Pratham Nagar 59 ML_EXP_6

```
import numpy as np
np.random.seed(seed=42)
I = np.random.choice([0,1], 3)
W = np.random.choice([-1,1], 3)
print(f'Input vector:{I}, Weight vector:{W}')
# Example output: Input vector:[1 0 1], Weight vector:[ 1 -1  1]

# Step 2: compute the dot product between the vector of inputs and weights
dot = I @ W
print(f'Dot product: {dot}')
# Example output: Dot product: 2

# Step 3: define the threshold activation function
def linear_threshold_gate(dot: int, T: float) -> int:
    if dot >= T:
        return 1
    else:
        return 0

# Step 4: compute the output based on the threshold value
T = 2
activation = linear_threshold_gate(dot, T)
print(f'Activation: {activation}')
# Example output: Activation: 1

T = 4
activation = linear_threshold_gate(dot, T)
print(f'Activation: {activation}')
# Example output: Activation: 0

# Application: boolean algebra using the McCulloch-Pitts artificial neuron
input_table = np.array([
    [0,0],
    [0,1],
    [1,0],
    [1,1]
])

print(f'input table:\n{input_table}')

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

# Step 2: compute the dot product between the matrix of inputs and weights
dot_products = input_table @ weights
print(f'Dot products: {dot_products}')
# Example output: Dot products: [ 0 -1  1  0]

# Step 3: define the threshold activation function
def linear_threshold_gate(dot: int, T: float) -> int:
    if dot >= T:
        return 1
    else:
        return 0

# Step 4: compute the output based on the threshold value
T = 1
for i in range(0,4):
    activation = linear_threshold_gate(dot_products[i], T)
    print(f'Activation: {activation}')
```

```
Input vector:[0 1 0], Weight vector:[-1 -1  1]
Dot product: -1
Activation: 0
Activation: 0
input table:
[[0 0]
 [0 1]
 [1 0]
 [1 1]]
weights: [ 1 -1]
Dot products: [ 0 -1  1  0]
```

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