Date: 23/1/24

DLP

=> Machine learning ~~[illegible]~~

→ Supervised : model training with labelled data

eg: Regression (LR), classification (KNN)

→ Unsupervised : model training with unlabelled data

eg: clustering (Kmeans)

→ Reinforcement : Model takes actions in the enviroment then received state updates and feedbacks

* if feature extraction (data preprocessing) is done by humans → it is Machine learning

- Feature extraction + model = Deep learning

↳ prediction on clean data

DL does feature engineering + model training

=> Applications of DL :

- GAN → deep fake, medical domain, receptionist avatar
  ↳ generating medical related images
- Vision Transformers → chatGPT
- CNN → news
- Face detection Vs Face Recognition
  ↳ either person or no ↳ specific person

Date:

- Anamoly detection → surveillance videos
- RNN → predicts next word like in keyboard

→ Shallow vs Deep Neural Network :

Shallow: have only one or two hidden layers

Deep: have dozens or hundreds of hidden layers → captures more complex patterns in the input data

→ Hyperparameters vs parameters :

Hyperparameters:
- allow the model to learn the rules from the data
- estimated during training with historical data
- eg: # of layers, depth, iterations, epoch, neuron in each layer

Parameters:
- control how the model is training
- values are set beforehand
- eg: w, b

→ Epoch :

Divide data into batches, each batch could be run a certain no. of times

Date:

→ Cross validation : Divide data into ratios
no. of pairs we take might
be biased

* K-fold cross validation → dataset is divided into k subsets/folds, model is evaluated and trained k time using a different fold as the validation set each time.

→ Overfitting vs Underfitting

↓ ↓

- good accuracy on training data but bad on testing
- variance in data ↑

- bad accuracy on both training and testing data
- biasness ↑

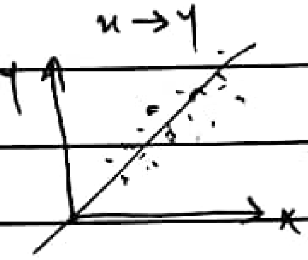
Train	Test	
0.9	0.4	→ overfitting
0.3	0.3	→ underfitting → parameters are not properly tuned

Date:

⇒ Linear Regression :

$x \rightarrow$ independent variable

$y \rightarrow$ dependent variable



$y = mx + c$

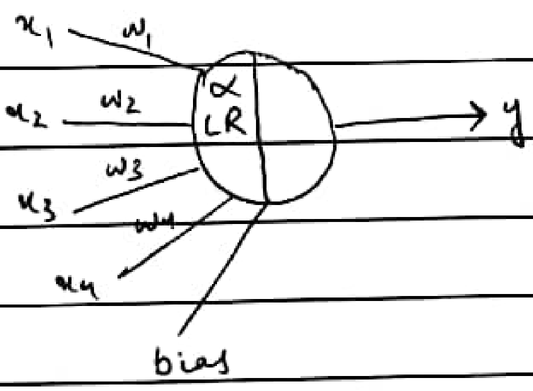
$y = b_1 x + b_0$ | $y = w_1 x + w_0$

Simple LR : $\hat{y} = bx + a$

Multiple LR : $\hat{y} = b_1 x_1 + b_2 x_2 + \dots + b_k x_k + a$

* Objective is to find weights

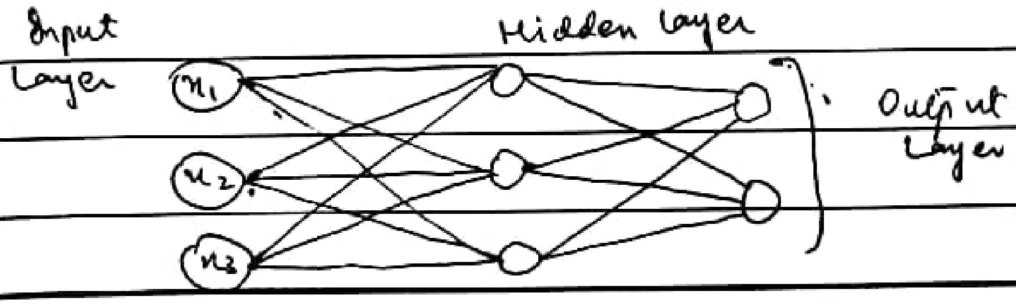
⇒ The Perceptron : Forward Propagation



$y = w_0 + \sum_{i=1}^{n} w_i x_i$

LR equation

* range for $y, x, w \rightarrow -\infty$ to $+\infty$



Date:

* No cyclic graph, moving forward

How many hidden layers?

↳ could vary based on kispe result accha araha hai

* weights tuning in back propagation — to minimize error by optimizing weights

parameters are tuned in back propagation

↳ weights, bias

→ ReLU, Sigmoid, Softmax, Tanh

→ Activation Functions — adds non-linearity, minimizes ranges

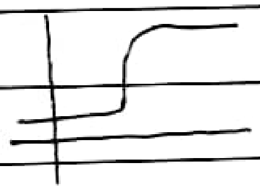
↳ $-\infty$ to $+\infty$

$\Downarrow$

0 to 1

A neuron is LR + Activation Function

• Sigmoid



$$\text{Sigmoid} = \frac{1}{1+e^{-z}}$$

$$\frac{1}{1+e^{-(-\infty)}} = \frac{1}{1+e^{\infty}} = \frac{1}{\infty} = 0$$

$$\frac{1}{1+e^{-(\infty)}} = \frac{1}{1+e^{-\infty}} = \frac{1}{1+\cancelto{0}{\frac{1}{e^{\infty}}}} = 1$$

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

$$g'(z) = g(z)(1-g(z))$$

CamScanner

Date:

- Hyperbolic Tangent

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

$$g'(z) = 1 - g(z)^2$$

- ReLU

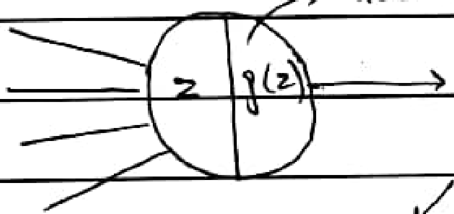
$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & x > 0 \\ 0, & \text{otherwise} \end{cases}$$

LOGISTIC REGRESSION

- Single neuron is equivalent to Logistic Regression
- Classes can only be two
- Linear Regression + Activation Function = Logistic Regression

activation function (sigmoid)



$$z = w_0 + \sum_{i=1}^{n} w_i x_i \quad | \quad wx + b$$

middle variable

$$\hat{y} = g(z)$$

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

- values in terms of logs
- probability of event occuring vs the ratio of not occuring

$$\ln\left(\frac{p}{1-p}\right)$$

↳ odds

- Instead of modeling the outcome, Y, directly, the method models the log odds (Y) using the logistic function

Date:

$$\text{LOGIT}(p) = \ln\left(\frac{p}{1-p}\right) = z \iff p = \frac{\exp(z)}{1+\exp(z)}$$

- Sigmoid ⇒ maps $-\infty/+\infty$ to 0/1
- Hyperbolic Tangent ⇒ maps variables to -1 to $+1$
- ReLU ⇒ negative values are clipped off
  positive values remain same
- activation function adds non-linearity (curvy line)

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

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

$$P(y=1) = \sigma(wx+b) = \frac{1}{1+e^{-(wx+b)}}$$

} probability of event occuring (class 1)

$$P(y=0) = 1-\sigma(wx+b) = 1-\frac{1}{1+e^{-(wx+b)}} = \frac{e^{-(wx+b)}}{1+e^{-(wx+b)}}$$

} probability of event not occuring (class 2)

mostly not used

Date:

⇒ Learning in Logistic Regression :

1) Loss Function / cost Function : how close the current label ($\hat{y}$) is to the true label y.

2) Optimization algorithm : for iteratively updating weights

⇒ Loss / Cost Function

Cross entropy loss

$L(\hat{y}, y)$

predicted ↙ ↘ actual

maximum likelihood

↳ maximize probability

- Conditional max likelihood estimation prefers the correct class labels of the training examples to be more likely

- We choose w, b that maximize the log probability of the true y labels in the training data given the observations x.

→ Max Likelihood Estimation

- For samples labelled as 1 : $\prod_{s \text{ in } y_i = 1} \hat{y}$

↳ $\hat{y}$ is as close to 1 as possible

Date:

- For samples labelled as 0 : $\prod_{i \text{ in } y_i=0} (1-\hat{y})$
  ↳ $(1-\hat{y})$ should be as close to 1 as possible

* On combining both, (w and b) should be such that the product of both of these products in max over all the elements of the dataset.

$$L(\beta) = \prod_{i \text{ in } y_i=1} \hat{y} * \prod_{i \text{ in } y_i=0} (1-\hat{y})$$

This function is the one we need to optimize and is called the 'likelihood function'

→ Loss Function :

if $y=1$: $p(y|x) = \hat{y}$
if $y=0$: $p(y|x) = 1-\hat{y}$

$$P(y|x) = \hat{y}^{y}(1-\hat{y})^{(1-y)}$$

→ Log likelihood :

this is a Bernoulli distribution since only two outcomes (0,1)

$$P(y|x) = \hat{y}^{y}(1-\hat{y})^{1-y}$$

Date:

$$\log p(y|x) = \log\left[\hat{y}^{y}(1-\hat{y})^{1-y}\right]$$
$$= y\log\hat{y} + (1-y)\log(1-\hat{y})$$

whatever values maximize a prob will also maximize the log of the probability

eg:

For class 1:

$$p(y|x) = \hat{y}^{y}(1-\hat{y})^{1-y}$$
$$= (0.7)^{1}(1-0.7)^{1-1}$$

For class 0:

$$p(y|x) = (0.7)^{0}(1-0.7)^{1-0}$$

- likelihood should be maximized
- loss should be minimized

→ Cross Entropy:

negative log likelihood loss = cross-entropy loss

$$L_{CE}(\hat{y}, y) = -\log p(y|x) = -\left[y\log\hat{y} + (1-y)\log(1-\hat{y})\right]$$

$$L_{CE}(w, b) = -\left[y\log\sigma(wx+b) + (1-y)\log(1-\sigma(wx+b))\right]$$

CamScanner

Date:

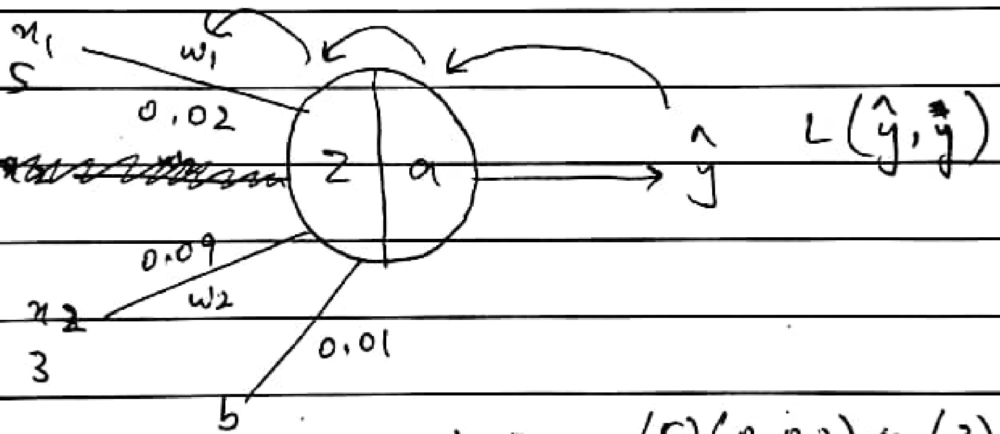
→ Binary cross Entropy loss:

↳ working on two classes

↳ calculated as the average ~~ss~~ cross-entropy across all data examples

$$L = -\frac{1}{N}\left[\sum_{j=1}^{N}\left[y\log\hat{y} + (1-y)\log(1-\hat{y})\right]\right]$$

⇒



$$\rightarrow z = (5)(0.02) + (3)(0.09) + 0.01$$
$$= 0.38$$
$$\hat{y} = a = \frac{1}{1+e^{-0.38}} = 0.5939$$

→ Cross entropy loss:

$$L(\hat{y}, y) = -y \ln(\overbrace{\sigma(wx+b)}^{a})$$
$$= -y \ln(0.5939)$$
$$= 0.5210$$

$$* \log a = \frac{1}{a}$$

Date:

→ Backpropogation :

$$\frac{dL}{dw_1} = \frac{dL}{da} \times \frac{da}{dz} \times \frac{dz}{dw_1}$$

$$L = \left[-y \log(\hat{y}) - (1-y)\log(1-\hat{y})\right]$$

$$= \left[-y \log a - (1-y)\log(1-a)\right]$$

$$\frac{dL}{da} = -y \cdot \frac{1}{a} - (1-y) \cdot \frac{1}{a} \cdot (-1)$$

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

$$a = \frac{1}{1+e^{-z}} \qquad \frac{da}{dz} = a(1-a)$$

$$z = w_1 x_1 + w_2 x_2 + b$$

$$\frac{dz}{dw_1} = x_1$$

$$\frac{dL}{dw_1} = \left[\frac{-y}{a} + \frac{(1-y)}{(1-a)}\right] \times \left[a(1-a)\right] \times \left[x_1\right]$$

$$= \frac{(1-a)(-y) + a(1-y)}{\cancel{a(1-a)}} \times a\cancel{(1-a)} \times x_1$$

Date:

$= (-y + \cancel{ay} + a - \cancel{ay})\, x_1$

$= (-y + a)\, x_1$

$a \Rightarrow$ predicted

$y \Rightarrow$ actual

$$\frac{\partial L}{\partial w_1} = a x_1 - y x_1 \;/\; (a-y)\, x_1$$

$$\frac{\partial L}{\partial w_2} = (a-y)\, x_2$$

$$\frac{\partial L}{\partial b} = (a-y)$$

$$\frac{\partial L}{\partial w_1} = (0.5939 - 1)(5) = -2.03$$

$$\frac{\partial L}{\partial w_2} = (0.5939 - 1)(3) = -1.218$$

$$\frac{\partial L}{\partial b} = (0.5939 - 1) = -0.4061$$

$\Rightarrow$ update weights

learning rate

$$w_1^{new} = w_1^{old} - \eta \frac{\partial L}{\partial w_1}$$

$\eta = 0.005$

$$w_1^{new} = 0.02 - 0.005(-2.03)$$

$$= 0.03$$

slides 25 —

Date:

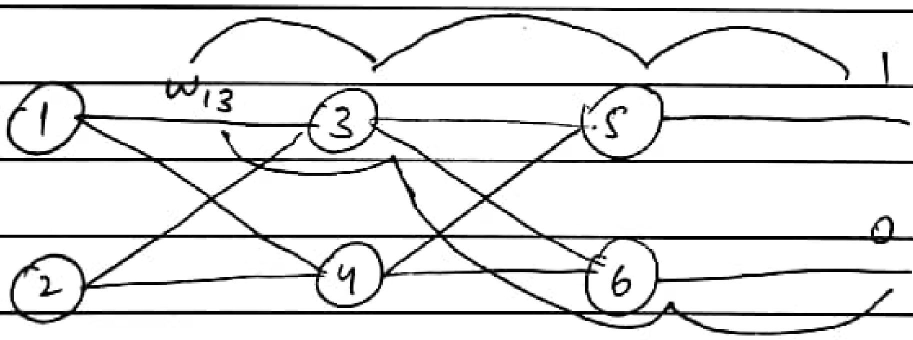
- for next iteration, use new w_1 as old w_1
- in each iteration, loss will be reduced

$$\frac{\partial L}{\partial w} = 0 \rightarrow \text{gradient descent} \; \text{slope zero}$$

NEURAL NETWORKS

— from slides

⇒



y_1 (actual) maximize

A — 1 0

y_2 (actual) minimize

B — 0 1

$$\frac{\partial L}{\partial w_{13}} = \boxed{\frac{\partial L_1}{\partial a_5}} \times \frac{\partial a_5}{\partial z_5} \times \frac{\partial z_5}{\partial a_3} \times \frac{\partial a_3}{\partial z_3} \times \frac{\partial z_3}{\partial w_{13}}$$

Loss

$$+ \boxed{\frac{\partial L_2}{\partial a_6}} \times \frac{\partial a_6}{\partial z_6} \times \frac{\partial z_6}{\partial a_3} \times \frac{\partial a_3}{\partial z_3} \times \frac{\partial z_3}{\partial w_{13}}$$

One-hot encoding → 4, 7, 2 — A (1, 0)

5, 6, 3 — B (0, 1)

Date:

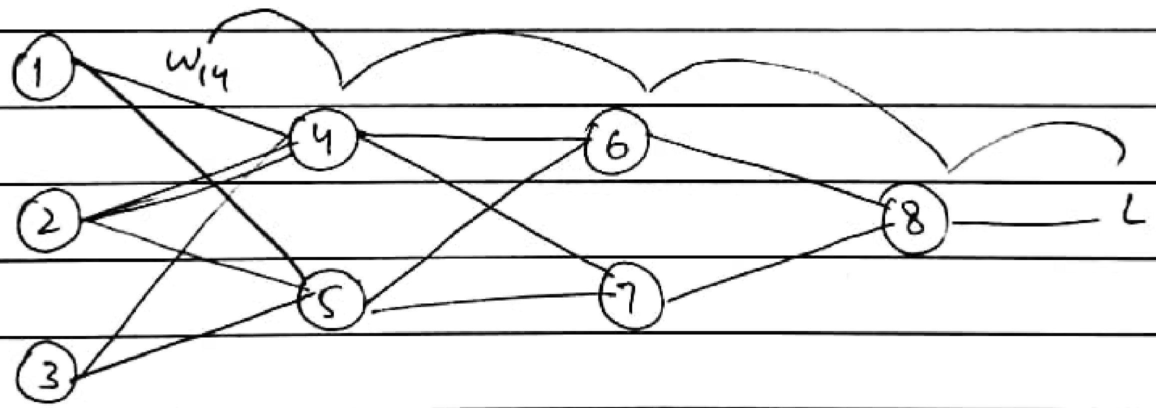
$$L = \frac{1}{2}(y-a)^2$$

$$= \frac{1}{2}(y_1 - a_5)^2 + \frac{1}{2}(y_2 - a_6)^2$$

$$= -(y_1 - a_5) \cdot a_5(1-a_5)a_3 \cdot a_3(1-a_3)x_1$$
$$-(y_2 - a_6) \cdot a_6(1-a_6)a_3 \cdot a_3(1-a_3)x_1$$

$$\hookrightarrow \frac{\partial L}{\partial a_6}$$

$\Rightarrow$



$$\frac{\partial L}{\partial w_{14}} = \frac{\partial L}{\partial a_8} \times \frac{\partial a_8}{\partial z_8} \cdots \frac{\partial a_4}{\partial z_4} \times \frac{\partial z_4}{\partial w_{14}}$$

Date: REGULARIZATION

overfitting — good accuracy on training, bad on testing
underfitting — bad accuracy on both

- NN are assumed to be overfitted models by default
- they are too complex
- training samples ko itne acchay se learn karlete hain k wo unseen data pe perform nahi karpata

$w_1 x_1 + w_2 x_2$
$0.1 x_1 + 0.3 x_2$

more dominantly affecting

there are so many variables that we cannot know their importance

⇒ Regularization makes slight modifications to the learning algorithm such that the model generalizes better and in turn improves the model's performance on the unseen data as well

- in ML, regularization penalizes the coefficients
- in DL, it penalizes the weight matrices of the nodes

Date:

- assume that the regularization coefficient is so high that some of the weight matrices are nearly equal to zero
- this will result in a much simpler linear network and slight underfitting of the training data
- a large value of the regularization coefficient is not that useful — we need to optimize the value of the coefficient to obtain a well-fitted model

Training, testing pe sahi kaam karay — appropriate fitting

⇒ L1 L2 Regularization :

L1 :

$$\text{Modified Loss Function} = \text{Loss Function} + \lambda \sum_{i=1}^{n} |w_i|$$

L2 :

$$\text{Modified Loss Function} = \text{Loss Function} + \lambda \sum_{i=1}^{n} w_i^2$$

$0.001 [w_{14} + w_{15} + \dots]$

when overfitting, weights values are higher like 45, 30..

↳ penalty to keep weights in range

Date:

⇒ Dropout :

- randomly skip some neurons
- random sampling of neurons
- har layer pe probability set karsakte hain ki us layer pe kitne neurons chahiye
- testing time pe neurons are set as it is

eg: layer pe 50% ki prob set ki k us layer k 50% neurons chahiye

- Back propagation is done on reduced network
- clipped off weights are not updated in B.P
- Different no. of weights are updated on each iteration
- Full network on testing
- Random selection of neurons on each iteration
- Overall accuracy is improved — overfitting pe nahi jaata

→ Method 1 :

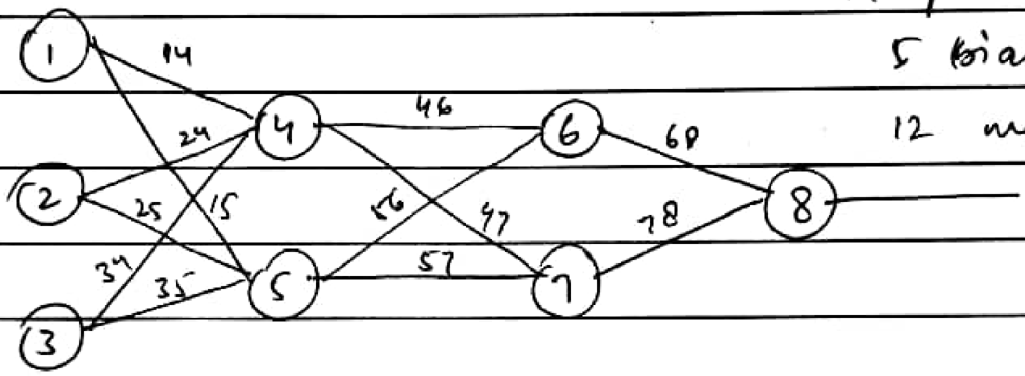
- Each unit is retained with a prob p. (train)
- weights multiplied with p (test)

→ Method 2 :

- weights multiplied with $\frac{1}{p}$ (train)
- no scaling (test)

Date:

- Dropout main poora neuron delete hota hai — uske ingoing and outgoing connections bhi



17 parameters
5 biases
12 weights

remove 5 — new parameters =

- On which layer to apply dropout?
  ↳ could be different combinations

CNN
  ↳ sharp edges, corners

- ideally $p = 0.5$
- 100 samples pe dropout ho tou error ziada ayega
  ↳ less data — model couldn't learn
- merge methods like dropout + L2