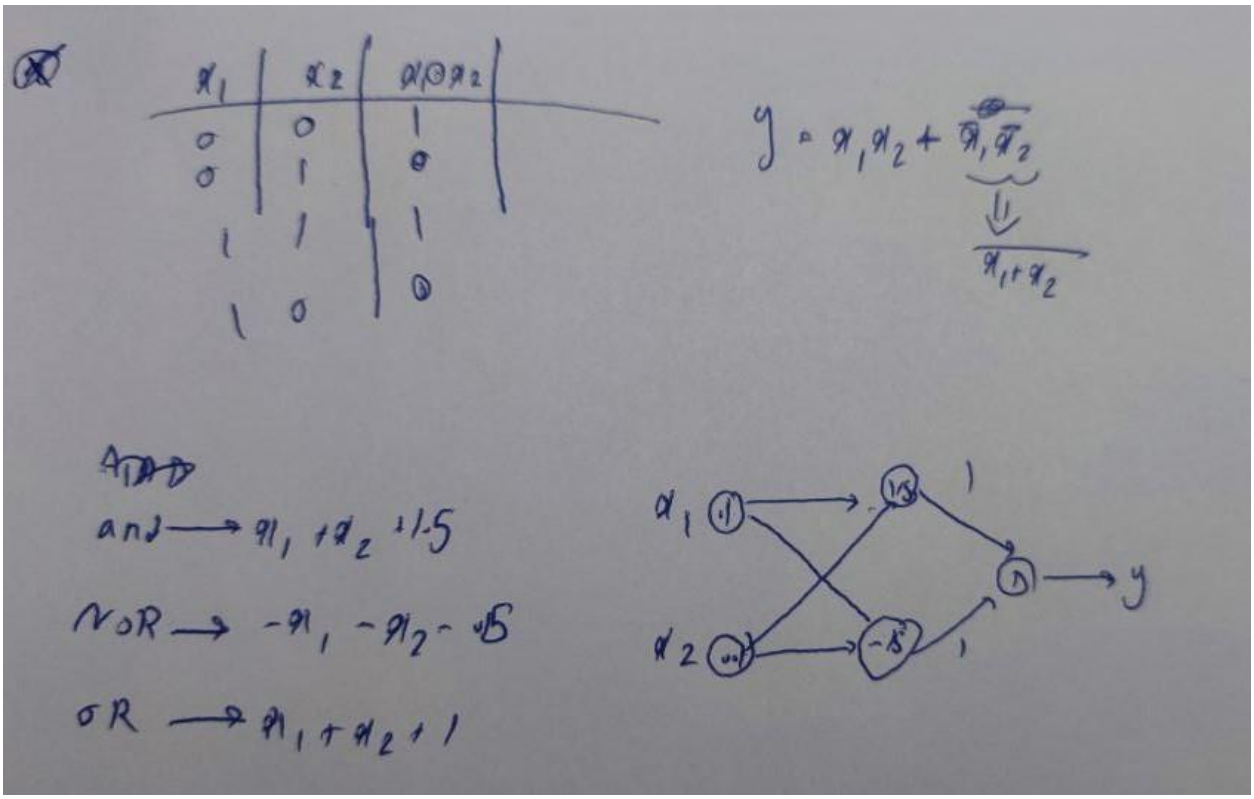
سوال2)

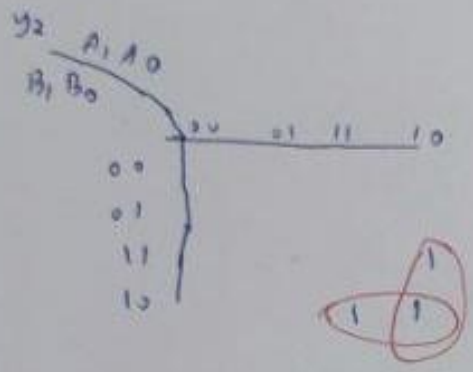
1)ابتدا ورودی و خروجی تابع را به دست میاوریم
2)تابع خروجی را به دست میاوریم تا بدانیم قرار است از چه گیت هایی استفاده کنیم
3)وزن هارا انتخاب میکنیم



حال یک ضرب کننده را بررسی میکنیم که۲تا ورودی۲بیتی و4تا خروجی دارد:

تابع تمام خروجی هارا به ازای ورودی های مختلف برسی میکنیم:

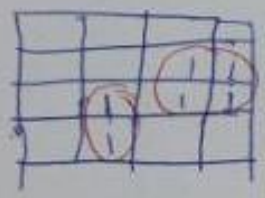
$y_3 = A_1 A_0 B_1 B_0$



$y = A_1 \bar{A}_0 B_1 + A_1 B_1 \bar{B}_0$

$= A_1 B_1 (\bar{A}_0 + \bar{B}_0)$

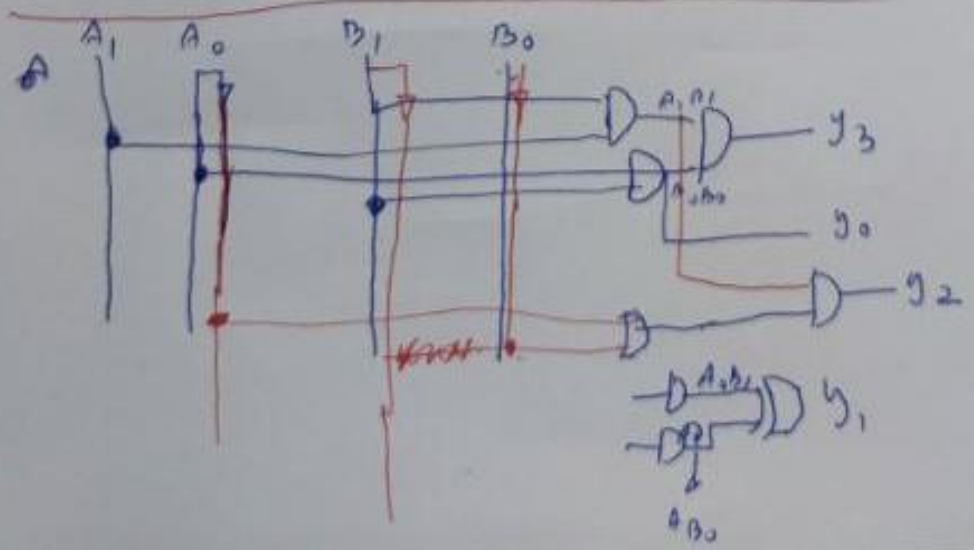
y_1



$\Rightarrow A_1 B_0 + \bar{A}_1 A_0 B_1 = y_1$

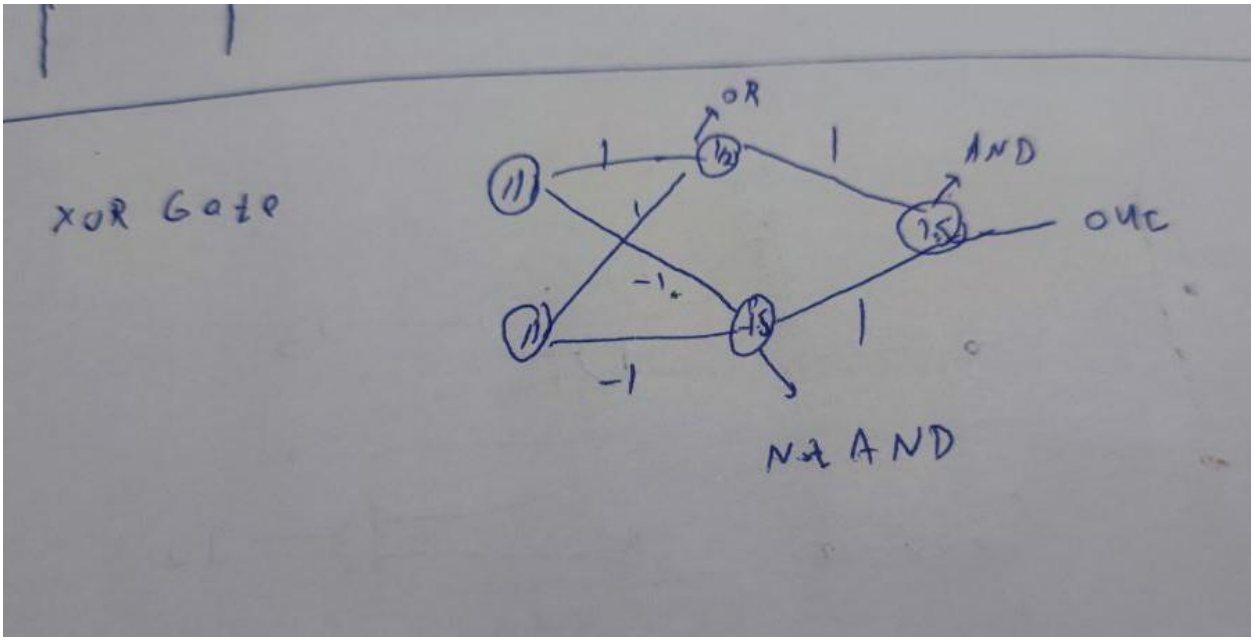
$\hookrightarrow A_0 B_1 \oplus A_1 B_0$

$y_0 = A_0 B_0$



حال میتوان مثل حالت قبل وزنها را انتخاب کرد.

ابتدا گیت notو گیت xorرا میسازیم .



حال با داشتن وزنها میتوان به کدنویسی پرداخت:

این کد یک مدل ساده از یک نورون مصنوعی را پیاده سازی میکند

این کد یک کلاس به نام McCulloch_Pitts_neuron ایجاد میکند که این نورون را پیادەسازی میکند. این

نورون دو ویژگی اصلی دارد: وزنها (weights) وآستانه(threshold).

که در واقع آستانه همان کاراکرد بایاس را دارد

در متد init، وزنها و آستانه به عنوان ورودیهای کلاس گرفته میشوند و در متغیرهای weights و Threshold ذخیره می شوند.

در متد model، ورودی X به عنوان ورودی نورون گرفته میشود و با استفاده از محاسبه ی ضرب داخلی بین وزنهاو ورودی

(self.weights @x)، مقدار خروجی نورون محاسبه میشود. اگر این مقدار بیشتر یا مساوی آستانه باشد، خروجی ۱ و در غیر این صورت، خروجی . خواهد بود.

```
class McCulloch_Pitts_neuron():

  def __init__(self , weights , threshold):
    self.weights = weights
    self.threshold = threshold

  def model(self , x):
    if self.weights @ x >= self.threshold:
        return 1
    else:

return 0


def XNOR(input):
  neur1 = McCulloch_Pitts_neuron([1, 1], 1.5)
  neur2 = McCulloch_Pitts_neuron([-1, -1], -0.5)
  neur3 = McCulloch_Pitts_neuron([1, 1], 1)

  z1 = neur1.model(np.array([input[0], input[1]]))
  z2 = neur2.model(np.array([input[0], input[1]]))
  z3 = neur3.model(np.array([z1, z2]))


  return list([z1, z2, z3])
```

این کد یک تابع به نام XNOR با استفاده از ۳ نورون McCulloch_Pitts پیاده سازی می کند.

در این تابع، سه نورون McCulloch_Pitts با وزنهای مشخص و آستانه های مشخص که در قسمت ۱ تعریف کردیم ایجاد میشوند. سپس با استفاده از ورودی های تابع(input)، خروجی های هر یک ازنورون ها محاسبه میشوند و در متغیرهای z1،z2 و Z3 ذخیره می شوند.

حال میخواهیم خروجی هارا برای تابعی که تعرییف کردیم ببینیم

ابتدا چنین چیزی ایجاد میکنیم:

[(1, 1), (1, 0), (0, 1), (0, 0)]

```
input = [1, 0]
X = list(itertools.product(input, input))
```

که خروجی کد بالا همان چیزی هست که ما میخواهیم

سپس برای هریک از حالات بالا خروجی را مشاهده میکنیم

```
for i in X:
    res = XNOR(i)
    print("XNOR with input as", str(i[0]) + str(" ")+str(i[1]), "goes to
output ", str(res[2]))
```

```
XNOR with input as 1 1 goes to output  1
XNOR with input as 1 0 goes to output  0
XNOR with input as 0 1 goes to output  0
XNOR with input as 0 0 goes to output  1
```

در ادامه ضرب کننده خواسته شده در صورت سوال را به دست میاوریم :y0

```
def Y0(input):
neur2 = McCulloch_Pitts_neuron([0, 1 , 0 , 1] , 1.5) #A0B0

z2 = neur2.model(np.array(input))
```

```
return list([z2])
```

y1:

```
def Y1(input):
  neur7 = McCulloch_Pitts_neuron([1, 0 , 0 , 1] , 1.5) #A1B0
  neur8 = McCulloch_Pitts_neuron([0, 1 , 1 , 0] , 1.5) #A0B1
  neur6 = McCulloch_Pitts_neuron([1, 1], 1) # or gate
  neur9 = McCulloch_Pitts_neuron([-1, -1], -1.5) # not and gate
  neur3 = McCulloch_Pitts_neuron([1, 1], 1.5) # and gate -> output is:
A1B0 XOR A0B1

  z7 = neur7.model(np.array(input))
  z8 = neur8.model(np.array(input))
  z6 = neur6.model(np.array([z7, z8]))
  z9 = neur9.model(np.array([z7, z8]))
  z3 = neur3.model(np.array([z6, z9]))

  return list([z7,z8,z6,z9,z3])
```

y2:

```
def Y2(input):
  neur1 = McCulloch_Pitts_neuron([1, 0 , 1 , 0] , 1.5) #A1B1
  neur4 = McCulloch_Pitts_neuron([0, -1 , 0 , 0] , 0) #NOT A0 = A0'
  neur5 = McCulloch_Pitts_neuron([0, 0 , 0 , -1] , 0) #NOT B0 = B0'
  neur6 = McCulloch_Pitts_neuron([1, 1], 1) #A0'+B0'
  neur3 = McCulloch_Pitts_neuron([1, 1], 1.5) #A1B1(A0'+B0')

  z1 = neur1.model(np.array(input))
  z4 = neur4.model(np.array(input))
  z5 = neur5.model(np.array(input))
  z6 = neur6.model(np.array([z4, z5]))
  z3 = neur3.model(np.array([z1, z6]))

  return list([z1,z4,z5,z6,z3])
```

y3:

```
def Y3(input):
  neur1 = McCulloch_Pitts_neuron([1, 0 , 1 , 0] , 1.5) #A1B1
  neur2 = McCulloch_Pitts_neuron([0, 1 , 0 , 1] , 1.5) #A0B0
  neur3 = McCulloch_Pitts_neuron([1, 1], 1.5) #A1B1A0B0

  z1 = neur1.model(np.array(input))
  z2 = neur2.model(np.array(input))
  z3 = neur3.model(np.array([z1, z2]))

  return list([z1,z2,z3])
```

سپس با استفاده از کد هایی که نوشتیم نتیجه نهایی را به دست میاوریم تا ببینیم نتیجه ی آن با نتیجه ی انتظارمون یکی است یا خیر

```
X = list(itertools.product([0, 1], repeat=4))

for i in X:
  res3 = Y3(i)
  res2 = Y2(i)
  res1 = Y1(i)
  res0 = Y0(i)

  print("Y with input as", str(i[0]) + " " + str(i[1]) + " " + str(i[2]) +
" " + str(i[3]), "goes to output ", str(res3[2]), str(res2[4]),
str(res1[4]), str(res0[0]))
```

نتیجه:

```
Y with input as 0 0 0 0 goes to output  0 0 0 0
Y with input as 0 0 0 1 goes to output  0 0 0 0
Y with input as 0 0 1 0 goes to output  0 0 0 0
Y with input as 0 0 1 1 goes to output  0 0 0 0
Y with input as 0 1 0 0 goes to output  0 0 0 0
Y with input as 0 1 0 1 goes to output  0 0 0 1
Y with input as 0 1 1 0 goes to output  0 0 1 0
Y with input as 0 1 1 1 goes to output  0 0 1 1
Y with input as 1 0 0 0 goes to output  0 0 0 0
Y with input as 1 0 0 1 goes to output  0 0 1 0
Y with input as 1 0 1 0 goes to output  0 1 0 0
Y with input as 1 0 1 1 goes to output  0 1 1 0
Y with input as 1 1 0 0 goes to output  0 0 0 0
Y with input as 1 1 0 1 goes to output  0 0 1 1
Y with input as 1 1 1 0 goes to output  0 1 1 0
Y with input as 1 1 1 1 goes to output  1 0 0
```

که دقیقا همان چیزی است که انتظار داشتیم