* Классификация относит каждый объект к одной из заранее определенных групп
* Кластеризация разбивает множество объектов на группы, которые формируются в процессе кластеризации.

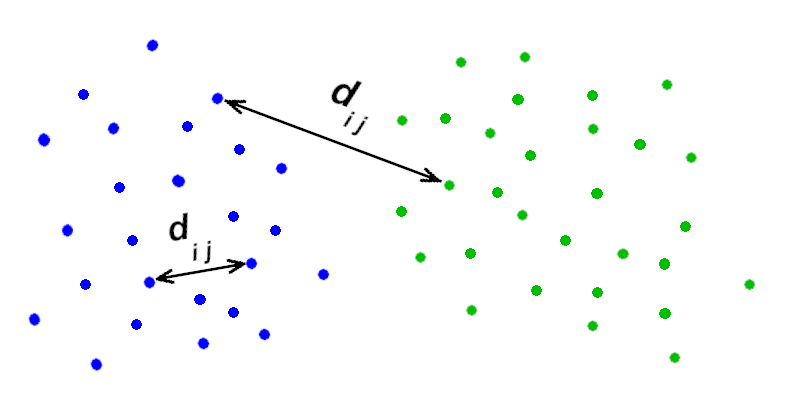
"Главное отличие кластеризации от классификации состоит в том, что перечень и характеристики групп не заданы и определяется в процессе работы алгоритма"

***Кластеризация – разбиение множества объектов на группы (кластеры) так, чтобы объекты одной группы были более похожи друг на друга чем объекты из разных групп.***

Необходимо договориться, что означает более похожи.

***Формализация задачи, построение математической модели:***

* Реальные объекты -> набор характеристик объекта -> множество элементов как совокупности характеристик.
* Пространство характеристик, вводим метрику в пространстве характеристик. Схожесть объектов ⬄ расстояние между точками в пространстве характеристик.

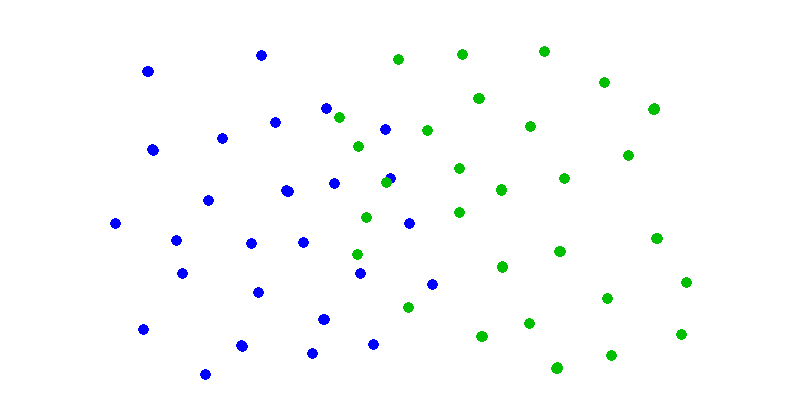


Можно ввести формальное определение кластеризации данных:

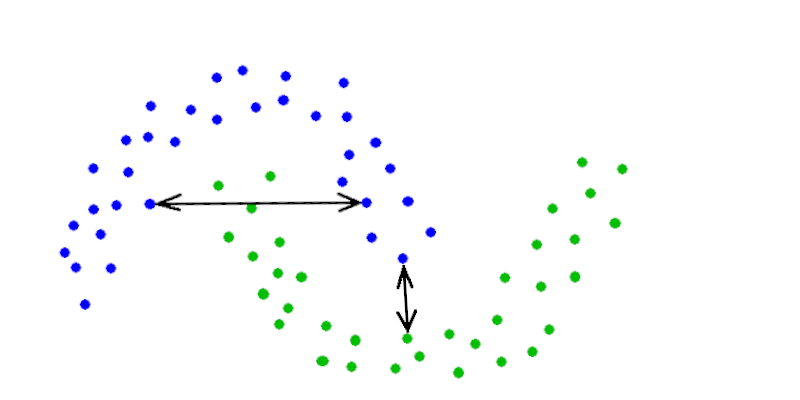
здесь – набор кластеров, – расстояние между парой объектов в соответствии с выбранной метрикой.

При таком определении определении требуется задать число кластеров. Такое определение кластеризации даёт хорошие результаты для кластеров «разнесённых» в пространстве.

В случае перекрывающихся кластеров



И в других случаях



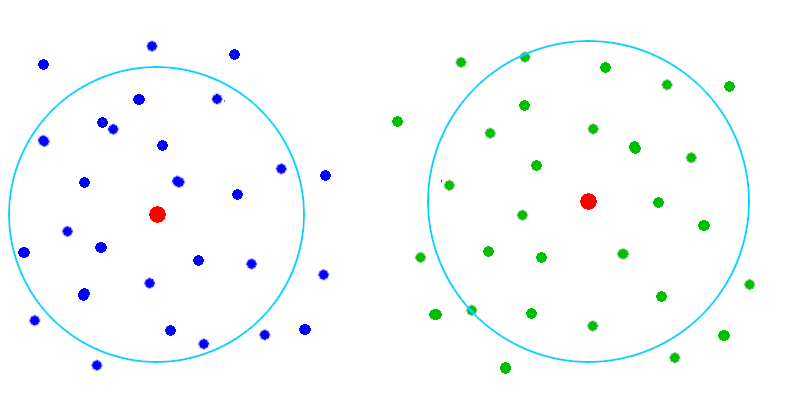
В некоторых случаях может приводить к неудовлетворительным результатам.

В подобных ситуациях используются другие определения кластеризации. Например, метод ближайших соседей или кластеризация, основанная на плотности.

**Метод** **К-средних (K-means)**

Учёт парных расстояний требует порядка O операций, что может оказаться трудоёмкой задачей для больших набора данных большого размера.

В методе К-средних каждый кластер представлен координатами его центра (и радиуса, если необходимо) в пространстве характеристик. Другими словами: От кластера, заданного как совокупность точек, мы переходим к кластеру, определённому как область пространства.



Решение задачи кластеризации сводится к минимизации квадратичных отклонений точек от центра кластера:

где – координаты центра кластера .

При этом, центр кластера определяется как среднее арифметическое координат входящих в него точек:

Радиус кластера может быть определён через разброс точек от центра кластера:

Алгоритм кластеризации можно представить в виде следующей последовательности шагов:

1. Задаём число кластеров и положение центров кластеров
2. Каждую точку входного множества включаем в тот кластер, центр которого оказался ближайшим
3. Вычисляем новые значения центров кластеров, в соответствии с распределением точек, полученным на предыдущем шаге
4. Повторяем шаги 2 и 3 до тех пор, пока распределение точек по кластерам не стабилизируется

Алгоритм завершается за конечное число шагов так как количество разбиений конечного множества на заданное число кластеров конечно и на каждой итерации суммарное квадратичное отклонение уменьшается.

Программная реализация:

import matplotlib.pyplot as plt

import random

from math import sqrt

from math import sin

from math import cos

POINT\_N = 300

CLUST\_N = 2

def DataClouds (A, N):

for i in range(N):

if random.random() < 0.5:

A[i].clust = 0

A[i].X = random.normalvariate(-1.0, 0.5)

A[i].Y = random.normalvariate(0.0, 0.2)

else:

A[i].clust = 1

A[i].X = random.normalvariate(1.0, 0.2)

A[i].Y = random.normalvariate(0.0, 0.75)

return

def DataMoons (A, N):

for i in range(N):

f = 3.14 \* random.random()

r = 0.2 \* random.normalvariate(0.0, 0.4) + 0.9

if random.random() < 0.5:

A[i].clust = 0

A[i].X = 0.5 + r \* cos(f)

A[i].Y = -0.25 + r \* sin(f)

else:

A[i].clust = 1

A[i].X = -0.5 + r \* cos(f)

A[i].Y = 0.25 - r \* sin(f)

return

class POINT:

def \_\_init\_\_(self, cl, x, y):

self.clust = cl

self.X = x

self.Y = y

class CLUSTER(POINT):

def \_\_init\_\_(self, cl, x, y):

POINT.\_\_init\_\_(self, cl, x, y)

self.N = 0

def Dist(self, p):

return sqrt( (self.X - p.X)\*\*2 + (self.Y - p.Y)\*\*2 )

def Eval\_Center(self, P):

self.N = 0

self.X = 0.0

self.Y = 0.0

for p in P:

if p.clust == self.clust:

self.N += 1

self.X += p.X

self.Y += p.Y

self.X /= self.N

self.Y /= self.N

Cl = [CLUSTER(0, 2.0\*random.random()-1.0, 2.0\*random.random()-1.0), CLUSTER(1, 2.0\*random.random()-1.0, 2.0\*random.random()-1.0)]

PP = [POINT(CLUST\_N, 0.0, 0.0) for i in range(POINT\_N)]

DataClouds(PP, POINT\_N)

#DataMoons(PP, POINT\_N)

for p in PP:

plt.scatter(p.X, p.Y, c='black', s=20)

plt.show()

while (1):

CC = [POINT(0, Cl[0].X, Cl[0].Y), POINT(1, Cl[1].X, Cl[1].Y)]

for cl in Cl:

cl.Eval\_Center(PP)

for p in PP:

if Cl[0].Dist(p) < Cl[1].Dist(p):

p.clust = 0

plt.scatter(p.X, p.Y, c='blue', s=20)

else:

p.clust = 1

plt.scatter(p.X, p.Y, c='green', s=20)

for cl in Cl:

plt.scatter(CC[cl.clust].X, CC[cl.clust].Y, c='cyan', marker="\*")

plt.scatter(cl.X, cl.Y, c='red', marker="\*")

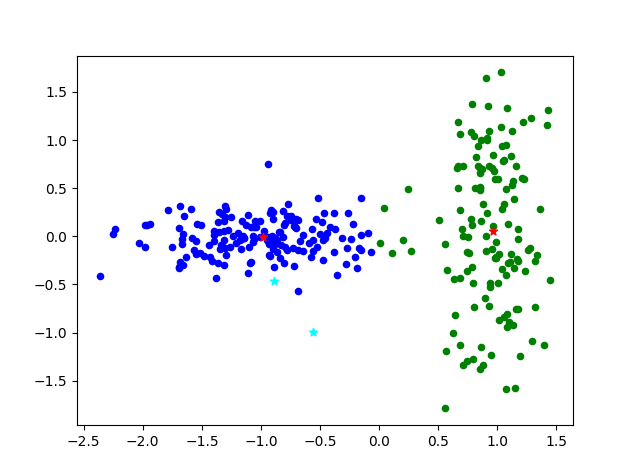
plt.show()

if (Cl[0].Dist(CC[0]) + Cl[1].Dist(CC[1])) < 0.1:

break

Изображение выглядит как диаграмма, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.



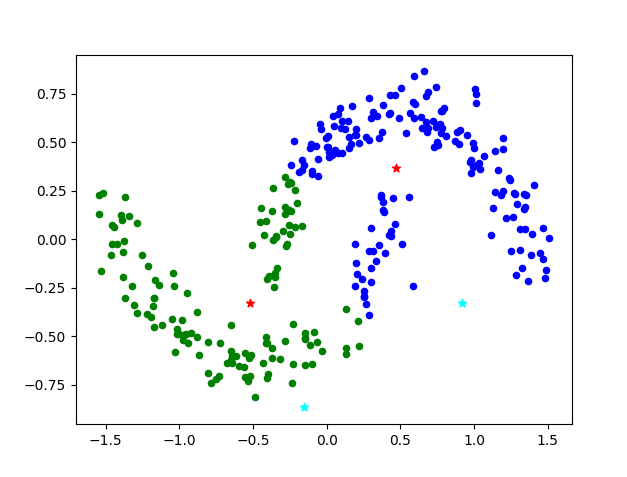
Изображение выглядит как снимок экрана, Красочность, диаграмма, линия

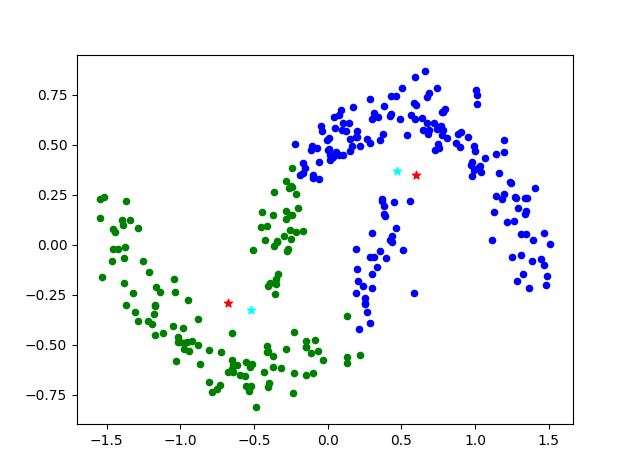
Контент, сгенерированный ИИ, может содержать ошибки.

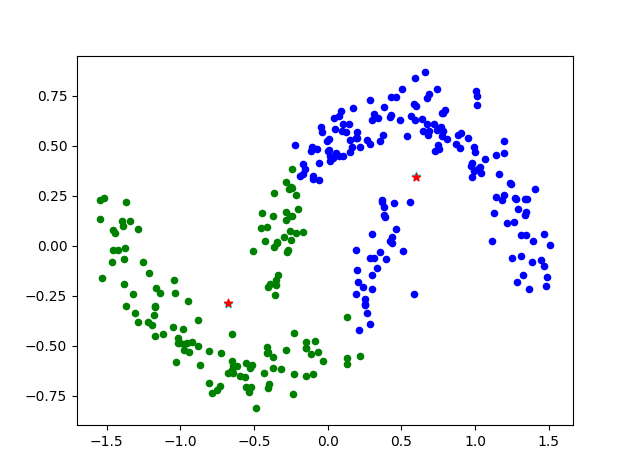
Другой пример:

Изображение выглядит как снимок экрана, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.







Недостатки алгоритма:

* Применим не для всех видов кластеров (см. картинки выше).
* Требуется задавать число кластеров.
* Не гарантируется достижение глобального минимума суммарного квадратичного отклонения.
* Результат зависит от выбора исходных центров кластеров, оптимальный выбор неизвестен.

# Развитие алгоритма

Имеются точки, не принадлежащие ни одному из кластеров:

Изображение выглядит как круг, Красочность, шаблон

Контент, сгенерированный ИИ, может содержать ошибки.

Введём радиус кластера:

Для нормального распределения:

Прежний критерий принадлежности кластеру

заменяем на

***(Реализовать самостоятельно)***

Форма кластеров отлична от сферы.

Алгоритм кластеризации Kmeans принадлежит к классу центроидных алгоритмов: кластер может быть представлен как центр и некоторая выпуклая область вокруг него.

Изображение выглядит как Цвет электрик, синий

Контент, сгенерированный ИИ, может содержать ошибки.

Левый кластер - центроидный, правый нет.

Вместо метрики Эвклида:

используем метрику Махаланобиса:

где – матрица, обратная матрице ковариации

Нечёткая кластеризация (Fuzzy Classifier Means - FCM)

В основе алгоритма FCM лежит использование матрицы , описывающей вероятность того, что i-й элемент принадлежит кластеру с номером k:

Параметр q задаёт "разброс" относительно центра кластера: чем больше значение q, тем больше вероятность включить в кластер далёкие точки.

Сам алгоритм аналогичен алгоритму К-средних, описанному ранее, с очевидной заменой критерия принадлежности объекта кластеру на 2-м шаге алгоритма.