

«Машинное обучение»

Методы кластеризации

Александр Дьяконов



План

Задача кластеризации, типы кластеризации

k-средних (Lloyd's algorithm) + обобщения

Иерархическая кластеризация (Hierarchical clustering)

DBSCAN = Density-Based Spatial Clustering of Applications with Noise

Сравнение алгоритмов кластеризации

Кластеризация

- разбиение множества объектов на группы похожих, такие группы – кластеры

неформально:

маленькие внутрикластерные расстояния
большие межкластерные расстояния

Самое важное и «скользкое»

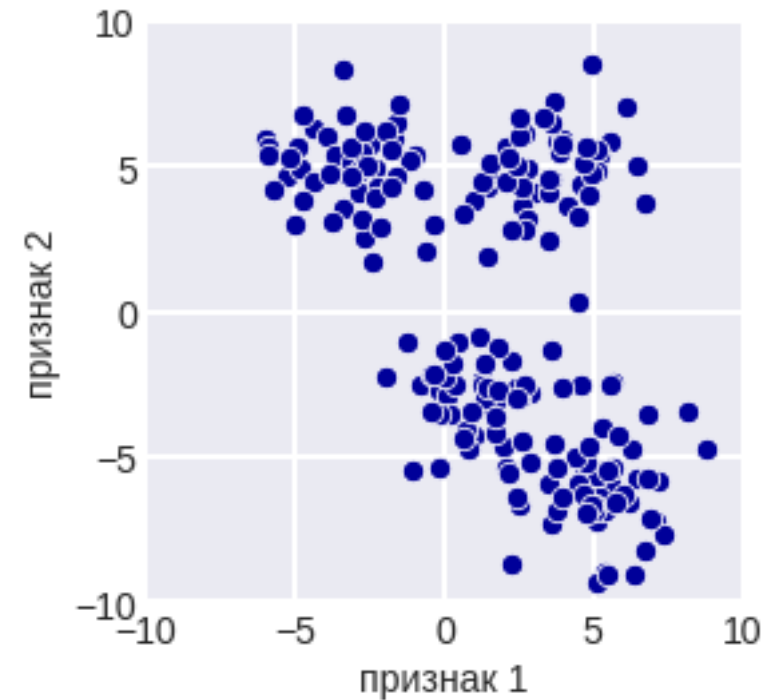
дальше предполагаем, что есть некоторая адекватная метрика!

С помощью её и будем осуществлять кластеризацию.

Входная информация для алгоритмов

- 1. (Feature-based) Признаковые описания объектов**
- 2. (Dis/similarity-based) Попарные сходства/различия**

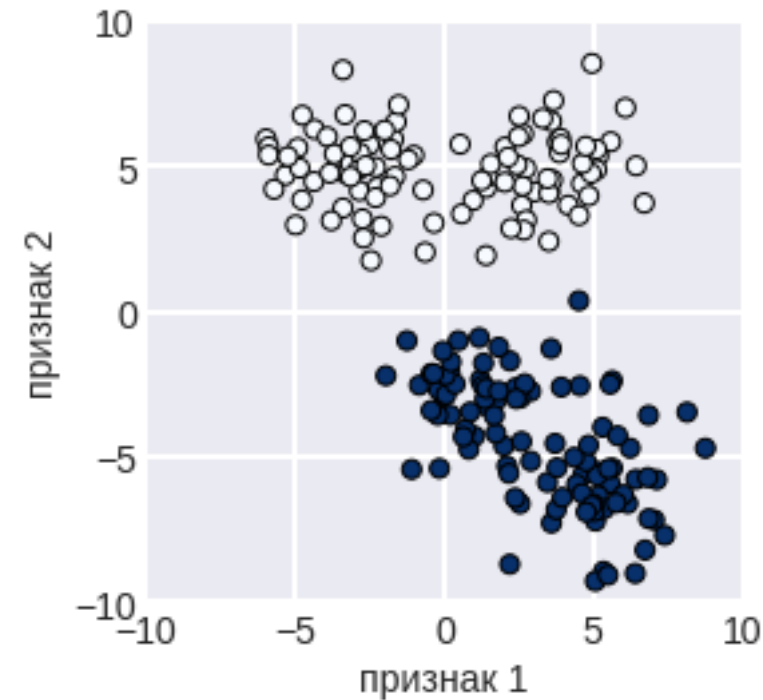
Модельная задача кластеризации



Даны объекты (без меток)

Надо разбить на группы похожих – кластеры

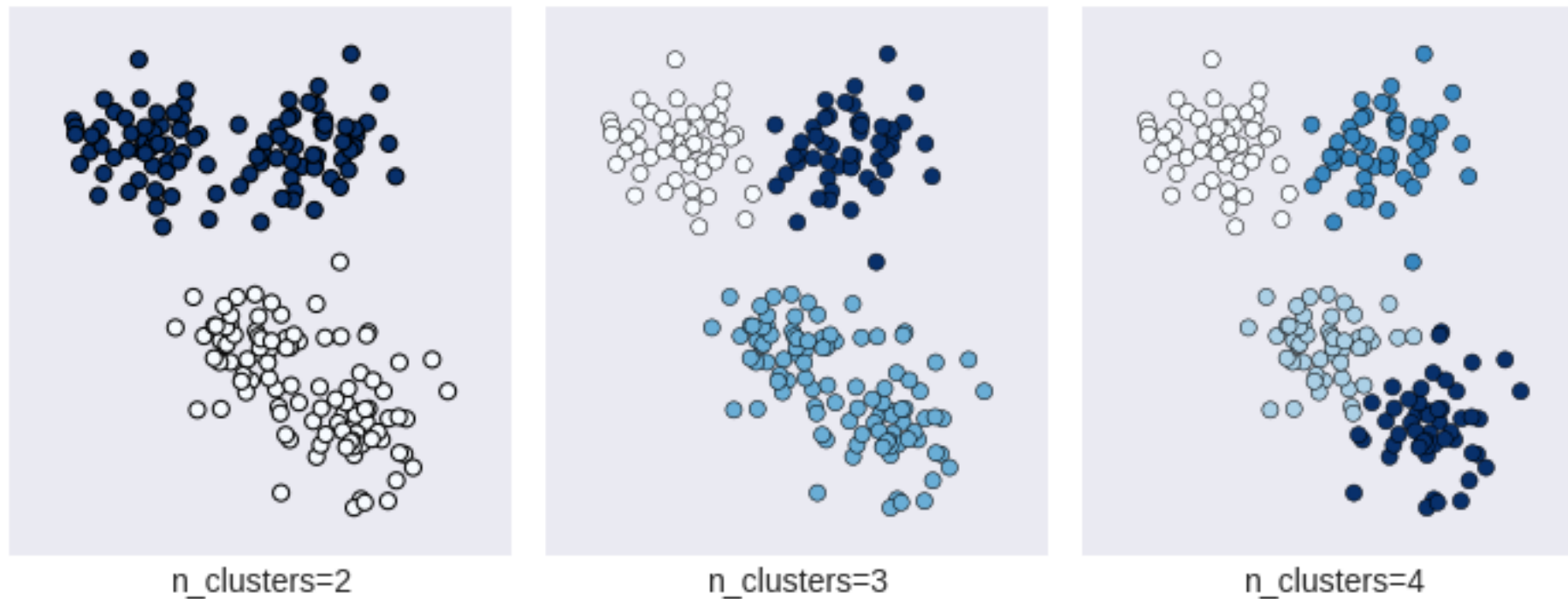
Модельная задача кластеризации



Даны объекты (без меток)

Надо разбить на группы похожих – кластеры

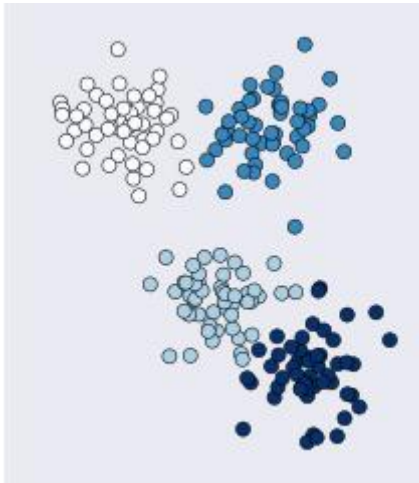
Модельная задача кластеризации



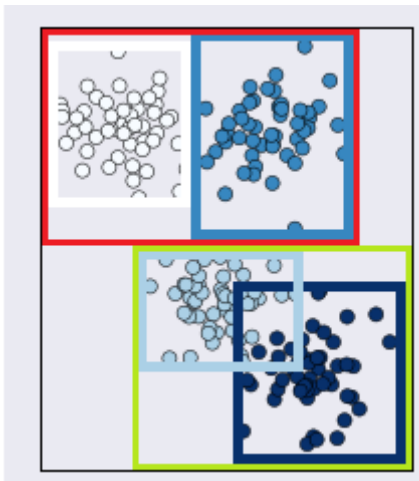
Много допустимых решений...

Нет правильного ответа!

Методы кластеризации



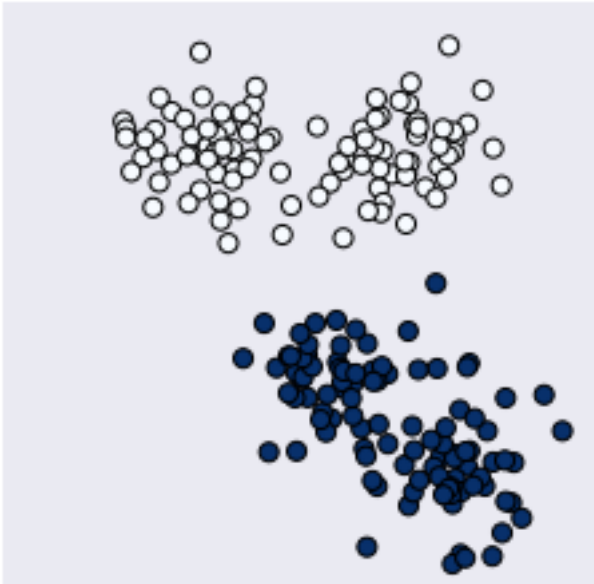
Плоские / разделяющие (Flat / Partitional) –
кластеризация на k непересекающихся кластеров



Иерархические (Hierarchical) – данные один большой кластер, далее рекурсивно «кластер = объединение подкластеров»

~ система каталогов

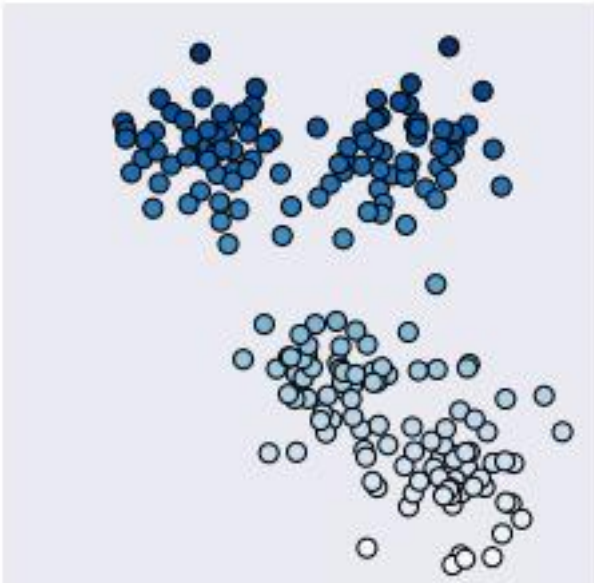
Типы кластеризации



чёткая (hard)

разбиение на непересекающиеся кластеры

$$\mathbb{X} = C_1 \cup \dots \cup C_k$$
$$i \neq j \Rightarrow C_i \cap C_j = \emptyset$$



нечёткая (мягкая, fuzzy)

определение степени принадлежности кластерам

$$x_i \rightarrow r_{it} \in [0, 1]$$
$$\forall i \in \{1, 2, \dots, m\} \sum_{t=1}^k r_{it} = 1$$

Зачем

- **кластеризация клиентов / товаров, иерархия сущностей (таксономия)**
(выработка таргетированной политики)

- **сжатие данных**

(при модерации проверять несколько представителей кластера, устранение однотипности вопросов, vector quantization – замена кластера центром)

- **сообщества клиентов**

(эффективное распространение новостей, предложение услуг)

- **анализ данных / признаков**

(классическая идея математики – факторизация)

- **важная составляющая решения других задач**

(semi-supervised, outlier detection, community detection)

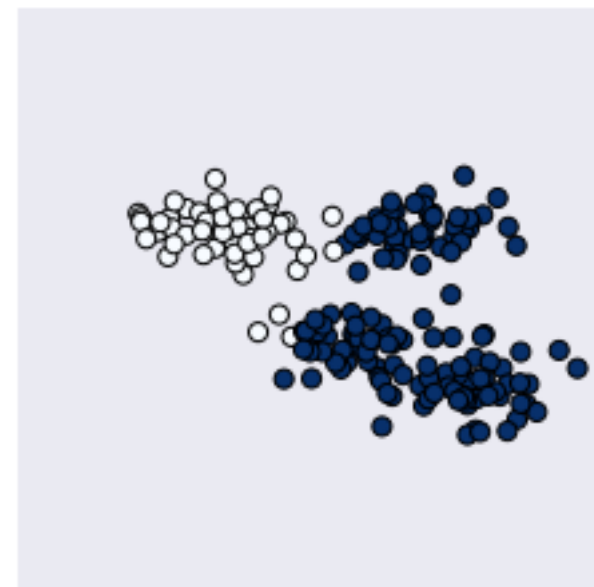
была в методах сэмплирования и много где ещё...

Редко бывает самостоятельной задачей! Но важный этап

Проблемы с расстоянием



сжали по одному признаку



сжали по второму признаку

Часто используют стандартизацию, но она может всё портить...

В кластеризации считаем, что метрика адекватна

k-средних (Lloyd's algorithm)**Вход:**

$$\{x_1, \dots, x_m\} \subseteq \mathbf{R}^n$$

Инициализация:

$$k \text{ центров кластеров } \{\mu_1, \dots, \mu_k\} \subseteq \mathbf{R}^n$$

случайные точки или случайные объекты
из обучающей выборки

Итерация:

повторять итерации до сходимости (пока
центры станут неподвижными)
«assignment»

1. Каждый объект приписать к тому кластеру, к центру которого он ближе:

$$C_t = \{i \mid \|x_i - \mu_t\| = \min_j \|x_i - \mu_j\|\}$$

Ничьи разрешаются произвольно

2. Пересчитать центры кластеров:

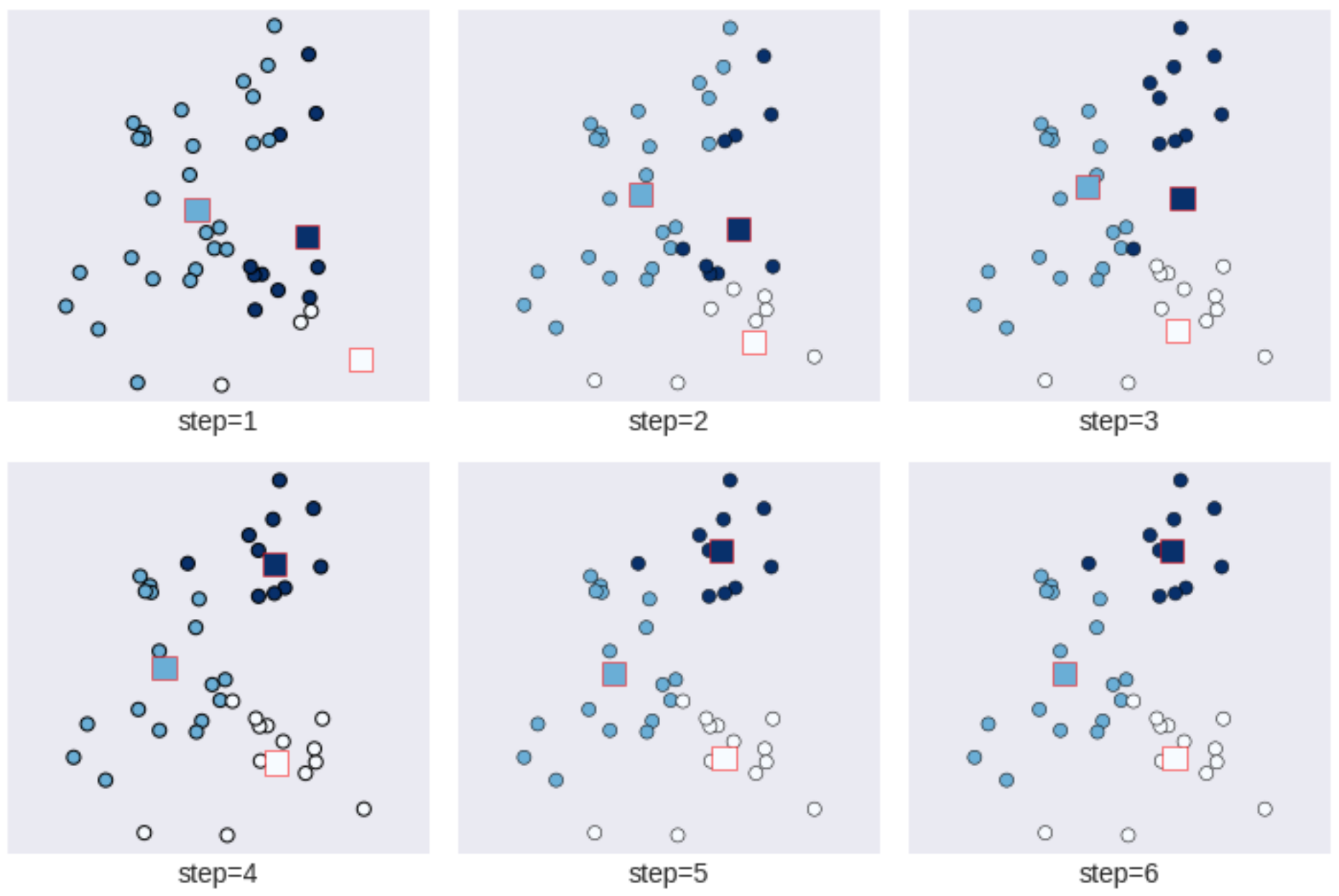
$$\mu_t = \frac{1}{|C_t|} \sum_{i \in C_t} x_i$$

«update»

при неопределённости вида «деление на ноль» оставляем центр неизменным

метод ~ координатный спуск

Итерации метода к-средних



Код

```
# запуск алгоритма
from sklearn.cluster import KMeans
model = KMeans(n_clusters=8,      # число кластеров
               init='k-means++',  # метод инициализации: «random», ndarray
               n_init=10,         # сколько раз алгоритм запускается
                                # (выбирается лучший ответ)
               max_iter=300,      # максимальное число итераций для каждого алгоритма
               tol=0.0001,        # порог для определения сходимости
               verbose=0,
               random_state=None,
               copy_x=True, # данные модифицируются и центрируются...
               algorithm='lloyd') # какой алгоритм использовать: «elkan» (использует
                                # неравенство треугольника, но не подходит для
                                # разреженных данных)

a = model.fit_predict(X)

# визуализация
import matplotlib.pyplot as plt
plt.scatter(X[:, 0], X[:, 1], c=a)
```

Разная начальная инициализация



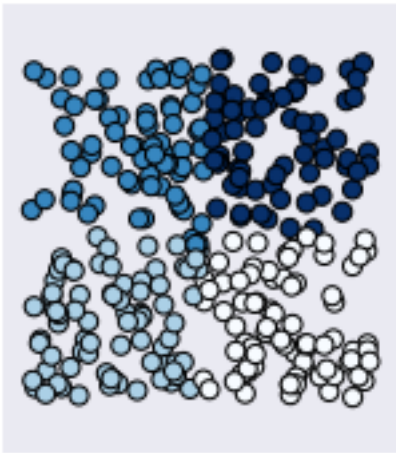
random_state=0



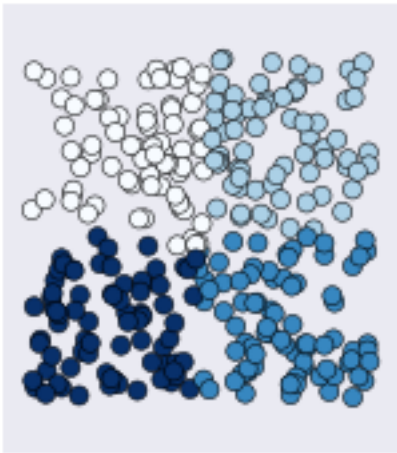
random_state=1



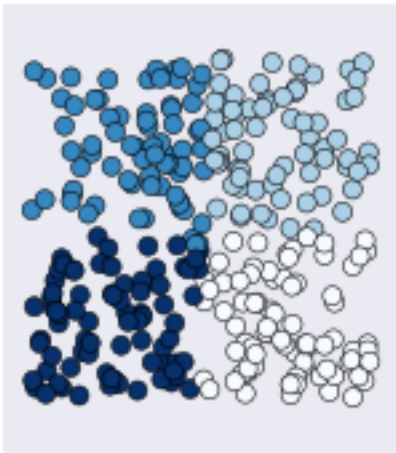
random_state=2



random_state=0

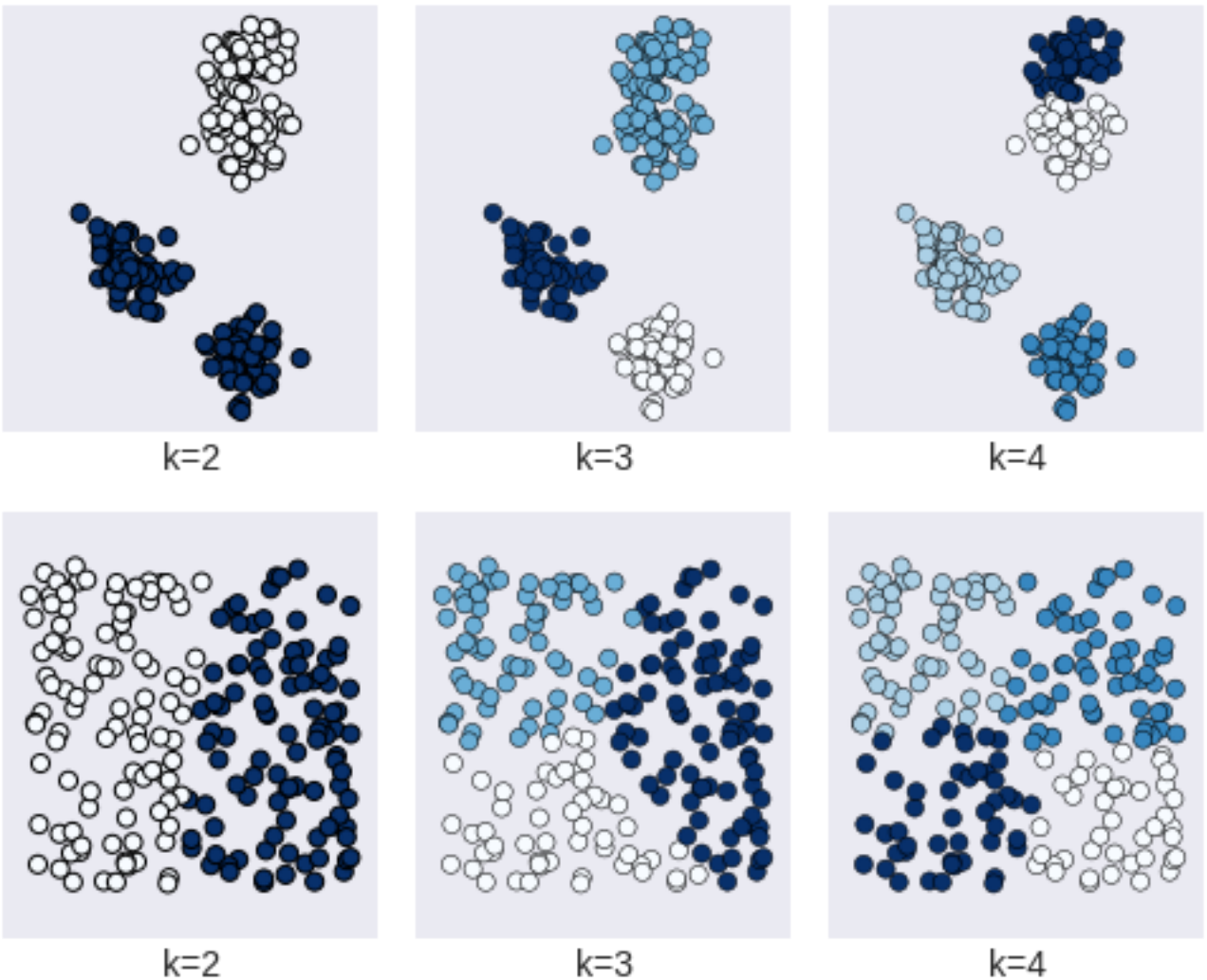


random_state=1

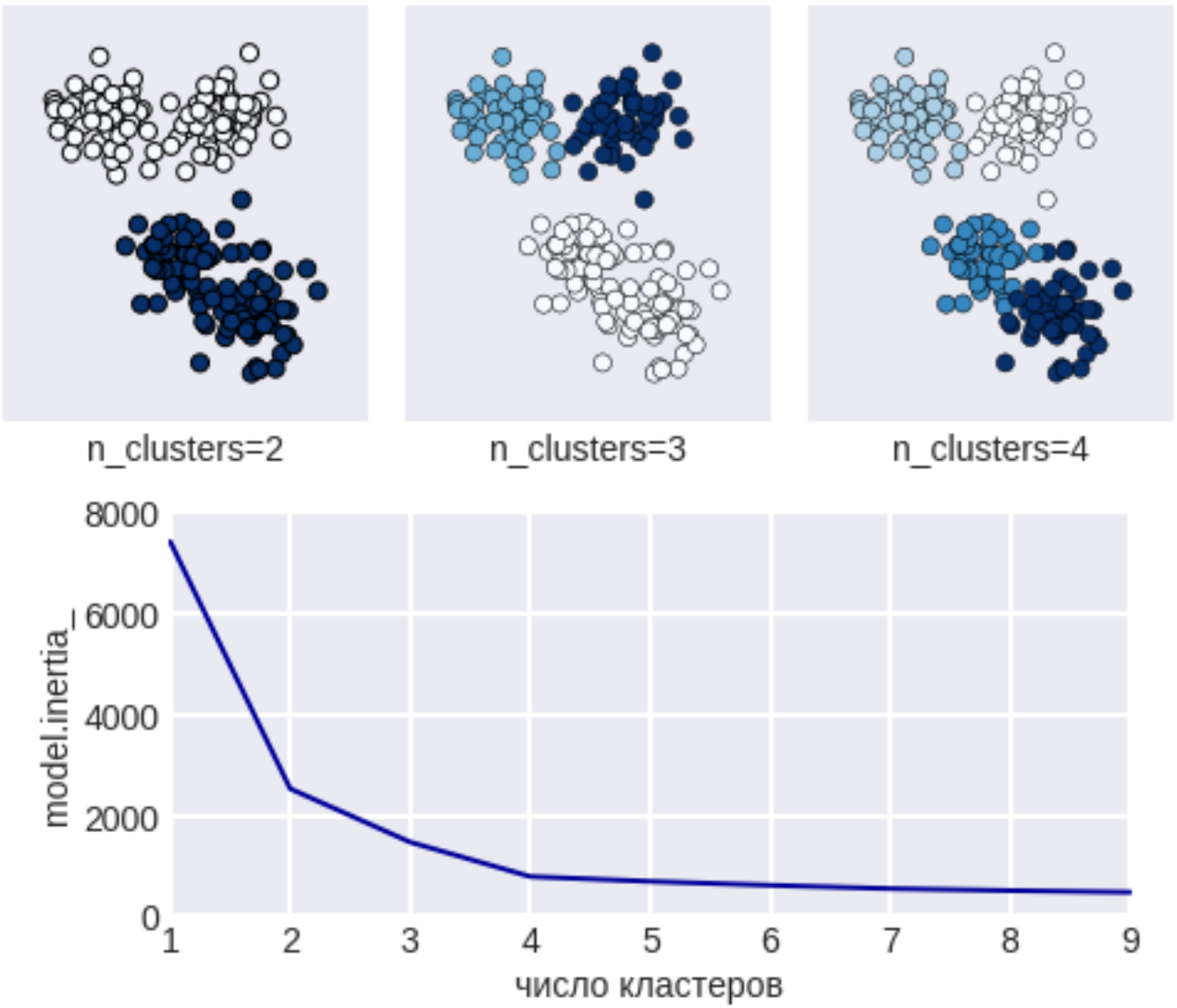


random_state=2

Разное число кластеров k



Вариант для выбора k



выбираем лучшую инициализацию из 5

Сходимость метода

– теоретически может сходиться долго, но на практике быстро

средняя сложность $O(kmn_{\text{iter}})$

худший случай $O(m^{(k+2/n)})$

D. Arthur, S. Vassilvitskii «How slow is the k-means method?» SoCG2006

– сходимость к локальному минимуму

– чувствителен к начальной инициализации (могут получаться разные ответы):

- качество кластеризации
- скорость сходимости

– нужен выбор k

Реализация на практике и обобщения

несколько разных инициализаций и перезапусков

эвристика: 1й центр – один из объектов, 2й – максимально удалённый от первого, 3й – максимально удалённый от предыдущих центров и т.д.

на самом деле тут вероятностный выбор, поэтому разные перезапуски
– **k-means++**

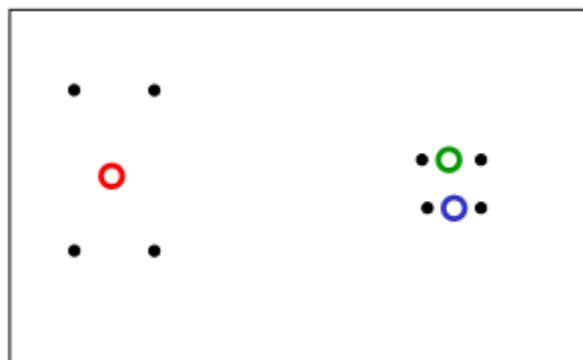
Fuzzy / soft k-means – вместо чёткой принадлежности – нечёткая **дальше**

k-medians (medoids) – другие усреднения (для борьбы с выбросами)

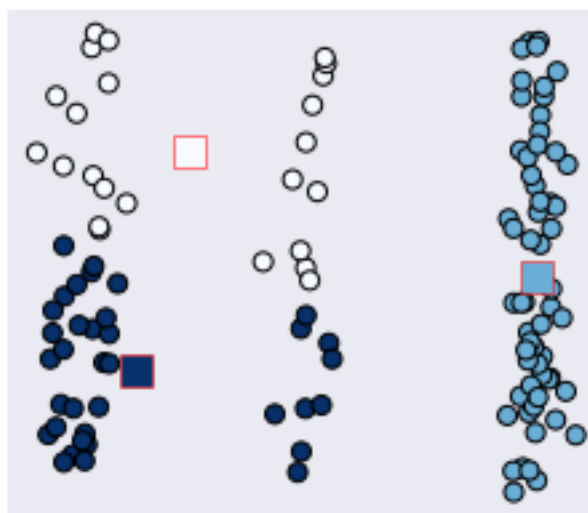
k-means + kernel tricks

k-means: ограничения

- хорошо работает для примерно равномогущих кластеров с «шаровой формой»
будут рисунки, потом поймём почему (GMM)
- оптимизируемый функционал не выпуклый



много локальных
минимумов

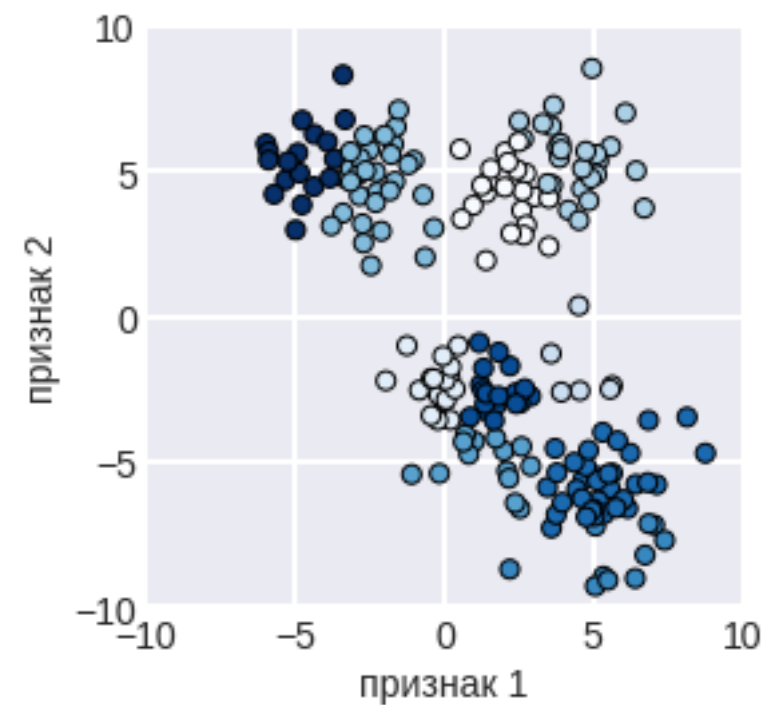
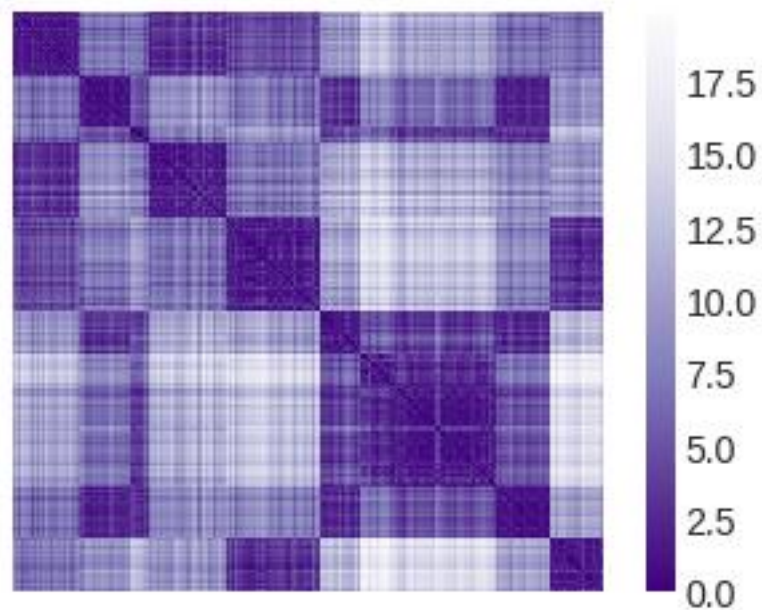


Иллюстрация

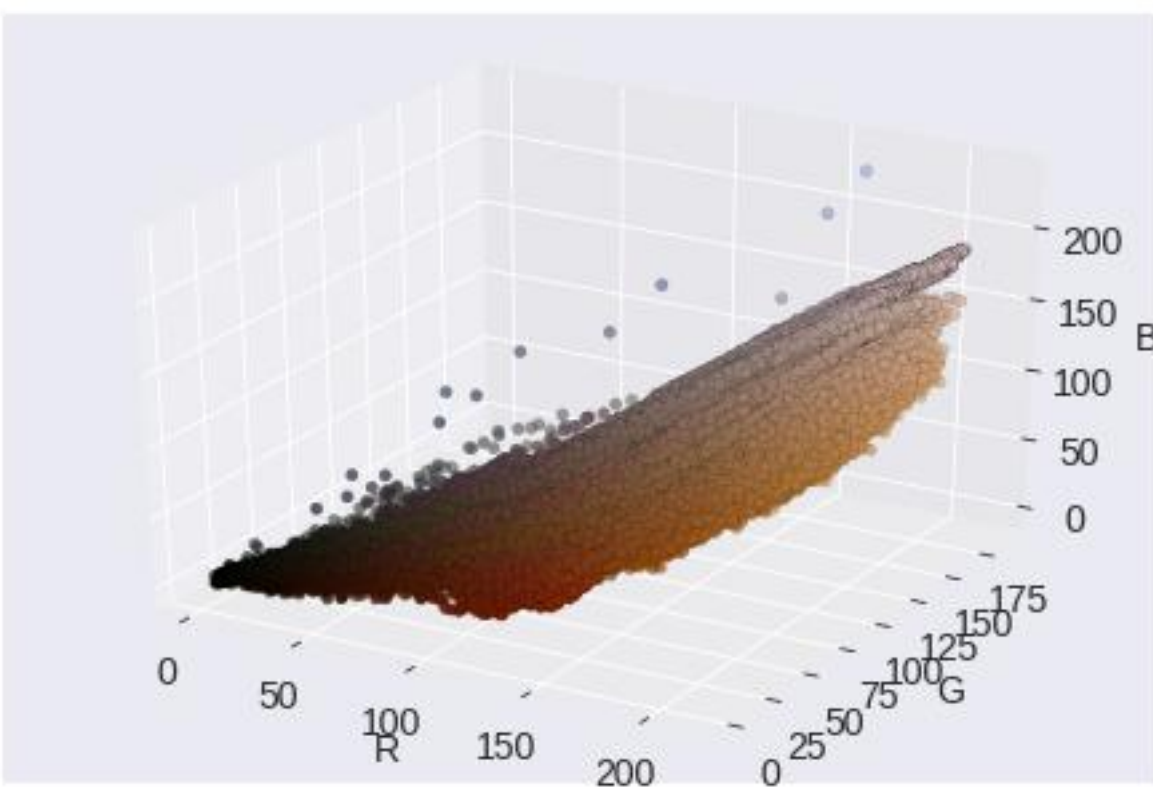
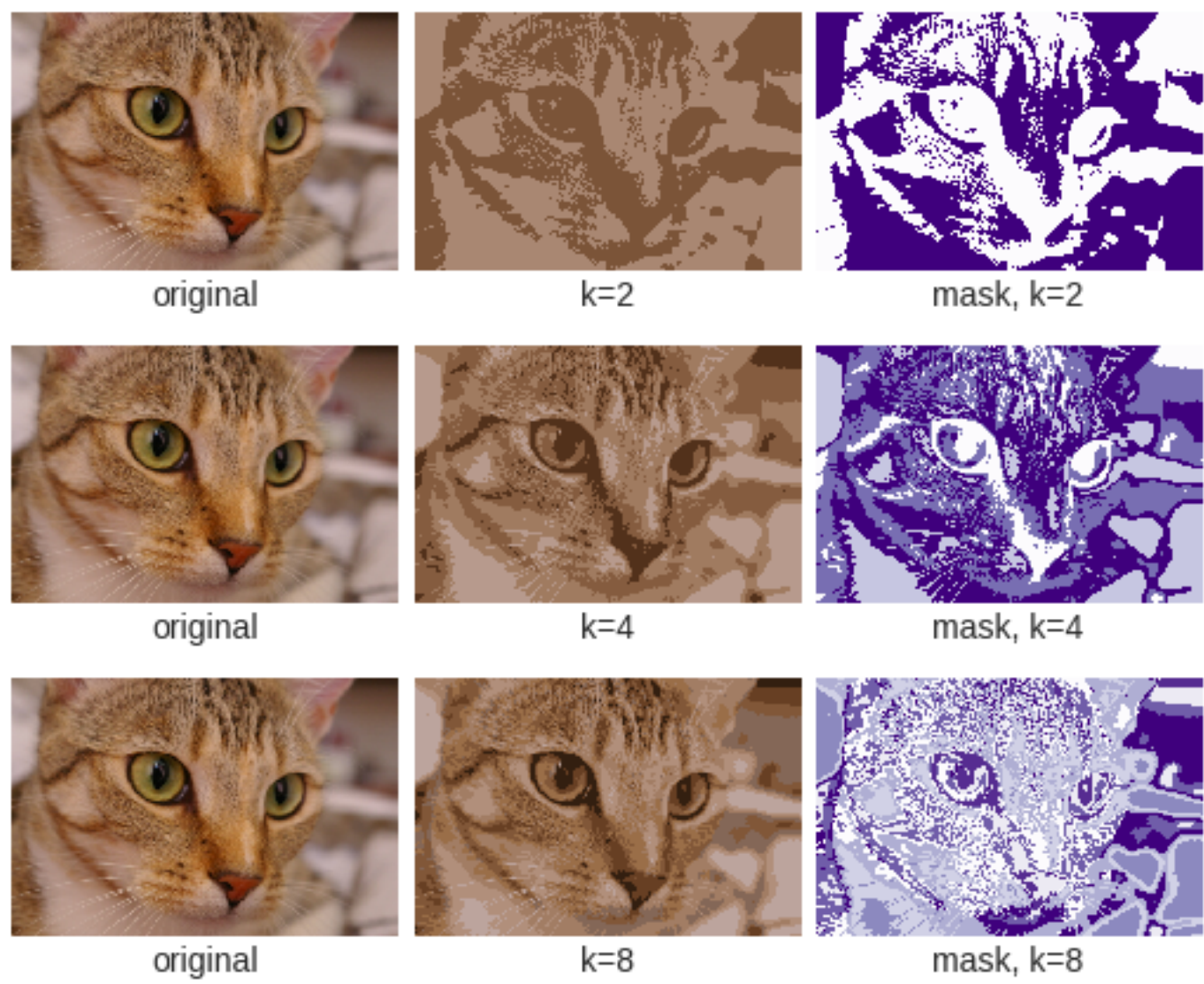
Матрица попарных расстояний до и после кластеризации
(объекты упорядочены по кластерам)



Иллюстрация



Пример k-means: сегментации в изображениях



Не надо хранить все цвета, а только k – хорошее сжатие

Пример k-means: сегментации в изображениях

```
def simplify_image(image, k=10):  
    image2 = image.copy()  
    image2.resize([image.shape[0]*image.shape[1], 3])  
    model = KMeans(n_clusters=k, random_state=11)  
    model.fit(image2)  
    result = model.cluster_centers_.round().astype(int)[model.labels_, :]  
    result.resize([image.shape[0], image.shape[1], 3])  
    mask = model.labels_  
    mask.resize([image.shape[0], image.shape[1]])  
    return result, mask
```



original



k=2



k=4

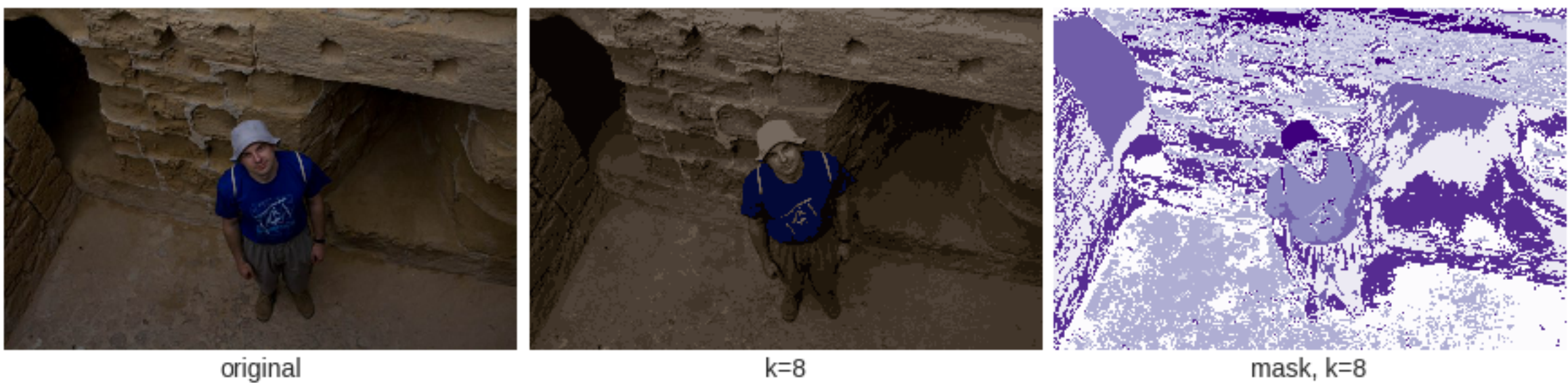
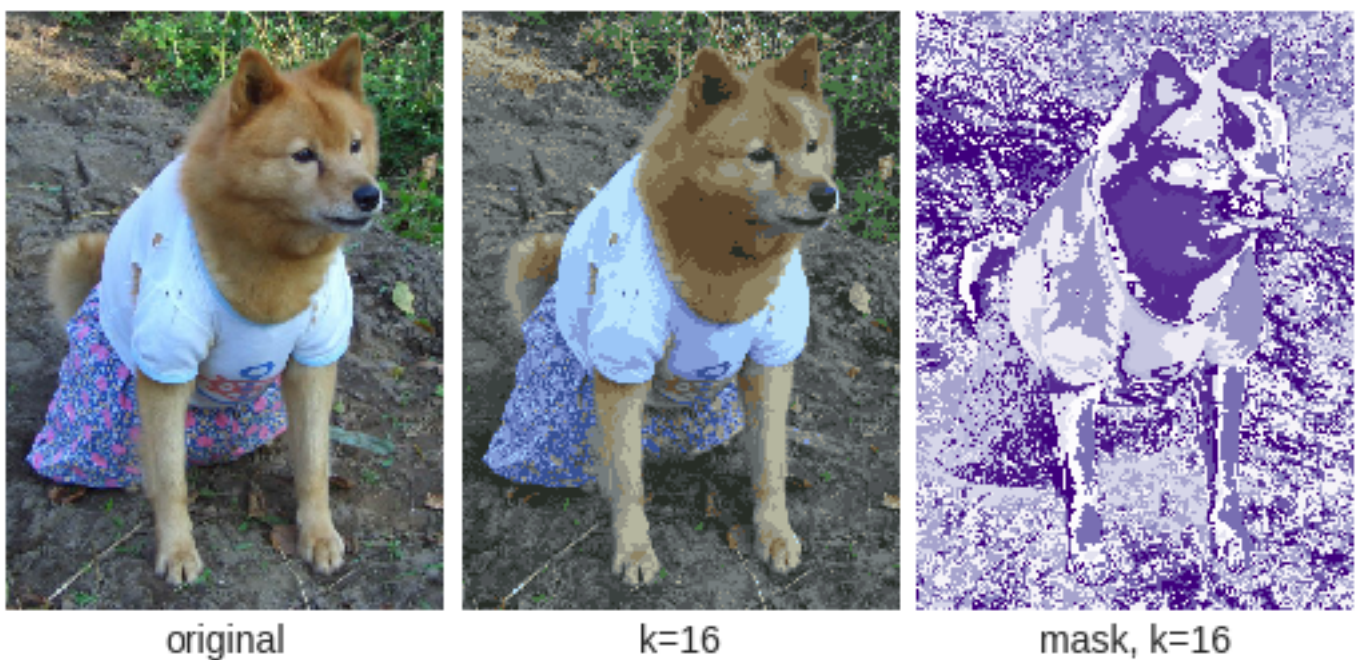


k=8



k=16

Пример k-means: сегментации в изображениях



k-Means – немного перепишем алгоритм

Вход:

$$\{x_1, \dots, x_m\} \subseteq \mathbf{R}^n$$

Изменения:

Инициализация:

k центров кластеров $\{\mu_1, \dots, \mu_k\} \subseteq \mathbf{R}^n$

Итерация:

1. Каждый объект приписать к тому кластеру, к центру которого он ближе:

$$C_t = \{i \mid \|x_i - \mu_t\| = \min_j \|x_i - \mu_j\|\}$$

2. Пересчитать центры кластеров:

$$\mu_t = \frac{1}{|C_t|} \sum_{i \in C_t} x_i$$

$$r_{it} = I[\|x_i - \mu_t\| = \min_s \|x_i - \mu_s\|]$$

считаем, что минимум единственный

$$\mu_t = \frac{1}{\sum_{i=1}^m r_{it}} \sum_{i=1}^m r_{it} x_i$$

soft k-Means**Вход:**

$$\{x_1, \dots, x_m\} \subseteq \mathbf{R}^n$$

Изменения:**Инициализация:**

k центров кластеров $\{\mu_1, \dots, \mu_k\} \subseteq \mathbf{R}^n$

Итерация:

1. Для каждого объекта – оценку принадлежности к каждому кластеру

$$r_{it} = \frac{\exp(-\beta \|x_i - \mu_t\|)}{\sum_j \exp(-\beta \|x_i - \mu_j\|)}$$

2. Пересчитать центры кластеров:

$$\mu_t = \frac{1}{\sum_{i=1}^m r_{it}} \sum_{i=1}^m r_{it} x_i$$

soft k-Means

+ мягкая кластеризация

+ иногда работает лучше

– выбор параметра «бета»

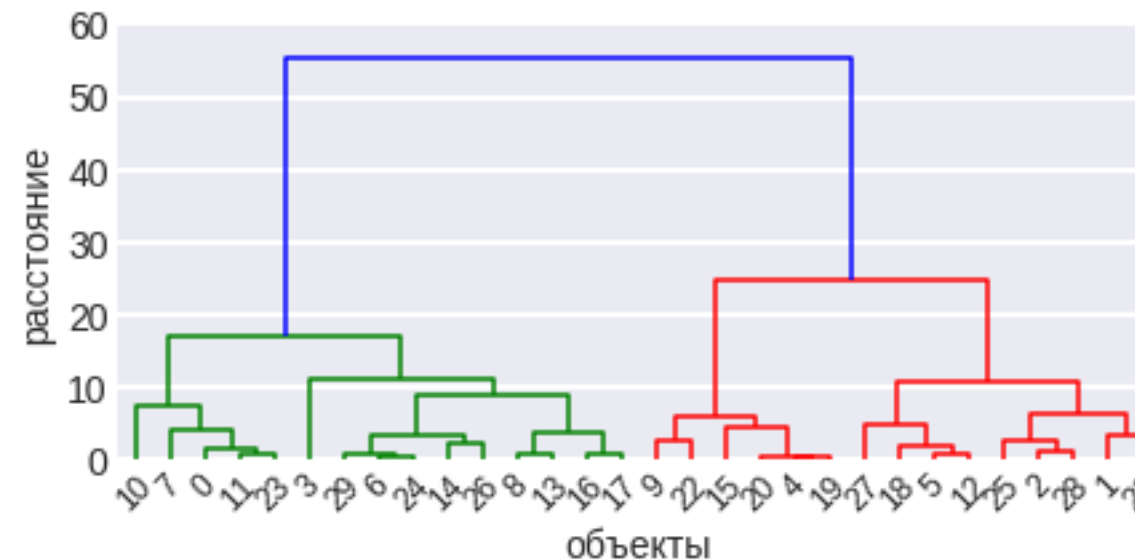
– остались проблемы

**«продолговатые кластеры»
кластеры разной мощности**

<http://qaru.site/questions/611120/minibatchkmeans-parameters>

Иерархическая кластеризация (Hierarchical clustering)

ИК в отличие от плоской (Flat Clustering) получает серию вложенных друг в друга кластеризаций (\Rightarrow не требует заранее заданного числа кластеров)



ИК может быть долгой...

https://en.wikipedia.org/wiki/Hierarchical_clustering

Два подхода к иерархической кластеризации

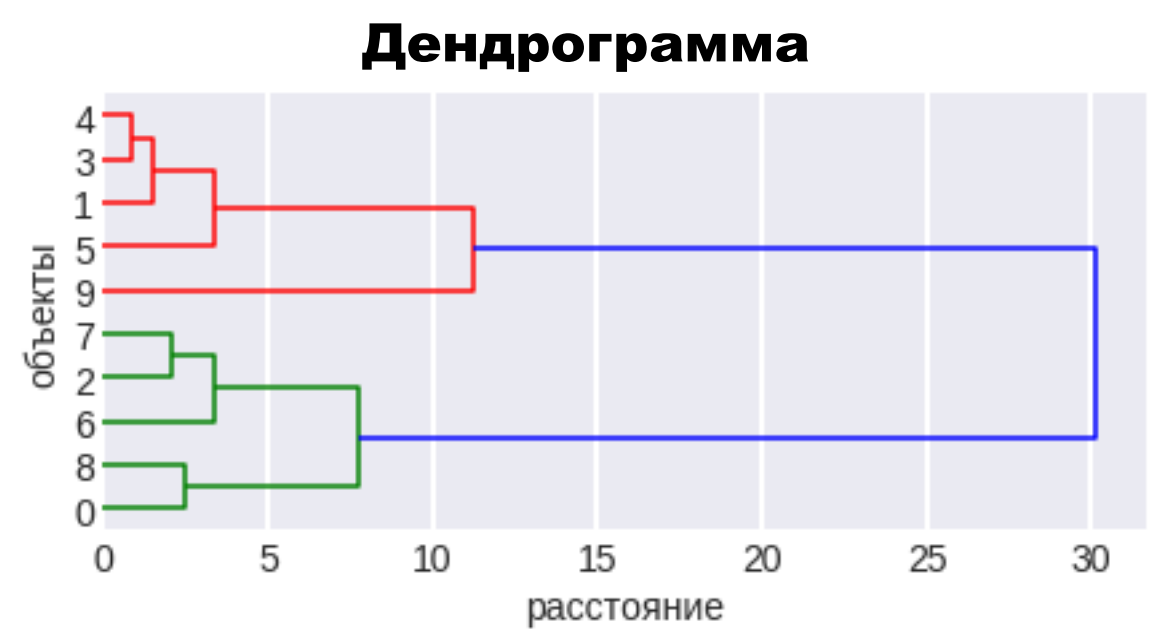
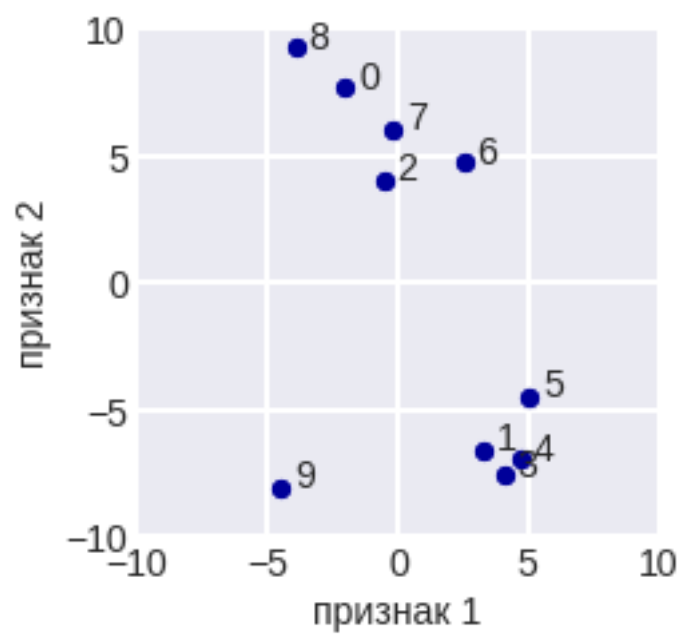
Агломеративный (Agglomerative / bottom-up)

- изначально число кластеров совпадает с числом объектов, каждый кластер содержит один объект обучения
- на каждом шаге объединяем два наиболее похожих кластера
- останавливаемся, если остаётся один кластер
- чаще используют (проще реализовать)

Дивизионный (Divisive / top-down)

- изначально есть один кластер, содержащий все объекты обучения
- расщепить кластер **с максимальными внутрикластерными расстояниями** на два
- останавливаемся, когда число кластеров совпадает с числом объектов

Визуализация иерархической кластеризации



Корень дерева – кластер, который содержит все объекты

Визуализация иерархической кластеризации

```
from scipy.cluster.hierarchy import dendrogram
from sklearn.cluster import AgglomerativeClustering

def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram
    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1 # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack([model.children_,
                                     model.distances_,
                                     counts]).astype(float)

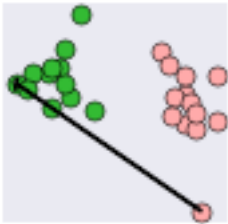
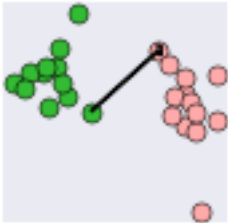

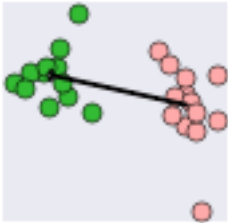
    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs, orientation = 'right')
    return linkage_matrix
```

```
model = AgglomerativeClustering(
    distance_threshold=0,
    n_clusters=None)

model = model.fit(X)

plot_dendrogram(model,
    truncate_mode='level',
    p=5,
    leaf_font_size=14)
```

Как измерить расстояния между кластерами – типы Linkage

Linkage		Описание
Complete		Maximal inter-cluster dissimilarity $\max_{x \in A, x' \in B} \text{dis}(x, x')$
Single		Minimal inter-cluster dissimilarity $\min_{x \in A, x' \in B} \text{dis}(x, x')$
Average		Mean inter-cluster dissimilarity $\frac{1}{ A \cdot B } \sum_{x \in A, x' \in B} \text{dis}(x, x')$
Centroid		Dissimilarity between the centroid for cluster A and B $\text{dis}(\bar{x}_A, \bar{x}_B)$

Как измерить расстояния между кластерами – типы Linkage

Расстояние Варда (Ward's measure)

изменение суммы квадратов

$$\lambda(X) = \sum_{x \in X} \|x - \bar{X}\|^2$$

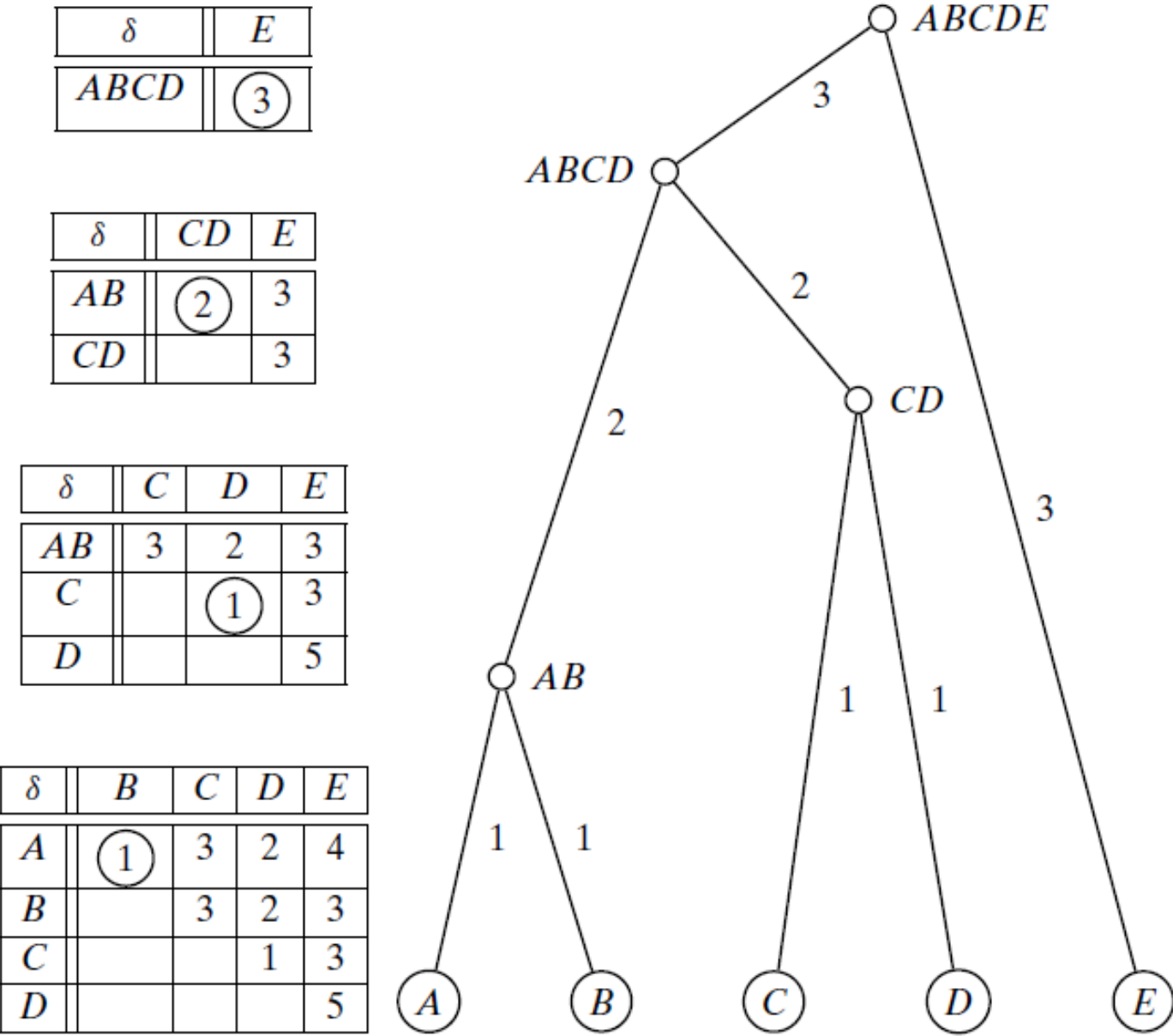
после объединения кластеров:

$$\lambda(A \cup B) - \lambda(A) - \lambda(B) = \frac{1}{\frac{1}{|A|} + \frac{1}{|B|}} \|\bar{A} - \bar{B}\|^2$$

Можно доказать...

https://en.wikipedia.org/wiki/Hierarchical_clustering

Single Link



Lance–Williams formula

**«Lance–Williams formula» при объединении кластеров
не будем рассматривать**

$$\delta(C_{ij}, C_r) = \alpha_i \cdot \delta(C_i, C_r) + \alpha_j \cdot \delta(C_j, C_r) + \\ \beta \cdot \delta(C_i, C_j) + \gamma \cdot |\delta(C_i, C_r) - \delta(C_j, C_r)|$$

Measure	α_i	α_j	β	γ
Single link	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete link	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
Group average	$\frac{n_i}{n_i + n_j}$	$\frac{n_j}{n_i + n_j}$	0	0
Mean distance	$\frac{n_i}{n_i + n_j}$	$\frac{n_j}{n_i + n_j}$	$\frac{-n_i \cdot n_j}{(n_i + n_j)^2}$	0
Ward's measure	$\frac{n_i + n_r}{n_i + n_j + n_r}$	$\frac{n_j + n_r}{n_i + n_j + n_r}$	$\frac{-n_r}{n_i + n_j + n_r}$	0

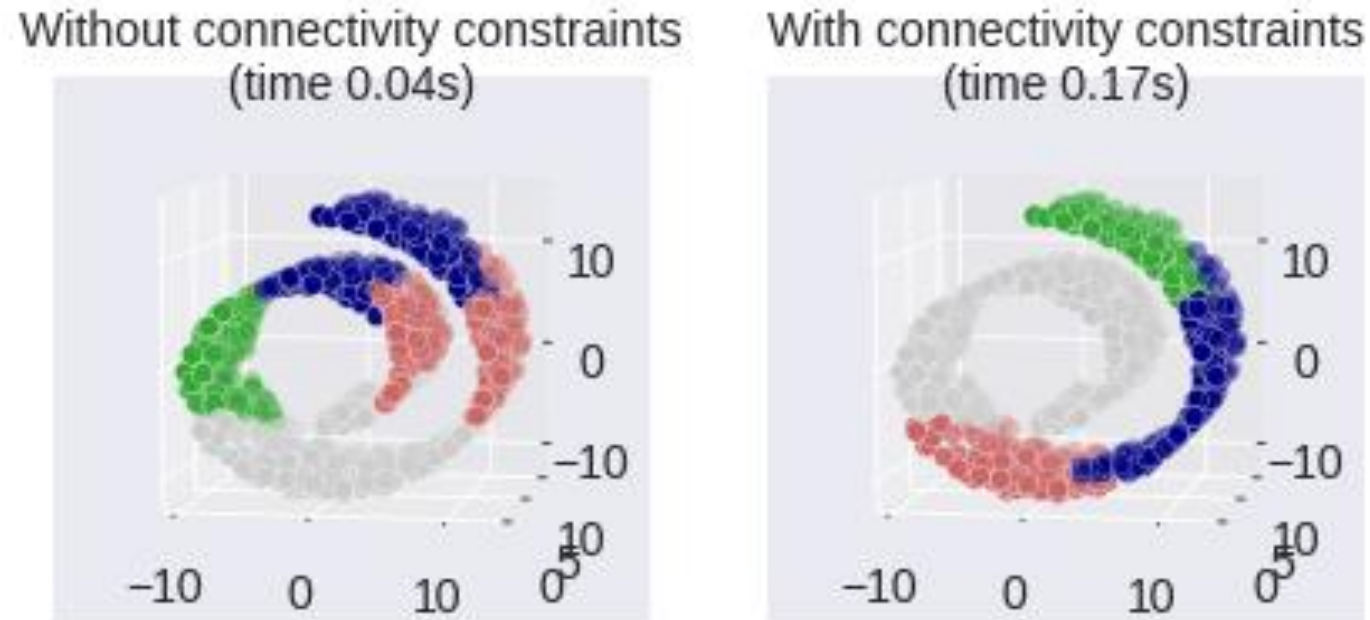
https://en.wikipedia.org/wiki/Ward's_method

Агломеративная кластеризация

```
clustering = AgglomerativeClustering(n_clusters=2, # число кластеров
    affinity='deprecated',
    metric=None, # «euclidean», «l1», «l2», «manhattan», «cosine»,
                # «precomputed»
    memory=None, # кэшировать ли память
    connectivity=None, # матрица связности (матрица или функция), чтобы только
                      # соседние кластеры объединялись, по умолчанию нет
    compute_full_tree='auto', # остановить построение дерева
                             # при достижении числа кластеров
    linkage='ward', # критерий слияния: «ward», «complete», «average»,
                  # «single»
    distance_threshold=None, # порог между кластерами - до которого они сливаются
    compute_distances=False).fit(X) # вычислять расстояния между кластерами
                                # (даже после достижения порога)

clustering.labels_
array([1, 1, 1, 0, 0, 0])
```

Использование матрицы связности



матрица k-соседства

Агломеративная кластеризация для сокращения признакового пространства

```
from sklearn import cluster
agglo = cluster.FeatureAgglomeration(n_clusters=32)
agglo.fit(X)
X_reduced = agglo.transform(X)
X.shape, X_reduced.shape
(1797, 64), (1797, 32)
```



original



k=16



mask, k=16

Кластеризация пикселей по признакам (R, G, B)



original



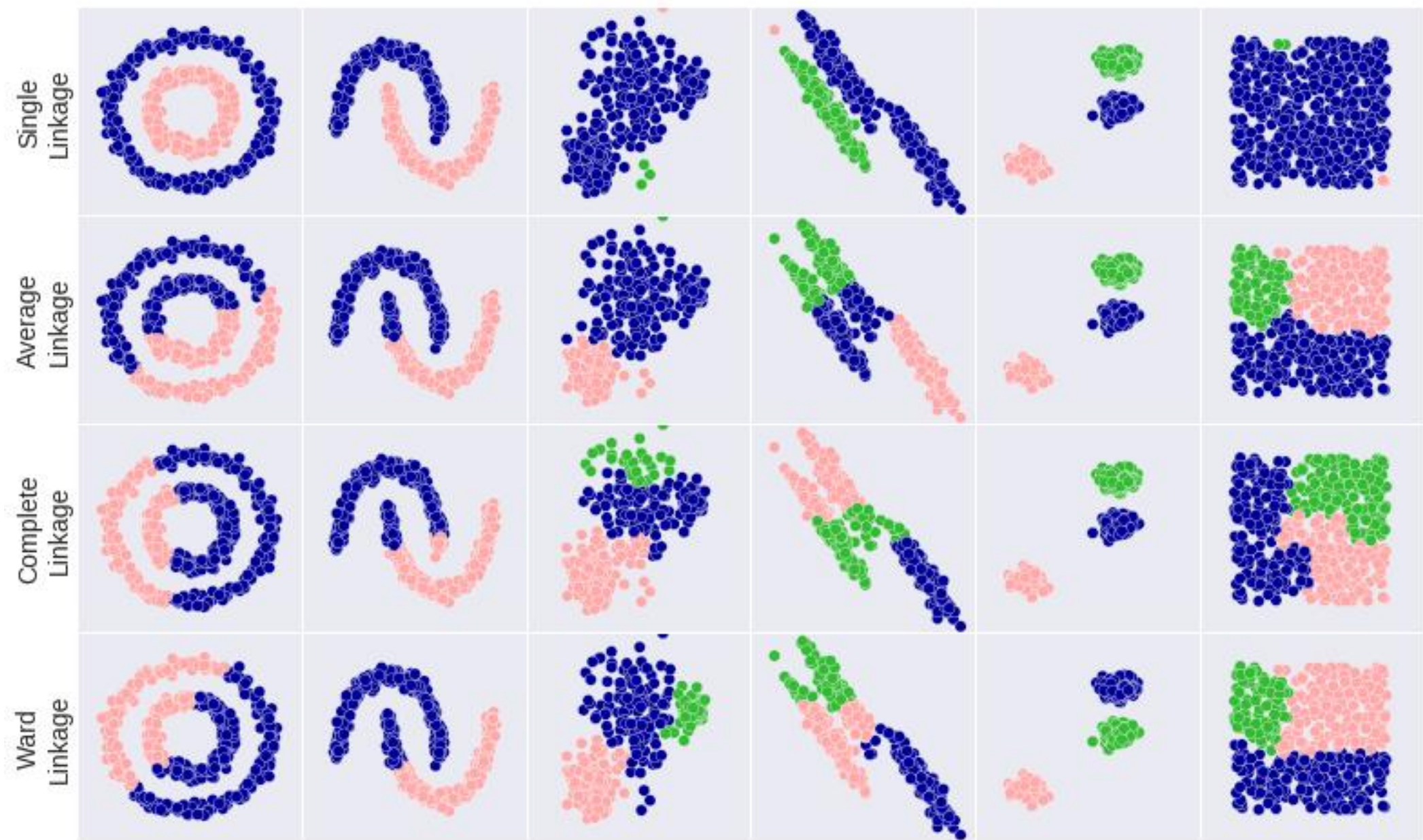
k=16



mask, k=16

Сегментации в изображениях по признакам (R, G, B) по графу 4-соседства

Разные виды связности



DBSCAN = Density-Based Spatial Clustering of Applications with Noise

Алгоритм с маркировкой точек как шум и т.п.

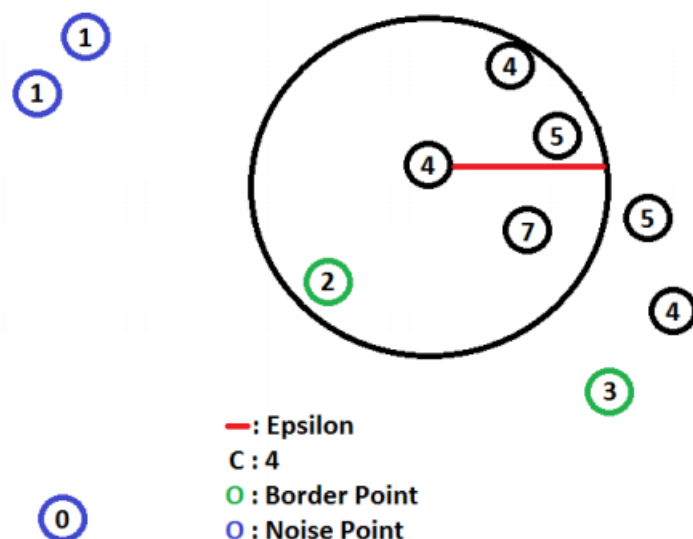
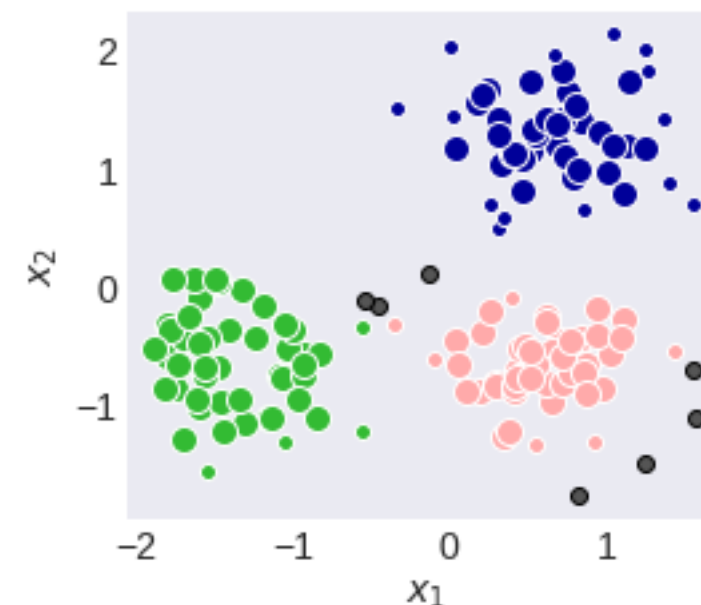
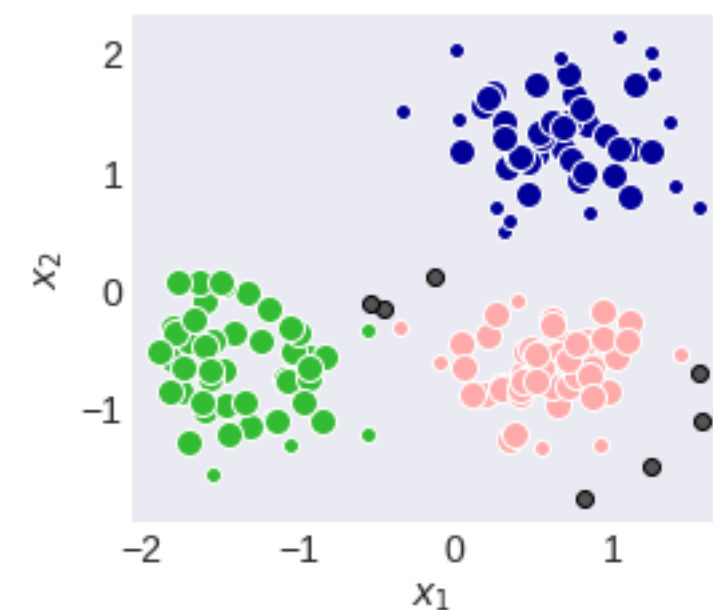
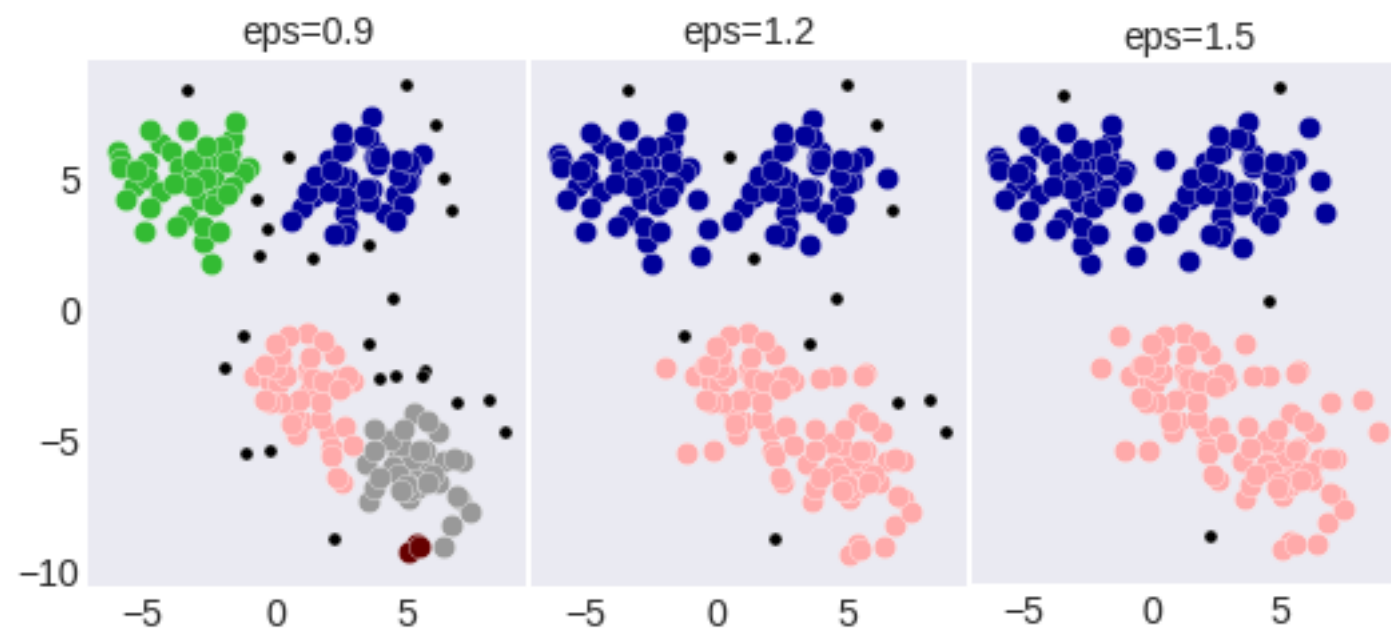


Figure 2: Noise Points in DBSCAN



- для каждой точки строим круг радиуса ε
- если в круге есть как минимум k точек – она плотная / основная (core point)
- если нет, но есть плотные точки – она граничная / достижимая (border point)
иногда – если есть путь точек на расстоянии $< \varepsilon$ до основной точки
- иначе – шум (noise point)
- множество основных точек – кластеризовать SingleLink
с пороговым расстоянием $\geq \varepsilon$
- множество граничных точек – к соответствующим кластерам

DBSCAN = Density-Based Spatial Clustering of Applications with Noise



DBSCAN = Density-Based Spatial Clustering of Applications with Noise

- + кластеры произвольной формы**
 - + можно работать с большими данными**
 - + детерминированный (нет случайности, ответ однозначный)**
 - но может зависеть от порядка данных**
 - (как приписываемые метки, так и приписывание неосновные точек)**
 - неудобные гиперпараметры**
 - когда плотности сильно неоднородны**
 - не применяют для высокоразмерных данных**
- у scikit-learn-реализации могут быть проблемы с памятью**

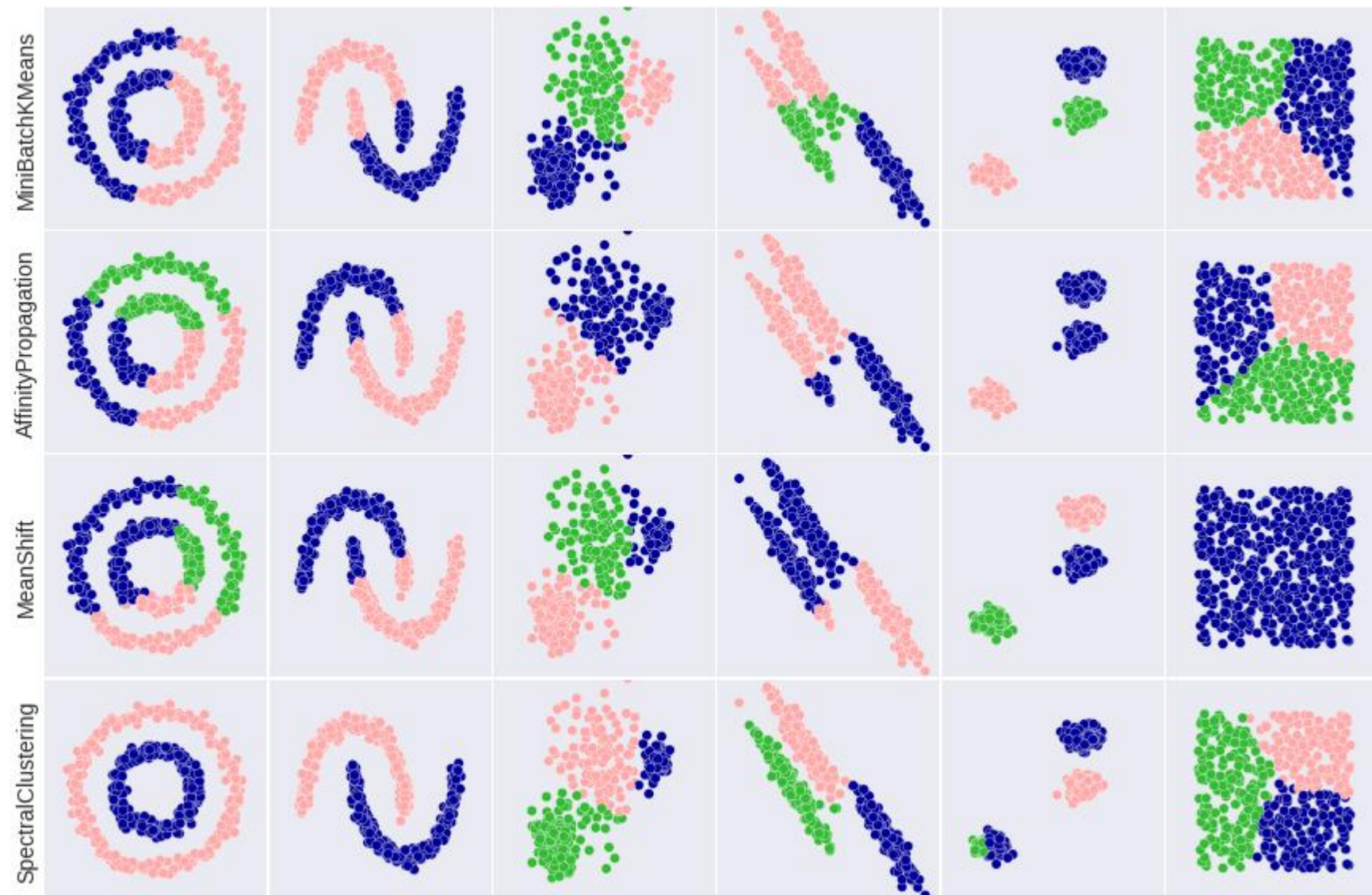
Ester, M., H. P. Kriegel, J. Sander, X. Xu «A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise» // In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press, pp. 226-231. 1996

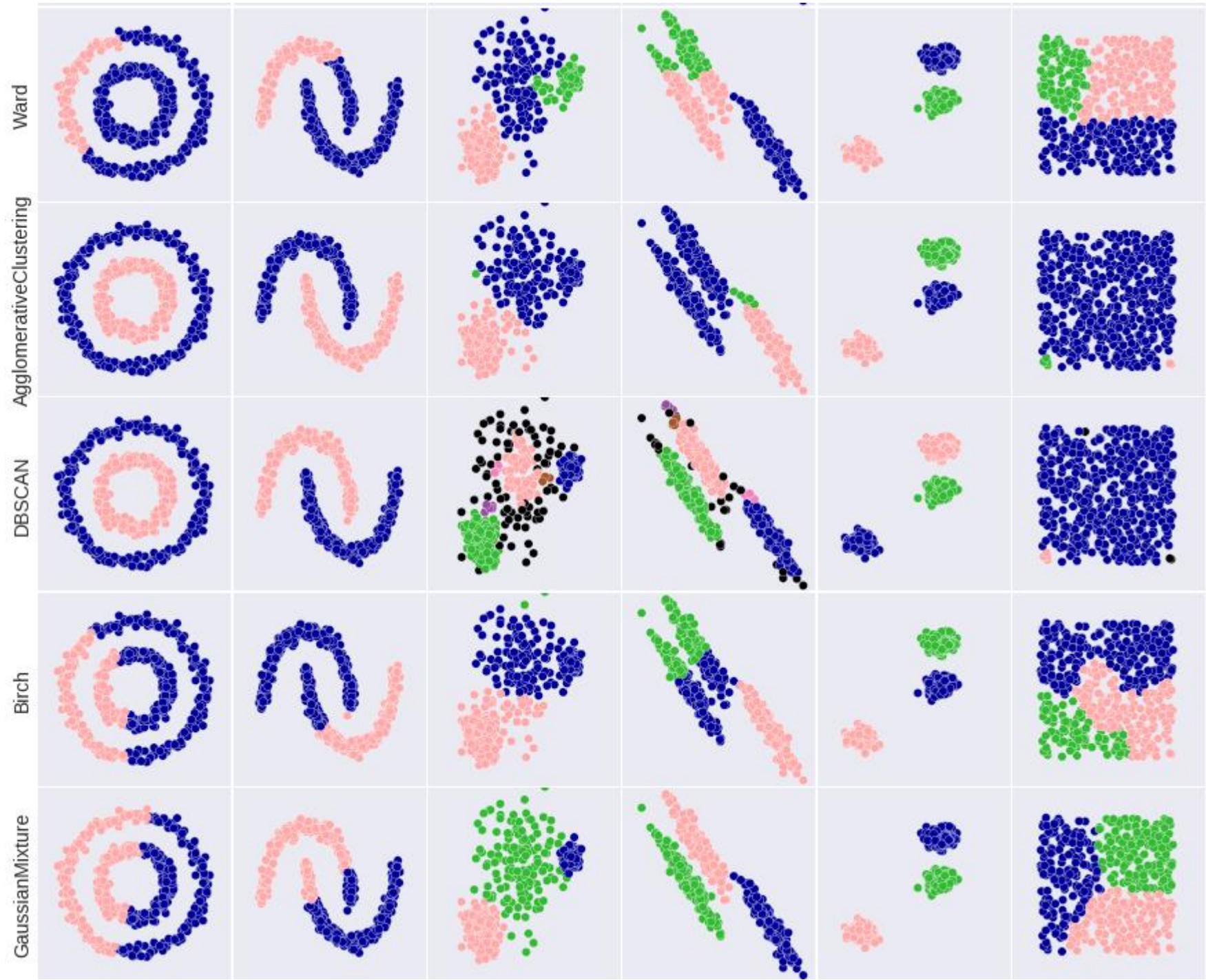
DBSCAN = Density-Based Spatial Clustering of Applications with Noise

```
from sklearn.cluster import DBSCAN
clustering = DBSCAN(eps=0.5, # порог близости
    min_samples=5, # число объектов в окрестности, чтобы считать точку основной
    metric='euclidean', # метрика
    metric_params=None, # параметры метрики
    algorithm='auto', # алгоритм для нахождения ближайших соседей: «auto»,
        # «ball_tree», «kd_tree», «brute»
    leaf_size=30, # размер листьев в BallTree / KDTree
    p=None, # степень в метрике Минковского
    n_jobs=None)

clustering.fit(X)
labels = clustering.labels_
```


Сравнение алгоритмов кластеризации





Оценка результатов кластеризации

Если знаем верную кластеризацию... внешняя оценка (External evaluation)

Вопрос: когда?

**ничего не знаем \Rightarrow согласованность с данными
внутренняя оценка (Internal evaluation)**

Оценка результатов кластеризации: «Internal evaluation»

Пусть чёткая (нет пересечений) кластеризация $U = u_1 \cup \dots \cup u_{|U|}$
множества $X = \{x_1, \dots, x_m\}$

Davies–Bouldin index

Использует центроиды и дисперсии

$$DB = \frac{1}{|U|} \sum_{i=1}^{|U|} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

Dunn index =

min между кластерами / max внутри
кластерами

$$D = \frac{\min_{1 \leq i < j \leq |U|} d(u_i, u_j)}{\max d_{\text{in}}(u_i)}$$

Silhouette

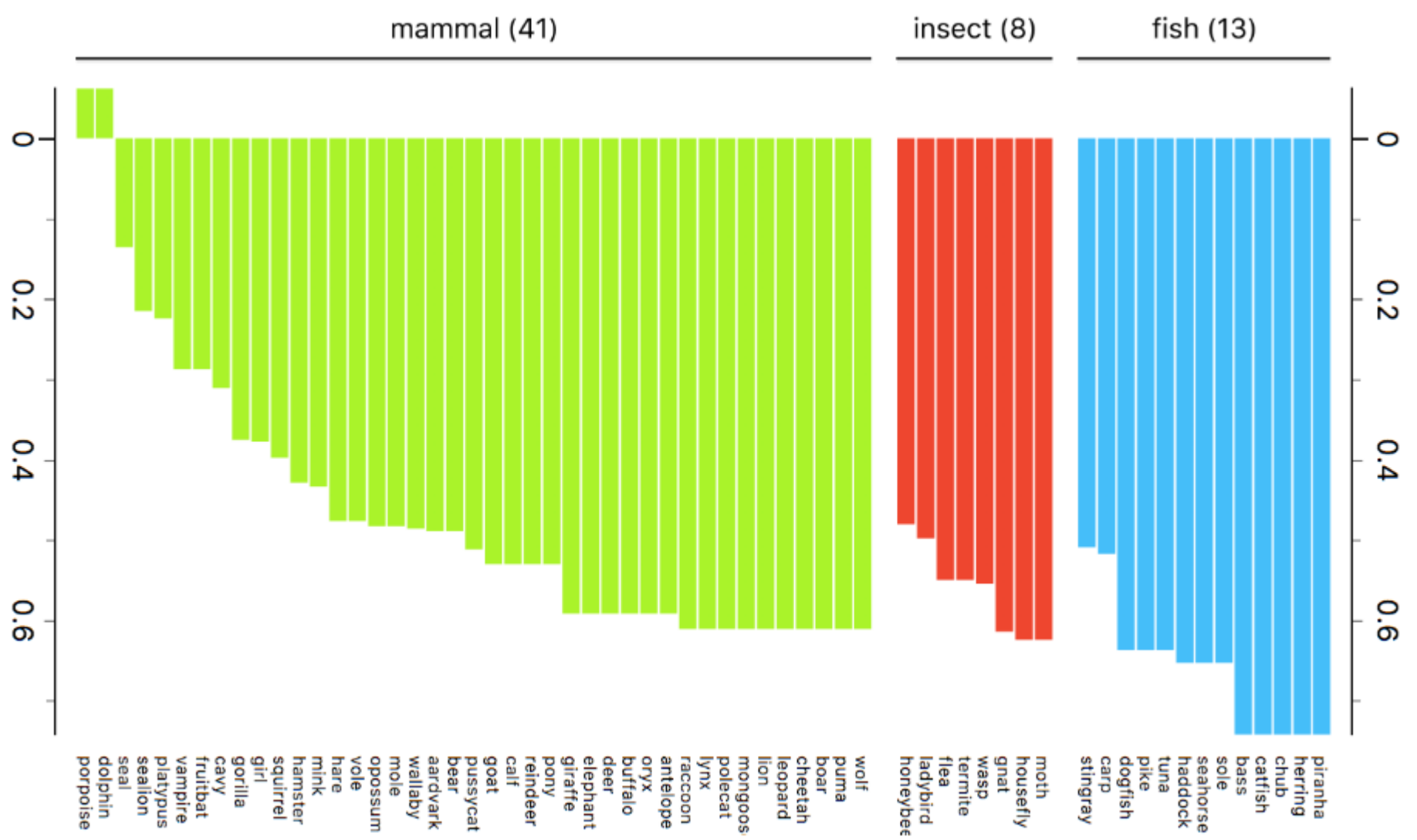
$$x_i \in u_1, d(x_i, u_2) \leq d(x_i, u_3) \leq \dots$$

Расстояние считается как среднее до
всех точек кластера

$$\text{silhouette}(x_i) = \frac{d(x_i, u_2) - d(x_i, u_1)}{\max(d(x_i, u_2), d(x_i, u_1))}$$

Можно усреднять по точкам

Оценка результатов кластеризации: «Internal evaluation»



[https://en.wikipedia.org/wiki/Silhouette_\(clustering\)](https://en.wikipedia.org/wiki/Silhouette_(clustering))

Calinski-Harabasz Index (Variance Ratio Criterion) ■

$$\frac{\text{trace} \left(\frac{1}{|U| - 1} \sum_{i=1}^{|U|} |U_i| (c - c_i)(c - c_i)^T \right)}{\text{trace} \left(\frac{1}{m - |U|} \sum_{i=1}^{|U|} \sum_{x \in U_i} (x - c_i)(x - c_i)^T \right)}$$

след матрицы межклассовой ковариации / след матрицы внутриклассовой ковариации
 c – центроид всей выборки

лучше подходит для выпуклых кластеров и евклидовой метрики

External evaluation: взаимная информация

Пусть чёткие (нет пересечений) кластеризации

$$U = u_1 \cup \dots \cup u_{|U|}$$

$$V = v_1 \cup \dots \cup v_{|V|}$$

множества $X = \{x_1, \dots, x_m\}$

$$p_i = \frac{|u_i|}{m}$$

$$H(U) = - \sum_{i=1}^{|U|} p_i \log p_i$$

Аналогично $H(V)$

$$p_{ij} = \frac{|u_i \cap v_j|}{m}$$

$$MI = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} p_{ij} \log \frac{p_{ij}}{p_i p_j}$$

потом MI ~ насколько более чётко определена U при знании V

External evaluation: нормализованная взаимная информация
Normalized Mutual Information

$$\text{NMI}(U, V) = \frac{\text{MI}(U, V)}{\text{mean}(H(U), H(V))}$$

External evaluation: скорректированная взаимная информация Adjusted mutual information

$$AMI(U, V) = \frac{MI(U, V) - E(MI(U, V))}{\max(H(U), H(V)) - E(MI(U, V))}$$

1 – если кластеризации равны
~0 – если кластеризации случайны

матожидание можно вычислить аналитически
нужно калибровать, т.к. чем больше кластеров в кластеризациях,
тем больше значение MI

```
from sklearn.metrics import mutual_info_score # MI
from sklearn.metrics import normalized_mutual_info_score # [0, 1]
from sklearn.metrics.cluster import adjusted_mutual_info_score
adjusted_mutual_info_score([0, 0, 1, 1], [0, 0, 1, 1])
```

https://en.wikipedia.org/wiki/Adjusted_mutual_information

External evaluation: V-мера

V – среднее гармоническое homogeneity и completeness

homogeneity ~ каждый кластер содержит только объекты отдельного класса

completeness ~ все объекты конкретного класса отнесены в один кластер

```
from sklearn.metrics.cluster import homogeneity_score
```

```
from sklearn.metrics.cluster import completeness_score
```

```
from sklearn.metrics.cluster import v_measure_score
```

```
v_measure_score([0, 0, 1, 1], [0, 0, 1, 1])
```

External evaluation: Adjusted Rand index

Аналогичная «Adjusted» идея, но проще...
поскольку кластеризация задаёт отношение эквивалентности

Rand index

$$R = \frac{|\{i, j : (i \sim_U j) \& (i \sim_V j)\}| + |\{i, j : (i \not\sim_U j) \& (i \not\sim_V j)\}|}{C_m^2}$$

теперь калибровка под случайную кластеризацию:

$X \backslash Y$	Y_1	Y_2	\dots	Y_s	Sums
X_1	n_{11}	n_{12}	\dots	n_{1s}	a_1
X_2	n_{21}	n_{22}	\dots	n_{2s}	a_2
\vdots	\vdots	\vdots	\ddots	\vdots	\vdots
X_r	n_{r1}	n_{r2}	\dots	n_{rs}	a_r
Sums	b_1	b_2	\dots	b_s	

$$\overbrace{ARI}^{\text{Adjusted Index}} = \frac{\overbrace{\sum_{ij} \binom{n_{ij}}{2}}^{\text{Index}} - \overbrace{[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}] / \binom{n}{2}}^{\text{Expected Index}}}{\underbrace{\frac{1}{2} [\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}]}_{\text{Max Index}} - \underbrace{[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}] / \binom{n}{2}}_{\text{Expected Index}}}$$

```
from sklearn.metrics.cluster import adjusted_rand_score
adjusted_rand_score([0, 0, 1, 1], [0, 0, 1, 1])
```

https://en.wikipedia.org/wiki/Rand_index#Adjusted_Rand_index

External evaluation: общий подход

Кластеризация ~ классификация пар

$$\{x_1, \dots, x_m\} \rightarrow \{(1,1), \dots, (i,j), \dots, (m,m)\}$$

$$a_U(i, j) = 1 \Leftrightarrow i \sim_U j$$

Можно сравнивать классификации a_U и a_V

Пример, Rand index:
$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

Fowlkes-Mallows index (FMI)

– среднее геометрическое точности и полноты

$$FMI = \frac{TP}{\sqrt{(TP + FP)(TP + FN)}}$$

```
from sklearn.metrics.cluster import fowlkes_mallows_score  
fowlkes_mallows_score([0, 0, 1, 1], [0, 0, 1, 1])
```

есть и много других...

Итоги

Общие подходы при кластеризации:
вычисление центров кластеров – итерационное
оценка распределений (пока не было)
оценка плотности (пока не было)
Иерархические – разбиение / слияние кластеров
Графовые методы

Есть ещё нейростевые... будет потом

Ссылки

Дьяконов, А. Г. «Анализ данных, обучение по прецедентам, логические игры, системы WEKA, RapidMiner и MatLab (практикум на ЭВМ кафедры математических методов прогнозирования)» МАКСПресс 2010 //

<http://www.machinelearning.ru/wiki/images/7/7e/Dj2010up.pdf>

Интересные алгоритмы кластеризации

<https://habr.com/ru/post/322034/>