```
import numpy as np
import itertools
```

```
# Initialization

dimension = 3      #to tune
variance = 1/dimension
mean = 0
weights = np.random.normal(0, np.sqrt(variance), dimension)
threshold = 0
N = 2**dimension
nIterations = 10**4  #to tune for high dimension
total_iterations = 0

#Constants
nEpochs = 20
eta = 0.05
count = 0

weights
```

```
array([-0.82016831, -0.34246551,  1.18263278])
```

```
#Binary combination overview
def generate_binary_combinations(dimension):
    for combination in itertools.product([0, 1], repeat=dimension):
        yield list(combination)
```

```
combinations_generator = generate_binary_combinations(dimension)
```

```
Patterns = list(combinations_generator)

Patterns
```

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

```
used_sample = set()

for i in range(nIterations):

    Boolean = np.random.choice([-1, 1], size=N)
    Boolean_tuple = tuple(Boolean)

    #check to avoid duplication
    if Boolean_tuple in used_sample:
        continue

    else:
        used_sample.add(Boolean_tuple)

        for epoch in range(nEpochs):
            total_distance = 0
            stop_inner_loop = False
            mu_patterns = []


            for mu in range(N):
                pattern = np.array(Patterns[mu])

                #g=sgn(b),g(0)=1
                b = np.dot(pattern, weights) - threshold
                if b == 0:
                    outputs = 1
                else:
                    outputs = np.sign(b)


                #weights and threshold iterations
                distance = Boolean[mu] - outputs
                weights += eta * distance * pattern
                threshold -= eta * distance
                total_distance += abs(distance)

        Outputs = np.dot(Patterns, weights) - threshold
        Outputs[Outputs == 0] = 1
        Outputs[Outputs != 0] = np.sign(Outputs[Outputs != 0])


        if np.all(np.array(Outputs) == Boolean):
            count += 1

print("Count:", count)
```

```
Count: 104
```