Day 1 (13/01/26)

- Mid : 15%
  End : 25%
  Continous Assessment : 60%

- In ML, experience ≡ dataset
  ↳ Machine + Dataset + Algorithm ⟹ Output
  (ground truth)

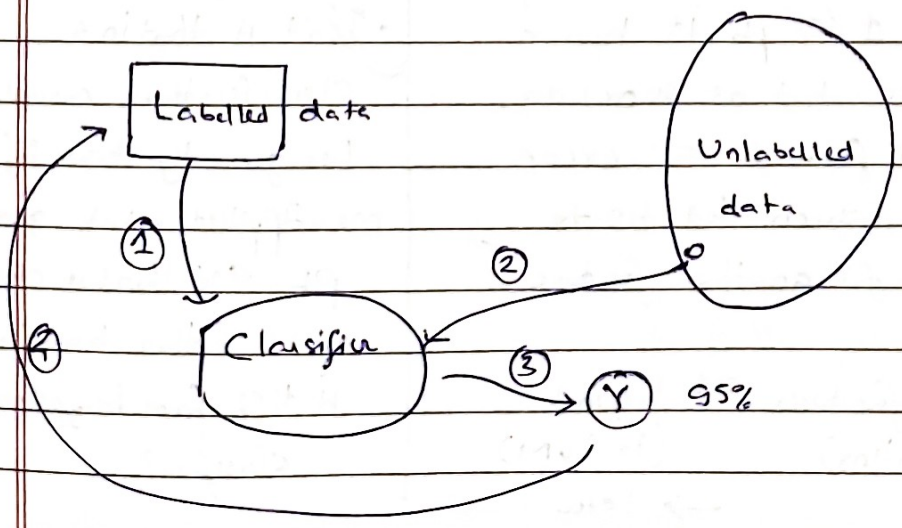
- ML Paradigms ⟶ i) Supervised : Labelled Training dataset
  ii) Unsupervised
  iii) Reinforcement : learning from trial and errors
  → Includes association rules mining
  → representation learning (embedding)
  → Identifies relationship

RL : Agents + Environment ⟲ Actions, Rewards + Feedback

Q. In unsupervised learning, since we don't label the data, in that case when there is an output how is the data labelled ? (The model transforms or organises the input into a new representation)

Q what includes in unsupervised ? (Algo?) (ex. clustering,)
↳ Includes algorithms where
i) no labels are provided
ii) the goal is to discover structure

- Semi-supervised learning



1 → train using Supervised
2 : predict / classify
3 : if correct
4 : add to data set

- In all ML tasks, a 'good representation of data' very imp: 1st Step!
  - ↳ even in DL
  - ↑↑
  - as in input
  - feature engineering (not req in DL, why?) - automatic feature learning
  - one type of representation: vector space model ⇒ includes feature vectors

(i) feature selection: out of all features (feature set), choosing some ← affects performance, computation

(ii) dimension reduction: no removal of features, but projection in lower dimension
↳ (3D) → 2D

- DL helps in feature engineering (✓) {note: no elimination of features; but automating it}
  - → done by the model itself

Day 2 (15/01/26)

- No of classes predefined in Supervised model/learning
- Probabilistic model — Bayesian classifier (at the last!)
  - ⇓
  - → note: softmax applied in NN!
- NOTE: when we had only one feature and we had 1/0 pb it was a good feature but as soon as, pb dropped from 1 ⇒ error introduced which led us to think/search for another feature
  - ⇓
  - increasing features ⇒ Kernel (in SVM)
  - (upscaling)
  - → later
- isn't it that in a classification model, we generally have soft-max applied which gives pb and then we classify? → but then how is it diff from bayesian classifier?

- Upscaling and downscaling always done on datasets

- Classical (Traditional) ML Methods for Classification

a) Bayesian classifier — (~~Suitable for 2 classes~~) — for more than 2 classes

(finding plane for separation)

$P(C_i | \bar{x}_j)$

C_i ↑ Class label

$\bar{x}_j$ ↑ new sample → (it is a feature vector)

$\max_{C_i} P(C_i | \bar{x})$

likelihood ↓ Prior ↓

NOTE: Bayes rule: $P(C | \bar{x}) = \frac{P(\bar{x} | C) P(C)}{P(\bar{x})}$

↓ posterior

↑ Evidence

- given the sample, what's the class label? ($P(C | \bar{x})$)
- $P(\bar{x} | C)$ — pb of picking x given the class C_j
- $P(C)$ — pb of picking the class C_i
- $P(\bar{x})$ — pb of picking x in the universe

NOTE: if $P(C_i | \bar{x}) > P(C_j | \bar{x})$

$\Rightarrow$ x is C_i

$P(x | C_i) * P(C_i) > P(x | C_j) * P(C_j)$

↳ This implies no significance of denominator

NOTE: if prior probability not known, then it is considered same! (That is distribution equal)

$\Rightarrow P(\bar{x} | C_i) = P(x_1 x_2 x_3 \ldots x_k | C_i)$ → (Conditions)

(Pb1) $= P(x_1 | x_2 x_3 \ldots x_k C_i) \cdot P(x_2 | x_3 \ldots x_k C_i) \cdot \ldots \cdot P(x_k | C_i)$

- (expensive task) — computationally

(Pb 2) if data set sparse, then (Condition)

may turn to 0 //

⇓

What's the way out?

(i) features are conditionally independent - considered as

↙ bag of words

(naive bayes classifier) (may loose some info) ⟹ $P(x_1 | x_2 x_3 \dots x_k C_j) = P(x_1 | C_j)$

(after assumption)

(ii)

NOTE: none of $P(x_i | C_j)$ should be zero after applying (i)

⇒ so we apply smoothing

$$P(x_i | C_j) = \frac{0 + \text{small}}{|C_j| + \text{large}}$$

how to estimate this pb? → probability

(among all elements in C_j, how many of them having x_i feature)

→ Bayesian classifier with Gaussian / Multinomial distribution

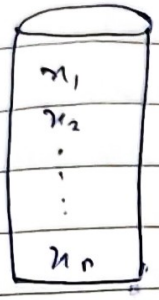
here the assumption is: it is already there in the dataset / dataset has gaussian distribution

(b) K Nearest Neighbours → (representative finding)

&

Centroid Based Classifier

⇒ Lazy classifier / Instance Based classifier



$\bar{x}_{new}$ → • find distance with all elements in dataset
• pick K least one : closest one

⇒ This is known as inference time : where we identify the class label for a new input sample!

⇒ dependent on size of dataset for (KNN) — The inference time

K : no. of features ⇐ $O(nK)$ — for KNN

n : no. of inputs in dataset

$O(K)$ — for Bayesian

(b1) Centroid Based Classifier

- We have representatives for each class
- take mean of all feature vectors ≡ Centroids

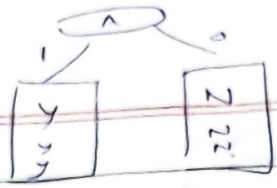
for a new input : find the distance for that from each class's representative

NOTE! Training over here ≡ finding centroid for each class

Inference time less!

(c) Decision Tree — (Rule based approach)
(Random Forest)

- Objective : to find rules for classification → (ie build the tree)
- leaf nodes — Class labels
  Internal nodes — the rules / conditions
- Many trees possible ≡ (but need to find best one)



Day 3 (19/01/26)

(C) Decision tree contd

Q If there are 2 features, which one should be root?

⇒ either of them can be taken and form tree

⇒ but one is ~~good~~ good : to have resultant dataset as homogenous !!

(i.e the uncertainty should be low)

achieved/done by information theory

$$X = \{x_1, x_2, x_3, \ldots, x_n\}$$

$$H(X) = \sum_{x \in X} P(x) \cdot \log\left(\frac{1}{P(x)}\right) = E\left(\log\left(\frac{1}{P(x)}\right)\right)$$

$$= -\sum_{x \in X} P(x) \log P(x)$$

entropy

expectation

Information content (or uncertainty)

- Data: 1, 2, 1, 5, 4, 2
- Mean: $\frac{\sum \text{Data}}{|\text{Data}|}$

$$= \frac{1+2+1+5+4+2}{6}$$

$$= 1.\left(\frac{1}{6}\right) + 2.\left(\frac{1}{6}\right) + 1.\left(\frac{1}{6}\right) + \ldots + 2.\left(\frac{1}{6}\right)$$

1 is value

$\frac{1}{6}$ is its probability

$$= 1.\left(\frac{2}{6}\right) + 2.\left(\frac{2}{6}\right) + 5.\left(\frac{1}{6}\right) + 4\left(\frac{1}{6}\right)$$

$$= \sum x \cdot P(x) = \text{expectation}$$

This gives: $X = \{1, 2, 4, 5\}$

- given an event, how much info we ~~have~~ / don't know
- Note: if $P(x)$ high, $\frac{1}{P(x)}$ low

⇒ less uncertain

→ $H(x) = n \cdot \frac{1}{n} \log\left(\frac{1}{n}\right) = \log(n)$ distributes $p = 1/n$!

(If distribution equally like highest entropy at middle)

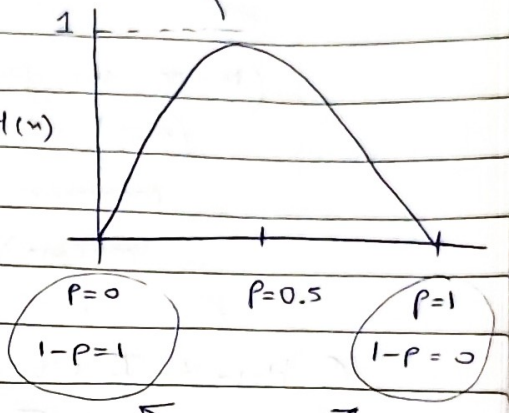
Date
Page

$X = \{H, T\}$

p: probability of get head

$1-p$: " " " tail

$$H(x) = -p \cdot \log p - (1-p)\log(1-p)$$



(entropy is 0)! as certain outcome

- Most fundamental Decision tree is ID3!

⇒ Information gain : difference in the entropy

⇒ if uncertainty present → keep on dividing till homogenous

⇒ if leaf nodes has only 1 element that means the tree construction is weak (i.e rules weak)

⇓

Random forest, Ensembling method } all applied on ID3 (& extension)

(d) Support Vector Machine (SVM)

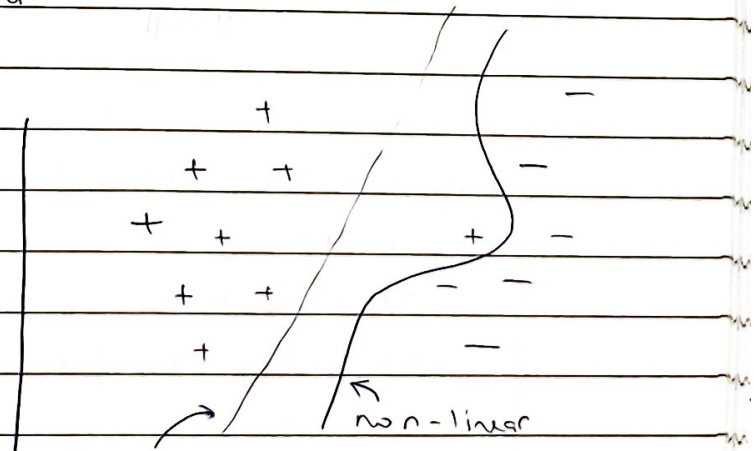
- discriminant based method
- finds a separating hyper-plane

Linear SVM

$g(x) = W^T x + W_0 = 0$

if $g(x) > 0$: positive class

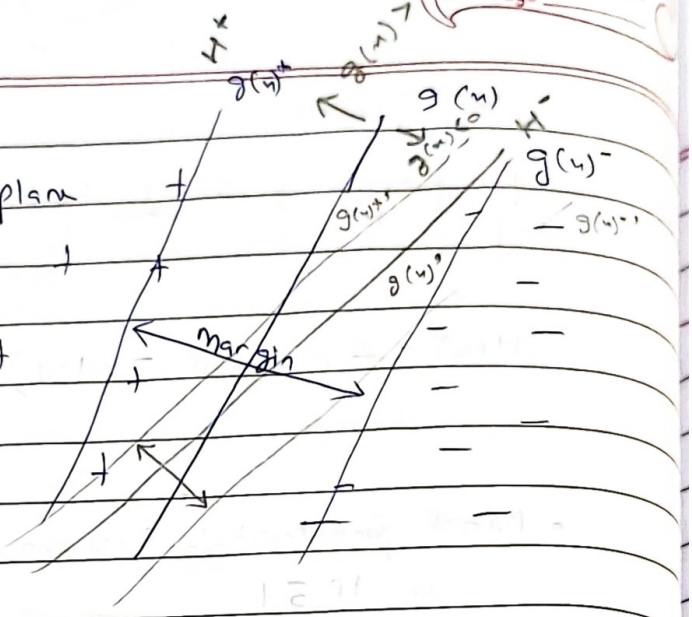
if $g(x) < 0$: negative "



linear SVM (LSVM)

SVM (complex) ⇒ but we will solve it by LSVM

(project in higher dimension!) ⇐ not possible in lower dimension (in terms of separating) ⇐ ⇒ req data modification (or separable)



(NOTE: We find that hyperplane ($g(x)$) which results in Maximum Separating margin)

Q How to find w of hyperplane

- take two points on $g(x)=0$

$g(x_1) = g(x_2) = 0$

$\Rightarrow w^T x_1 + w_0 = w^T x_2 + w_0$

$\Rightarrow w^T (x_1 - x_2) = 0$

$\Downarrow$

This indicates w is orthogonal to $g(x)=0$!

- $g(x)$ better than $g(x)'$ parallel
- NOTE: $g(x)^+$ is the hyperplane to $g(x)$ passing through nearest positive class similarly $g(x)^-$
- Note: We have infinite $(g(x)^+ - g(x) - g(x)^-)$ planes but ideally we take the middle one!

NOTE: $r = \frac{g(x)}{\|w\|}$: distance of point from hyperplane

Day 4 (20/01/26)

(d) SVM (Linear)

→ Video lesson will be uploaded for implementation

→ binary classifier

Q What is the magnitude of separating margin?

$H^+ : g(x) \geq 1$

$H^- : g(x) \leq -1$

$\Rightarrow y_i (w^T x_i + w_0) \geq 1 \quad \forall x_i$

$y_i = 1$ if H^+

$= -1$ if H^-

classmate
Date
Page

Margin
= distance of origin from H^+ − distance of origin from H^-

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

{What about points b/t +1 to 0 and 0 to -1?
considered inside margin
penalized later
NOTE: margin boundaries $g(x) = 1$ or $g(x) = -1$}

- Maximising $\frac{2}{\|w\|} \equiv$ minimising $\frac{\|w\|}{2}$

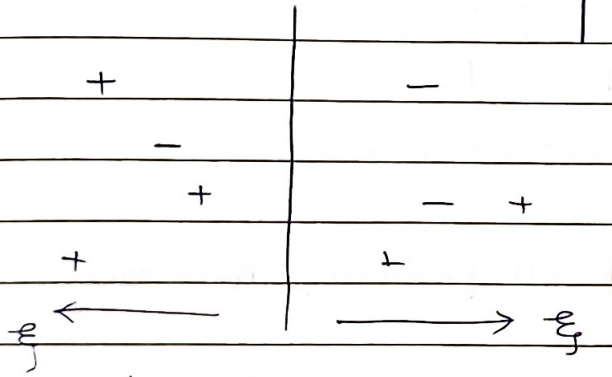
$$\equiv \| \frac{w^T w}{2} \quad \text{: objective function}$$

Subject to: $y_i(w^T x_i + w_0) \geq 1 \; \forall x_i$

⇒ using Lagrange multiplier

$f(x)$
s.t $g(x) = c$
$\Rightarrow f(x) \pm \lambda(g(x) = c)$

Q What if data is not linearly separable?
→ 2 approaches
→ Approach 1: Slack variable



$\Rightarrow y(w^T x_i + w_0) \geq 1 + \xi$

→ not linearly separable
→ so pull both the sides by some ξ

→ Approach 2: Kernel (How to formulate: look at vid?)
- not separable in lower dim → go to higher dim (but costly)
- So Kernel idea is: don't transform (project) to higher dim but solve it like how it is solved there!

Q What are support vectors? (Points that touch decision and margin boundary or fall between them)

- Vectors lying on separating hyperplane
- $\lambda_i \neq 0$ for non-support vectors

(Check)
- ~~Why~~ only 3 points?
- 3 types of support vectors (on margin, inside margin, misclassified)!

Q How to evaluate our model? (# Evaluation)

(i) Dataset types:

- Training Dataset ⇒ Building
- Testing " ⇒ Evaluation

} Labelled Dataset

- Development data: used for fine tuning parameters
  may or may not req

(ii)

		Predicted Label +	Predicted Label −	
Actual label	+	TP	FN	P
	−	FP	TN	N
		P'	N'	

(iii) Accuracy: out of all predicted, how many correct

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

(iv) Precision: among all positive predicted, how many actually correct

- Binary in nature
- Keep one var as positive } *
  Similarly for Recall

$$\frac{TP}{TP+FP}$$

(iii) Recall : out of actual positive, how many predicted correctly

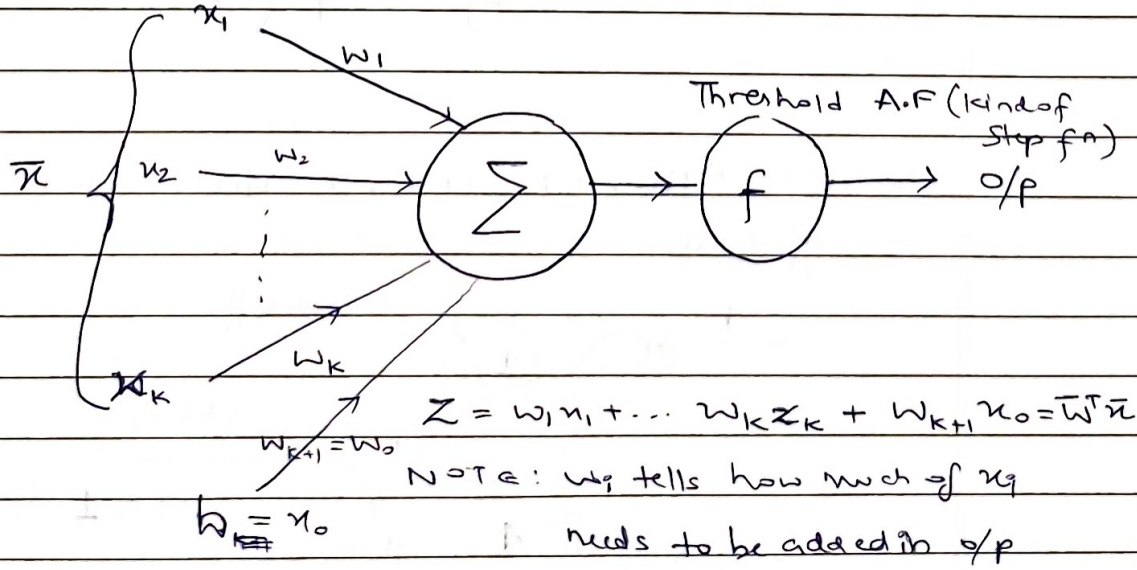
$$\frac{TP}{TP+FN}$$

(iv) Fmeasure : harmonic mean of Precision and recall

Clustering
~~(i)~~ K-mean
~~(ii)~~ hierarchical clustering
(iii) density based "

Regression & Video Clustering lessons

Day 5 (21/08/26)
Artificial Neural Network
(i) Perceptron
→ Solves the linearly Separable Problem
→ Start with some g(x) which is not separating completely at first but then we want it to move in certain dir^n so it separates the points



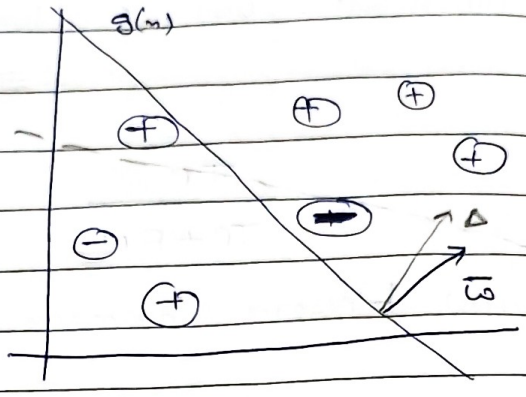
$$Z = w_1 x_1 + \dots w_K x_K + w_{K+1} x_0 = \bar{w}^T \bar{x}$$

NOTE: w_i tells how much of x_i needs to be added in o/p

$w' = w^0 - \eta \Delta$

learning rate

$\eta \in [0, 1]$



$g(x) > 0 \Rightarrow +$

$g(x) < 0 \Rightarrow -$

$z = \bar{w}^T \bar{x}$

M: misclassified examples

NOTE: loss $L_i = \begin{cases} -y_i(w.x_i) & \text{if misclassified} \\ 0 & \text{otherwise} \end{cases}$

$E = \sum L_i = -\sum y_i (w.x_i)$

objective function: $E = -\sum_{x_i \in M} (\bar{w}^T \bar{x}_i, y_i)$ } Why is the error term weird?

NOTE: if correct classified, $y_i (w.x_i) > 0$

else $y_i (w.x_i) < 0$

minimizing $\Rightarrow \frac{\partial E}{\partial w} = -x_i y_i$ for one misclassified point

$\Rightarrow w^{t+1} = w^t + \eta x_i y_i$ ← $x_i \in M$

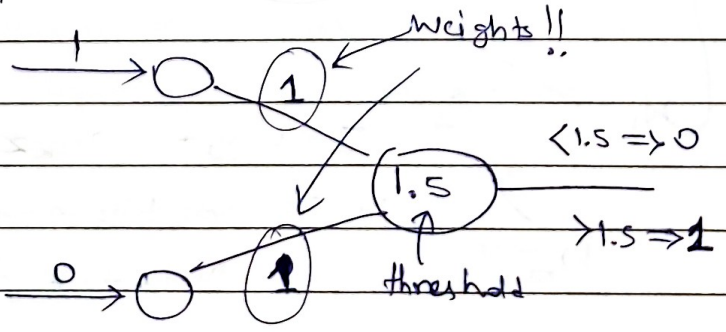
$\Rightarrow w^{t+1} = w^t + \eta \sum x_i y_i$ (for all)

↳ only for misclassified samples

Paper: Neural Networks for pattern recognition by Bishop (Ch 2) } Single layer N.N

NOTE: AND, OR, NOR, NAND : solved by perceptron

AND

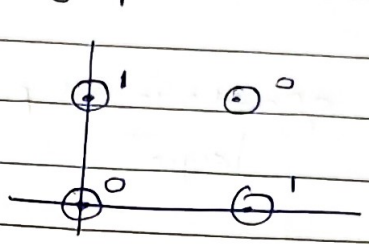


NAND $\Rightarrow [-1, -1] = w$, threshold $= -1.5$

OR : [1, 1] = w, threshold = 0.5

XOR : not linearly separable

0	0	0
0	1	1
1	0	1
1	1	0

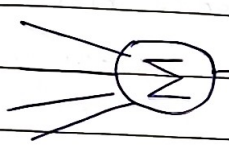


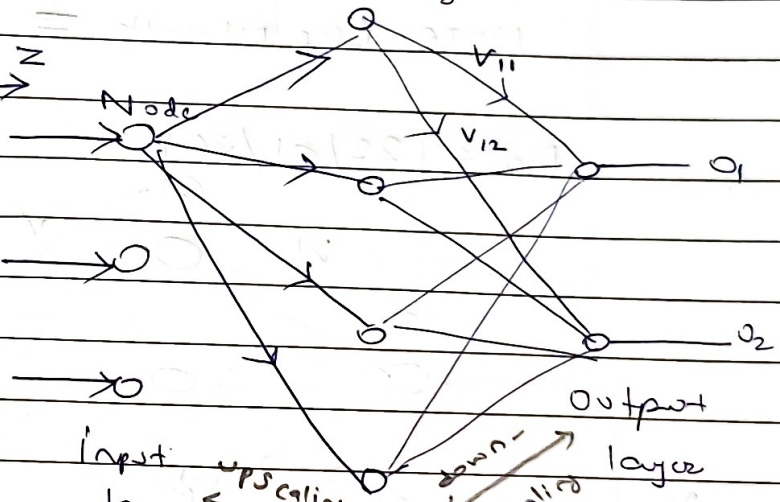
⇒ add more layers

(ii) Multilayer Perceptron / Feedforward NN / ANN

⇒ Constraints the o/p to some range

Node :





$z = f(w^T x) + b$

(by how much the shift is done)

activation function : linear hard!

$W =$

	h_1	h_2	h_3	h_4
x_1				
x_2				
x_3				

$h_1 = f(w_{11}x_1 + w_{21}x_2 + w_{31}x_3)$

$= f_h(w_{y1}^T x)$ ↳ ye no. of (x)

(Deep network == multiple hidden layers)

feedforward

$\Rightarrow \bar{h} = f(\bar{w}^T \bar{x})$

(Output of hidden layer!)

Similarly, $O = f_o(\bar{V}^T h)$

$O_1 = f(V_{11}h_1 + V_{21}h_2 + V_{31}h_3 + V_{41}h_4)$

$\Rightarrow O = f_o(\bar{V}^T f_h(w^T \cdot \bar{x}))$

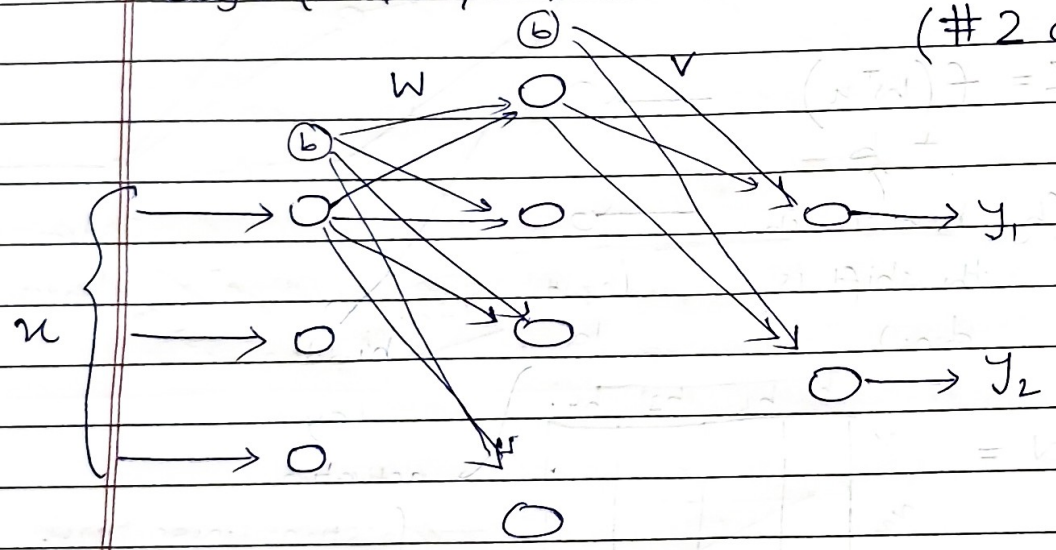
for classification pb,

- no. of nodes in input layer ≡ no. of features in a feature vector
- no. of nodes in out put layer ≡ no. of classes
  ↓
  pb assigned to each class

for regression → only 1 node in o/p layer

NOTE: Deep network ≡ more than 1 hidden layer

Day 6 (22/01/26)



(# 2 class classification)

$$h = f(\underbrace{W^T x}_{z_h})$$

bias for hidden

params = 3×4 + (4) + 4×2 + (2)

bias for output

Q What is W? → unknown

⇒ Start with unknown values

$$\bar{y} = f(\underbrace{V^T h}_{z_o})$$

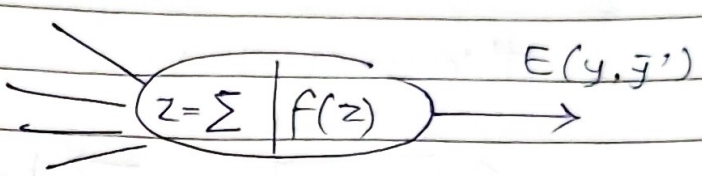
NOTE: output of nn can be non-discrete as well

expected is ground truth ($\bar{y}$)

observed is o/p of nn ($\bar{y}'$)

$$E = e(y, \bar{y}')$$

Backpropagation



$$z = \sum v_{ji} h_i$$

$$\Rightarrow z_1 = V_{11} \times h_1 + V_{21} h_2 + V_{31} h_3 + V_{41} h_4$$

$$y = f(z)$$

E = error function

$$\Rightarrow \frac{\partial E}{\partial v} = 0 = \Delta v$$

$$\frac{\partial E}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial v} = 0$$

} Do this for all parameters (v_{ji})

$$\frac{\partial z_1}{\partial v_{11}} \cdot \frac{\partial y_1}{\partial z_1} \cdot \frac{\partial E}{\partial y_1} = 0$$

$$\Delta V_i = \frac{\partial z_i}{\partial v_{ji}} \cdot \frac{\partial y_i}{\partial z_i} \cdot \frac{\partial E}{\partial y_i} = 0$$

$$\boxed{V' = V^o + \eta \Delta v}$$

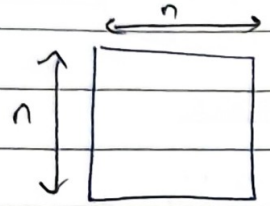
$$\Rightarrow \Delta w = \frac{\partial E}{\partial w} = 0$$

$$\frac{\partial z_h}{\partial w} \cdot \frac{\partial h}{\partial z_h} \frac{\partial z_o}{\partial h} \frac{\partial y}{\partial z_o} \frac{\partial E}{\partial y} = 0$$

z_o ↑ output layer's output

NOTE: if # classes = 2 we needn't req 2 nodes in o/p layer → one is enough with threshold

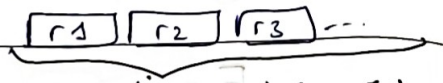
Image Classification



→ ANN

⇒ how to give this to prev nn architecture?

- Convert 1024X1024 into a vector
- Concatenate each rows back to back:

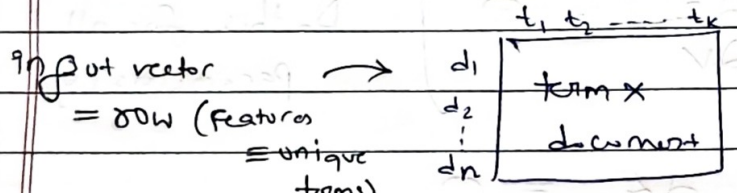


constitute an input vector
#features = n^2

Text Processing / Classification

↳ What features to consider?

(i) Unique words in corpus — so build a matrix like



where |term| = k ↑ unique words

A cell tells if t_j term present in d_i document ↑ a single word

Instead we can have, ⟵

(ii) We can have sort of frequency stored also instead of just binary counting

↓

bigram : 2 words at a time

trigram : 3 words at a time

NOTE: Representation imp for a good model ⇒ to be able to use in ANN

↑ Prediction!

as in unigram, bigram } etc

Error functions

(i) Supervised Problem

- MSE – req continuous, differentiable, monotonic function

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

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

Joint Probability *

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

if independent : $P(x)\,P(y)$

(ii) Cross entropy

↳ uncertainty in a r.v

$$H(X) = \sum_{x \in X} P(x) \cdot \log\left(\frac{1}{P(x)}\right)$$

↑ to expectation of information content

$$= -\sum_{x \in X} P(x) \cdot \boxed{\log(P(x))}$$

↳ Information content

↳ Amt of uncertainty of the event

→ uncertainty of two variables together

Joint Probability : Probability of both events happening

$$H(X,Y) = -\sum_{x \in X}\sum_{y \in Y} P(x,y) \cdot \log(P(x,y))$$

Joint entropy

if X & Y independent ⇒ $H(X,Y)$ ↑

else ⇒ $H(X,Y)$ ↓

Relative entropy : measures how much one is diff from other

$$H(X|Y) = \sum_{y \in Y} P(y) \cdot H(X|y)$$

↑ for unsupervised learning (KL divergence)

↑ marginalisation on condition y

$$= -\sum_{y \in Y} P(y) \cdot \sum_{x \in X} P(x|y) \log(P(x|y))$$

$$= -\sum_{y \in X}\sum_{x \in X} P(x,y) \log(P(x|y))$$