Date ..../..../.....

Chapter - 5

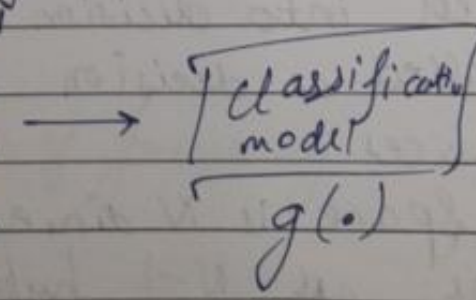
Standard Algorithm
→ The steps to solve a problem are well defined
→ Steps are coded in some ordered sequence to transform the input from one form to another
→ rules are unambiguous
→ sufficient knowledge is available to fully solve the problem.

Classification
→ Detect Spam Emails
→ Identify category
→ review +ve or -ve

classification is the task of assigning predefined dis-joint categories to objects

⇒ Definition of Classification
→ The input is a collection of records
→ Each record is represented by a tuple (x, y)
→ $x = x_1, x_2 \cdots x_n$ & $y = y_1, y_2 \cdots y_n$ are the input features & the classes respectively
→ $x \in R^2$ is a vector - the set of observed variable
→ (x, y) are related by an unknown function.
The goal is to estimate the unknown function $g(\cdot)$, also known as classifier function, such that $g(x) = f(x)$ $\forall x$

(x) features as input



output

y_1

y_2

$y = f(x)$

Date ..../..../.....

$g(x) \approx f(x)$ $\quad y = f(x)$

model expected

What does the classifier function do?

Assuming that we have a linearly separable x, the linear classifier function $g(.)$ implements decision rule :-

→ Fitting a straight line to a given data set requires two parameters (w_0 and w)

w_0 = bias $\quad w$ = weight

→ The decision rule divides the data space into two sub-spaces - separating two classes using a boundary

→ The distance of the boundary from the origin : $\frac{w_0}{\|w\|}$

→ Distance of any point from the boundary = $d = \frac{g(x)}{\|w\|}$

Linear models for classification

→ The goal of classification is to take a vector x & assign it to one of the N discrete classes C_n where $n = 1, 2, 3 \ldots N$

→ The classes are disjoint & an input is assigned to only one class.

→ The input space is divided into decision regions.

→ The boundaries are called as decision boundaries or decision surfaces

→ In general, if the input space is N dimensional then $g(x)$ would define as $N-1$ hyperplane

Spiral

Date ..../..../.....

1 D - Decision boundary FOR AND GATE

The decision regions are separated by a hyperplane & it is defind by $y(x) = 0$. this separate linearly separable classes C_1 & C_2

Decision boundary for sentiments

Let us consider some +ve & -ve sentiment terms which are contained in two class C_p & C_n.

C_p = [achieve efficient improve profitable] = +1
C_N = [termination penalties misconduct] = -1

Perceptron

⇒ Law of Association

The law of similarity — If two things are similar, the thought of one will tend to trigger the thought of other - word 2 vec.

The law of contrast — Seeing or recalling something may also trigger the recollection of something completely opposite

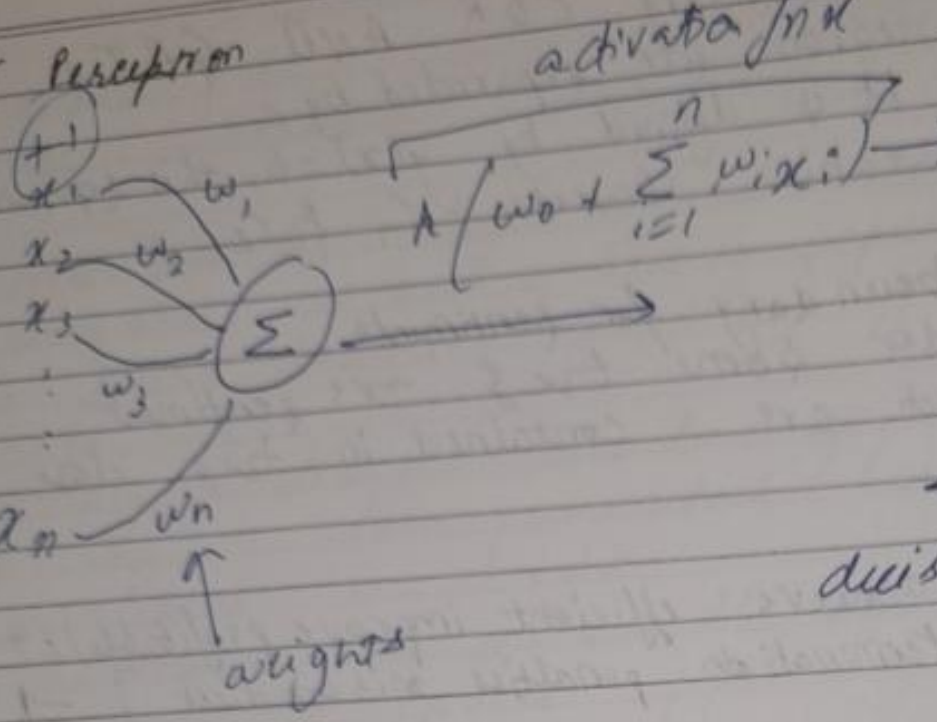
The law of contiguity — Things or events that occur close to each other in space or time tend to get linked together in the mind

The law of frequency — The more often two things or events are linked, the more powerful will be that association.

Spiral

Date ..../..../.....

Perceptron



Perceptron Learning

→ Perceptron learns the weights
→ They are adjusted until the output is consistent with the target output in the training examples
→ $w^{k+1} \propto (y - \hat{y})$
→ The weights are updated as below:-

$$w_j^{k+1} = w_j^{(k)} - \eta (y_i - \hat{y}^k) x_{ij}$$

where $w^{(k)}$ is the weight parameter associated with the i^{th} input at k^{th} iteration.
η is the learning parameter and x_{ij} is the j^{th} attribute of the i^{th} training sample.
→ If $(y - \hat{y}) \approx 0$, no prediction error
→ During the training the weights contributing most to the error require adjustments

Date ..../..../.....

Algorithm for Perceptron Learning

1] Total no. of input vectors = k
2] Total no. of features = n
3] Learning parameter $\eta = 0.01$ where $0 < \eta < 1$
4] epoch count $t=1$, $j=1$
5] Initialize weights w_i with random numbers
6] Initialize the input layer with $\vec{x_j}$
7] Calculate the output using $\sum w_i x_i + w_0$
8] Calculate the $(y - \hat{y})$
9] Update the weights $w_j(t+1) = w_j - \eta(y-\hat{y})x_j$
10] Repeat steps 7 to 9 untill the error is less than 0 or a predetermined no of epochs have been completed

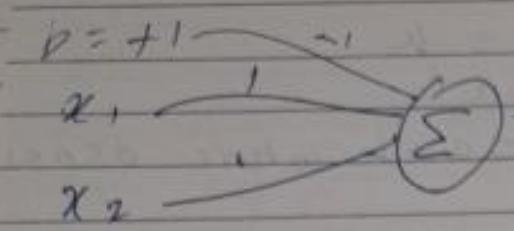
To provide a stable weight update for this step $w_j(t+1) = w_j - \eta(y-\hat{y})x_j$ we require a small η. This results in slow learning. Bigger η would be good for fast learning

Activation functions
→ Hard Threshold → Tanh → Leaky ReLU
→ Sigmoid → ReLU → Softmax

Spiral

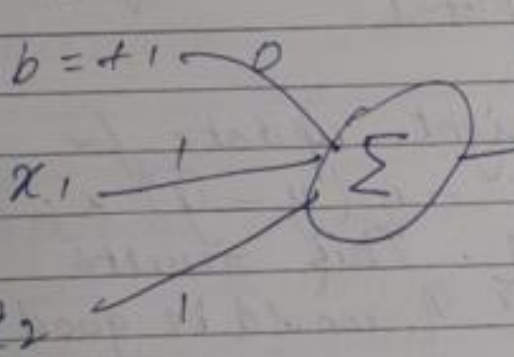
Date ..../..../.....

LOGICAL AND



Input x_1	Input x_2	$x_1 w_1 + x_2 w_2 + b$	output y
0	0	$0.1 + 0.1 + (-1)$	0
0	1	$0.1 + 1.1 + (-1)$	0
1	0	$1.1 + 0.1 + (-1)$	0
1	1	$1.1 + 1.1 + (-1)$	1

LOGICAL OR

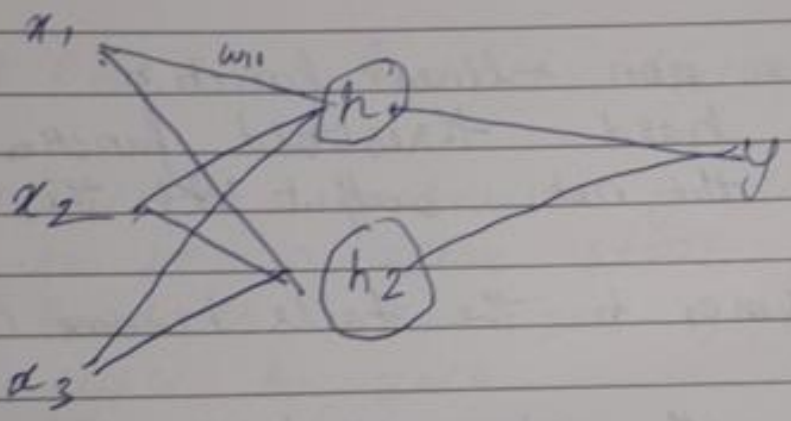


input x_1	input x_2	$x_1 w_1 + x_2 w_2 + b$	output y
0	0	$0.1 + 1.0 + 0$	0
0	1	$0.1 + 1.1 + 0$	1
1	0	$1.0 + 0.0 + 0$	1
1	1	$1.1 + 1.1 + 0$	1

Date ../../.....

⇒ Sentiment Analysis - Using Perceptron
→ Ability to classify reviews as +ve or -ve
→ +ve & negative words for training
→ Glove word embedding as features - input
→ Fo element word embedding
→ Training data generated using the intersection of the sentiment word list & word embedding from Glove.

LOGICAL XOR
→



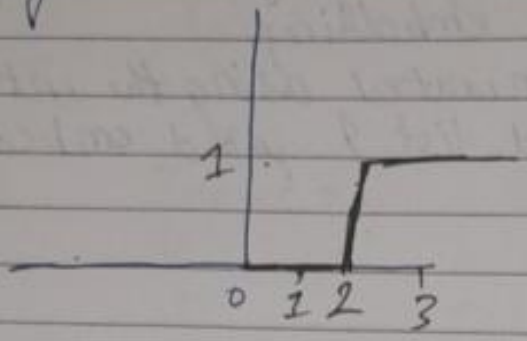
Input x_1	x_2	b	h_1	h_2	y_0
0	0		0	0	0
0	1		1	0	1
1	0		1	0	1
1	1		1	1	0

Date ..../..../.....

Activation Functions

→ Hard Threshold

If value is greater than threshhold then 1, if below that then 0



Sigmoid

→ Sigmoid is a non-linear function

→ Better than hard threshold function as it squashes the net output into the range [0, 1]

→ The value closer to the tails become 0 or 1

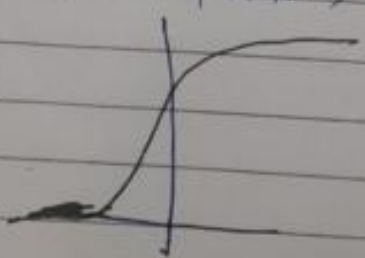
→ In some cases, the value quickly saturate at 0 or 1

→ At the bottom tail, the most values become zero during the training & hence the most important aspect of learning of neural network is inhibited.

→ Sigmoid outputs are not zero-centered

→ It is undesirable to have all the values squashed near the tails, where the gradient is 0

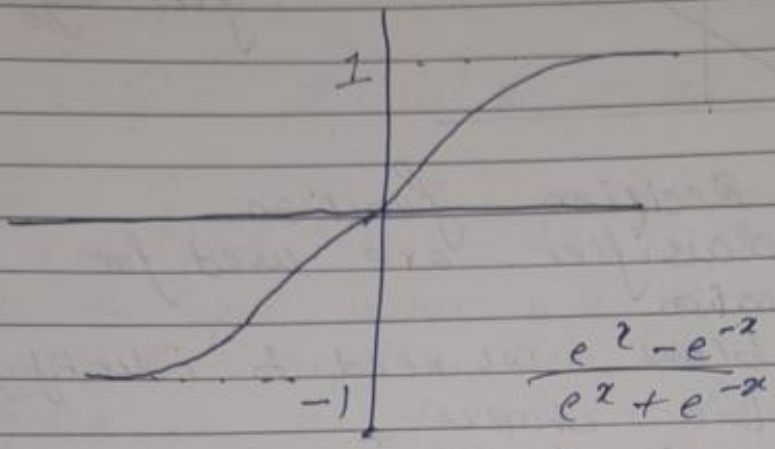
$$\frac{1}{1+e^{-x}}$$



Date ..../..../.....

TAN H

→ This a zero based non-linear function



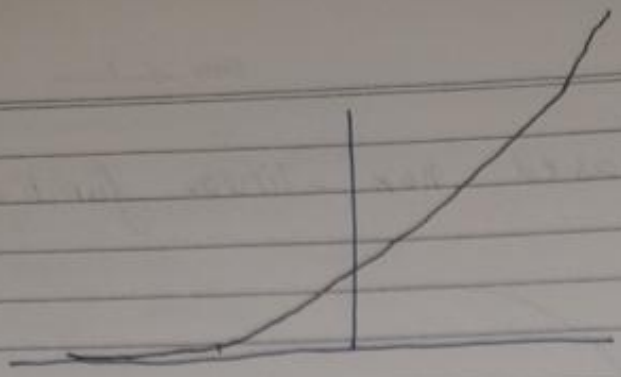
RELU - Rectified Linear Unit

→ There is a continuous gradient for the neurons to be in active state
→ Produces a non-zero gradient for values closer to zero
→ Leaky ReLu

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}$$

→ Produces efficient propagation of the gradient
→ computationally efficient
→ scale invariant
→ Unbounded and not zero centered
→ learning rate (usually, very small) has to be fine tuned to minimize death of neurons.

Date ..../..../.....



$f(x) = \max(0, x)$
$f(x) = \log_{10}(1+e^x)$

Multiclass Decision function
→ All linear classifier are used for binary classification
→ In NLP problems, we need to identify more than two classes
→ document classification → Sentiment Analysis
→ We need a decision function that predicts more than two classes by providing appropriate values
→ An extension of the case for 2 would be hard to manage

Softmax
→ Need a function that takes as input a vector of size with N real numbers, and normalize it into K classes
→ need a function that normalizes the net output & classes well separated (ideal condition)
→ Need a function that fits the classes using probability and distributes the probability density

$$\text{Softmax}(a_j) = P(C_k | x_j) = \frac{e^{a_j}}{\sum_{k=1}^{K} e^{a_k}}$$

K & x_j is the j^{th} input vector belonging to class k & $a_j = x_j \cdot w_{ij}$ $j=1$

Spiral

Date ..../..../.....

Gradient Descent

⇒ Loss function

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

where $\hat{y}$ is the predicted value
w_0 is the bias
x is the input vector w is the weight vector
if y is the target, then the loss for x is defined as a squared function

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

The main idea is to reduce the residual $(y - \hat{y})$. When the value of L becomes negligible, we have predicted the vector to belong to a known class. The loss function computes the error for a single training example.

Cost function

Let us assume that we have a set of vectors for training. In the case of the sentiment analysis, these are the vectors obtained using any of word embedding methods, representing the sentiment words. We could also use one hot vectors representing the same

$$X = [x_1, x_2, x_3 \ldots x_n]$$

Combining the model parameters $w_0, w_1, w_2 \ldots w_n$ with the loss function, we get a

Spiral

Date ..../..../.....

cost function, averages over all the input training samples

$$J(\theta) = \frac{1}{2} \sum L(y_i, \hat{y}_i)^2$$

→ The loss is a function of prediction & target values
→ The cost is a fnx of model parameters and bias

Gradient Descent

→ GD iteratively used to adjust the weights & as a result to minimize the cost fnx.
→ Initialize the weights to random values
→ Iteratively adjust the weights in the direction of the steepest descent or in the direction that most decreases the cost fnx. To update the weights in the steepest descent, a learning parameter η is used

$$w_j \leftarrow w_j - \eta \frac{\partial J(\theta)}{\partial w_j}$$

$$\leftarrow w_j - \eta \sum_{i=1}^{n} x_j^i (y - \hat{y})$$

η = learning rate $0.01 == 0.001$

GD Advantages:-

→ Iterative → computationally efficient → generic → can solve non-linear equation → suitable for large model
→ very slow when it reaches close to local minima

Spiral