

# Unsupervised classification

Лекция 3. K-means, Hierarchical clustering, DBSCAN





# Определения


**Кластеризация** — это задача разбиения множества объектов на подмножества, называемые кластерами.

- Внутри каждой группы должны оказаться «похожие» объекты, а объекты разных группы должны быть как можно более отличны.
- Главное отличие кластеризации от классификации состоит в том, что для обучающей выборки отсутствуют правильные ответы.

# Классификация



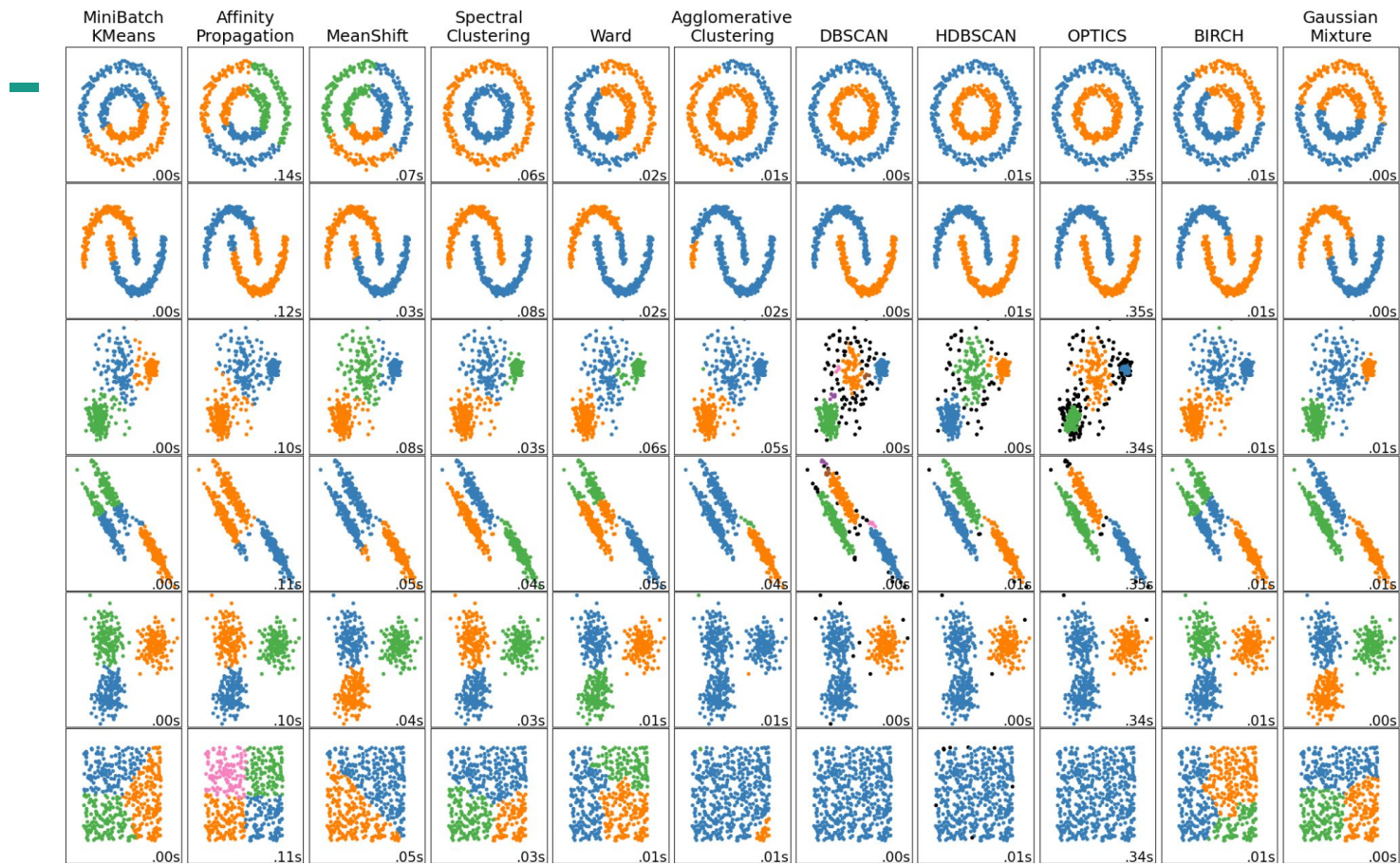
1. Partitioning clustering [k-means]
2. Hierarchical clustering
3. Density-based clustering [DBSCAN]
4. Model-based clustering [Gaussian Mixture Models (GMM)]
5. Graph-based clustering [Spectral clustering]
6. Grid-based clustering

- 
1. Hard clustering - каждому объекту присваивается одна метка класса
  2. Soft (or fuzzy) clustering - объекты могут попадать не в один класс

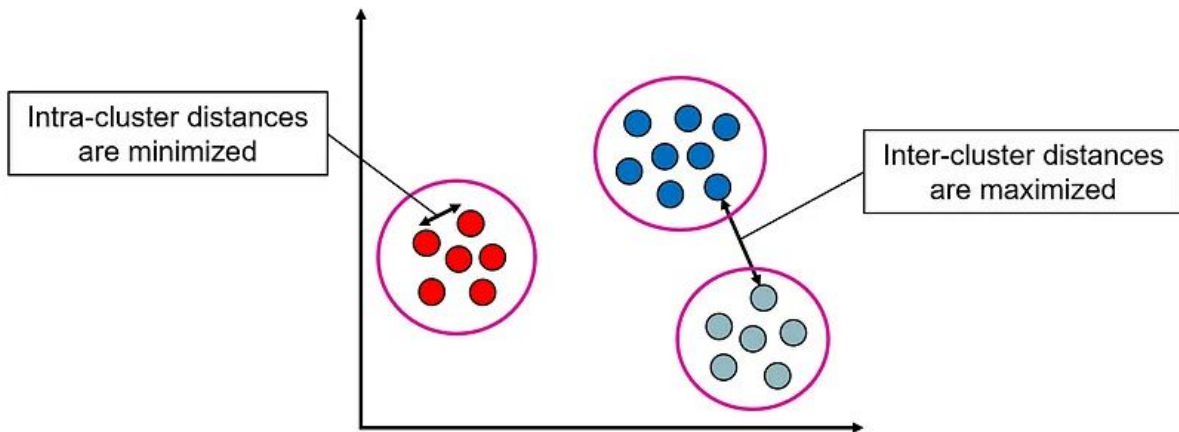
# Задачи кластеризации



1. **Исследование рынка и сегментация клиентов:** Определение групп клиентов на основе их покупательных привычек и предпочтений для направленного маркетинга и персонализированных предложений обслуживания.
2. **Анализ социальных сетей:** Выявление сообществ и понимание социальной динамики.
3. **Анализ биологических данных:** Группировка генов или белков с похожими функциональными характеристиками.
4. **Финансы:** Группировка акций по секторам на основе различных финансовых показателей и поведения рынка.
5. **Градостроительство:** Кластеризация помогает в градостроительстве, объединяя области с похожими характеристиками использования земли или демографии.
6. **Поиск информации:** Группировка похожих документов для облегчения эффективного поиска релевантной информации на основе запросов пользователей или интересов.
7. **Интернет поиск:** Поисковые системы используют кластеризацию для группировки похожих результатов.
8. **Сегментация изображений:** Разделение изображений на значимые кластеры, такие как выделение объектов на фоне или идентификация похожих особенностей на изображениях.
9. **Системы рекомендаций:** Рекомендация продуктов или контента путем группировки похожих пользователей на основе их предпочтений или прошлого поведения.
10. **Обнаружение аномалий:** Кластеризация может быть использована для обнаружения необычных паттернов или аномалий в областях, таких как выявление мошенничества и мониторинг сетей.



# Цели при обучении



В результате мы бы хотели:

1. Чтобы объекты внутри кластера располагались плотно
2. А расстояние между кластерами было бы большим



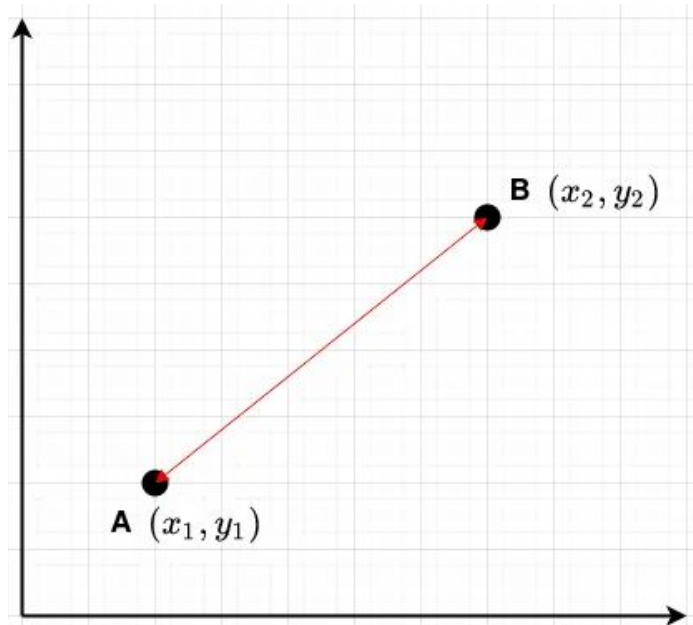
# Метрики расстояния

1. Manhattan distance
2. Euclidian distance
3. Minkowski distance
4. Cosine distance

Т. Д.



## Euclidian distance

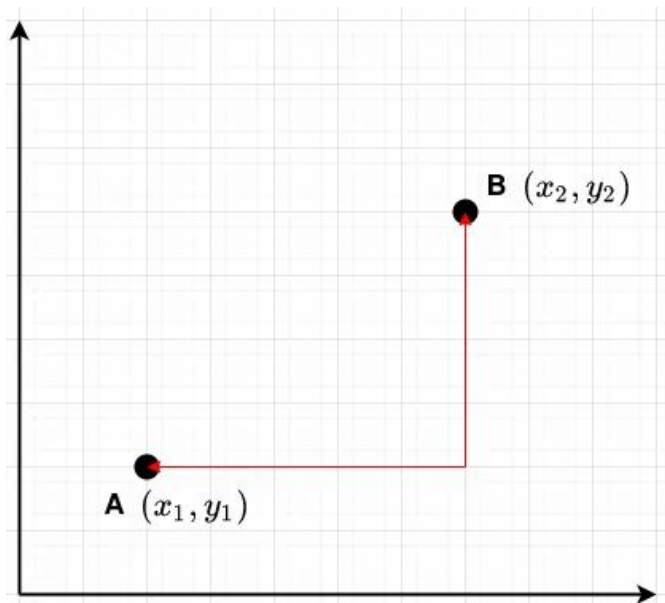


$$\text{Euclidean: } d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- Наиболее часто используемая метрика
- Также существует квадрат Евклидова расстояния(делает расстояние между удалёнными объектами ещё больше)



Manhattan distance



$$\text{Manhattan: } d(A, B) = \sum_{i=1}^p |x_i - y_i|$$

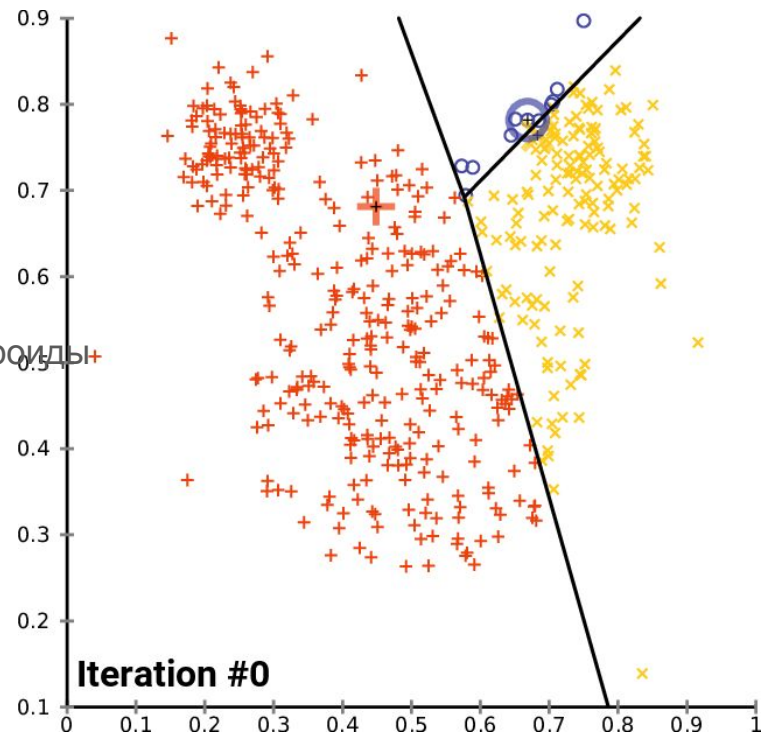
Практически всегда даёт на выходе тоже самое, что и Евклидово расстояние, но более устойчиво к выбросам



# K-means

Что нам нужно?

1. Самостоятельно прикинуть число кластеров и выбрать центры
2. Определить наши объекты к ближайшему кластеру
3. Пересчитать центры кластеров
4. Повторять шаги 2 и 3 пока ничего не перестанет меняться





## Как выбрать центроиды?

1. Рандомные точки в нашем пространстве
2. Выбираем из объектов выборки



## Игрушечный пример



## Недостатки

- Главный недостаток -> нужно знать  $k$  заранее
- От того какие именно точки для  $k$  мы выберем зависит и результат кластеризации
- Не подходит для работы с большими данными, которые формируют кластеры сложной структуры (признаки должны быть распределены равномерно)



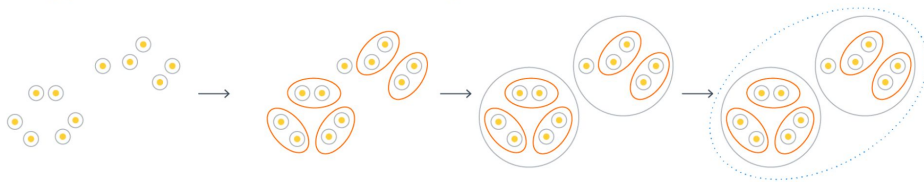
## **Что нужно сделать с данными для успешного успеха, возможного...**

1. Cleaning
2. Imputation
3. Scale
4. Reduction

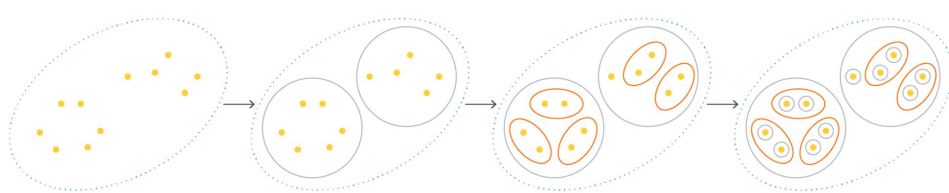
# Hierarchical clustering

- Agglomerative (bottom-up approach)
- Divisive (top-down approach)

Agglomerative Hierarchical Clustering



Divisive Hierarchical Clustering







# Agglomerative Hierarchical clustering

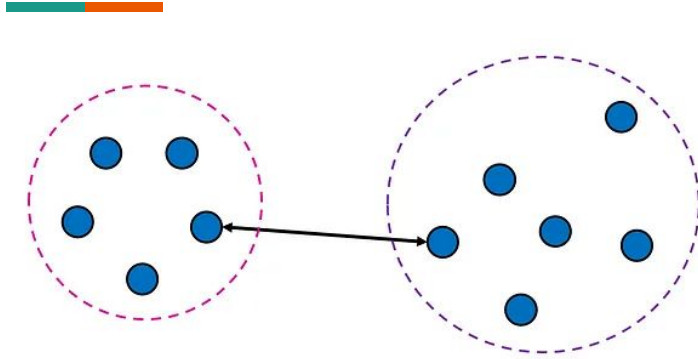
1. Первым шагом рассчитаем матрицу расстояний между каждым объектом выборки
2. Будем исходить из того что каждый объект является кластером
3. Сливаем ближайшие кластеры в один
4. Повторяем 3 шаг пока все наши объекты не превратятся в одно большое дерево
5. Потом анализируем дерево



# Linkage functions

1. Single
2. Complete
3. Average
4. Centroid-based
5. Ward

# Single linkage



$$d(C_1, C_2) = \min_{\mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2} d(\mathbf{x}_1, \mathbf{x}_2)$$

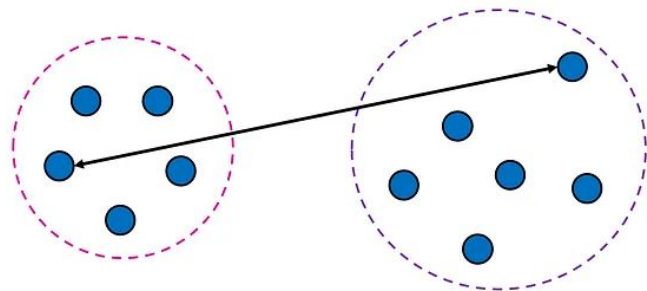
*Плюсы:*

- Может улавливать кластера сложной со сложной геометрией
- Подходит для всех метрик расстояния

*Минусы:*

- Чувствителен к шуму и выбросам
- Подвержен “эффекту цепочки”

# Complete linkage



$$d(C_1, C_2) = \max_{\mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2} d(\mathbf{x}_1, \mathbf{x}_2)$$

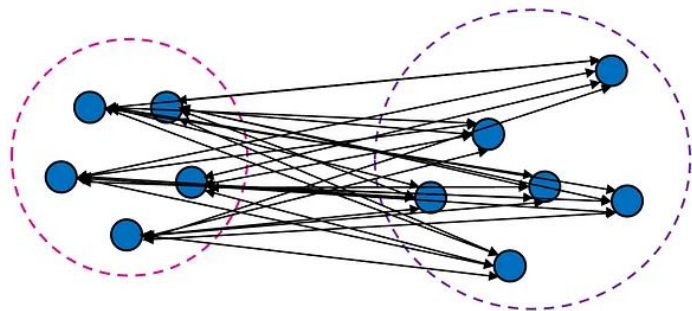
*Плюсы:*

- Более устойчивый к выбросам

*Минусы:*

- Склонен разбивать большие кластеры на более мелкие

# Average linkage



$$d(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{\mathbf{x}_1 \in C_1} \sum_{\mathbf{x}_2 \in C_2} d(\mathbf{x}_1, \mathbf{x}_2)$$

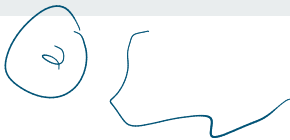
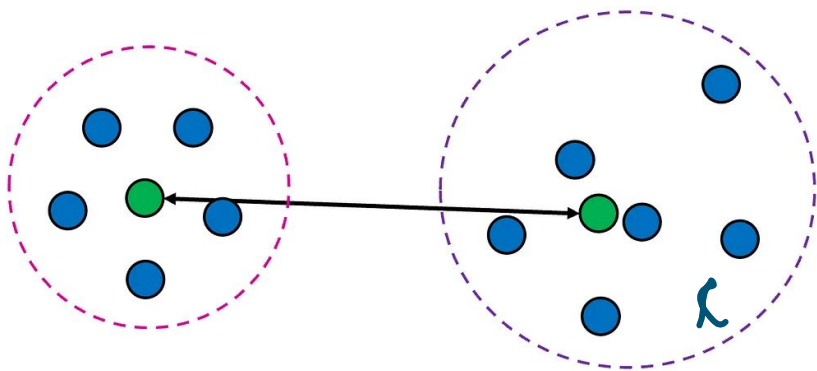
*Плюсы:*

- Сбалансированная версия между single linkage & complete linkage
- Более устойчив к выбросам и шуму

*Минусы:*

- Склонен выделять шаровидные кластера

# Centroid linkage



Плюсы:

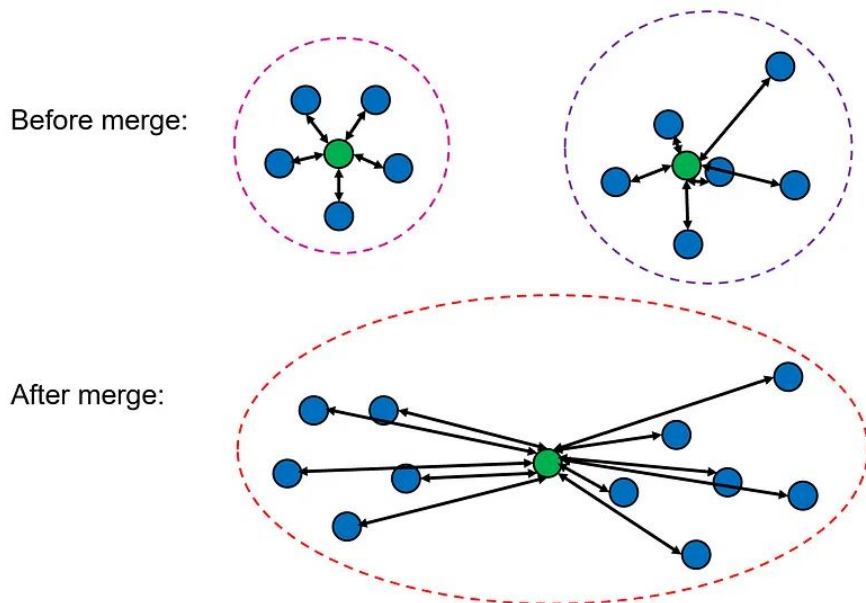
- Лучше чем single

Минусы:

- Можно использовать только с Евклидовым расстоянием
- При слиянии двух кластеров может увеличиваться расстояние до всех остальных кластеров

$$d(C_1, C_2) = d(\mathbf{c}_1, \mathbf{c}_2) = d\left(\left(\frac{1}{|C_1|} \sum_{\mathbf{x} \in C_1} \mathbf{x}\right), \left(\frac{1}{|C_2|} \sum_{\mathbf{x} \in C_2} \mathbf{x}\right)\right)$$

# Ward linkage



$$d(C_1, C_2) = \frac{|C_1||C_2|}{|C_1| + |C_2|} d(\mathbf{c}_1, \mathbf{c}_2)^2$$

*Плюсы:*

- Из всех перечисленных методов наверное самый оптимальный

*Минусы:*

- Можно использовать только с Евклидовым расстоянием
- Лучше всего работает когда кластера примерно одного размера

# Игрушечный пример, используем *Single linkage*



$i$	$x_1$	$x_2$
1	0	1
2	1	4
3	1	9
4	2	2
5	2	7
6	3	8
7	4	7
8	5	3
9	6	4
10	7	3

На самом деле алгоритм принимает *матрицу расстояний*

Point	1	2	3	4	5	6	7	8	9	10
1	0									
2	10	0								
3	65	25	0							
4	5	5	50	0						
5	40	10	5	25	0					
6	58	20	5	37	2	0				
7	52	18	13	29	4	2	0			
8	29	17	52	10	25	29	17	0		
9	45	25	50	20	25	25	13	2	0	
10	53	37	72	26	41	41	25	4	2	0



# Steps



Point	1	2	3	4	5,6	7	8	9	10
1	0								
2	10	0							
3	65	25	0						
4	5	5	50	0					
5,6	40	10	5	25	0				
7	52	18	13	29	2	0			
8	29	17	52	10	25	17	0		
9	45	25	50	20	25	13	2	0	
10	53	37	72	26	41	25	4	2	0

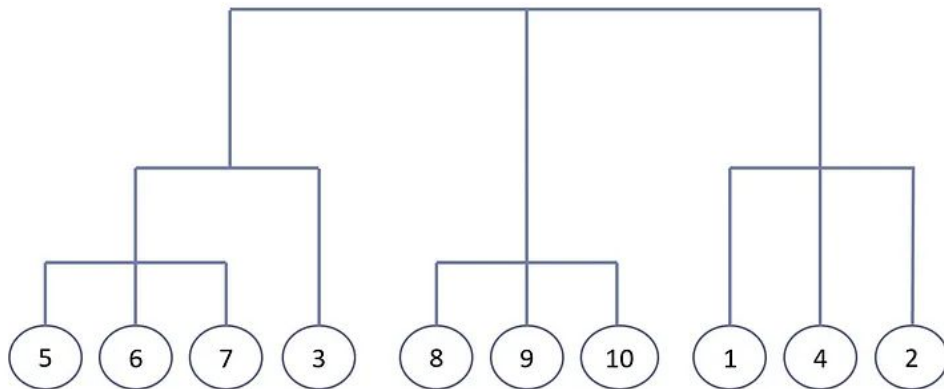
## Step 2



Point	1	2	3	4	5,6,7	8,9	10
1	0						
2	10	0					
3	65	25	0				
4	5	5	50	0			
5,6,7	40	10	5	25	0		
8,9	29	17	50	10	13	0	
10	53	37	72	26	25	2	0

# И т.д.

В итоге мы получаем **дендрограмму**. По оси X наши объекты, а по оси Y расстояние между ними, чтобы выбрать конечное количество кластеров, нужно обрезать дерево



# DBSCAN

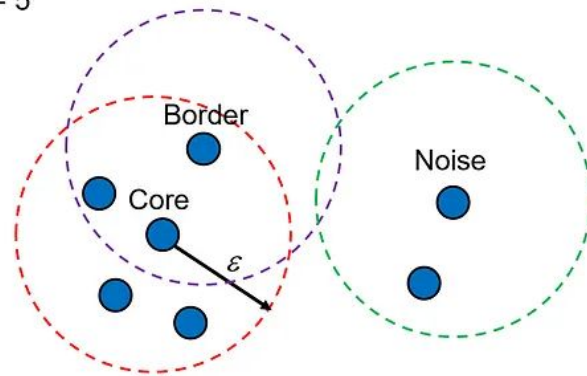
Критерии для присвоения каждому объекту метки:

1. Epsilon ( $\epsilon$ ): радиус вокруг которой он будет искать близкие точки
2. Minimum Points: минимальное число точек внутри радиуса Epsilon

Метки:

1. Core - точка является коровой если в радиус вокруг неё попадает минимальное число точек
2. Edge - точка в радиус которой не попадает достаточное количество других точек, но она находится в радиусе *core point*
3. Noise

MinPts = 5

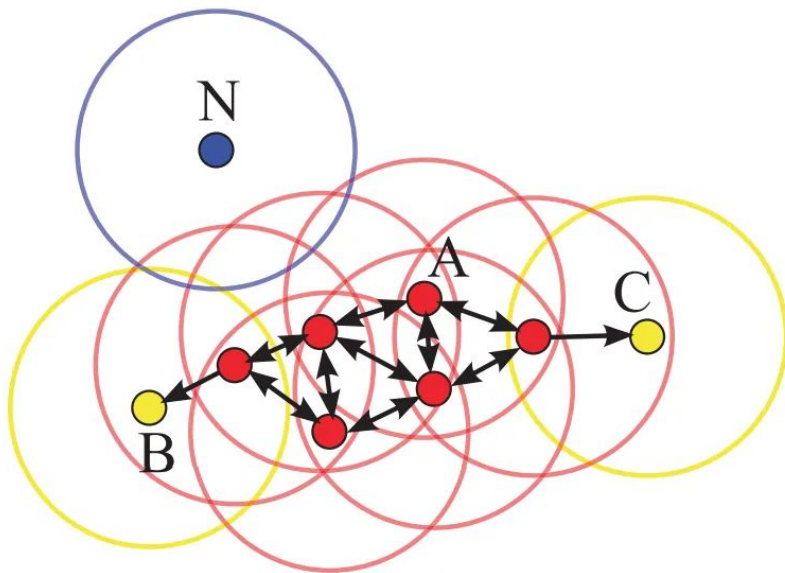




## Гиперпараметры:

1. eps
2. min\_samples
3. + distance

Немного обсудим как это будет работать...





## Преимущества

- Не требует задавать число кластеров
- Определяет кластеры сложной структуры в том числе и вложенные
- С помощью этого алгоритма можно и нужно детектить выбросы



## Недостатки

1. Сложно назвать это недостатком, но он не определяет центр кластера
2. Другая неприятность заключается в том, что кластеры могут сливаться или в них могут образовываться дырки
3. Достаточно сложно подобрать оптимальные гиперпараметры

Хорошая новость заключается в том, что у этого алгоритма есть много вариаций и они очень даже широко используются в анализе биологических данных







# Affinity Propagation

Данный алгоритм строит три матрицы:

1. Similarity
2. Responsibility
3. Availability



# Крутости

1. Не надо заранее знать число кластеров
2. Выдаёт осмысленные и хорошо структурированные кластеры, которые можно красиво реализовать

Из минусов это то, что плохо работает с большими данными



# Metrics

- Внешние
- Внутренние



## Внутренние

1. Среднее внутрикластерное расстояние - WCSS (Within-Cluster Sum of Squares)
2. Среднее межкластерное расстояние - BCSS (Between-Cluster Sum of Squares)
3. Коэффициент силуэта (Silhouette Index)

## BCSS & WCSS



BCSS (Between-Cluster Sum of Squares) - это взвешенная сумма квадратов евклидовых расстояний между центроидом (средним) каждого кластера и общим центроидом (средним) данных (the higher the better).

$$BCSS = \sum_{i=1}^k n_i \|\mathbf{c}_i - \mathbf{c}\|^2$$

WCSS (Within-Cluster Sum of Squares) - это сумма квадратов евклидовых расстояний между точками данных и соответствующими центроидами кластеров (the smaller the better).

$$WCSS = \sum_{i=1}^k \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mathbf{c}_i\|^2$$



## Коэффициент силуэта

$$s(\mathbf{x}_i) = \frac{b(\mathbf{x}_i) - a(\mathbf{x}_i)}{\max\{a(\mathbf{x}_i), b(\mathbf{x}_i)\}}$$

$$SI = \frac{1}{n} \sum_{i=1}^n s(\mathbf{x}_i)$$

$b(i)$  -> среднее расстояние между  $\mathbf{x}_i$  и объектами ближайшего класса

$a(i)$  -> среднее расстояние между  $\mathbf{x}_i$  и объектами класса

$$a(\mathbf{x}_i) = \frac{1}{|C_i| - 1} \sum_{\mathbf{x}_j \in C_i, j \neq i} d(\mathbf{x}_i, \mathbf{x}_j)$$

$$b(\mathbf{x}_i) = \min_{j \neq i} \frac{1}{|C_j|} \sum_{\mathbf{x}_j \in C_j} d(\mathbf{x}_i, \mathbf{x}_j)$$



# Интерпретация

Коэффициент силуэта, равный  $+1$ , указывает на то, что объект находится далеко от соседних кластеров.

Коэффициент силуэта, равный  $0$ , указывает на то, что объект находится на границе принятия решения между двумя соседними кластерами или очень близко к ней.


Коэффициент силуэта  $< 0$  указывает на то, что эти объекты, возможно, были отнесены к неправильному кластеру или являются выбросами.





## Внешние

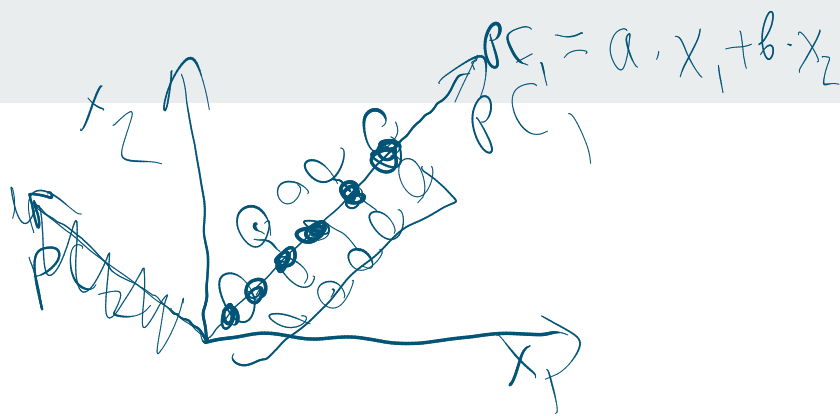
1. Гомогенность
2. Полнота
3. V- мера



Measure	Type	Range	Adjusted for Chance	Assumptions on Clusters	Scikit-Learn Function
Silhouette Index (SI)	Internal	$[-1, 1]$ higher is better	No	Spherical	<code>silhouette_score</code>
Calinski-Harabasz Index (CHI)	Internal	$[0, \infty]$ higher is better	No	Spherical, similar sized	<code>calinski_harabasz_score</code>
Davies-Bouldin Index (DBI)	Internal	$[0, \infty]$ lower is better	No	Spherical, similar sized	<code>davies_bouldin_score</code>
Adjusted Rand Index (ARI)	External	$[-1, 1]$ higher is better	Yes	None	<code>adjusted_rand_score</code>
V-Measure	External	$[0, 1]$ higher is better	No	None	<code>v_measure</code>
Fowlkes-Mallows Index (FMI)	External	$[0, 1]$ higher is better	Yes	None	<code>fowlkes_mallows_score</code>



## t-SNE, UMAP




[https://www.youtube.com/watch?v=hBZvQNcdOT4&list=PLjKdf6AHvR-G2KoSy1KW\\_1B9O3V8rUoSb&index=29](https://www.youtube.com/watch?v=hBZvQNcdOT4&list=PLjKdf6AHvR-G2KoSy1KW_1B9O3V8rUoSb&index=29)

t-SNE



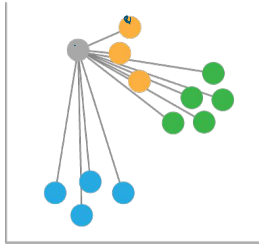
## t-SNE (t-distributed stochastic neighbor embedding)



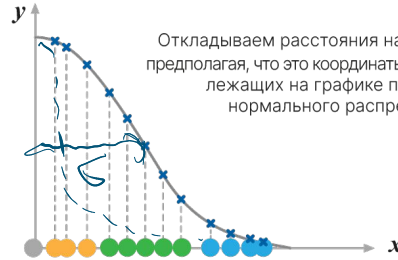
Идея состоит не в том, чтобы напрямую максимизировать дисперсию, а найти такое пространство в котором расстояние между объектами будет сохраняться или по крайней мере не сильно меняться. При этом будем больше беспокоиться о расстоянии между близкими объектами, нежели о расстоянии между далекими

# Описываем расстояния в исходном пространстве

1.



Считаем все расстояния от заданной точки до остальных



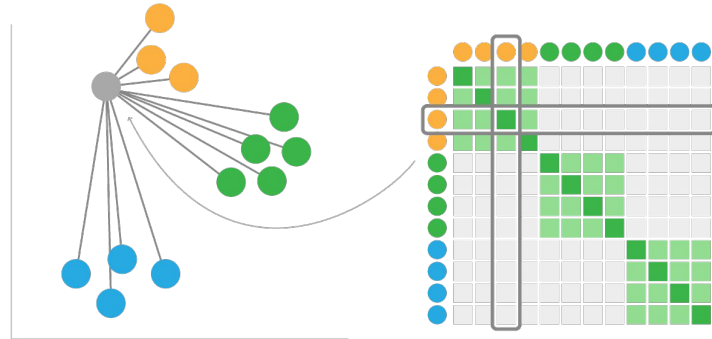
Откладываем расстояния на прямой, предполагая, что это координаты  $\mathbf{x}$  точек, лежащих на графике плотности нормального распределения

2.

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|_2 / 2\sigma_i^2)}$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

3.

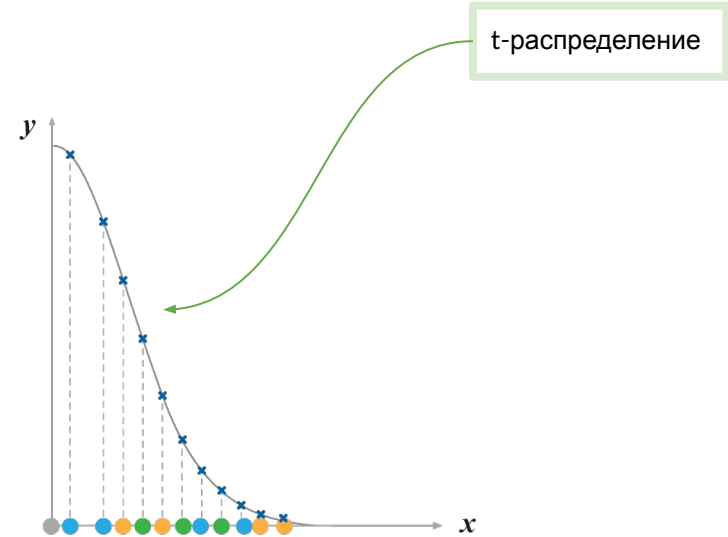


■ Высокая similarity  
■ Низкая similarity

# Описываем расстояния в пространстве низкой размерности



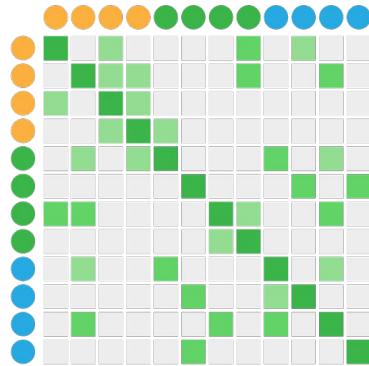
Снова считаем “похожести”...  
на этот раз назовем их не  $p$ , а  $q$



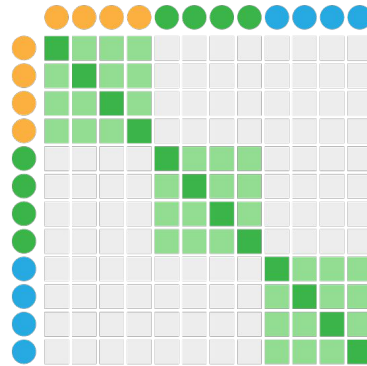
Но они же не похожи...

KL - дивергенция

k - кучность  
L - линейность



Матрица расстояний  
в пространстве низкой размерности



Матрица расстояний  
в пространстве высокой размерности

$$Loss = KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

L





# Важные параметры tSNE



## **perplexity**

Определяет то, как подбирается стандартное отклонение для распределения расстояний для каждой точки. Чем больше perplexity - тем более на глобальную структуру мы смотрим

## **metric**

Как считаются расстояния между точками - metric. По умолчанию используется евклидово расстояние, но часто помогают и другие (например, косинусное)

## **learning\_rate**

Шаг градиентного спуска, тоже влияет на полученное представление

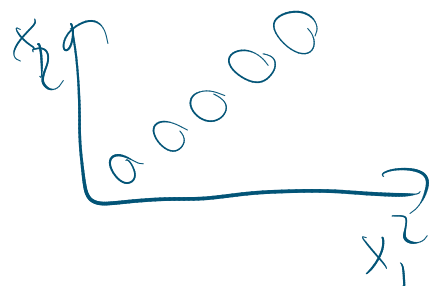
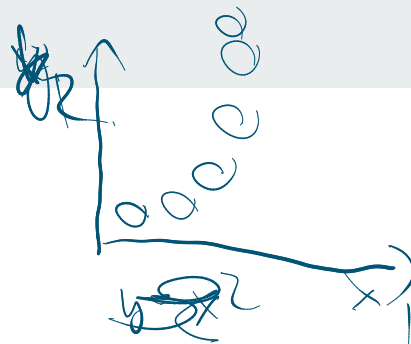
# Минусы tSNE



1. Стохастичность
2. Добавление новых точек
3. Расстояния между кластерами точек могут ничего не значить (плохо сохраняются далекие расстояния)
4. Размеры кластеров ничего не значат
5. Можно увидеть артефактные кластеры
6. Можно увидеть не ту структуру, которая по идее должна быть

# Tricks

1. Инициализация при помощи PCA
2. Kernel PCA



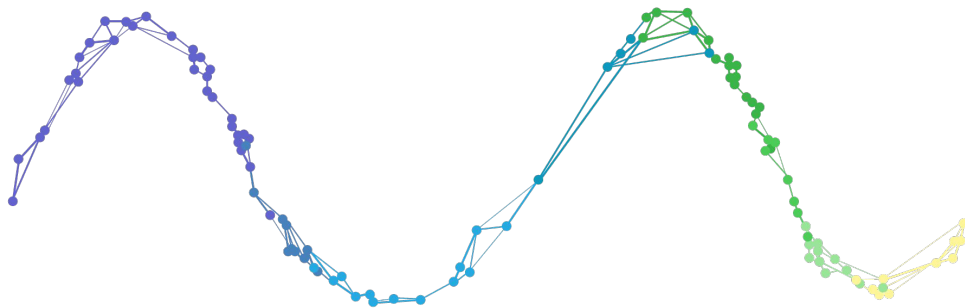
$$PC_1 = a \cdot x_1^2 + x_2$$

UMAP

# UMAP (uniform manifold approximation and projection)



Внутри себя метод строит граф, в котором ребрами соединены между собой  $k$  ближайших соседей. При этом эти ребра неравноправны - если для данной пары точек расстояние между ними сильно больше, чем расстояния между ними и другими точками - то и ребро будет иметь маленький вес.



Далее задача состоит в том, чтобы в пространстве более низкой размерности получился граф похожий на тот, который был в высокой размерности. Для этого опять же, оптимизируем низкоразмерное представление градиентным спуском

# Плюсы



1. Быстрее чем tSNE
2. Можно добавлять новые данные

**Отдыхайте**





## Источники

1. [Учебник по машинному обучению ШАД](#)
2. [k-means++: The Advantages of Careful Seeding](#)