Deep learning

→ models require very little human intervention but need huge amounts of data.

→ No need of intermediate step: Feature extraction.

→ DL is a specialized subset of ML.

→ Relies on a layered structure of algo called as Artificial Neural networks

Artificial Neuron

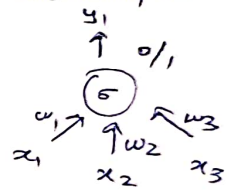
→ understand pattern of data, we add layers. Each layer

→ fundamental unit of deep NN is artificial neuron. (mathematical features)

like x_1, x_2, x_3 → features (given as i/p) --> algo

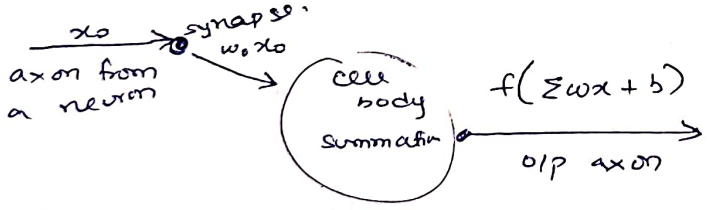
make/give importance to features x_2 carries larger wt ⇒ how much influence & wt

→ Neural networks are set of algos that have been developed to mitate the human brain to identify patterns.



→ The inspiration for the neuron comes from our understanding on biological neurons, also called neural processing units.

Neurons (Biological vs Artificial)



1st Artificial NN: McCulloch Pitts Neuron.

replicate OR function

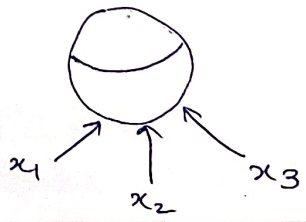
→ have n features

→ adding all our features (values can be either 0 or 1) also a limitation.

→ design a threshold ie; $\geq \theta \rightarrow 1$
$< \theta \rightarrow 0$

→ Proposed a highly simplified computational model of the neuron.

→ The o/p $y = 0$, if any x_i is inhibitory,
otherwise
$y = f(g(x)) = 1$;



Prediction-Based Models - Word2Vec

→ word2vec is a neural network-based model created by Google researchers in 2013 to learn word-embeddings (ie; vector representations of words).

→ These embeddings capture semantic relationships b/w words, enabling similar words to have similar vector represent^{n} in a high-dim space.

CBOWM

Train an ML model

8 unique words : The quick brown fox jumped over the lazy dog.

each word represented by vectors.

n=2

check 2 words before and 2 words after.

→ The input (first) layer is represented by a one-hot encoded vector, and consists of the context words surrounding the target word.

→ Training data: All n-word windows in the corpus.

→ The hidden (second) layer is where the word embeddings are learned & the size depends on the dimensionality of w.E you want to learn.

Skip-gram

→ unlike CBOW which uses surrounding words to predict the centre word, skip-gram uses the centre word to predict the surrounding words.

CBOW Doubt kills more dreams than failure ever will.

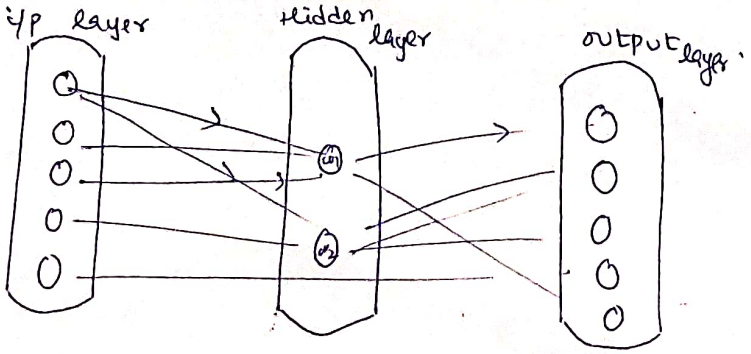
skip gram : opposite.

pipeline

n=2 2 words before, after.

Text corpus

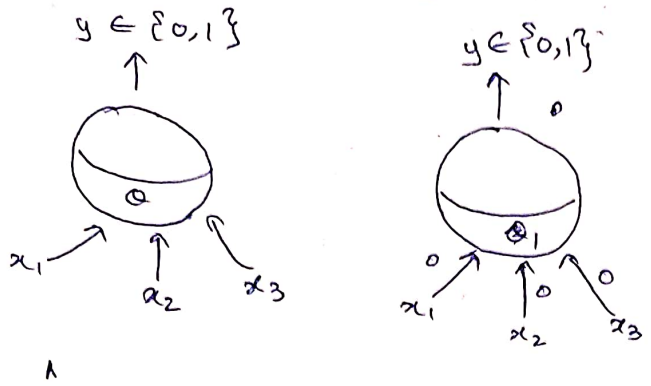
NN architecture.



Advanced NN→ much better than statistical transformer large corpus of text capturing semantic analysis check window size.

Recurrent NN.

Boolean functions using MP Neurons



able to replicate logic gates.

→ MP Neuron can be used to represent Boolean functions which are linearly seperable.

→ It produces a linear seperability for Boolean functions, such that all i/ps which produce a 1, lie on one side of line and 0 on the other side.

Shortcomings:
- Not capable of non-boolean inputs
- All i/ps given equal wts. Not all have equal weightage
- The threshold θ must be chosen by hand.
- works only for linearly seperable.

* Rosenblatt's Perceptron

Features:
- It can process non-Boolean inputs.
- Different weights can be assigned to each i/p automatically
- threshold θ is assigned automatically.

Simple image classification was able to be performed.

Give image values → perceptron based models

Pixels x_1, x_2, x_3

* Minsky & Papert

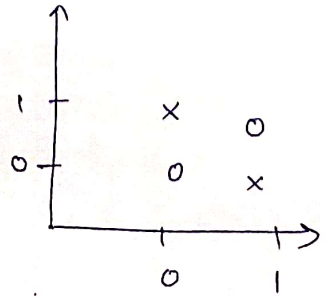
→ limitation of perceptron: failed to replicate exclusive OR
Couldn't find values of w_1, w_2, w_3.

→ 1st AI winter: stopped working, not a good model.

→ An intro to computational geometry.

XOR logical fn.

x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0



able to find w_0, w_1, w_2.

$y = w_0 + w_1 x_1 + w_2 x_2$

XOR (x_1, x_2) = linear class^n x

↓ combination of simple gates.

& combine learning.

A single perceptron is not able to learn anything about non-linear data. So, we make layers / network of perceptron so that one learns about and, one about or.

$h_1 = \sigma(w_{11} x_1 + w_{12} x_2 + w_{01})$

train other perceptron (h_2).

$h_2 = \sigma(w_{21} x_1 + w_{22} x_2 + w_{02})$

$h_1, h_2 \rightarrow$ hidden layer.
actually helps in learning non-linear data.

ex: h_1 is learning or
h_2 is learning

what single perceptron couldn't do, it was able to achieve by constructing group of

multi-layer
i/p layer: h_1, h_2 (depending on number of features)
middle layer: hidden layer (4 neurons)
learning different values with & weights.

wts → relative in
that learn from dataset

when we have two or more hidden layers ⇒ Deep Neural Network.

whenever we are working with non-linear data
these neurons will learn non-linearity, (combination of small st lines)
understanding a small part of data.

Q. How to decide how many neurons & hidden layers we need?

n features → n neurons.

As we increase layers
1 layer → 4 neurons what should be the wt. parameters.
tend to overfit
understand structure / amount of data
adding more layers ⇒ learn more or learning extra may learn noise.