UNIT-II RECOMMENDED SYSTEM

RECOMMENDED SYSTEM:- Recommender system is an information-filtering technique that provides user with recommendation for which they might be interested in.

↳ Recommender system acts as a solution for your day-to-day choices.

For eg- which websites will you find interesting.
- which degree and university are best for your future.
- which is the best investment for supporting the education of your children.
- which digital camera should you buy.
- " is the best holiday for you and your family
- which trending course will be helpful for your future.

- "RS are important class of ML algorithm that offers relevant suggestions to users."

Features - Recommender System recommends items that are most popular among all users.

- They Divide users into multiple segment based on their preferences and recommends items to them based on the segment they belongs to.

Purpose of Recommender System -

1> Predictive Perspective :-
- They predicts to what degree a user likes an item.
- Most popular scenerio in research.

2) Interaction Perspective -
- Educate users about the complete product domain.

3). Retrieval Perspective -
- Reduces search costs.
- provides correct proposals.

④ Recommendation Perspective -
- identifies items from a long tail.

Recommender System reduces the information overload by estimating relevance.

collaborative Filtering Recommendation:- Recommends what's popular among your peers.

- it uses the rating data to calculate the similarity between items.
- it does not consider the semantic relationship between different items.

collaborative method work with interaction matrix that can also be called as rating matrix. Here the task of RC Algo^m is to learn a function that predicts the utility of items to each user.

Here set of users U and set of items I to be recom. for the users.

	DDLJ	DCH	Idiot
John	5	1	3
Tom	?	?	?
Alia	4	?	5

learn a func based on the past data that predicts the utility of each item $i \in I$ to each user $u \in U$.

- If uses User behaviour to predict the recommended items.

content-based recommendation:- Display products that are similar to the product you have liked before. ex:

↳ it works with the data provided by user. either explicitly through rating or implicitly (by clicking on a link).

↳ It generate the data - user profile.

LECTURE NOTES

Campus: PCE Course: BTECH in CSE Class/Section: III Yr. Section- A Date: .....

Name of Faculty: Praveen Kumar Yadav Name of Subject: Machine Learning Code: 6CS4-02

Date (Prep.): ..... Date (Del.): .....Unit No.:.....Lect. No: .....

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc., which will be taught in this lecture)

CONTENT BASED RECOMMENDED SYSTEM.

IMPORTANT & RELEVANT QUESTIONS:

FEED BACK QUESTIONS (AFTER 20 MINUTES):

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:

श्री गणेशाय नमः

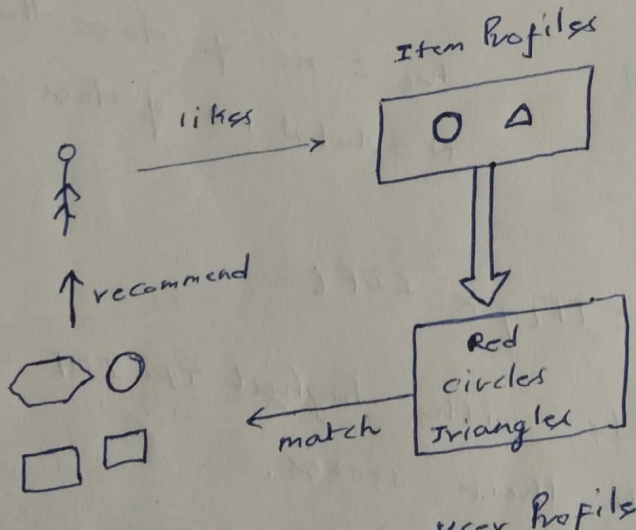
Content-Based Recommendations System:-

Main-Idea- To Recommend items to customer x similar to previous items rated highly by x.

eg- Movies -
same Actor, director, genre

website, blogs, news
- Articles with "similar" content.

People
- Recommends people with many common friends.



(Item Profiles). Here for each item, create an item profile

- Profile is a set of features.
  - Movies - author, title, actor, director.
  - Images, videos - meta data and tags.
  - People - set of friends.
- convenient to think of the item profile as a vector.
  - one entry per feature
  - vector might be boolean or real valued.

Text-Features- Profile - set of "important" words in item (document).

TF (Term Freq vs Inverse document Freq):-

f_{ij} = freq of an feature i in a document j.

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$ (feature i appearing in other k^{th} documents)

w_i = no of docs that mention term i

N = total NO of docs

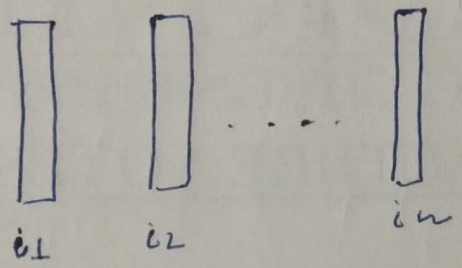
$$IDF_i = \log N/w_i$$

TF-IDF score $w_{ij} = TF_{ij} * IDF_i$

Doc profile: set of words with highest TF-IDF scores, together with their scores.

User - Profiles -

Here User has rated items with profiles $i_1 \ldots i_n$.



simple - (weighted) average of rated item profiles

variant - Normalize weights using average rating of a user.

eg-1. Boolean Utility Matrix

eg items are movies, only feature is "actor"

- item profile - vector with 0 or 1 for each vector.

5 movies - 2 Movies - actor A

3 " - " B

- User Profile = mean of item profiles

Feature 'A' weight = 2/5 = 0.4

Feature 'B' " = 3/5 = 0.6

e.g. 2. star Rating -

1 - 5 star ratings

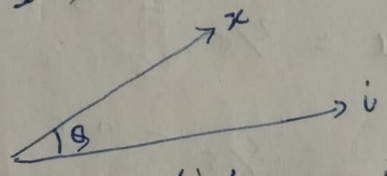
- Actor A' movies rated 3 and 5 ← higher rating
- Actor B " " 1, 2 and 4.
  lower rating (-ve ")

Useful step - Normalize rating by subtracting user's mean rating (3)

- Actor A's normalized rating = 0, +2
  Profile weight = (0+2)/2 = 1
- Actor B's normalized rating = -2, -1 + 1
  Profile weight = -2/3

Making Predictions - User Profile x, item profile i

Estimate $U(x, i) = \cos(\theta) = x \cdot i / (|x||i|)$



Technically, the cosine distance is actually the angle θ and the cosine similarity is the angle 180-θ. We use cos(θ) as our cosine similarity measure.

if θ is less which means x and i are more similar in nature.

Cons : Content-based Approach

- Finding the appropriate features is hard.
  eg. images, movies, music
- Overspecialization
  - Never Recommends items outside user's content profile.
  - People might have multiple interest.
  - Unable to exploit quality judgements of other users.
- Cold-start problem for new users
  - How to build a user profile.

Pros :

- No need for data on other users
- Able to recommends to users with unique tastes.
- Able to recommend new and unpopular items.
  - No First rater problem.
- Explanations for recommended items.

LECTURE

Campus: PCE Course: BTECH in CSE Class/Section: III Yr. Section- A Date: ...

Name of Faculty: Praveen Kumar Yadav Name of Subject: Machine Learning Code: 60

Date (Prep.): ..... Date (Del.): .....Unit No.: V .....Lect. No: .....

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc., will be taught in this lecture)

COLLABORATIVE FILTERING

IMPORTANT & RELEVANT QUESTIONS:

FEED BACK QUESTIONS (AFTER 20 MINUTES):

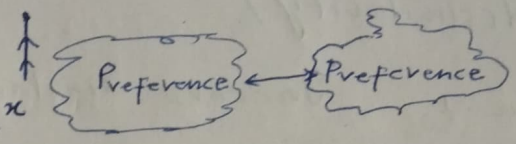
OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points abou students' feedback on this lecture, level of understanding of this lecture by students etc.)

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:

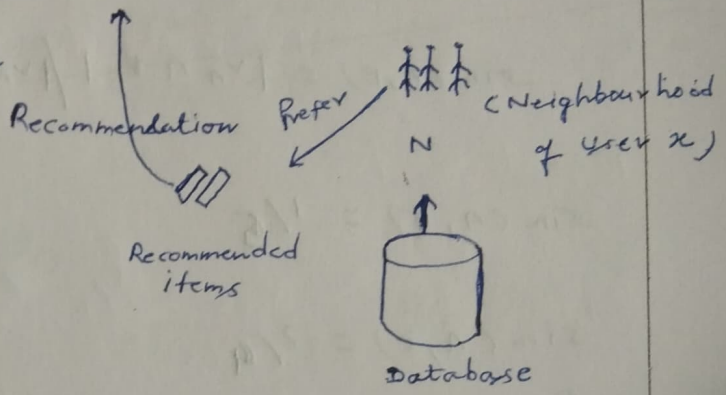
COLLABORAIVE FILTERING:-

Also known as User-User collaborative filtering.

- Consider a user x to whom we want to recommendations.



- Find a set of N of other users whose likes or dislikes (ratings) are similar to the likes and dislikes of user x.



- Estimate x's ratings based on ratings of users in N.

Here we have to find the notion of similarities to find the neighbourhood of user x.

eg-1.

	HP1	HP2	HP3	TW	SW1	SW2	SW3	rating vector
A	4			5	1			r_A
B	5	5				4	5	r_B
C			2					r_C
D		3					3	r_D

consider user x and y with rating vectors r_x and r_y.

we need to find a similarity metric sim(x, y)

capture intution that sim(A, B) > sim(A, C). (correctly classified the user based on similarity and disimilarity).

option-1. JACCARD similarity.

$$sim(A,B) = |r_A \cap r_B| / |r_A \cup r_B|$$

$sim(A,B) = 1/5$

$sim(A,C) = 2/4$

- Here $sim(A,B) < sim(A,C)$

Prob- it ignores rating values.

option-2. Cosine similarity

$$sim(A,B) = \cos(r_A, r_B)$$

Here all unknown values are initialized with zero.

$sim(A,B) = 0.38$

$sim(A,C) = 0.32$

- $sim(A,B) > sim(A,C)$, but not by much.

Prob- Treats missing rating as negative. (Bad Assumption)

option-3. Centered cosine

Normalize ratings by substracting row mean.

	HP1	HP2	HP3	TW	SW1	SW2	SW3	Avg row Mean
A	4			5	1			10/3
B	5	5	4					14/3
C				2	4	5		11/3
D		3					3	6/2

⇓

	HpL	HP2	HP3	TW	SW1	SW2	SW3	Sum = 0
				5/3	-7/3			
A	4-10/3 = 2/3							
B	5/3	5/3	-2/3					
C				-5/3	4/3	5/3		
D		0					0	

Here 0 is average rating.

Now $sim(A, B) = cos(r_A, r_B) = 0.09$

$sim(A, C) = -0.56$

$sim(A, B) > sim(A, C)$

Captures intution better

- Missing rating treated as Average
- Handles "tough raters" and "easy raters".

- Also known pearson corelation constant.

PAGE NO.

RATING PREDICTIONS :-

Let r_x be the vector of user x's ratings.

Let N be the set of k users most similar to x who have also rated item i.

prediction for user x and item i.

option-1. $r_{xi} = 1/k \sum_{y \in N} r_{yi}$

option-2- weighted Average $r_{xi} = \sum_{y \in N} s_{xy} r_{yi} \Big/ \sum_{y \in N} s_{xy}$

where $s_{xy} = sim(x, y)$.

That Approach is also known as User-user collaborative filtering.

Item-item Collaborative Filtering:-

Another view - item - item

- For item i, find other similar items.
- Estimate rating for item i based on ratings for similar items.
- Can use same similarity metrices and predictions functions as in user-user model.

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

r_{xi} ↙ user x, item i

s_{ij} - similarity of item i and j

r_{xj} - rating of user x on item j

$N(i;x)$ - set of items rated by x similar to i.

eg. item-item CF ($N=2$) ↑ Neighbourhood size = 2, find two similar movies.

Movies ↑	1	2	3	4	5	6	7	8	9	10	11	12	→ Users; Sim(1,m) ↑ Centered cosine distance from movie 1
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	[2]		3		4	3	5		0.41
4		2	4	✗	5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		[3]			2			4		0.59

[ ] - Unknown rating

Rating between 1 to 5

PAGE NO.

goal - estimate the rating of Movie 1 by user 5

step1 - Find other movie that are similar to movie 1.

Here we use Pearson corelation is similarity.

Subtract mean rating m_i from each movie i.

$m_1 = (1+3+5+5+4)/5 = 3.6$

row 1 : [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]

step2. Compute cosine similarities between rows.

$s_{13} = 0.41$ $s_{16} = 0.59$

step3 - Predict by taking weighted Average

$r_{15} = (0.41 * 2 + 0.59 * 3) / (0.41 + 0.59) = 2.6$

FOUN

Estd. 2000

LECTURE NOTES

Campus: PCE Course: BTECH in CSE Class/Section: III Yr. Section- A Date: .....

Name of Faculty: Praveen Kumar Yadav Name of Subject: Machine Learning Code: 6CS4-02

Date (Prep.): ..... Date (Del.): .....Unit No.: 5 .....Lect. No: .....

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc., which will be taught in this lecture)

Neural Network.

IMPORTANT & RELEVANT QUESTIONS:

FEED BACK QUESTIONS (AFTER 20 MINUTES):

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points abou students' feedback on this lecture, level of understanding of this lecture by students etc.)

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:

In 1950 or earlier the people come up with the idea that we can probably trying to replicate/ reproduce the functionality of brain (i.e what is happening of a human brain).

The neuron receives the input, then neurons somehow figureout what is important / what's not, so it can activate that signal or ignore that signal. you have chains of all those neurons connected to each other and somehow we processing all the input data that are coming in neurons and we can do something on that processed information.

so same process can Also be replicated in the machine through the Algorithm.

$x_i w_i$ = if $w_i = 0$ then o/p is 0 (ignore the signal)

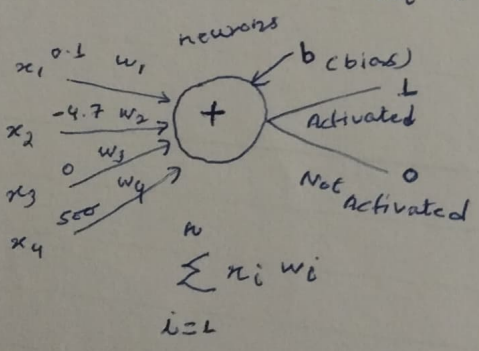
for eg $w_i >= 0$

$w_2 = 500$ then o/p - Amplify that signal

$w_i <= 0$

then - deAmplify the signal.

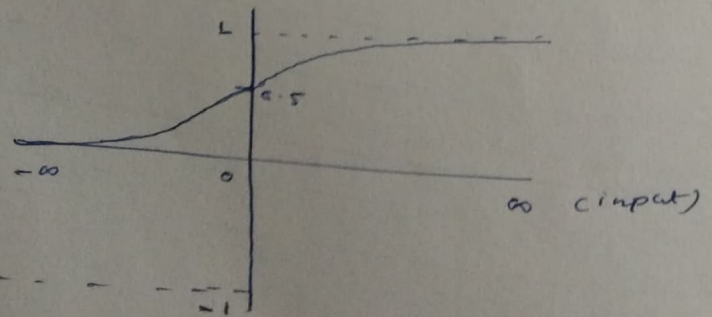
$w_4 = 0.0001$



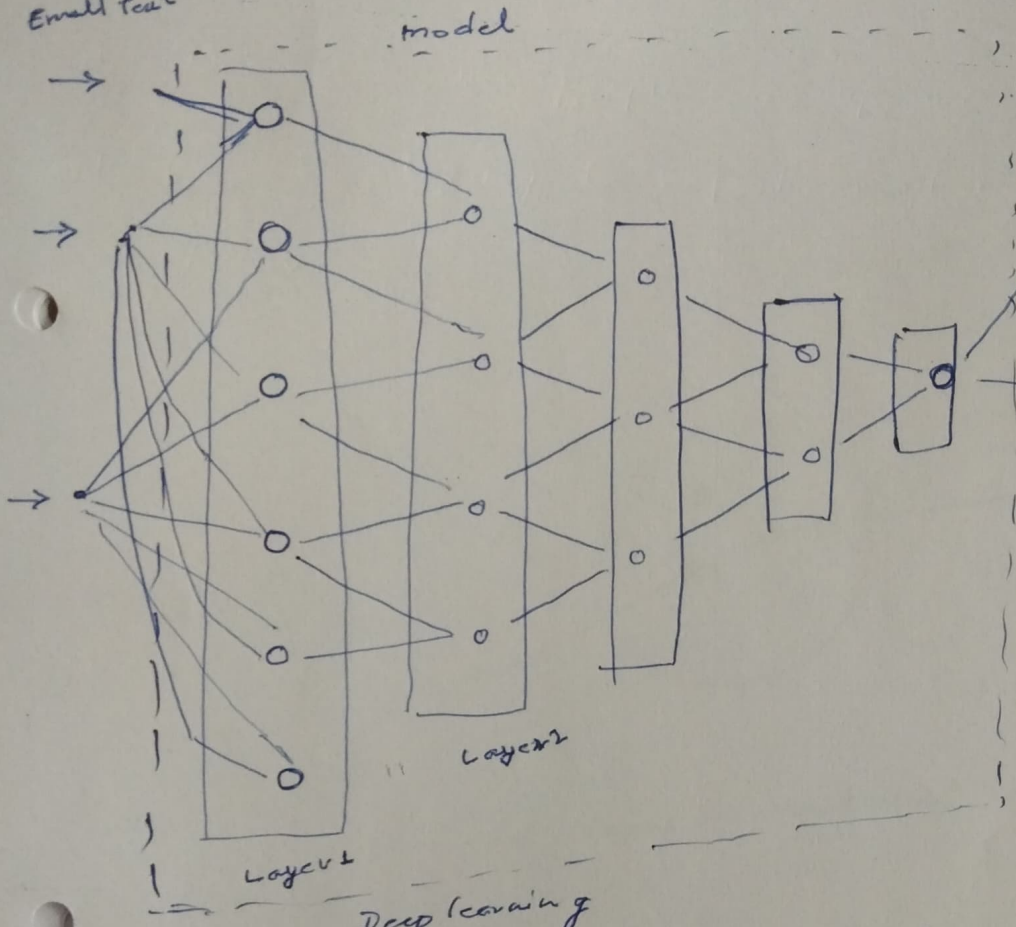
(−∞ to +∞) ——— (0, 1)

input range — Transformation done by sigmoid function

↘ These are rules that what we defined.



Email Text



2012 ImageNet Competition.
Alexnet - 7 or 8 Layer
Standford University
1.4 Million Images
CONN) 95%.

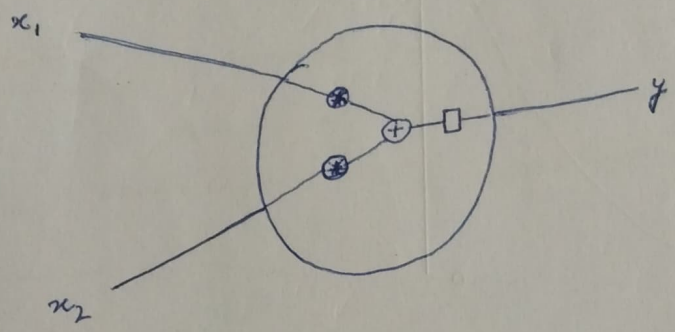
Deep learning

So In Deep learning you can stacks these neuron in one layer and processing simultaneously and on top of that we have another layers of neuron of partially or fully connected Previous layer and you can have multiple layers like that and last output layer produces output of all processed signals.

That kind of Architecture is known as Deep learning neural network.

Artificial Neural Network :-

Neuron: A neuron takes input, does some math with them and produces one output.



$x_1 = w_1 * x_1$
$x_2 = w_2 * x_2$
- each input is multiplied by a weight

Next, All the weighted inputs are added together with a bias b-

i.e

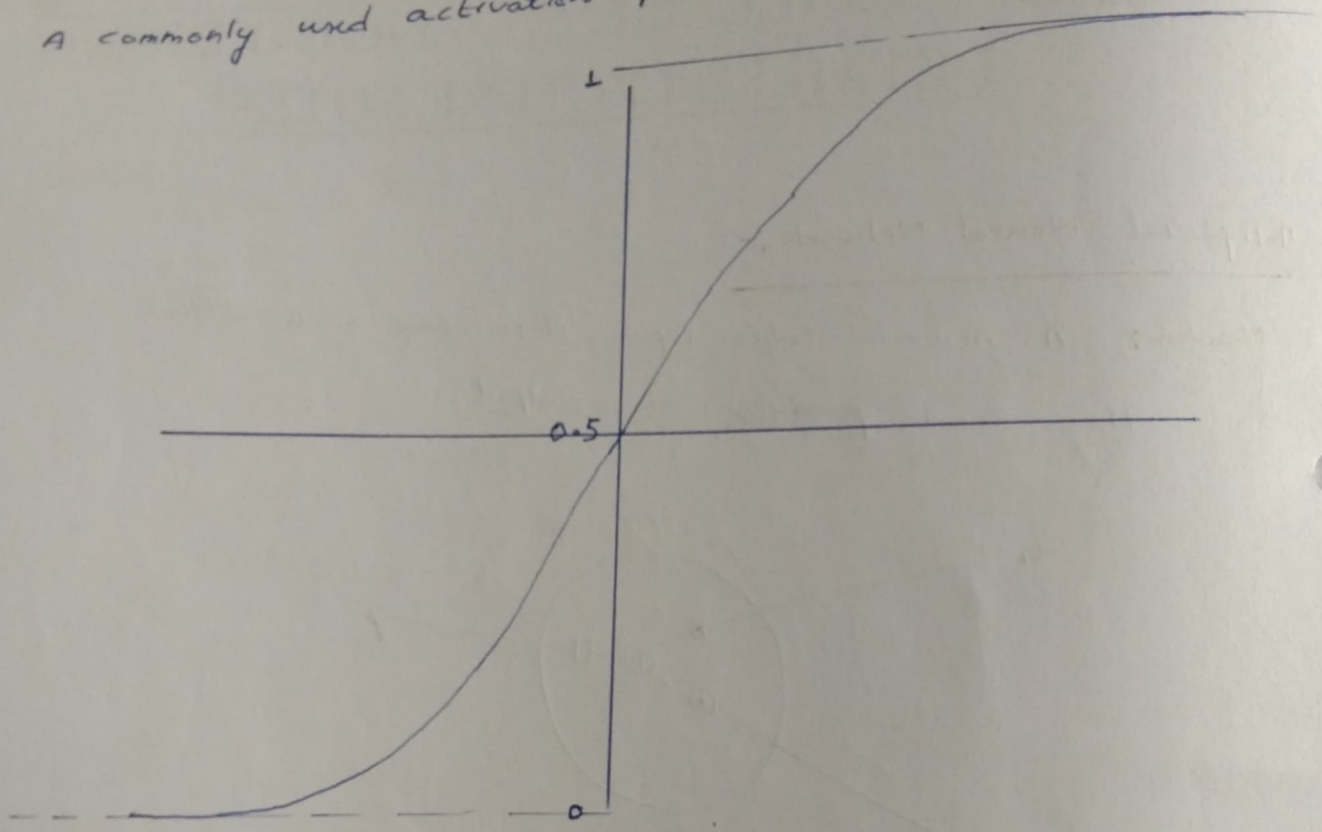
$$(x_1 * w_1) + (x_2 * w_2) + b$$

Finally, the sum is passed through an activation function-

$$y = f(x_1 * w_1 + x_2 * w_2 + b)$$

The activation function is used to turn an unbounded input into an output that has a nice, predictable form.

A commonly used activation function is the sigmoid function.



The sigmoid function only outputs numbers in the range (0, 1). You can think of it is as compressing (−∞ to +∞) to (0, 1) - big negative numbers becomes ~ 0 and big positive numbers become ~ 1.

For eg. $w = [0, 1]$ (w_1 w_2)

$b = 4$

$x = [2, 3]$ (x_1 x_2)

$(w.x + b) = ((w_1 * x_1) + (w_2 * x_2)) + b$

$= 0*2 + 1*3 + 4$

$= 7$

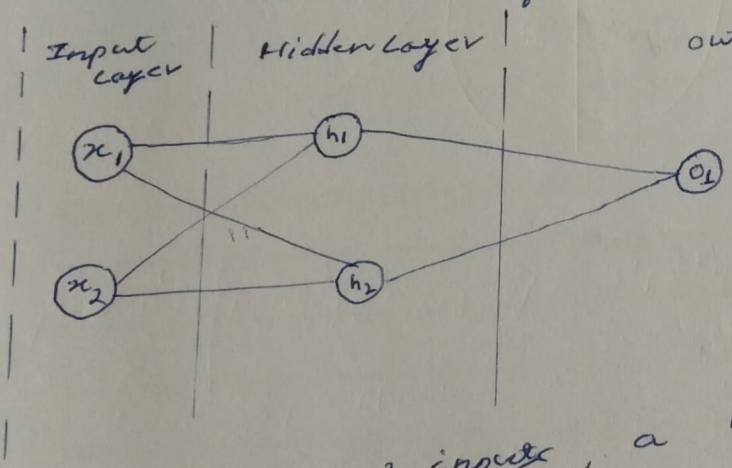
$y = f(w.x + b)$

$= f(7) = 0.9999$

Here the neurons outputs 0.999 given the inputs $x = [2, 3]$

This process of passing inputs forword to get an output is known as **Feed-Forword, Neural Network.**

Neural Network:- A neural network is nothing more than a bunch of neurons connected together.



This network has 2 inputs, a hidden layer with 2 neurons (h_1 and h_2) and output layer with 1 neurons (O_1). Here input for O_1 are the outputs from h_1 and h_2 - that's what makes this a network.

A Hidden layer is any layer between the input (first) layer and output (last layer).

Feed-Forward Neural Network :- A neural network can have any number of layers with any number of neuron's in those layers. So in the feed-forward neural network we are feed the input(s) forward through the neuron's in the network to get the output(s) at the end. This is called as **Feed-Forward Neural Networks**.

eg- $w = [0, 1]$ (w_1, w_2) for all neurons

bias $b = 0$ and same sigmoid function.

h_1, h_2, o_1 denote the outputs of neurons they represent.

$x = [2, 3]$ (x_1, x_2) (input Layer)

$$h_1 = h_2 = f(w \cdot x + b)$$
$$= f((0 * 2) + (1 * 3) + 0)$$
$$= f(3) = 0.9526$$

$$o_1 = f(w \cdot [h_1, h_2] + b)$$
$$= f((0 * h_1) + (1 * h_2) + 0)$$
$$= f(0.9526)$$
$$= 0.7216$$

So the output of a neural network for input $x = [2, 3]$ is 0.7216.

LECTURE NOTES

Campus: PCE Course: BTECH in CSE Class/Section: III Yr. Section- A Date: .....

Name of Faculty: Praveen Kumar Yadav Name of Subject: Machine Learning Code: 6CS4-02

Date (Prep.): ..... Date (Del.): .....Unit No.:.....V.....Lect. No: .....

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc., which will be taught in this lecture)

Perceptron

IMPORTANT & RELEVANT QUESTIONS:

FEED BACK QUESTIONS (AFTER 20 MINUTES):

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:

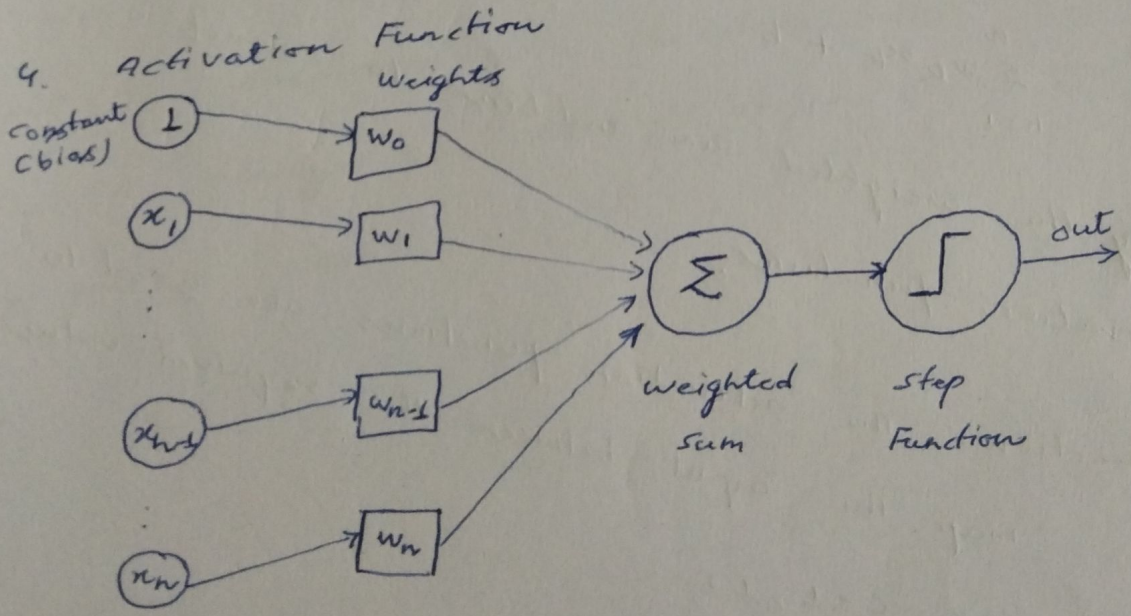
PAGE NO. ...

Perceptron :- A perceptron is a single layer network and a multilayer perceptron is called as Neural networks.

Perceptron is a linear classifier (binary). Also it is used in supervised learning.

The Perceptron consists of 4 parts.

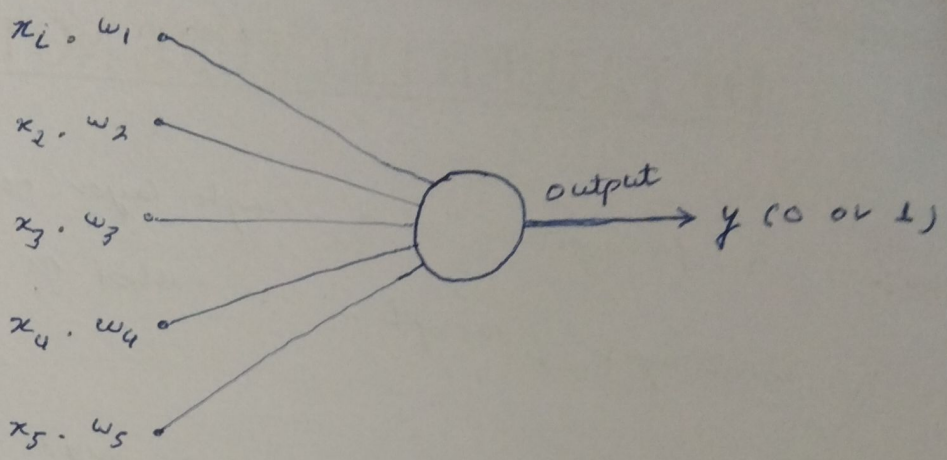
1. Input values or one Input layer
2. weights and bias.
3. Net sum
4. Activation Function



(Perceptron)

How Perceptron work:-

1. All input x are multiplied with their weights w. Let's call it K.



2. Add All the multiplied value and call them weighted Sum. and Term.

$$\sum_{K=1}^{n} w_K x_K$$

(K formula for K^{th} Term)

Sigma → for Summation

3. Add bias (b) to the weighted sum

$$\sum_{K=1}^{n} w_K x_K + b$$

4. Apply the weighted sum and bias to correct activation functions.

Activation Function - The activation functions are used to map the input between the required values like (0, 1) or (-1, 1).

weights :- weight shows the strength of the particular node.

bias :- bias is a constant which helps the model in a way that it can best fit for the given data.

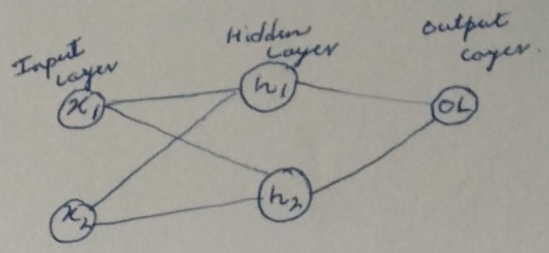
bias unit appended to the start/end of the input and each hidden layer and it is not affected by the values in previous layer.

It allow you to move the line up and down to fit the prediction with the data better. without b the line always goes through origin (0,0) and you may get a poorer fit.

Coding a Neuron -

```
import numpy as np

def sigmoid(x):
  # our activation function: f(x) = 1/(1 + e^(-x))
  return 1/(1+np.exp(-x))

class Neuron:
  def __init__(self, weights, bias):
    self.weights = weights
    self.bias = bias

  def feedforward(self, inputs):
    total = np.dot(self.weights, inputs) + self.bias
    return sigmoid(total)

weights = np.array([0, 1])   # w1 = 0, w2 = 1
bias = 4                     # b = 4

n = Neuron(weights, bias)

x = np.array([2,3])          # x1 = 2, x2 = 3
print(n.feedforward(x))
```

Multilayer Neural Network -



Coding a Neural Network:-

```
import numpy as np

class OurNeuralNetwork:

    def __init__(self):
        weights = np.array([0, 1])
        bias = 0

        self.h1 = Neuron(weights, bias)
        self.h2 = Neuron(weights, bias)
        self.o1 = Neuron(weights, bias)

    def feedforward(self, x)
        out_h1 = self.h1.feedforward(x)
        out_h2 = self.h2.feedforward(x)

        out_o1 = self.o1.feedforward(np.array([out_h1,
                                               out_h2]))

        return out_o1

network = OurNeuralNetwork()

x = np.array([2, 3])

print(network.feedforward(x))
```

LECTURE NOTES

Campus: PCE Course: BTECH in CSE Class/Section: III Yr. Section- A Date: .....

Name of Faculty: Praveen Kumar Yadav Name of Subject: Machine Learning Code: 6CS4-02

Date (Prep.): ..... Date (Del.): .....Unit No.: 5th .....Lect. No: .....

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc., which will be taught in this lecture)

Multilayer Network

Multilayer Perceptron.

IMPORTANT & RELEVANT QUESTIONS:

FEED BACK QUESTIONS (AFTER 20 MINUTES):

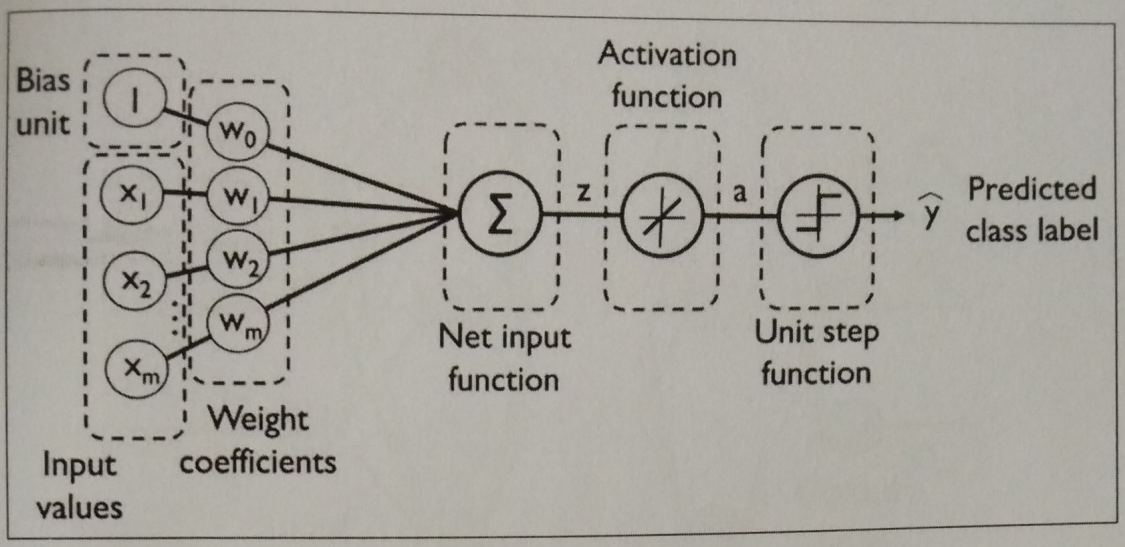
OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:

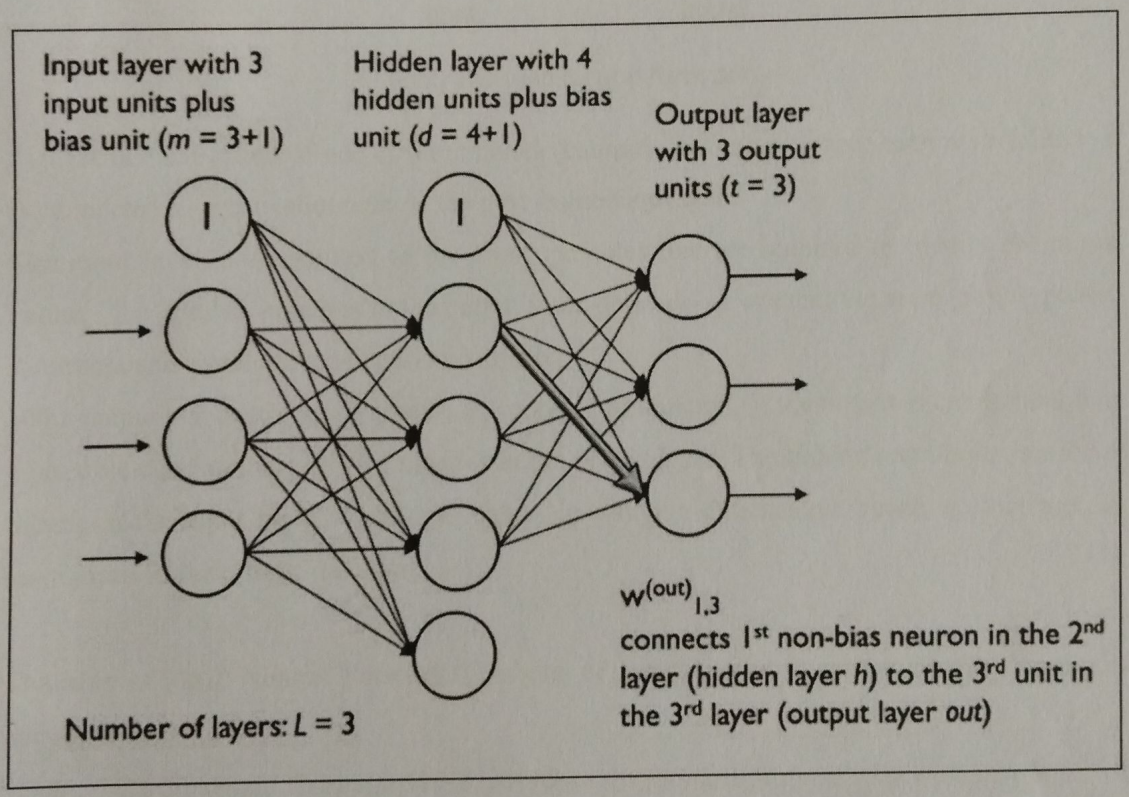
Multilayer Neural Network:- A fully connected multi-layer neural network is called a Multilayer Perceptron.

↳ MLP is a typical example of a feed forward artificial neural network.

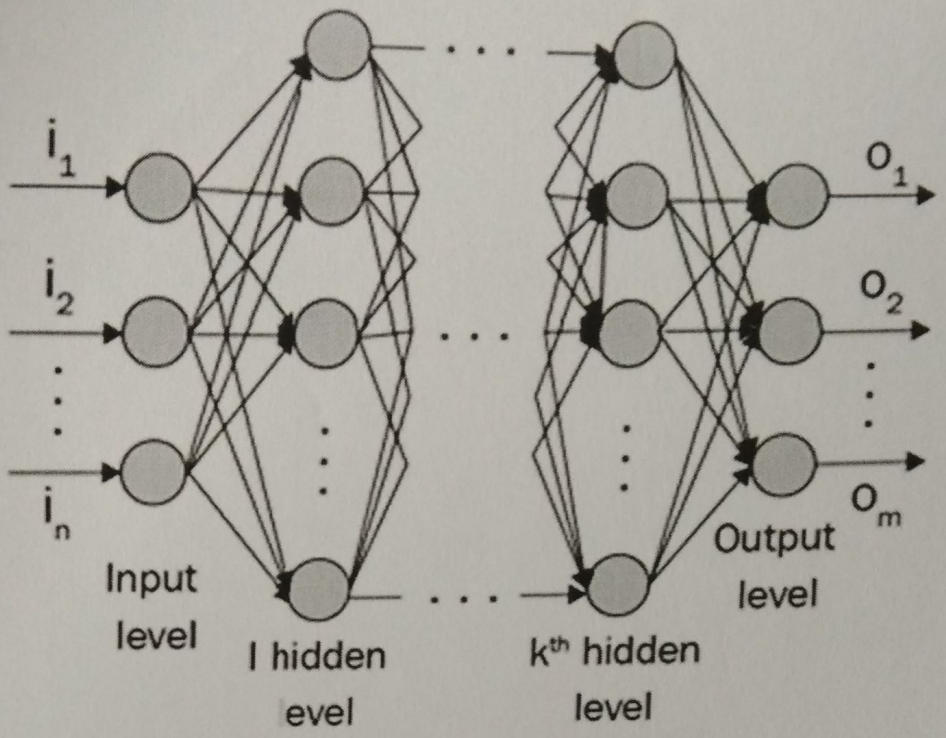
↳ MLP is basically used to solve non-linear seperable problems.



Single Layer Neural Network



A Multilayer Perceptron



Feed-Forward Neural Network

A layer is a collection of one or more nodes (computation units), where each node in a layer is connected to every other node in the next immediate layer.

The input layer is constituted of the input variables that are required to predict the output values. The number of nodes in the output layer depends on whether we are trying to predict a continuous variable or a categorical variable.

If the output is a continuous variable, the output has one unit. If the output is categorical with n possible classes, there will be n nodes in the output layer. The hidden level/layer is used to transform the input layer values into values in a higher-dimensional space, so that we can learn more features from the input.

Training of MLP Neural Network: Training of Neural network can be done by means of two algorithm-

1. **Feed Forward Propagation Algorithm:** In forward-propagation, we apply a set of weights to the input data, pass it through the hidden layer, perform the nonlinear activation on the hidden layer output, and then connect the hidden layer to the output layer by multiplying the hidden layer node values with another set of weights. For the first forward-propagation, the values of the weights are initialized randomly.

2. **Back Propagation Algorithm:** In back-propagation, we try to decrease the error by measuring the margin of error of output and then adjust weight accordingly. Neural networks repeat both forward- and back-propagation to predict an output until the weights are calibrated.

POORNIMA FOUNDATION

LECTURE NOTES

Campus: PCE Course: BTECH in CSE Class/Section: III Yr. Section- A Date: .....

Name of Faculty: Praveen Kumar Yadav Name of Subject: Machine Learning Code: 6CS4-02

Date (Prep.): ..... Date (Del.): .....Unit No.:.....Lect. No: .....

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc., which will be taught in this lecture)

Deep Neural networks -

MLP Training - chain-rule Implementation

IMPORTANT & RELEVANT QUESTIONS:

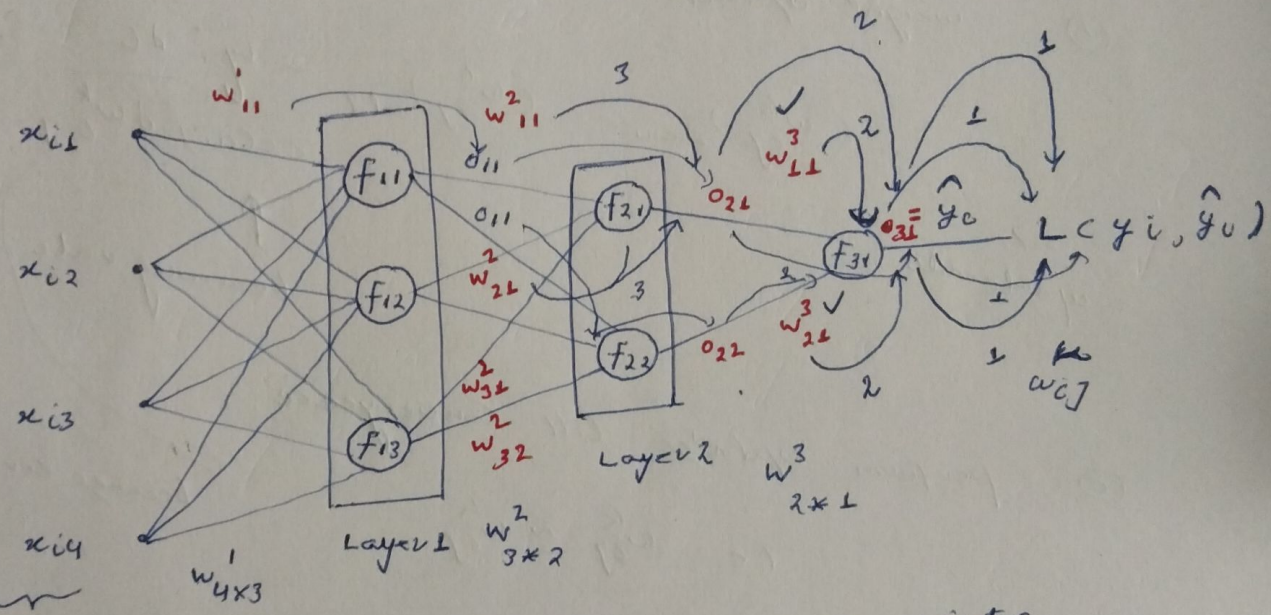
FEED BACK QUESTIONS (AFTER 20 MINUTES):

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:

Training an MLP - chain rule :-

$D = \{x_i \; y_i\}$ $\quad x_i \in R^4$, $y_i \in R$ } reg. prob. → square loss



$w^1_{4\times3}$, w^2_{3*2}, w^3_{2*1} = 20 data points

determine 12 6 2

step 1 - Define the cost Function -

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \text{reg. (regularization pm) on weights}$$

(square loss)

$$L_i = (y_i - \hat{y}_i)^2$$

$$L = \sum_{i=1}^{n} L_i + \text{reg.}$$

$\sum_{i,j,k} (w^k_{ij})^2$ — L2 reg.

$\sum_{i,j,k} |w^k_{ij}|$

Step 2- minimize L → SQUARE LOSS + Regularization pm.

w^1, w^2, w^3 i.e

$$\min_{w^k_{i,j}} L$$

Step 3- Use SGD or GD

(a) initialization of w^k_{ij} randomly.

(b) weight updation rule

Most Imp.

$$(w^k_{ij})_{new} = (w^k_{ij})_{old} - \eta \cdot \frac{dL}{dw^k_{i,j}}$$

η (eta) → learning rate

→ update rule

becomes zero

(c) perform updates till Convergence.

i.e $w^k_{ij} \simeq w^k_{ij}$

w^3_{11}

$$\frac{\partial L}{\partial w^3_{11}} = \frac{\partial L}{\partial o_{31}} \cdot \frac{\partial o_{31}}{\partial w^3_{11}}$$ (chain rule of differentiation)

$$\frac{\partial L}{\partial w^3_{21}} = \frac{\partial L}{\partial o_{31}} \cdot \frac{\partial o_{31}}{\partial w^3_{21}}$$

w^2_{11}

$$\frac{\partial L}{\partial w^2_{11}} = \frac{\partial L}{\partial o_{31}} \cdot \frac{\partial o_{31}}{\partial o_{21}} \cdot \frac{\partial o_{21}}{\partial w^2_{11}}$$

$$\frac{\partial L}{\partial w^2_{21}} = \frac{\partial L}{\partial o_{31}} \cdot \frac{\partial o_{31}}{\partial o_{21}} \cdot \frac{\partial o_{21}}{\partial w^2_{21}}$$

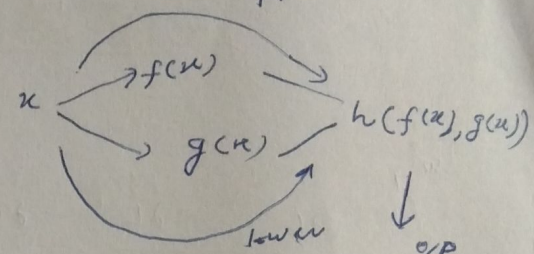
$$\frac{\partial L}{\partial w^2_{31}} = \frac{\partial L}{\partial o_{31}} \cdot \frac{\partial o_{31}}{\partial o_{21}} \cdot \frac{\partial o_{21}}{\partial w^2_{21}}$$

w'_{11}

$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial o_{31}} \cdot \frac{\partial o_{31}}{\partial o_{21}} \cdot \frac{\partial o_{21}}{\partial o_{11}}$$

$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial o_{31}} \cdot \frac{\partial o_{31}}{\partial w'_{11}}$$

upper



$$\frac{\partial o_{31}}{\partial w'_{11}} = \frac{\partial o_{31}}{\partial o_{21}} \cdot \frac{\partial o_{21}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial w'_{11}}$$

$$\frac{\partial h}{\partial x} = \left[\frac{\partial h}{\partial f} \cdot \frac{\partial f}{\partial x}\right] + \left[\frac{\partial h}{\partial g} \cdot \frac{\partial g}{\partial x}\right]$$

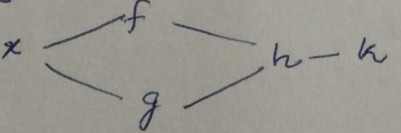
upper path — lower path

$$+ \frac{\partial o_{31}}{\partial o_{21}} \cdot \frac{\partial o_{22}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial w'_{11}}$$

$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial o_{31}} \cdot \left(\frac{\partial o_{31}}{\partial o_{21}} \cdot \frac{\partial o_{21}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial w'_{11}} + \right.$$

$$\left. \frac{\partial o_{31}}{\partial o_{22}} \cdot \frac{\partial o_{22}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial w'_{11}} \right)$$

using chain rules



$$\partial k/\partial x = \frac{\partial k}{\partial h} \cdot \frac{\partial h}{\partial x}$$

$$\frac{\partial h}{\partial x} = \frac{\partial h}{\partial f} \cdot \frac{\partial f}{\partial x} + \frac{\partial h}{\partial g} \cdot \frac{\partial g}{\partial x}$$

$$\partial k/\partial x = \frac{\partial k}{\partial h} \times \left\{ \frac{\partial h}{\partial f} \cdot \frac{\partial f}{\partial x} + \frac{\partial h}{\partial g} \cdot \frac{\partial g}{\partial x} \right\}$$

POORNIMA FOUNDATION

LECTURE NOTES

Campus: PCE Course: BTECH in CSE Class/Section: III Yr. Section- A Date: .....

Name of Faculty: Praveen Kumar Yadav Name of Subject: Machine Learning Code: 6CS4-02

Date (Prep.): ..... Date (Del.): .....Unit No.:.....Lect. No: .....

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc., which will be taught in this lecture)

Back propagation Algorithm For Perceptron

IMPORTANT & RELEVANT QUESTIONS:

FEED BACK QUESTIONS (AFTER 20 MINUTES):

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:

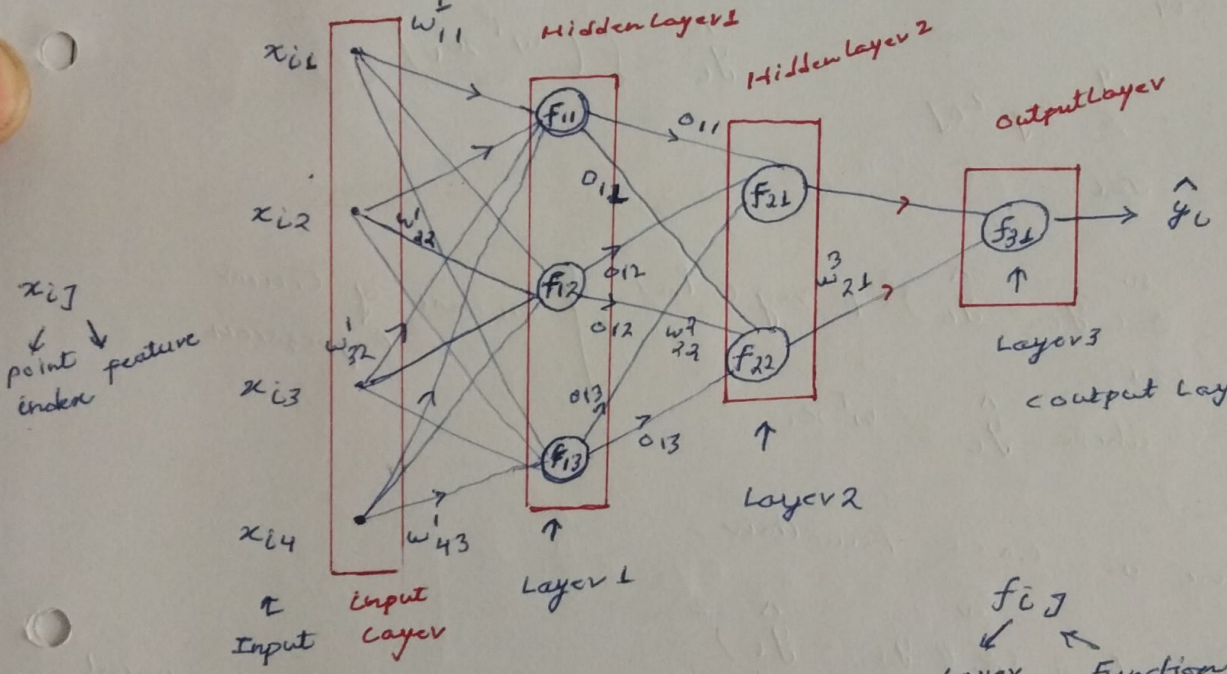
Backpropagation Algorithm in Neural network :-

Notations :-

$D = \{x_i, y_i\}$

$x_i \in R^4$; $y_i \in R$

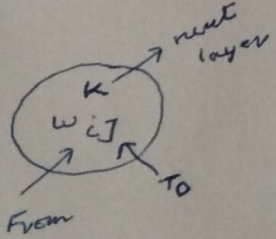
$W^1_{4\times3}$ $W^2_{3\times2}$ $W^3_{2\times1}$



x_{ij} ← point index, → feature index

f_{ij} ← Layer no, → Function index (neuron index)

o_{ij} ← i^{th} layer, → j^{th} neuron index



Fully Connected Multilayer Perceptron.

input × 3 = 12
4

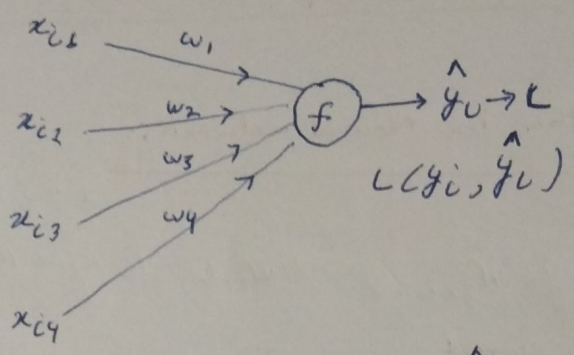
$$W^1 = \begin{bmatrix} w^1_{11} & w^1_{12} & w^1_{13} \\ w^1_{21} & w^1_{22} & w^1_{23} \\ w^1_{31} & w^1_{32} & w_{33} \\ w_{41} & w_{42} & w^1_{43} \end{bmatrix}_{4\times3}$$

Training a single Neuron model:-

↳ To Find the best edge-weights using D_{Tr} (Training Data)

Perceptron & LR → Single Neuron models for classification

$$D_{Tr} = \{x_i, y_i\}_{i=1}^{n}$$



Lr. Regression $x_i \in R^{d(R^d)}$ $y_i \in R$

$$\hat{y}_i = \sum_{j=1}^{d} w_j x_{ij}$$

$$\hat{y}_i = w^T x_i$$

$$\hat{y}_i = f(w^T x_i)$$

$$= \sum_{j=1}^{4} w_j x_{ij}$$

$$\min_{w_i} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \text{reg}(P_m)$$

where $\hat{y}_i = w^T x_i$

} In case of Linear Regression

Step 1 - Define a Loss Function -

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$L_i = (y_i - \hat{y}_i)^2$ square loss of i^{th} training point

$$L = \sum_{i=1}^{n} L_i$$

Step 2 optimization problem -

$$\min_{w_i} \sum_{i=1}^{n} (y_i - w^T x_i)^2$$

y_i

$$\min_{w_i} \sum_{i=1}^{n} (y_i - f(w^T x_i))^2 + \text{regularization}$$

PAGE NO.

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - f(w^T x_i))^2 + \text{reg}.$$

3) solve the optimization problem.

a) initialization of w_i s -> random initialization

b) Compute the partial Derivative of L w.r.t to w i.e.

$$\nabla_w L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \vdots \\ \frac{\partial L}{\partial w_4} \end{bmatrix} \quad w \in R^4 \quad x_i \in R^4$$

c) $w_{new} = w_{old} - \eta \nabla_w L$

for iter = 1 to k

$$(w_i)_{new} = (w_i)_{old} - \eta \left[\frac{\partial L}{\partial w_i}\right]_{(w_i)old}$$

weight updation rule

$$\nabla_w L = \left[\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial w_3}, \frac{\partial L}{\partial w_4} \right]^T$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial f} \cdot \frac{\partial f}{\partial w_1} \rightarrow \text{chain rule of differentiation}$$

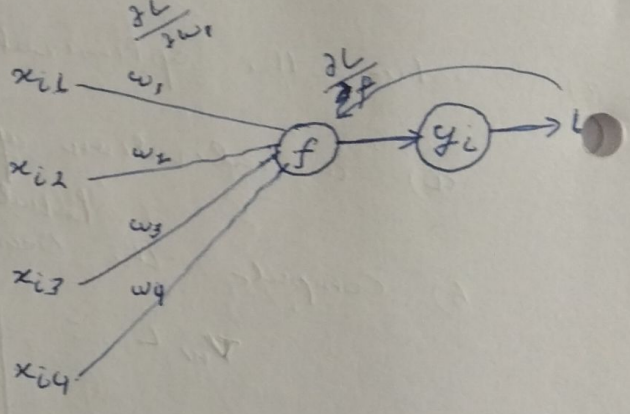
partial

chain rule

eg. $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial f_1} \frac{\partial f_1}{\partial f_2} \frac{\partial f_2}{\partial x}$

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$= \sum_{i=1}^{n} (y_i - f(w^T x_i))^2$$



$$\frac{\partial L}{\partial f} = -\sum_{i=1}^{n} 2(y_i - f(w^T x_i))$$

$$\frac{\partial f}{\partial w_1} = x_i$$

$$\boxed{\frac{\partial L}{\partial w_1} = \sum_{i=1}^{n} -2 x_i (y_i - \hat{y}_i)}$$

$$\frac{\partial L}{\partial w_2} \quad \cdots \quad \frac{\partial L}{\partial w_4}$$

POORNIMA FOUNDATION

LECTURE NOTES

Campus: PCE **Course: BTECH in CSE** **Class/Section: III Yr. Section- A** **Date: .....**

Name of Faculty: Praveen Kumar Yadav **Name of Subject: Machine Learning** **Code: 6CS4-02**

Date (Prep.): ..... Date (Del.): .....Unit No.:.....Lect. No: .....

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc., which will be taught in this lecture)

Back propagation Algorithm.

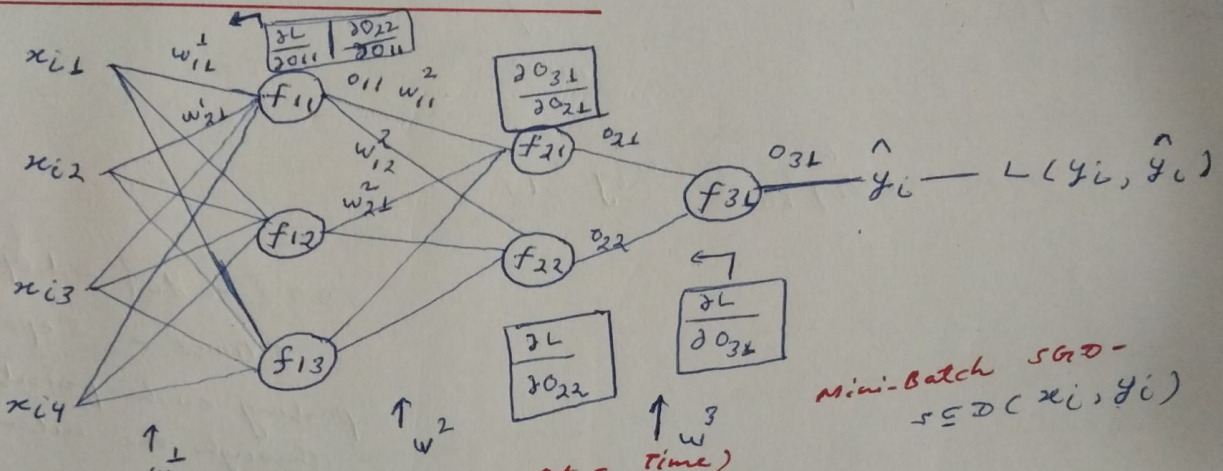
IMPORTANT & RELEVANT QUESTIONS:

FEED BACK QUESTIONS (AFTER 20 MINUTES):

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:

Backpropagation Algorithm in MLP:-



Mini-Batch SGD - $S \subseteq D (x_i, y_i)$

SGD Algorithm: (only one point at a time)

$D = \{x_i, y_i\}$

step 1 :- Initialize w^k_{ij}'s randomly.

step 2 :- for each x_i in D

(a) - pass x_i forward through the network. (Forward propagation)

and calculate Loss

(b) - compute $L(\hat{y}_i, y_i)$

(c) Compute all derivatives using chain rule of derivatives and Memorization $\left(\frac{\partial L}{\partial w^k_{ij}}\right)$

(d) Update weight from end of the network to the start. (Backward propgation)

For example -

$$(w^3_{11})_{new} = (w^3_{11})_{old} - \eta \frac{\partial L}{\partial w^3_{11}}$$

$$\downarrow$$

$$\frac{\partial L}{\partial o_{31}} \cdot \frac{\partial o_{31}}{\partial w^3_{11}}$$

Step3- repeat step2 till convergence to solution.

$$\downarrow$$

means.

$$(w^k_{ij})_{new} \simeq (w^k_{ij})_{old}$$

$D = \{x_i, y_i\}$ - 1 epoch

5 times - 5 epoch

passing each points through the network.

Backprop -

① init

② 1 epoch — forward prob
— compute losses
— compute derivatives (chain rule and Memorization)
— update weight.

③ repeat step2 till convergence

so basically a backpropagation is a basically multiple epoch training methodologies where we leverage the concept of chain rule and memorization to update the weights of NN.

v. very. Important points.

↳ Backpropgation is only work when the activation function is differtiable other-wise training of NN is not possible.

PAGE NO.

(SGD Approach)

② Instead of sending one point we can also send the batch of points in the network (Mini-batch SGD). s.t D(xi yi) and if we send All the data points through the network - then it is called BGD (Gradient Descent Approach).

③ All of the data points in RAM and computing "y" using GD is extremely Time Consuming. (in case of Gradient Descent Algorithm).