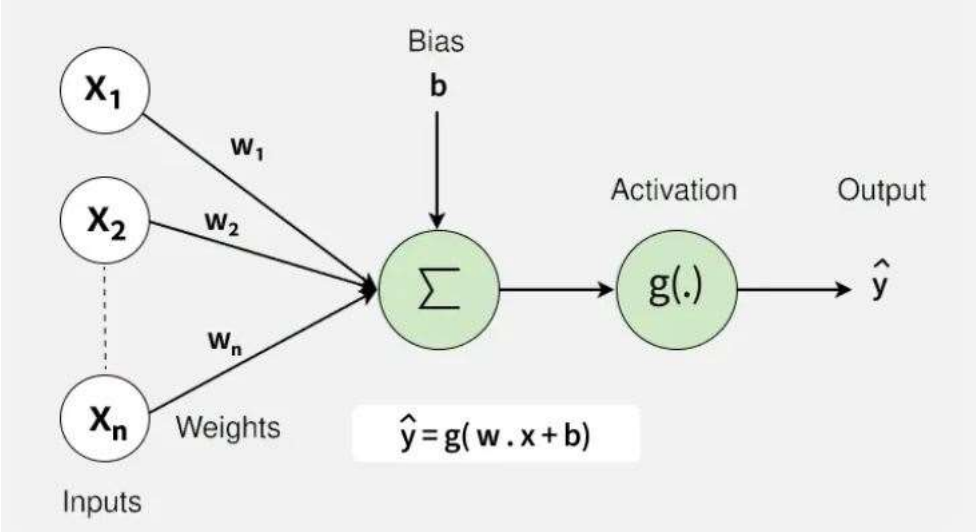
Single Layer Perceptron

CT-466 | Week 2 - Lecture 3

Instructor: Mehar Fatima Shaikh

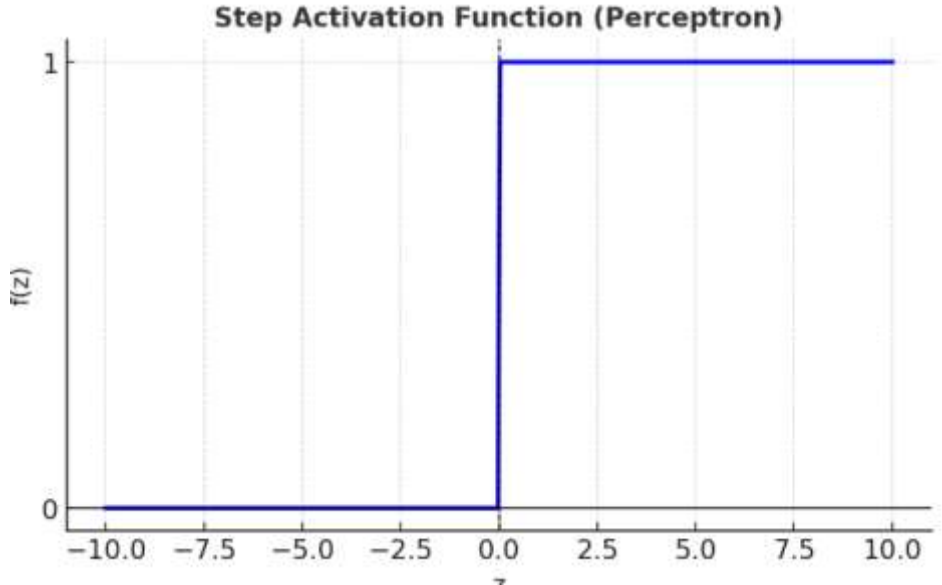
Perceptron

- A perceptron is one of the simplest models of an artificial neural network and is often considered the foundation of modern deep learning.
- A perceptron is a computational model of a single artificial neuron. It takes multiple inputs, applies weights, adds a bias, and passes the result through an activation function to produce a single binary output.



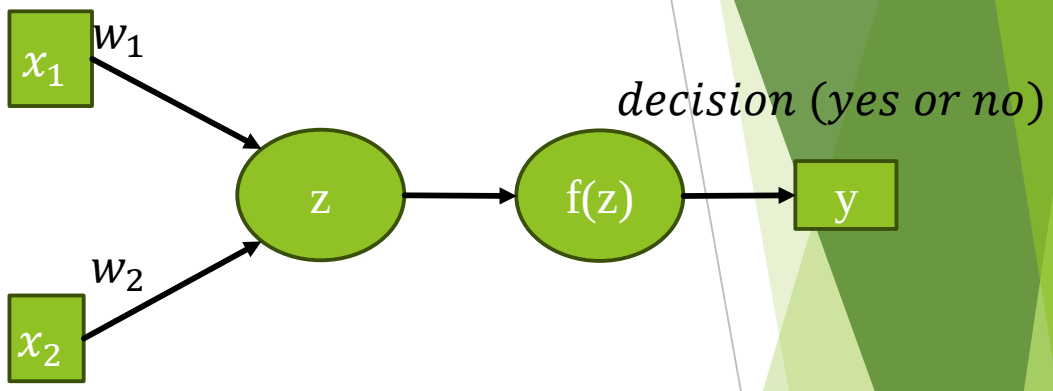
Perceptron

- For inputs $x_1, x_2, x_3, \ldots, x_n$ with weights $w_1, w_2, w_3 \ldots, w_n$ and bias b:
- $z = w_1x_1 + w_2x_2 + w_3x_3 + \ \ldots + w_nx_n + b = \sum_{i=1}^{n} x_i w_i + b$
- The output is: $y = f(z) = \begin{cases} 1, & if \ \sum_{i=1}^{n} x_i w_i + b \ > 0 \\ 0, & if \ \sum_{i=1}^{n} x_i w_i + b \ \leq 0 \end{cases} = \begin{cases} 1, & if \ \sum_{i=1}^{n} x_i w_i \ > -b \\ 0, & if \ \sum_{i=1}^{n} x_i w_i \ \leq -b \end{cases}$
- Where $x_i \ is \ the \ inputs, w_i$ is the weights, b is the bias and f is the **activation function** (originally a step function).
- This sharp jump at z=0 is what makes the perceptron act as a **binary classifier** (yes/no decision).



Example

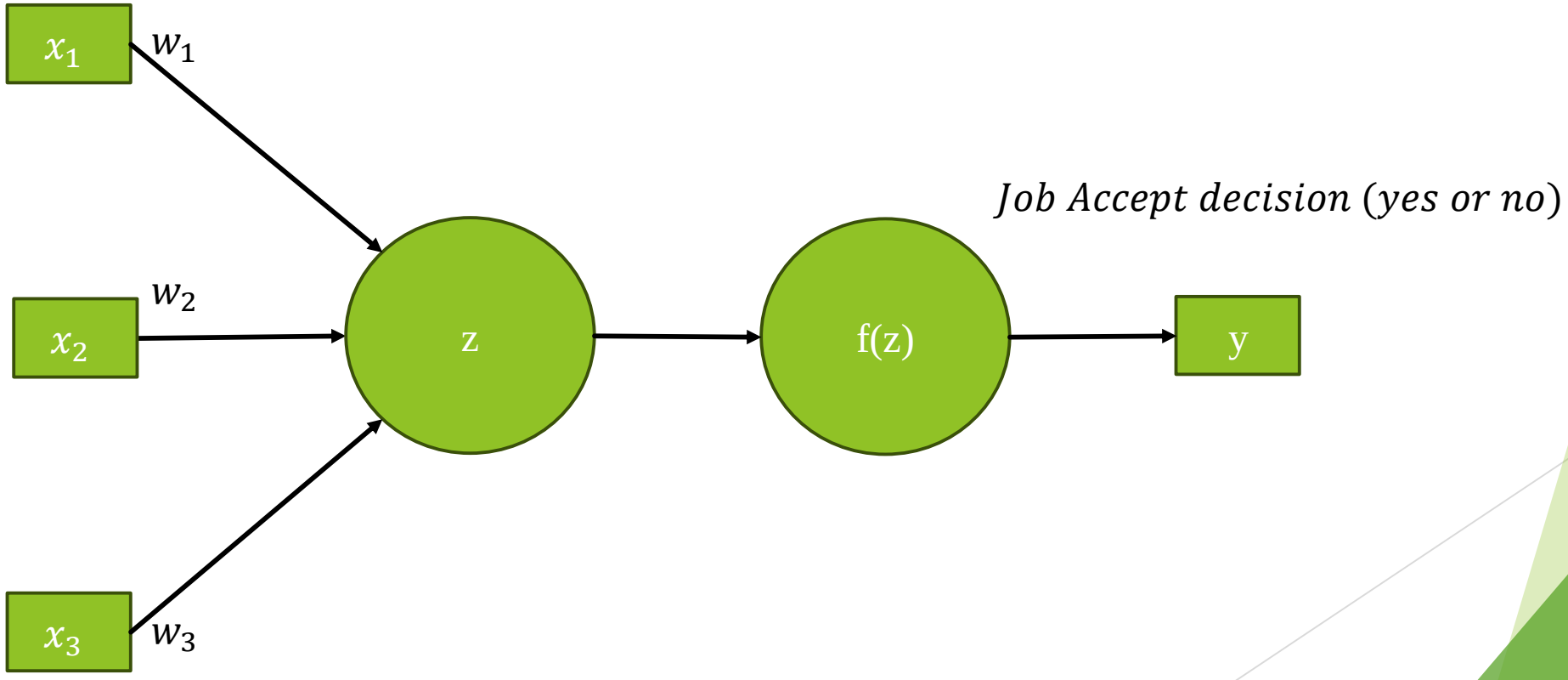
- Imagine a perceptron that predicts if you should play football depending on:
  - x1 = Weather (sunny=1, rainy=0)
  - x2 = Temperature (hot=1, cold=0)
  - w1=0.6 (Weather more important)
  - w2=0.4(Temperature less important)
  - Threshold (b) = 0.5
- If the weighted sum is high enough → output = 1 ("Play").
- Otherwise → output = 0 ("Don't play").
- $z = w_1 x_1 + w_2 x_2$
- $f(z) = \begin{cases} 1, & if\ z > b\ (\ Play) \\ 0, & if\ z \leq b\ (\ Don't\ Play) \end{cases}$



Weather (x1)	Temperature (x2)	Weighted Sum	y	Decision
0 (Rainy)	0 (Cold)	0.0	0	Don't Play
0 (Rainy)	1 (Hot)	0.4	0	Don't Play
1 (Sunny)	0 (Cold)	0.6	1	Play
1 (Sunny)	1 (Hot)	1.0	1	Play

Example 2

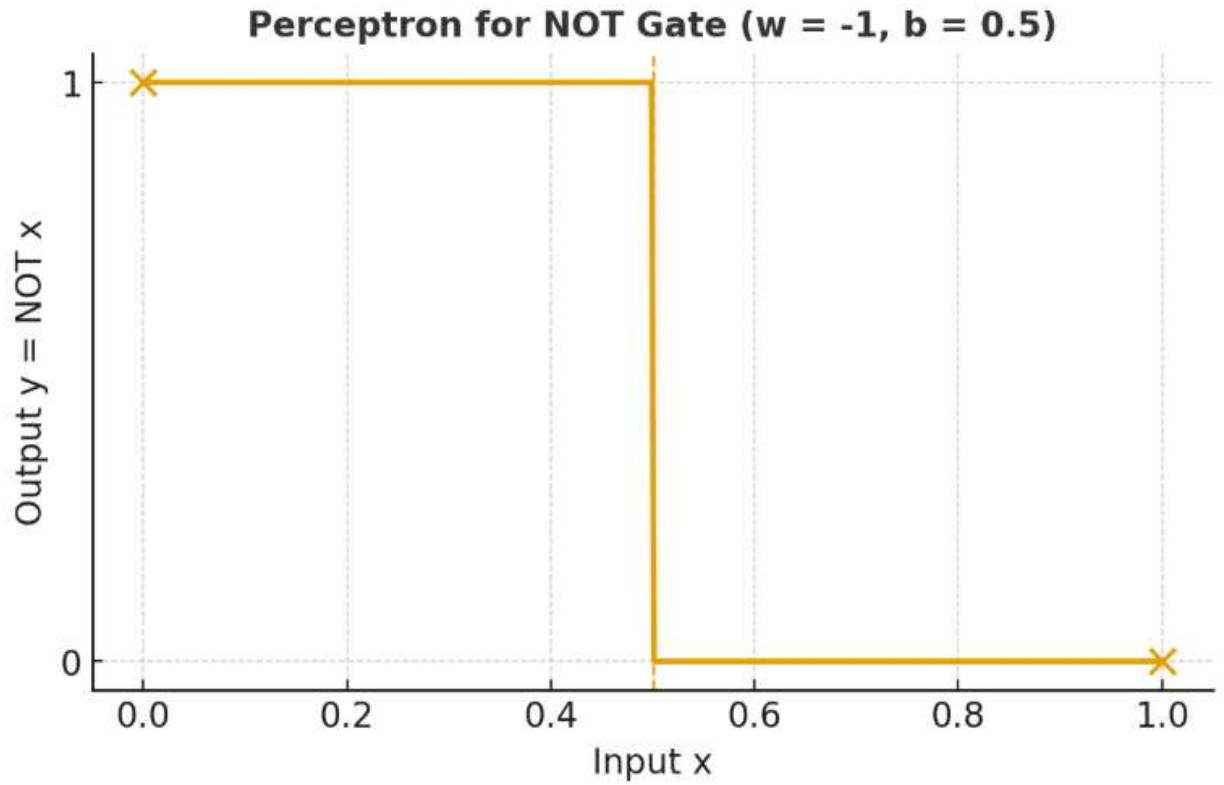
Inputs (x): { Salary, Transport, Company }
Weights (w): {0.4 , 0.2 , 0.3}
Threshold= 0.5
Bias (b): assume b=0 for simplicity (we can add later).



Example 2

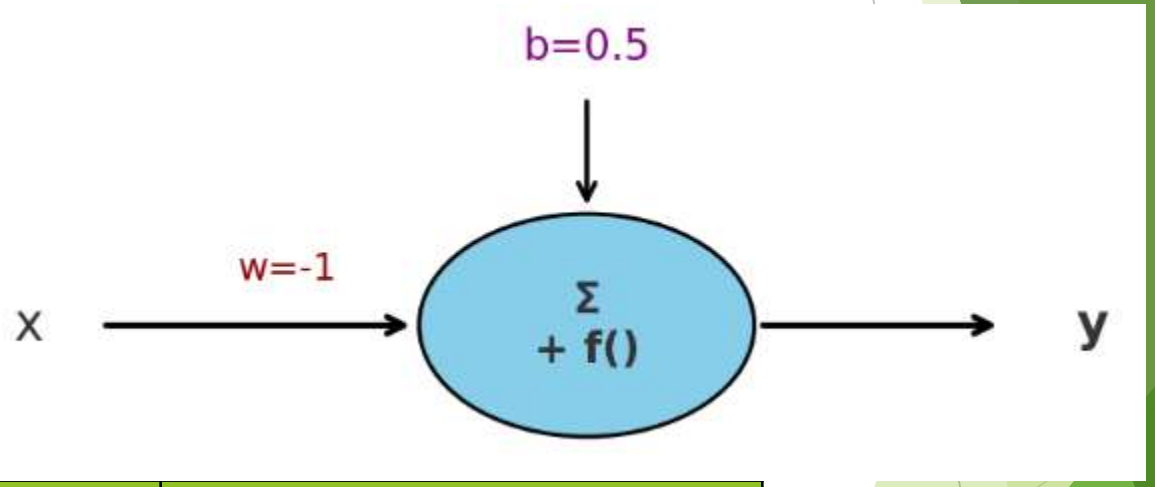
S	T	C	Weighted Sum (z)	y
0	0	0	0	0
0	0	1	0.3	0
0	1	0	0.2	0
0	1	1	0.2+0.3=0.5	0
1	0	0	0.4	0
1	0	1	0.4+0.3=0.7	1
1	1	0	0.4+0.2=0.6	1
1	1	1	0.4+0.2+0.3=0.9	1

Not Gate


Perceptron for NOT Gate (w = -1, b = 0.5)
Output y = NOT x
1
0
0.0
0.2
0.4
0.6
0.8
1.0
Input x

Not Gate

- Input $x \in \{0,1\}$
- $z = f(wx + b)$, $f(z) = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}$
- Suppose w = -1, b=0.5
- If x=0: z= (-1*0) +0.5 = 0.5 > 0 => y = 1
- If x=1: z= (-1*1) +0.5 = -0.5 < 0 => y = 0



x	z=wx+b	y=f(z) (NOT x)
0	0.5	1
1	−0.5	0

Python Implementation of NOT Gate:

```
255  # importing Python library
256  import numpy as np
257
258  # define Unit Step Function
259  def unitStep(v):
260      if v >= 0:
261          return 1
262      else:
263          return 0
264
265  # design Perceptron Model
266  def perceptronModel(x, w, b):
267      v = np.dot(w, x) + b
268      y = unitStep(v)
269      return y
270
271  # NOT Logic Function
272  # w = -1, b = 0.5
273  def NOT_logicFunction(x):
274      w = -1
275      b = 0.5
276      return perceptronModel(x, w, b)
277
278  # testing the Perceptron Model
279  test1 = np.array(1)
280  test2 = np.array(0)
281
282  print("NOT({}) = {}".format(1, NOT_logicFunction(test1)))
283  print("NOT({}) = {}".format(0, NOT_logicFunction(test2)))
```

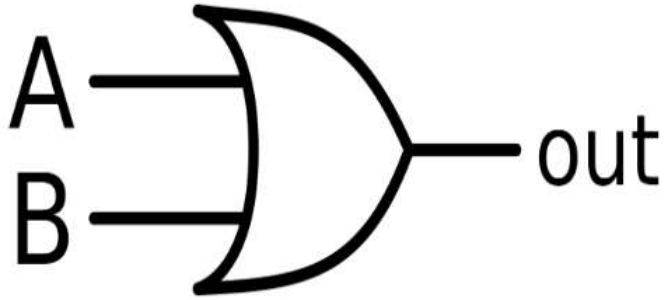
Python Implementation of NOT Gate:

```
In [2]: import numpy as np
   ...:
   ...: def NOT(x):
   ...:     w, b = -1, 0.5
   ...:     return 1 if (w * x + b) >= 0 else 0
   ...:
   ...: # Testing
   ...: for i in [0, 1]:
   ...:     print(f"NOT({i}) = {NOT(i)}")
   ...:
NOT(0) = 1
NOT(1) = 0

In [3]:
```

OR gate using a single perceptron

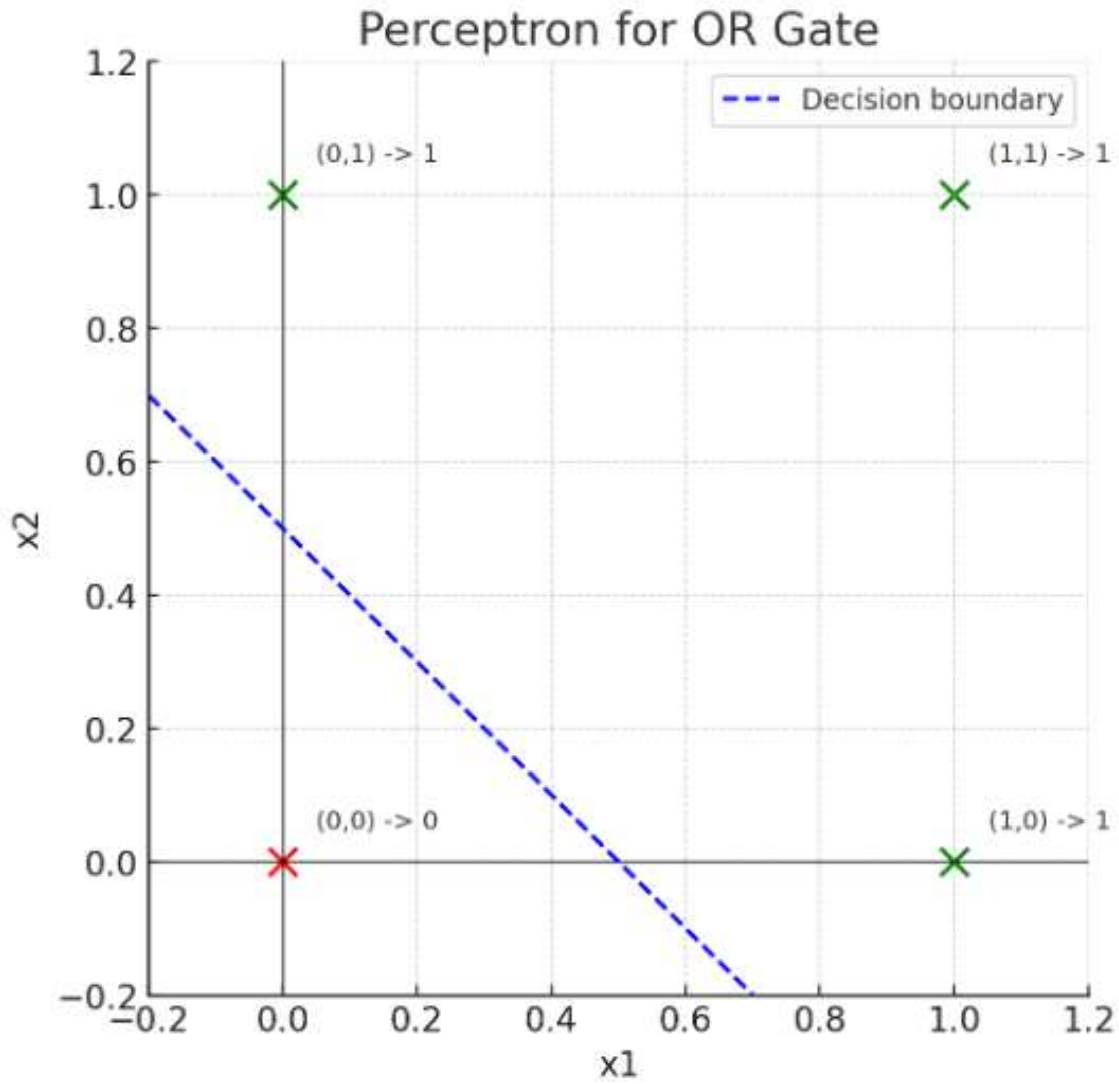
- Input $x_1, x_2 \in \{0,1\}$
- Weights $w_1 = 1\ , w_2 = 2, bias: b = -0.5$
- $z = w_1x_1 + w_2x_2 + b,\ y = f(z) = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}$



- Suppose
- If (x_1, x_2) = (0 , 0) : z = (1*0) + (1*0) - 0.5 = -0.5 < 0 => y = 0
- If (x_1, x_2) = (0 , 1) : z = (1*0) + (1*1) - 0.5 = 0.5 > 0 => y = 1
- If (x_1, x_2) = (1 , 0) : z = (1*1) + (1*0) - 0.5 = 0.5 > 0 => y = 1
- If (x_1, x_2) = (1 , 1) : z = (1*1) + (1*1) - 0.5 = 1.5 > 0 => y = 1

x1	x2	z=x1+x2−0.5	Output y (OR)
0	0	−0.5	0
0	1	0.5	1
1	0	0.5	1
1	1	1.5	1

OR gate using a single perceptron



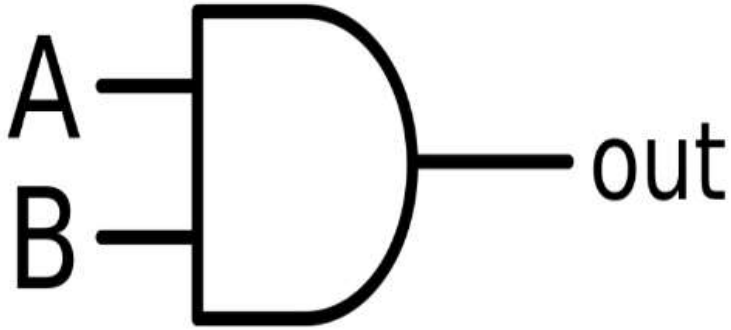
Python Implementation of OR Gate:

```
In [3]: import numpy as np
   ...:
   ...: def OR(x1, x2):
   ...:     w1, w2, b = 1, 1, -0.5
   ...:     v = w1*x1 + w2*x2 + b
   ...:     return 1 if v >= 0 else 0
   ...:
   ...: # Testing
   ...: for x1 in [0, 1]:
   ...:     for x2 in [0, 1]:
   ...:         print(f"OR({x1}, {x2}) = {OR(x1, x2)}")
   ...:
OR(0, 0) = 0
OR(0, 1) = 1
OR(1, 0) = 1
OR(1, 1) = 1

In [4]:
```

AND gate using a single perceptron

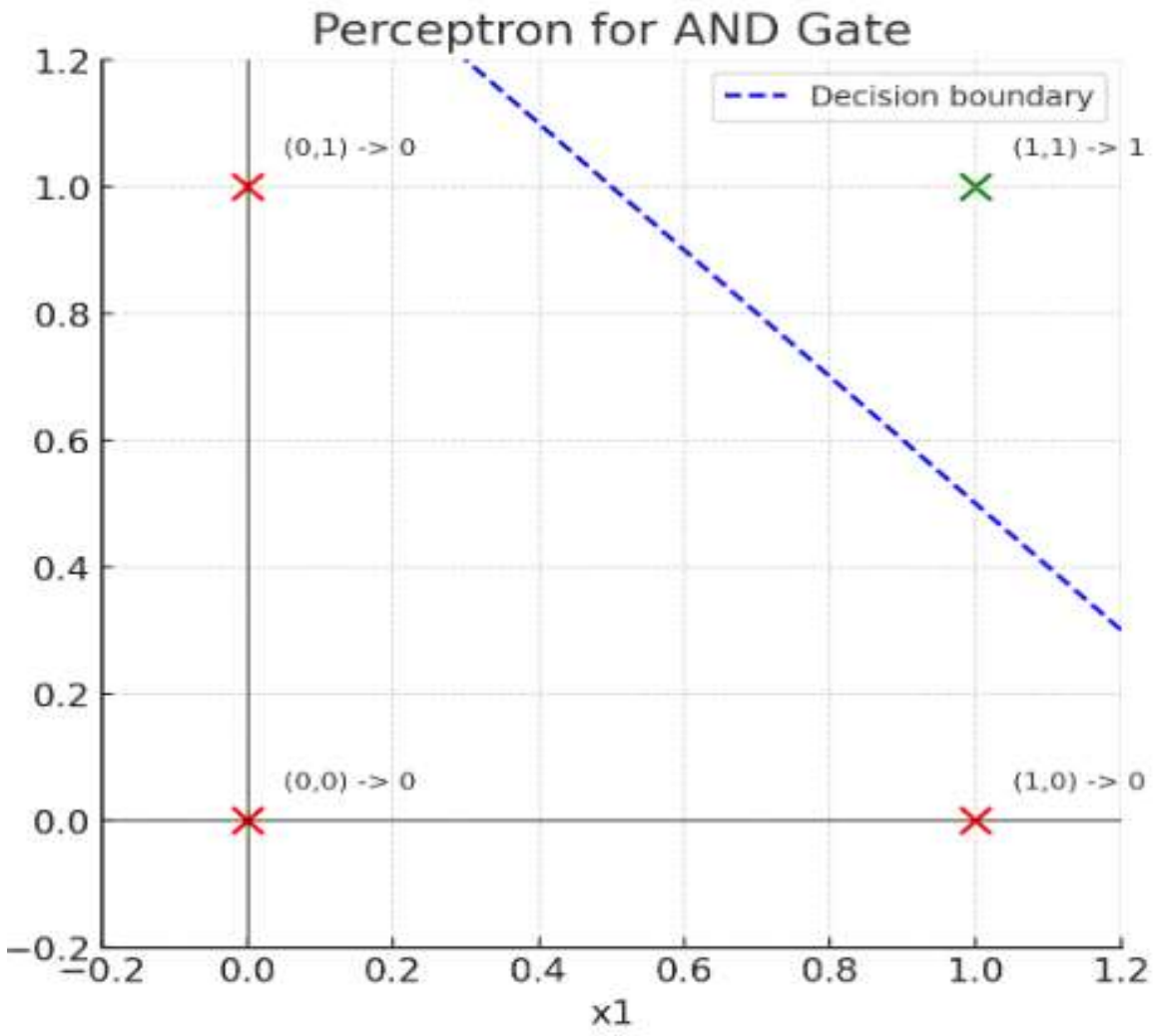
- Input $x_1, x_2 \in \{0,1\}$
- Weights $w_1 = 1\ , w_2 = 2, bias: b = -1.5$
- $z = w_1 x_1 + w_2 x_2 + b,\ y = f(z) = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}$



- Suppose
- If (x_1, x_2) = (0 , 0) : z = (1*0) + (1*0) - 1.5 = -1.5 < 0 => y = 0
- If (x_1, x_2) = (0 , 1) : z = (1*0) + (1*1) - 1.5 = -0.5 < 0 => y = 0
- If (x_1, x_2) = (1 , 0) : z = (1*1) + (1*0) - 1.5 = -0.5 < 0 => y = 0
- If (x_1, x_2) = (1 , 1) : z = (1*1) + (1*1) - 1.5 = 0.5 > 0 => y = 1

x1	x2	z=x1+x2−1.5	Output y (AND)
0	0	−1.5	0
0	1	−0.5	0
1	0	−0.5	0
1	1	0.5	1

AND gate using a single perceptron



Python Implementation of AND Gate:

```
In [4]: import numpy as np
   ...:
   ...: def AND(x1, x2):
   ...:     w1, w2, b = 1, 1, -1.5
   ...:     v = w1*x1 + w2*x2 + b
   ...:     return 1 if v >= 0 else 0
   ...:
   ...: # Testing
   ...: for x1 in [0, 1]:
   ...:     for x2 in [0, 1]:
   ...:         print(f"AND({x1}, {x2}) = {AND(x1, x2)}")
   ...:
AND(0, 0) = 0
AND(0, 1) = 0
AND(1, 0) = 0
AND(1, 1) = 1
```