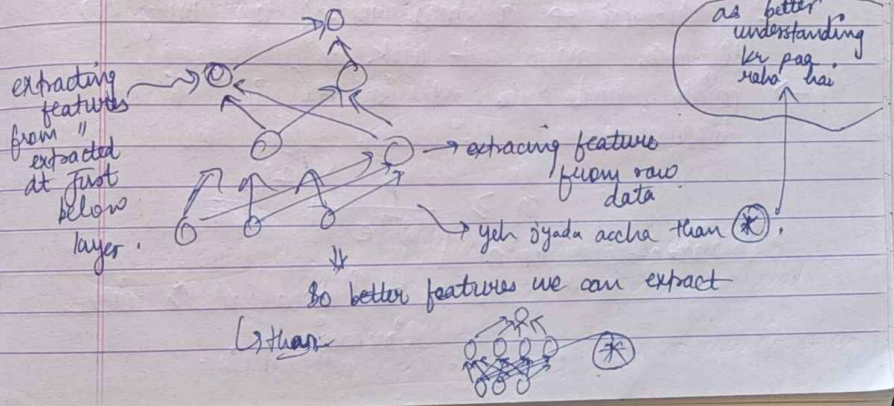
$$L = \frac{1}{2}\|t - y\|^2$$

Backward pass :-

$$\bar{L} = 1, \quad \bar{y} = \bar{L}(y - t), \quad \bar{w}^{(2)} = \bar{y}\,h^T,$$

$$\bar{b}^{(2)} = \bar{y}, \quad \bar{h} = (w^{(2)})^T \bar{y}, \quad \bar{z} = \bar{h} \circ \sigma'(z),$$

$$\bar{w}^{(1)} = \bar{z}\,x^T, \quad \bar{b}^{(1)} = \bar{z}$$



14 oct 2024
Monday

Lec :-

$\{x_i, y_i\}_{i=1}^{n}$ → n → total no. of training data

$x_i \in \mathbb{R}^d$

d → attribute / quality / feature
d → dimension

↓ may be independent / dependent or anything.

→ we've 'd' features.
→ There can be 2 things :-
(1) 'n' very large, or,
(2) 'd' very large.

Curse of dimensionality
→ As you ↑ dimension, points tend to stay in outer region / periphery.
→ 'Distance' → as measure of similarity / difference may not work any longer!
→ points, tends to get isolated in higher dimensions.

1st curse → sparse data ↑ dimension increase

2nd curse → hardly any ↓ in difference similarity

$$n \geq \left(\frac{\rho}{2\pi e}\right)^{\rho/2} (\sqrt{\rho\pi})$$

n → no. of observation

↑ how 'n' ↑s with dimension 'p'.

Fluctuations Cumulate :-

$$E[\|f(X_1) - f(\theta_1)\|^2] \leq E[|\epsilon_1|^2] = \sigma^2$$

where,
σ^2 → variance of noise

⇒ In 'p' dimension, this error becomes $(p\sigma^2)$.
⇒ This error can be very large in high-dimensional settings, even if σ^2 is small.
→ every data → point is a vector (from origin)
↳ as sirf tip bnaate hain vector ki, issiliye 'point' bola.

→ In higher dimensions, numerical computations can become very intensive.

Circumventing the curse of dimensionality :-
Assumptions on which techniques are based :-
① Need to go to lower dimensions.
→ features are not INDEPENDENT.
find synthetic features which are independent, low in number.
② → a) points lying in high dimension are actually embedded in low dimension.
b) → points locally lie in Euclidean dimension space. (low dimensional Euclidean space)
like vese earth ka 'gol' hai par locally 'flat' hai.

known as Riemannian Manifold Assumption

WORK is worship.

① An unsupervised learning task :-
Dimensionality Reduction

$A \in \mathbb{R}^M$, $M < D$

↑ fo

$x \in \mathbb{R}^D$

Useful for :-
1) Data compression (lossy)
2) Dataset visualization (2D or 3D)
3) discovering most important features.

① PCA (Principal components Analysis) :- (Statistical interpretation)
data → want to project to lower dimension (uncorrelated features)
↳ statistical independent
↳ orthogonal features

Assumption of PCA :-

(1) features in high 'd' are independent, so
(a) can project to low dimensions.
(b) features are independent
(c) features (m) = linear combination of features in higher dimension

* linear regression.
↳ linear combination of non-linear vector.
↳ linear in y and parameters.

WORK is worship.

$y = a_0 x + a_1 x^2 + a_2 x^3 + \dots$

→ not linear in y and x.

$a_0, a_1, a_2 \dots$ are parameters

like, every signal is linear combination of sinusoidal fns.
↳ non-linear.

(*) PCA = basis set (→ orthogonal) of features.

→ PCA projects data in least square sense.
↳ it captures big (principal) variability in the data and ignores small variability.
→ PCA is way to reduce data dimensionality.
→ PCA projects high dimensional data to a lower dimension.

⇒ say x_i, $i = 1, 2, \dots N$ data points in 'p' dimensions ('p' is large)

$x_0 = m = \frac{1}{N} \sum_{i=1}^{N} x_i$

deviation about mean is minimum thus mean like to represent

but quite uninformation.

⇒ so represent data by straight line :-

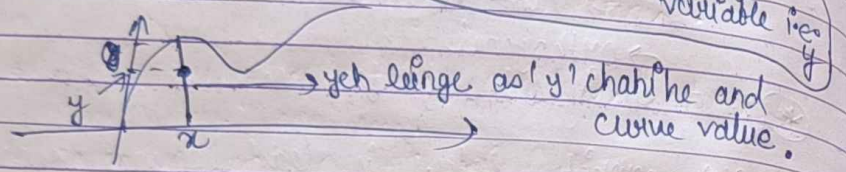
$x = m + ae$

e → unit vector along straight line
a → signed distance of x from m

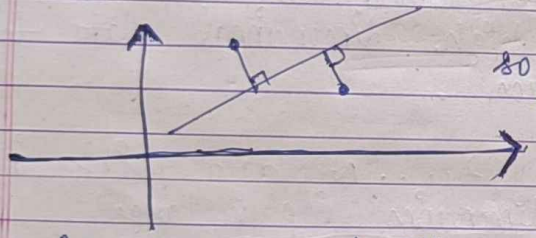
Thus, training points projected on straight line ⇒

$x_i = m + a_i e \quad , i = 1, 2, \ldots, N$

In linear regression :- (parallel to dependent variable i.e. y)



but in PCA → perpendicular lenge !.



→ points have Normal distribution

↳ Independent ;

↳ and sum them , total variance $= n\sigma^2$

Standard deviation $= (\sqrt{n})(\sigma)$

so any like → σ * by $\sqrt{n}$. (???)

↓ 'factor'.

Now ⇒ $J_1(a_1, \ldots, a_N, e) = \sum_{i=1}^{N} \| m + a_i e - x_i \|^2$ (2)

↓ so parabolic (convex)

$J_1(a_1, \ldots, a_N, e) = \sum_{i=1}^{N} a_i^2 \underbrace{\|e\|^2}_{\text{unit vector, thus } = 1} - 2\sum_{i=1}^{N} a_i e^T (x_i - m) + \sum_{i=1}^{N} (x_i - m)^2$

* differentiate wrt a_i and equating to zero

⇒

$a_i = e^T (x_i - m)$

thus, plugging these value ⇒

$J_1(e) = -e^T S e + \sum_{i=1}^{N} (x_i - m)^2$

where,

$S = \sum_{i=1}^{N} (x_i - m)(x_i - m)^T$

(scatter matrix)

→ minimizing $J_1 \equiv$ maximize $e^T S e$.

constraint ⇒ $e^T e = 1$ (as e is unit vector)

By Lagrange multiplier method ⇒

(say) $J_1 = e^T S e + \lambda (e^T e - 1)$

↑ objective fn

differentiate wrt 'e' and equate to '0'

$2Se + 2\lambda e = 0$

$Se = \lambda e$

so, e is eigenvector of S corresponding to largest eigenvalue.

so reduces to Eigen value problem

WWMK

*. Eigen value problem :-

U have vector $\vec{x}$, multiply by a, s.t. it got either enhanced or diminished w/o changing direction.

so,

Dimensionality reduction problem reduces to Eigen value problem.

→ 1st eigen vector is 1st principal component.

→ 2nd ——— 2nd ———.

WWMK

lec- :-

(*) Dimensionality Reduction

↳ curse of Dimensionality

↳ more data (features) do not translate into better understanding

↳ Linear Dimensionality Reduction (PCA).

↓

→ features not independent.

↓

smaller no. of independent features (dim - k)

→ linear features in d-dimensions gives independent features in m-dimensions ($m < d$).

PCA → going to lower dimensions

↳ Eigen value Problem

↓

WWMK

↓ covariance matrix

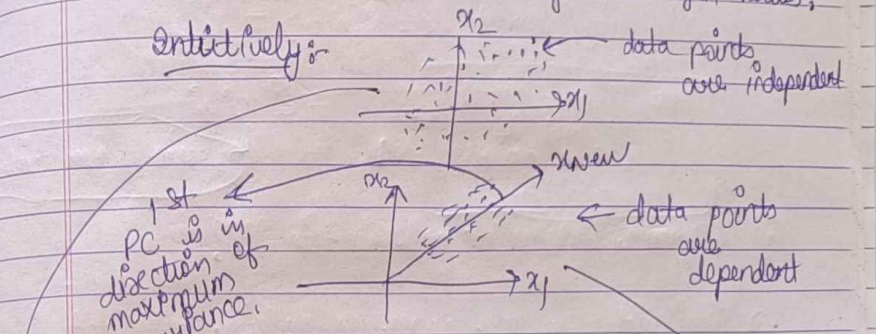
XX^T

↳ Centred → Eigen value dec (SVD).

→ sort eigen values in descending order.

↳ Take largest 'k' eigen values.

Intuitively :-



instead of x_1, x_2,

take x_2 & x_{new},

↳ as then can work in 1 dimension only !!

PC → principal component.

covariance matrix = $\begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$

covariance matrix = $\begin{bmatrix} \sigma_1^2 & cov(x_1,x_2) \\ cov(x_2,x_1) & \sigma_2^2 \end{bmatrix}$ (High, High)

and $cov(x_1,x_2) = cov(x_2,x_1)$

WJMK ☺ 30 mm.

→ say: matrix given telling similarity b/w different weights.

say ⇒ position : x, y
distance : $d(x,y)$ ← (Lp2 Norm)
similarity (kernel) $K = \langle x, y \rangle = x^T y$

↑ Here (linear kernel), but it can be any kernel

$= \Sigma x_i y_i$ ← cosine similarity

⊙ difference b/w vector and function ????

⇒ Similarity in any space is nothing but (dot product).

Pythagoras Theorem :-
↳ ulta Reverse
↳ given distance, how to find best points in k-dimension (2, 3, ... etc)
↳ so can be used for dimension reduction.

PCA → x ko 'k' mein convert karta hai

Position x → Distance d → similarity K → Position x

WJMK ☺ 30 mm.

XX^T ... X

Position - Similarity :-

$K = XX^T$

→ what is Distance Matrix in Neural Networks?

similarity - Distance :-

$$d(x_i, x_j)^2 = \langle x_i - x_j, x_i - x_j \rangle$$
$$= \langle x_i, x_i \rangle + \langle x_j, x_j \rangle - 2\langle x_i x_j \rangle$$
$$= k_{ii} + k_{jj} - 2k_{ij}$$
$$= D_{S,ij}$$

we can determine D_S from K.

(Kernel SVM mein pdha tha).

why so special?
↳ as all decisions are made on basis of (dot product).

$w \cdot x(t) < 0$ → then line k neeche, o/w if > 0, toh upar, if $=$ then on line.
↳ (inclination)

⇒ thus classification done on basis of dot product.
↳ with kernel-trick ⇒ w/o going to higher dimension, hmein milta of $\langle w, x \rangle$ (dot product)

That's gives us (kernel-version)

↳ works on 3 assumption :-
(i) all decision requires $\langle w, x \rangle$
(ii) $w \leftarrow f^n$ of x_i's.
(iii) we can calculate $\langle w, x \rangle$ in high-d, w/o going to high-d.

Distance to Position – MDS (multi dimensional scaling) :-

$M = (\text{distance}(x_i, x_j))^2$

$= \|x_i - x_j\|^2$

$= \|x_i\|^2 + \|x_j\|^2 - 2\langle x_i^T, x_j \rangle$

$\begin{bmatrix} \|x_1\| & \cdots & \|x_n\| \\ \vdots & & \vdots \\ \|x_n\| & \cdots & \|x_n\| \end{bmatrix}$

want to get rid of this,

$JMJ = -2X^TX = -2B$

↳ (Double-centering)

The manifold hypothesis :-

Assumption :-

→ Natural data in high dimensional spaces concentrates close to lower dimensional manifolds.

→ probability density decreases very rapidly when moving away from the supporting manifold.

⇒ probability distribution is nothing but a histogram.

stand with prob = 1/2, what he will do?

↳ Toss a coin,

if head → stand (say), → agar 1 baar bola

o/w sit.

→ agar 100 baar bola,

toh 50 baar sit, and other 50 times stand.

→ manifold assumption holds for smaller distances.

↳ thus flat

euclidean

↳ like earth flat for smaller distances.

4/11 Monday

loc :-

GMM E-M → Expectation – Maximization algorithm,

→ neither of model we learned till like SVM, KNN etc. is a generative model, as we are not learning distributions in them,

→ from pdf (any)

↳ ∞ data can be generated!

Thus, Hallucination is one of major challenge till Now in gen AI.

Product rule :-

$P(x, y) = P(x) \cdot P(y|x)$

$= P(y) \cdot P(x|y)$ → which to choose?

depends on availability of data set.

→ Thus direction matters!

Work ☺

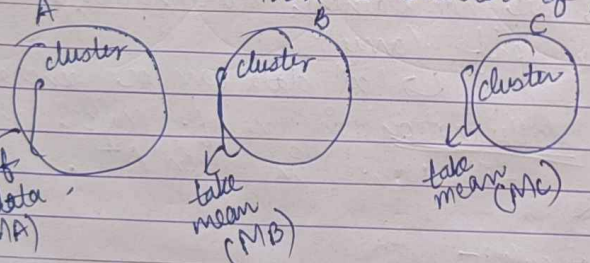
Naive Bayes Model → (extension of Bayes Th^m)

If directed, Bayesian belief N/w.
↳ dependency graph.
A depends on B or B depends on A ... etc.

K-Means
↳ unsupervised (labels not available)
↳ no. of clusters = k.

2-fold
1. ① compute mean ② assign nodes to clusters

→ If k=3 → choose any random 3 pts. (A, B, C) from data set
then har pt. ka inn teeno se distance lo, unka minimum lo, teeno ka.
↳ uss cluster ko that pt. will ∈.
say if pt. D closest to A, then D ∈ cluster of A.



Now take each pt. → take distance from MA, MB, MC, jo minimum, uss cluster mein daalte jaao ... fir se mean lo ...
Jab tak mean change hota jaaye !!

* Drawback of K-mean :-
① complexity of this algo ????

Work ☺

Hidden Markov model

N → data
K → cluster
M → iterations
TC = O(N×K×M) → very high!

② using only euclediane distance, what will happen? → will create circle (shape of cluster)
↓
so behaves like circular or spherical cluster about mean.

③ outliers will misguide

→ If know hidden variables, can predict output in more better way. (like coin biased or not)

* Can u fit "gaussian curve" to get cluster?
↳ Data set hoga
↳ uska μ and σ² nikaalo and gaussian curve bnaalo simple! (ii)

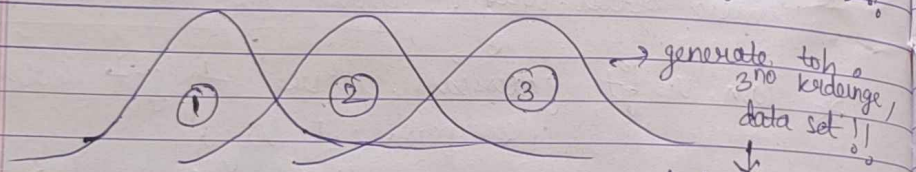
→ Say want to fit 2 gaussian! how?
↳ Bitne dialube can see from bar chart (har baar 'bar chart' bnana not easy).
→ How to mix 2 gaussian?

6/11/2024
Wednesday

Lec-11

GMM (Gaussian Mixture Model):-

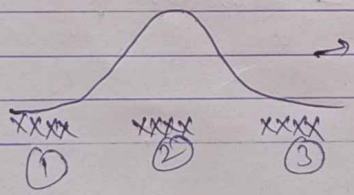
→ as gaussian curve is from $-\infty$ to ∞, thus can generate same dataset from ∞ distribution.
→ any distribution (:normal) can generate data set (any).
→ if can generate a particular data set from any normal distribution (any μ, σ), then why we want to learn particular normal distribution (just take random μ, σ), then where is difference ???



→ generate toh 3no kidhar se data set !!
↓
but only one will give high probability (more likely) to generate that data set.

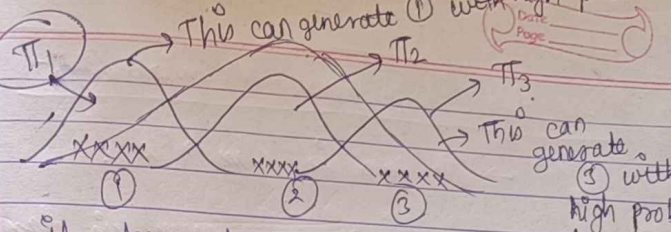
→ If have 100 data, learn μ, σ, then can generate '∞' data, not only 100. (yehi hota hai in Gen AI)
→ But kya sara data valid hai? need validation!
↳ some may be fake, thus not acceptable.

Eg:-



→ single gaussian → then
① with high prob,
②, ③ with less prob.

mixing coefficient



It does not mean we can not generate ② or ① by this.
→ That's why we have to use multiple gaussian curves sometimes.

$\sum_{k=1}^{K} \pi_k$ → weighted sum / convex combination

$\pi_k \geq 0 \ \forall \ k$

$\sum_{k=1}^{K} \pi_k = 1$ ✓

Σ_k ⇒ covariance (thus multidimensional data).

$$p(x) = \sum_{k=1}^{K} \pi_k \left(N(x \mid \mu_k, \Sigma_k)\right)$$

* $p(x) = N(\mu, \Sigma)$
↳ $x \rightarrow$ iid type data.
→ $p(x_1, x_2, \ldots, x_n) = \prod (N(\mu, \Sigma))$
→ $\log\left(p(x_1, x_2, \ldots, x_n)\right) = \sum \log\left(N(\mu, \Sigma)\right)$
↳ then differentiate wrt 'Σ'. (???)

Here ⇒ $p(x_1, x_2, \ldots, x_n) = \prod \sum_{k=1}^{K} \pi_k N(x \mid \mu_k, \Sigma_k)$

$\log\left(p(x_1, x_2, \ldots, x_n)\right) = \sum \log \sum \pi_k \left(N(x \mid \mu_k, \Sigma_k)\right)$

complex, no close form exist to solve.

WJMK

4/w Imp :-

→ 2-step algo.

(*) E-M algo (Expectation - Maximization) :-

$r_k^{(i)}$ → i^{th} data in class → 'k'.

* firstly learn 'z' → means kaun kis class mein hai usko udhar bhejdo, then har class ka μ and σ nikaalo.

↳ 2-step process.

⇒ can't learn z, μ, σ simultaneously, almost impossible.

* using ANN, can approximate any f^n.

↳ so powerful.