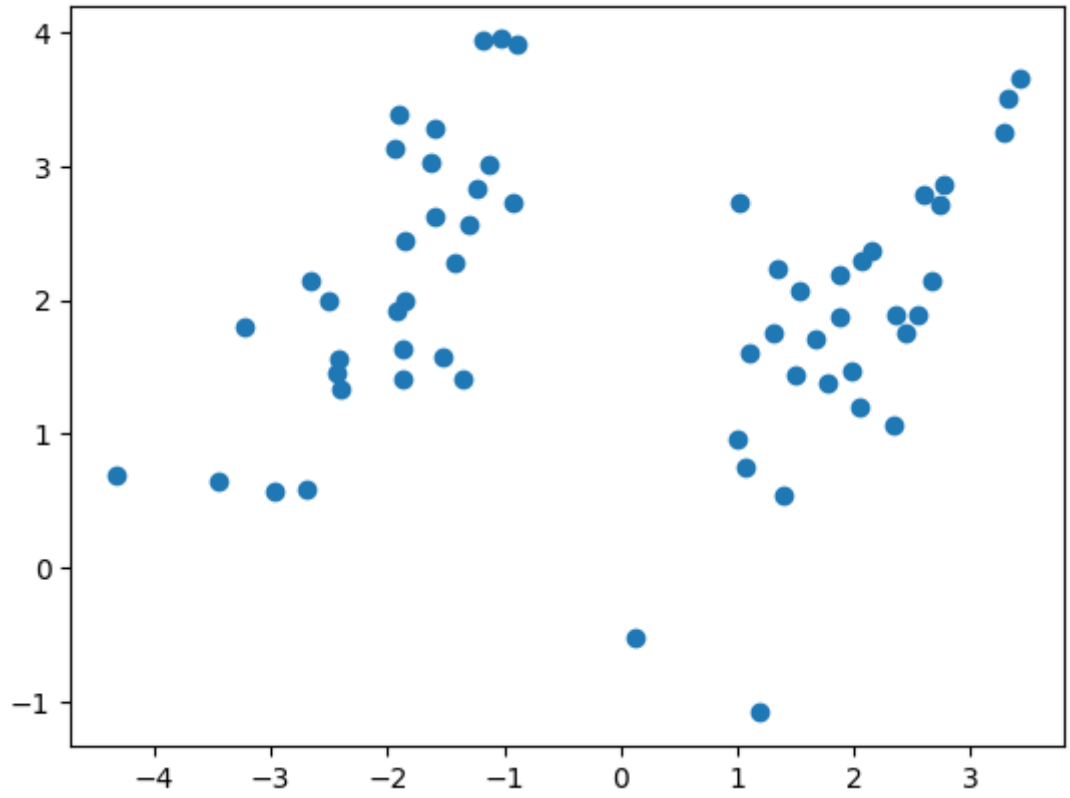
In [833…

```
from sklearn.datasets import make_classification

X, y = make_classification(n_samples=60,
                           n_features=2,
                           n_redundant=0,
                           n_informative=2,
                           n_clusters_per_class=1,
                           n_classes=2,
                           random_state=9,
                           class_sep=2)
```

In [834…

```
import matplotlib.pyplot as plt

plt.scatter(X[:, 0], X[:, 1])
```

Out[834]: <matplotlib.collections.PathCollection at 0x2628a0dd570>



In [835…

```
import numpy as np


def update_cluster_centers(X, c):
    centers = np.zeros((2, 2))
    for i in range(1, 3):
        ix = np.where(c == i)
        centers[i - 1, :] = np.mean(X[ix, :], axis=1)
    return centers
```

In [836…

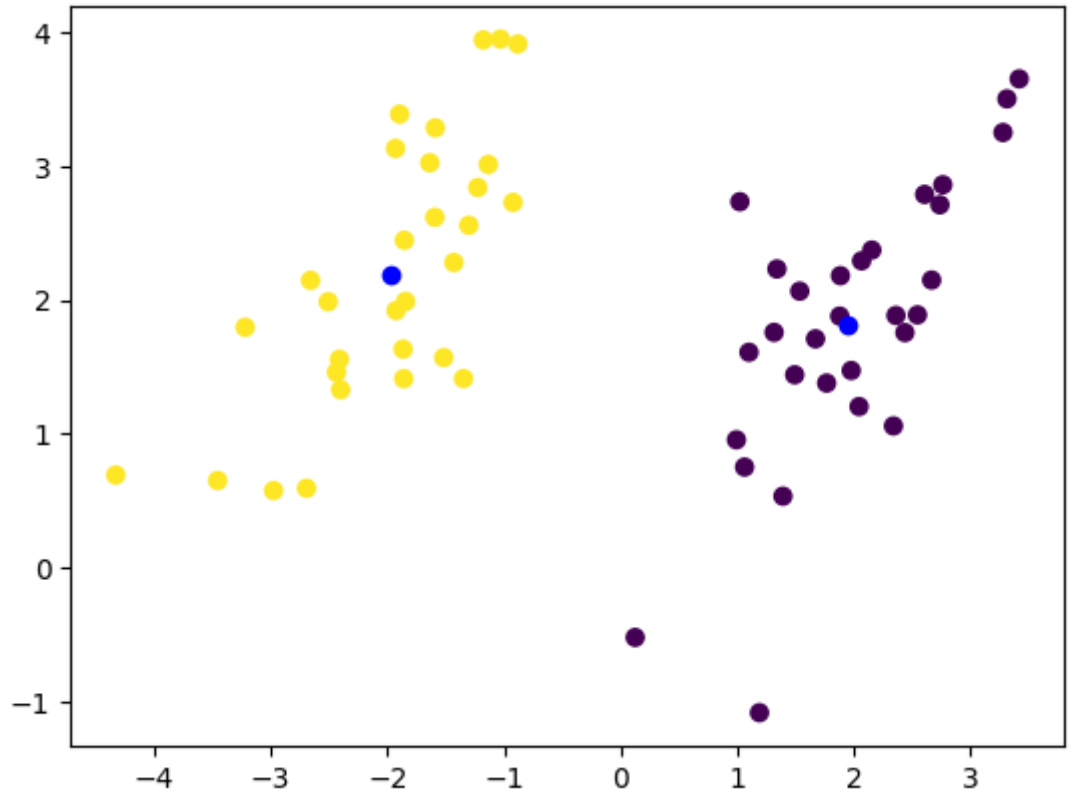
```
from scipy.cluster.hierarchy import fcluster, linkage

mergings = linkage(X, method='ward')
T = fcluster(mergings, 2, criterion='maxclust')
```

```
clusters = update_cluster_centers(X, T)
clusters
```

Out[836]:
```
array([[ 1.94772593,  1.8161229 ],
       [-1.96705206,  2.19521329]])
```

In [837…
```
plt.scatter(X[:, 0], X[:, 1], c=T)
plt.scatter(clusters[:, 0], clusters[:, 1], c='blue')
```

Out[837]:
```
<matplotlib.collections.PathCollection at 0x2628a0a8b50>
```



In [838…
```
import math


class SOM:
    def __init__(self, n, c):
        """
        n - количестов атрибутов
        C - количество кластеров
        """
        self.n = n
        self.c = c
        self.a = [0 for _ in range(n)]

    def calculate_a(self, i):
        """
        Вычисление значение шага относительного текущего выбора
        """
        return (50 - i) / 100

    def winner(self, weights, sample):
        """
        Вычисляем выигравший нейрон (вектор) по Евклидову расстоянию
        """
        d0 = 0
```

```
        d1 = 0
        for i in range(len(sample)):
            d0 += math.pow((sample[i] - weights[0][i]), 2)
            d1 += math.pow((sample[i] - weights[1][i]), 2)

        if d0 > d1:
            return 0
        else:
            return 1

    def update(self, weights, sample, j):
        """
        Обновляем значение для выигравшего нейрона
        """
        for i in range(len(weights)):
            weights[j][i] = weights[j][i] + self.calculate_a(self.a[j]) * (sampl

        print(f'\nШаг для {j} кластера = {self.calculate_a(self.a[j])}')
        self.a[j] += 1
        print(f'Веса после обновления:')
        print(weights)

        return weights
```

In [839…

```
# Обучающая выборка (m, n)
# m - объем выборки
# n - количество атрибутов в записи
np.random.shuffle(X)
T = X
m, n = len(T), len(T[0])

# Обучающие веса (n, C)
# n - количество атрибутов в записи
# C - количество кластеров
C = 2

weights = np.random.normal(100, 10, size=(n, C)) / 100
weights
```

Out[839]:

```
array([[1.05593892, 0.89880423],
       [0.9626736 , 1.10335081]])
```

In [840…

```
som = SOM(n, C)
som
```

Out[840]:

```
<__main__.SOM at 0x2628a1a6b30>
```

In [841…

```
for i in range(m):
    sample = T[i]
    J = som.winner(weights, sample)
    weights = som.update(weights, sample, J)
```

```
Шаг для 0 кластера = 0.5
Веса после обновления:
[[-0.08408794  1.86953595]
 [ 0.9626736   1.10335081]]

Шаг для 1 кластера = 0.5
Веса после обновления:
[[-0.08408794  1.86953595]
 [-0.44018272  1.54641377]]

Шаг для 1 кластера = 0.49
Веса после обновления:
[[-0.08408794  1.86953595]
 [ 1.11581734  2.11724587]]

Шаг для 1 кластера = 0.48
Веса после обновления:
[[-0.08408794  1.86953595]
 [ 0.01532176  2.99345972]]

Шаг для 1 кластера = 0.47
Веса после обновления:
[[-0.08408794  1.86953595]
 [-1.16974269  2.52066602]]

Шаг для 1 кластера = 0.46
Веса после обновления:
[[-0.08408794  1.86953595]
 [-0.08455724  0.86331803]]

Шаг для 1 кластера = 0.45
Веса после обновления:
[[-0.08408794  1.86953595]
 [-1.14218358  1.13194486]]

Шаг для 1 кластера = 0.44
Веса после обновления:
[[-0.08408794  1.86953595]
 [ 0.81668481  2.17497437]]

Шаг для 1 кластера = 0.43
Веса после обновления:
[[-0.08408794  1.86953595]
 [-0.02217253  2.53528625]]

Шаг для 1 кластера = 0.42
Веса после обновления:
[[-0.08408794  1.86953595]
 [-1.02499203  2.12466573]]

Шаг для 0 кластера = 0.49
Веса после обновления:
[[-0.78574601  1.72283682]
 [-1.02499203  2.12466573]]

Шаг для 1 кластера = 0.41
Веса после обновления:
[[-0.78574601  1.72283682]
 [ 0.36237029  2.0262595 ]]
```

```
Шаг для 0 кластера = 0.48
Веса после обновления:
[[0.2589339  1.15264783]
 [0.36237029 2.0262595 ]]

Шаг для 1 кластера = 0.4
Веса после обновления:
[[ 0.2589339   1.15264783]
 [-1.51313882  1.49293515]]

Шаг для 1 кластера = 0.39
Веса после обновления:
[[ 0.2589339   1.15264783]
 [-0.53615909  1.28352158]]

Шаг для 1 кластера = 0.38
Веса после обновления:
[[0.2589339  1.15264783]
 [0.48531024 1.69755111]]

Шаг для 0 кластера = 0.47
Веса после обновления:
[[-0.75281869  2.20354405]
 [ 0.48531024  1.69755111]]

Шаг для 0 кластера = 0.46
Веса после обновления:
[[0.06336527 2.44765815]
 [0.48531024 1.69755111]]

Шаг для 0 кластера = 0.45
Веса после обновления:
[[0.63737413 2.35058465]
 [0.48531024 1.69755111]]

Шаг для 1 кластера = 0.37
Веса после обновления:
[[0.63737413 2.35058465]
 [1.32713403 2.12803525]]

Шаг для 1 кластера = 0.36
Веса после обновления:
[[0.63737413 2.35058465]
 [0.38100989 2.28295847]]

Шаг для 0 кластера = 0.44
Веса после обновления:
[[-0.69875595  1.90156886]
 [ 0.38100989  2.28295847]]

Шаг для 1 кластера = 0.35
Веса после обновления:
[[-0.69875595  1.90156886]
 [-0.79320772  1.68580772]]

Шаг для 1 кластера = 0.34
Веса после обновления:
[[-0.69875595  1.90156886]
 [-1.00946201  1.88785065]]
```

```
Шаг для 1 кластера = 0.33
Веса после обновления:
[[-0.69875595  1.90156886]
 [-0.31346402  1.79605948]]

Шаг для 1 кластера = 0.32
Веса после обновления:
[[-0.69875595  1.90156886]
 [-0.50827349  2.0946222 ]]

Шаг для 1 кластера = 0.31
Веса после обновления:
[[-0.69875595  1.90156886]
 [-1.42120739  1.64735831]]

Шаг для 1 кластера = 0.3
Веса после обновления:
[[-0.69875595  1.90156886]
 [-0.49398308  1.66619952]]

Шаг для 1 кластера = 0.29
Веса после обновления:
[[-0.69875595  1.90156886]
 [-0.64944489  2.328225  ]]

Шаг для 1 кластера = 0.28
Веса после обновления:
[[-0.69875595  1.90156886]
 [-0.98960537  2.13357624]]

Шаг для 1 кластера = 0.27
Веса после обновления:
[[-0.69875595  1.90156886]
 [-0.43572071  1.76067885]]

Шаг для 0 кластера = 0.43
Веса после обновления:
[[ 0.69535238  1.89616509]
 [-0.43572071  1.76067885]]

Шаг для 0 кластера = 0.42
Веса после обновления:
[[-0.37743467  1.69305513]
 [-0.43572071  1.76067885]]

Шаг для 1 кластера = 0.26
Веса после обновления:
[[-0.37743467  1.69305513]
 [ 0.2087728   1.6159993 ]]

Шаг для 0 кластера = 0.41
Веса после обновления:
[[0.54944977 1.89343658]
 [0.2087728  1.6159993 ]]

Шаг для 0 кластера = 0.4
Веса после обновления:
[[-0.30551113  2.45043526]
 [ 0.2087728   1.6159993 ]]
```

```
Шаг для 1 кластера = 0.25
Веса после обновления:
[[-0.30551113  2.45043526]
 [-0.32569058  1.99509841]]

Шаг для 1 кластера = 0.24
Веса после обновления:
[[-0.30551113  2.45043526]
 [-0.62960188  2.14464535]]

Шаг для 1 кластера = 0.23
Веса после обновления:
[[-0.30551113  2.45043526]
 [-0.18239756  2.05580545]]

Шаг для 1 кластера = 0.22
Веса после обновления:
[[-0.30551113  2.45043526]
 [-0.72685682  2.07583821]]

Шаг для 1 кластера = 0.21
Веса после обновления:
[[-0.30551113  2.45043526]
 [-0.08329217  1.8624351 ]]

Шаг для 0 кластера = 0.39
Веса после обновления:
[[ 0.76358169  2.17982486]
 [-0.08329217  1.8624351 ]]

Шаг для 1 кластера = 0.2
Веса после обновления:
[[0.76358169 2.17982486]
 [0.46631361 1.9197792 ]]

Шаг для 1 кластера = 0.19
Веса после обновления:
[[0.76358169 2.17982486]
 [0.7699791  1.99065567]]

Шаг для 1 кластера = 0.18
Веса после обновления:
[[0.76358169 2.17982486]
 [1.10024453 2.13428743]]

Шаг для 1 кластера = 0.17
Веса после обновления:
[[0.76358169 2.17982486]
 [0.63526814 2.28574453]]

Шаг для 0 кластера = 0.38
Веса после обновления:
[[-0.74917971  2.03412719]
 [ 0.63526814  2.28574453]]

Шаг для 0 кластера = 0.37
Веса после обновления:
[[0.74027479 2.48444842]
 [0.63526814 2.28574453]]
```

```
Шаг для 0 кластера = 0.36
Веса после обновления:
[[-0.01090303  2.09882641]
 [ 0.63526814  2.28574453]]

Шаг для 0 кластера = 0.35
Веса после обновления:
[[0.61064133 1.84710675]
 [0.63526814 2.28574453]]

Шаг для 1 кластера = 0.16
Веса после обновления:
[[0.61064133 1.84710675]
 [0.10295908 2.01512877]]

Шаг для 1 кластера = 0.15
Веса после обновления:
[[0.61064133 1.84710675]
 [0.10617462 1.63484793]]

Шаг для 1 кластера = 0.14
Веса после обновления:
[[0.61064133 1.84710675]
 [0.3001162  1.60779736]]

Шаг для 1 кластера = 0.13
Веса после обновления:
[[0.61064133 1.84710675]
 [0.70507393 1.87353292]]

Шаг для 1 кластера = 0.12
Веса после обновления:
[[0.61064133 1.84710675]
 [0.51477633 2.11821223]]

Шаг для 1 кластера = 0.11
Веса после обновления:
[[0.61064133 1.84710675]
 [0.66462354 2.0919697 ]]

Шаг для 1 кластера = 0.1
Веса после обновления:
[[0.61064133 1.84710675]
 [0.40578432 2.07491548]]

Шаг для 1 кластера = 0.09
Веса после обновления:
[[0.61064133 1.84710675]
 [0.54710545 2.0207957 ]]

Шаг для 1 кластера = 0.08
Веса после обновления:
[[0.61064133 1.84710675]
 [0.6261041  2.02443088]]

Шаг для 0 кластера = 0.34
Веса после обновления:
[[-0.22708821  2.0508661 ]
 [ 0.6261041   2.02443088]]
```

In [842…]
```
s = X[0]
J = som.winner(weights, s)

print(f"Элемент принадлежит к {J} кластеру, на самом деле к {y[0]} кластеру")
print("Обученные веса: ")
print(weights)
```

```
Элемент принадлежит к 1 кластеру, на самом деле к 1 кластеру
Обученные веса:
[[-0.22708821  2.0508661 ]
 [ 0.6261041   2.02443088]]
```

In [843…]
```
predicted = np.array([som.winner(weights, s) for s in X])
predicted
```

Out[843]:
```
array([1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1])
```

In [844…]
```
y == predicted
```

Out[844]:
```
array([ True,  True,  True, False, False, False, False,  True,  True,
        True,  True,  True, False, False, False,  True,  True, False,
       False,  True, False,  True, False,  True,  True,  True, False,
        True,  True,  True, False, False,  True,  True,  True,  True,
       False,  True,  True,  True,  True,  True,  True, False,  True,
        True,  True,  True, False,  True,  True,  True,  True, False,
        True,  True,  True,  True,  True,  True])
```

In [845…]
```
from sklearn.metrics import accuracy_score

print(f'Точность кластеризации: {accuracy_score(y, predicted) * 100}%')
```

```
Точность кластеризации: 70.0%
```

In [845…]
```

```