ASSIGNMENT - 6

ANIKEIT SETHI (190001003)

Ques 1: -

In [16]:

```
import numpy as np
class Model:
  def __init__(self, num_inputs):
    self.weights = np.zeros(num_inputs)
    # self.weights[0]=0
    # self.weights[1]=2
    self.bias = 0

  def set_weight(self,num_inputs):
    for i in range(len(num_inputs)):
      self.weights[i]=num_inputs[i]

  def predict(self, inputs):
    sum = np.dot(inputs, self.weights[:]) + self.bias
    if sum >= 0:
      prediction = 1
    else:
      prediction = 0
    return prediction


  def fit(self, inputs, label, epochs=3, learning_rate=0.5):
    ep = 0
    while True:
      flag = True
      for val, res in zip(inputs, label):
        predictedOutput=self.predict(val)
        if(predictedOutput != res):
          flag = False
          break
      if flag == True:
        break
      print(str(ep)+"th Epoc")
      ep = ep + 1
      for val, res in zip(inputs, label):
        prediction= self.predict(val)
        #print("Prediction->"+str(prediction))
        self.weights[:] += learning_rate * (res-prediction) * val
        self.bias += learning_rate *(res-prediction)
      print("Updated Weights: ")
      for i in self.weights:
        print(i,end=" ")
      print("")
      print(f"Updated Bias: {self.bias}")
      print("")
```

In [17]:

```
import numpy as np

train_inputs= np.array([
[1,1],
[1,2],
[2,-1],
[2,0],
[-1,2],
[-2,1],
[-1,-1],
[-2,-2]
])

labels1= np.array([0,0,0,0,1,1,1,1])
labels2= np.array([0,0,1,1,0,0,1,1])

mcp1 = Model(2)
mcp1.fit(train_inputs,labels1)
```

```
mcp2 = Model(2)
mcp2.fit(train_inputs,labels2)

for inputs in train_inputs:
    print(inputs, end="")
    print(mcp1.predict(inputs), mcp2.predict(inputs))
```

```
0th Epoc
Updated Weights:
-1.0 0.5
Updated Bias: 0.0

1th Epoc
Updated Weights:
-1.5 -0.5
Updated Bias: -0.5

0th Epoc
Updated Weights:
0.5 -1.0
Updated Bias: 0.0

[1 1]0 0
[1 2]0 0
[ 2 -1]0 1
[2 0]0 1
[-1  2]1 0
[-2  1]1 0
[-1 -1]1 1
[-2 -2]1 1
```

Ques 2: -

In [34]:

```
import numpy as np
class Model:
 def __init__(self, num_inputs):
   self.weights = np.zeros(num_inputs)
   self.bias = 0

 def set_weight(self,num_inputs):
   for i in range(len(num_inputs)):
     self.weights[i]=num_inputs[i]

 def predict(self, inputs):
   sum = np.dot(inputs, self.weights[:]) + self.bias
   if sum >= 0:
     prediction = 1
   else:
     prediction = 0
   return prediction

 def fit(self, inputs, label, learning_rate=0.5):
   ep = 0
   while True:
     flag = True
     for val, res in zip(inputs, label):
       predictedOutput=self.predict(val)
       if(predictedOutput != res):
         flag = False
         break
     if flag == True:
       break
     print(f"{ep}th epoch")
     ep = ep + 1
     for val, res in zip(inputs, label):
       prediction= self.predict(val)
       self.weights[:] += learning_rate * (res-prediction) * val
       self.bias += learning_rate *(res-prediction)
     print(f"Updated Weights: {self.weights}")
     print(f"Updated Bias: {self.bias}\n")
```

In [35]:

```
#2inputs
import numpy as np

train_inputs= []
train_inputs.append(np.array([0, 0]))
train_inputs.append(np.array([0, 1]))
train_inputs.append(np.array([1, 1]))
train_inputs.append(np.array([1, 0]))

labels1= np.array([0, 0, 1, 1])
labels2= np.array([0, 1, 1, 0])

mcp1 = Model(2)
mcp1.fit(train_inputs,labels1)

mcp2 = Model(2)
mcp2.fit(train_inputs,labels2)

for inputs in train_inputs:
    print(inputs, end="")
    print(mcp1.predict(inputs), mcp2.predict(inputs))
```

```
0th epoch
Updated Weights: [0.5 0.5]
Updated Bias: 0.0

1th epoch
```

```
Updated Weights: [1.  0.5]
Updated Bias: -0.5

2th epoch
Updated Weights: [1. 0.]
Updated Bias: -1.0

0th epoch
Updated Weights: [-0.5  0.5]
Updated Bias: -0.5

1th epoch
Updated Weights: [-0.5  1. ]
Updated Bias: -0.5

[0 0]0 0
[0 1]0 1
[1 1]1 1
[1 0]1 0
```

In [36]:

```
# 3 input
import numpy as np

train_inputs= []
train_inputs.append(np.array([0, 0, 0]))
train_inputs.append(np.array([0, 0, 1]))
train_inputs.append(np.array([0, 1, 0]))
train_inputs.append(np.array([0, 1, 1]))
train_inputs.append(np.array([1, 0, 0]))
train_inputs.append(np.array([1, 0, 1]))
train_inputs.append(np.array([1, 1, 0]))
train_inputs.append(np.array([1, 1, 1]))

labels1= np.array([0, 0, 0, 0, 1, 1, 1, 1])
labels2= np.array([0, 0, 1, 1, 0, 0, 1, 1])
labels3= np.array([0, 1, 0, 1, 0, 1, 0, 1])


mcp1 = Model(3)
mcp1.fit(train_inputs,labels1)

mcp2 = Model(3)
mcp2.fit(train_inputs,labels2)

mcp3 = Model(3)
mcp3.fit(train_inputs,labels3)

for inputs in train_inputs:
    print(inputs, end="")
    print(mcp1.predict(inputs), mcp2.predict(inputs), mcp3.predict(inputs))
```

```
0th epoch
Updated Weights: [0.5 0.  0. ]
Updated Bias: 0.0

1th epoch
Updated Weights: [0.5 0.  0. ]
Updated Bias: -0.5

0th epoch
Updated Weights: [0. 1. 0.]
Updated Bias: 0.0

1th epoch
Updated Weights: [0. 1. 0.]
```

```
[0  0  0]0  0  0
[0  0  1]0  0  1
[0  1  0]0  1  0
[0  1  1]0  1  1
[1  0  0]1  0  0
[1  0  1]1  0  1
[1  1  0]1  1  0
[1  1  1]1  1  1
```

```
# It cannot be done as it is going in infinite loop with 2 or 3 neurons
# Because to classify into 2 classes we have to use another layer which we
# Cannot take
```

Ques 3: -

In [29]:

```
import numpy as np
class Model:
  def __init__(self, num_inputs):
    self.weights = np.zeros(num_inputs)
    self.bias = 0

  def set_weight(self,num_inputs):
    for i in range(len(num_inputs)):
      self.weights[i]=num_inputs[i]

  def predict(self, inputs):
    sum = np.dot(inputs, self.weights[:]) + self.bias
    if sum >= 0:
      prediction = 1
    else:
      prediction = 0
    return prediction
  def fit(self, inputs, label, epochs=3, learning_rate=0.5):
    ep = 0
    while True:
      flag = True
      for val, res in zip(inputs, label):
        predictedOutput=self.predict(val)
        if(predictedOutput != res):
          flag = False
          break
      if flag == True:
        break
      ep = ep + 1
      for val, res in zip(inputs, label):
        prediction= self.predict(val)
        self.weights[:] += learning_rate * (res-prediction) * val
        self.bias += learning_rate *(res-prediction)
    print("Updated Weights: ")
    for i in self.weights:
      print(i,end=" ")
    print("")
    print(f"Updated Bias: {self.bias}")
    print("")
```

In [31]:

```
train =np.array([
[1,4],[1,5],[2,4],[2,5],[3,1],[3,2],[4,1],[4,2]
])

label = [0, 0, 0, 0, 1, 1, 1, 1]

m = Model(2)
m.fit(train,label)

for inputs in train:
  print(inputs, m.predict(inputs))
```

```
Updated Weights:
1.0 -1.5
Updated Bias: 0.0

[1 4] 0
[1 5] 0
[2 4] 0
[2 5] 0
[3 1] 1
[3 2] 1
[4 1] 1
[4 2] 1
```