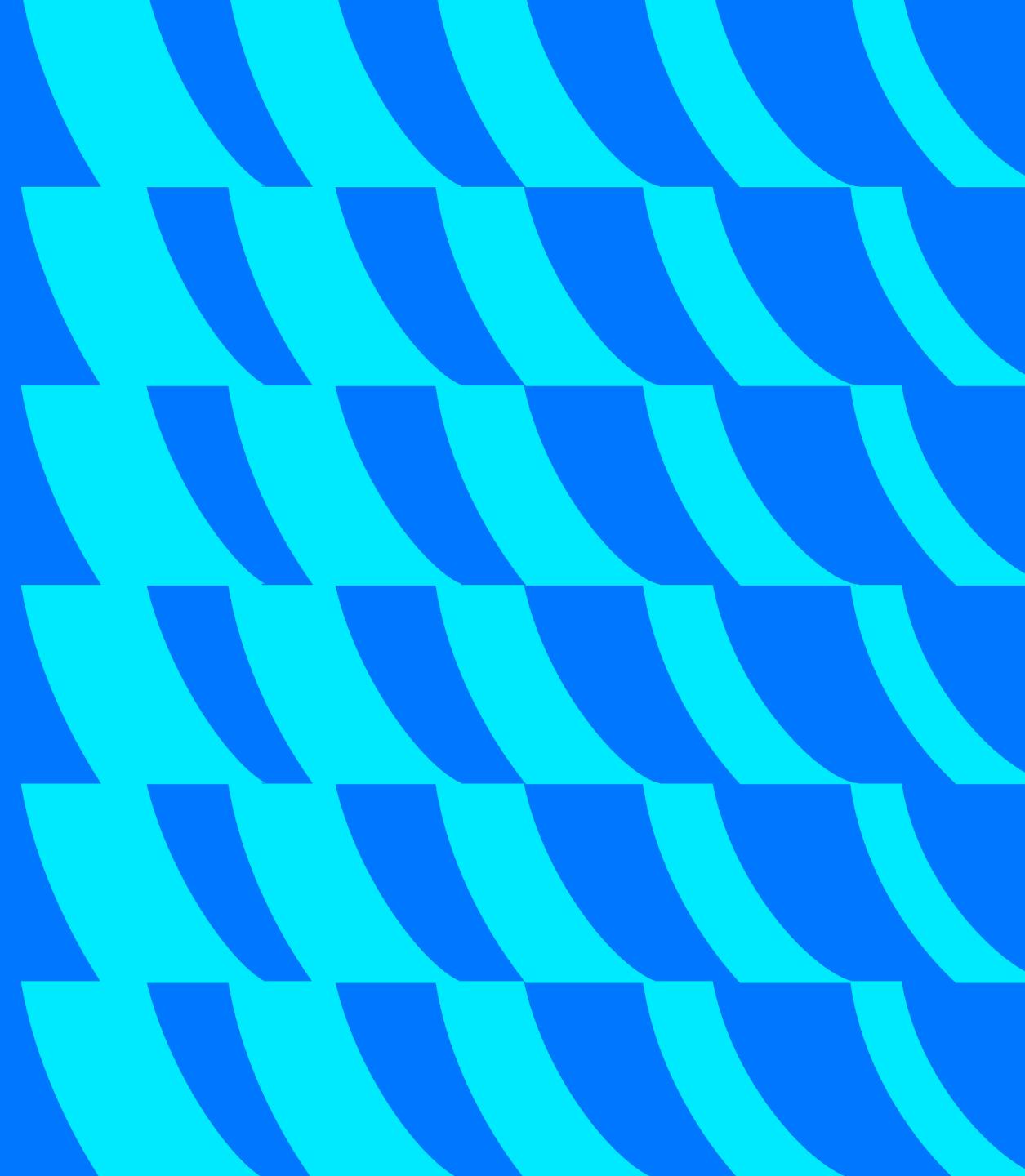


Алгоритмы кластеризации

Лекция 6 - Ноябрь 2024
Владислав Ефимов



Постановка задачи кластеризации

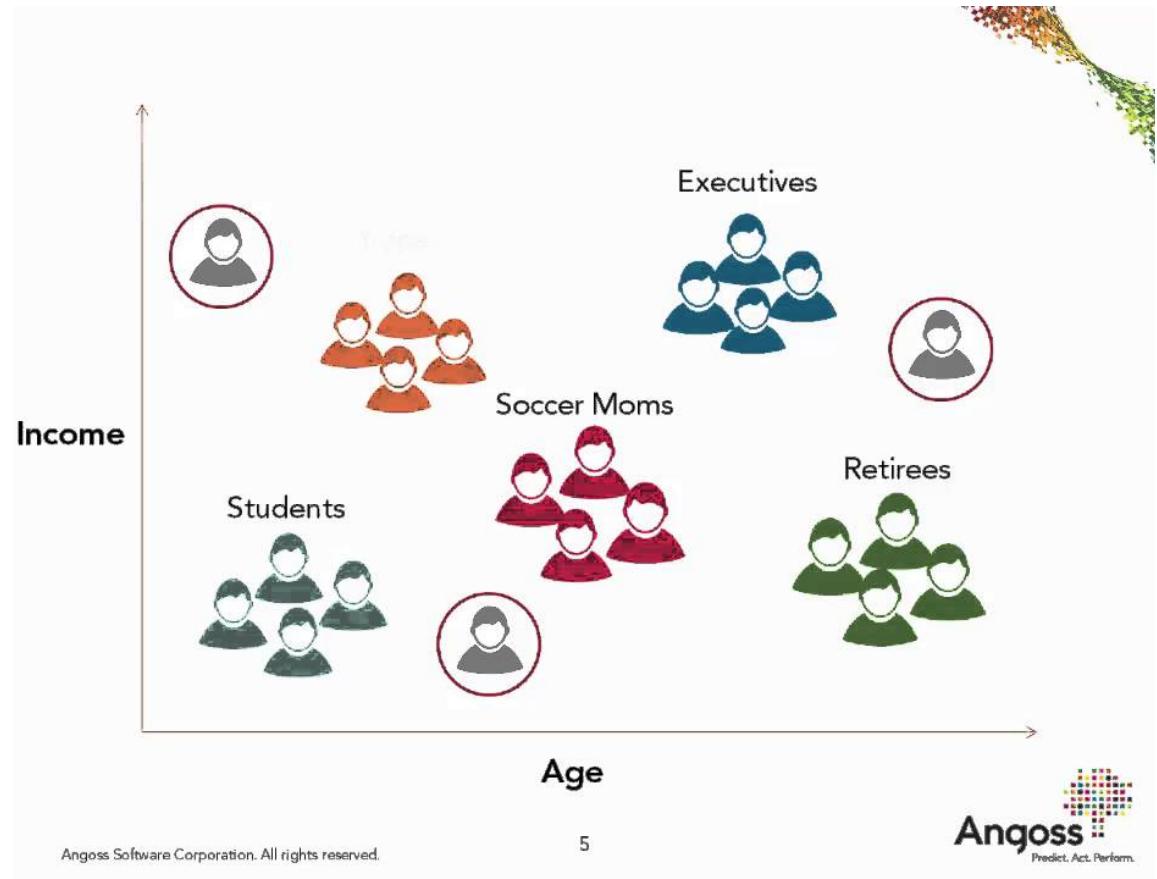


Мы учимся без учителя

When we're learning to see, nobody's telling us what the right answers are — we just look. Every so often, your mother says "that's a dog", but that's very little information. You'd be lucky if you got a few bits of information — even one bit per second — that way. The brain's visual system has 10^{14} neural connections. And you only live for 10^9 seconds. So it's no use learning one bit per second. You need more like 10^5 bits per second. And there's only one place you can get that much information: from the input itself. — Geoffrey Hinton, 1996

Задача кластеризации

Разбиение исходного набора объектов на группы таким образом, чтобы объекты в группе были похожи друг на друга, а объекты из разных групп – отличались. Обучение без учителя.



Цели кластерного анализа



1

Поиск структуры в
данных и её
интерпретация

2

Поиск аномальных
объектов

3

Детальный анализ
отдельных кластеров

4

Формирование
признаков на основе
кластеризации

Группы методов

1

Методы,
основанные на
прототипах

Каждый кластер
ассоциируется с
виртуальным
"эталонным"
объектом

2

Иерархические
методы

Не простое
разбиение, а
целая иерархия

3

Плотностные
методы

Ищем плотные
скопления точек в
признаковом
пространстве

4

Вероятностные
методы

Предполагаем, что
данные порождены
некоторой смесью
вероятностных
распределений

5

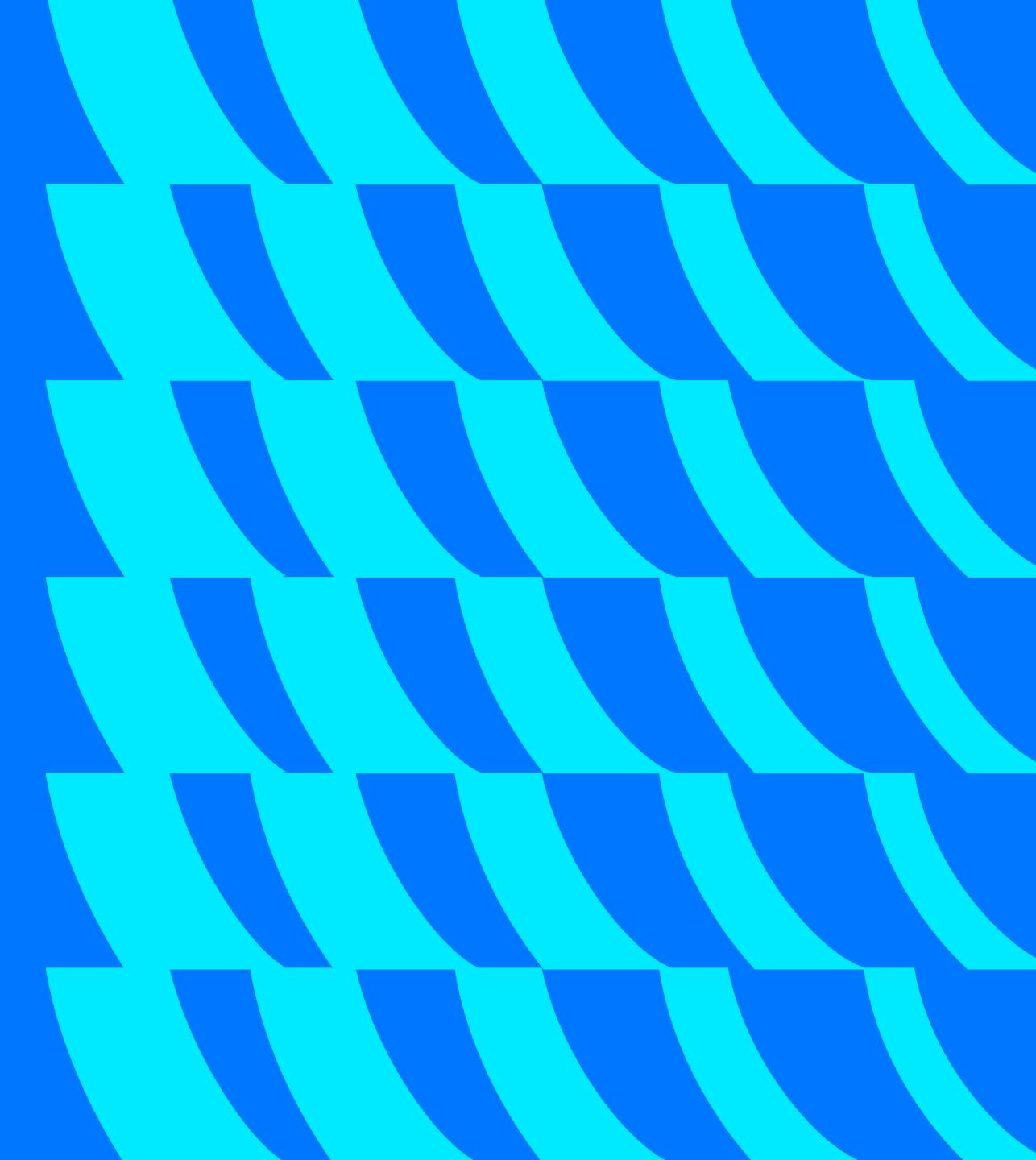
Спектральные
методы

Используем
замечательные
спектральные
свойства разных
матриц

Примеры применения

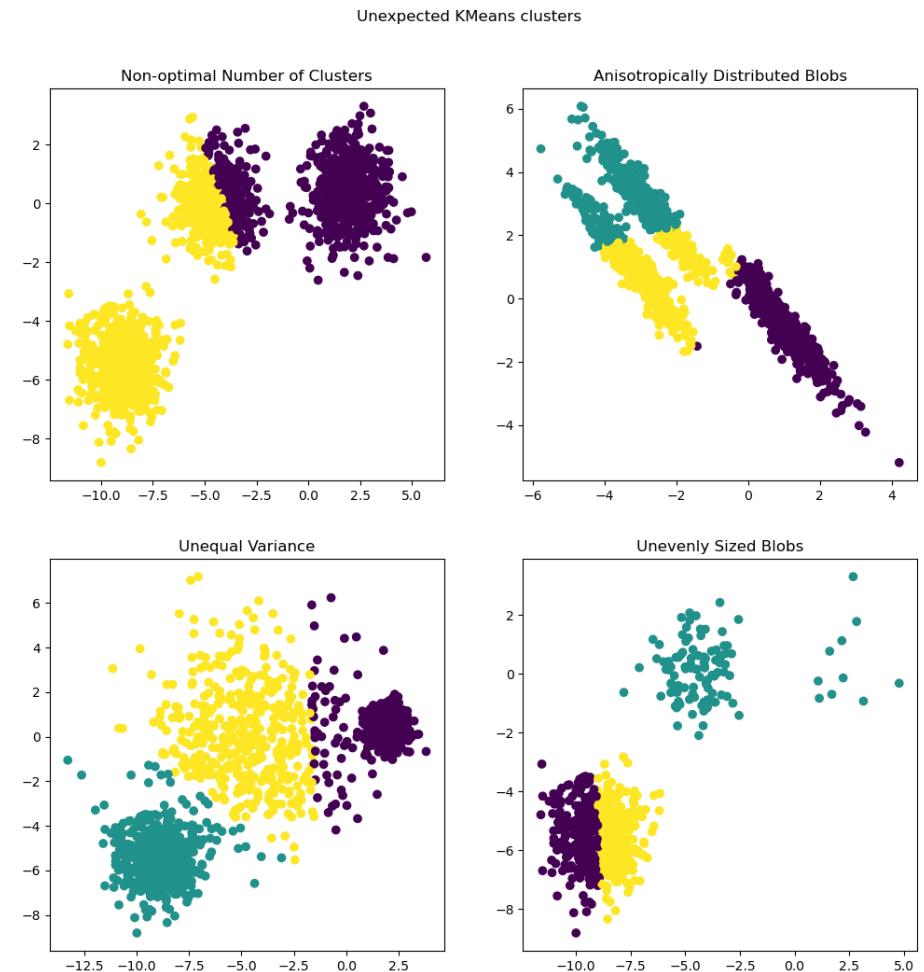
- Сегментация
- Суммаризация
- Сжатие данные
- Обнаружение аномалий
- Тематическое моделирование
- В помощь для классификации/регрессии
- ...

Алгоритм k-means



Постановка

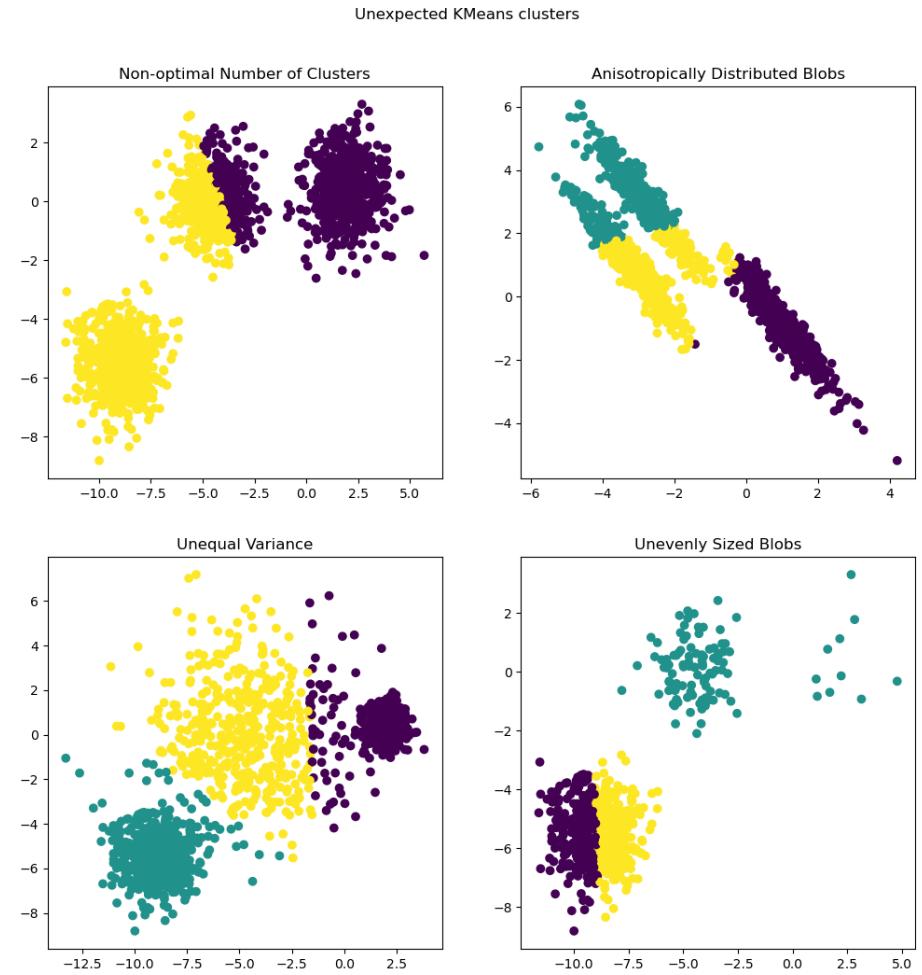
- Дано множество объектов $X = \{x_1, x_2, \dots, x_N\}$



[Реализация sklearn](#)

Постановка

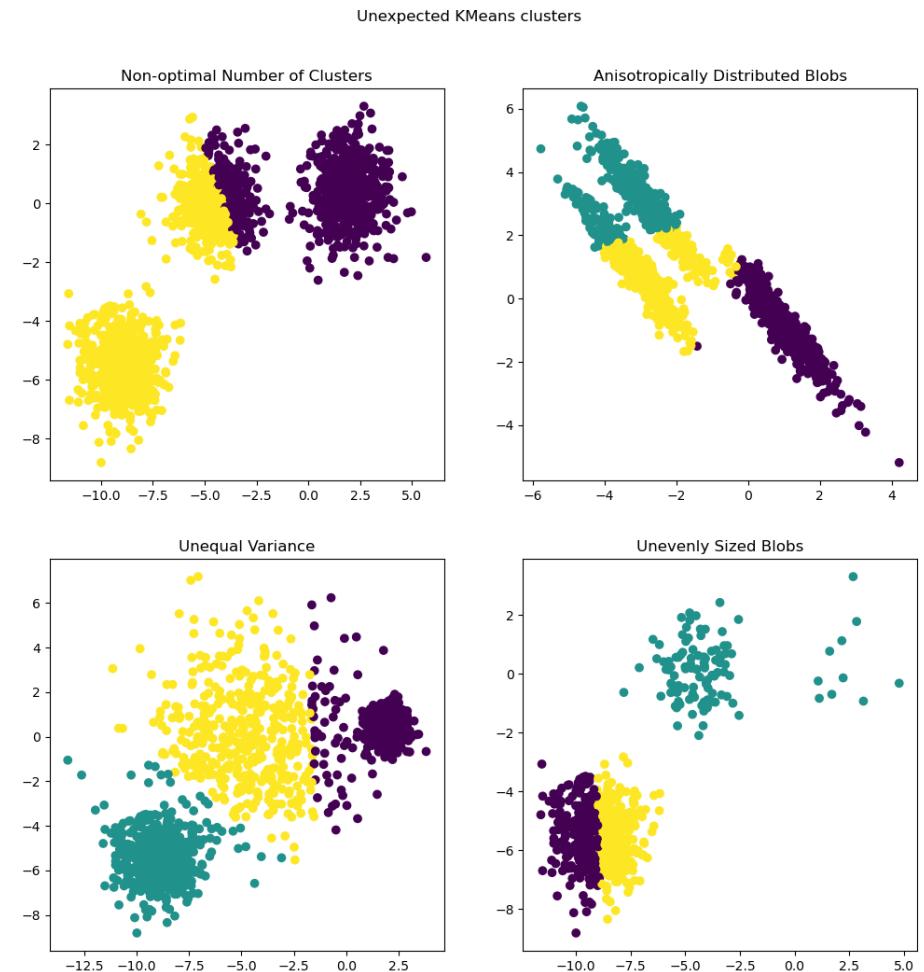
- Дано множество объектов $X = \{x_1, x_2, \dots, x_N\}$
- Кластер $C_k \Leftrightarrow$ центроид μ_k



[Реализация sklearn](#)

Постановка

- Дано множество объектов $X = \{x_1, x_2, \dots, x_N\}$
- Кластер $C_k \Leftrightarrow$ центроид μ_k
- Объект $x_i \in C_k \Leftrightarrow \mu_k = \arg \min_{\mu_j} \|x_i - \mu_j\|^2$



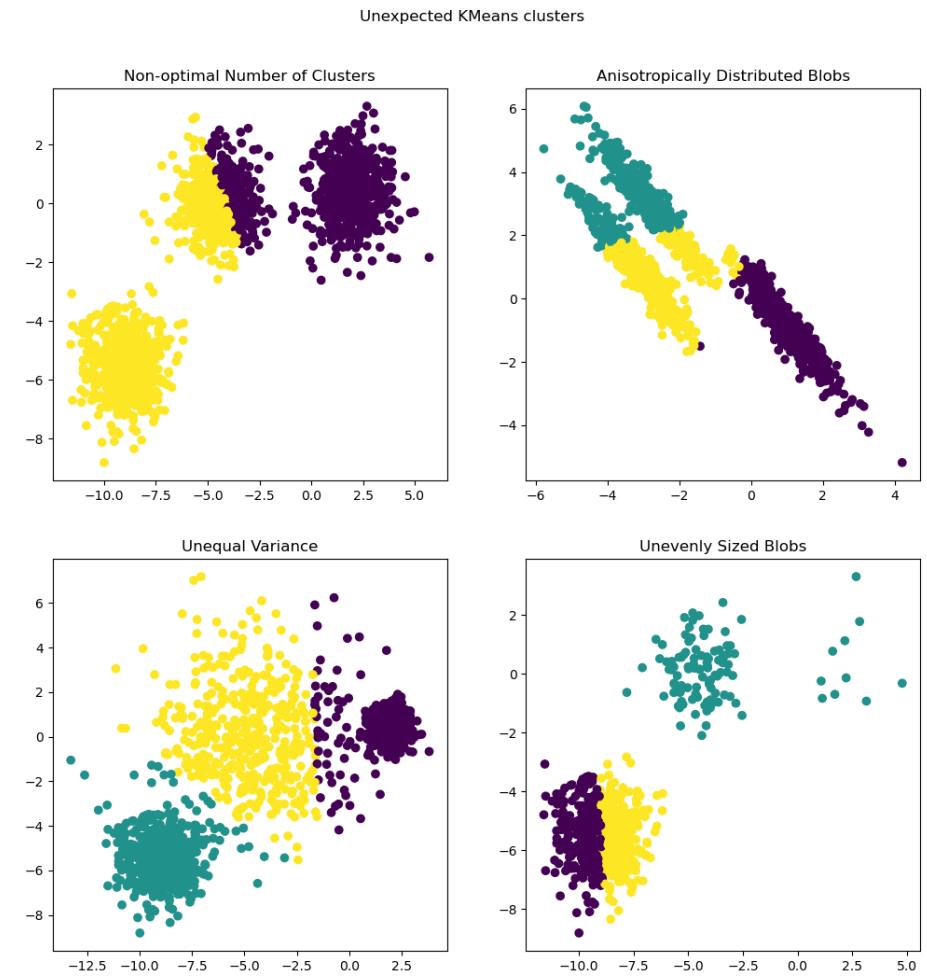
[Реализация sklearn](#)

Постановка

- Дано множество объектов $X = \{x_1, x_2, \dots, x_N\}$
- Кластер $C_k \Leftrightarrow$ центроид μ_k
- Объект $x_i \in C_k \Leftrightarrow \mu_k = \arg \min_{\mu_j} \|x_i - \mu_j\|^2$
- Надо найти такое разбиение на K кластеров, чтобы минимизировать функционал:

$$L(C) = \sum_{k=1}^K \sum_{i \in C_k} \|x_i - \mu_k\|^2 \rightarrow \min_C$$

$$\mu_k = \frac{1}{|C_k|} \sum_{x_n \in C_k} x_n$$



[Реализация sklearn](#)

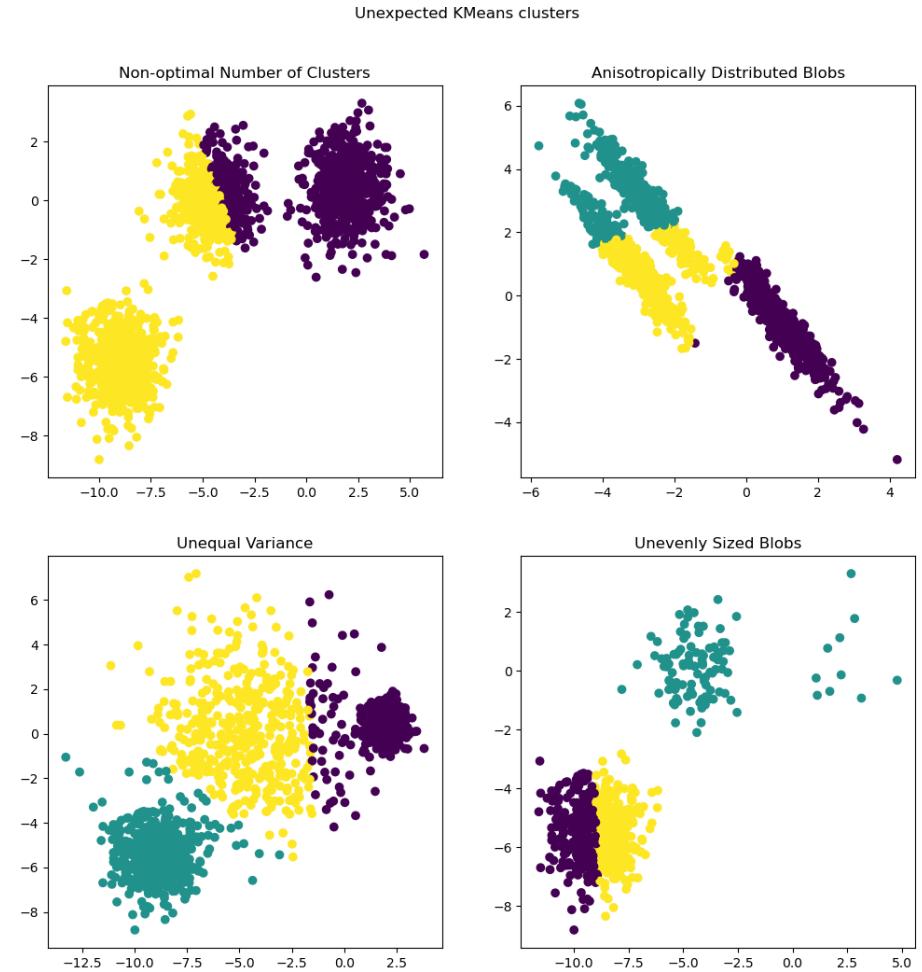
Постановка

- Дано множество объектов $X = \{x_1, x_2, \dots, x_N\}$
- Кластер $C_k \Leftrightarrow$ центроид μ_k
- Объект $x_i \in C_k \Leftrightarrow \mu_k = \arg \min_{\mu_j} \|x_i - \mu_j\|^2$
- Надо найти такое разбиение на K кластеров, чтобы минимизировать функционал:

$$L(C) = \sum_{k=1}^K \sum_{i \in C_k} \|x_i - \mu_k\|^2 \rightarrow \min_C$$

$$\mu_k = \frac{1}{|C_k|} \sum_{x_n \in C_k} x_n$$

- Метод k-средних является итеративным алгоритмом разбиения множества объектов на K кластеров



[Реализация sklearn](#)

Описание алгоритма

- Выбрать K начальных центроидов случайным образом $\mu_k, k = 1 \dots K$

Описание алгоритма

- Выбрать K начальных центроидов случайным образом $\mu_k, k = 1 \dots K$
- Для каждой точки из датасета присвоить кластер, соответствующий ближайшему центроиду:

$$C_k = \{x_n : ||x_n - \mu_k||^2 \leq ||x_n - \mu_l||^2 \quad \forall l \neq k\}$$

Описание алгоритма

- Выбрать K начальных центроидов случайным образом $\mu_k, k = 1 \dots K$
- Для каждой точки из датасета присвоить кластер, соответствующий ближайшему центроиду:

$$C_k = \{x_n : ||x_n - \mu_k||^2 \leq ||x_n - \mu_l||^2 \quad \forall l \neq k\}$$

- Обновить центроиды:

$$\mu_k = \frac{1}{|C_k|} \sum_{x_n \in C_k} x_n$$

Описание алгоритма

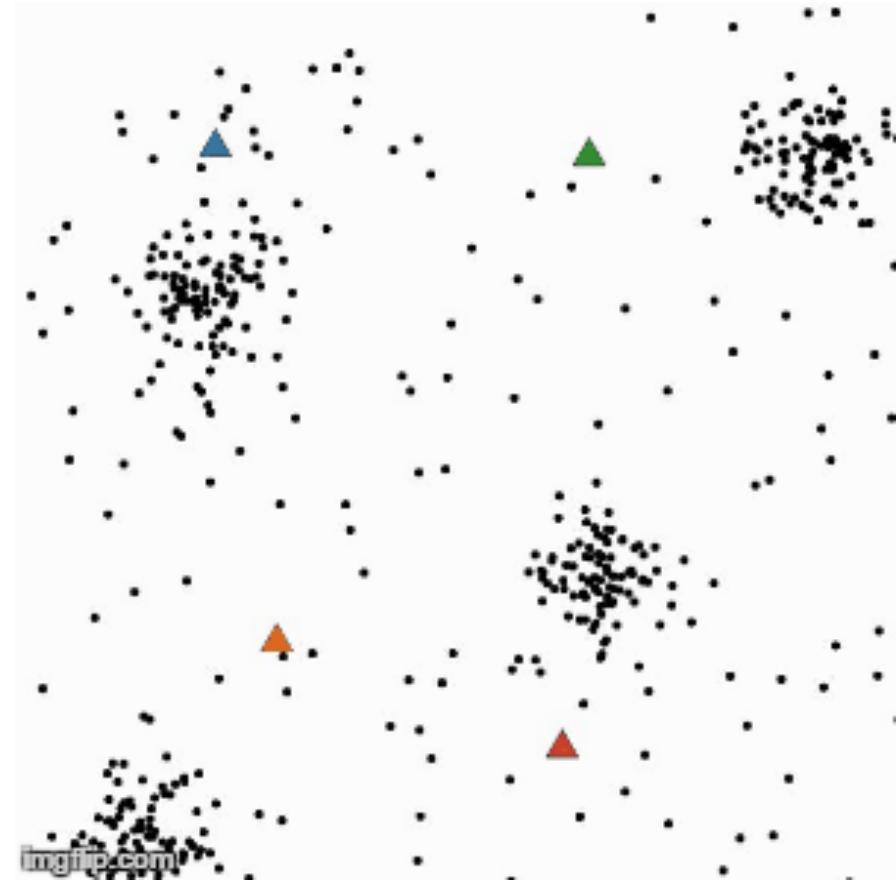
- Выбрать K начальных центроидов случайным образом $\mu_k, k = 1 \dots K$
- Для каждой точки из датасета присвоить кластер, соответствующий ближайшему центроиду:

$$C_k = \{x_n : ||x_n - \mu_k||^2 \leq ||x_n - \mu_l||^2 \quad \forall l \neq k\}$$

- Обновить центроиды:

$$\mu_k = \frac{1}{|C_k|} \sum_{x_n \in C_k} x_n$$

- Повторять шаги 2 и 3 до тех пор, пока изменения перестанут быть существенными



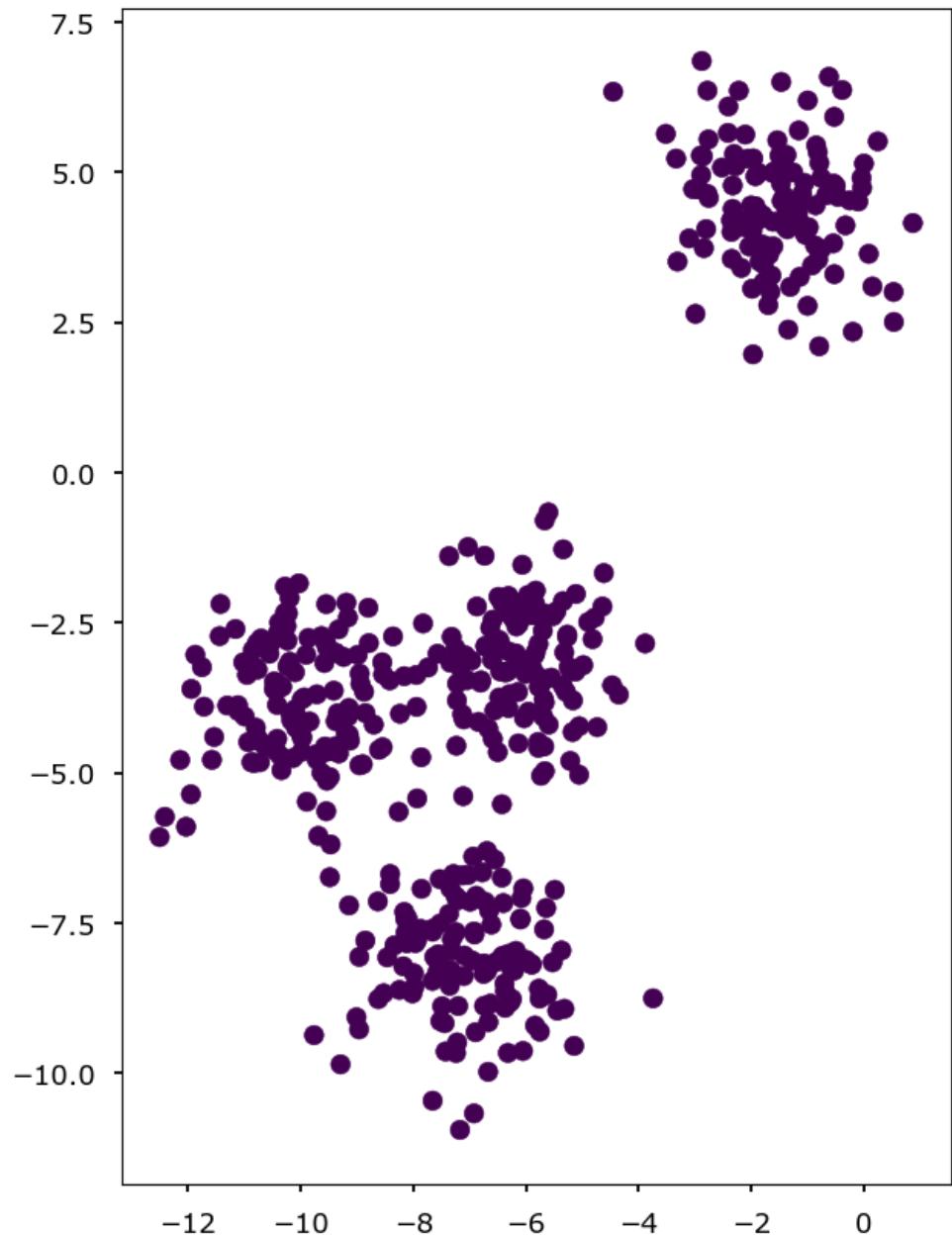
[Демо](#)

Основные факторы

- Начальная инициализация центроидов
- Количество кластеров

Как выбрать K?

- Не пользоваться обычным k-means (X-means, ik-means)
- Посмотреть на меры качества кластеризации
- Воспользоваться эвристиками



Elbow Method («метод локтя»)

- Критерий минимизации k-means:

$$L(C) = \sum_{k=1}^K \sum_{i \in C_k} \|x_i - \mu_k\|^2 \rightarrow \min_C$$

- Давайте возьмём все возможные K , для каждого запустим алгоритм, посчитаем на результате $L(C)$ и выберем минимум!

Elbow Method («метод локтя»)

- Критерий минимизации k-means:

$$L(C) = \sum_{k=1}^K \sum_{i \in C_k} \|x_i - \mu_k\|^2 \rightarrow \min_C$$

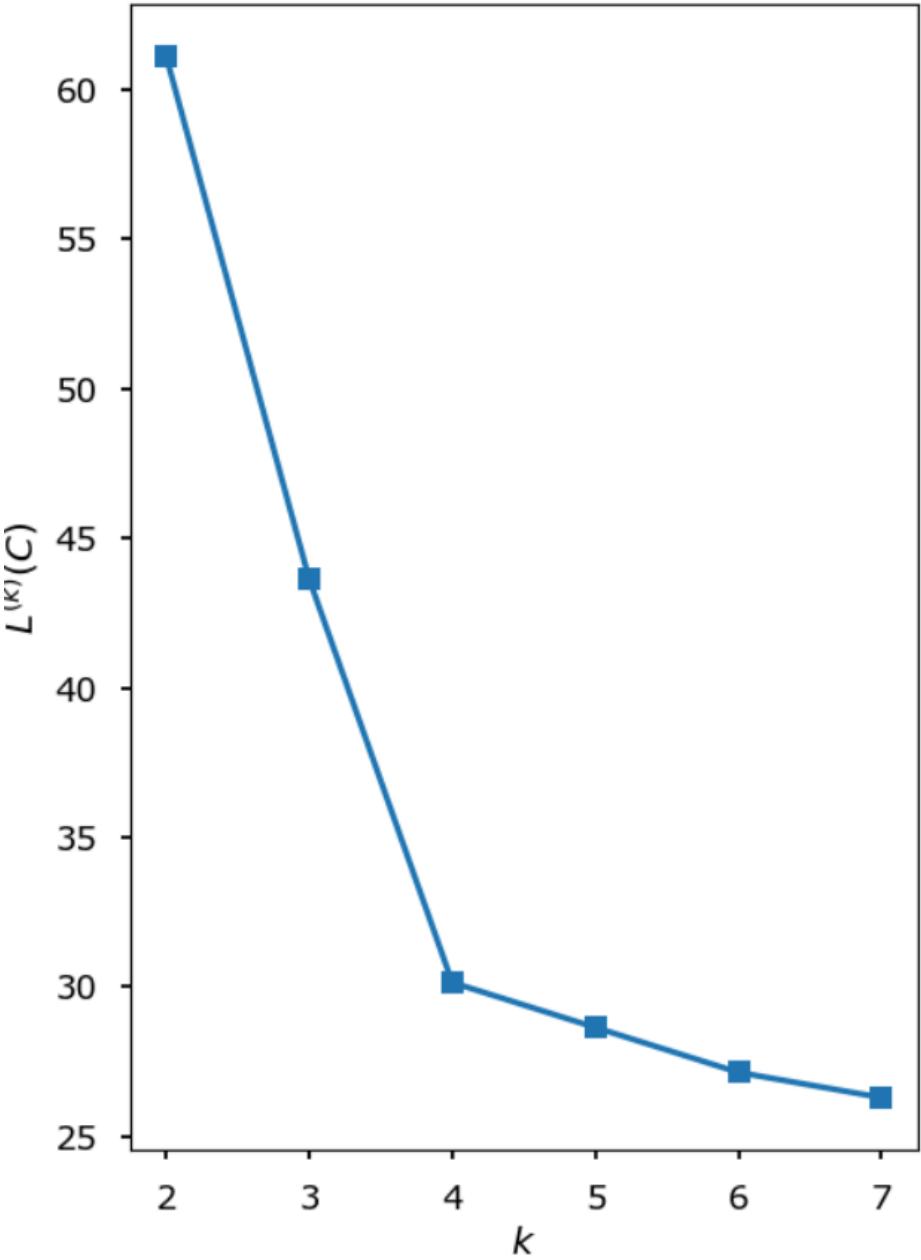
- Давайте возьмём все возможные K , для каждого запустим алгоритм, посчитаем на результате $L(C)$ и выберем минимум!
- Ничего не выйдет... Почему?

Elbow Method («метод локтя»)

- Критерий минимизации k-means:

$$L(C) = \sum_{k=1}^K \sum_{i \in C_k} \|x_i - \mu_k\|^2 \rightarrow \min_C$$

- Выбирают такое k , после которого функционал $L(C)$ уменьшается не слишком быстро



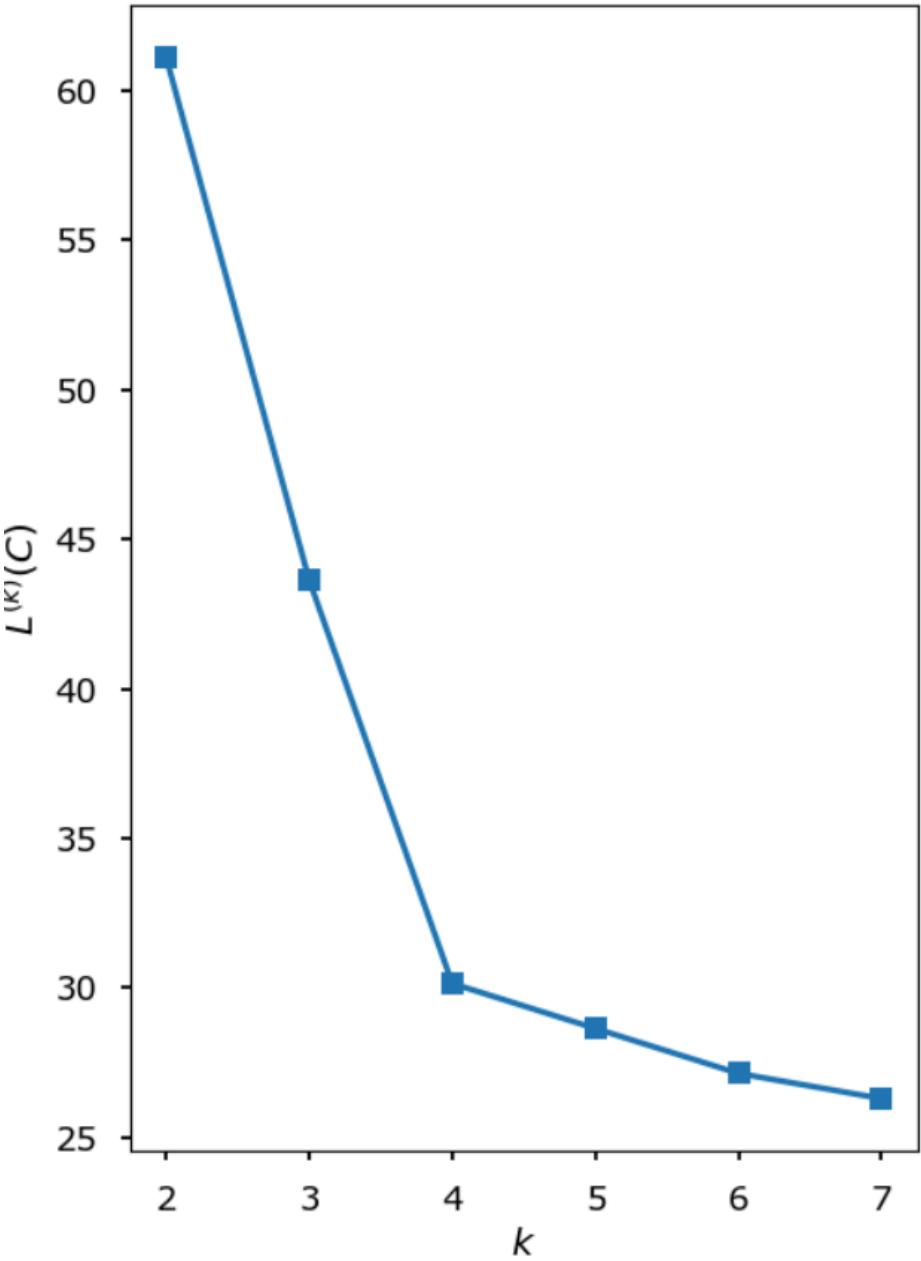
Elbow Method («метод локтя»)

- Критерий минимизации k-means:

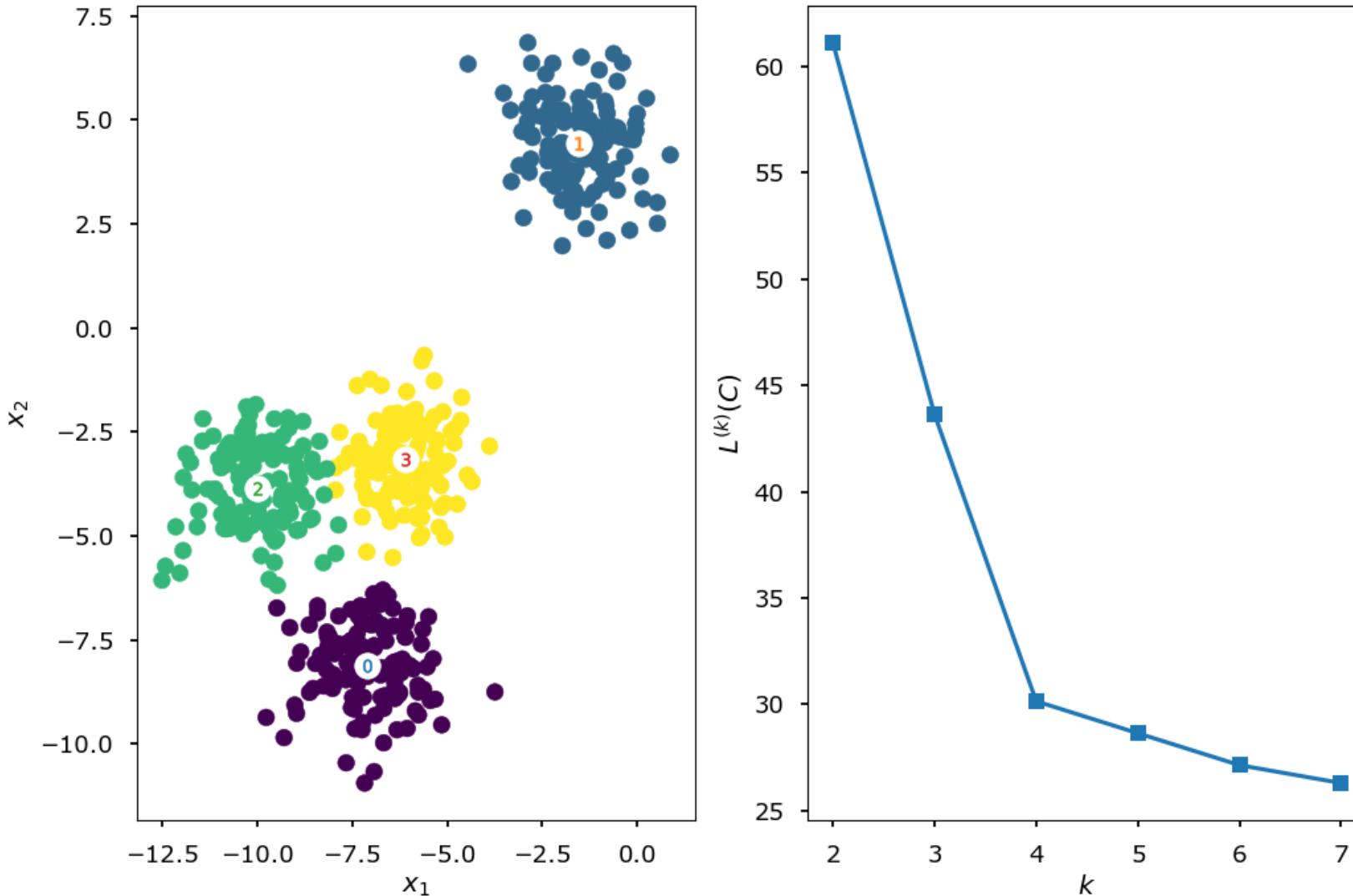
$$L(C) = \sum_{k=1}^K \sum_{i \in C_k} \|x_i - \mu_k\|^2 \rightarrow \min_C$$

- Выбирают такое k , после которого функционал $L(C)$ уменьшается не слишком быстро
- Чуть более формально, значение функционала невелико:

$$D(k) = \frac{|L^{(k)}(C) - L^{(k+1)}(C)|}{|L^{(k-1)}(C) - L^{(k)}(C)|}$$



Elbow Method («метод локтя»)



Важное про эвристики

- Эвристика и меры качества кластеризации носят лишь рекомендательный характер!
- Если они ничего не дают, то лучше ориентироваться на свои знания в предметной области
- Или выжать из полученной кластеризации максимум
- 3 из 5 полученных кластеров интерпретируются — и то хорошо

Начальная инициализация центроидов



1

Выбрать координаты
 K случайных
объектов из
датасета

2

Производить
случайные запуски
много раз и выбрать
наиболее оптимальную
инициализацию

3

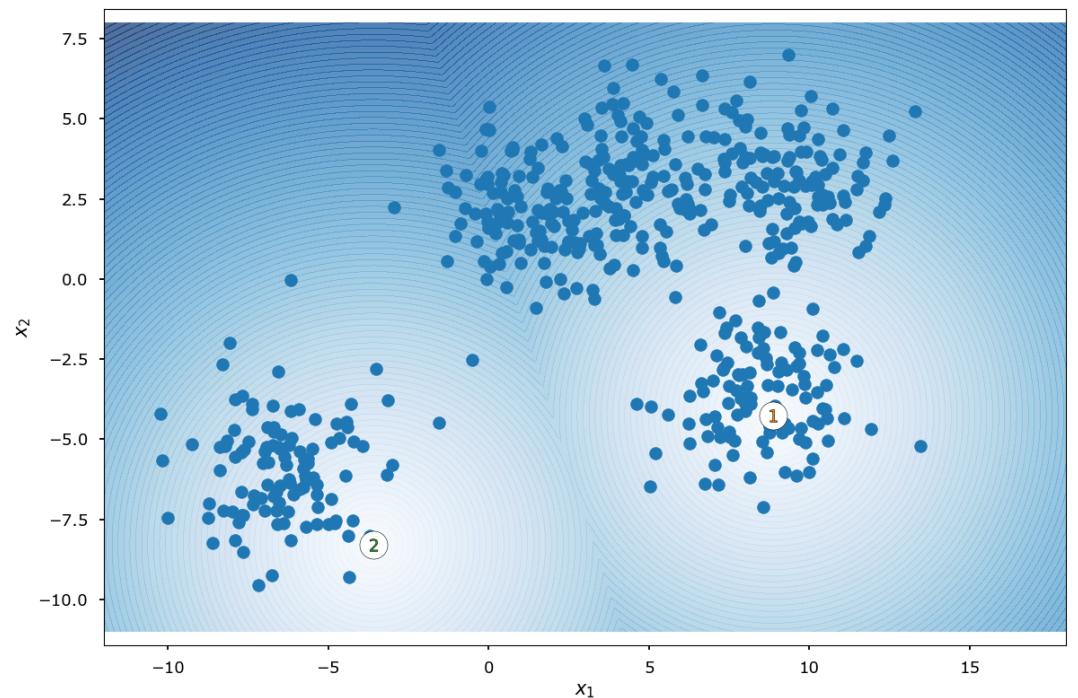
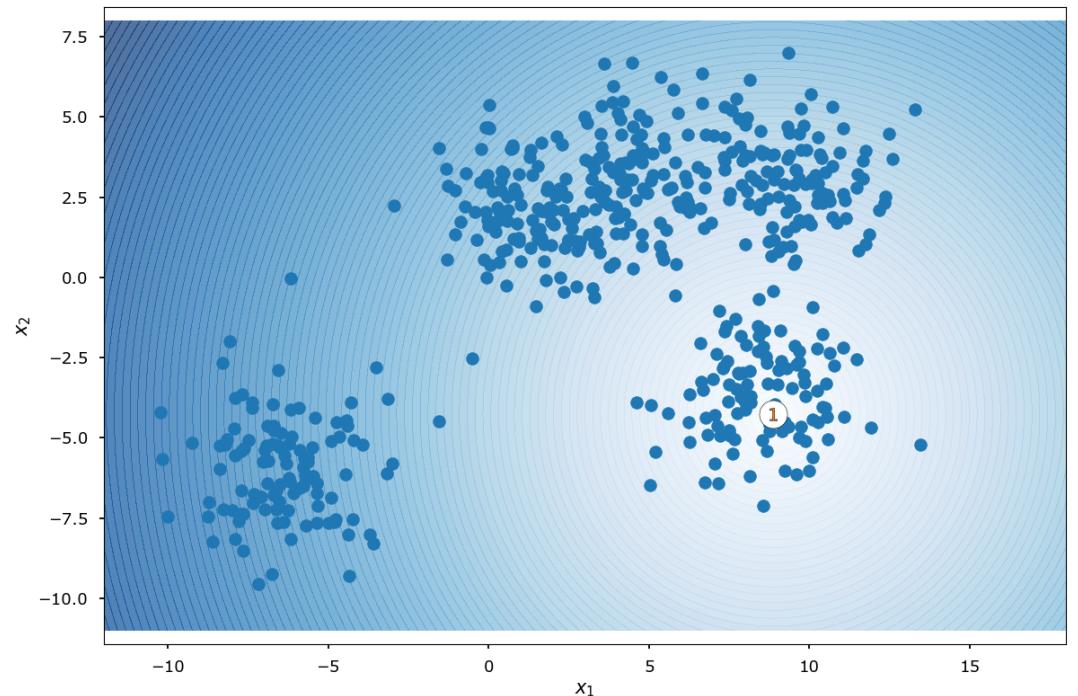
Использовать
результат другой
кластеризации на K
кластеров

4

k-means++

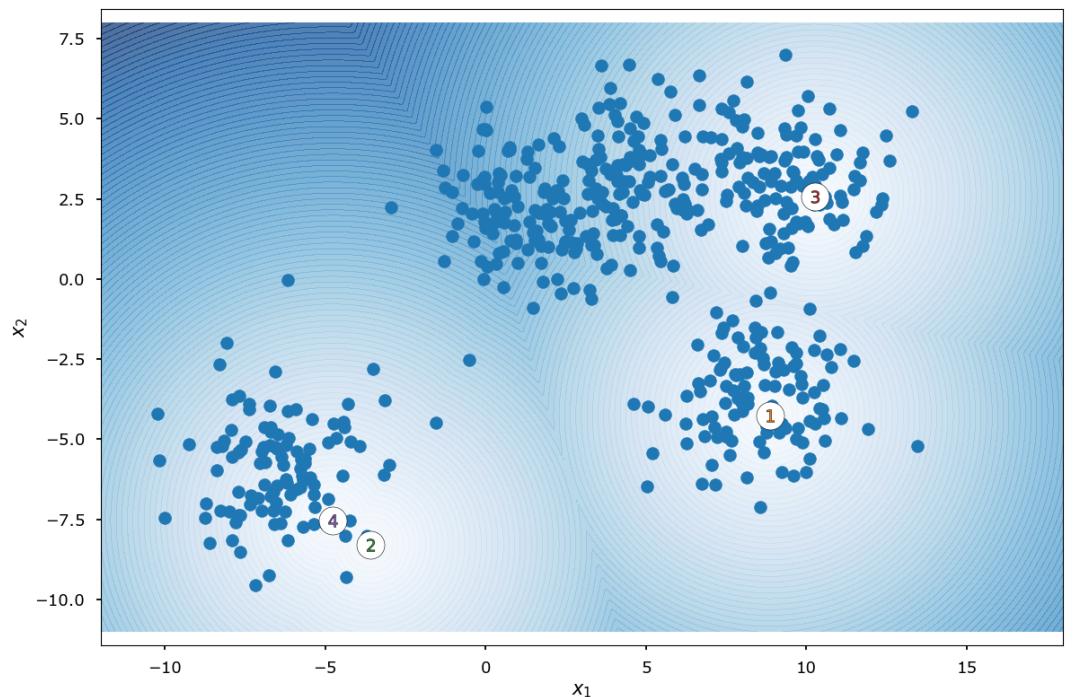
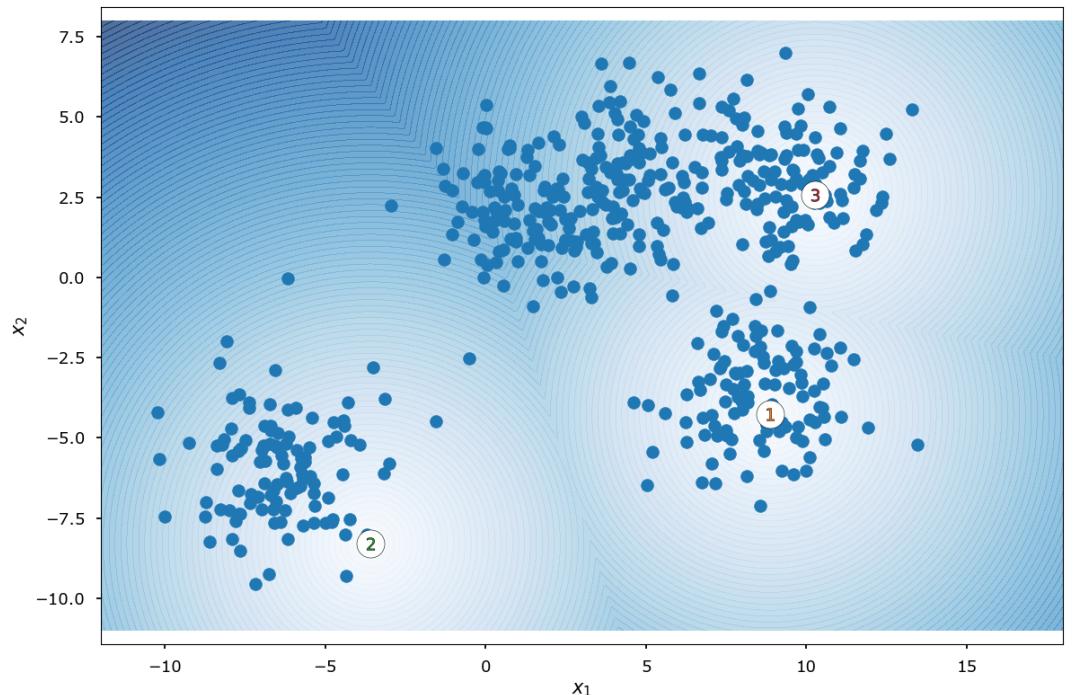
k-means++

- Первый центроид выбираем случайным образом из объектов датасета
- Для каждой точки рассчитываем расстояние $d_{\min}(x_i) = \min_{\mu_j} \|x_i - \mu_j\|^2$
- Точка назначается следующим центроидом с вероятностью $p(x_i) \propto d_{\min}(x_i)$



k-means++

- Первый центроид выбираем случайным образом из объектов датасета
- Для каждой точки рассчитываем расстояние $d_{\min}(x_i) = \min_{\mu_j} \|x_i - \mu_j\|^2$
- Точка назначается следующим центроидом с вероятностью $p(x_i) \propto d_{\min}(x_i)$



k-medoids

А если у нас есть только расстояния между объектами?

- Инициализируем случайно медианные объекты
- Для каждого объекта находим ближайший к нему
- Найти медианные объекты
- Повторять шаги 2 и 3 до сходимости

Резюме

- Метод k-средних — жадный итеративный алгоритм.
- Зависит от начальных центроидов и их количества.

Преимущества



1

Прост

2

Имеет множество
модификаций

3

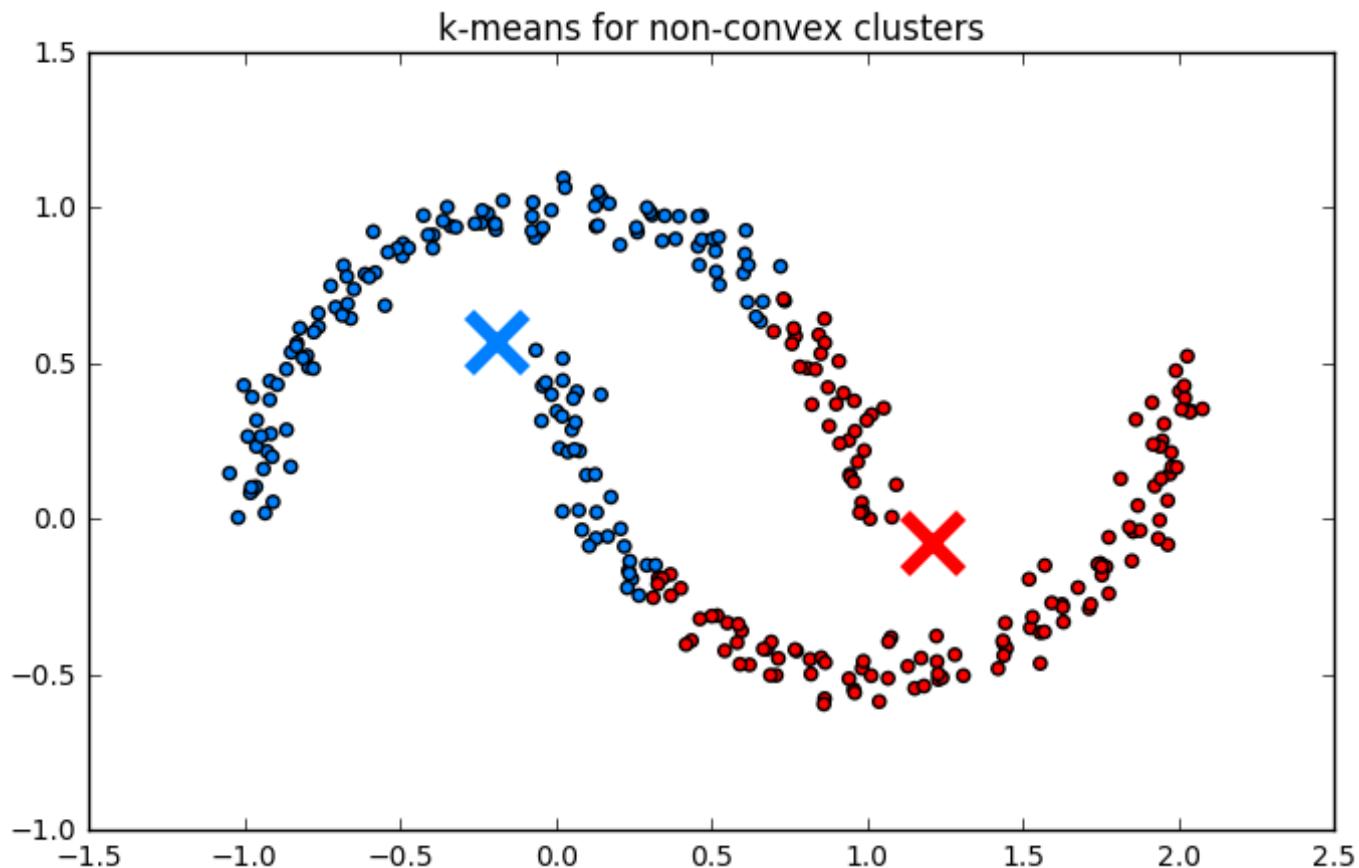
Интерпретация
кластеров через
центроиды

4

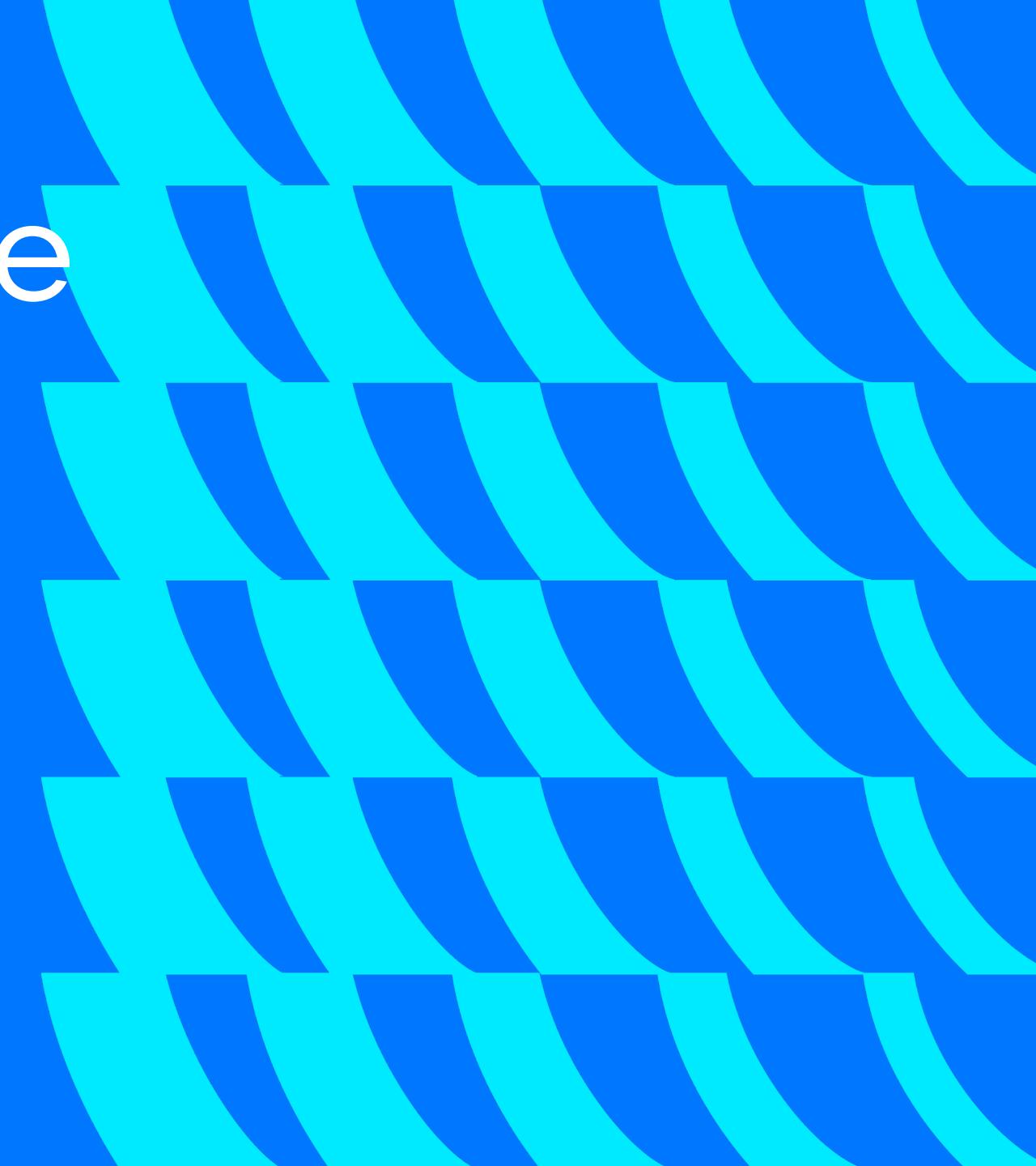
Почти всегда на
выходе будет k
кластеров

Недостатки

- Подразумевает выпуклые кластеры
- Нужно знать или хорошо уметь подбирать параметр k — число кластеров

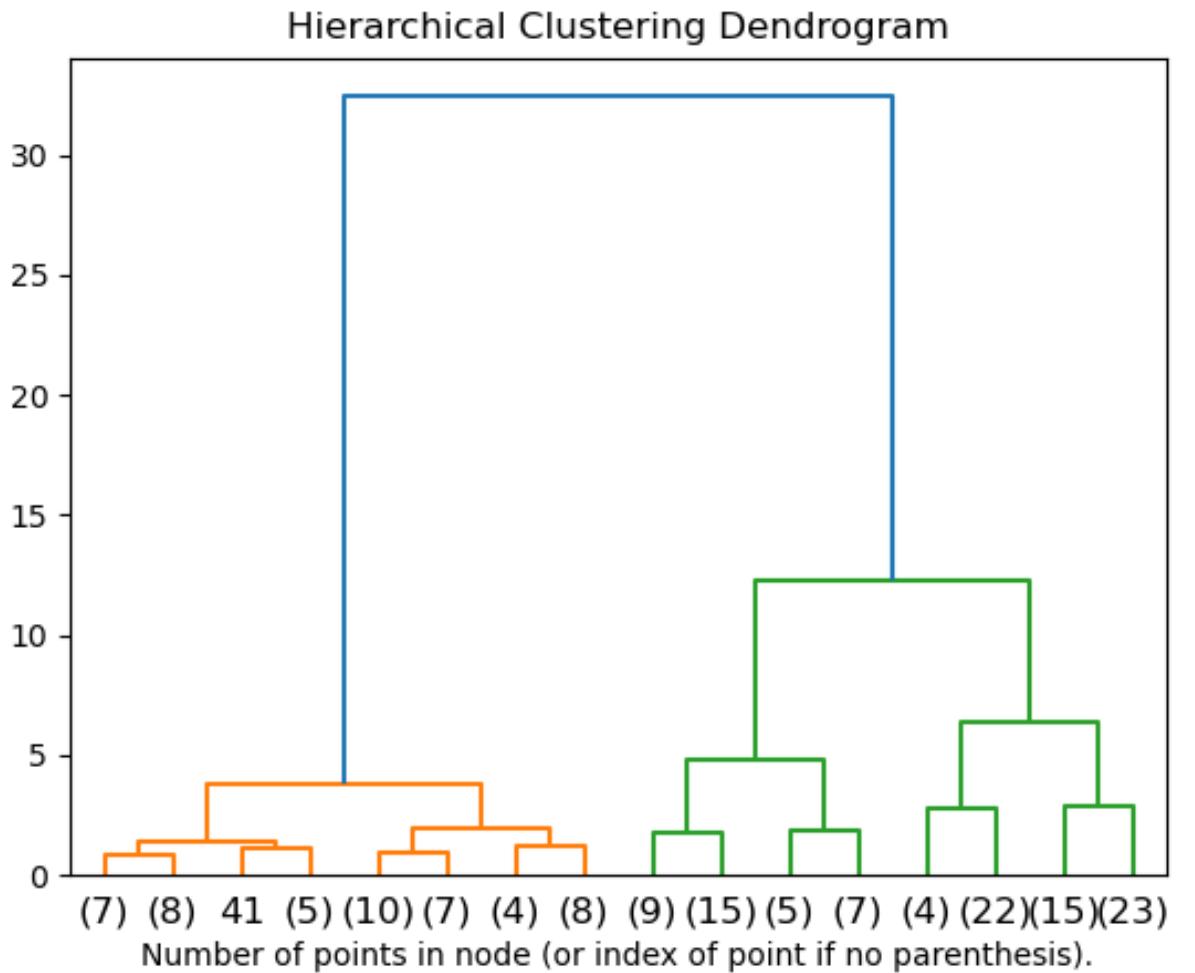


Иерархические алгоритмы: Agglomerative Clustering



Основная идея

- Идём снизу вверх, вначале все точки в своём кластере
- На каждом шаге объединяем два ближайших кластера
- Продолжаем, пока не дойдём до корня



[Реализация sklearn](#)

Что значит ближайшие кластеры, как мерить расстояние?

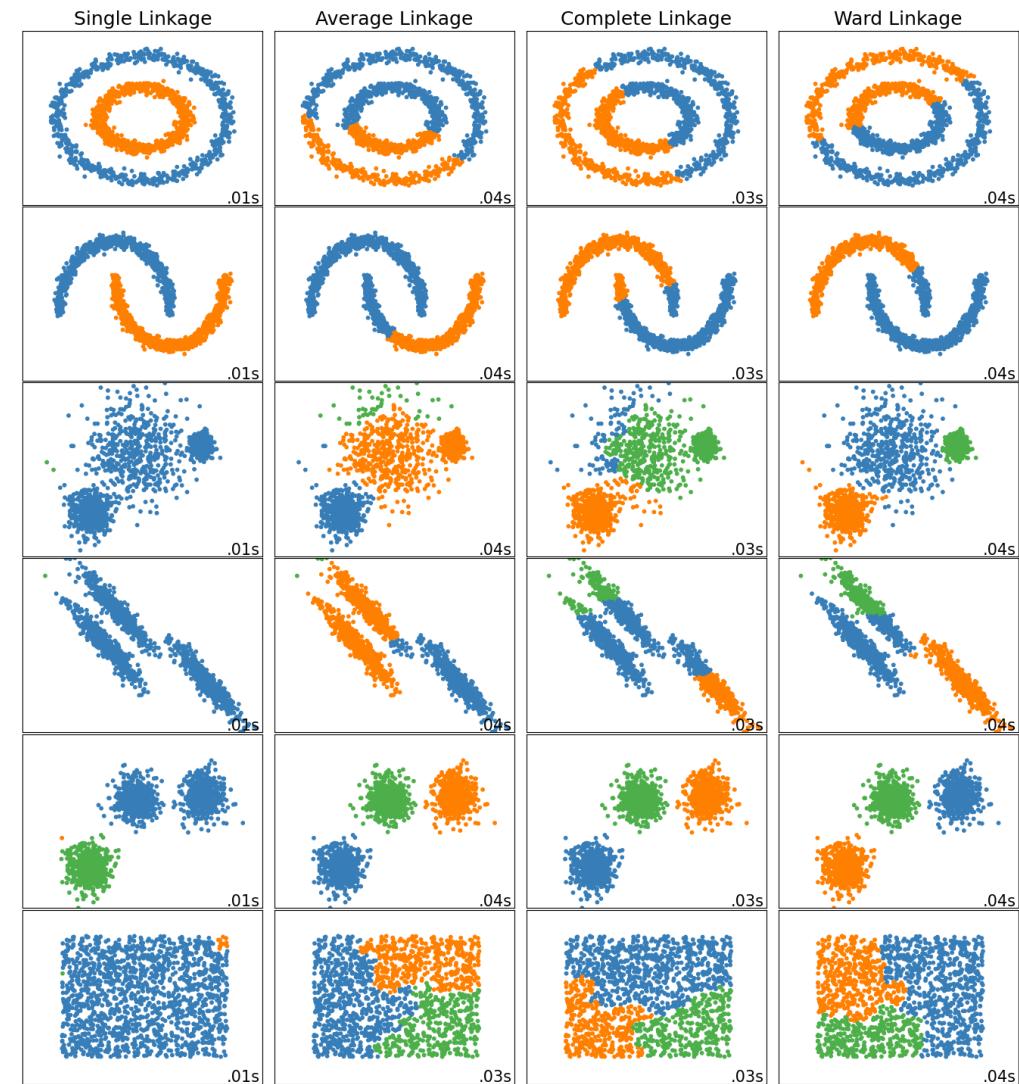
Есть несколько стратегий, которые можно использовать для объединения кластеров:

- Single linkage — минимизирует расстояние между ближайшими точками разных кластеров:

$$\min_{c_1 \neq c_2 \in C} \min_{x \in c_1, y \in c_2} \rho(x, y)$$

- Average linkage — минимизирует среднее расстояние между точками разных кластеров:

$$\min_{c_1 \neq c_2 \in C} \frac{1}{|c_1|} \sum_{x \in c_1} \frac{1}{|c_2|} \sum_{y \in c_2} \rho(x, y)$$



Что значит ближайшие кластеры, как мерить расстояние?

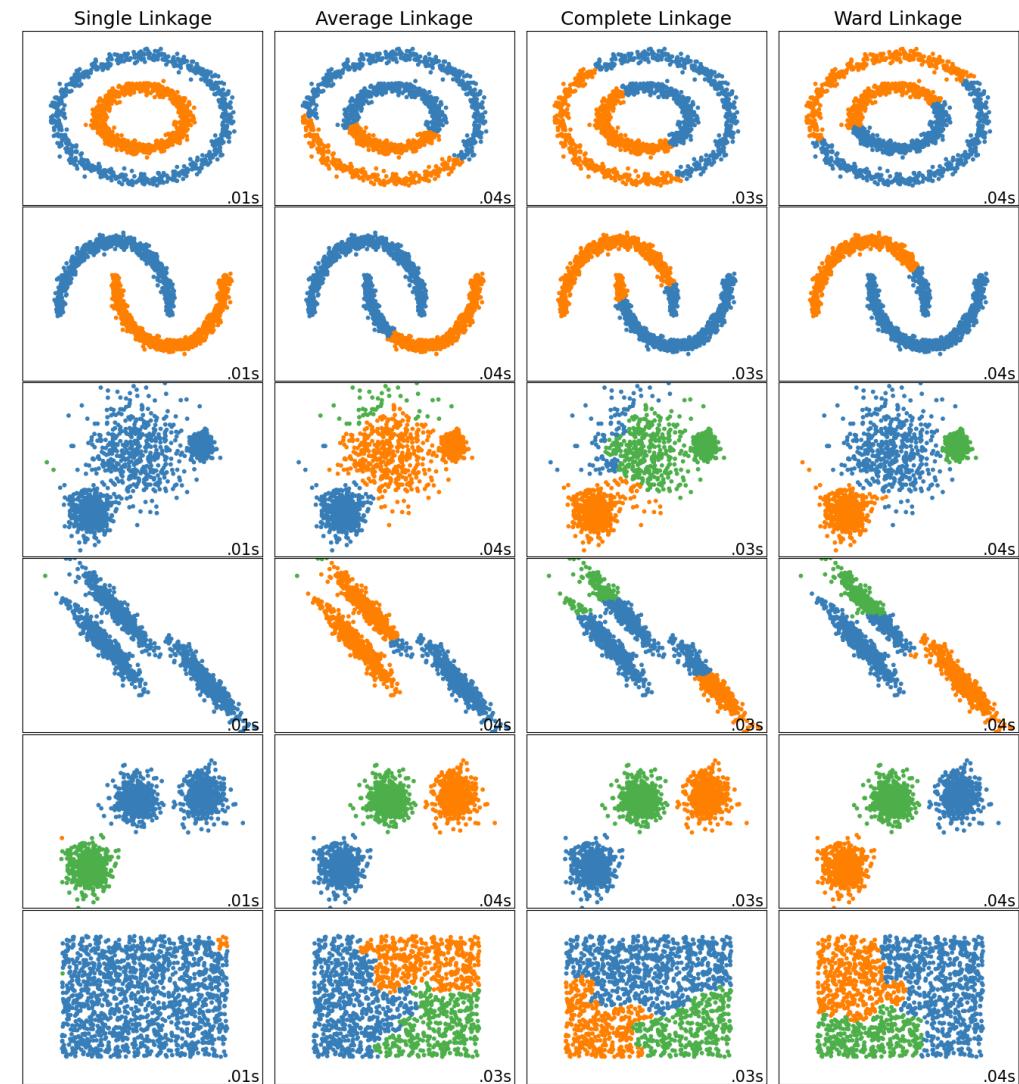
Есть несколько стратегий, которые можно использовать для объединения кластеров:

- Complete (maximum) linkage — минимизирует максимальное расстояние между точками разных кластеров:

$$\min_{c_1 \neq c_2 \in C} \max_{x \in c_1, y \in c_2} \rho(x, y)$$

- Ward — минимизирует расстояния между точками в одном кластере:

$$\sum_{x,y \in c} ||x - y||^2 \rightarrow \min$$



Преимущества



1

Иерархия кластеров

4

Анализ данных с
помощью
дендрограммы

2

Нужно только задать
расстояния

5

Понятно, как найти
выбросы

3

Выдает любое число
кластеров

Недостатки

1

2

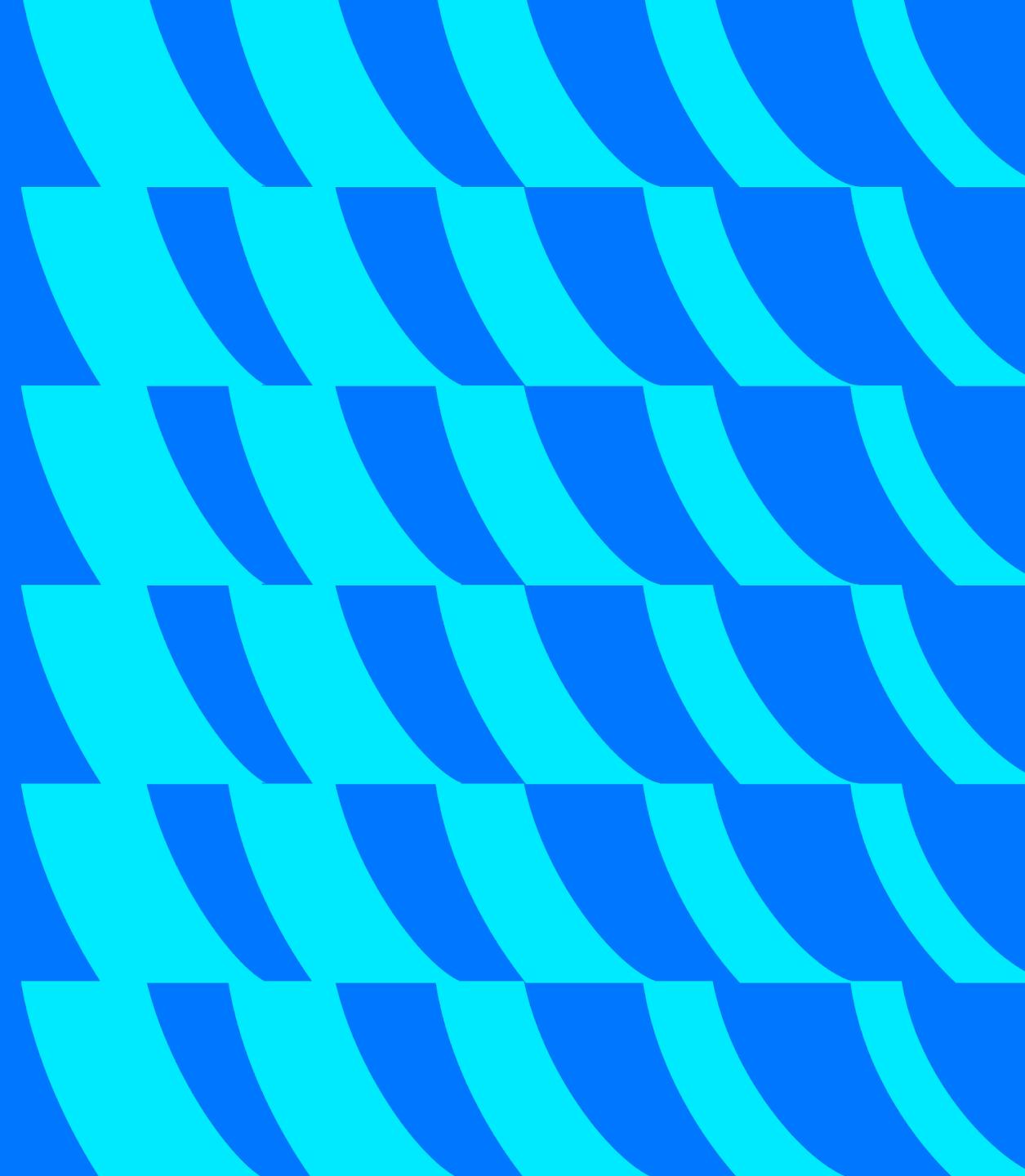
Очень долго

Нельзя дообучать

3

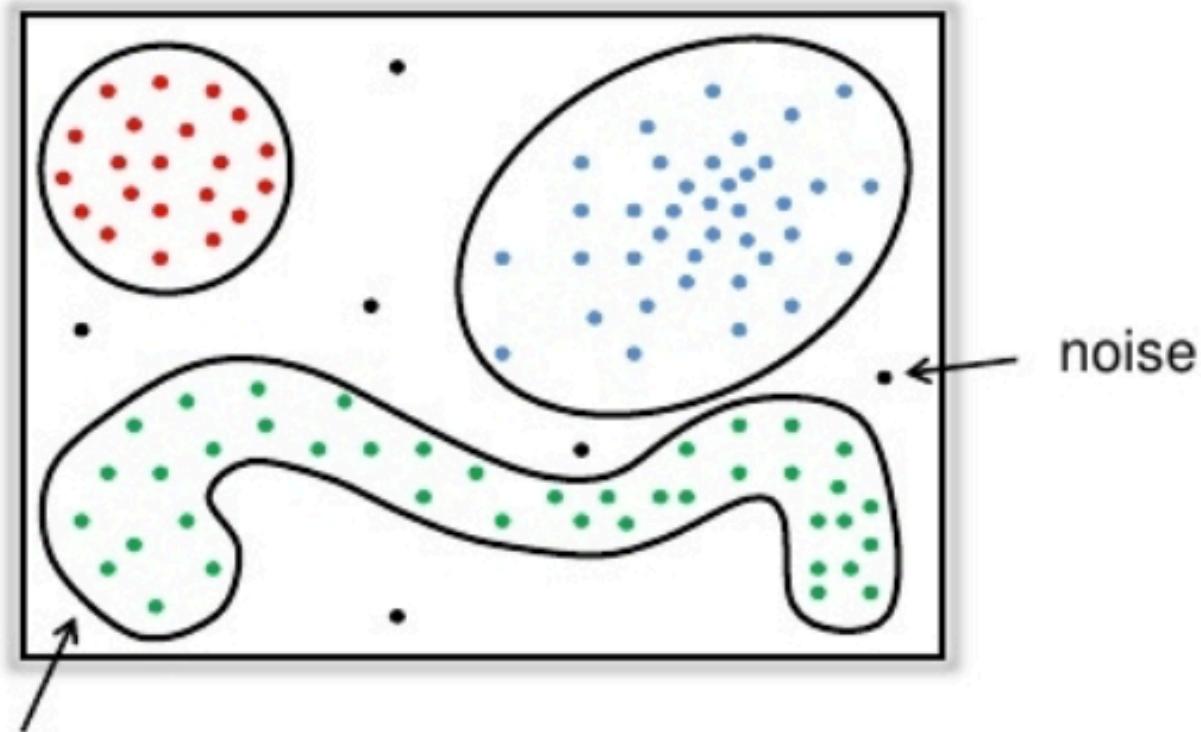
Не всегда удобно
задавать расстояния

Алгоритмы,
основанные на
плотности:
DBSCAN



Хотелось бы...

Получить кластеры высокой плотности, разделённые участками низкой плотности



arbitrarily shaped clusters

Основная идея

- Для каждой точки кластера её окрестность заданного радиуса ϵ должна содержать не менее некоторого числа точек `min_pts`.

Основная идея

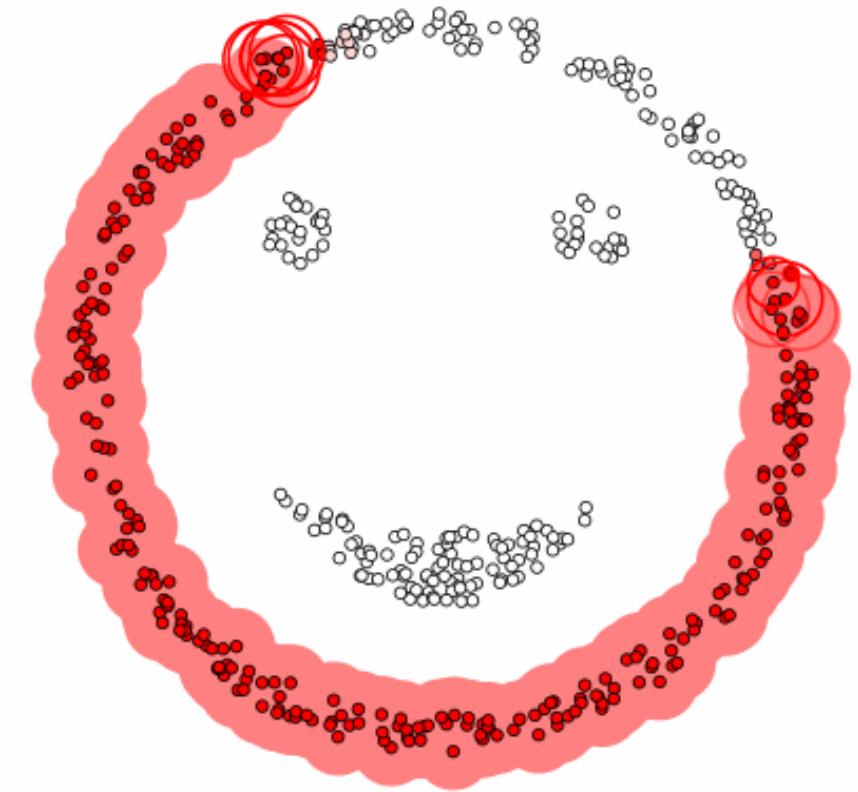
- Для каждой точки кластера её окрестность заданного радиуса ϵ должна содержать не менее некоторого числа точек `min_pts`.
- С такой точки можно начать расширение «плотного» кластера:
 - то есть каждая точка в ϵ окрестности будет добавляться в кластер
 - её соседи тоже будут проверяться на критерий `min_pts`

Основная идея

- Для каждой точки кластера её окрестность заданного радиуса ϵ должна содержать не менее некоторого числа точек min_pts .
- С такой точки можно начать расширение «плотного» кластера:
 - то есть каждая точка в ϵ окрестности будет добавляться в кластер
 - её соседи тоже будут проверяться на критерий min_pts
- Расширение текущего кластера закончится, когда объекты перестанут удовлетворять условию min_pts

Основная идея

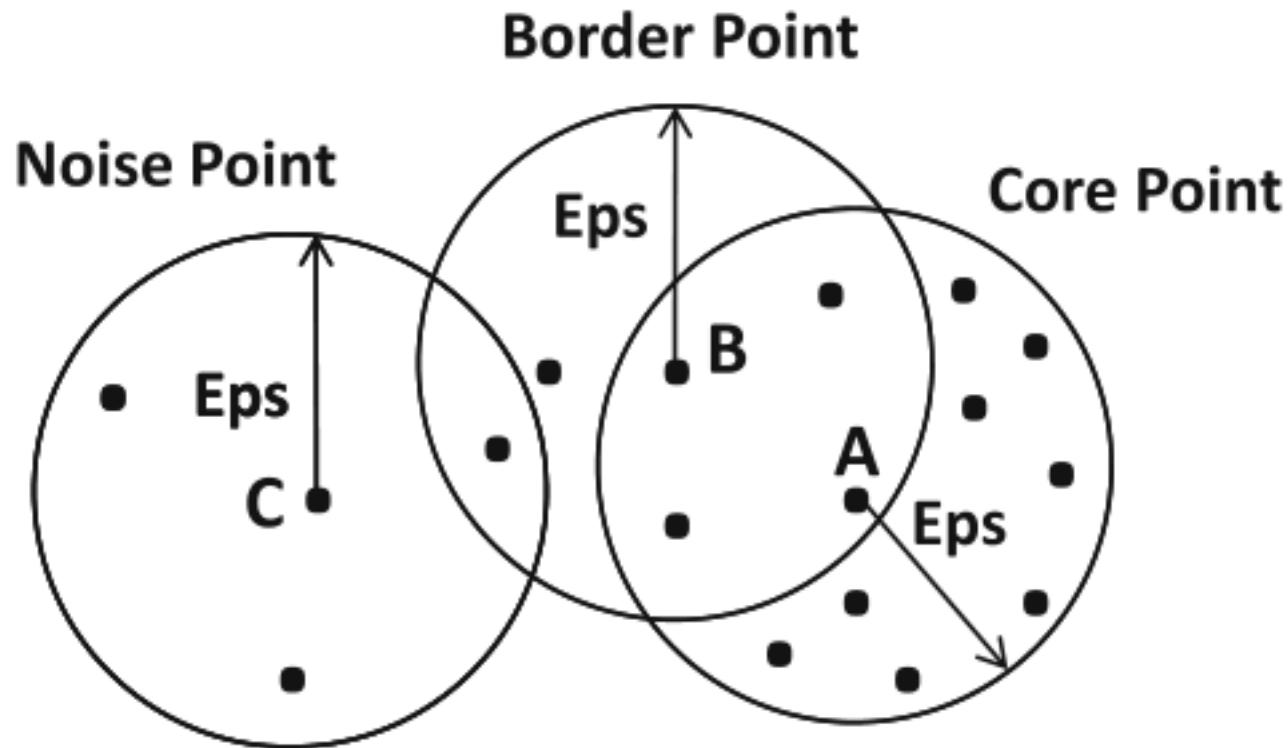
- Для каждой точки кластера её окрестность заданного радиуса ϵ должна содержать не менее некоторого числа точек `min_pts`.
- С такой точки можно начать расширение «плотного» кластера:
 - то есть каждая точка в ϵ окрестности будет добавляться в кластер
 - её соседи тоже будут проверяться на критерий `min_pts`
- Расширение текущего кластера закончится, когда объекты перестанут удовлетворять условию `min_pts`



[Демо](#)

Типы точек

- Core point: точки, в ε -окрестности которых $\geq \text{min_pts}$ точек
- Border point: не core, но содержит хотя бы 1 core-точку в ε -окрестности
- Noise point: всё остальное



DBSCAN

ПСЕВДОКОД

```
1.function dbscan(X, eps, min_pts):  
2.    initialize NV = X # not visited objects  
3.    for x in NV:  
4.        remove(NV, x) # mark as visited  
5.        nbr = neighbours(x, eps) # set of neighbours  
6.        if nbr.size < min_pts:  
7.            mark_as_noise(x)  
8.        else:  
9.            C = new_cluster()  
10.           expand_cluster(x, nbr, C, eps, min_pts, NV)  
11.           yield C
```

```
1.function expand_cluster(x, nbr, C, eps, min_pts, NV):  
2.    add(x, C)  
3.    for x1 in nbr:  
4.        if x1 in NV: # object not visited  
5.            remove(NV, x1) # mark as visited  
6.            nbr1 = neighbours(x1, eps)  
7.            if nbr1.size >= min_pts:  
8.                # join sets of neighbours  
9.                merge(nbr, nbr_1)  
10.               if x1 not in any cluster:  
11.                   add(x1, C)
```

Преимущества



1

Не требует k

2

Кластеры
произвольной
формы

3

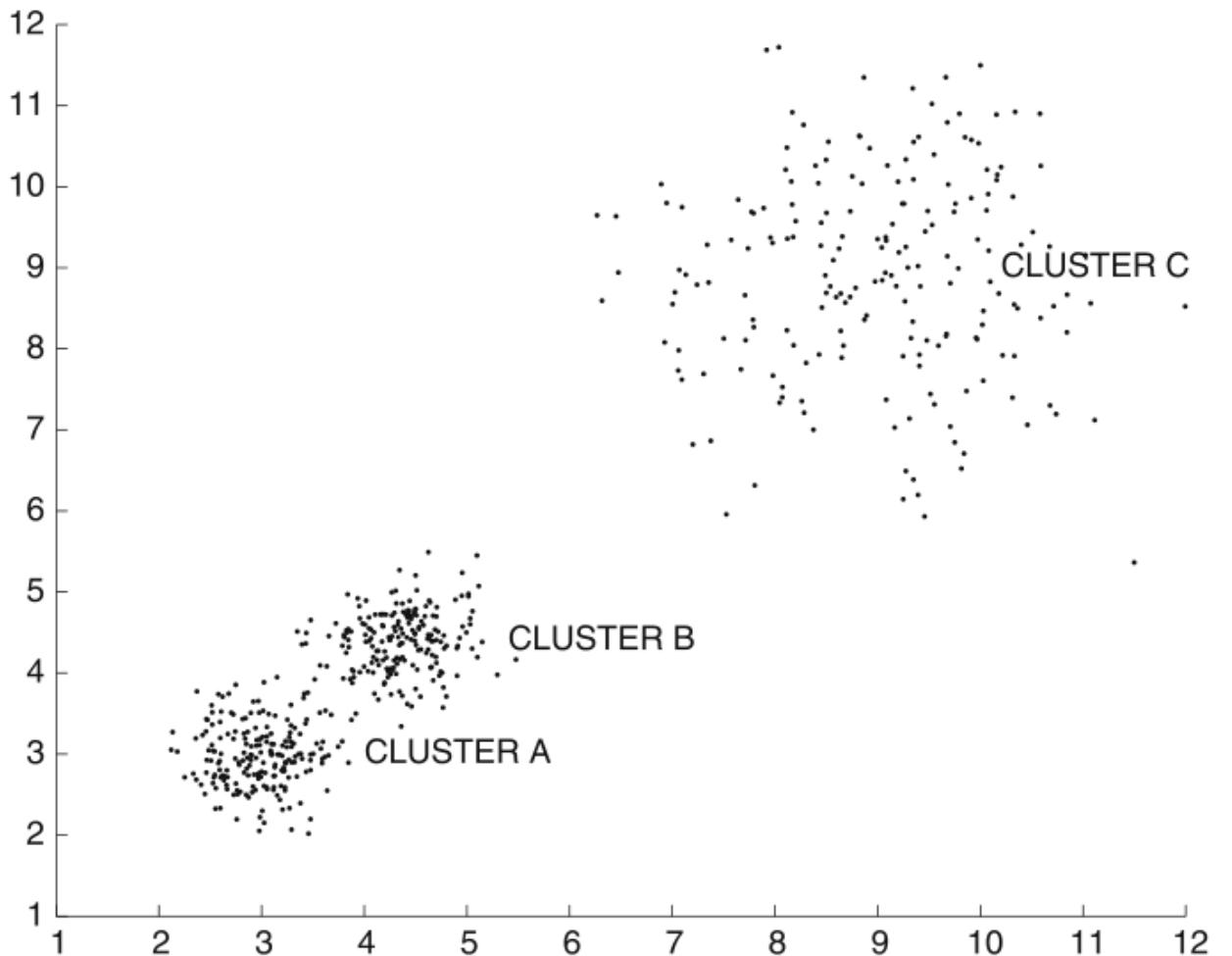
Учитывает выбросы

4

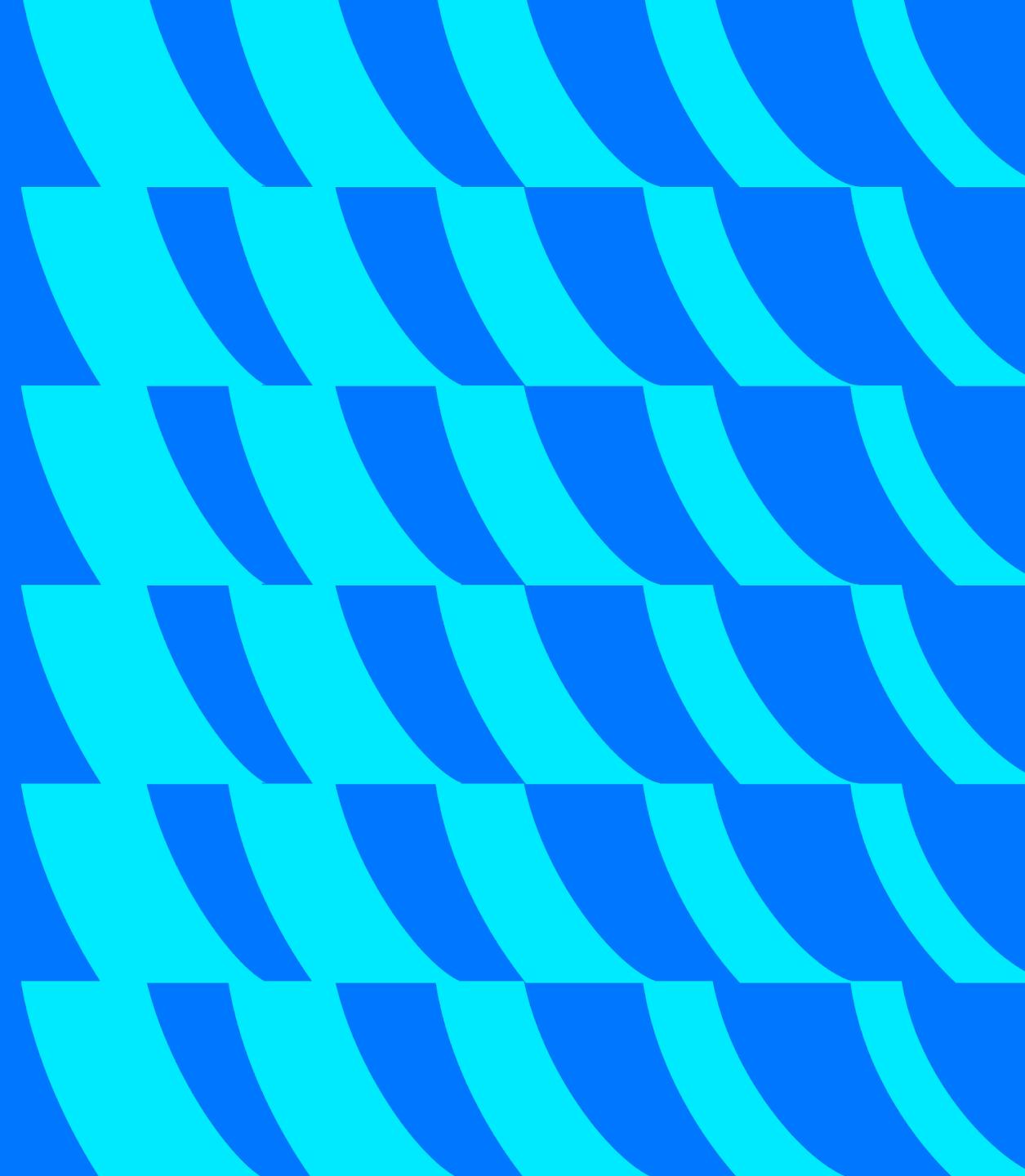
Довольно быстр, если
умеем быстро считать
соседей

Недостатки

- Не работает при различных плотностях кластеров
- Не всегда выявит кластеры

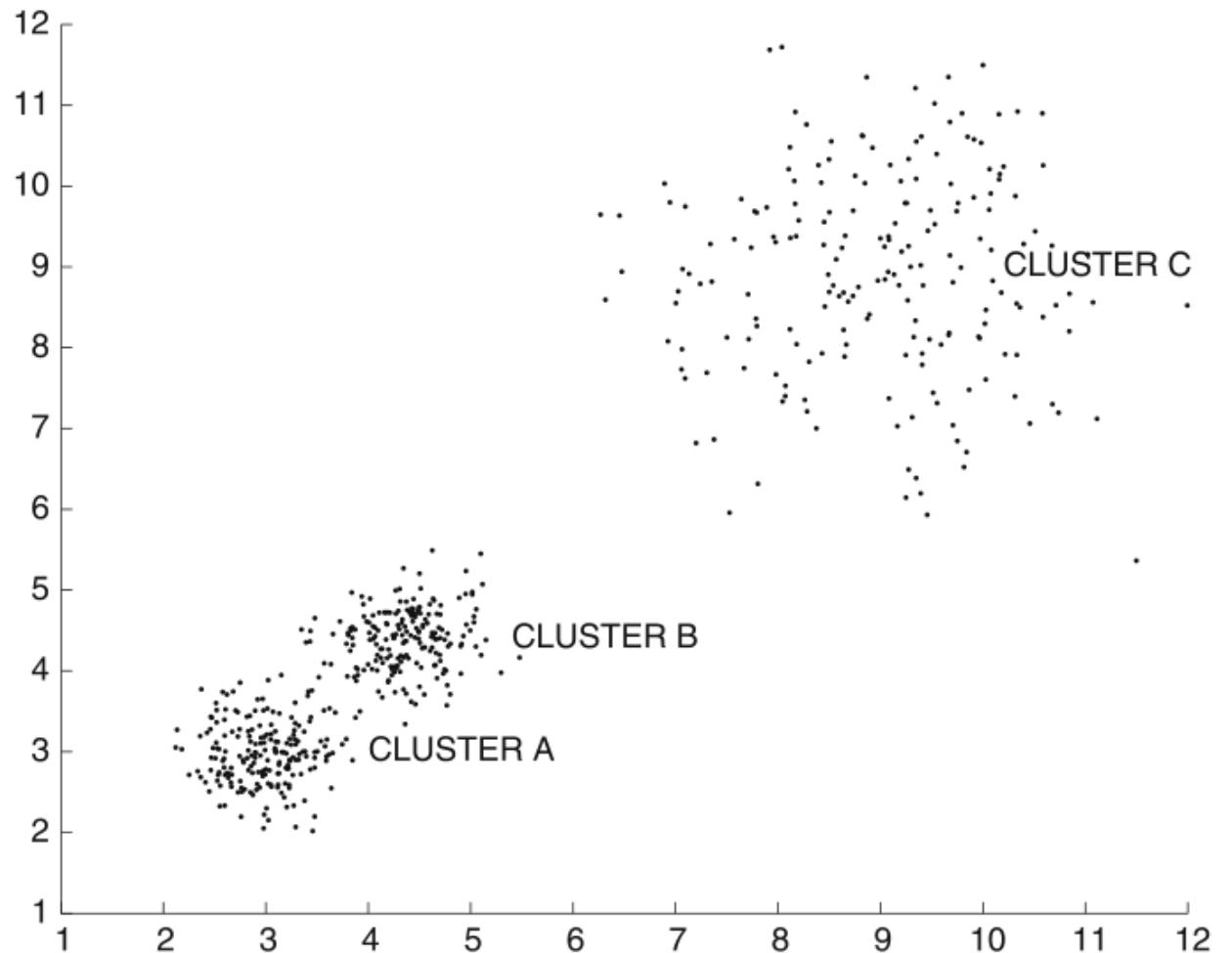


Алгоритмы,
основанные на
плотности:
HDBSCAN



Мотивация

- Хотелось бы всё-таки искать кластеры среди точек разной плотности

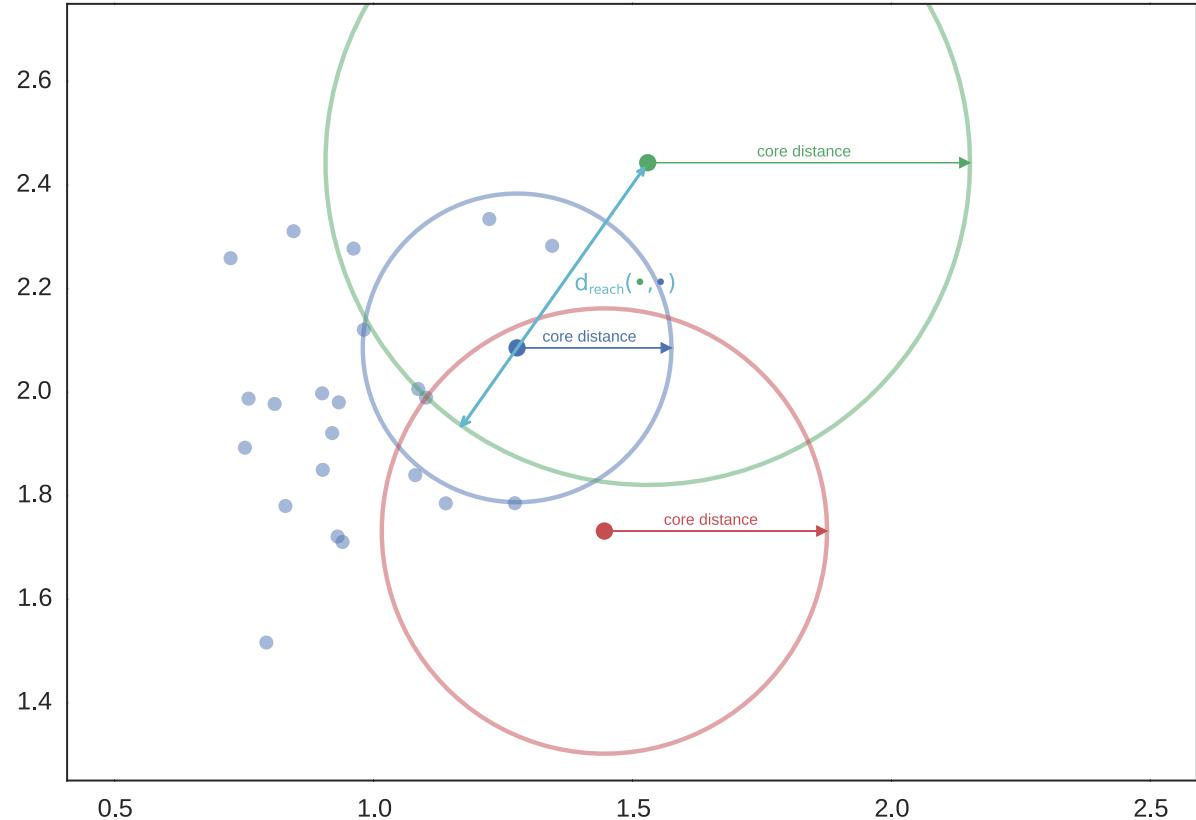


Модифицируем расстояние

- По сути, мы пытаемся найти «островки» повышенной плотности среди всего разреженного «моря» точек
- Давайте разнесём наше «море» и «острова» подальше друг от друга, то есть сделаем «море» ещё более разреженным, а точки внутри «островов» не будем трогать
- Введём новое расстояние доступности (Reachability):

$$d_{\text{mreach}-k}(a, b) = \max\{\text{core}_k(a), \text{core}_k(b), d(a, b)\},$$

где $\text{core}_k(a)$ — core-расстояние точки a , как в DBSCAN, $d(a, b)$ — оригинальное расстояние между точками

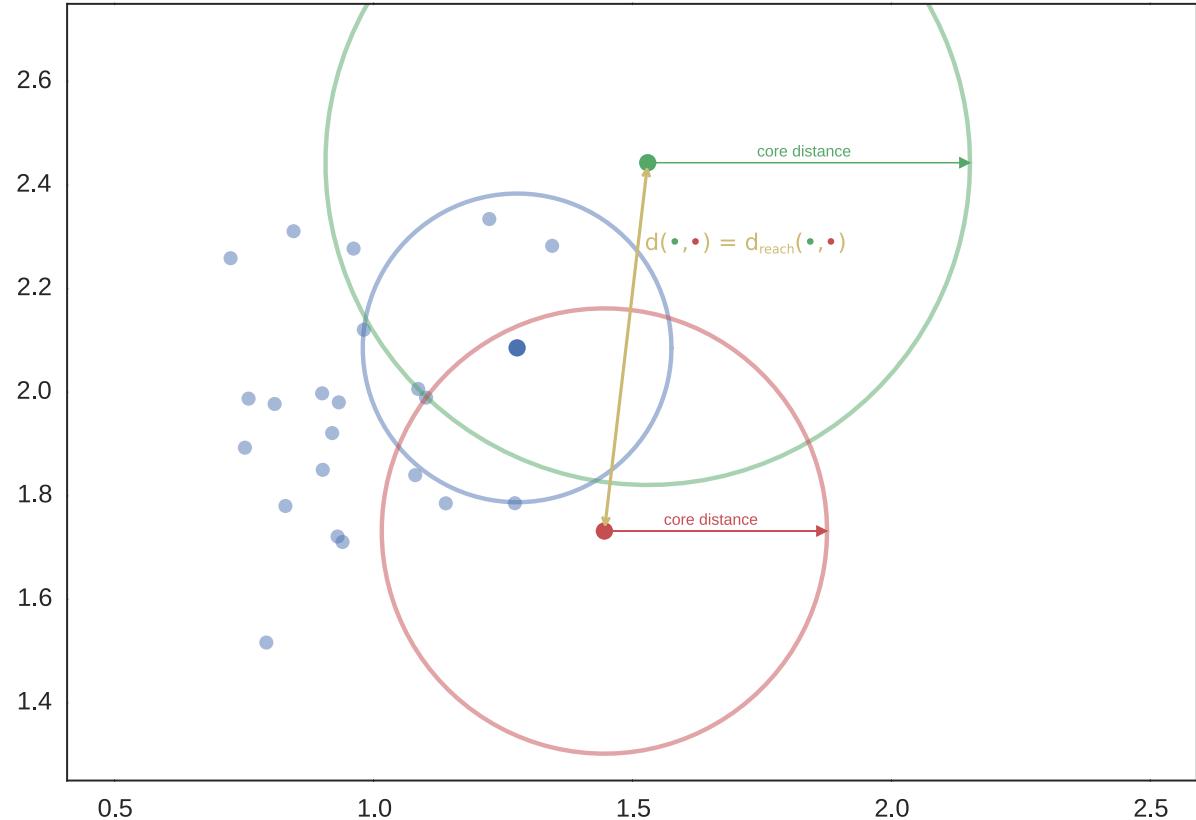


Модифицируем расстояние

- По сути, мы пытаемся найти «островки» повышенной плотности среди всего разреженного «моря» точек
- Давайте разнесём наше «море» и «острова» подальше друг от друга, то есть сделаем «море» ещё более разреженным, а точки внутри «островов» не будем трогать
- Введём новое расстояние доступности (Reachability):

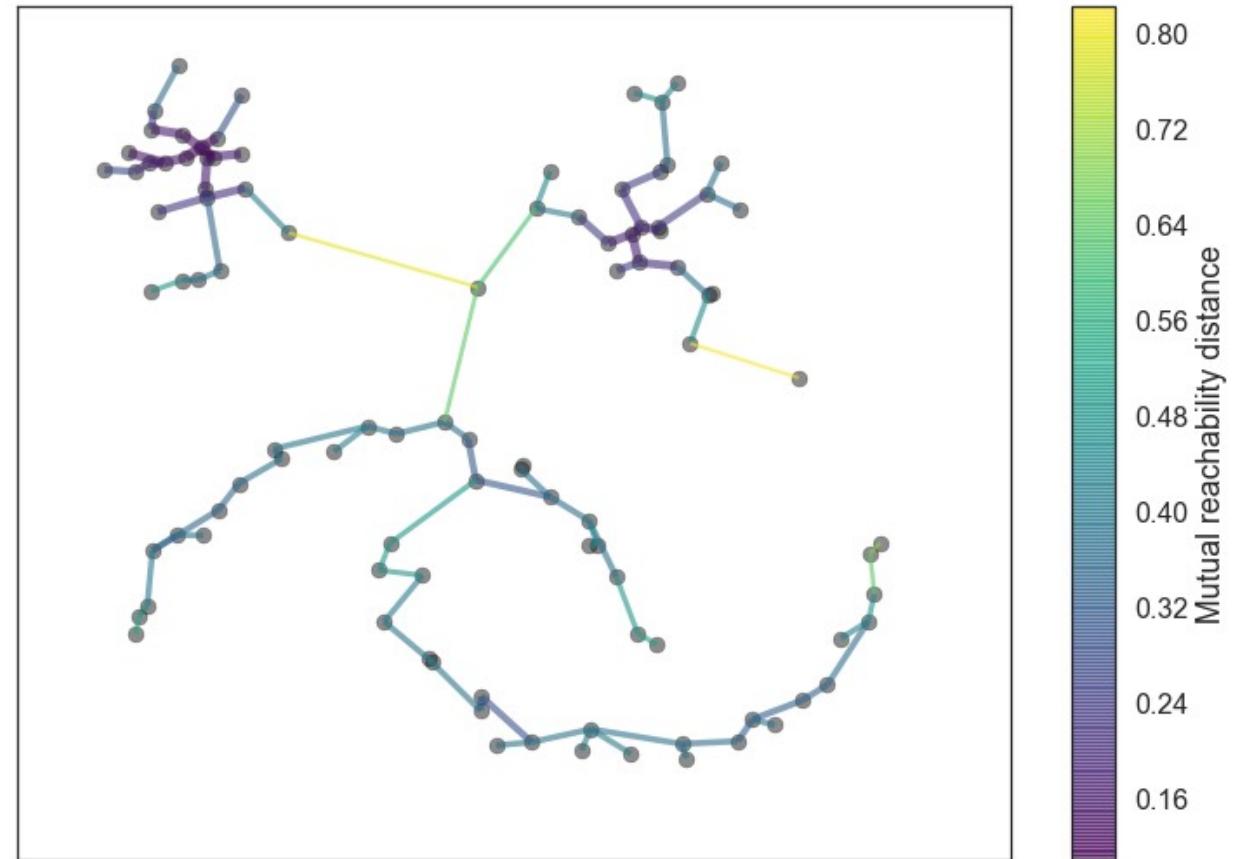
$$d_{\text{mreach}-k}(a, b) = \max\{\text{core}_k(a), \text{core}_k(b), d(a, b)\},$$

где $\text{core}_k(a)$ — core-расстояние точки a , как в DBSCAN, $d(a, b)$ — оригинальное расстояние между точками



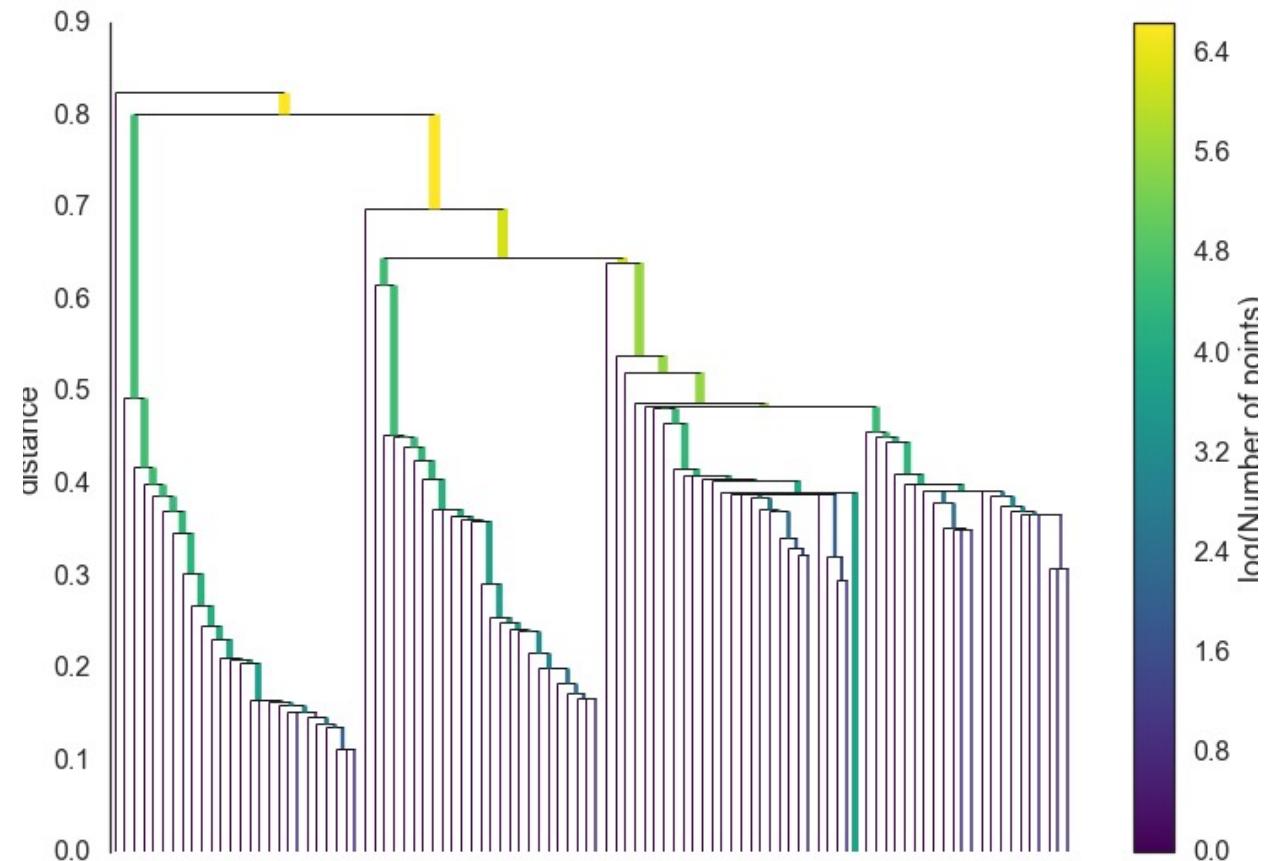
Минимальное оставное дерево

- Строим по полученным новым расстояниям граф (вершина — исходная точка, ребро соединяет две точки с весом $d_{\text{mreach}-k}(a, b)$)
- Получим большой граф, который надо уменьшить. Можно при построении сразу выкидывать ребра больше некоторого ϵ
- Построим минимальное оставное дерево, например, с помощью [алгоритма Прима](#)



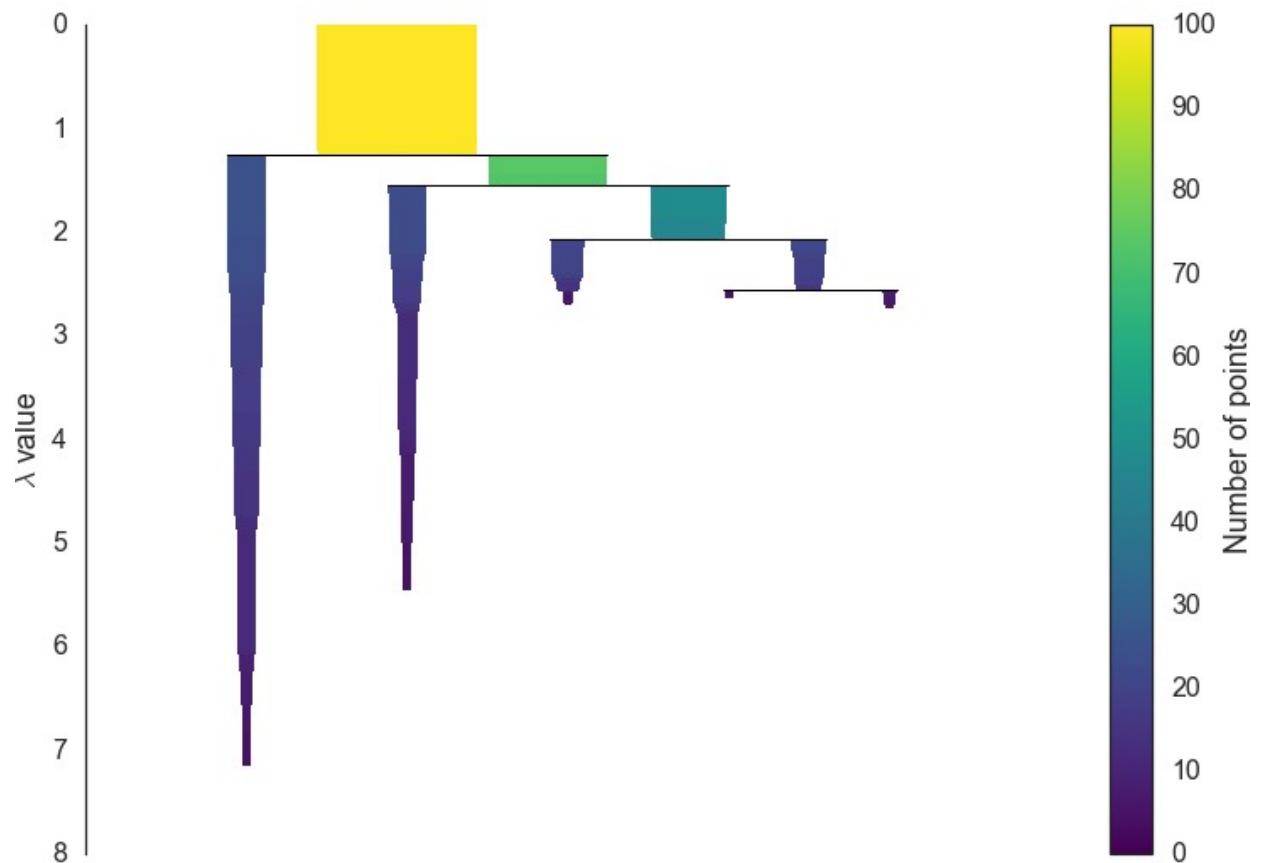
Строим иерархию

- Идём снизу вверх, как обычно, а именно от самых коротких к самым длинным рёбрам
- Объединяем рёбра в кластеры, используя структуру Union-Find



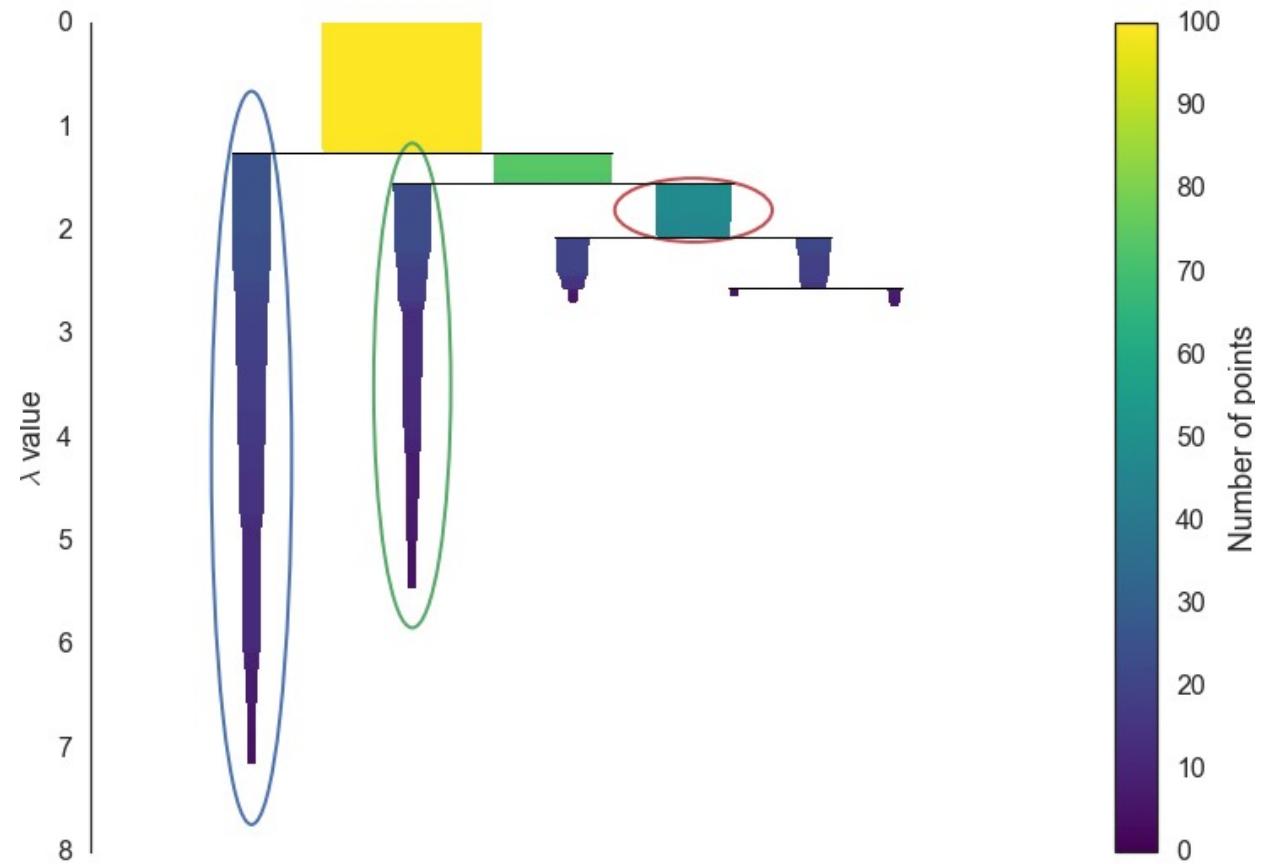
Конденсируем дерево

- Идём сверху вниз и анализируем ветвления
- Точки либо просто отваливаются (один из кластеров $< \text{min_cluster_size}$), либо образуют два полноценных кластера



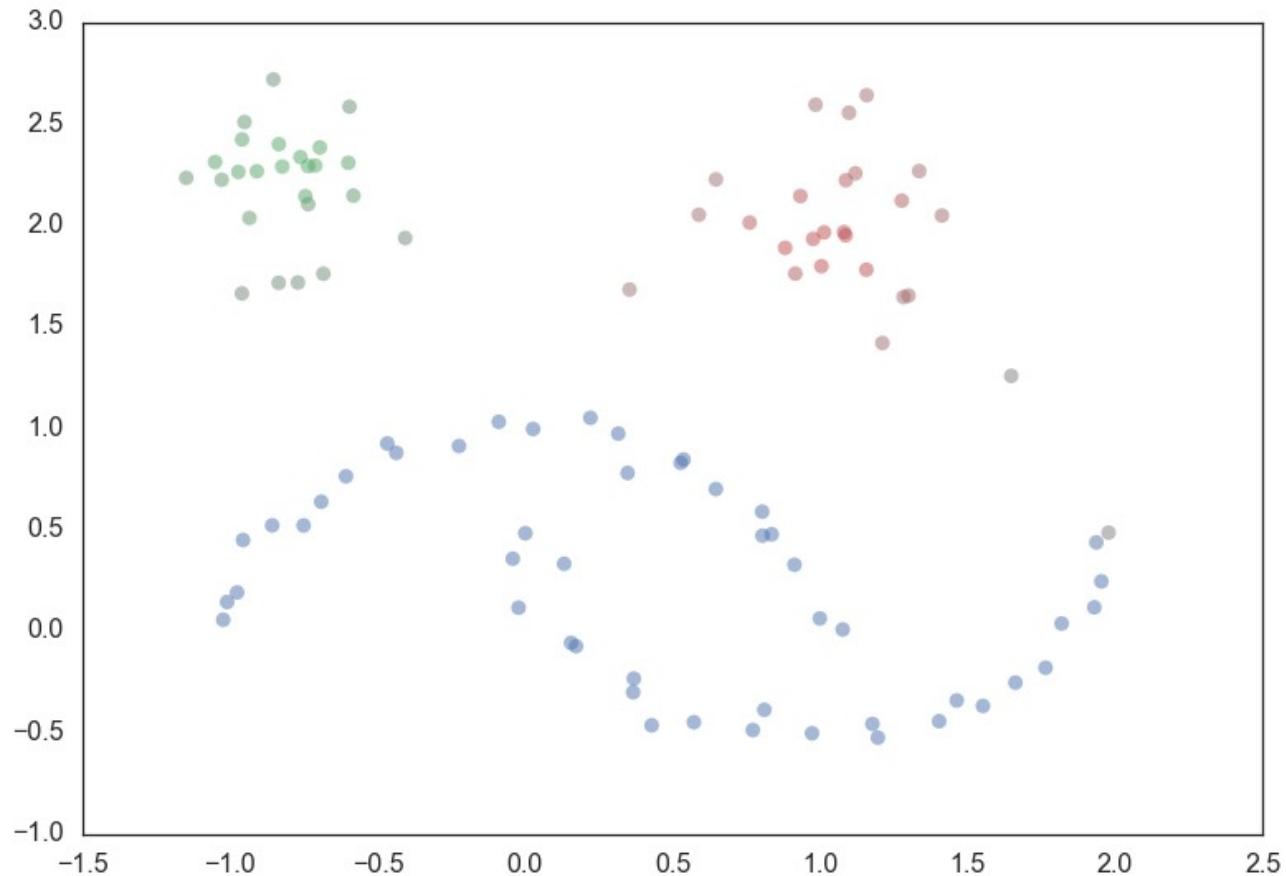
Выделяем кластеры

- Хотим видеть в качестве кластеров наиболее долгоживущие
- Не хотим учитывать кластеры с малым временем жизни: это, скорее всего, какие-то артефакты
- Если выбираем вершину в качестве кластера, то мы не можем далее выбрать её «потомков» в качестве отдельных кластеров
- Условно считаем площади этих «сосулек». Если площадь «детей» больше площади «родителя», то разделение правомерно, и мы считаем площадью «родителя» сумму его «детей». Иначе разделение плохое — склеиваем обратно



Пример

Более подробно про алгоритм можно почитать в [официальной документации](#)



Преимущества



1

Не требует k

2

Кластеры
произвольной
формы

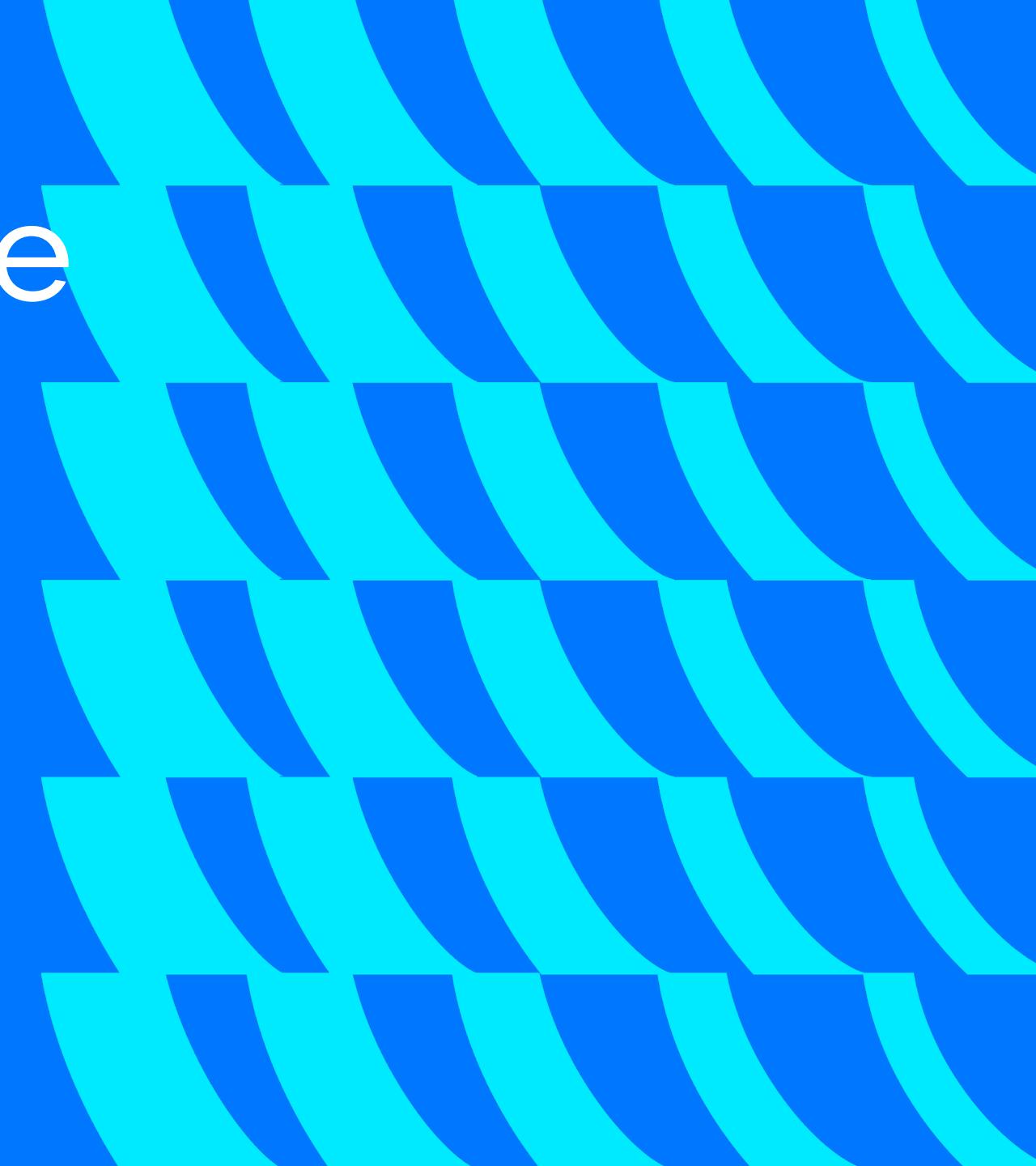
3

Учитывает выбросы

4

Работает при
различных плотностях
кластеров

Вероятностные алгоритмы: Gaussian Mixture

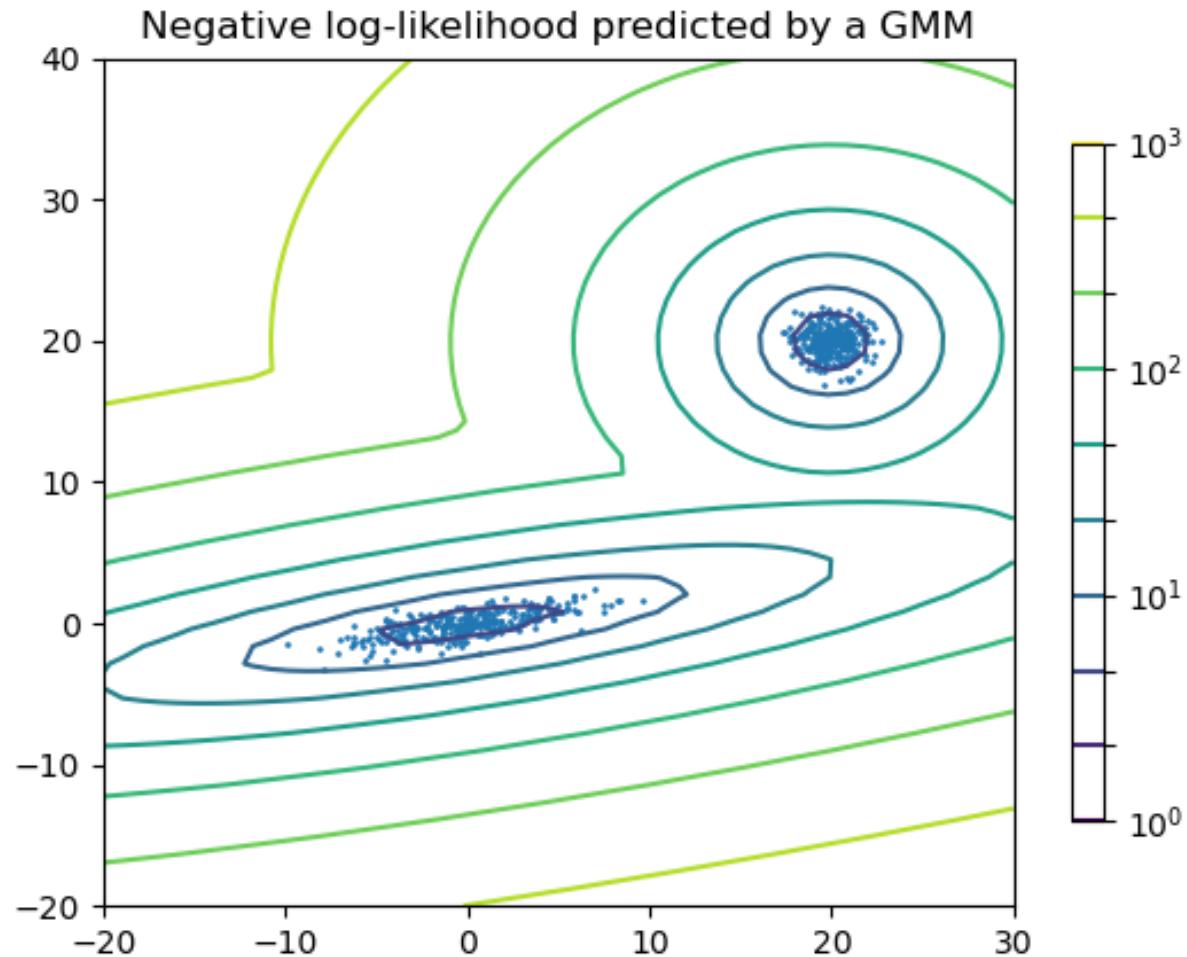


Идея

- Предположим, что наши данные — это смесь данных из разных гауссовых распределений
- Тогда можем попытаться оценить параметры этих распределений, основываясь на ММП:

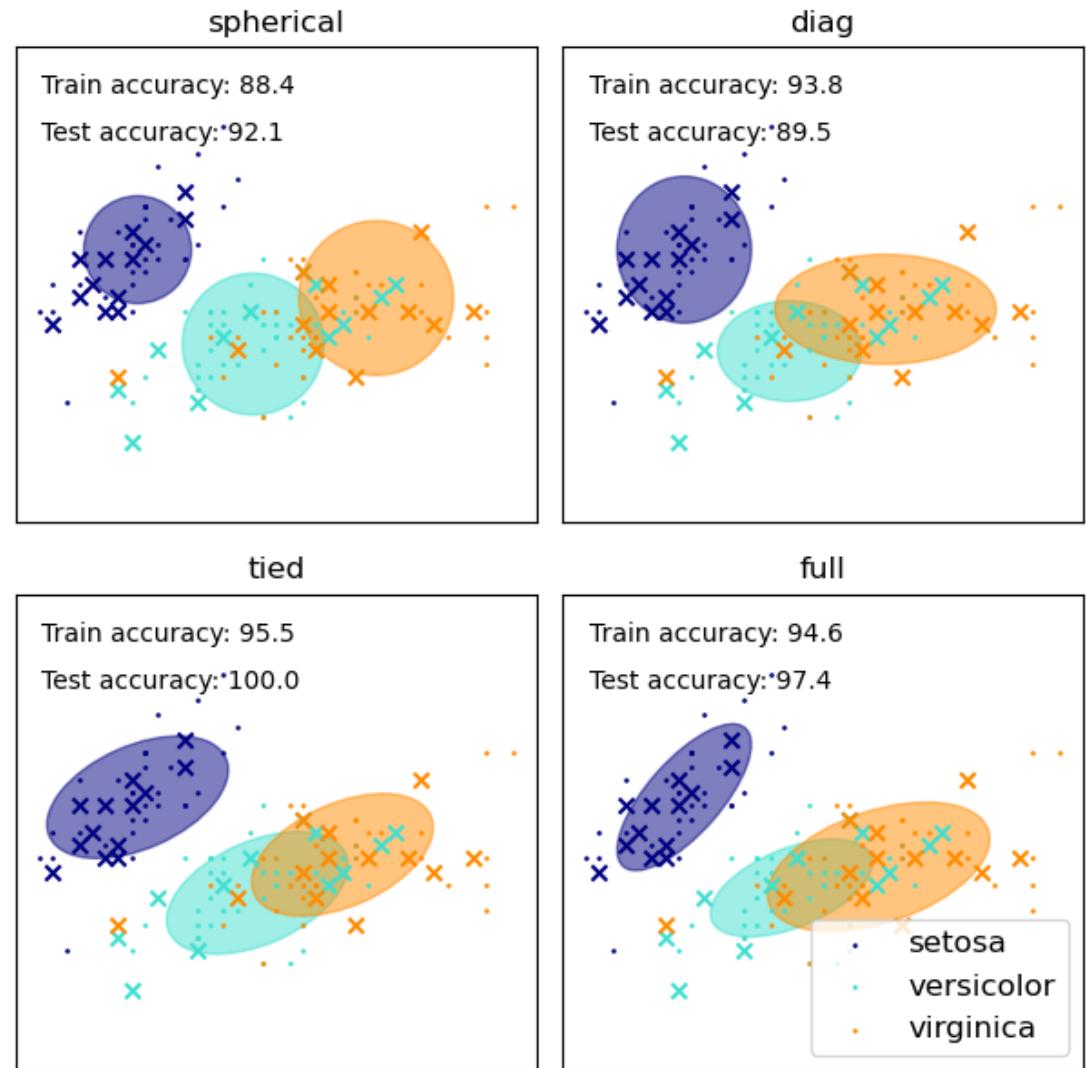
$$L(\boldsymbol{\theta}; \mathbf{X}) = p(\mathbf{X} | \boldsymbol{\theta}) = \int p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) d\mathbf{Z} = \int p(\mathbf{X} | \mathbf{Z}, \boldsymbol{\theta}) p(\mathbf{Z} | \boldsymbol{\theta})$$

- Для оценки параметров распределений используется ЕМ-алгоритм
- Можно думать об алгоритме как о некотором обобщении k-means



Параметры алгоритма

- `n_components`: количество компонент (распределений), из которых состоят данные
- `covariance_type`: тип ковариаций (задает ограничения на форму)
- `init_params`: инициализация параметров (лучше оставлять по-умолчанию)



Преимущества



1

Один из самых
быстрых алгоритмов

2

Может работать с
разными формами
гауссовых
распределений

Недостатки



1

Нужно знать
параметр k

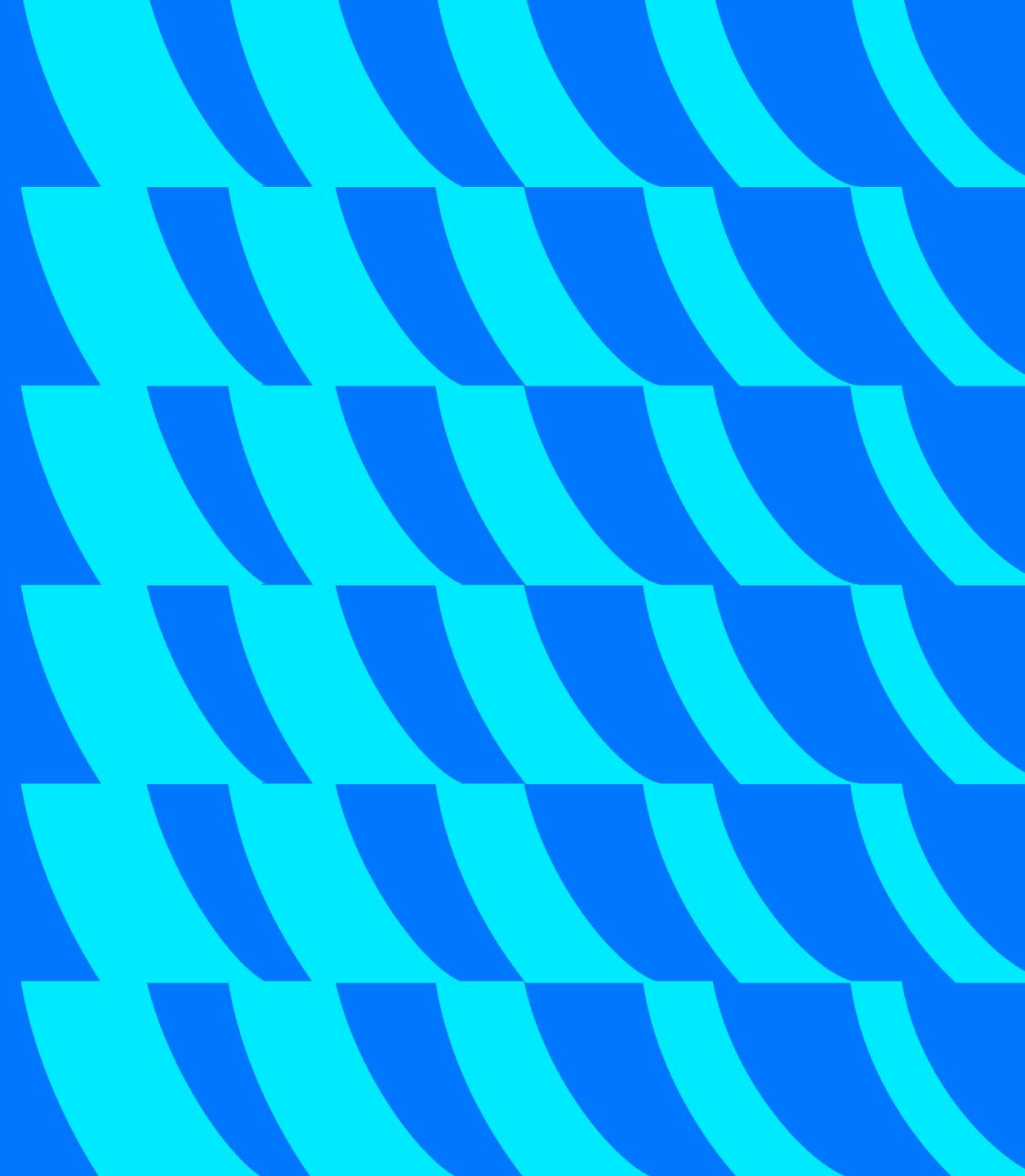
2

Ограничение на
«гауссовость»
данных

3

Может расходиться
на компонентах с
очень высокой
плотностью

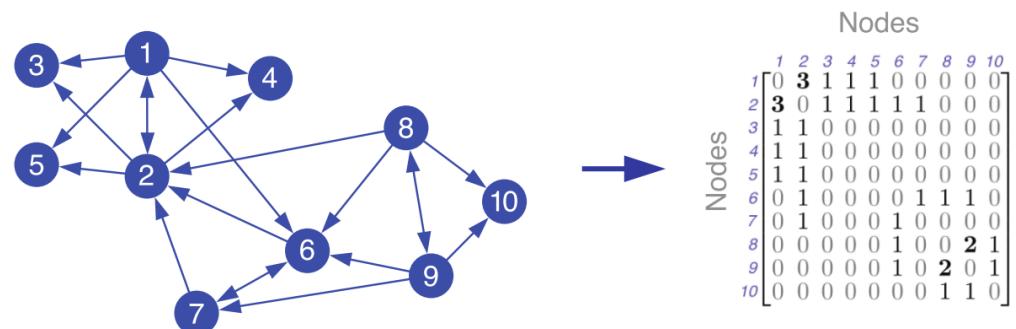
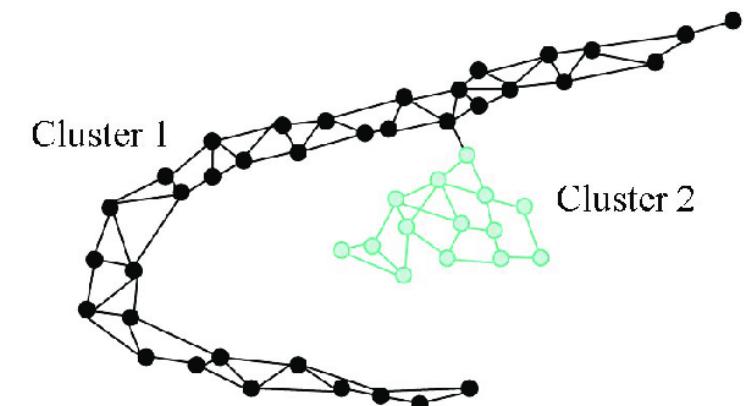
Spectral Clustering



Идея, граф на данных

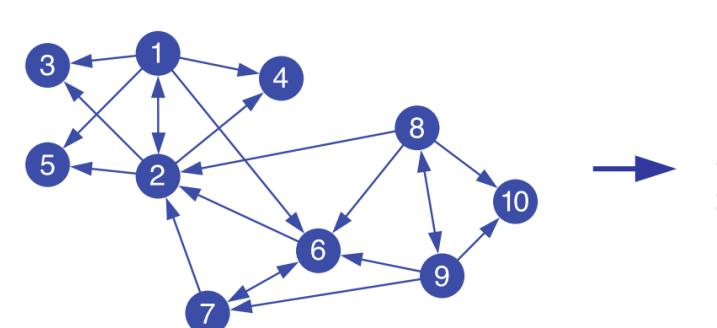
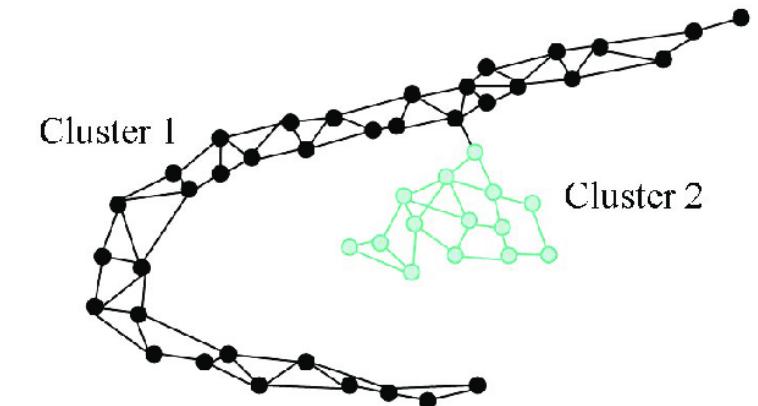
- Вернёмся к идеи DBSCAN, где наши точки объединяются в один кластер, если лежат в некоторой окрестности друг друга
- Тогда мы можем построить граф на наших данных: вершины — точки, рёбра проводим между точкой и всеми точками в её ϵ окрестности
- Альтернативно можем строить граф из ближайших соседей (соединяем вершину ребром с k ближайшими соседями).
- Ещё один вариант: можно построить полный граф, задав веса ребра в зависимости от расстояния между точками. Можно для разреживания данных отрезать веса меньше некоторого ϵ . Часто в качестве веса берут гауссово ядро (RBF):

$$S(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$



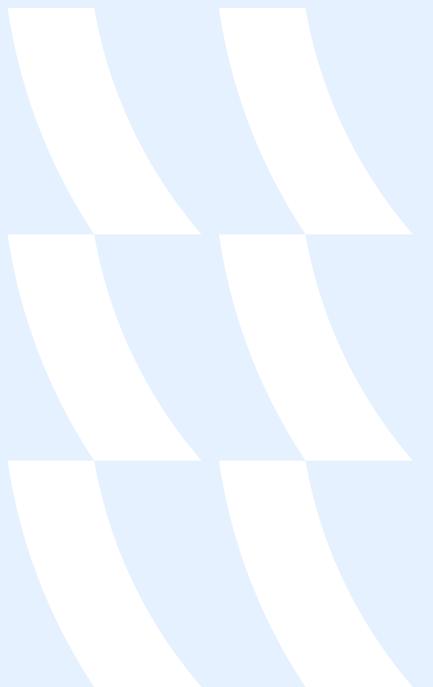
Алгоритм

- Имея граф, мы можем применить к нему алгоритм спектральной кластеризации
- В общем случае по матрице смежности мы должны получить её лапласиан. Однако в случае формирования матрицы методом RBF или ближайшими соседями можно использовать напрямую полученную матрицу
- В итоге для полученной матрицы мы ищем k собственных векторов
- Проецируем данные на полученные векторы, то есть получаем представление наших данных в пространстве меньшей размерности k .
- Применяем стандартный алгоритм кластеризации (например, k-means)



Nodes	1	2	3	4	5	6	7	8	9	10
1	0	3	1	1	1	0	0	0	0	0
2	3	0	1	1	1	1	0	0	0	0
3	1	1	0	0	0	0	0	0	0	0
4	1	1	0	0	0	0	0	0	0	0
5	1	1	0	0	0	0	0	0	0	0
6	0	1	0	0	0	0	1	1	1	0
7	0	1	0	0	0	1	0	0	0	0
8	0	0	0	0	1	0	0	2	1	0
9	0	0	0	0	0	1	0	2	0	1
10	0	0	0	0	0	0	0	1	1	0

Преимущества



1

Хорошо подходит
для невыпуклых
данных

2

Размечает все точки

3

По сравнению с
DBSCAN, HDBSCAN
может находить более
интересные кластеры,
нежадный алгоритм

Недостатки

1

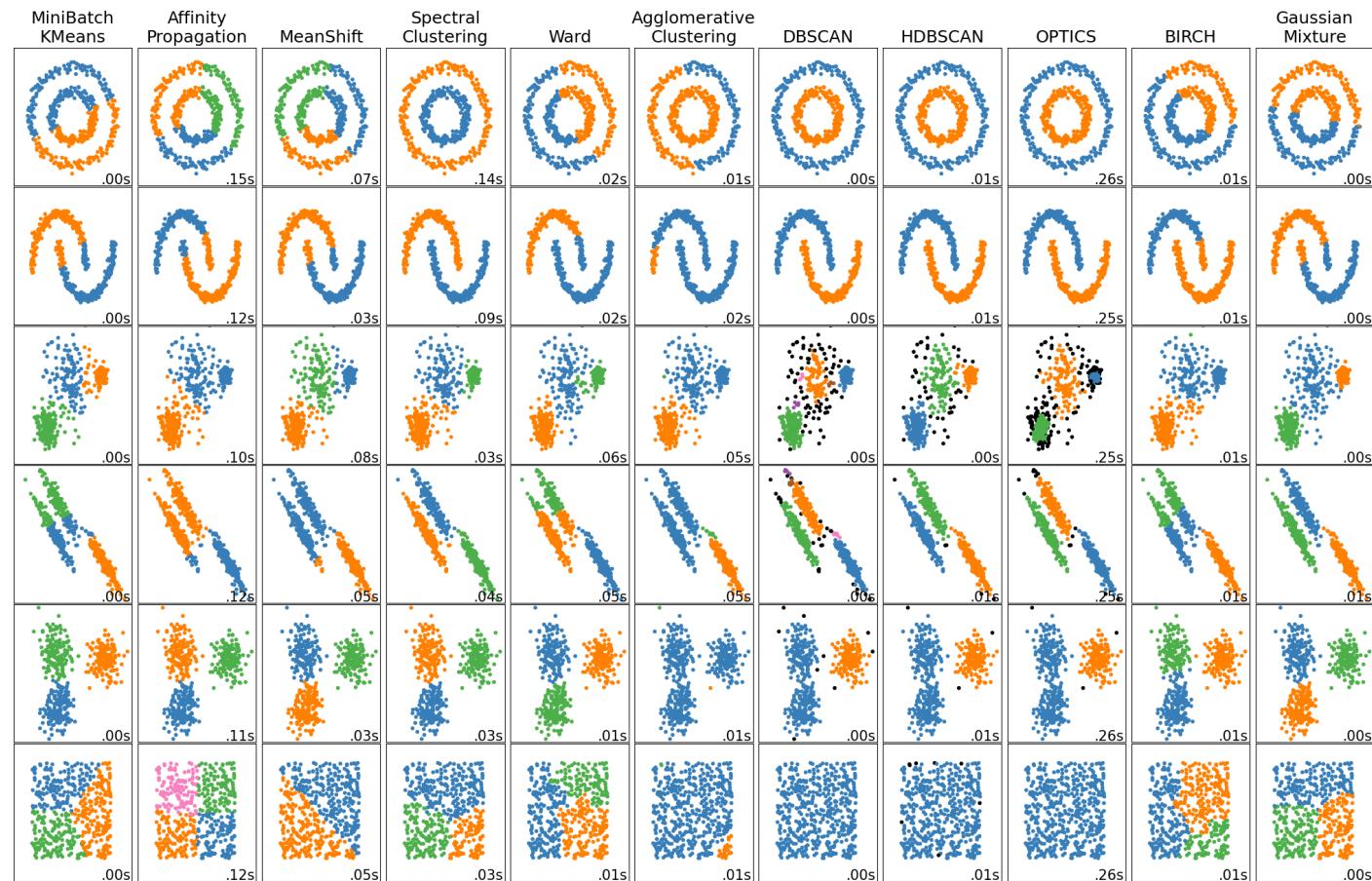
Нужно знать
параметр k

2

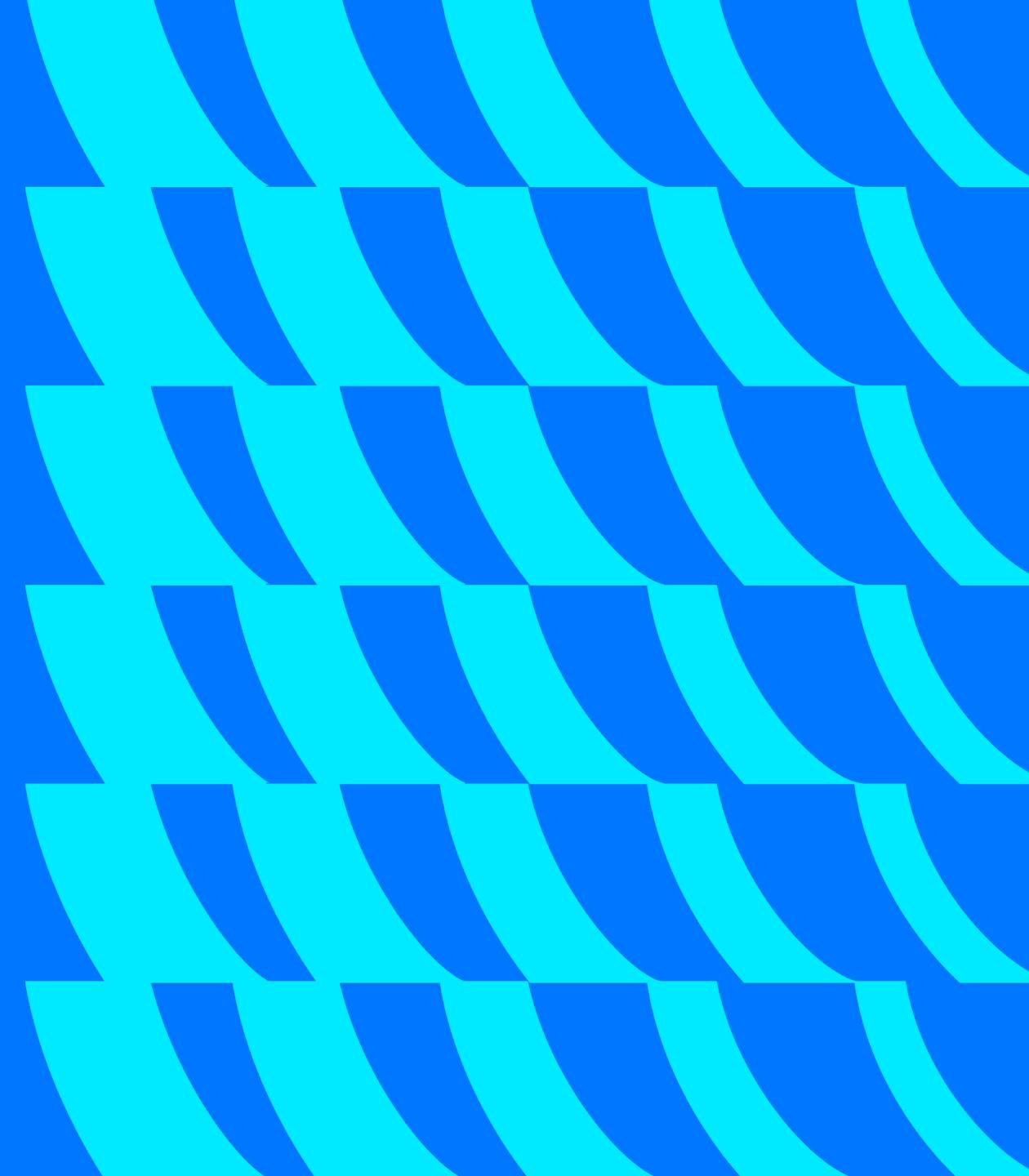
Может занять много
времени, особенно
если данные не
разреженные

Есть ещё много разных алгоритмов

Ознакомиться можно в [документации sklearn](#)



Оценка качества кластеризации



Adjusted Rand Index

Пусть $\hat{\pi}$ — это полученное разбиение на кластеры, а π^* — ground truth.

$$\text{Rand}(\hat{\pi}, \pi^*) = \frac{a + d}{a + b + c + d},$$

где

- a — количество пар объектов, находящихся в одинаковых кластерах в $\hat{\pi}$ и π^* ,
- b (c) — количество пар объектов в одном и том же кластере в $\hat{\pi}$ (π^*), но в разных в π^* ($\hat{\pi}$),
- d — количество пар объектов в разных кластерах в $\hat{\pi}$ и π^* .

- $\pi^* = [0,1,1,1,2,2,2,2]$
- $\hat{\pi} = [2,1,1,2,3,0,3,0]$

$\hat{\pi}$

	π^*				
	0	1	2	Σ	
0	0	0	2	2	2
1	0	2	0	2	2
2	1	1	0	2	2
3	0	0	2	2	2
Σ	1	3	4	8	

Индекс Жаккара, точность, полнота и F-мера

- $Jac(\hat{\pi}, \pi^*) = \frac{a}{a + b + c}$
- $Precision(\hat{\pi}, \pi^*) = \frac{a}{a + b}$
- $Recall(\hat{\pi}, \pi^*) = \frac{a}{a + c}$
- $F - measure(\hat{\pi}, \pi^*) = \frac{2Precision \cdot Recall}{Precision + Recall}$

$\hat{\pi}$

- $\pi^* = [0,1,1,1,2,2,2,2]$
- $\hat{\pi} = [2,1,1,2,3,0,3,0]$

	π^*				
	0	1	2	Σ	
0	0	0	2	2	2
1	0	2	0	2	2
2	1	1	0	2	2
3	0	0	2	2	2
Σ	1	3	4	8	

Меры валидности кластеров (без ground truth)

- Измеряют полученные разбиения по отношению к качествам хорошей кластеризации:
 - Компактность объектов внутри кластера
 - Разделимость кластеров друг от друга
- Про различные меры качества кластеризации и меры валидности кластеров в sklearn можно почитать [тут](#)

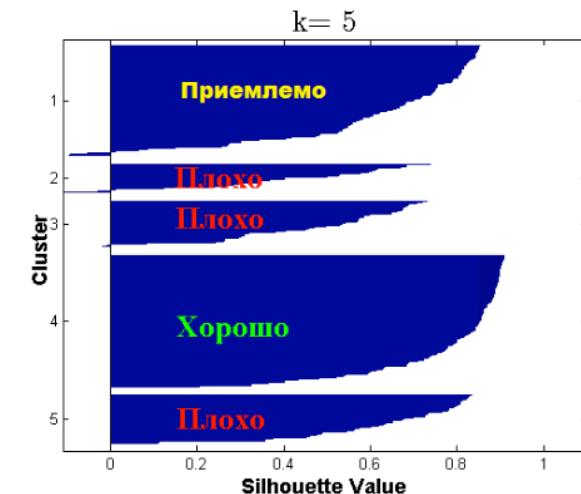
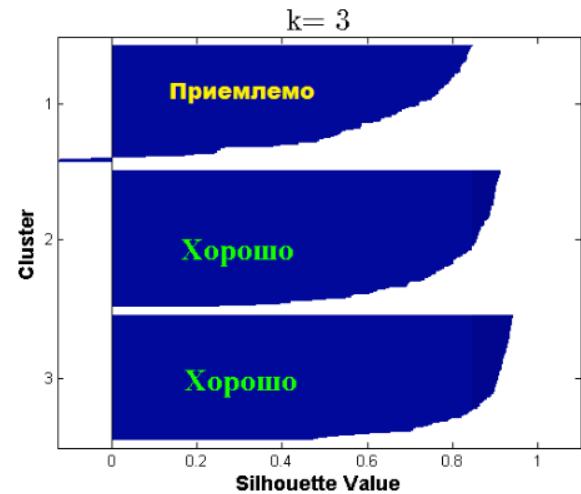
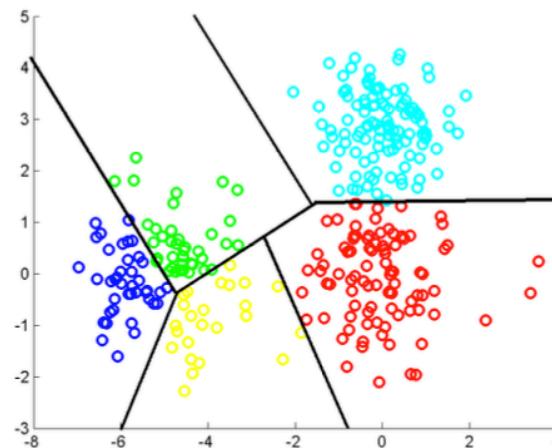
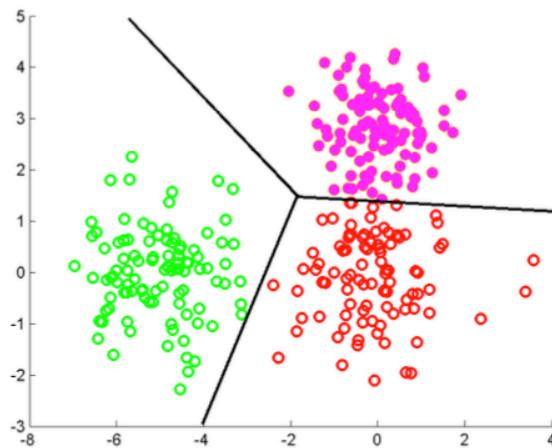
Критерий silhouette

Пусть дана кластеризация в K кластеров, и объект i попал в C_k :

- $a(i)$ — среднее расстояние от i объекта до объектов из C_k .
- $b(i) = \min_{j \neq k} b_j(i)$, где $b_j(i)$ — среднее расстояние от i объекта до объектов из C_j .
- Критерий:

$$silhouette(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

- Средний silhouette для всех точек из \mathbf{X} является критерием качества кластеризации



Важное про эвристики (Еще раз!)

- Эвристика и меры качества кластеризации носят лишь рекомендательный характер!
- Если они ничего не дают, то лучше ориентироваться на свои знания в предметной области
- Или выжать из полученной кластеризации максимум
- 3 из 5 полученных кластеров интерпретируются — и то хорошо

Литература

- Mohammed J. Zaki. Data Mining and Analysis: Fundamental Concepts and Algorithms. Ch. 3
- Jure Leskovec, Anand Rajaraman, Jeffrey D. Ullman. Mining of Massive Datasets. Ch. 7
- Andrew R. Webb, Keith D. Copsey. Statistical Pattern Recognition. Ch. 11
- A Tutorial on Spectral Clustering

Спасибо за внимание