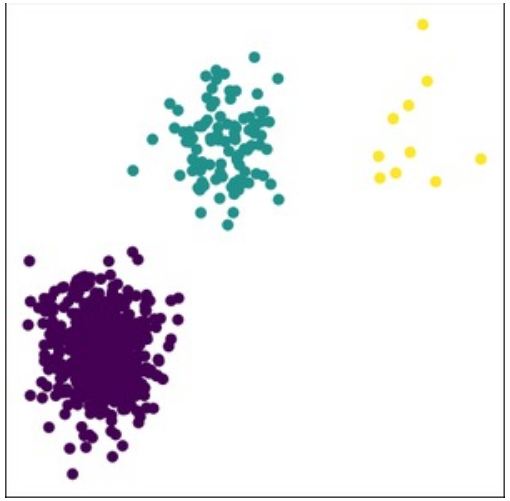
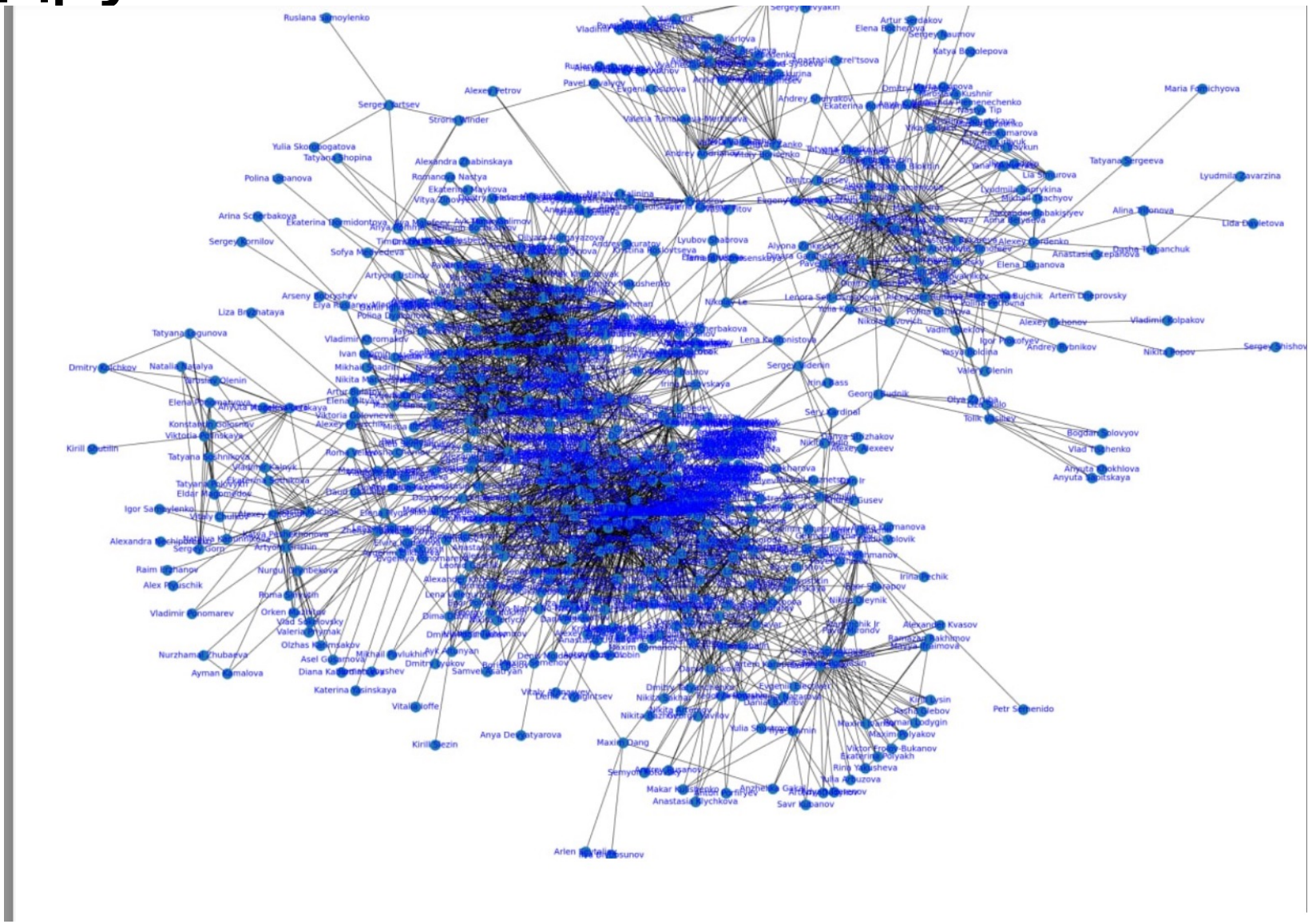
Clustering

Кластеризация

- **Кластеризация** — задача группировки множества объектов на подмножества (**кластеры**) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров по какому-либо критерию.
- Задача кластеризации относится к классу задач обучения без учителя.



Граф друзей

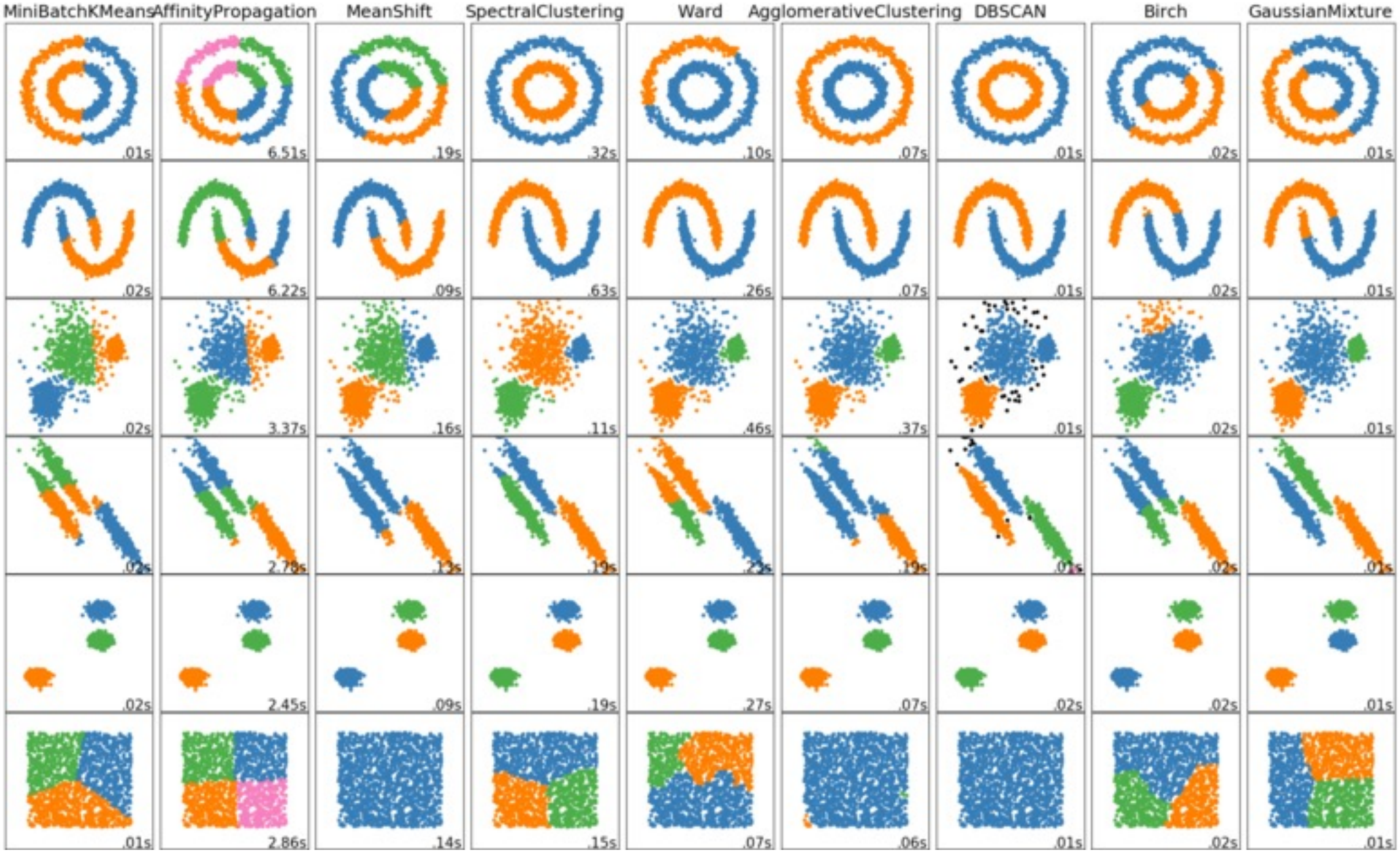


- Пусть X — множество объектов, Y — множество идентификаторов (меток) кластеров. На множестве XX задана функция расстояния между объектами $\rho(x,x')$. Дана конечная обучающая выборка объектов $Xm=\{x1,\ldots,xm\}\subset X$. Необходимо разбить выборку на подмножества (кластеры), то есть каждому объекту $xi\in Xm$ сопоставить метку $yi\in Y$, таким образом чтобы объекты внутри каждого кластера были близки относительно метрики ρ, а объекты из разных кластеров значительно различались.

- Множество Y в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров, с точки зрения того или иного *критерия качества*кластеризации.Кластеризация (обучение без учителя) отличается от классификации (обучения с учителем) тем, что метки объектов из обучающей выборки yi изначально не заданы, и даже может быть неизвестно само множество Y.

Решение задачи кластеризации объективно неоднозначно по ряду причин:

- Не существует однозначного критерия качества кластеризации. Известен ряд алгоритмов, осуществляющих разумную кластеризацию "по построению", однако все они могут давать разные результаты. Следовательно, для определения качества кластеризации и оценки выделенных кластеров необходим эксперт предметной области;
- Число кластеров, как правило, заранее не известно и выбирается по субъективным критериям. Даже если алгоритм не требует изначального знания о числе классов, конкретные реализации зачастую требуют указать этот параметр;
- Результат кластеризации существенно зависит от метрики. Однако существует ряд рекомендаций по выбору метрик для определенных классов задач.


MiniBatchKMeans
AffinityPropagation
MeanShift
SpectralClustering
Ward
AgglomerativeClustering
DBSCAN
Birch
GaussianMixture
.01s
6.51s
.19s
.32s
.10s
.07s
.01s
.02s
.01s
.02s
6.22s
.09s
.63s
.26s
.07s
.01s
.02s
.01s
.02s
3.37s
.16s
.11s
.46s
.37s
.01s
.02s
.01s
.02s
2.78s
.13s
.19s
.23s
.19s
.01s
.02s
.01s
.02s
2.45s
.09s
.19s
.27s
.07s
.02s
.02s
.01s
.01s
2.86s
.14s
.15s
.07s
.06s
.01s
.02s
.01s

Методы кластеризации

- Графовые алгоритмы кластеризации. Наиболее примитивный класс алгоритмов. В настоящее время практически не применяется на практике;
- Вероятностные алгоритмы кластеризации. Каждый объект из обучающей выборки относится к каждому из кластеров с определенной степенью вероятности:
  - EM-алгоритм;
- Иерархические алгоритмы кластеризации. Упорядочивание данных путем создания иерархии вложенных кластеров;
- Алгоритм k-средних[на 28.01.19 не создан] (англ. k-*means*). Итеративный алгоритм, основанный на минимизации суммарного квадратичного отклонения точек кластеров от центров этих кластеров;
- Распространение похожести (англ. *affinity propagation*). Распространяет сообщения о похожести между парами объектов для выбора типичных представителей каждого кластера;
- Сдвиг среднего значения (англ. *mean shift*). Выбирает центроиды кластеров в областях с наибольшей плотностью;
- Спектральная кластеризация (англ. *spectral clustering*). Использует собственные значения матрицы расстояний для понижения размерности перед использованием других методов кластеризации;
- Основанная на плотности пространственная кластеризация для приложений с шумами (англ. *Density-based spatial clustering of applications with noise*, *DBSCAN*). Алгоритм группирует в один кластер точки в области с высокой плотностью. Одиноко расположенные точки помечает как шум.

Меры качества кластеризации

Для оценки качества кластеризации задачу можно переформулировать в терминах задачи дискретной оптимизации. Необходимо так сопоставить объектам из множества X метки кластеров, чтобы значение выбранного функционала качества приняло наилучшее значение. В качестве примера, стремятся достичь минимума среднего внутрикластерного расстояния $F_0 = \frac{\sum_{i<j} [y_i = y_j] \cdot \rho(x_i, x_j)}{\sum_{i<j} [y_i = y_j]}$ или максимума среднего межкластерного расстояния $F_1 = \frac{\sum_{i<j} [y_i \neq y_j] \cdot \rho(x_i, x_j)}{\sum_{i<j} [y_i \neq y_j]}$.

DBCSAN

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise) - это алгоритм кластеризации, который основывается на плотности данных. Он может эффективно обнаруживать кластеры произвольной формы в пространстве данных, не требуя заранее заданного числа кластеров. Алгоритм DBSCAN определяет кластеры как непрерывные области высокой плотности данных, разделенные областями низкой плотности (шумом).

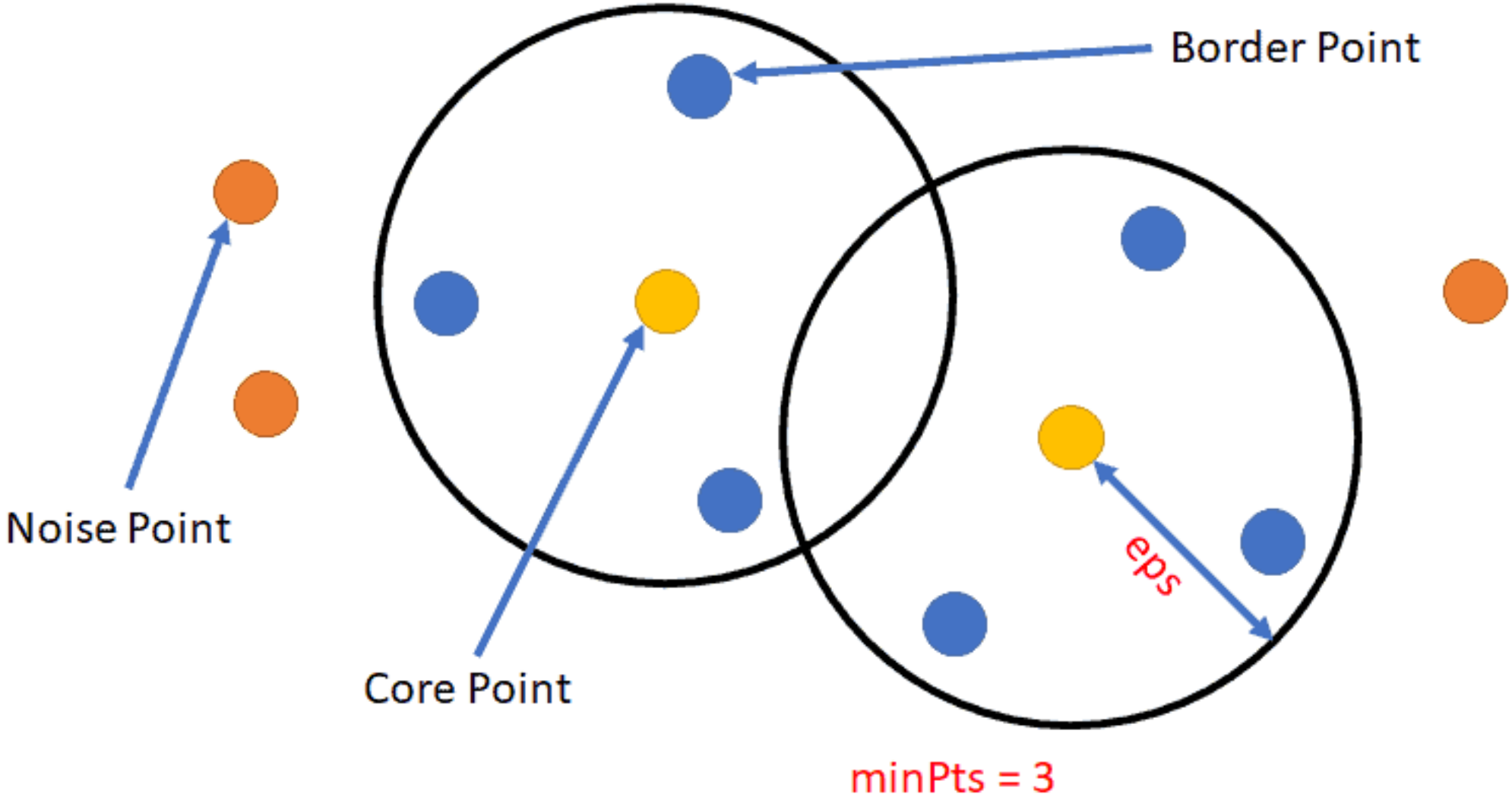
Определения

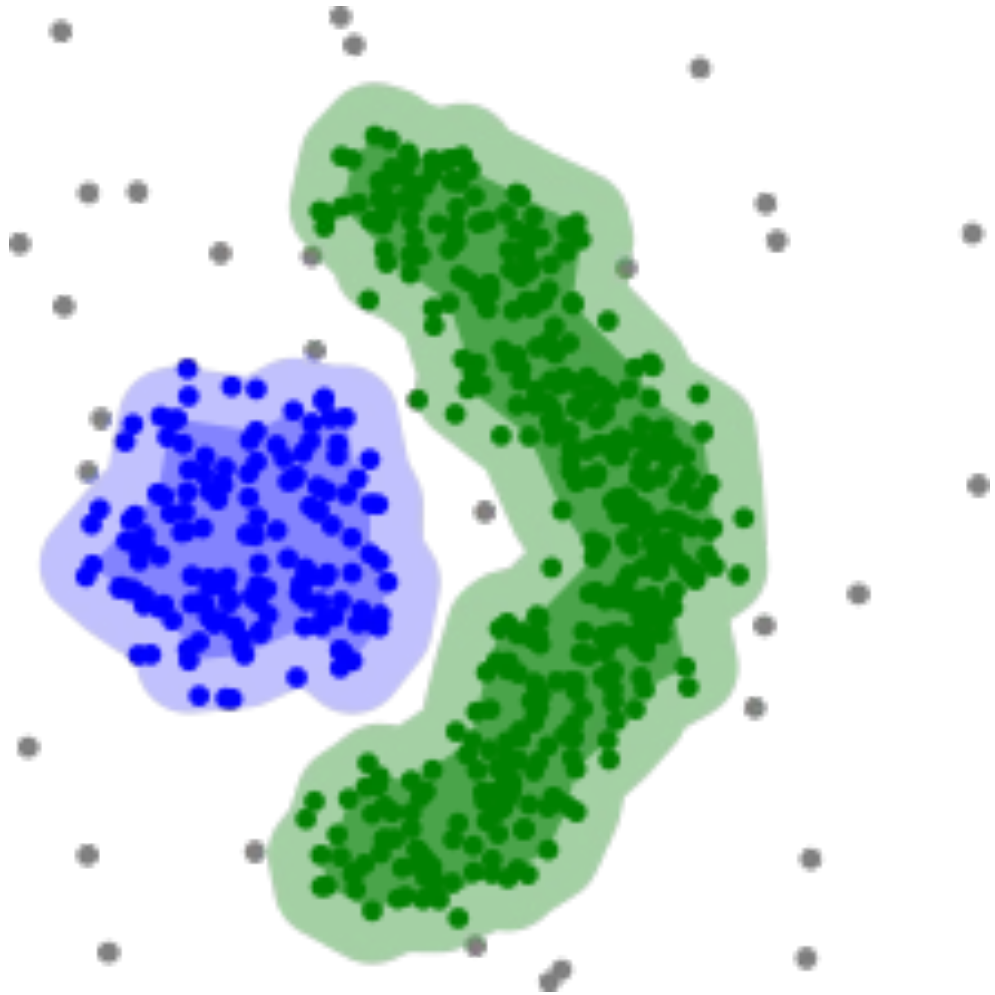
1. **Eps (ε)**: Это параметр, который определяет радиус окрестности каждой точки. Точки считаются соседями, если расстояние между ними меньше или равно ε.
2. **MinPts**: Это минимальное количество точек, которые должны находиться в радиусе ε для того, чтобы точка считалась основной (core point). Если в окрестности точки находится не менее MinPts точек (включая саму точку), то эта точка считается основной.
3. **Core point**: Точка, которая имеет по крайней мере MinPts точек в радиусе ε.
4. **Border point**: Точка, которая находится внутри окрестности другой основной точки, но сама не является основной и имеет меньшее количество соседей, чем MinPts.
5. **Noise point (шум)**: Точка, которая не является ни основной, ни граничной.

Шаги алгоритма DBSCAN

- Выбор случайной точки из набора данных.
- Нахождение всех соседей этой точки в пределах радиуса ε.
- Если число соседей больше или равно MinPts, то это основная точка.
- Рекурсивно находятся все соседи этих соседей и так далее, пока не будут обработаны все достижимые точки.
- Повторяется процесс для всех точек в наборе данных.

- В результате выполнения алгоритма получаются кластеры точек, которые достижимы из основных точек. Точки, которые не принадлежат ни к одному кластеру, могут рассматриваться как шум или выбросы.
- DBSCAN показывает хорошие результаты на данных с присутствием шума и кластерами произвольной формы, но он чувствителен к выбору параметров ε и MinPts.


Border Point
Noise Point
Core Point
eps
minPts = 3



```
DBSCAN(DB, distFunc, eps, minPts) {
   C=0                                                  /* Счётчик кластеров */
   for each point P in database DB {
      if label(P) ≠ undefined then continue              /* Точка была просмотрена во внутреннем цикле */
      Neighbors N=RangeQuery(DB, distFunc, P, eps)      /* Находим соседей */
      if|N|< minPts then {                              /* Проверка плотности */
         label(P)=Noise                                 /* Помечаем как шум */
         continue
      }
      C=C + 1                                           /* следующая метка кластера */
      label(P)=C                                        /* Помечаем начальную точку */
      Seed set S=N \ {P}                                /* Соседи для расширения */
      for each point Q in S {                            /* Обрабатываем каждую зачаточную точку */
         if label(Q)=Noise then label(Q)=C             /* Заменяем метку Шум на Край */
         if label(Q) ≠ undefined then continue          /* Была просмотрена */
         label(Q)=C                                     /* Помечаем соседа */
         Neighbors N=RangeQuery(DB, distFunc, Q, eps)   /* Находим соседей */
         if|N|≥ minPts then {                           /* Проверяем плотность */
            S=S ∪ N                                     /* Добавляем соседей в набор зачаточных точек */
         }
      }
   }
}
```

K-means

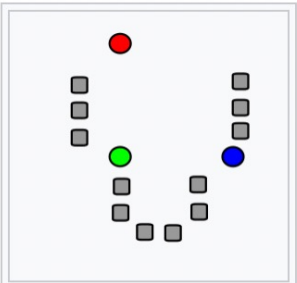
- K-means - это простой и популярный алгоритм кластеризации, который разбивает набор данных на заранее определенное количество кластеров (k). Каждый кластер представляет собой группу точек, которые имеют наименьшее среднее расстояние до центроида кластера. Алгоритм стремится минимизировать сумму квадратов расстояний от каждой точки до центроида своего кластера.

Шаги алгоритма K-means

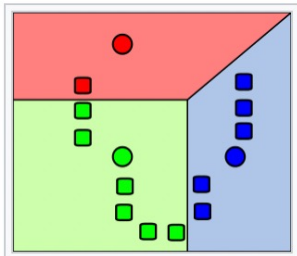
1. **Инициализация центроидов**: Начинается с выбора случайных центроидов для каждого из k кластеров. Центроид - это точка, которая представляет среднее значение всех точек в кластере.
2. **Присвоение точек к ближайшему центроиду**: Каждая точка в наборе данных присваивается к ближайшему центроиду на основе их расстояния. Обычно используется евклидово расстояние.
3. **Пересчет центроидов**: После присвоения всех точек кластерам пересчитываются центроиды путем вычисления среднего значения точек в каждом кластере.
4. **Повторение шагов 2 и 3**: Шаги 2 и 3 повторяются до тех пор, пока центроиды не перестанут изменяться или пока не будет достигнуто максимальное количество итераций.
5. **Сходимость**: Алгоритм считается сойденным, когда изменения центроидов меньше определенного порога или когда достигнуто максимальное количество итераций.
6. **Оценка качества кластеризации**: После сходимости можно оценить качество кластеризации с использованием различных метрик, таких как сумма квадратов расстояний (SSE), индекс Silhouette и другие.

$$V = \sum_{i=1}^{k} \sum_{x \in S_i} (x - \mu_i)^2$$

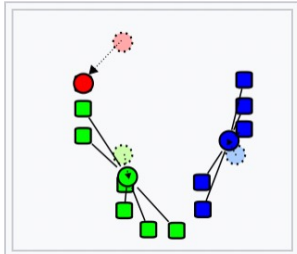
где k — число кластеров, S_i — полученные кластеры, $i = 1, 2, \ldots, k$, а μ_i — центры масс всех векторов x из кластера S_i.



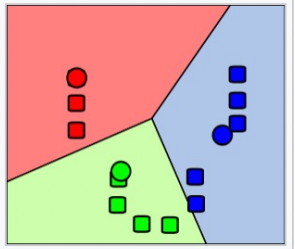
Исходные точки и случайно выбранные начальные центры.



Точки, отнесённые к начальным центрам. Разбиение на плоскости — диаграмма Вороного относительно начальных центров.



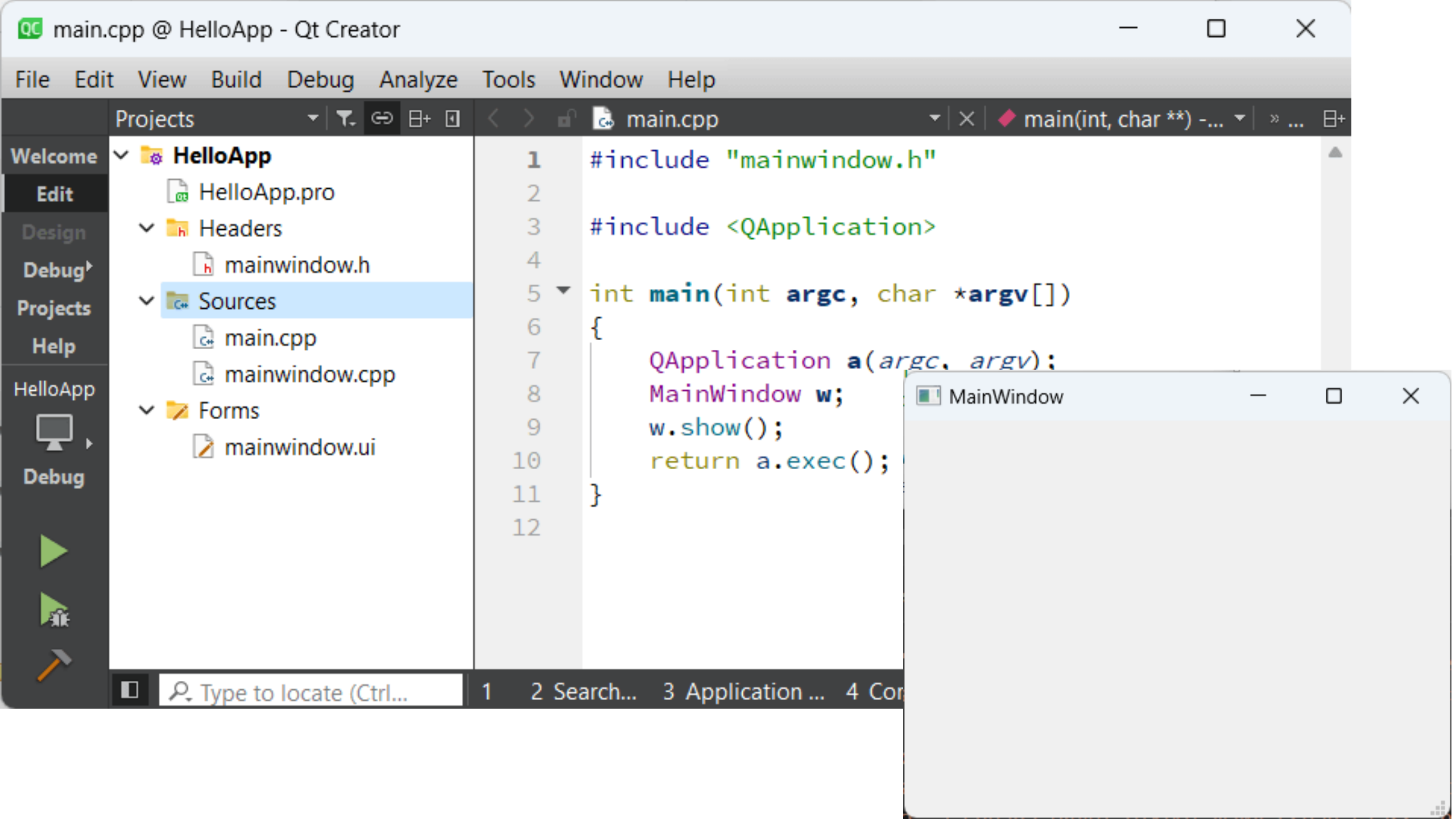
Вычисление новых центров кластеров (Ищется центр масс).



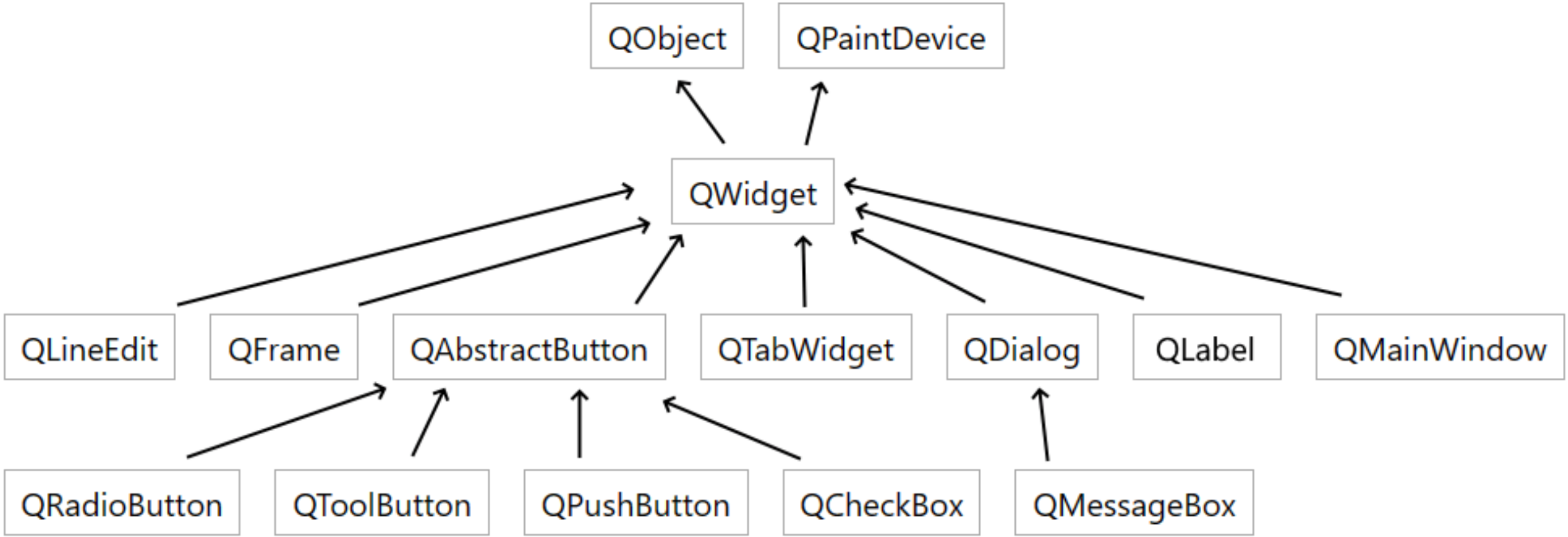
Предыдущие шаги, за исключением первого, повторяются, пока алгоритм не сойдётся.

Особенности

- Простота реализации и понимания.
- Эффективен на больших наборах данных.
- Работает хорошо на сферических или гауссовых кластерах.
- Чувствителен к начальному выбору центроидов, поэтому может сойтись к локальному оптимуму.
- Не подходит для кластеризации с произвольными формами кластеров или шумовыми точками.


main.cpp @ HelloApp - Qt Creator
File Edit View Build Debug Analyze Tools Window Help
Projects
HelloApp
HelloApp.pro
Headers
mainwindow.h
Sources
main.cpp
mainwindow.cpp
Forms
mainwindow.ui
Welcome
Edit
Design
Debug
Projects
Help
HelloApp
Debug
main.cpp
main(int, char **) -...
#include "mainwindow.h"
#include <QApplication>
int main(int argc, char *argv[])
{
QApplication a(argc, argv);
MainWindow w;
w.show();
return a.exec();
}
Type to locate (Ctrl...
1 2 Search... 3 Application ... 4 Cor
MainWindow

```
QT += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++17

# You can make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
    main.cpp \
    mainwindow.cpp

HEADERS += \
    mainwindow.h

FORMS += \
    mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```


QObject
QPaintDevice
QWidget
QLineEdit
QFrame
QAbstractButton
QTabWidget
QDialog
QLabel
QMainWindow
QRadioButton
QToolButton
QPushButton
QCheckBox
QMessageBox