```
import numpy as np


dataset = np.array([
            [1, 0, 0, 1, 0, 1, 0, 0, 1],
            [1, 0, 0, 0, 0, 1, 1, 0, 1],
            [0, 1, 0, 0, 1, 0, 0, 1, 0],
            [0, 0, 1, 0, 1, 0, 1, 1, 0],
            [0, 0, 1, 0, 1, 0, 0, 1, 0]
            ])
print(f"Input vectors : \n {dataset}\n")

num_clusters = 3
num_iterations = 6
learning_rate = 0.6
sigma = 1

print(f"Initial learning rate : {learning_rate}")
print(f"Initial neighbourhood sign : {sigma}")
print(f"Number of features : {num_clusters}")

weights = np.array([
    [0.4, 0.9, 0.2, 0.6, 0.9, 0.4, 0.3, 0.5, 0.7],
    [0.2, 0.5, 0.3, 0.4, 0.8, 0.7, 0.6, 0.6, 0.2],
    [0.5, 0.3, 0.9, 0.8, 0.5, 0.2, 0.4, 0.2, 0.7]
    ])

print(f"\nInitial weight: \n{np.round(weights,2)}\n")


for i in range(num_iterations):

   x = dataset[np.random.randint(0, dataset.shape[0])]

   distances = np.linalg.norm(weights - x, axis=1)

   bmu_index = np.argmin(distances)
   bmu = weights[bmu_index]

   decay_factor = 0.5
   learning_rate = learning_rate * decay_factor
   sigma = sigma * decay_factor
```

```
    for j in range(num_clusters):
        distance_to_bmu = np.abs(j - bmu_index)
        neighbor_factor = np.exp(-np.square(distance_to_bmu) / (2 *
np.square(sigma)))
        delta = learning_rate * neighbor_factor * (x - weights[j])
        weights[j] += delta

    if(bmu_index == 0):
        print(f"Best matching unit for input {x} is D{bmu_index + 2}")
    elif(bmu_index == 1):
        print(f"Best matching unit for input {x} is D{bmu_index}")
    elif(bmu_index == 2):
        print(f"Best matching unit for input {x} is D{bmu_index+1}")


print(f"\nFinal weight: \n{np.round(weights,2)}")
```