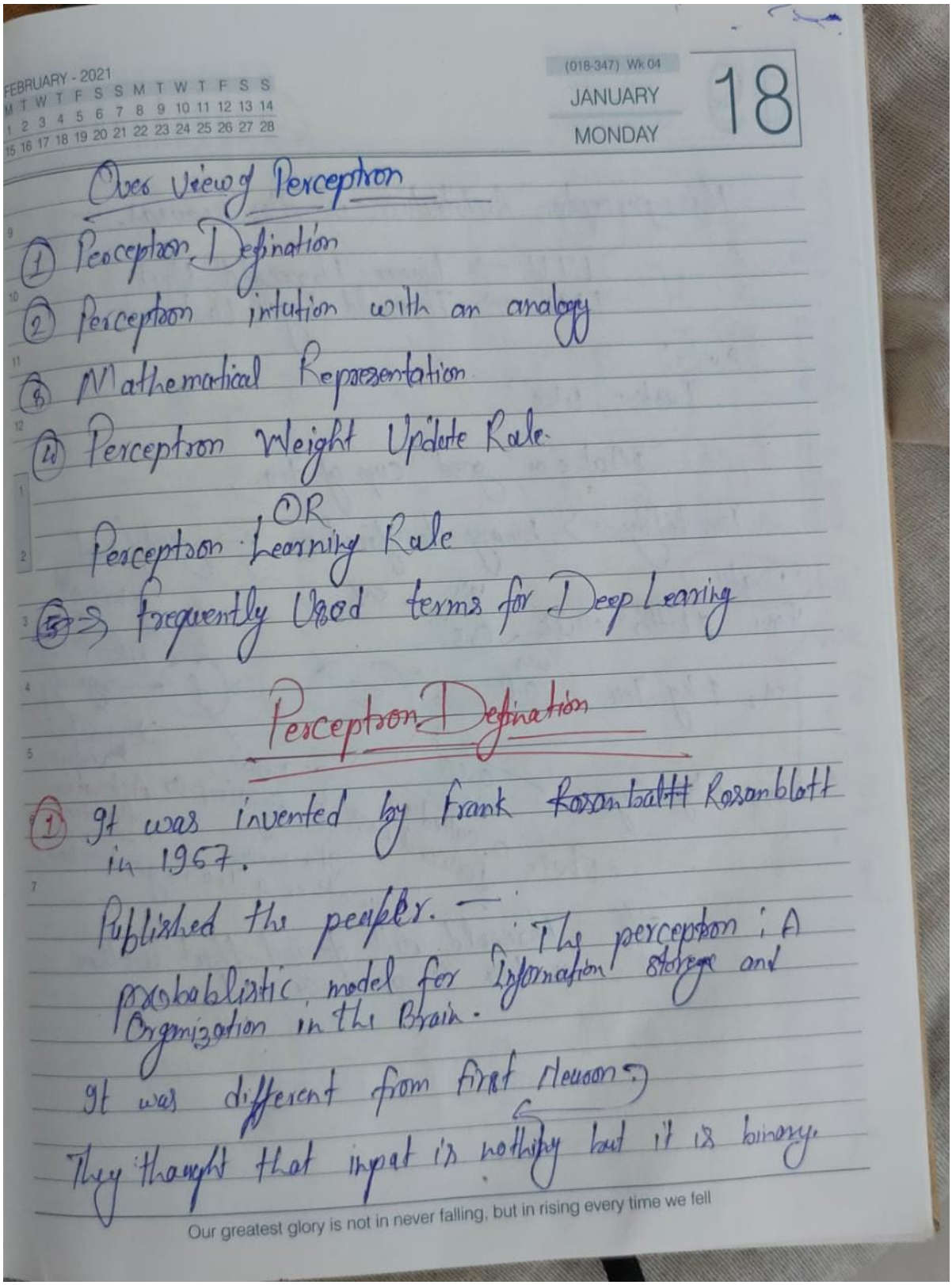
Perceptron


FEBRUARY - 2021
M T W T F S S M T W T F S S
1 2 3 4 5 6 7 8 9 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25 26 27 28
(018-347) Wk 04
JANUARY
MONDAY
18
Over View of Perceptron
1 Perceptron Defination
2 Perceptron intution with an analogy
3 Mathematical Representation
4 Perceptron Weight Update Rule
OR
Perceptron Learning Rule
5 => Frequently Used terms for Deep Learning
Perceptron Defination
1 It was invented by Frank ~~Rosanbaltt~~ Rosanblatt in 1967.
Published the peaper. —
The perceptron : A probablistic model for Information storage and Organization in the Brain.
It was different from first neuron :)
They thought that input is nothing but it is binary.
Our greatest glory is not in never falling, but in rising every time we fell

19 (019-346) Wk 04
JANUARY
TUESDAY

JANUARY - 2021
M T W T F S S M T W T F S S
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 27 28 29 30 31

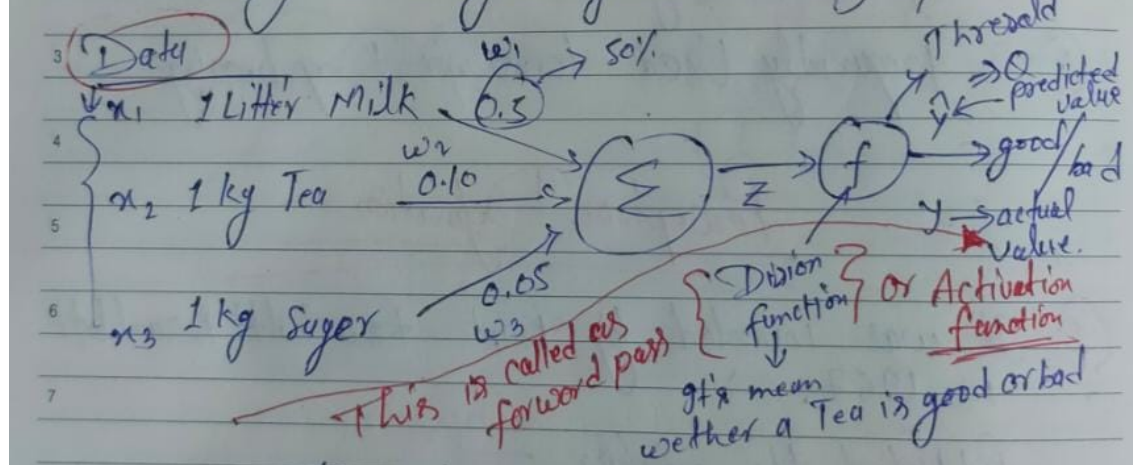
This perceptron Architecture is Also knowas. —

LTU → Linear Threshold Unit
TLU → Threshold Logic Unit.

Ex. -;
Task -; like

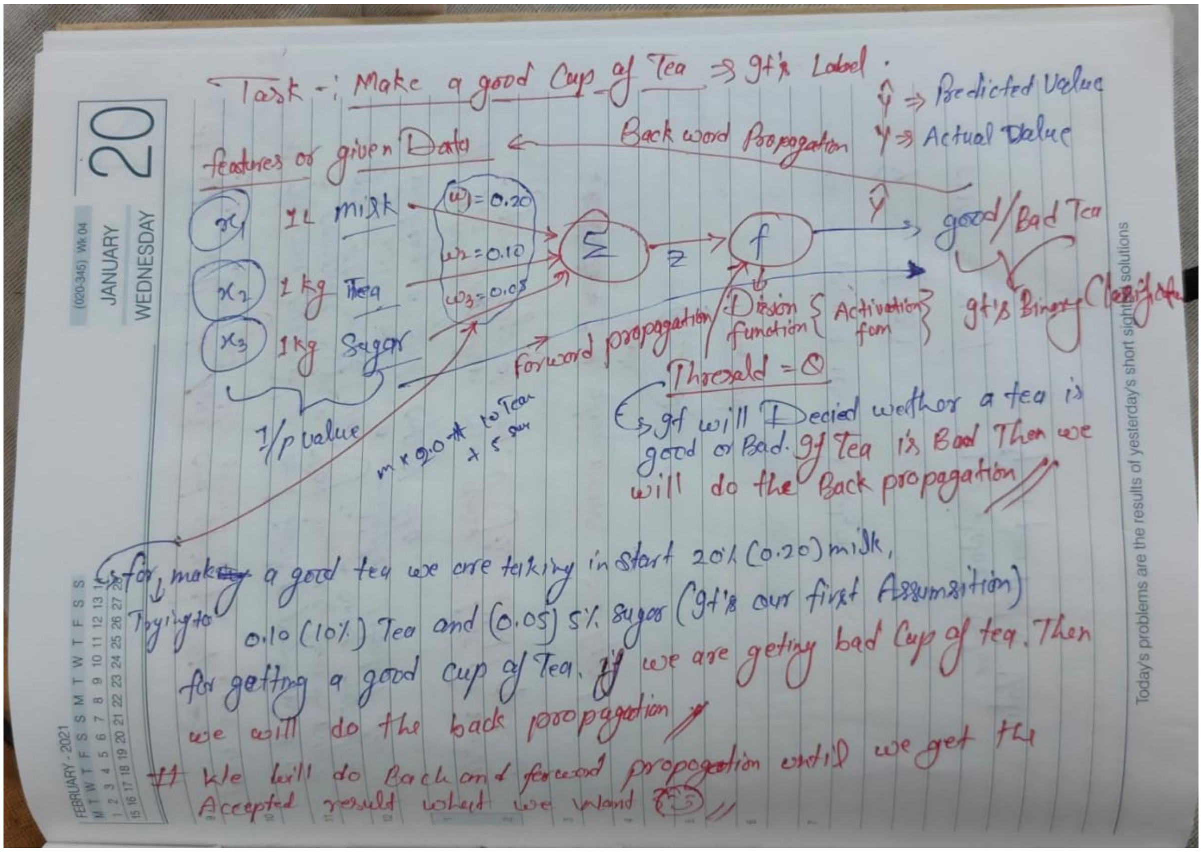
Make a good cup of tea.

Possibility → binary classification → good/bad.



Threshold will decied that wether a tea is good or bad.

Worry is the interest paid on trouble before it falls due


Task -: Make a good Cup of Tea ⇒ It's Label.
ŷ ⇒ Predicted Value
y ⇒ Actual Value
features or given Data
Back word Propagation
x1 1L milk
x2 1kg Tea
x3 1kg Sugar
w1 = 0.20
w2 = 0.10
w3 = 0.05
Σ
z
f
ŷ
good/Bad Tea
I/p value
forward propagation
Dision function { Activation fun }
It's Binary Classification
Threshold = 0
It will Decied wether a tea is good or Bad. If Tea is Bad Then we will do the Back propagation
for, making a good tea we are taking in start 20% (0.20) milk, 0.10 (10%) Tea and (0.05) 5% sugar (It's our first Assumsition) Trying to for getting a good cup of Tea. If we are geting bad Cup of tea. Then we will do the back propagation
We will do Back and forward propogation until we get the Accepted result what we want

21 (021-344) Wk 04
JANUARY
THURSDAY

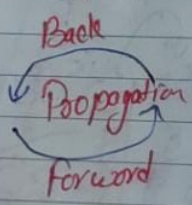
JANUARY - 2021
M T W T F S S M T W T F S S
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 27 28 29 30 31

We will do back and forward propagation until we get the result what we want

forward propagation and back propagation. It is a complete cycle. May be we ~~will~~ can't get the result (O/P) in a single repeation it will make a multiple iteration for making a good cup of tea.

The Repeation are going on here for making a good cup of tea. is called epoch

1 iteration means → 1 Epoch



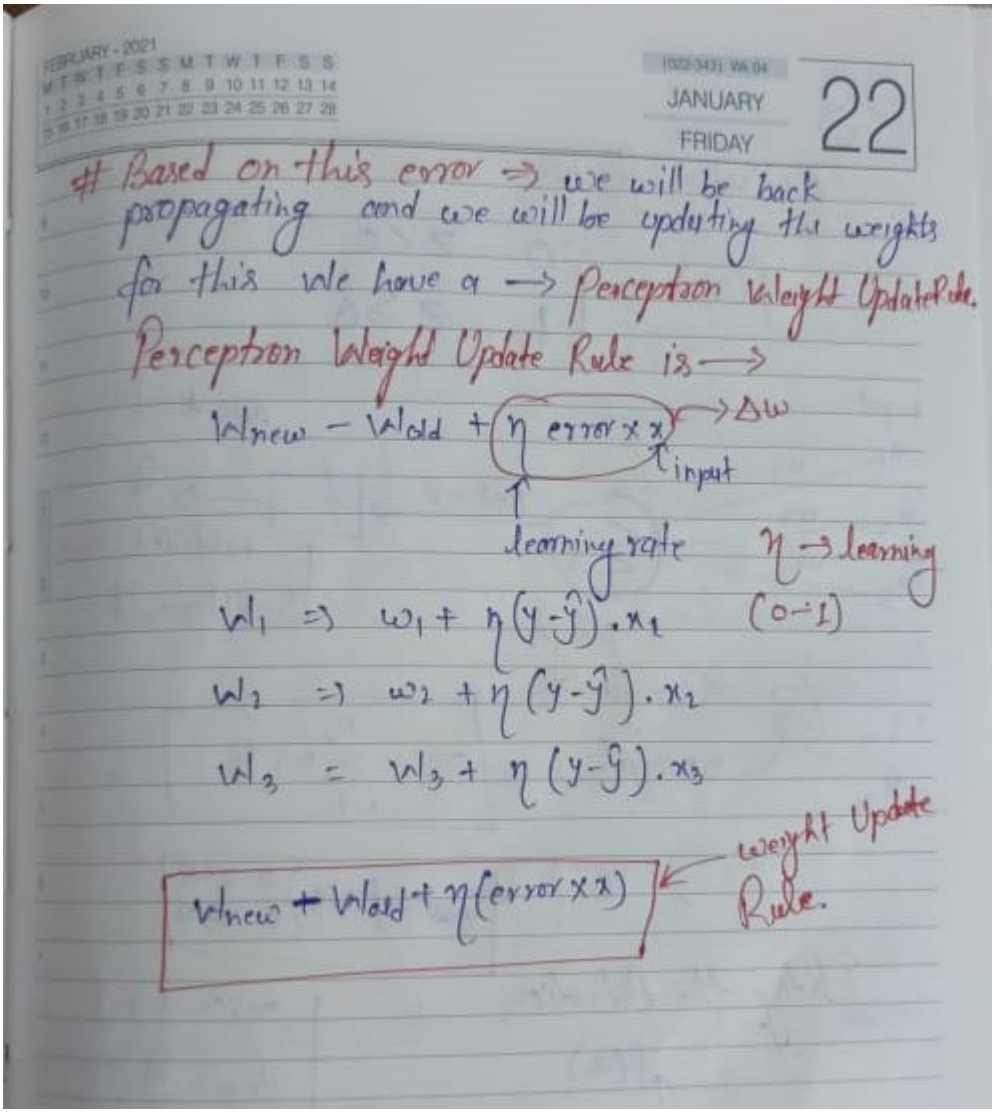
$$z = w_1x_1 + w_2x_2 + w_3x_3$$

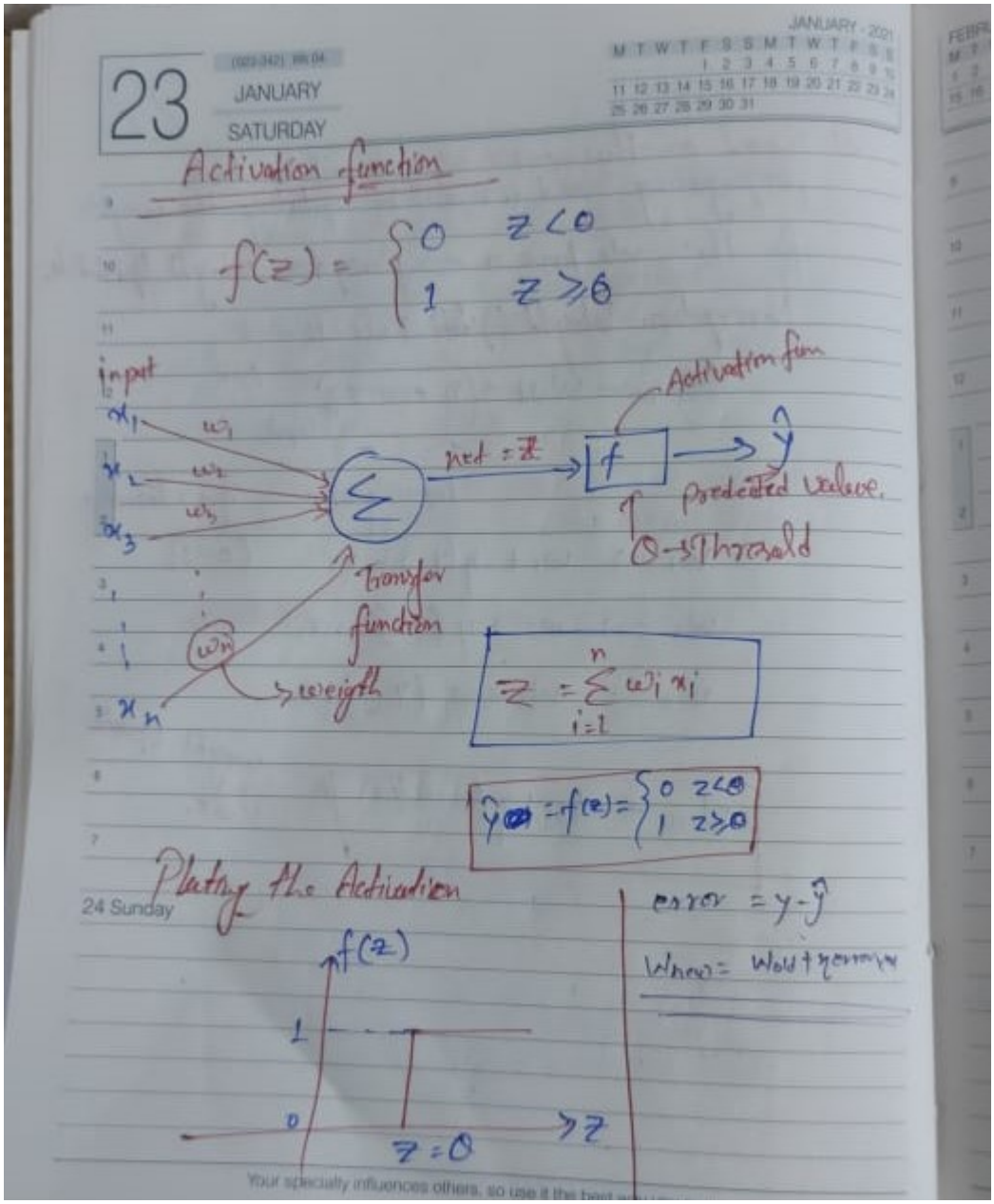
$$\hat{y} = f(z)$$

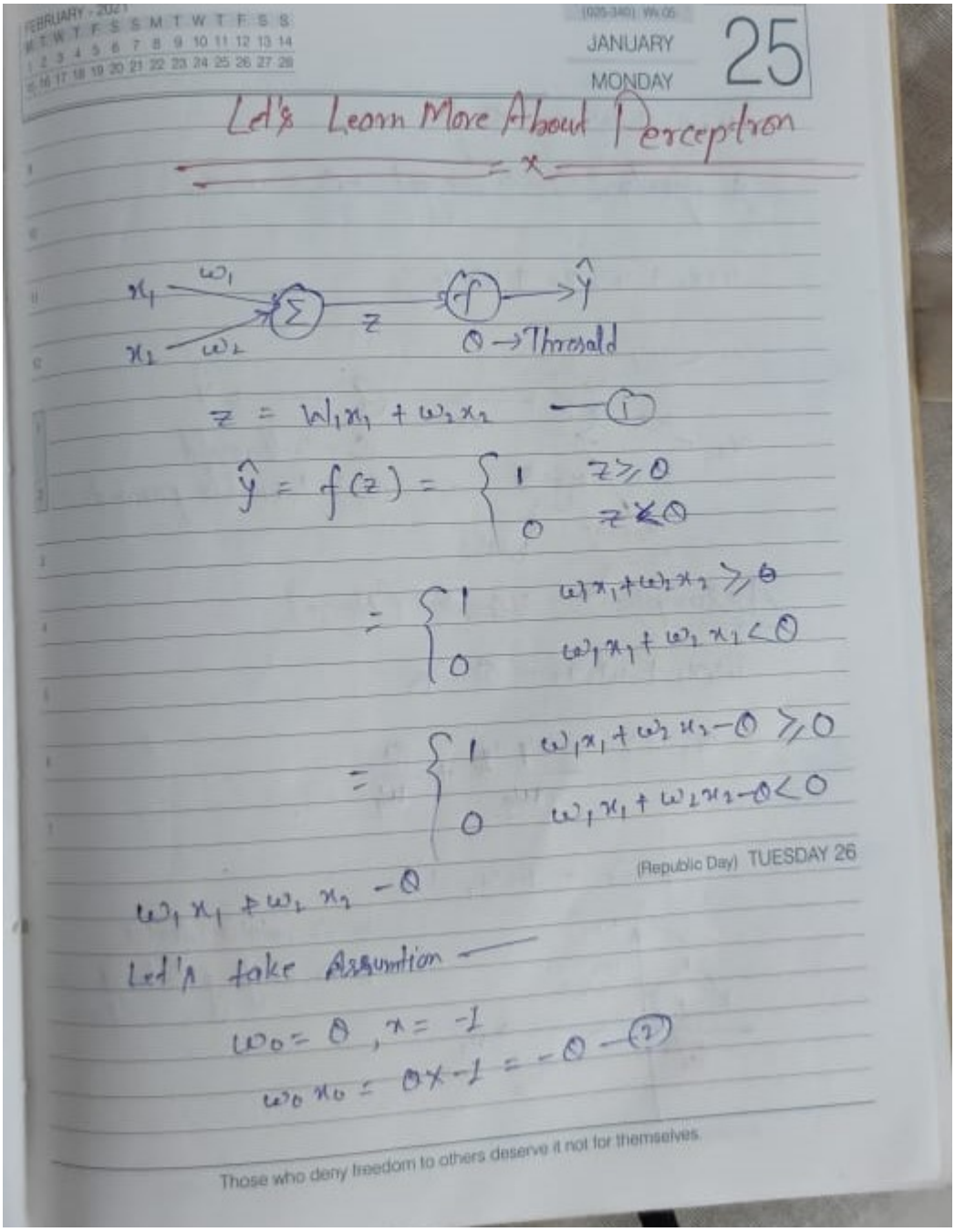
$$error = y - \hat{y}$$

y => Actual Value
$\hat{y}$ => predicted Value.

Life is great; make the most out of it. Be an Optimist


JANUARY
FRIDAY
22
Based on this error ⇒ we will be back propagating and we will be updating the weights for this we have a → Perceptron Weight Update Rule.
Perceptron Weight Update Rule is →
$w_{new} = w_{old} + \eta \, error \times x$
Δw
input
learning rate
η → learning (0-1)
$w_1 \Rightarrow w_1 + \eta (y - \hat{y}) . x_1$
$w_2 \Rightarrow w_2 + \eta (y - \hat{y}) . x_2$
$w_3 = w_3 + \eta (y - \hat{y}) . x_3$
$w_{new} = w_{old} + \eta (error \times x)$
weight Update Rule.


23
JANUARY
SATURDAY
Activation function
$f(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$
input
x_1
x_2
x_3
x_n
w_1
w_2
w_3
w_n
Σ
net = z
f
$\hat{y}$
Activation fun
predicted value.
0-threshold
transfer function
weight
$z = \sum_{i=1}^{n} w_i x_i$
$\hat{y} = f(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$
Plotting the Activation
24 Sunday
f(z)
1
0
z = 0
z
error = $y - \hat{y}$
Wnew = Wold + error
Your specialty influences others, so use it the best


JANUARY
MONDAY
25
Let's Learn More About Perceptron
x_1
x_2
w_1
w_2
Σ
z
f
$\hat{y}$
$\theta \rightarrow$ Threshold
$z = w_1 x_1 + w_2 x_2$ — (1)
$\hat{y} = f(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$
$= \begin{cases} 1 & w_1 x_1 + w_2 x_2 \geq \theta \\ 0 & w_1 x_1 + w_2 x_2 < \theta \end{cases}$
$= \begin{cases} 1 & w_1 x_1 + w_2 x_2 - \theta \geq 0 \\ 0 & w_1 x_1 + w_2 x_2 - \theta < 0 \end{cases}$
(Republic Day) TUESDAY 26
$w_1 x_1 + w_2 x_2 - \theta$
Let's take Assumption —
$w_0 = \theta$, $x = -1$
$w_0 x_0 = \theta \times -1 = -\theta$ — (2)
Those who deny freedom to others deserve it not for themselves

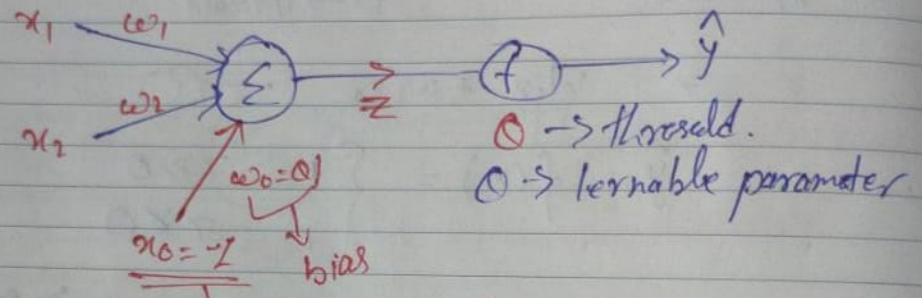
27 (027-338) Wk 05
JANUARY
WEDNESDAY

JANUARY - 2021
M T W T F S S M T W T F S S
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 27 28 29 30 31

$[w_1 x_1 + w_2 x_2 - \theta]$

In Equation (2) → we got $-\theta = w_0 x_0$

$w_1 x_1 + w_2 x_2 + w_0 x_0$



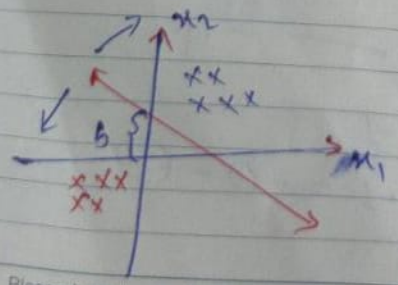
θ → Threshold.
θ → learnable parameter

Assumption → Constant (Noise)

$w_1 x_1 + w_2 x_2 - \theta = 0$

$x_2 \Rightarrow \frac{-w_1 x_1}{w_2} + \frac{\theta}{w_2}$

$x_2 \Rightarrow -m x_1 + b$



m ⇒ rotation factor
b ⇒ shifting factor / translation factor

Blessed are those who can give without remembering and take without forgetting.

FEBRUARY - 2021

M	T	W	T	F	S	S	M	T	W	T	F	S	S
1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28

(028-337) Wk 05
JANUARY
THURSDAY
28

To create a model more generic, Here they are trying to introduce Noise and provide a weight called bias.

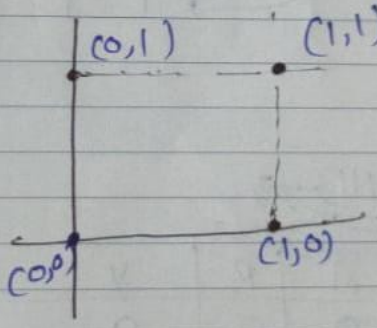
Perceptron Draw Back

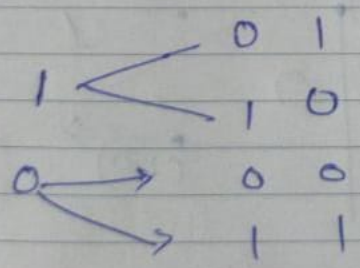
It is only work for linear ~~separat~~ seprated data

Example for XOR

x	y	o/p
0	0	0
0	1	1
1	0	1
1	1	0

Graph:





Here we are not able to seprate the data

Here Single layer perceptron is not working

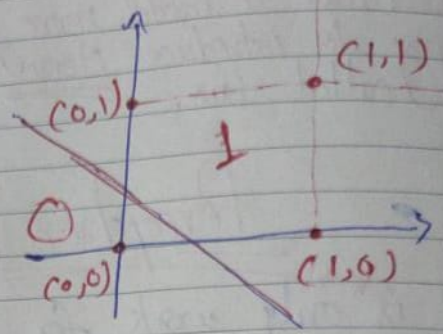
Loneliness comes when one forget that God is the supreme companion

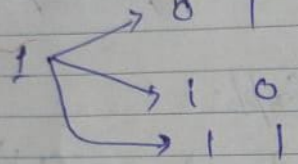
29 (029-336) Wk 05
JANUARY
FRIDAY

JANUARY
M T W T F S S M T W T F
1 2 3 4 5 6 7 8
11 12 13 14 15 16 17 18 19 20 21 22
25 26 27 28 29 30 31

OR ⇒

A	B	y
0	0	0
0	1	1
1	0	1
1	1	1



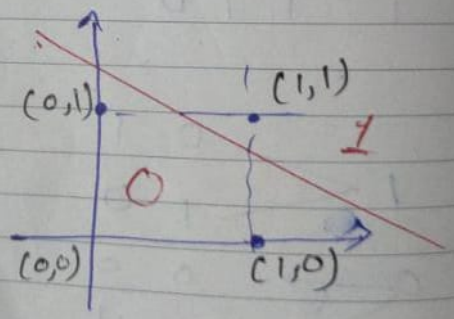


0 → 0 0

Here it is easily we seprate the data

AND ⇒

A	B	y
0	0	0
0	1	0
1	0	0
1	1	1



Here also we can easily seprate the data.

Quality consciousness is important for the growth of an organization

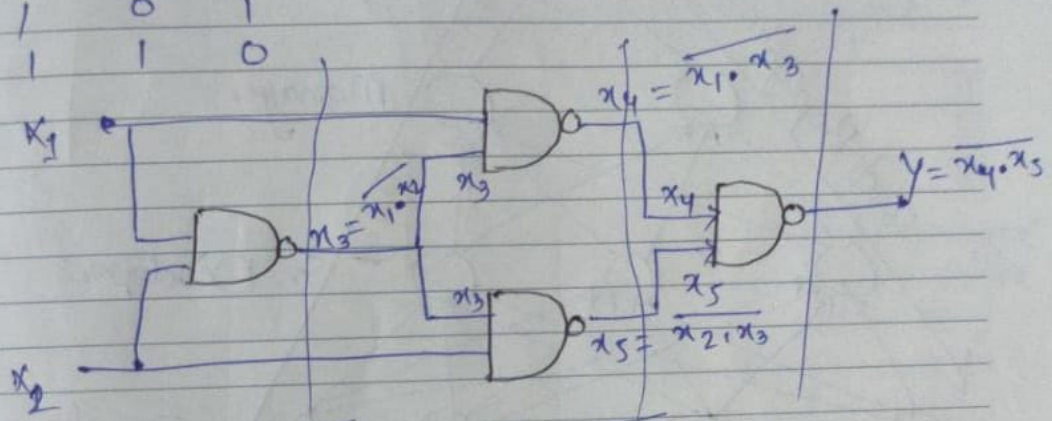
RUARY - 2021
W T F S S M T W T F S S
3 4 5 6 7 8 9 10 11 12 13 14
17 18 19 20 21 22 23 24 25 26 27 28

(030-335) Wk 05
JANUARY
SATURDAY
30

We can implement XOR with the help of NAND GATE

NAND GATE ⇒

X_1	X_2	Y
0	0	1
0	1	1
1	0	1
1	1	0



x_1	x_2	$x_3 = \overline{x_1 \cdot x_2}$	$x_4 = \overline{x_1 \cdot x_3}$	$x_5 = \overline{x_2 \cdot x_3}$	$y = \overline{x_4 \cdot x_5}$
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	0

Sunday 31

Assume all the level of NAND GATE is Neuron
So we have a multiple NAND so we can say it's multilayer Perceptron

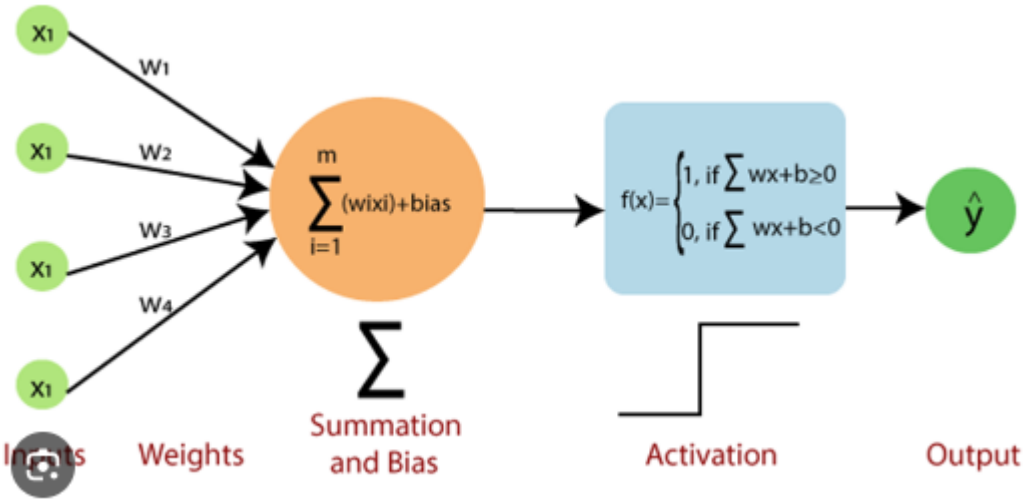
It does not cost a penny, to speak true and sweet words

Type *Markdown* and LaTeX: α^2

Preceptron Precatical

In [1]:
```
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import joblib #saving the model in binary formate
from matplotlib.colors import ListedColormap

plt.style.use("fivethirtyeight")
```

In [ ]:
```

```



Perceptron skeleton :)

In [ ]:
```

```

In [ ]:
```

```

In [2]:

```python
#the blue Print of perceptron

class Perceptron:
    def __init__(self,eta: float=None, epochs: int=None):
        #eta stand for learning rate
        #epochs =complete cycle of (forword_propogation+backword_propogati
        try:
            pass
        except Exception as e:
            raise e

    def _z_outcome(self):
        try:
            pass
        except Exception as e:
            raise e

    def activation_funtion(self):
        try:
            pass
        except Exception as e:
            raise e

    def fit(self):
        try:
            pass
        except Exception as e:
            raise e

    def predict(self):
        try:
            pass
        except Exception as e:
            raise e

    def total_loss(self):
        try:
            pass
        except Exception as e:
            raise e

    def _create_dir_return_path(self):
        try:
            pass
        except Exception as e:
            raise e

    def save(self):
        try:
            pass
        except Exception as e:
            raise e

    def load(self):
        try:
            pass
        except Exception as e:
```

```
58             raise e
59
60
```

Let's Implement the Perceptron

In [ ]:
```
1
```

In [ ]:
```
1
```

In [3]:

```python
#the blue Print of perceptron

class Perceptron:
    def __init__(self,eta: float=None, epochs: int=None):
        #eta stand for learning rate
        #epochs =complete cycle of (forword_propogation+backword_propogati
        try:
            self.weights = np.random.randn(3)*1e-4 # giving small random w
            training = (eta is not None) and (epochs is not None)

            if training:
                print(f"Initial weights before training :\n {self.weights}
            self.eta = eta
            self.epochs = epochs

        except Exception as e:
            raise e

    def _z_outcome(self, inputs_with_bais, weights):
        try:
            return np.dot(inputs_with_bais,weights)
        except Exception as e:
            raise e

    def activation_funtion(self,z):
        try:
            return np.where(z > 0 ,1,0)
        except Exception as e:
            raise e

    def fit(self, x, y):
        try:
            self.x = x
            self.y = y

            x_with_bais = np.c_[self.x,-np.ones((len(self.x), 1))]
            print(f"X with bais : \n{x_with_bais}")

            for epoch in range(self.epochs):
                print("--"*10)
                print(f"for epoch >> {epoch}")
                print("--"*10)

                z = self._z_outcome(x_with_bais, self.weights)
                y_hat = self.activation_funtion(z)
                print(f"Predicted value after forward pass : \n{y_hat}")

                self.error = self.y-y_hat
                print(f"error : \n{self.error}")

                self.weights = self.weights+ self.eta*np.dot(x_with_bais.T
                print(f"Updated weights after epochs : {epoch+1}/{self.epo
                print("===="*10)

        except Exception as e:
            raise e

```

```
    def predict(self,x):
        try:
            x_with_bais = np.c_[x,-np.ones((len(x), 1))]
            z = self._z_outcome(x_with_bais,self.weights)
            return self.activation_funtion(z)

        except Exception as e:
            raise e

    def total_loss(self):
        try:
            total_loss = np.sum(self.error)
            print(f"\n total loss : {total_loss}\n")
            return total_loss

        except Exception as e:
            raise e

    def _create_dir_return_path(self, model_dir, filename):
        try:
            os.makedirs(model_dir,exist_ok= True)
            return os.path.join(model_dir,filename)

        except Exception as e:
            raise e

    def save(self,filename,model_dir=None):
        try:
            if model_dir is not None:
                model_file_path = self._create_dir_return_path(model_dir,
                joblib.dump(self,model_file_path)

            else:
                model_file_path = self._create_dir_return_path('model',fil
                joblib.dump(self,model_file_path)
        except Exception as e:
            raise e

    def load(self,filepath):
        try:
            return joblib.load(filepath)
        except Exception as e:
            raise e


```

Let's Utilize our Perceptron class

Before that Let's create a function for Ploting

In [4]:

```
def save_plot(df, model, plot_dir = 'plots' ,filename='plot.png'):
    try:
        def _create_base_plot(df):
            df.plot(kind = 'scatter',x='x1',y='x2',c= 'y',s = 100,cmap='co
            plt.axhline(y=0,color = 'black', linestyle = '--',linewidth=1)
            plt.axvline(x=0,color = 'black', linestyle = '--',linewidth=1)

            figure = plt.gcf()
            figure.set_size_inches(10,8)

        def _plot_decision_regions(X, y, classifier, resolution=0.02):
            colors = ('cyan', 'lightgreen')
            cmap = ListedColormap(colors)

            X = X.values
            x1 = X[:,0]
            x2 = X[:,1]

            x1_min,x1_max = x1.min() - 1, x1.max() + 1
            x2_min,x2_max = x2.min() - 1, x2.max() + 1

            xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                                   np.arange(x2_min, x2_max, resolution)
                                  )

            y_hat = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]
            y_hat = y_hat.reshape(xx1.shape)

            plt.contourf(xx1,xx2, y_hat, alpha = 0.3, cmap =cmap)
            plt.xlim(xx1.min(), xx1.max())
            plt.xlim(xx2.min(), xx2.max())

            plt.plot()
            plt.show()


        X,y = prepare_data(df)

        _create_base_plot(df)
        _plot_decision_regions(X,y,model)

        os.makedirs(plot_dir,exist_ok=True)
        plot_path = os.path.join(plot_dir,filename)

        plt.savefig(plot_path)

    except Exception as e:
        raise e

```

In [ ]:

```
1
```

Creating a function for Cepreating the data into X (Independent) and Y (Dependent)

In [5]:
```
def prepare_data(df,target_cols:str = 'y'):
    try:
        x = df.drop(target_cols,axis=1)
        y=df[target_cols]

        return x,y
    except Exception as e:
        raise e
```

AND

A	B	A **AND** B
0	0	0
0	1	0
1	0	0
1	1	1

In [6]:
```
AND = {
    'x1': [0,0,1,1],
    'x2': [0,1,0,1],
    'y' : [0,0,0,1]
}

# dict data converting into a data frame

df_AND = pd.DataFrame(AND)
df_AND
```

Out[6]:

	x1	x2	y
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

In [7]:
```
x,y = prepare_data(df_AND,'y')
print(x)
print('=='*10)
print(y)
```

```
   x1  x2
0   0   0
1   0   1
2   1   0
3   1   1
====================
0    0
1    0
2    0
3    1
Name: y, dtype: int64
```

Here i am calling Perceptron Function for creating model

In [8]:
```
x,y = prepare_data(df_AND)
ETA = 0.1 # eta lies between 0 to 1
EPOCHS = 10

model_and = Perceptron(ETA,EPOCHS)
model_and.fit(x,y)
_ = model_and.total_loss()
```

```
Initial weights before training :
 [-1.56638776e-05 -7.12828881e-05 -4.17482972e-05]
X with bais :
[[ 0.  0. -1.]
 [ 0.  1. -1.]
 [ 1.  0. -1.]
 [ 1.  1. -1.]]
--------------------
for epoch >> 0
--------------------
Predicted value after forward pass :
[1 0 1 0]
error :
0   -1
1    0
2   -1
3    1
Name: y, dtype: int64
Updated weights after epochs : 1/10 :
```

Saving the Model in the directory

In [9]:
```
model_and.save(filename='and.model',model_dir="Logical_model")
```

Loading the model

In [10]:
```
load_and = Perceptron().load(filepath='Logical_model/and.model')
load_and
```

Out[10]:
```
<__main__.Perceptron at 0x1a958c7ab20>
```

Doing the prediction

In [11]:
```
load_and.predict(x=[[0,0]])
```

Out[11]:
```
array([0])
```

In [12]:
```
## Let's Predict Hole data
x
```

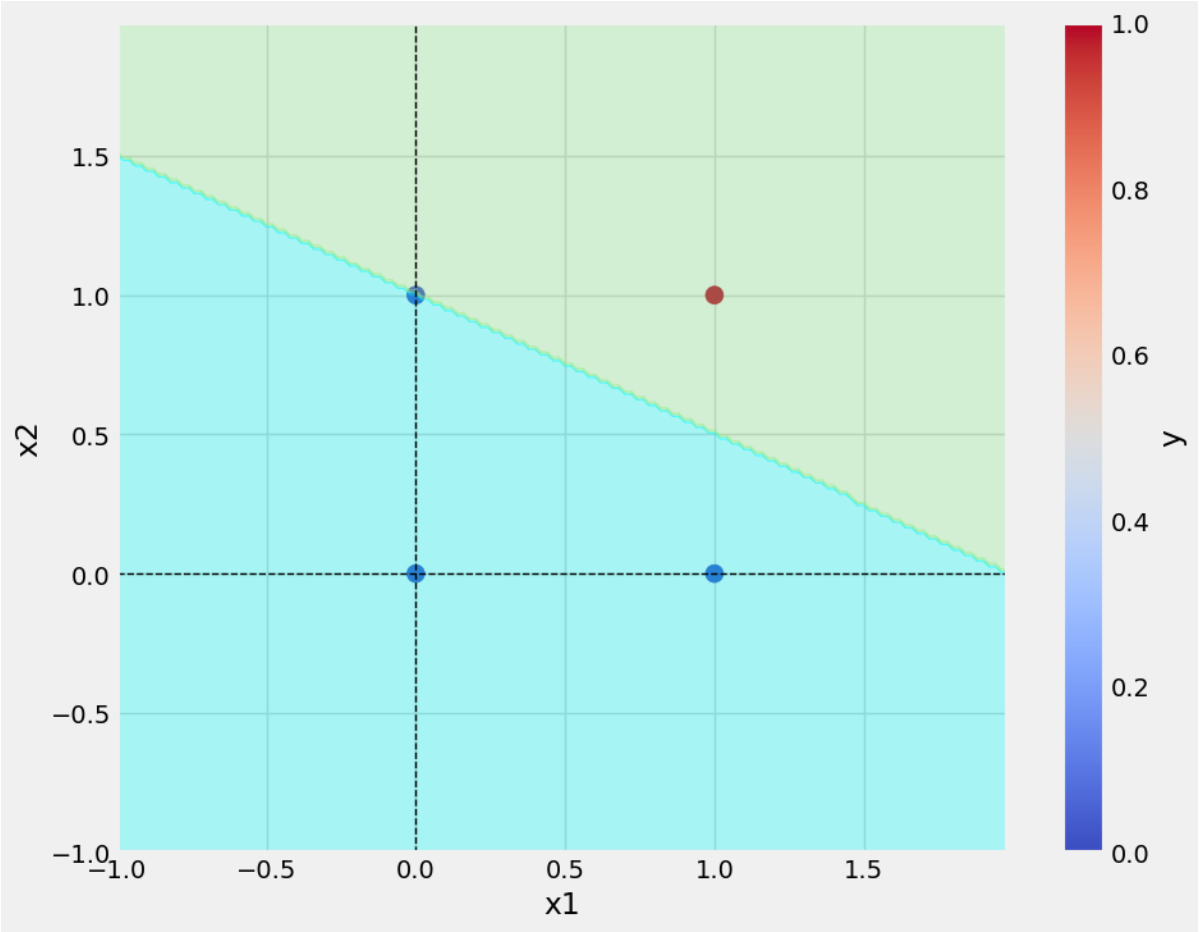
Out[12]:

	x1	x2
0	0	0
1	0	1
2	1	0
3	1	1

In [13]:
```
#
prediction = load_and.predict(x)
print(f"Predicted values : {prediction}")
print(f'Actual values :{list(y)}')
```

```
Predicted values : [0 0 0 1]
Actual values :[0, 0, 0, 1]
```

In [14]:
```
1 save_plot(df=df_AND,model=model_and,filename='AND.png')
```



<Figure size 640x480 with 0 Axes>

OR Data

Input A	Input B	OR Operator A | B
0	0	0
0	1	1
1	0	1
1	1	1

In [15]:
```
OR = {
    'x1': [0,0,1,1],
    'x2': [0,1,0,1],
    'y' : [0,1,1,1]
}

# dict data converting into a data frame

df_OR = pd.DataFrame(OR)
df_OR
```

Out[15]:

	x1	x2	y
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

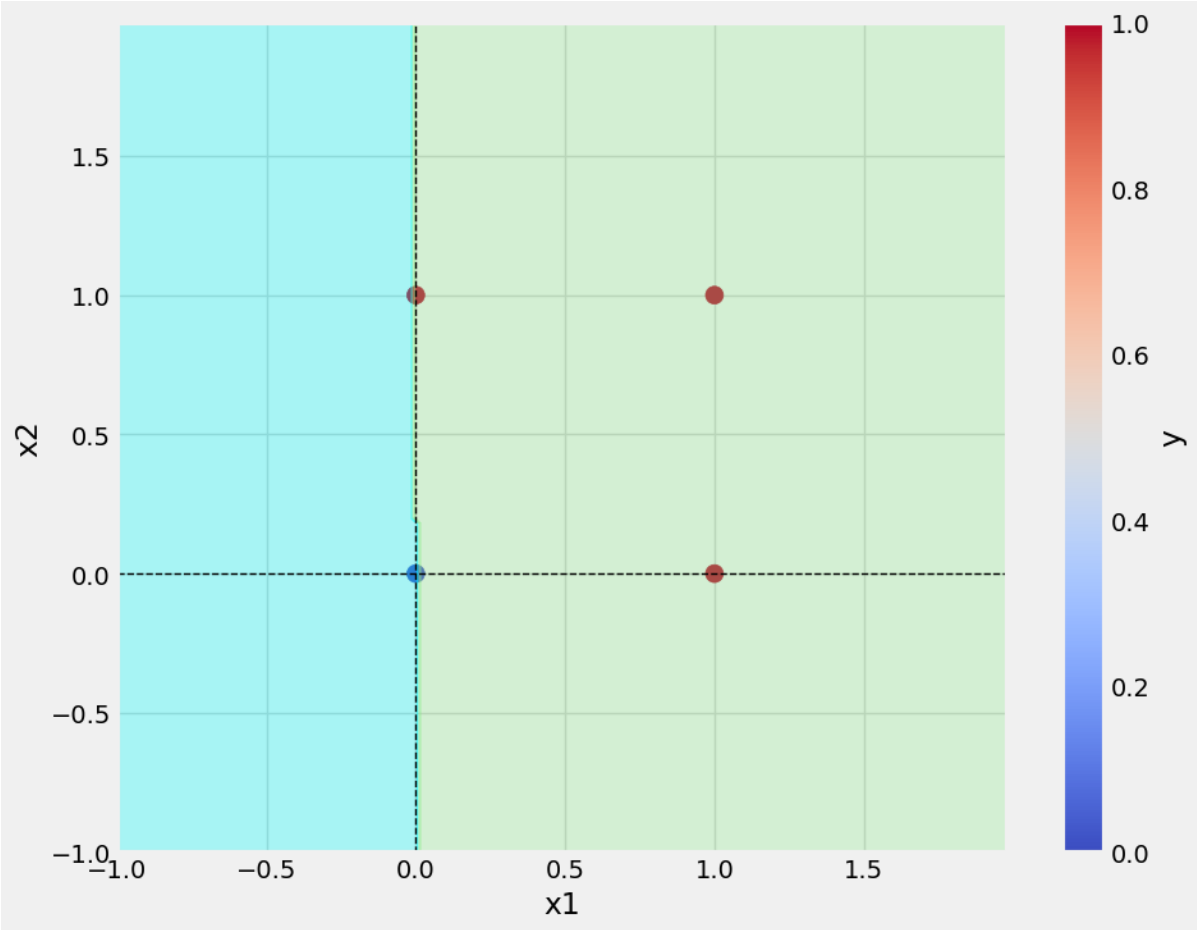
In [16]:
```
X,y = prepare_data(df_OR)

ETA = 0.1
EPOCHS = 10

model_or = Perceptron(eta=ETA,epochs=EPOCHS)
model_or.fit(X,y)

_ = model_or.total_loss()

```

```
Initial weights before training :
 [1.58477266e-05 1.01655688e-04 1.90209052e-05]
X with bais :
[[ 0.  0. -1.]
 [ 0.  1. -1.]
 [ 1.  0. -1.]
 [ 1.  1. -1.]]
--------------------
for epoch >> 0
--------------------
Predicted value after forward pass :
[0 1 0 1]
error :
0    0
1    0
2    1
3    0
Name: y, dtype: int64
Updated weights after epochs : 1/10 :
```

In [17]:
```
prediction_or = model_or.predict(df_OR.iloc[:,:-1])
print('=='*20)
print(f"Predicated values : {prediction_or}")
print('=='*20)
print(f"Actual values : {list(df_OR['y'])}")
```

```
========================================
Predicated values : [0 1 1 1]
========================================
Actual values : [0, 1, 1, 1]
```

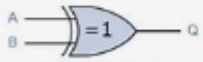
In [18]:
```
save_plot(df=df_OR,model=model_or,filename='OR.png')
```




```
<Figure size 640x480 with 0 Axes>
```

XOR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Symbol	Truth Table		
	A	B	Q
	0	0	0
	0	1	1
	1	0	1
	1	1	0
2-Input Ex-OR Gate			
Boolean Expression Q = A XOR B			

In [19]:
```
XOR = {
    'x1': [0,0,1,1],
    'x2': [0,1,0,1],
    'y' : [0,1,1,0]
}

# dict data converting into a data frame

df_XOR = pd.DataFrame(XOR)
df_XOR
```

Out[19]:

	x1	x2	y
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

In [20]:
```
X,y = prepare_data(df_XOR)

ETA = 0.1
EPOCHS = 20

model_xor = Perceptron(eta=ETA,epochs=EPOCHS)
model_xor.fit(X,y)

_ = model_xor.total_loss()


```

```
Initial weights before training :
 [-0.00019012 -0.00018722  0.00021099]
X with bais :
[[ 0.  0. -1.]
 [ 0.  1. -1.]
 [ 1.  0. -1.]
 [ 1.  1. -1.]]
--------------------
for epoch >> 0
--------------------
Predicted value after forward pass :
[0 0 0 0]
error :
0    0
1    1
2    1
3    0
Name: y, dtype: int64
Updated weights after epochs : 1/20 :
```

In [21]:
```
df_XOR
```

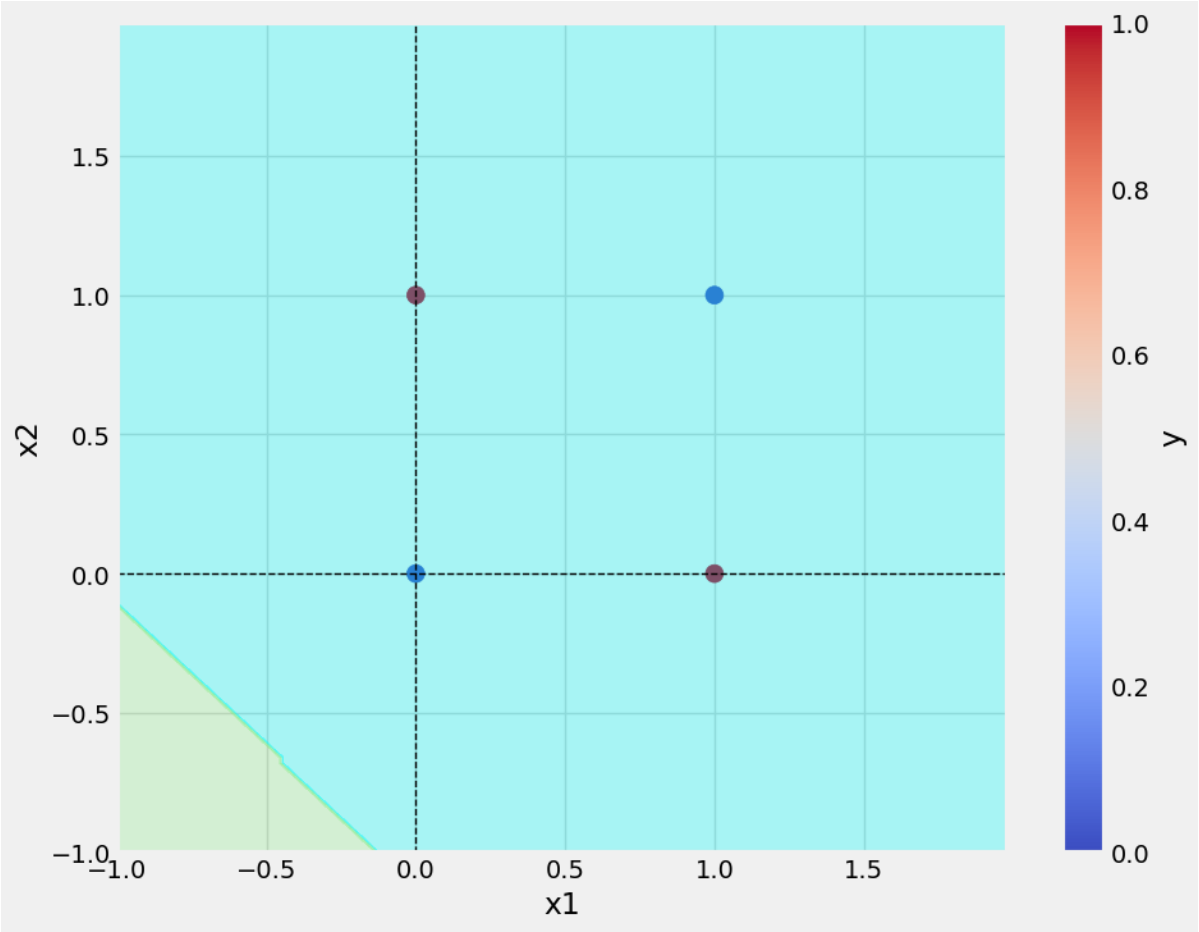
Out[21]:

	x1	x2	y
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

In [22]:
```
prediction_xor = model_xor.predict(df_XOR.iloc[:, :-1])
print('=='*20)
print(f"Predicted value :{prediction_xor}")
print('=='*20)
print(f"Actual values : {list(df_XOR['y'])}")
```

```
========================================
Predicted value :[0 0 0 0]
========================================
Actual values : [0, 1, 1, 0]
```

In [23]:
```
1 save_plot(df=df_XOR,model=model_xor,filename='xor.png')
```




```
<Figure size 640x480 with 0 Axes>
```

Perceptron DrawBack

Here in x-or we are not able to sepreate the data ,we are not able to Draw a line So for that we need multilayer Perceptron

In [ ]:
```
1
```