⇒ Understanding feature space

(Q.) Define I/p data of a typical ML alg ?

Ans:- I/p data is a collection of features that have been quantitatively measured from some object or event that we want the machine learning system to process.

Linear Regression

Task is to prepare a smoothie and serve.

Goal : To predict the calories

case I

Experience : ?

performance : ?

Only we know are

Smoothie = 1 cup of yogurt + 1 secret ingredient + some color dyes.

Calories : ?

case II Experience : 15 Smoothies are given 265, 281, 207, 148, 240, 154, ---, 191

Smoothie	calories
1	265
2	281
15	191

first Mean = 236.9

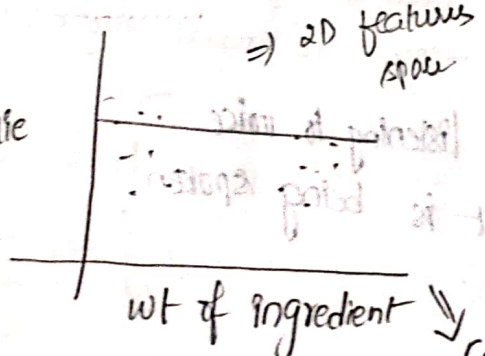
find Median = 240

If 16 - 237 actually 400

case III you can weight the (Smoothie) of ingredient

S.NO	weight	calorie
1	188 g	265
2	156 g	281
15	110 g	191

⇒ 2D features space



Calories = mean + θ * weight

↳ Intercept ↳ slope

for another model

we need

Task or Goal : Pred

Model 1 : wild gue

↳ very bad

↳ we are no

Model 2 : we use

268, ... 191 =

↳ NO

↳ Pred

but actual

↳ Pr

↳ Unc

Model 3 :-

⇒ Let us C

Calories

Mode

Model 3

(Root Mean S

for another model $c = w_0 + w_1 * \text{weight}$

we need to find w_0, w_1 from the data

Task or Goal : Predict calories

Model 1 : wild guess

↳ very bad model

↳ we are not having any experience

Model 2 : we use experience of 15 previously made smoothies

268, ... 191 ⇒ mean = 236.9

Median = 240

↳ NO features are used

↳ Prediction for 16th smoothie is 238

but actual calories = 445

445
238
207

↳ Ingredient quich of 88 gm

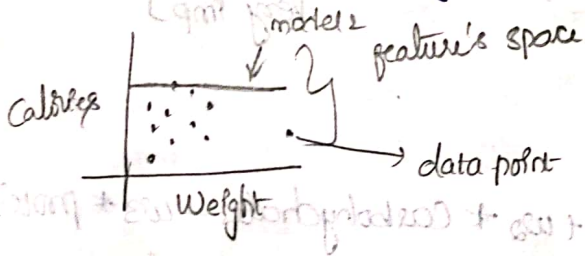
↳ Under estimation by 208

Model 3 :— more data

↳ doesn't mean more data points

↳ it means more information about the data.

⇒ Let us consider weight of the ingradient

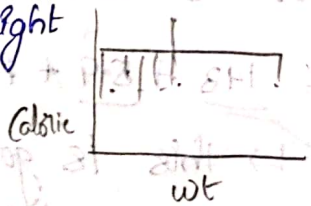


feature o/p

weight	calorie

Model 2 calories = $236.9 + 0 * \text{weight}$

↳ Not a good model.



Model 3 :— calories = $w_0 + w_1 * \text{weight}$

RMSE = 84.6 (for model 2)

(Root Mean Square error)

Simple regression

Model 3 :- $C = 236.9 + 0.5 * \text{weight}$ RMSE = 69.3

4 :- $C = 236.9 + 1 * \text{weight}$ = 69.2

5 :- $C = 200 + 1 * \text{weight}$ = 50.8

finally :- $C = 175 + 1.35 * \text{weight}$ RMSE = 48.8

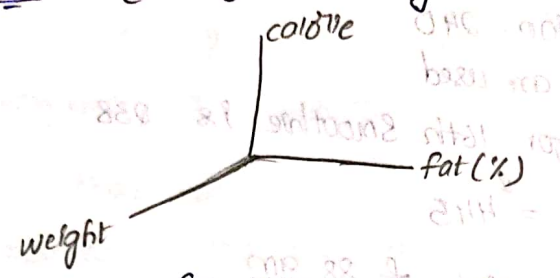
(to make it much better)

Model 6 :- → for 16th smoothie the prediction = 293.8

↳ underestimated by 455 − 293.8 = 151.2

Multiple regression

Model 7 :- $C = w_0 + w_1 * \text{weight} + w_2 * \text{fat (\%)}$ → plane (3D)



for model 3 : straight line

$cal = w_0 + w_1 * wt$

$C = 163 + 1.49 * \text{weight} + 51.9 * \text{Fat(\%)}$

There are the best parameters that gives good performance

RMSE = 46.9

16th smoothie prediction is underestimated by an amount = 143

So consider the other features (9-4-4 ratio) (domain knowledge is very imp)

- Fat (%)
- carbohydrates
- protein

Model 8 :- $C = w_0 + w_1 * \text{Fat(\%)} + w_2 * \text{Carbohydrates} + w_3 * \text{protein}$

$C = 143 + 8.9 * \text{Fat(\%)} + 3.9 * \text{Carbohydrates} + 4.3 * \text{protein}$

↳ this is going to be good model

The calories present in 1 cup of yogurt 16th smoothie RMSE < 10

pred = 431

Fat	Car	protein	calorie

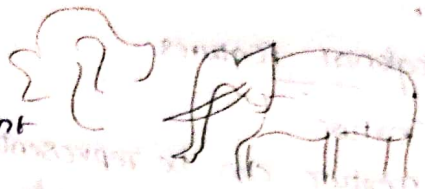
Summary :- Feature selection is very imp

Scanned with CamScanner

Classification – computer vision

Task 1: Lion or Elephant

Shape features are sufficient

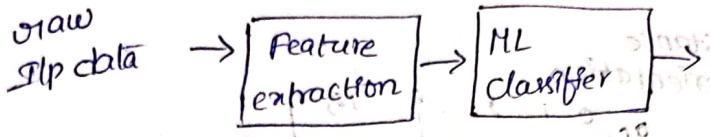


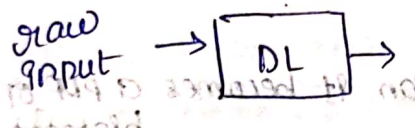
Task 2: horse vs zebra

Shape features are not useful

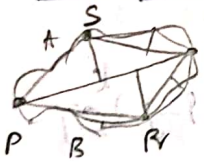
but regional (texture) features are useful

ML vs DL





Shape Descriptors : Polygonal representation



The borders of an object

→ by joining it we get a 'polygon shape (Perfect)

If we want better shape → repeat the process.

{ → start treating it as some fun

rotate by 90° ⇐ when we normalised ⇒ we call it bar probability density fun?

So, It has

mean (μ) =

variance (σ^2) =

skewness =

for Boundary segment 1 (B1)

Find μ =

σ^2 =

Skew =

B2 ⇒

for 8×3 = (24) features

30/12/22.

Regional Features

① Texture

Texture can be represented using cooccurance matrix

$A_p(i,j)$ 4-bit image of size 8×8
- gray level i
- gray level j
- some position vector

Raw Image

10	9	7	9	5	8	11	9
65	5	15	12	4	6	3	2
9	3	2	10	6	8	4	5
8	2	4	3	7	5	6	1
2	0	11	8	10	9	8	2
8	4	4	1	6	0	7	2
2	3	8	9	11	6	3	9
7	2	8	8	6	12	6	7

$i=4,\ j=8$ $\quad P = \langle d, \theta \rangle = \langle 1, 45^\circ \rangle$
- d → distance
- θ → orientation

⇐ Count

$A_p(i,j) = $ [matrix with rows/columns 0, 1, 2, …, 8, …, 15; entry at (4, 8) = 3] ⇒ cooccurance matrix

$\langle 1, 45^\circ \rangle$

$A_p(i,j)$ 16×16 → normalize → then it becomes a PDF or histogram

given $\langle 1, 45^\circ \rangle$ and denote it has c_{ij} 16×16

① Max prob: $\max_{i,j} c_{ij}$

② Elementwise difference moment: $\sum_i \sum_j (i-j)^k c_{ij}$

③ Inverse elementwise difference: $\sum_i \sum_j c_{ij} / (i-j)^k$

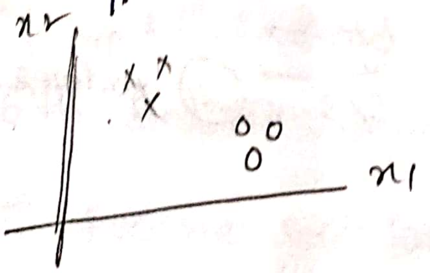
④ Uniformity: $\sum_i \sum_j c_{ij}^2$ ✓

⑤ Entropy: $-\sum_i \sum_j c_{ij} \log(c_{ij})$

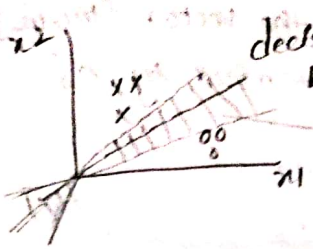
⇒ Dataset

	x1	x2				
	①	②	③	④	⑤	Label
						horse/zebra

y.



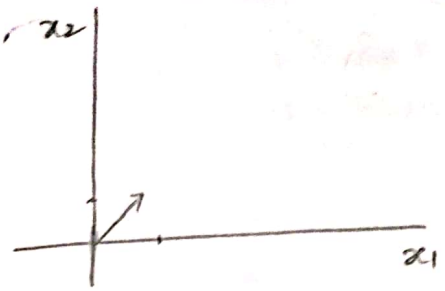
Linear Classification :-



decision boundary $w_1x_1+w_2x_2+w_0=0$

Decision boundary is the normal of the vector $[w_1, w_2]^T$

$[w_1, w_2]^T = [1,1]^T$



$\Rightarrow \sum_{i=1}^{d} w_i x_i + w_0 = 0$, the decision boundary $w^Tx + w_0 = 0$

Decision rule is

If $w^Tx + w_0 > 0$; then $x \in C_1$

If $-w^Tx + w_0 < 0$; then $x \in C_2$

We want to unify them into one rule.

$$[w_1\ w_2 \cdots w_d] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} + w_0 = 0$$

$$\underbrace{[w_1\ w_2 \ldots w_d\ w_0]}_{a} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix}}_{y} = 0$$

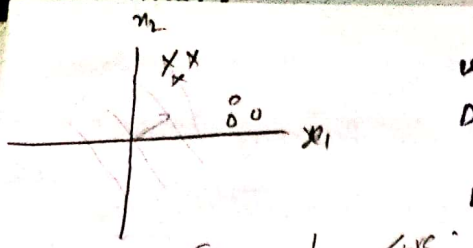
Decision rule :

$a^Ty > 0$, $y \in C_1$

$a^Ty < 0$, $y \in C_2$

Now, negate y of class 2

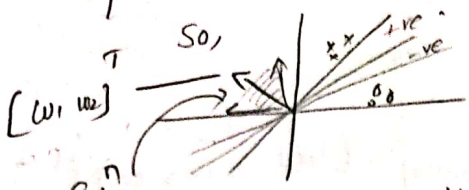
Unified decision rule : $\boxed{a^Ty \geq 0}$



$w_1x_1 + w_2x_2 + w_0 = 0$

Decision boundary
↳ is normal to the vector $[w_1, w_2]^T$
↳ position is determined by w_0

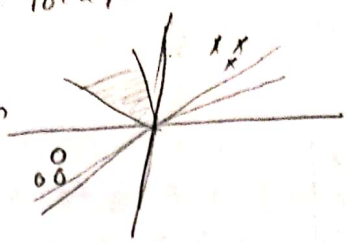
So, $[w_1\ w_2]^T$



Soln zone

After negation $\forall y \in C_2$

for $a^Ty > 0$



Algorithm SGD

$a(0) \leftarrow$ arbitrary Initialization

$a(k-1) \rightarrow a(k)$

2|1|23.

Regression
↳ fitting the hyperplane
Classification
↳ Finding the DB which separates two classes

→ Input data ↳ features etc.
→ Model Architecture
→ Training strategy
→ Optimization Alg
→ RMSE Loss fun^n
→ finally performance on Test set.

We can achieve above tasks using loss fun^n.

In Regression Calories = $w_0 + w_1 * Fat(\%) + w_2 * Cabs + w_3 * protein$

Prediction = $w_0 + w_1 * feature1 + w_2 * f_2 + w_3 * f_3$

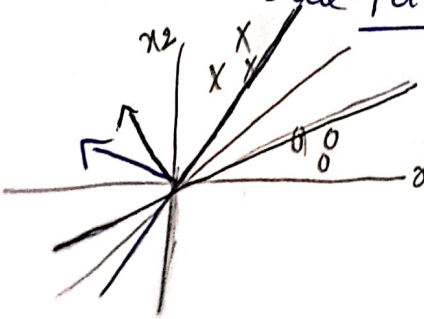
In classification decision boundary is a hyperplane

SGD

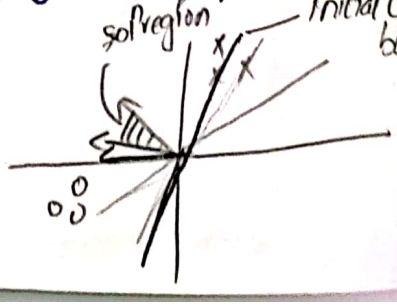
$w_1 * f_1 + w_2 * f_2 + w_0 = 0$

Decision rule $a^Ty > 0$

$a = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \\ w_0 \end{bmatrix}$ $y = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix}$



⇒ After negating $y \in$ class 2



Training Alg

① $a(0) \leftarrow$ arbitrary

②(a) $a(k) \leftarrow a(k-1)$ h...

how much the corresp...

$a^Ty > 0$

$w_1 * x_1 + w_2 * ...$

(b) Error $J = ...$

$\nabla J = ...$

(c) $a(k) = a(k...$

$\eta - ...$

$a(k) = a...$

Importance

if $\eta = 1 \Rightarrow a(...$

$\eta = 0.1, 0.2$ Cus...

Inference

In Regression ...

In classification ...

⇒ Inference mean... performing the ... by the model.

In the train...

$MSE = ...$

Training Alg

① $a(0) \leftarrow$ arbitrary (number) Initialize

②(a) $a(k) \leftarrow a(k-1)$ In such a way that it min some performance index

how much the corresponding $a(k)$ wt is useful in deciding the boundary

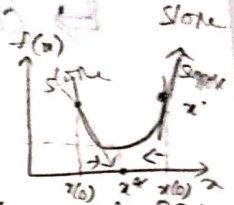
$a^T y > 0$

$w_1 * x_1 + w_2 * x_2 + w_0 > 0$

(b) Error $J = -\sum a^T y$ (Gradient gives slopes)

$\forall y$ misclassified

$\boxed{\nabla J = -\sum y}$ we use this gradient to update the weights.



$x^* = \arg\min_x f(x)$

here x is wt

$x(k) = x(k-1) - \eta \frac{\partial f(x)}{\partial x}$

+ve slope → decrease x

-ve slope → increase x

(c) $a(k) = a(k-1) - \eta \nabla J$

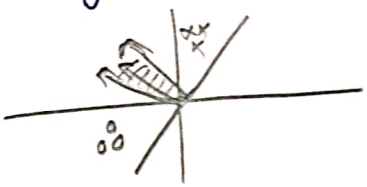
η – learning rate.

$\boxed{a(k) = a(k-1) + \eta \sum y}$ $\forall y$ misclassified

Importance of η

if $\eta = 1 \Rightarrow a(k) = a(k-1) + \sum y$ $\forall y$ misclassified

$\eta = 0.1, 0.2$ (usually)



Inference

In Regression $w_1 x_1 + w_2 x_2 + w_0 = 0$

In classification $a^T y = 0$ or $w^T x = 0$ → Ag by 1 → augmented by w_0

where $w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_0 \end{bmatrix}_{(d+1)*1}$

⇒ Inference means (trying to do that) performing the tasks on an unseen data by the model. (Test data) so $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ 1 \end{bmatrix}$.

In the training phase, have training err $= \frac{1}{m} \|\hat{y} - y\|_2^2$ (In reg tasks, 2nd Norm)

$\boxed{MSE = \frac{1}{m} \|y - \hat{y}\|_2^2}$

* Tasks :-

Now we have

$X^{(train)}$ $y^{(train)}$ $X^{(test)}$ $y^{(test)}$ & W

Training Error $MSE = \frac{1}{m} || \vec{y}^{(train)} - y^{(train)} ||_2^2$

$= \frac{1}{m} || w^T x^{(train)} - y^{(train)} ||_2^2$

Similarly we have Test error on Test set

Test error $= || \vec{y}^{(test)} - y^{(test)} ||_2^2$

↳ Called as Generalization Error.

Optimization vs ML

Generalize the model the well on the test data

Now we have
$x^{(train)}$ $y^{(train)}$ $x^{(test)}$ $y^{(test)}$ & w

Training Error MSE $= \frac{1}{m} \| \hat{y}^{(train)} - y^{(train)} \|_2^2$

$= \frac{1}{m} \| w^T x^{(train)} - y^{(train)} \|_2^2$

Similarly we have Test error on Test set

Test error $= \| \hat{y}^{(test)} - y^{(test)} \|_2^2$

↳ called as Generalisation Error.

Optimization vs ML

Generalize the model the well on the test data

3/1/12

Generalization :—

It is the ability of machine learning system to perform well on the new unseen data.

⇒ It means to say we want the testing error also too small as well

⇒ Capacity, Underfitting & Overfitting

prediction $= w_1 x_1 + w_2 x_2 + w_0$

The different weight vector values gives us different functions.

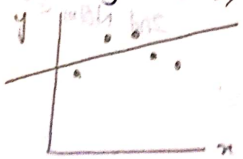
→ The range of functions which the model can learn is called the capacity capacity of that model.

→ The goal then becomes selecting one among those functions which will best fit the data (or) which will well separate the data corresponding to two categories respectively.

Regression

straight line

For Regression



⇒ Underfitting the data

$y = w_1 x + w_0$

It's capacity is it only

⇒ low capacity

We only a straight line

* Challenges of ML

① the Curse of dimensionality

Suppose we have

+ 2nd feature

10^2 ⇒ 100 regions

+ 3rd feature

10^3 ⇒ 1000 regi

② Local cons

③ Manifold

↳ Suppose

ML Appro

Raw i/p → Feature Extraction

DL Approa

Raw i/p

Reasons the model complexity and distribution are almost identical

$y = w_0 + w_1 x + w_2 x^2 + \dots$

→ Model capacity is fit the given data it is best fitting

$y = w_0 + w_1 x + w_2 x^2 + \dots + w_{20} x^{20}$

Many variety of curves

→ Model complexity

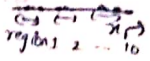
→ Over fitting

→ All these will are Linear regression since parameters are zero & here training error is zero

* Challenges of ML driving us to DL :— 3 challenges

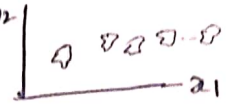
① the Curse of ~~data~~ dimensionality

Suppose we have only 1 feature → need only one unit



+ 2nd feature

$10^2 \Rightarrow$ 100 regions



→ we need to separate with 100 decision boundaries

+ 3rd feature

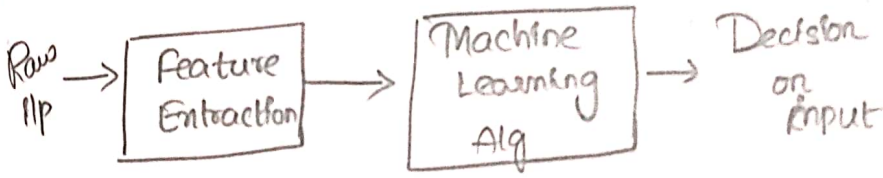
$10^3 \Rightarrow$ 1000 regions

② Local constancy and smoothness regularization.

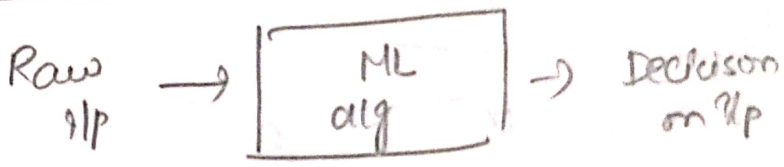
③ Manifold learning.

↳ Suppose taker an eg ⇒ differentiating dog & cat

ML Approach



DL Approach

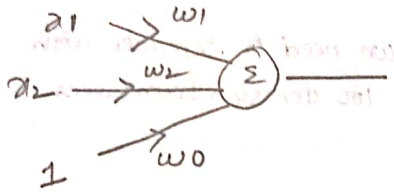


In traditional ML the features are decided by the user accordingly we need feature extraction algorithm. In case of deep learning we will not tell what features are to be analyzed, instead during the training phase the model learns features from the training samples that are useful in taking decisions.

⇒ During inference similar features are extracted by the model from the so far unseen data inorder to make a decision.

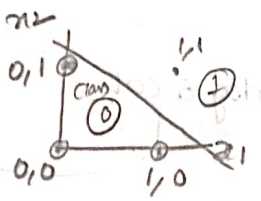
Perceptron :-

$y = w_1 x_1 + w_2 x_2 + w_0$



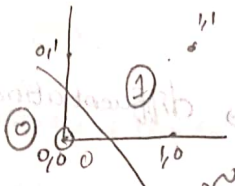
$$\vec{y} = \begin{cases} 1 & \text{if } w^T x > 0 \\ 0 & \text{if } w^T x < 0 \end{cases}$$

AND

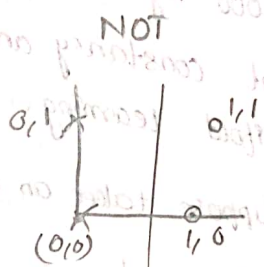


x_1	x_2	AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

OR



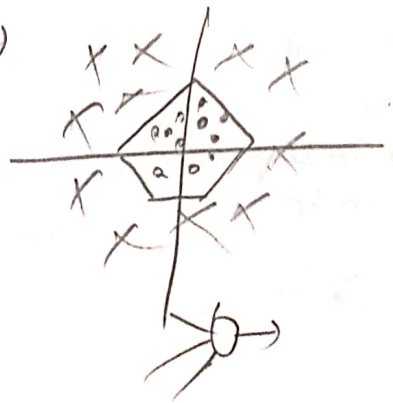
NOT



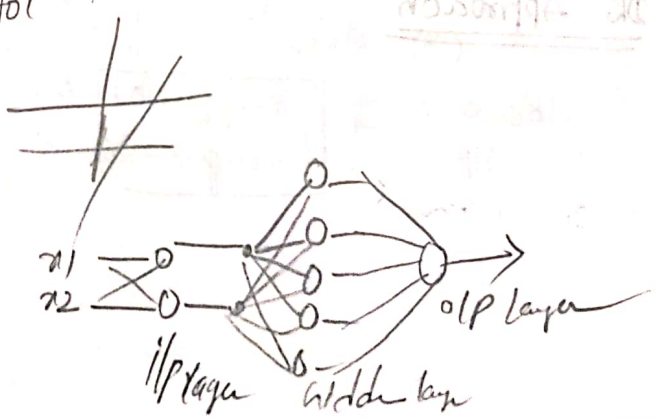
NOT	
1	0
0	1

1	
1	
0	
0	

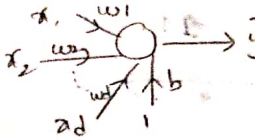
⇒



for



a/1/23. $y = w^T x$



⇒

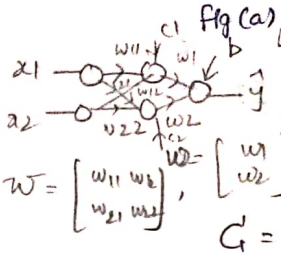


fig (a)

$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$, $W = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$

$G =$

we tried

i.e $W \in R^{2\times 2}$

$w \in R^{2\times 1}$

$G \in R^2$

$b \in R^1$

⇒ for above fig

More clearly,

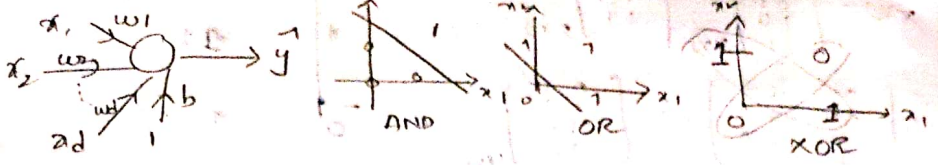
⇒ $\vec{y} = f^{(m)}(f$

⇒ for fig (a),

$W = [$

Feed forwa

9/1/23. $y = w^T x$



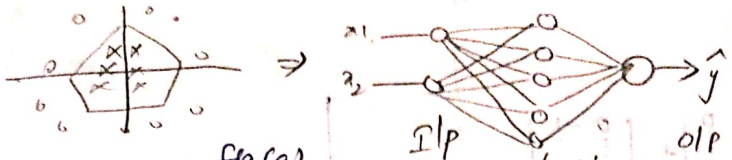
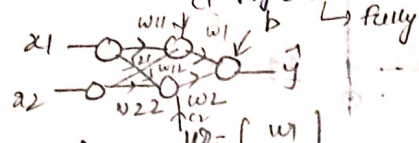


fig (a) ↳ fully connected hidden N/w



$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$, $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_{2\times1}$ $\quad y = f^*(x)$

$C = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}_{2\times1}$; $b_{1\times1}$ ↳ The fun representing XOR

↳ we want to learn this fun

we tried $\hat{y} = f(x; w, b)$ ↳ linear classifier ↳ this is failed to seperate XOR

i.e $W \in R^{2\times2}_{2\times1}$ where

$w \in R$ R is set of real nos

$C \in R^2$

$b \in R^1$

⇒ for above fig (a) ⇒ $\hat{y} = f^2(f(x)) = f(x)$ ↳ I/p layer

↳ fun for 1st layer (or) hidden layer

↳ second layer (or) O/p layer

More clearly, $\hat{y} = f(x; W, C, w, b)$

(or) ↳ The Parameters (or) Learnable or trainable parameters

⇒ $\hat{y} = f^{(m)}(f^{(m-1)}(\ldots(f^1(x))\ldots))$

⇒ for fig (a), $X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$ & $y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$ for XOR gate

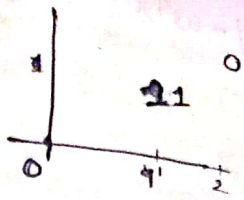
$W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ $w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$ $C = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$ $b = 0$

Feed forward Neural N/w

$$XW = \begin{bmatrix} 0&0\\0&1\\1&0\\1&1 \end{bmatrix}_{4\times2} \begin{bmatrix} 1&1\\1&1 \end{bmatrix}_{2\times2}$$

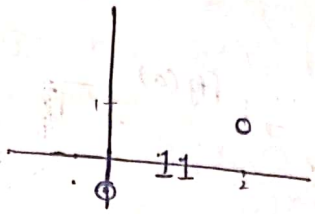
$$= \begin{bmatrix} 0+0 & 0+0\\ 0+1 & 0+1\\ 1+0 & 1+0\\ 1+1 & 1+1 \end{bmatrix} = \begin{bmatrix} 0&0\\1&1\\1&1\\2&2 \end{bmatrix}_{4\times2}$$

$$y = \begin{bmatrix} 0\\1\\1\\0 \end{bmatrix}$$



$$XW + C = \begin{bmatrix} 0&0\\1&1\\2&2 \end{bmatrix} + \begin{bmatrix} 0&-1\\0&-1\\0&-1 \end{bmatrix}$$

$$= \begin{bmatrix} 0&-1\\1&0\\2&1 \end{bmatrix} \quad \& \quad y = \begin{bmatrix} 0\\1\\0 \end{bmatrix}$$



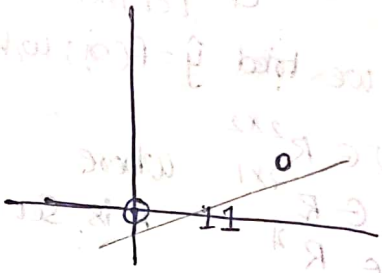
$w_1x_1 + w_2x_2 + \dots + w_0$

↳ is always linear

Consider $ReLU(XW+C)$

$ReLU(x) = \max(0, x)$

$$ReLU(XW+C) = \begin{bmatrix} 0&0\\1&0\\1&0\\2&1 \end{bmatrix} \& \; y = \begin{bmatrix} 0\\1\\1\\0 \end{bmatrix}$$



$ReLU(XW+C)wt\ b$

$$= [\ ] \begin{bmatrix} 0&0\\1&0\\1&0\\2&1 \end{bmatrix}_{4\times2} \begin{bmatrix} 1\\-2 \end{bmatrix}_{2\times1} + \begin{bmatrix} 0\\ \vdots \\ \end{bmatrix}_{4\times1}$$

$$= \begin{bmatrix} 0\\1\\1\\0 \end{bmatrix}_{4\times1} = \hat{y}$$

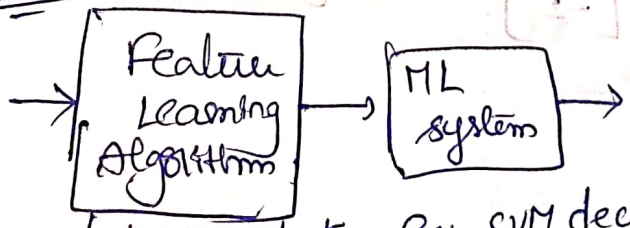
here $y = \hat{y}$, Error = 0

$\Rightarrow f(x) = f^2(f^1(x))$

↳ ~~Linear classifier~~ feature Learning layer

↳ ~~feature learning layers~~ Linear classifier

ML vs DL



↳ Designer has to decide which features to be considered & how to get them

Ex:- SVM, decision trees, Random forests

DL

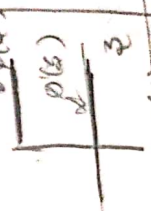
Feature Learning + classifier

18/01/23

Graphs

MLP

1D Graphs



Derivation

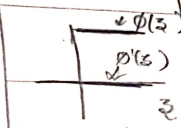
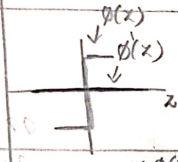
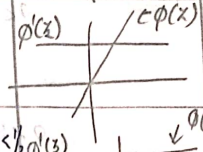
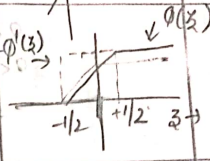
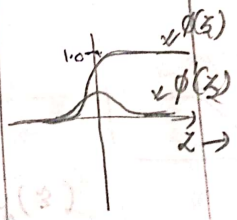
$\phi'(z) = 0$

Equation

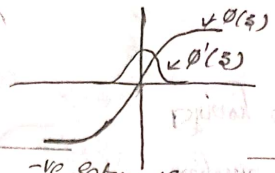
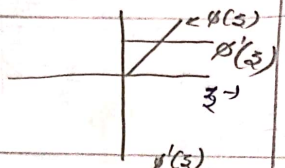
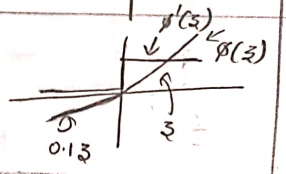
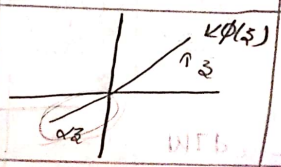
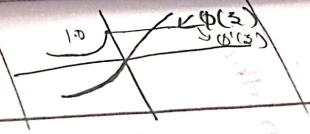
$\phi(z) = \begin{cases} 0 & z<0 \\ 0.5 & z=0 \end{cases}$

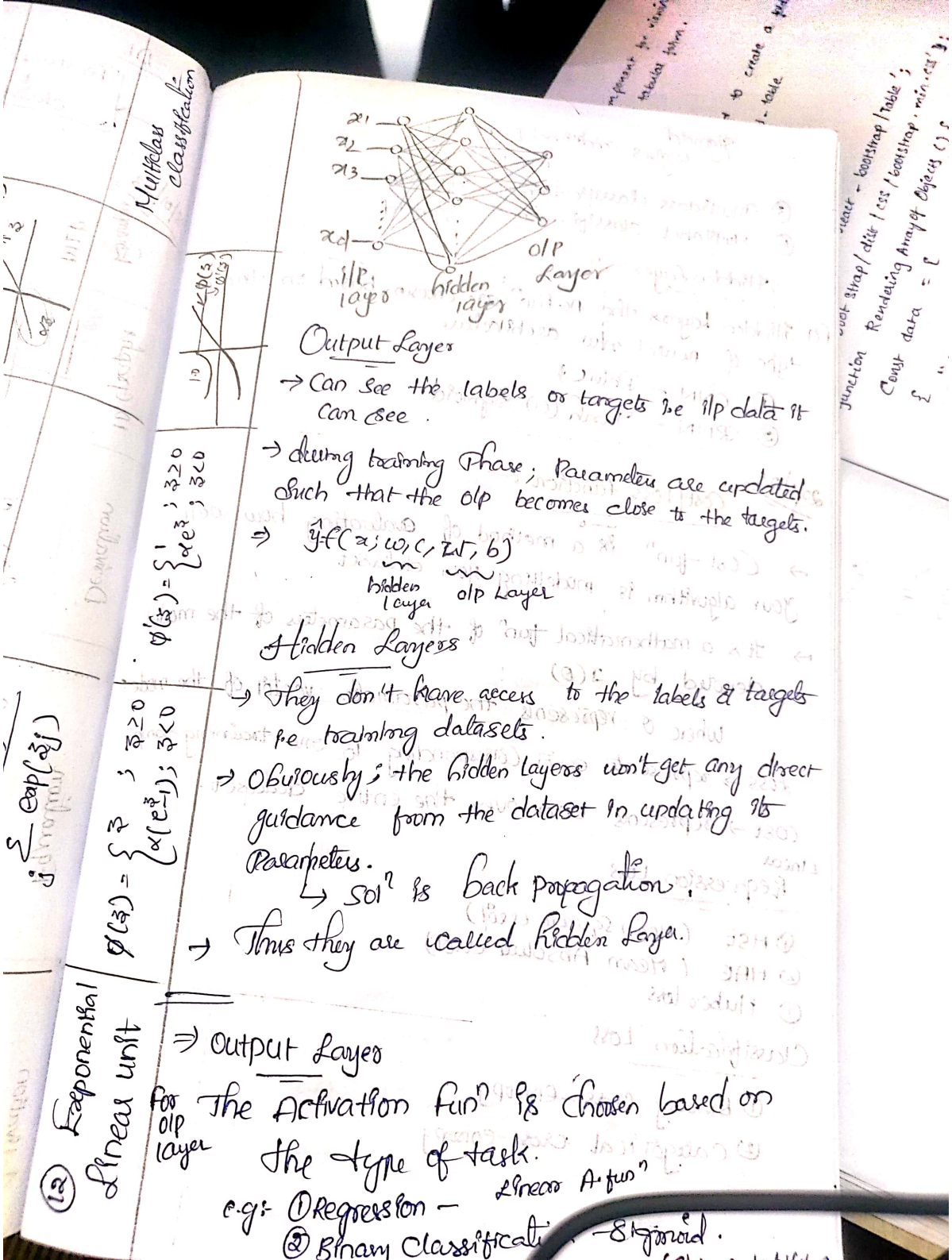
Activation Function

Binary step(θ)

Activation Function	Equation	Derivation	1D Graphs	Examples
① Binary step (or) Unit step (or) Heaviside	$\phi(z) = \begin{cases} 0 & z<0 \\ 0.5 & z=0 \\ 1 & z>0 \end{cases}$	$\phi'(z) = 0$		MLP
② Signum	$\phi(z) = \begin{cases} -1 & z<0 \\ 0 & z=0 \\ +1 & z>0 \end{cases}$	$\phi'(z) = 0$		MLP
③ Linear	$\phi(z) = z$	$\phi'(z) = 1$		
④ Piece wise Linear	$\phi(z) = \begin{cases} 1 & z \geq 1/2 \\ z+1/2 & -1/2 < z < 1/2 \\ 0 & z \leq -1/2 \end{cases}$	$\phi'(z) = \begin{cases} 1 & ; -\frac{1}{2} < z < \frac{1}{2} \phi'(z) \\ 0 & ; \text{Other wise} \end{cases}$		
⑤ Sigmoid $\sigma(z)$	$\phi(z) = \frac{1}{1+e^{-z}}$ $= \frac{e^z}{1+e^z}$	$\phi'(z) = \sigma(z)[1-\sigma(z)]$		Logistic Regression Binary classifica

DL — Feature learning + classifier

6) Tanh	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$\phi'(z) = 1 - \tanh^2(z)$	 -ve saturation
7) Softplus Note:— Sigmoid = differentiation of Softplus	$\phi(z) = \log(1+e^z)$	$\phi'(z) = \frac{e^z}{1+e^z}$	
8) Relu	$\phi(z) = \max(0, z)$ (or) $\phi(z) = \begin{cases} z & ; z \geq 0 \\ 0 & ; z < 0 \end{cases}$	$\phi'(z) = \begin{cases} 1 & ; z \geq 0 \\ 0 & ; z < 0 \end{cases}$	
9) Leaky Relu	$\phi(z) = \max(0.1z, z)$ (or) $\phi(z) = \begin{cases} z & ; z \geq 0 \\ 0.1z & ; z < 0 \end{cases}$	$\phi'(z) = \begin{cases} 1 & ; z \geq 0 \\ 0.1 & ; z < 0 \end{cases}$	
10) Parametric Relu	$\phi(z) = \max(\alpha z, z)$ (or) $\phi(z) = \begin{cases} z & ; z \geq 0 \\ \alpha & ; z < 0 \end{cases}$	$\phi'(z) = \begin{cases} 1 & ; z \geq 0 \\ \alpha & ; z < 0 \end{cases}$	
11) Softmax	$\phi(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$		Multiclass classification
12) Exponential linear unit	$\phi(z) = \begin{cases} z & ; z \geq 0 \\ \alpha(e^z - 1) & ; z < 0 \end{cases}$	$\phi'(z) = \begin{cases} 1 & ; z \geq 0 \\ \alpha e^z & ; z < 0 \end{cases}$	



Sigmoid
↳ gives conditional probability
$P(y=1|x)$

③ Multiclass classification - Softmax
④ Multilabel classification - Sigmoid.

Hidden Layer

For Hidden Layers the Ac. fun^n are choosen based on the type of neural n/w architecture

① CNN - Relu
② RNN - Tanh (or) Sigmoid

20/1/23.

Cost/Loss function :-

→ Cost fun^n is a method of evaluating how well your algorithm is modelling your dataset.

→ It's a mathematical fun^n of the parameters of the model denoted by $J(\theta)$

Where θ represents the parameter vector of the model

Loss → represents error corresponding to one training sample

Cost → represents error over the entire dataset.

Linear

Regression Loss

① MSE (Mean Square error)
② MAE (Mean Absolute error)
③ Huber loss

Classification Loss

① Binary Cross-entropy
② Categorical cross-entropy

Auto encoders

① KL divergence

GAN

① Discriminative
② Minmax GAN

Object Detection

① Focal loss

word Embeddings

① Triplet Los

→ Problem Setting

$x, y \sim P_{data}$

$\hat{y} \sim P_{model}$

$\hat{y} = f^*$

hidden

W, c

$\hat{y} =$

Auto encoders

① KL divergence Loss

GAN

① Discriminative Loss
② Minmax GAN Loss

Object Detection

① Focal Loss

word Embeddings

① Triplet Loss

→ Problem Setting

$x, y \sim P_{data}$ ⇒ The entire dataset coming from distribution P_{data}

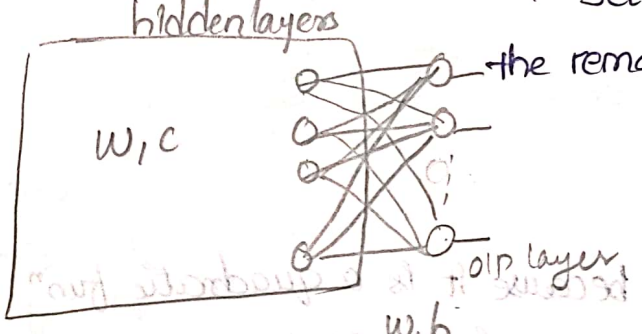
$\hat{y} \sim P_{model}$ ⇒ Predictions produced by the model according to its own learned distribution

P_{model} θ → complete set of model parameters.

$\hat{y} = f^*(x; \theta)$ θ → W, C, w, b

↳ set of parameters corresponding to olp layer

↳ set of parameters of all the remaining layers of DL model



$z = w^T h + b$

↳ feature vector from the last hidden layer.

↳ Linear olp of the olp layer.

$\hat{y} = \phi(z) = \phi(w^T h + b)$

↳ Activation fun^n

This decides what is the type of olp units.

e.g:- If you choose ϕ to be Sigmoid then the olp unit is said to be of sigmoid type.

Goal → prediction → target
→ we want $\hat{y}_i$ as close to y_i as possible $\forall i$ (training samples $i = 1, 2, 3, 4, \ldots N$)

→ If they are not equal, then there is error given by $y_i - \hat{y}_i$ < +ve / -ve

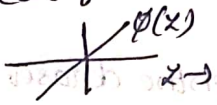
→ we define cost fun^n $J(\theta)$ which is a function of model parameters & we try to update θ such that $J(\theta)$ gets minimized.

Linear Regression Loss

which $\phi(x)$ is to be used ?

$\phi(x)$ must be linear fun^n

$\phi(x) = x$



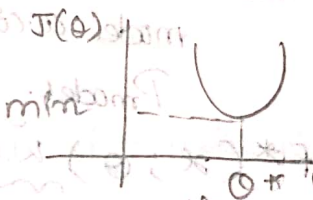
S.No	features x_1 x_2 ... x_d	y	$\hat{y}$
1			
2			
.			
N			

→ $\hat{y} = \phi(w^T h + b) = w^T h + b$

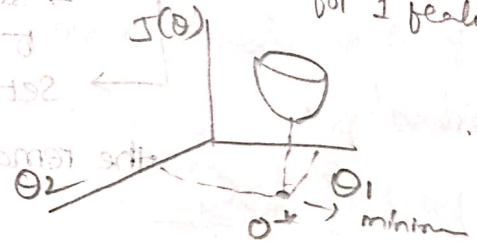
① MSE cost fun^n

$$J(\theta)_{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

$$\theta^* = \arg\min_{\theta} J(\theta)$$



for 1 feature



Advantages

① Easy to Interpret

② Always differentiable because it is a quadratic fun^n

③ Only one local ~~the~~ minimum is present

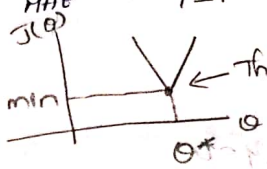
Disadvantages

① MSE is not robust to outliers,

② MAE

$\hat{y} = \phi(w^T h + b) =$

$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} | y_i -$



Advantage

① Intuitive & easy

② Robust to outl

Disadvantages

① Not differentia
used directly
algorithm.

③ Huber Loss

$J_{Huber}(\theta) =$

δ - hyper pa
The hyper pa
the transition
to linear.

NOTE :- It

① It

② It

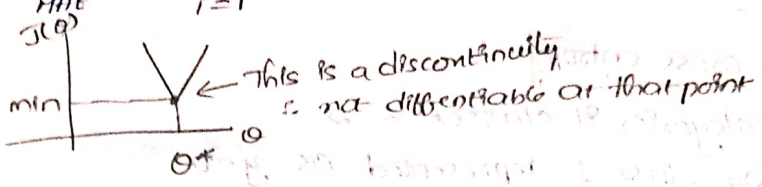
①

we also nee
requirement

② MAE

$\hat{y} = \phi(w^T h + b) = w^T h + b$

$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$ (Error is +ve)



Advantage

① Intuitive & easy to implement

② Robust to outliers

Disadvantages

① Not differentiable so gradient descent cannot be used directly but we can use sub gradient descent algorithm.

③ Huber Loss

$$J_{Huber}(\theta) = \begin{cases} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2}(y_i - \hat{y})^2 & \text{if } \delta |y_i - \hat{y}_i| \le \delta \\ \frac{1}{N} \sum_{i=1}^{N} \delta(|y_i - \hat{y}_i| - \frac{1}{2}\delta) & \text{if } |y_i - \hat{y}_i| > \delta \end{cases}$$

(quadratic; linear)

δ - hyper parameter

The hyperparameter (δ) defines the point where the transition cost fun $J(\theta)$ transitions from quadratic to linear.

NOTE :- It's Performance Adv

① It's robust to outliers.

② It's Performance lies blw MAE & MSE

Disadv

① Associated Complexity. The hyperparameter(δ) will also needed to be optimized which is an additional requirement on the training algorithm.

Note :- For linear regression the o/p layer contains only one o/p unit and with linear Activation.

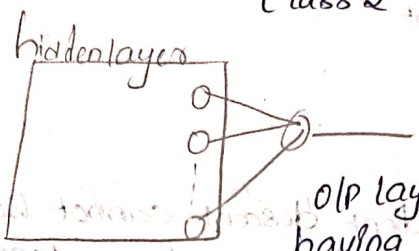
Classification

① Binary - cross entropy

No. of categories & classes = 2

Let us say class 1 represented as $y=0$

class 2 represented as $y=1$.

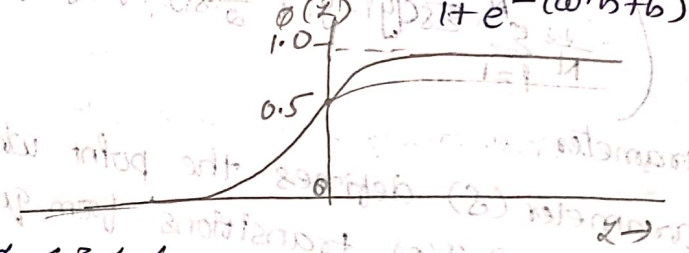


o/p layer having only one o/p unit with Sigmoid fun^

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

$z = w^T h + b$

$\hat{y} = \sigma(z) = \frac{1}{1+e^{-(w^T h + b)}}$



① $-\infty < z < \infty$

② min value of $\sigma(z) = 0$

③ max value of $\sigma(z) = 1$

④ $\therefore 0 \le \hat{y} \le 1$ ⇒ can be interpreted as probability

$\sigma(z)$ is converting the 'z' into probability.

⇒ we know that y has one of the two possible values

$y=0$ or $y=1$

$y=0$ if $x \in$ class 1

$y=1$ if $x \in$ class 2

⇒ But $\hat{y}$ can take any ↳ represents

⇒ $\hat{y} = P_{model}(y=1|x)$

Now →

i.e $\hat{y} = P_{model}(y=1$

If $P_{model}(y=1|x) =$

If $P_{model}(y=1|x) =$

$$J(\theta) = -\sum_{i=1}^{N} y_i \log$$

Case 1 :- Assume

$y_i = 1$ and

Loss = $-1 \cdot \log$

Case 2 :- $y_i = 0$ and

Loss =

Case 3 :- $\hat{y}_i$ is

respectively

⇒ Then there be) and adding us $J(\theta)$.

Advantages

The cost fun^

Disadvantages

① The loss

② This is

⇒ But $\hat{y}$ can take any real no. in blw 0 & 1 (including)
↳ represents the probability

⇒ $\hat{y} = P_{model}(y=1|x)$ (or) $\hat{y} = P_{model}(y=0|x)$

Now →
i.e $\hat{y} = P_{model}(y=1|x)$ | 0 0 0 → $\hat{y} = P_{model}(y=1|x)$

If $P_{model}(y=1|x) = \hat{y} > 0.5$ then $x \in$ class 2
If $P_{model}(y=0|x) = \hat{y} < 0.5$ then $x \in$ class 1

$$J(\theta) = -\sum_{i=1}^{N} y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i)$$

Case 1 :- Assuming, Suppose Accuracy = 1 (100%) $\quad$ Loss $= -y_i \log \hat{y}_i - (1-y_i)\log(1-\hat{y}_i)$

$y_i = 1$ and the model prediction also $\hat{y}_i = 1$

$$\text{Loss} = -1 \cdot \log 1 - (1-1)\log(1-1) = 0$$

Case 2 :- $y_i = 0$ and $\hat{y}_i = 0$

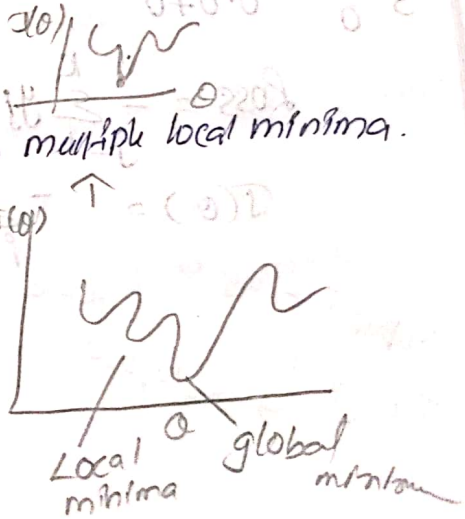
$$\text{Loss} = 0 \cdot \log 0 - (1-0)\log(1-0) = 0$$

Case 3 :- $\hat{y}_i$ is other than 0 (or) 1 when y_i is 0 (or) 1 respectively. Example : $\hat{y}_i = 0.21, 0.47, 0.78, 0.68 \ldots$

⇒ Then there will be non-zero loss (that leads to be) and adding losses corresponding to all training samples gives us $J(\theta)$.

Advantages
The cost fun is differentiable

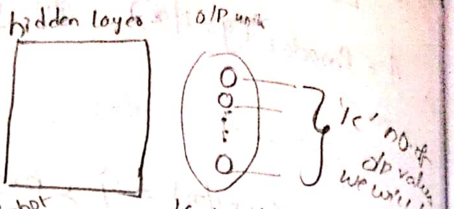
Disadvantage
① The loss landscape will have multiple local minima.
② This is not intuitive



② Multiclass classification loss

\# Catagories = # classes = k

Class 1 $y=0$
Class 2 $y=1$
⋮
Class k $y=k-1$ ← integer encoder



← one-hot encoder

S.No		y	$\hat{y}$
1		2	[, ,] $_{1 \times k}$
2		4	[, , , ,] $_{1 \times k}$

Softmax $\phi(z)_i = \dfrac{\phi(z_i)}{\sum_{j=1}^{k} \phi(z_j)}$

e.g.- Iris dataset

e.g.

$k=3$ integer Encoder One-hot Encoder

	integer	one-hot
Setosa	0	[1,0,0]
versinia	1	[0,1,0]
versicolor	2	[0,0,1]

I	y_i	$\hat{y}_i$
1	1	0.775
2	0	0.155
3	0	0.070

← Loss = 0.3675 → Model accuracy = 63.25 %

after for epochs

$\hat{y}_i$
0.938
0.028
0.036

Loss = 0.095 ⇓ Accuracy = 99.05%

$$Loss = -\sum_{j=1}^{k} y_j \log \hat{y}_j$$

$$J(\theta) = -\sum_{i=1}^{N} \sum_{j=1}^{k} y_{ij} \log \hat{y}_{ij}$$

MNIST - Digit Recognize $k=10$

Image-net ↳ $k=10{,}000$

Sequential model
compile
summary
plot_model

fit
predict
evaluate

2/1/23

Circularity = $\frac{4\pi A}{P^2}$

S.NO	Circularity	shape
	0.3	

P(x) = probability distrib...

P(y = Triangle) = 0.5
P(y = Rect) = 0.5 } a-...

P(y = T|x) =
P(y = R|x) = }

1.0

P(x|y=T) → P(x) giv...
P(x|y=R) → Cond...

P(y=T|x) = P(x|...

[P(y=T|x) P(x) = ...

P(y=R|x) = P(x|...

P(y=T|x) > ...
< ...

⇒ P(x|y=T) P(y...

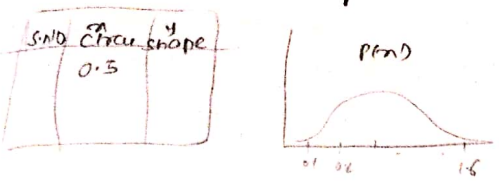
P(x|y=...

21/1/23

Suppose

$$\text{Circularity} = \frac{4\pi A}{P^2} = 1$$

A = Area
P = Perimeter

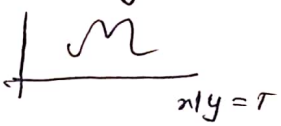


$P(x)$ = probability distribution of actual measurements

$P(y = \text{Triangle}) = 0.5$
$P(y = \text{Rect}) = 0.5$
} a-priori probabilities

$P(y = T|x) =$
$P(y = R|x) =$
} a-posterior probabilities
1.0

↳ we want these probabilities
↳ But very different to get



- $P(x|y=T)$ → $P(x)$ given that $y=T$
- $P(x|y=R)$ → Conditional probability distribution

$$P(y=T|x) = \frac{P(x|y=T)\,P(y=T)}{P(x)}$$ [Bayes theorem]

$$[P(y=T|x)\,P(x) = P(x|y=T)\,P(y=T)]$$

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

$P(y=T|x) > P(y=R|x) \Rightarrow$ Triangle
$<$ $\Rightarrow$ Rectangle.

→ In Bayes theorem all =

$\Rightarrow P(x|y=T)\,P(y=T) > P(x|y=R)\,P(y=R)$ (if same)

$P(x|y=T) > P(x|y=R) \Rightarrow$ Triangle
$<$ $\Rightarrow$ Rectangle