# McCulloch-Pitts Neural Model

February 2, 2023

McCulloch-Pitts Neural Model for AND case

Step 1: Generate a vector of inputs and a vector of weights.

[ ]:
```
import numpy as np

input_table = np.array([
 [0,0],
 [0,1],
 [1,0],
 [1,1]
])
print(f'Input Table:\n{input_table}')
```

```
Input Table:
[[0 0]
 [0 1]
 [1 0]
 [1 1]]
```

[ ]:
```
weights = np.array([1,1])
print(f'weights: {weights}')
```

```
weights: [1 1]
```

Step 2: Compute the dot product between the matrix of inputs and weights

[ ]:
```
dot_products = np.dot(input_table, weights)
print(f'Dot products: {dot_products}')
```

```
Dot products: [0 1 1 2]
```

Step 3: Define the threshold activation function

[ ]:
```
def  linear_threshold_gate(dot, threshold):
    if dot >= threshold:
        return 1
    else:
        return 0
```

Step 4: Compute the output based on the threshold value

[ ]:
```
T = 2
for i in range(0,4):
 activation = linear_threshold_gate(dot_products[i], T)
 print(f'Activation: {activation}')
```

```
Activation: 0
Activation: 0
Activation: 0
Activation: 1
```

McCulloch-Pitts Neural Model for OR case

For OR case, Threshold can be 1 or greater than 1

[ ]:
```
input_table = np.array([
 [0,0],
 [0,1],
 [1,0],
 [1,1]
])
print(f'Input Table:\n{input_table}')
```

```
Input Table:
[[0 0]
 [0 1]
 [1 0]
 [1 1]]
```

[ ]:
```
weights = np.array([1,1])
print(f'weights: {weights}')
```

```
weights: [1 1]
```

[ ]:
```
dot_products = np.dot(input_table, weights)
print(f'Dot products: {dot_products}')
```

```
Dot products: [0 1 1 2]
```

[ ]:
```
T = 1
for i in range(0,4):
 activation = linear_threshold_gate(dot_products[i], T)
 print(f'Activation: {activation}')
```

```
Activation: 0
Activation: 1
Activation: 1
Activation: 1
```

For NOR case, Threshold will be 0 and weights will be negative

[ ]:
```
input_table = np.array([
 [0,0],
 [0,1],
 [1,0],
 [1,1]
])
print(f'Input Table:\n{input_table}')
```

```
Input Table:
[[0 0]
 [0 1]
 [1 0]
 [1 1]]
```

[ ]:
```
weights = np.array([-1,-1])
print(f'weights: {weights}')
```

```
weights: [-1 -1]
```

[ ]:
```
dot_products = np.dot(input_table, weights)
print(f'Dot products: {dot_products}')
```

```
Dot products: [ 0 -1 -1 -2]
```

[ ]:
```
T = 0
for i in range(0,4):
 activation = linear_threshold_gate(dot_products[i], T)
 print(f'Activation: {activation}')
```

```
Activation: 1
Activation: 0
Activation: 0
Activation: 0
```