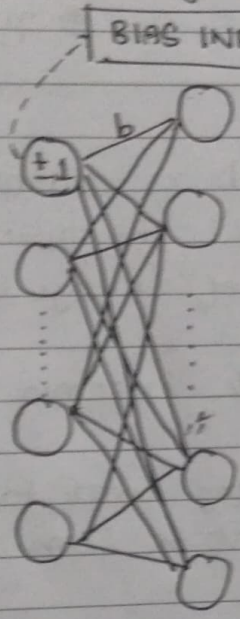
* PERCEPTRON: (SINGLE LAYER PERCEPTRON)

→ A linear model used for binary classification

→ Step Function for activation i.e $f(x) = \begin{cases} 0, & x \leq \alpha \\ 1, & x > \alpha \end{cases}$

$\alpha \Rightarrow$ THRESHOLD VALUE.

→ Contains only input layer and output layer.



used to control weather the Neuron should fire or not fire if all the inputs are '0'

At each output neuron K we have,

$x_0 = \pm 1$ and $w_{0K} = b$ which is weight of bias

$$\hat{y}_K = f\left(\sum_{i=0}^{m} x_i w_{iK}\right) \Rightarrow \text{Output of Kth neuron.}$$

⇒ The weights of the connections are updated through perceptron learning algorithm.

At the K^{th} output neuron the error E_K is Given by

$$E_K = \hat{y}_K - y_K$$ where $\hat{y}_K$ = predicted value

y_K = actual value.

DATE

if $E_k > 0 \Rightarrow$ weights should be reduced

if $E_k < 0 \Rightarrow$ weights should be increased

$$\Rightarrow \boxed{\Delta w_{ik} = -x_i(\hat{y}_k - y_k)}$$

Therefore the weight updation is given by.

$$\boxed{w_{ik} = w_{ik} + \eta \Delta w_{ik} = w_{ik} - \eta \cdot x_i(\hat{y}_k - y_k)}$$

The parameter η is called as learning Rate. It determines how fast should the perceptron learn by controlling the amount of change in weights.

if η is too high $\Rightarrow$ network becomes unstable.

if η is too low $\Rightarrow$ learning will take large amount of time

$$\boxed{0.1 \leq \eta \leq 0.4} \Rightarrow \text{optimal Range}$$

---

* MULTILAYER PERCEPTRON (MULTILAYER FEED FORWARD NETWORK):

→ Neurons are arranged into groups called as layers

→ Contains 3 main layers: ① Input layer
② Hidden layers.
③ Output layer

DATE

⇒ Once a given layer all the Neurons will have same activation function

⇒ For input layer, input will be the actual data we're given. For other layers, input will be activation of previous layer

⇒ Number of neurons in the input layer will be equal to number of features in the input.

⇒ We have two main steps:

① Forward Propogation (Predicting Output)

② Backward Propogation (Reducing Error by tuning weight and bias value).
(Back propogate of error)

⇒ Each neuron can be given a seperate bias value (OR) each layer can have a common bias value, This is based on the implimentation

⇒ BACKPROPOGATION: it is the method of calculating and propagation error from output layer to all Hidden layer.

→ To reduce the error we using "optimisation algorithm" ⇓ (X)

error is calculated using "loss function" ⇒ The goal of optimiser to to minimise the loss function

linear Non-linear

DATE

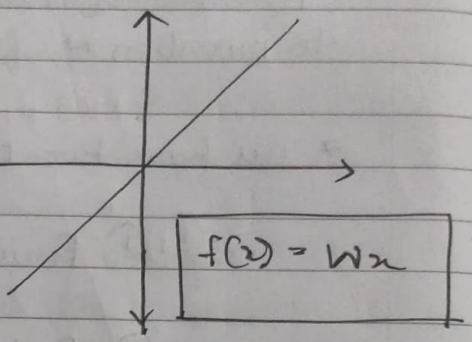
⇒ ACTIVATION FUNCTION : Used to introduce non-linearity in the decision boundary (Because not all problems are linearly Separable)

↳ without activation function the neural network will just perform linear weighted Sum (Similar to linear Regression).

① LINEAR : $f(x) = Wx$

⇒ Identity function

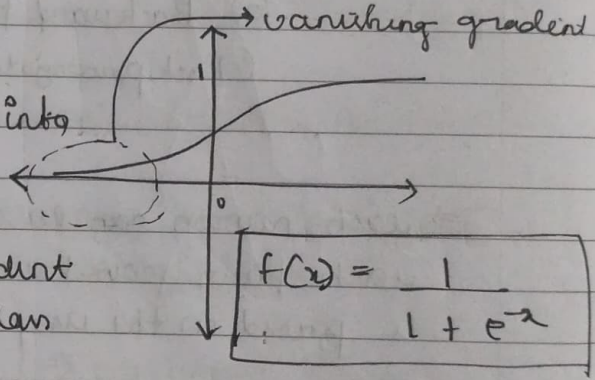
⇒ Used in input layer ?



② SIGMOID :

⇒ Reduce the input into the Range 0 – 1.

⇒ Outputs independent probability for each class

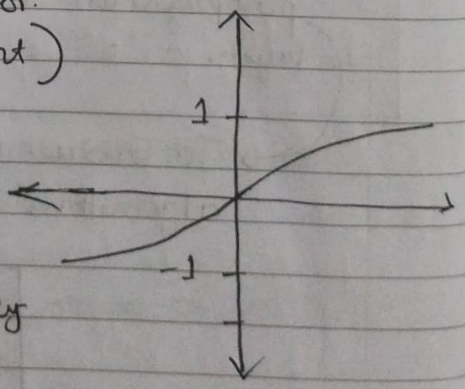


[redacted]: ↳ used in binary classification output layer.

③ TANH : (Hyperbolic Tangent)

⇒ Reduces input to the Range –1 to 1

⇒ it can deal more easily with negative values.



Both Sigmoid and tanh can cause vanishing Gradient problem (For extreme high / low value) ⇒ So we use ReLU.
But it can cause exploding Gradient so need to Scale input

④ SOFTMAX:

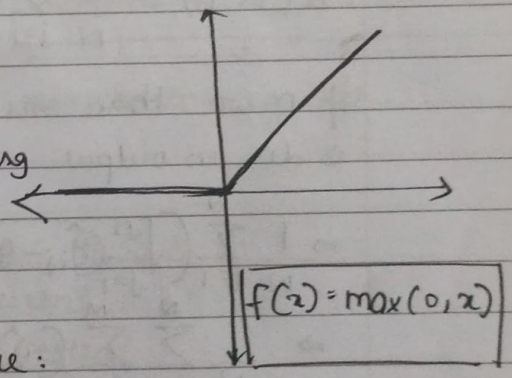
→ provides the probability distribution of each class.
→ used in output layer for multi-class classification

$f(x) = \frac{e^x}{\sum_i e^{x_i}}$ ⇒ if output layer contains M neurons. and X is the output value/activation

⇒ $f(x_j) = \frac{e^{x_j}}{\sum_i^M e^{x_i}}$ (Sum of all probability will be 1)
(class with maximum probability will be the Result).

⑤ RECTIFIED LINEAR UNIT (ReLU):

⇒ overcomes vanishing / exploding gradient problem.
⇒ ReLU is highly used in hidden layer
⇒ Some variant of ReLU are:



(i) Leaky ReLU = $f(x) = \max(0.01x, x)$.

(ii) Parametric ReLU = $f(x) = \max(\alpha x, x)$.

(iii) Exponential ReLU = $f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha(e^x - 1), & \text{if } x < 0 \end{cases}$

$\Downarrow$

$\max(\alpha(e^x - 1), x)$.

DATE

LOSS FUNCTION: It quantifies how close the predicted value is to the Real value.

For Given weights W and Bais B we have $L(W,B)$ as loss function

$\hat{Y}_i = h_{W,B}(X_i)$ ⇒ predicted value for ith Sample
Y_i ⇒ Real value for ith Sample

LOSS FUNCTION FOR REGRESSION

① Mean Squared Error

$$L(W,B) = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2$$

if more than one feature is dir in output

$$\Rightarrow \frac{1}{N}\sum_{i=1}^{N}\left(\sqrt{\sum_{j=1}^{M}(\hat{y}_{ij}-y_{ij})^2}\right)^2.$$

$$\Rightarrow \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}(\hat{y}_{ij}-y_{ij})^2. \;//$$

② Mean Absolute Error

$$L(W,B) = \frac{1}{N}\sum_{i=1}^{N}|\hat{Y}_i - Y_i|$$

$$\Rightarrow \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}|\hat{y}_{ij}-y_{ij}|.$$

③ Root Mean Squared

$$L(W,B) = \sqrt{MSE}$$

LOSS FUNCTION FOR CLASSIFICATION

Cross entropy loss.
(OR)
Negative Log Likelihood.

- Binary
- Categorical.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} Y_i \log \hat{Y}_i + (1-Y_i)\log(1-\hat{Y}_i).$$

$$CCE = -\frac{1}{N}\sum_{i=1}^{1}\sum_{j=1}^{M} Y_{ij}\log(\hat{Y}_{ij}).$$

Hing loss:
↳ designed for SVM
Final should be −1 / 1.

$$L(W,B) \frac{\sum_{i=1}^{N}\max(0, 1-Y_i\times\hat{Y}_i}{N}$$

$$\Rightarrow \frac{1}{N}\sum_{i=1}^{N}$$

classmate

PAGE

DATE

$\sum_{i=1}^{N}\sum_{j=1}^{M} \max(0, y_{ij} \times \hat{y}_{ij})$ => only used for binary classification

MSE and MAE are prone to outliers due to which we can use **HUBER LOSS** => For Regression

$$\hookrightarrow L(W,B) = \begin{cases} \frac{1}{2}(Y_i - \hat{Y}_i)^2, & |Y_i - \hat{Y}_i| \leq \delta \\ \delta |y_i - \hat{Y}_i| - \frac{1}{2}\delta^2, & \text{else.} \end{cases}$$

Here δ is a hyperparameter which depicts the tolerance to outliers.

Huber loss is like a combination MSE and MAE

* GRADIENT DESCENT :

→ An optimisation algorithm for ~~minimising~~ minimising the cost / loss function

(i) Batch Gradient Descent
(ii) Sochastic Gradient Descent
(iii) Mini Batch Gradient Descent

=> In (i) the complete training Set is used for every iteration / epoch.
=> In (ii) Only Single Sample is used for one iteration (ie. Number of iteration / epochs = Number of Samples in training Set → maximum)
=> In (iii) A Batch of Samples used for each iteration from the training dataset.

=> Some other optimisers are : RMSProps, ADADelta, ADAGrad, Adam.

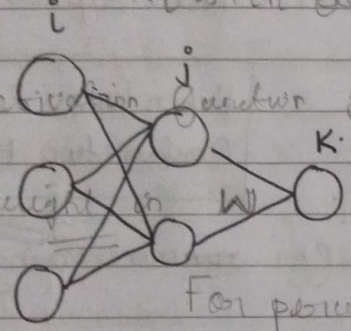
DATE

* BACK PROPOGATION DERIVATION:

⇒ Backpropogation is the method for calculating and propogating error from output layer to the entire network.

⇒ It updates weight of the network using Gradient Descent Algorithm.

⇒ Consider a 3 layer MLP.



⇒ weights between layers i and j is W_{ij} and j and k is W_{jk}

For perceptron $E = \sum_{i=1}^{N} \hat{y}_i - y_i$ (N output neurons)

$$F = \frac{1}{2} \sum_{i=1} (Y_i - \gamma_i)^2$$

(INPUT) (HIDDEN) (OUTPUT)

we will using Sigmoid activation function for the hidden and output layers.

Let i = input layer, if L Layers ⇒

$$O_K = \frac{1}{1 + e^{-I_K}}$$ Hidden $O_j = \frac{1}{1 + e^{-I_j}}$ $i = 1$ Output layer $K = L$

O_K = Output of neuron in layer K

$$I_K = \sum O_j \cdot W_{jK}, \quad I_j = \sum x_i W_{ij} \quad \cdots L-1$$

O_j = Output of neuron in layer K.

$$E = \frac{1}{2} \sum_{i=1}^{N} \left(\sum_{K=1}^{M} g(w_{jK} a_j) \right)$$

Here I_K and I_j are inputs.

DATE

According to gradient descent we have,

$\Delta W \propto -\frac{\partial E}{\partial W}$ (ΔW is change in weights)

($\frac{\partial E}{\partial W}$ is Gradient of loss function).

$\Delta W = -\eta \frac{\partial E}{\partial W}$

η = LEARNING RATE

* On output layer

$E = \frac{1}{2}\sum_{K=0}^{N_0} (Y_K - Y_K)^2 = \frac{1}{2}\sum_{K=0}^{N_0} \left(g\left(\sum_{j=0}^{N_{Hj}} W_{jK} a_j\right) - Y_K\right)^2$

$\Delta W_{jK} = -\eta \frac{\partial E_K}{\partial W_{jK}}$

Consider a perceptron ⇒ No Hidden layer

$\Rightarrow E = \frac{1}{2}\sum_{K=0}^{N_0} (O_T - O_K)^2$

By chain Rule method

$\Delta W_{jK} = -\eta \left[\frac{\partial E}{\partial O_K} \cdot \frac{\partial O_K}{\partial I_K} \cdot \frac{\partial I_K}{\partial W_{jK}}\right]$

$\frac{\partial E}{\partial O_K} = \frac{1}{2}\sum_{K=0}^{N_0} 2(O_K - O_T) \cdot 1$

$\frac{\partial O_K}{\partial I_K} \Rightarrow \frac{\partial}{\partial I_K}\left(\frac{1}{1+e^{-I_K}}\right)$

$= \frac{\partial}{\partial I_K}(1+e^{-I_K})^{-1} = -(1+e^{-I_K})^{-2}(-e^{-I_K})$

$= \frac{e^{-I_K}}{(1+e^{-I_K})^2} = O_K(1-O_K)$

PAGE

classmate

DATE

$$\frac{\partial O_K}{\partial I_K} = O_K(1-O_K)$$

$$\frac{\partial I_K}{\partial W_{jK}} = \frac{\partial}{\partial W_{jK}} \sum O_j W_{jK} = O_j$$

$$\Rightarrow \boxed{\Delta W_{jK} = \eta \cdot O_K \cdot (1-O_K)(O_T - O_K)(O_j).}$$

$$\Rightarrow (\text{new})\, W_{jK} = (\text{old})\, W_{jK} + \Delta W_{jK}$$

For Hidden Layer,

$$\Delta W_{ij} = -\eta \frac{\partial E}{\partial W_{ij}}$$

$$= \frac{\partial E}{\partial O_K} \cdot \frac{\partial O_K}{\partial I_K} \cdot \frac{\partial I_K}{\partial O_j} \cdot \frac{\partial O_j}{\partial I_j} \cdot \frac{\partial I_j}{\partial W_{ij}} \quad (\text{By chain Rule})$$

$$\frac{\partial I_K}{\partial O_j} = W_{jK} \qquad \frac{\partial O_j}{\partial I_j} = O_j(1-O_j)$$

$$\frac{\partial I_j}{\partial W_{ij}} = x_i$$

$$\boxed{\Delta W_{ij} = \eta \cdot x_i \cdot O_K \cdot O_j \cdot (O_T - O_K)(1-O_K)(1-O_j) W_{jK}}$$

From ΔW_{jK} and ΔW_{ij} we have

$$E_K = O_K(1-O_K)(O_T - O_K)$$

$$E_j = O_j(1-O_j) \sum_{K=1}^{N} W_{jK} E_K$$

$$\Rightarrow W_{ij} = W_{ij} + \eta O_i E_j$$

$$\Rightarrow B_i = B_i + \eta(+1)(E_i).$$

classmate

PAGE

DATE

* HYPERPARAMETERS:

⇒ Configuration options that can be varied in order to improve the performance of model

1. No. of. Hidden Layers
2. No. of. Neurons in each layer
3. Activation Function
4. Loss Function
5. Optimizer Function
6. No. of. epochs / Iterations
7. Type of Regularisation
8. Weight initialisation
9. Learning Rate & Momentum.

* BIAS AND VARIANCE: (Trade off).

⇒ when a model is trained it captures the patterns in the training data and tries to generalise it.

⇒ BIAS is the measure of our models ability to learn patterns from data set.

⇒ when model has HIGH BIAS ⇒ did not train well in training set

when model has LOW BIAS ⇒ it has identified neural patterns from Training Set properly.

⇒ VARIANCE is the measure of models fluctuation in performance when there is a change is data set.

DATE

⇒ UNDERFITTING: when a model does not perform well in training data.
(HIGH BIAS).

⇒ OVERFITTING: when a model perform well in train data but it lacks performance in testing data
(LOW. BIAS, HIGH VARIANCE)

⇒ A Good model should have LOW BIAS and LOW VARIANCE.

MACHINE LEARNING: A Algorithm is said to be learning from experience E with Respect to Some task T and performance measure P if the performer. P. increases with increase in experience E.

TYPES OF MACHINE LEARNING:

① SUPERVISED → CLASSIFICATION, REGRESSION

② UNSUPERVISED → CLUSTERING.

③ SEMI-SUPERVISED

④ REINFORCEMENT.

LINEAR SEPERABILITY:

A data is Said to be linearly Seperable if only a single line / Plane / Hyperplane can partition the data into 2 classes.

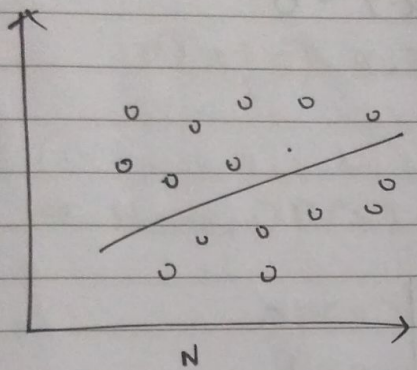
DATE

Eg: Perceptron can only work on a data set that is linearly separable.

→ One of the main reason for MLP improvement

LINEAR REGRESSION:

Consider the following equation $y = \theta_0 x + \theta_1$



Linear regression tries to fit a line of this form on the given dataset.

$$E = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2$$ ⇒ where $\hat{Y}_i$ is the predicted value and Y is the actual value

The error E is called as MEAN SQUARED ERROR. We have to figure this line. This is called LEAST SQUARE REGRESSION.

$$E = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(\theta_0 x_i + \theta_1 - Y_i)^2$$

$$\frac{\partial E}{\partial \theta_0} = \cancel{2} \times \frac{1}{N} \times \sum_{i=1}^{N}(\theta_0 x_i + \theta_1 - Y_i)(x_i + 0 - 0) = 0$$

$$= \theta_0 \overline{X^2} + \theta_1 \bar{X} - \overline{X \cdot Y} = 0 \quad -①$$

$$\frac{\partial E}{\partial \theta_1} = \cancel{2} \times \frac{1}{N} \times \sum_{i=1}^{N}(\theta_0 x_i + \theta_1 - Y_i)(0 + 1 - 0) = 0$$

PAGE

classmate

DATE

$= \theta_0 \bar{x} + \theta_1 - \bar{y} = 0$

$\Rightarrow \theta_1 = \bar{y} - \theta_0 \bar{x}$ -(2)

From (1) and (2) we have.

$\theta_0 \overline{x^2} + (\bar{y} - \theta_0 \bar{x}) \cdot \bar{x} - \overline{x \cdot y} = 0$

$\theta_0 \overline{x^2} + \bar{x} \cdot \bar{y} - \theta_0 \bar{x}^2 - \overline{x \cdot y} = 0$

$\theta_0 = \dfrac{\overline{x \cdot y} - \bar{x} \cdot \bar{y}}{\overline{x^2} - \bar{x}^2}$

$\Rightarrow$

$$\theta_0 = \frac{\overline{x \cdot y} - \bar{x} \cdot \bar{y}}{\overline{x^2} - \bar{x}^2}$$

$$\theta_1 = \bar{y} - \theta_0 \cdot \bar{x}$$

$\Rightarrow$ Why Huber Loss for Regression?

MSE is Highly Sensitive to outliers because we Square the over all error value. MAE just take the absolute value of error it give same weightage to all errors due to which outlier can be completely Neglected. Huber Loss Provide Balance (S).

DATE

* BAYESIAN STATISTICS :
  ↳ (methods that used Bayes Theorm)

⇒ BAYES THEORM : Finds the probability of an event occuring, given the probability of another event that has already occured
  ↳ (called as conditional probability)

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

⇒ Here, ·) P(B) is called as evidence.

·) P(A) is called as prior probability (i.e, probability occurance of event A without considering event B).

·) P(A|B) is called as posterior Probability (i.e, probability of occurance of event A with considering event B)

⇒ we can P(Y|X) using above method.

Y ⇒ Target class and X = Feature Vector.

$$\Rightarrow P(Y_j \mid X_1, X_2, \ldots, X_M) = \frac{P(Y_j|X_1) \cdot P(Y_j|X_2) \cdots P(Y_j|X_M) \cdot P(Y_j)}{P(X_1) \cdot P(X_2) \cdots P(Y_M)}$$

↳ since same evidence is same.

for all target we can ignore this

DATE

⇒ Result will be class j having max $P(Y_j|X)$.

⇒ There are various variation of Naive Bayes algorithms like

(i) Gaussian / Normal NB
(ii) Multinomial NB
(iii) Bernoulli NB
⋮

⇒ Important assumption for Naive Bayes algorithm is all the features are independent.

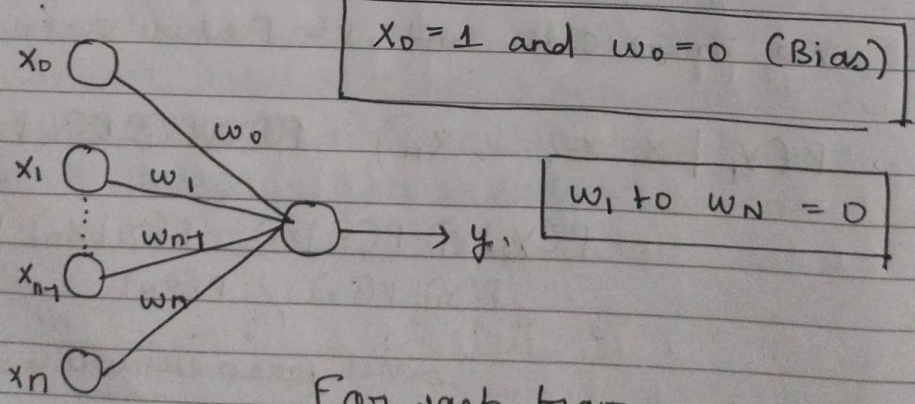
i.e $P(X_i | X_j) = P(X_i)$ (X)

---

* HEBBIAN LEARNING RULE (HEBB NETWORK)

→ Inputs can only be 1 and −1

→ Initialise all weights and bias to 0. in Starting



For each training sample we do the following

DATE

$\Rightarrow \quad \Delta w_i = x_i y \quad \text{and} \quad w_i = w_i + \Delta w_i$

— x —

PROBLEMS:

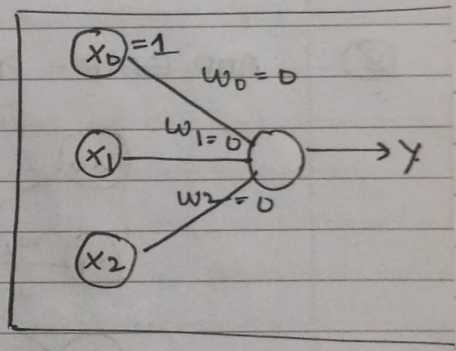
① AND GATE WITH HEBB NETWORK:

(B_x)				$(\Delta B w)$		
X_0	X_1	X_2	Y	Δw_0	Δw_1	Δw_2
1	-1	-1	-1	-1	1	1
1	-1	1	-1	-1	1	-1
1	1	-1	-1	-1	-1	1
1	1	1	1	1	1	1
				-2	2	2

initial value of $w_0 = 0$ (Bw) $X_0 = 1$ (B_x).

$w_1 = 0$

$w_2 = 0$.



$\Rightarrow$ Final weights are $w_0 = -2$

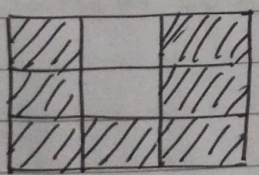
$w_1 = 2$

$w_3 = 2$.

— x —

V and T.

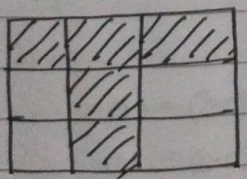
② 3 x 3 IMAGE OF ■ WITH HEBB NETWORK.



$\leftrightarrow$

1	-1	1
1	-1	1
1	1	1

(Target = 1)



$\leftrightarrow$

1	1	1
-1	1	-1
-1	1	-1

(Target = -1)

initial value of w_0 to w_9 is 0 and $x_0 = 1$

DATE

X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	Y
1	1	-1	1	1	-1	1	1	1	1	1
1	1	1	1	-1	1	-1	-1	1	-1	-1

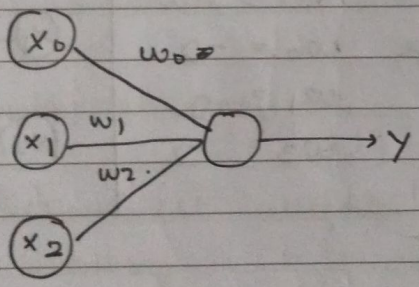
Δw_0	Δw_1	Δw_2	Δw_3	Δw_4	Δw_5	Δw_6	Δw_7	Δw_8	Δw_9
1	1	-1	1	-1	-1	1	1	1	1
-1	-1	-1	-1	1	-1	1	1	-1	1
0	0	-2	0	2	-2	2	2	0	2.

⇒ Final weights are

w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
0	0	-2	0	2	-2	2	2	0	2.

—— x ——

(3) AND GATE - PERCEPTRON :



$X_0 = 1$ (BIAS INPUT)

$w_0 = -0.5$, $w_1 = 0.2$, $w_2 = 0.3$
(BIAS WEIGHT)

X_0	X_1	X_2	Y	$\hat{Y}$	E
1	0	0	0	$(1)(-0.5)+(0)(0.2)+(0)(0.3) = 0$	0
1	0	1	0	$(1)(-0.5)+(0)(0.2)+(1)(0.3) = 0$	0
1	1	0	0	$(1)(-0.5)+(1)(0.2)+(0)(0.3) = 0$	0
1	1	1	1	$(1)(-0.5)+(1)(0.2)+(1)(0.3) = 0$	1

DATE

$w_0 = -0.5 \oplus (0.4)(-1)(1) = -0.1$
$w_1 = 0.2 \oplus (0.4)(-1)(1) = 0.6$
$w_2 = 0.3 \oplus (0.4)(-1)(1) = 0.7$

ITERATION #2: $(w_0 = -0.1, w_1 = 0.6, w_2 = 0.7)$

X_0	X_1	X_2	Y	$\hat{y}$	E
1	0	0	0	$(1)(-0.1)+(0)(0.6)+(0)(0.7) = 0$	0
1	0	1	0	$(1)(-0.1)+(0)(0.6)+(1)(0.7) = 1$	-1
1	1	0	0	$(1)(-0.5)+(1)(0.6)+(0)(0.3) = 1$	1
1	1	1	1	$(1)(-0.9)+(1)(0.2)+(1)(0.3) = 0$	-1

$w_0 = -0.1 - (0.4)(-1)(1) = -0.5$
$w_1 = 0.6 - (0.4)(-1)(0) = 0.6$
$w_2 = 0.7 - (0.4)(-1)(1) = 0.3$

$w_0 = -0.5 - (0.4)(1)(1) = -0.9$
$w_1 = 0.6 - (0.4)(1)(1) = 0.2$
$w_2 = 0.3 - (0.4)(1)(0) = 0.3$

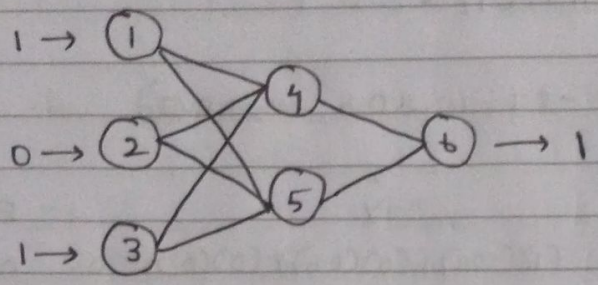
$w_0 = -0.9 - (0.4)(-1)(1) = -0.5$
$w_1 = 0.2 - (0.4)(-1)(1) = 0.6$
$w_2 = 0.3 - (0.4)(-1)(1) = 0.7$

⇒ weights are ITERATION #2 are $w_0 = -0.5$
$w_1 = 0.6$
$w_2 = 0.7$

—— x ——

DATE

④ MULTILAYER PERCEPTRON:



$w_{14} = 0.2$ $w_{15} = -0.3$ $w_{46} = -0.5$
$w_{24} = 0.4$ $w_{25} = 0.1$ $w_{56} = -0.2$
$w_{34} = -0.5$ $w_{35} = 0.2$

$b_4 = 0.218$ $b_5 = 0.194$ $b_6 = 0.408$

ITERATION # 1

$$I_4 = (1)(0.2) + (0)(0.4) + (1)(-0.5) + (1)(0.218)$$
$$= -0.082$$

$$O_4 = \frac{1}{1 + e^{0.082}} = 0.479$$

$$I_5 = (1)(-0.3) + (0)(0.1) + (1)(0.2) + (1)(0.194)$$
$$= 0.094$$

$$O_5 = \frac{1}{1 + e^{-0.094}} = 0.523$$

$$I_6 = (0.479)(-0.5) + (0.523)(-0.2) + (1)(0.408)$$
$$= 0.064$$

$$O_6 = \frac{1}{1 + e^{-0.064}} = 0.516$$

classmate

PAGE

DATE

$E_6 = (0.516)(1-0.516)(1-0.516) = 0.121$

$E_5 = (0.523)(1-0.523)(-0.2 \times 0.121) = -0.006$

$E_4 = (0.479)(1-0.479)(-0.5 \times 0.121) = -0.015$

[redacted]

$w_{14} = 0.2 + (0.2)(1)(-0.015) =$

$w_{24} = 0.4 + (0.2)(0)(-0.015) =$

$w_{34} = -0.5 + (0.2)(1)(-0.015) =$

$w_{15} = -0.3 + (0.2)(1)(-0.006) =$

$w_{25} = 0.1 + (0.2)(0)(-0.006) =$

$w_{35} = 0.2 + (0.2)(1)(-0.006) =$

$w_{46} = -0.5 + (0.2)(0.479)(0.121) =$

$w_{50} = -0.2 + (0.2)(0.523)(0.121) =$

$b_4 = 0.218 + (0.2)(1)(-0.15) =$

$b_5 = 0.194 + (0.2)(1)(-0.006) =$

$b_6 = 0.408 + (0.2)(1)(0.121) =$

PAGE

classmate

DATE

NAIVE BAYES

⑤ $P(YES) = \frac{7}{10}$ $P(NO) = \frac{3}{10}$

*

CGPA	P(YES)	P(NO)
≥ 9	3/7	1/3
≥ 8	4/7	0/3
< 8	0/7	2/3

*

INTERACTIVNESS	P(YES)	P(NO)
YES	5/7	1/3
NO	2/7	2/7

*

PRACTICAL-KNOW	P(YES)	P(NO)
EXCELLENT	2/7	0/3
GOOD	4/7	1/3
AVERAGE	1/7	2/3

*

COMM-SKILL	P(YES)	P(NO)
GOOD	4/7	1/3
MODERATE	3/7	0/3
POOR	0/7	2/3

$X = \{ \geq 9, YES, AVERAGE, GOOD \}$.

$P(YES|X) = \frac{3}{7} \times \frac{5}{7} \times \frac{1}{7} \times \frac{4}{7} \times \frac{7}{10} = 0.0174$

$P(NO|X) = \frac{1}{3} \times \frac{1}{3} \times \frac{2}{3} \times \frac{1}{3} \times \frac{3}{10} = 0.0074$

⇒ **RESULT : YES**

classmate

PAGE

DATE

LINEAR REGRESSION

⑥

X	Y	X·Y	X^2
1	4	4	1
2	5	10	4
3	8	24	9
4	2	8	16

$\overline{X} = 2.5$ $\overline{Y} = 4.75$ $\overline{X^2} = 7.5$

$\overline{XY} = 11.5$ $\overline{X} \cdot \overline{Y} = 11.875$

$$\theta_0 = \frac{11.875 - 11.5}{6.25 - 7.5} = \frac{0.375}{-1.25} = -0.3$$

$\theta_0 = -0.3$

$\theta_1 = 4.75 + 0.3 \times 2.5 = 5.5$

$\theta_1 = 5.5$

$Y = -0.3X + 5.5$

⑦ LOSS FONCTION SUM

DATE

* CONVOLUTION NEURAL NETWORK:

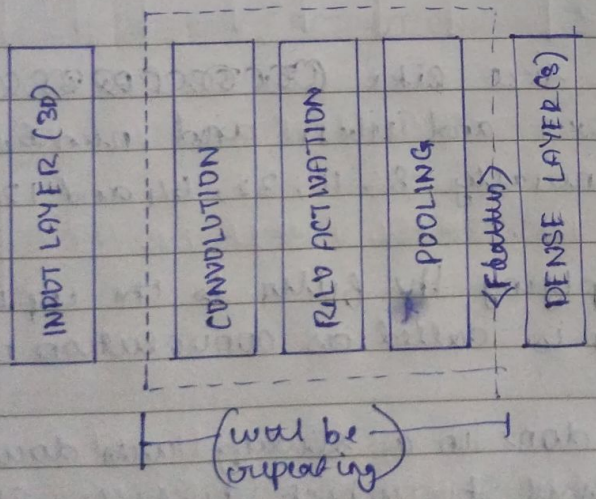
→ A fully connected neural network creates massive number of connection while modelling image data. (i.e large number of neurons and large number of connections).

→ disadvantage of Normal ANN to work with image

→ In CNN, we can arrange neurons in 3-D structure Based on

(i) width ⇒ image width in PX
(ii) Height ⇒ image height in PX
(iii) Depth ⇒ Number of Channels (RGB = 3, GRAY = 1).

→ common pattern in CNN architectures are:



→ this general pattern of a CNN model is convolution layer followed by pooling layer

DATE

⇒ CONVOLUTION LAYER:

Some of the important concepts in a convolution layer are:

(i) Filters
(ii) Strides
(iii) Zero Padding
(iv) Convolution operation

→ A convolution layer can contain a set of filters of fixed size

→ These filters are also called as Kernels. Two Hyperparams are used in a convolution layer to control the filters / kernels they are.

(i) number_of_filters
(ii) filter_size

→ Filters have common size like 3×3, 5×5, 7×7, 9×9 and 11×11 and number of filters used are generally 8, 16, 32, 64 and 128.

→ The process of applying the filter to the input of a convolution layer is called as convolution operation

→ This operation is done in a sliding window fashion. The amount by which the filter should move is controlled by Stride value

→ The main operation performed in convolution is DOT PRODUCT.

classmate

PAGE

DATE

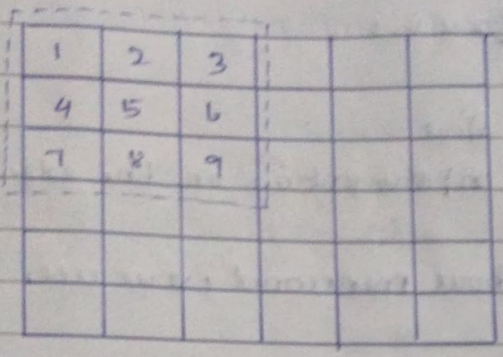
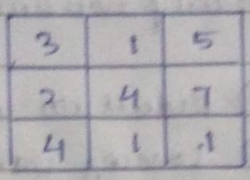


IMAGE OF SIZE 6x6



FILTER OF SIZE 3x3

=> 1x3 + 2x1 + 5x3 + 4x2 + 5x4 + 6x7 + 7x4 + 8x1 + 9x1

= 135

This will be repeated until whole image is convolved based on the Stride value.

→ The output obtained as a result of the convolution operation from a convolution layer is called as FEATUREMAP / ACTIVATIONMAP.

→ Each of the filters present in a convolution layer will identify a specific feature from an image → Filters extract feature from the image.

ZERO PADDING:

→ Applying the convolution operation reduces the size of our image. If image size is N x N and filter size is F x F the output image will be (N-F+1) x (N-F+1).

$I_w - F_w + 1 = A_w$

$\hookrightarrow (I_w + P) - F_w + 1 = I_w$

$P = \frac{F_w - 1}{2}$

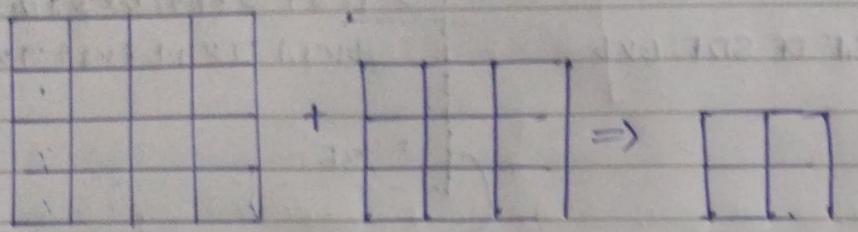
DATE

→ Two main problem we face is:

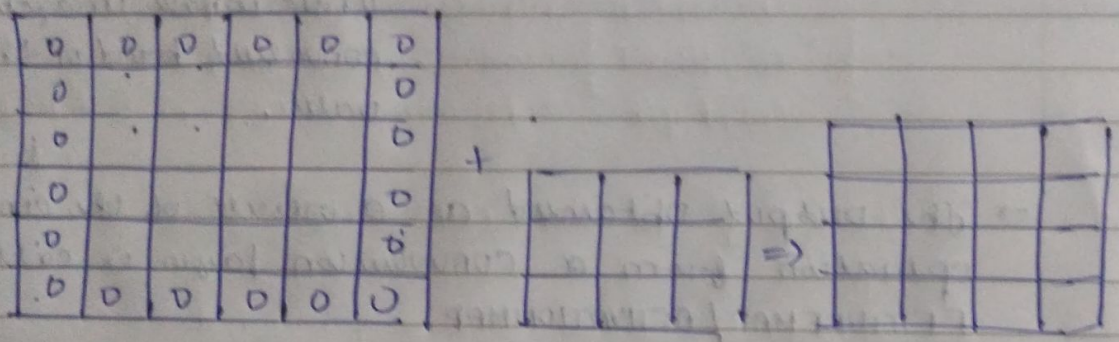
(i) Reduction in the Size

(ii) Loosing valuable information on the edge.

→ So two overcome the above mentioned issue we use zero padding.



4 X 4 IMAGE 3X3 FILTER 2 X 2 ACTIVATION MAP



4X4 IMAGE + 0-PADS 3 X 3 FILTE 4 X 4 ACTIVATION

(X) WHY ReLU ACTIVATION IN CONVOLUTION LAYER ???

GIVEN:

Image Size : N x N

Filter Size : F x F

Padding : P

Stride : S

→ Size of activation $\left(\frac{N - F + 2P + 1}{S}\right) \times \left(\frac{N - F + 2P + 1}{S}\right)$.

classmate

PAGE

DATE

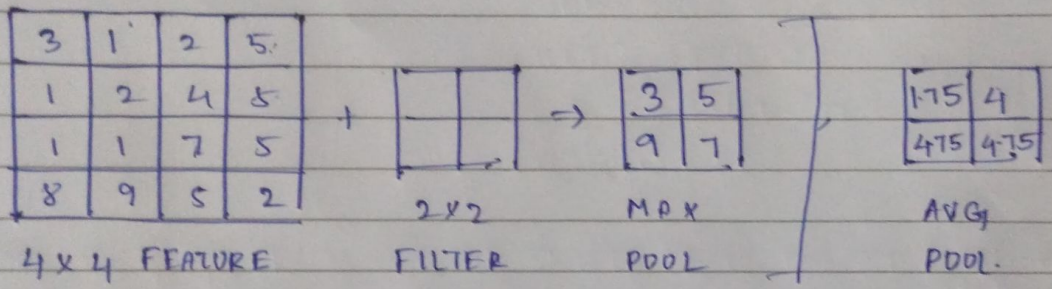
⇒ POOLING LAYERS:

→ pooling layers are inserted between successive convolution layers to Reduce the Size of activation map. gradually the process is called down sampling

→ Two main types of pooling are.

① Max Pooling ② Average Pooling.

→ Zero-padding are not be used in pooling layer.



GIVEN:

Feature Map of Size : N × N.
Filter Size : F × F
Stride : S.

⇒ Output Size : $\left(\frac{N-F}{S}+1\right) \times \left(\frac{N-F}{S}+1\right)$

* APPLICATIONS OF CNN:	* SOME CNN ARCHITECTURE
→ Image Classification	→ VGG16
→ NLP Application	→ VGG 19
→ Image Segmentation	→ ResNet 50
→ Image Generation	→ AlexNet
→ Image Feature extraction	→ LeNet.