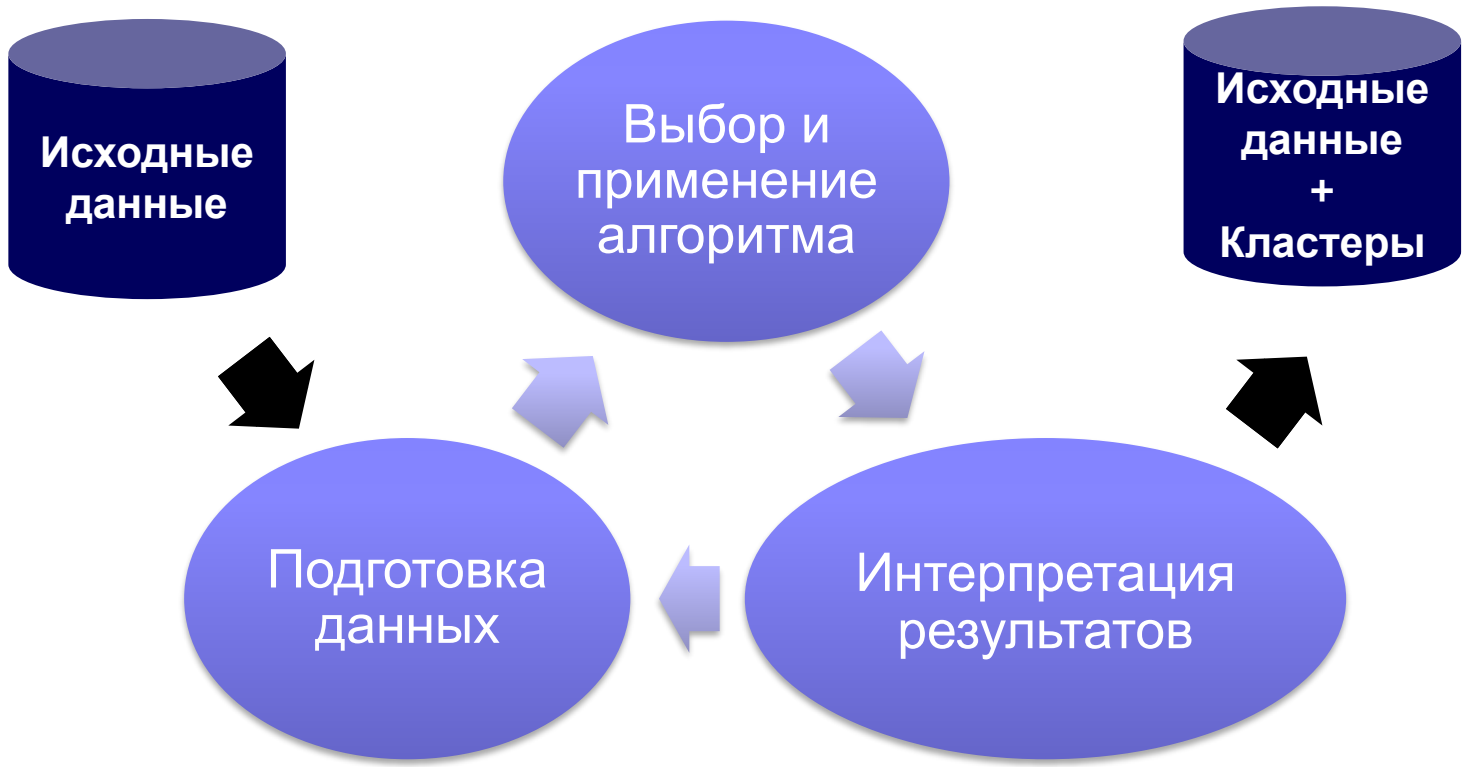
Лекция 16: Кластеризация

Что есть кластер?

- Кластер: группа «похожих» объектов
  - «похожих» между собой в группе (внутриклассовое расстояние)
  - «не похожих» на объекты других групп
  - определение неформальное, формализация зависит от метода
- Кластерный анализ -
  - разбиение множество объектов на группы (кластеры)
- Тип моделей:
  - «описательный» (descriptive) => одна из задач - наглядное представление кластеров
  - «прогнозный» (predictive) => разбиение на кластеры, а затем «классификация» новых объектов
- Тип обучения - всегда «без учителя» (unsupervised) => тренировочный набор не размечен
- Много приложений:
  - Предобработка данных
  - Группировка и профилирование
  - Разведочный анализ

Этапы кластерного анализа



Постановка задачи кластеризации

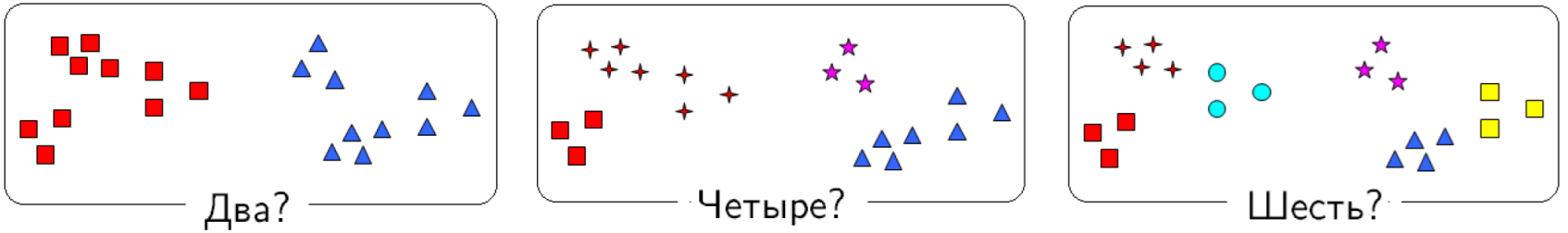
- Дано:
  - X – пространство объектов;
  - $X^l = \{x_1, \dots, x_l\}$ – обучающая выборка;
  - $\rho: X \times X \to [0, \infty)$ – функция расстояния между объектами.
- Найти:
  - Y – множество кластеров;
  - $a: X \to Y$ – алгоритм кластеризации,
  - такие, что:
    - каждый кластер состоит из близких объектов;
    - объекты разных кластеров существенно различны.
- Это задача обучения без учителя (unsupervised learning).

Характеристики метода кластеризации

- Масштабируемость
- Поддержка различных типов атрибутов и структур данных
- Возможность находить кластеры сложной формы
- Отсутствие обязательных требований к наличию априорных знаний о выборке (например, о распределениях)
- Устойчивость к «шуму» и выбросам
- Возможность работы с высокой размерностью и с большой выборкой
- Возможность включать пользовательские ограничения и зависимости
- Интерпретируемость и наглядность (прототипы, границы, правила, функции принадлежности и т.п.)
- Интуитивность параметров кластеризации

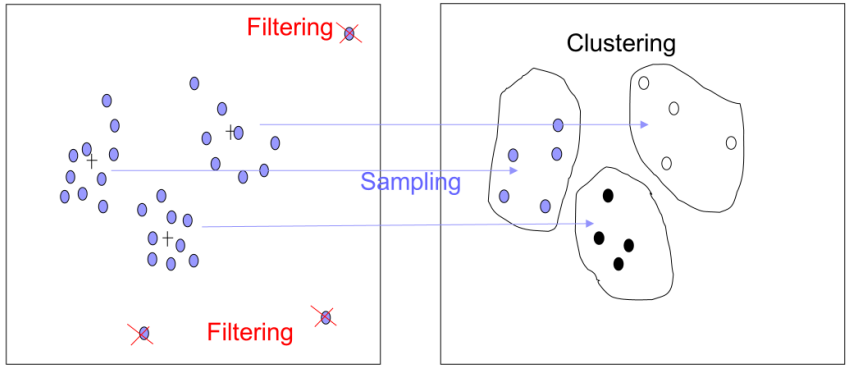
Качество кластеризации

- Хороший метод кластеризации находит кластеры
  - с высоким «внутриклассовым» сходством объектов
  - и низким «межклассовым» сходством объектов
- Оценка качества кластеризации (нет понятия «точность»)
  - необходима, так как влияет на выбор параметров метода
  - определяется либо экспертом – субъективная величина
  - либо «перекрестной» проверкой целевой функции кластеризации
- Качество кластеризации зависит:
  - от метода кластеризации и меры сходства (или расстояния)

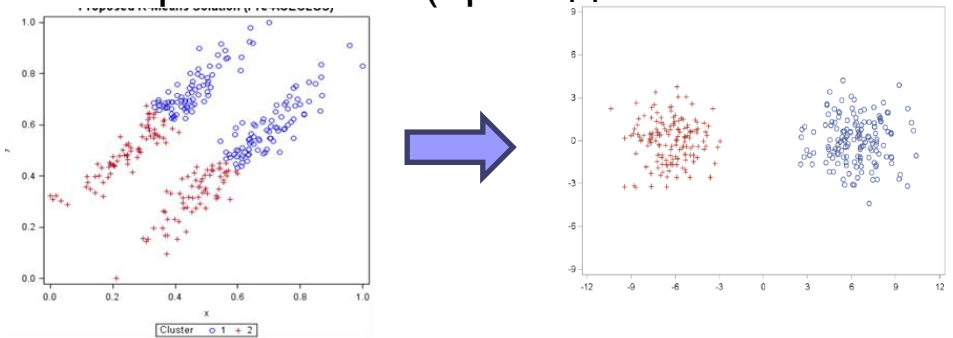


Подготовка данных для кластеризации

- Отбор наблюдений
  - Что я разбиваю на кластеры?
  - Исключить выбросы
  - Уменьшить выборку

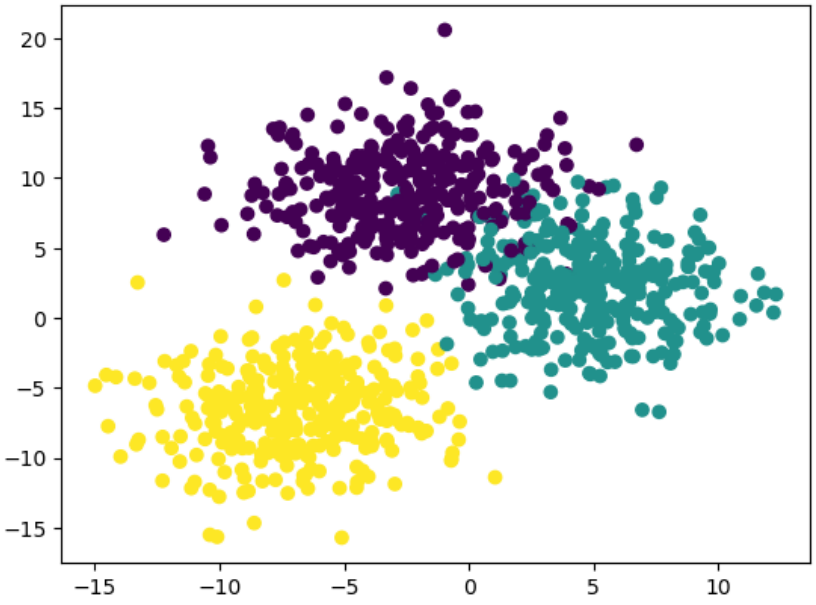


- Отбор и трансформация переменных
  - Какие характеристики объектов важны? Выбирает эксперт.
  - Переменные коррелируют? (методы отбора)
  - Распределения переменных симметричны? (преобразования)
  - Сравнимы ли масштабы переменных? (приведение к близким шкалам)

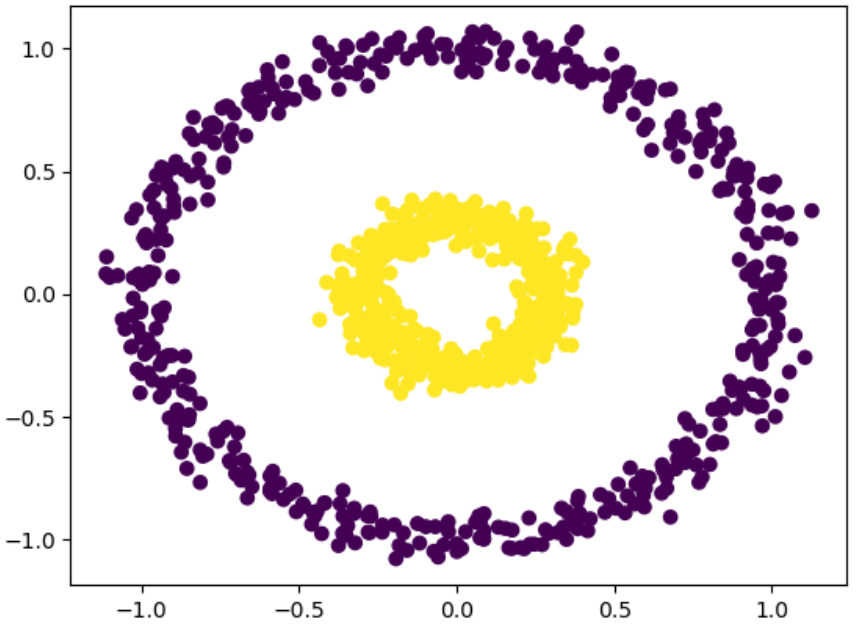


Модельные пример данных

```python
# Генерация данных 3х облаков точек
from sklearn.datasets import make_blobs

n_samples = 1000
X_blob, y_blob = make_blobs(n_samples=n_samples,
                            centers=3, cluster_std=3,
                            random_state=42)
plt.scatter(X_blob[:, 0], X_blob[:, 1], c=y_blob)
```




```python
# Генерация данных концентрических кругов
from sklearn.datasets import make_circles

n_samples = 1000
X_cl, y_cl = make_circles(n_samples=n_samples, factor=0.3,
                          noise=0.05, random_state=0)
plt.scatter(X_cl[:, 0], X_cl[:, 1], c=y_cl)
```



Представление исходных данных

- Матрица признаков:
  - Числовые
  - Бинарные
  - Номинальные (категориальные)
  - Упорядоченные шкалы
  - Нелинейные шкалы

$$\begin{bmatrix} x_{11} & \dots & x_{1f} & \dots & x_{1p} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{if} & \dots & x_{ip} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & \dots & x_{nf} & \dots & x_{np} \end{bmatrix}$$

- Матрица различия (или сходства):
  - «Естественные» расстояния предметной области
  - Экспертные оценки (противоречивы, нетранзитивны, недостоверны)

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \dots & \dots & 0 \end{bmatrix}$$

Основные типы алгоритмов кластеризации

- Иерархические:
  - Создается иерархическая декомпозиция исходного множества объектов в соответствии с некоторой стратегией «объединения» (восходящая кластеризация) или «разбиения» (нисходящая)
- На основе группировки (partitioning):
  - Направленный перебор вариантов разбиения исходного множества объектов, выбор лучшего по некоторому критерию
  - Обычно на основе прототипов - k-means, k-medoids
- На основе связности или непараметрической оценки плотности:
  - Кластеры ищутся в виде связных областей с помощью локальной оценки числа ближайших соседей или ядерной оценки плотности
- Моделе-ориентированые (статистические):
  - Выбирается некоторая гипотеза (параметрическая модель) о структуре кластеров и находятся, параметры, наилучшим образом приближающие эту модель

Natural Grouping Criterion

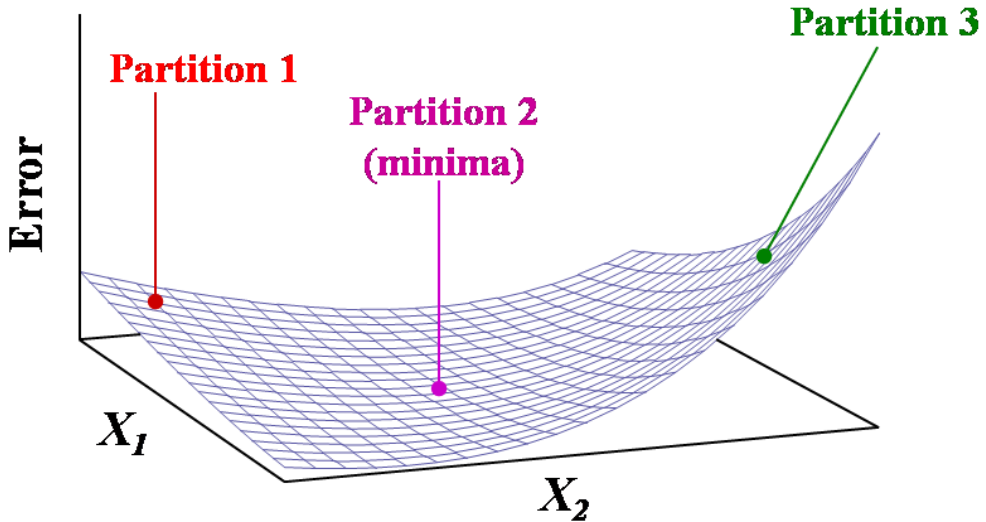
Мера похожести объектов

Малое внутрикластерное расстояние
→ F уменьшается

$$F = \frac{\sum_k \sum_{l,m} \Phi_1 \left(\begin{matrix} \text{расстояние между} \\ \text{наблюдениями } x_l \text{ и } x_m \text{ в кластере } k \end{matrix} \right)}{\sum_{i,j} \Phi_2(\text{расстояние между кластерами } i \text{ и } j)}$$

Большое межкластерное расстояние
→ F уменьшается

Мера близости кластеров



Качество кластеризации в метрическом и векторном пространствах

- Пусть известны только попарные расстояния между объектами.
  - $a_i = a(x_i)$ – кластеризация объекта x_i
  - Среднее внутрикластерное расстояние:

$$\mathrm{F}_1 = \frac{\sum_{i<j}[a_i = a_j]\rho(x_i, x_j)}{\sum_{i<j}[a_i = a_j]} \to min$$

  - Среднее межкластерное расстояние:

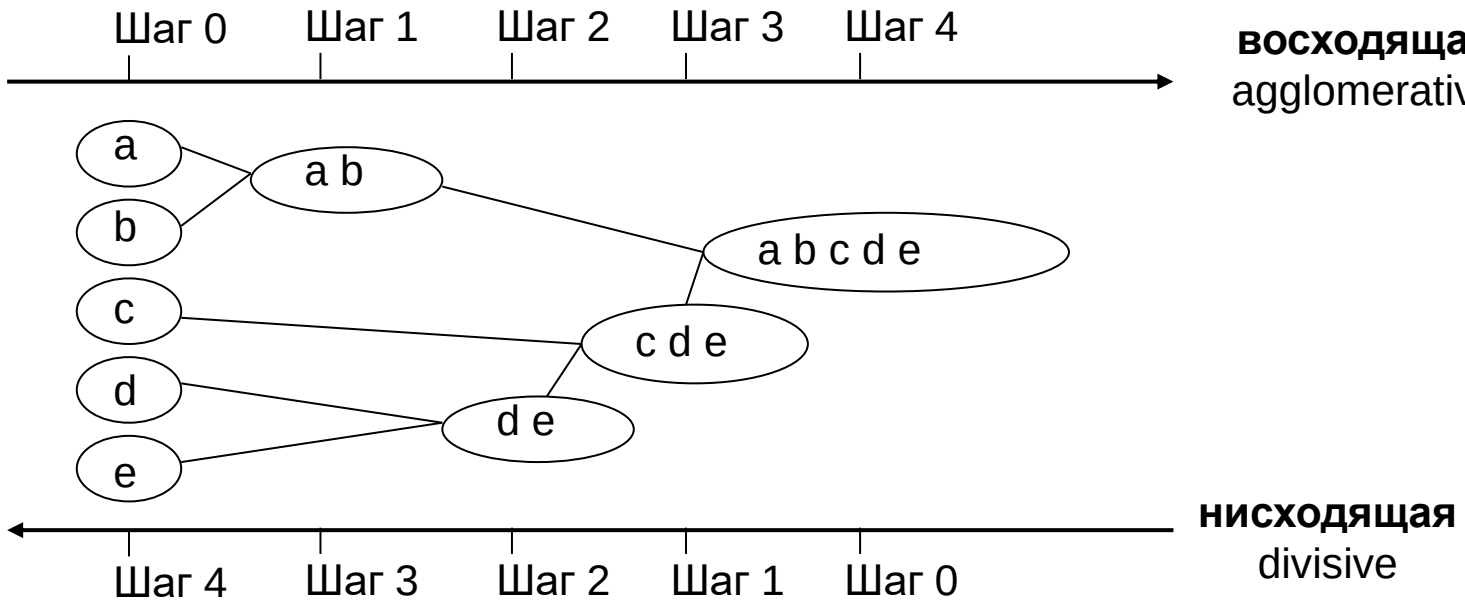
$$\mathrm{F}_2 = \frac{\sum_{i<j}[a_i \neq a_j]\rho(x_i, x_j)}{\sum_{i<j}[a_i \neq a_j]} \to max.$$

- Пусть объекты x_i задаются векторами $\left(f_1(x_i), \dots, f_n(x_i)\right)$.
  - Сумма средних внутрикластерных расстояний:

$$F_1 = \sum_{a \in Y} \frac{1}{|X_a|} \sum_{i:a_i=a} \rho(x_i, \mu_a) \to min,$$

  $X_a = \{x_i \in X^l | a_i = a\}$ – кластер a, μ_a – центр масс кластера a.
  - Сумма межкластерных расстояний: $F_2 = \sum_{a,b \in Y} \rho(\mu_a, \mu_b) \to max.$
- Отношение пары функционалов: $NGC = \frac{\mathrm{F}_1}{\mathrm{F}_2} \to min.$

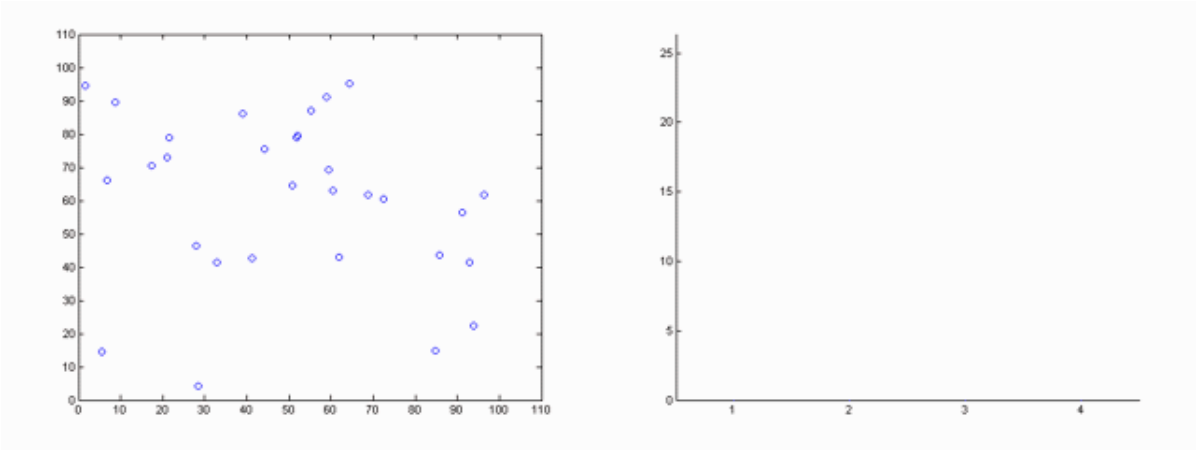
Иерархическая кластеризация



- Параметры:
  - Обычно используется только матрица сходства (различия) и не требуется дополнительных параметров (например, числа кластеров)
- Процесс:
  - «Пошаговое» объединение ближайших кластеров (восходящая) или разбиение наиболее удаленных (нисходящая)

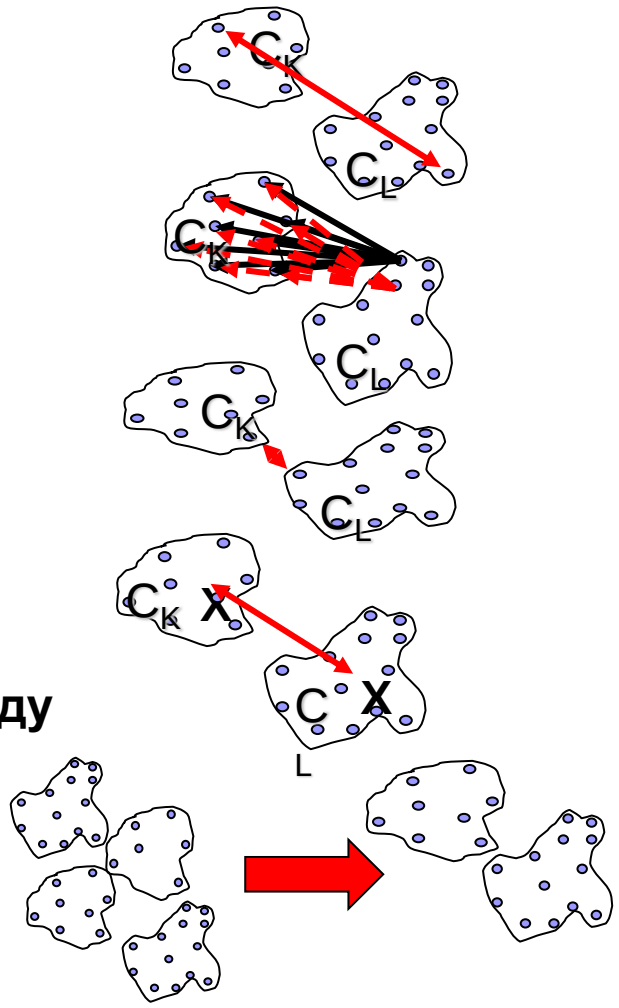
Агломеративная иерархическая кластеризация

- Алгоритм иерархической кластеризации (Ланс, Уильямс, 1967):
  - Итеративный пересчет расстояний R_{UV} между кластерами U, V.
  - $C_1 := \{\{x_1\}, \ldots, \{x_l\}\}$ – все кластеры 1-элементные;
  - $R_{\{x_i\}\{x_j\}} := \rho(x_i, x_j)$ – расстояния между ними;
- Для всех $t = 2, \ldots, l$ (t – номер итерации):
  - Найти в C_{t-1} пару кластеров (U, V) с минимальным R_{UV};
  - Слить их в один кластер: $W := U \cup V$; $C_t := C_{t-1} \cup \{W\} \setminus \{U, V\}$;
  - Для всех $S \in C_t$ найти $R_{ws} := \alpha_U R_{US} + \alpha_V R_{VS} + \beta R_{UV} + \gamma |R_{US} - R_{VS}|$



Оценка близости кластеров

- Расчет расстояния на основе попарных расстояний между элементами различных кластеров:
  - **Полное связывание (расстояние дальнего соседа)**: наибольшее попарное расстояние. Дает компактные сферические кластеры.
  - **Среднее связывание (групповое среднее расстояние)**: усредненное попарное расстояние. Редко используется.
  - **Единственное связывание (расстояние ближайшего соседа)**: наименьшее попарное расстояние. Дает «растянутые» кластеры сложной формы.
  - **Центроидное связывание (расстояние между центрами)*:*** расстояние между центрами (мат. ожидание) кластеров.
  - **Метод Ward'a** – минимизирует внутрикластерные дисперсии



Частные случаи формулы Ланса-Уильямса

$$R_{ws} := \alpha_U R_{US} + \alpha_V R_{VS} + \beta R_{UV} + \gamma |R_{US} - R_{VS}|$$

- Расстояние ближнего соседа:

$$R^{\text{б}}_{WS} = \min_{w \in W, s \in S} \rho(w,s);\ \alpha_U = \alpha_V = \frac{1}{2},\ \ \beta = 0,\ \ \gamma = -\frac{1}{2}.$$

- Расстояние дальнего соседа:

$$R^{\text{д}}_{WS} = \max_{w \in W, s \in S} \rho(w,s);\ \alpha_U = \alpha_V = \frac{1}{2},\ \ \beta = 0,\ \ \gamma = \frac{1}{2}.$$

- Групповое среднее расстояние:

$$R^{\text{г}}_{WS} = \frac{1}{|W||S|} \sum_{w \in W} \sum_{s \in S} \rho(w,s);\ \alpha_U = \frac{|U|}{|W|}, \alpha_V = \frac{|V|}{|W|}, \beta = \gamma = 0.$$

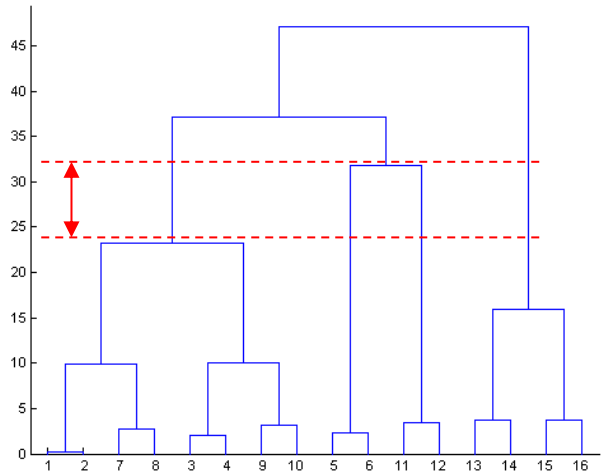
- Расстояние между центрами:

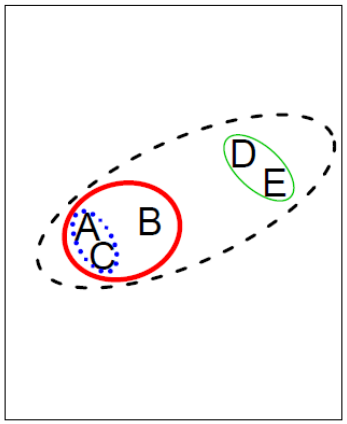
$$R^{\text{ц}}_{WS} = \rho^2\left(\sum_{w \in W} \frac{w}{|W|}, \sum_{s \in S} \frac{s}{|S|}\right);\ \alpha_U = \frac{|U|}{|W|}, \alpha_V = \frac{|V|}{|W|}, \beta = -\alpha_U \alpha_V, \gamma = 0.$$

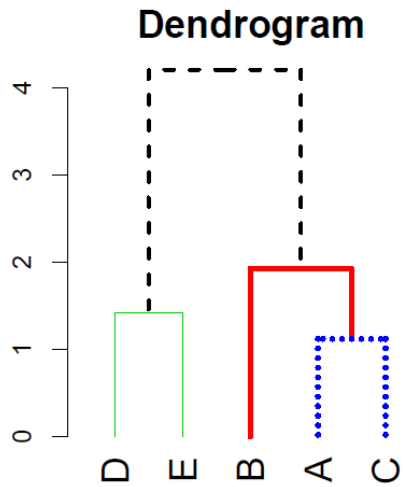
- Расстояние Уорда:

$$R^{y}_{WS} = \frac{|S||W|}{|S|+|W|} \rho^2 \left(\sum_{w \in W} \frac{w}{|W|}, \sum_{s \in S} \frac{s}{|S|}\right);\ \alpha_U = \frac{|S|+|U|}{|S|+|W|}, \alpha_V = \frac{|S|+|V|}{|S|+|W|}, \gamma = 0, \beta = \frac{-|S|}{|S|+|W|}.$$

Представление иерархических кластеров - Дендрограмма







- бинарное дерево, описывающее все шаги разбиения
- корень – общий кластер, листья – отдельные объекты
- «высота» ветвей (до пересечения) – порог расстояния «склейки» («разделения»)
- результат кластеризации – «срез» дендрограммы
- можно выбирать лучший C_t по

$$\max_t |R_t - R_{t-1}|$$

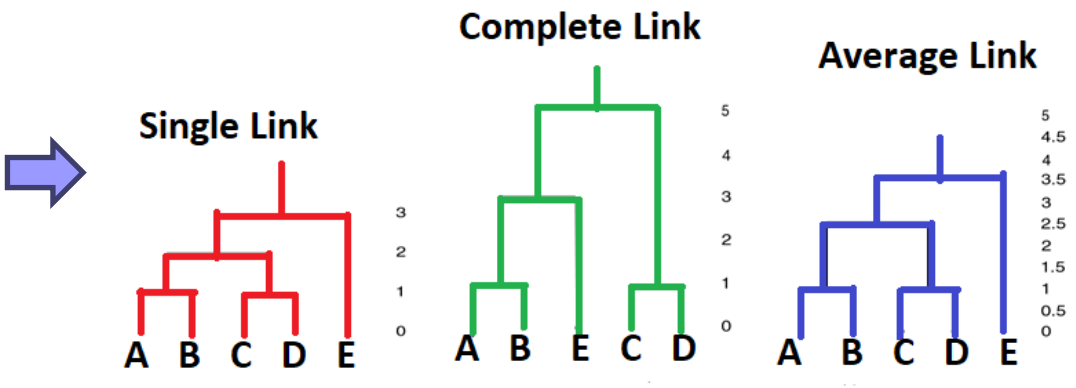
Основные свойства иерархической кластеризации

- Монотонность:
  - дендрограмма не имеет самопересечений, при каждом слиянии расстояние между объединяемыми кластерами только увеличивается: $R_2 \leq R_3 \leq \cdots < R_l$.
- Свойства разных типов расстояний:
  - Сжимающее расстояние: $R_t \leq \rho(\mu_U, \mu_V), \forall t$.
  - Растягивающее расстояние: $R_t \geq \rho(\mu_U, \mu_V), \forall t$.
  - $R^{\text{б}}$ – сжимающее; $R^{\text{д}}, R^{\text{у}}$ – растягивающие.
- Теорема (Миллиган, 1979)
  - Кластеризация монотонна, если выполняются условия
  - $\alpha_U \geq 0, \alpha_V \geq 0, \alpha_U + \alpha_V + \beta \geq 1, \min\{\alpha_U, \alpha_V\} + \gamma \geq 0$
- $R^{\text{ц}}$ не монотонно; $R^{\text{б}}, R^{\text{д}}, R^{\text{г}}, R^{\text{у}}$ – монотонны.

Обсуждение иерархической кластеризации

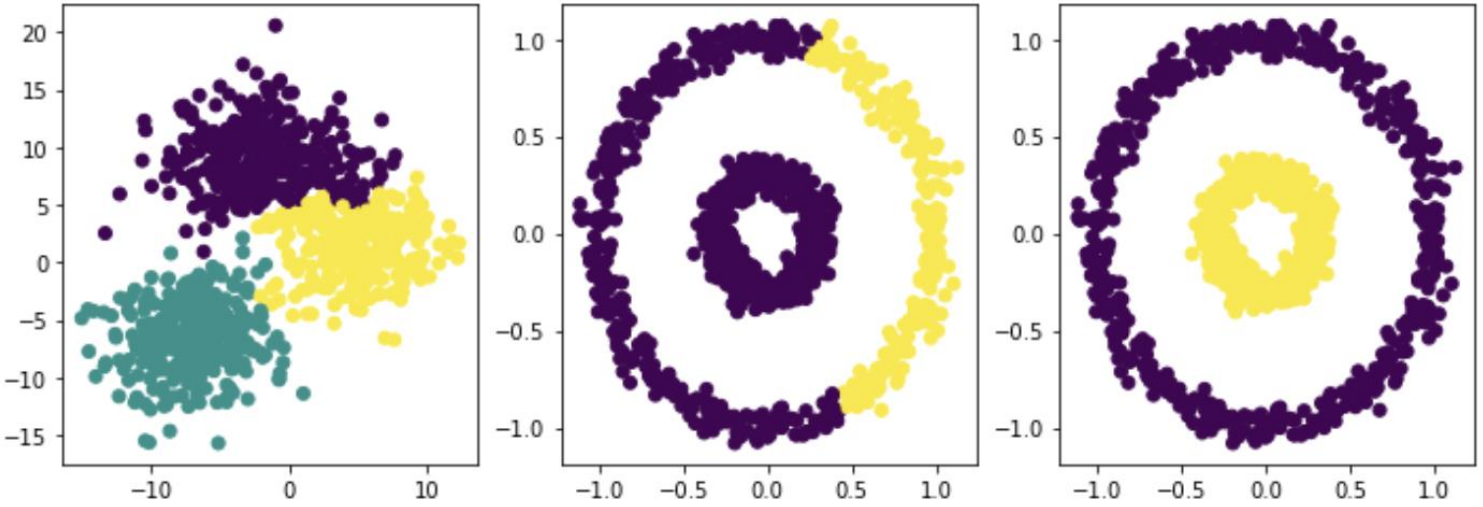
- Достоинства:
  - Очень просто и понятно, легко реализовать
  - Наглядные дендрограммы
  - Нет метапараметров кроме межкласстерного расстояния
- Недостатки:
  - не масштабируется: временная квадратичная сложность от числа кластеризуемых объектов
  - жадный алгоритм - локальная оптимальность с точки зрения NGC
  - относительно слабая интерпретируемость результата и многое зависит от расстояния

	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0



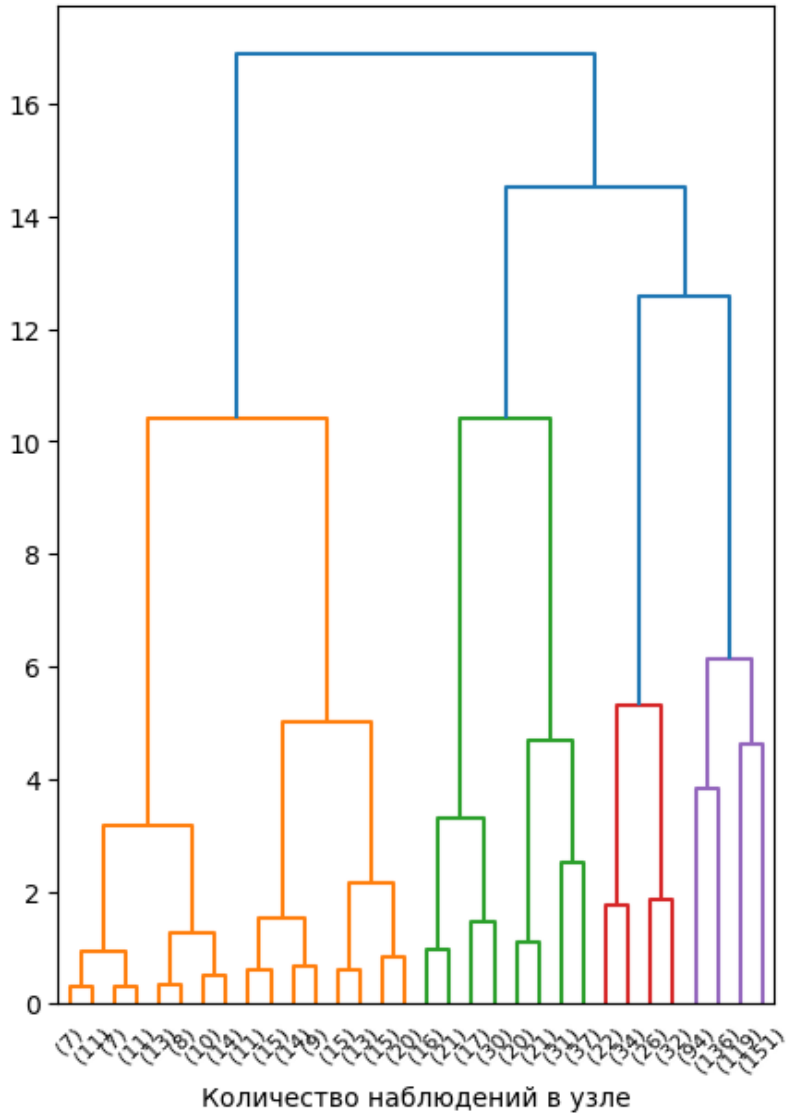
Пример использования

```
from sklearn.cluster import AgglomerativeClustering

fig, axes = plt.subplots(ncols=3, figsize=(12,4))
agg_bl = AgglomerativeClustering(n_clusters=3)
axes[0].scatter(X_blob[:, 0], X_blob[:, 1], c=agg_bl.fit_predict(X_blob))

agg_cl = AgglomerativeClustering(n_clusters=2)
axes[1].scatter(X_cl[:, 0], X_cl[:, 1], c=agg_cl.fit_predict(X_cl))

agg_cl1 = AgglomerativeClustering(n_clusters=2, linkage="single")
axes[2].scatter(X_cl[:, 0], X_cl[:, 1], c=agg_cl1.fit_predict(X_cl))
```



Пример использования дендрограммы

```python
from scipy.cluster.hierarchy import dendrogram

def plot_dendrogram(model, **kwargs):
    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
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack(
        [model.children_, model.distances_, counts]
    ).astype(float)
    dendrogram(linkage_matrix, **kwargs)
    plt.xlabel("Количество наблюдений в узле")

model = AgglomerativeClustering(distance_threshold=0,
                                n_clusters=None)
model = model.fit(X_cl)
plot_dendrogram(model, truncate_mode="level", p=4)
```



Пример использования с матрицей данных и матрицей расстояний на основе Jaccard

```
df = pd.read_csv("C:\\SAS\\edu\\course\\DM 2017\\assc_TRANSACTION.csv")
df=df.groupby("CUSTOMER")["PRODUCT"].value_counts().unstack(fill_value=0)
df.head(5)
```

PRODUCT CUSTOMER	apples	artichok	avocado	baguette	bordeaux	bourbon	chicken	coke	corned_b	cracker	ham	heineken	hering	ice_crea	olives	peppers
0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	1	0
1	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0
2	0	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0
3	0	0	0	0	0	1	0	1	0	0	1	0	0	1	1	1
4	1	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0

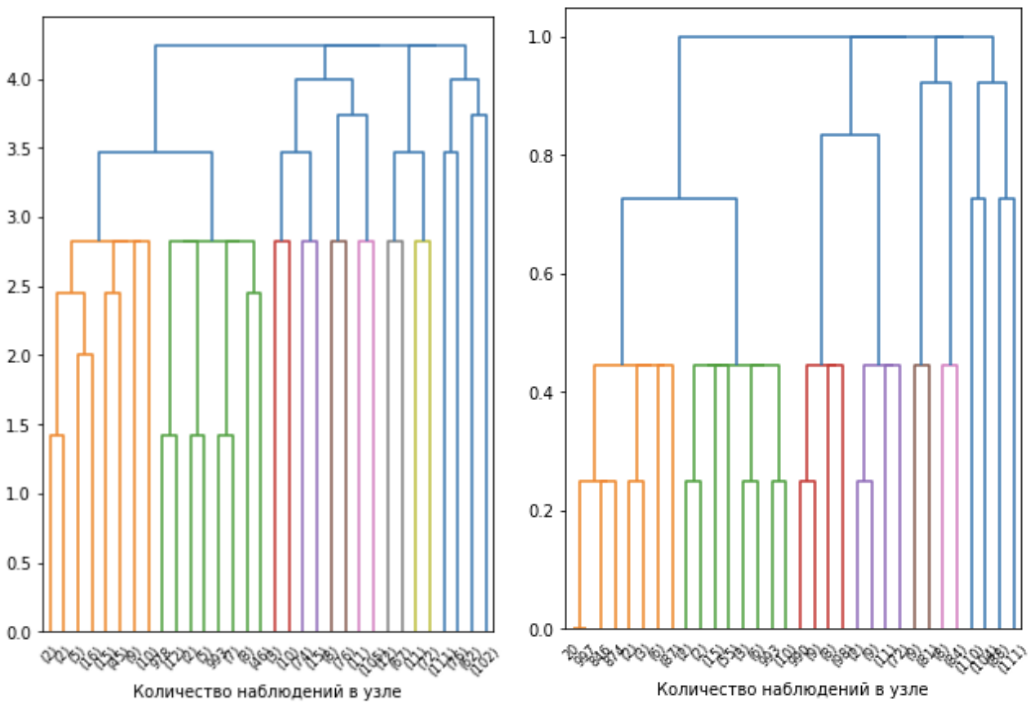
```
#рассчет матрицы сходства на основе коэф Jaccard'а
from sklearn.metrics import jaccard_score as js

N = df.axes[0].values.size
dist_matrix=np.zeros(N*N)
dist_matrix=dist_matrix.reshape(N,N)
for i in range(1,N):
    for j in range(0,i):
        dist_matrix[i,j]=dist_matrix[j,i]=1-js(df.iloc[j]>0,df.iloc[i]>0)
```

```
pd.DataFrame(dist_matrix).head(5)
```

	0	1	2	3	4	5	6	7	8	9	...	991	992	993	994	99
0	0.000000	0.727273	0.833333	0.444444	0.600000	0.833333	0.600000	0.727273	0.833333	0.727273	...	0.600000	0.916667	0.727273	0.923077	0.83333
1	0.727273	0.000000	0.833333	0.923077	0.727273	0.923077	0.833333	0.727273	0.444444	0.727273	...	0.833333	0.818182	0.600000	0.727273	0.44444
2	0.833333	0.833333	0.000000	0.833333	0.833333	0.727273	0.833333	0.923077	0.833333	0.923077	...	0.923077	0.375000	0.833333	0.727273	0.83333
3	0.444444	0.923077	0.833333	0.000000	0.833333	0.600000	0.444444	0.727273	0.727273	0.833333	...	0.444444	0.916667	0.923077	1.000000	0.92307
4	0.600000	0.727273	0.833333	0.833333	0.000000	1.000000	0.727273	0.923077	0.923077	0.833333	...	0.727273	0.916667	0.444444	0.727273	0.83333

Пример использования с матрицей данных и матрицей расстояний на основе Jaccard

```
model = AgglomerativeClustering(distance_threshold=0, linkage="complete",
                                n_clusters=None)
model = model.fit(df)
plot_dendrogram(model, truncate_mode="level", p=4)

model = AgglomerativeClustering(affinity="precomputed", linkage="complete",
                                distance_threshold=0, n_clusters=None)
model = model.fit(dist_matrix)
plot_dendrogram(model, truncate_mode="level", p=4)
```



Кластеризация на основе группировки (partitioning) с прототипами

- Основная постановка задачи $\mathrm{F}_1 \to min$ при фиксированном числе кластеров K: найти разбиение $X = \{x_i\}_{i=1}^{l}$ на K непересекающихся подмножеств $\{\mathrm{C}_k\}_{k=1}^{K}$, покрывающих X так, чтобы минимизировать внутриклассовое расстояние:

$$\min_{C_i \cap C_j = \emptyset, \cup C_k = X} \sum_{k=1}^{K} \sum_{x_i \in C_k} \sum_{x_j \in C_k} \rho(x_i, x_j)$$

- Точное решение – перебор с отсечением (число комбинаций велико): $S(l,K) = \frac{1}{K!} \sum_{k=1}^{K} (-1)^{K-k} \binom{K}{k} K^l$
- Аппроксимация (μ_k - прототипы кластеров):

$$\min_{C_i \cap C_j = \emptyset, \cup C_k = X} \sum_{k=1}^{K} \sum_{x_j \in C_k} \rho(\mu_k, x_j)$$

Метод K-средних (K-means) для кластеризации в векторном пространстве

- Минимизация суммы квадратов внутрикластерных расстояний:

$$\sum_{i=1}^{l}\|x_i - \mu_{a_i}\|^2 \to \min_{\{a_i\},\{\mu_a\}}, \quad \|x_i - \mu_a\|^2 = \sum_{j=1}^{n}\left(f_j(x_i) - \mu_{aj}\right)^2$$

- Алгоритм Ллойда
  - Вход: $X^l, K = |Y|$; выход: центры кластеров $\mu_a, a \in Y$;
  - $\mu_a :=$ начальное приближение центров, для всех $a \in Y$;
  - Повторять

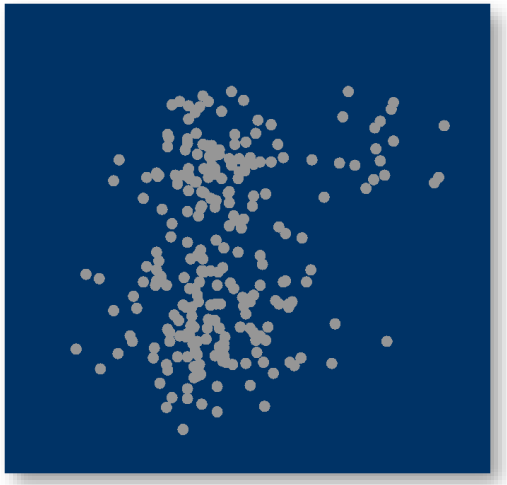
    Отнести каждый x_i к ближайшему центру:

$$a_i := \arg\min_{a \in Y}\|x_i - \mu_a\|$$

    Вычислить новые положения центров:

$$\mu_a := \frac{\sum_{i=1}^{l}[a_i=a]x_i}{\sum_{i=1}^{l}[a_i=a]}, a \in Y;$$

  - Пока a_i не перестанут изменяться.



Метод K-средних (K-means) для кластеризации в векторном пространстве

- Минимизация суммы квадратов внутрикластерных расстояний:

$$\sum_{i=1}^{l}\left\|x_i-\mu_{a_i}\right\|^2 \to \min_{\{a_i\},\{\mu_a\}}, \quad \|x_i-\mu_a\|^2=\sum_{j=1}^{n}\left(f_j(x_i)-\mu_{aj}\right)^2$$

- Алгоритм Ллойда
  - Вход: $X^l, K=|Y|$; выход: центры кластеров $\mu_a, a\in Y$;
  - $\boldsymbol{\mu_a} :=$ **начальное приближение центров, для всех** $\boldsymbol{a}\in\boldsymbol{Y}$**;**
  - Повторять

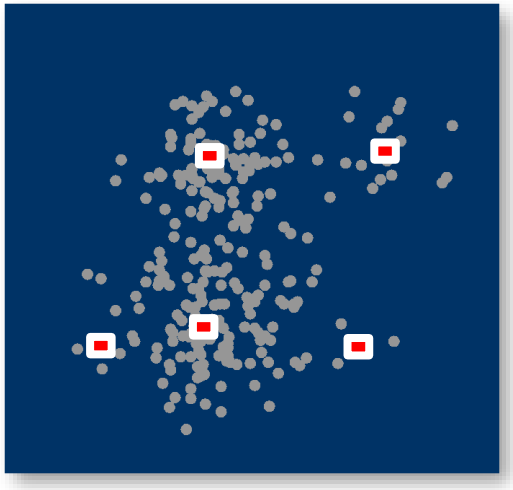
    Отнести каждый x_i к ближайшему центру:

$$a_i := \arg\min_{a\in Y}\|x_i-\mu_a\|$$

    Вычислить новые положения центров:

$$\mu_a := \frac{\sum_{i=1}^{l}[a_i=a]x_i}{\sum_{i=1}^{l}[a_i=a]}, a\in Y;$$

  - Пока a_i не перестанут изменяться.



Метод K-средних (K-means) для кластеризации в векторном пространстве

- Минимизация суммы квадратов внутрикластерных расстояний:

$$\sum_{i=1}^{l}\left\|x_i-\mu_{a_i}\right\|^2 \to \min_{\{a_i\},\{\mu_a\}}, \quad \|x_i-\mu_a\|^2=\sum_{j=1}^{n}\left(f_j(x_i)-\mu_{aj}\right)^2$$

- Алгоритм Ллойда
  - Вход: $X^l, K=|Y|$; выход: центры кластеров $\mu_a, a\in Y$;
  - $\mu_a :=$ начальное приближение центров, для всех $a\in Y$;
  - Повторять

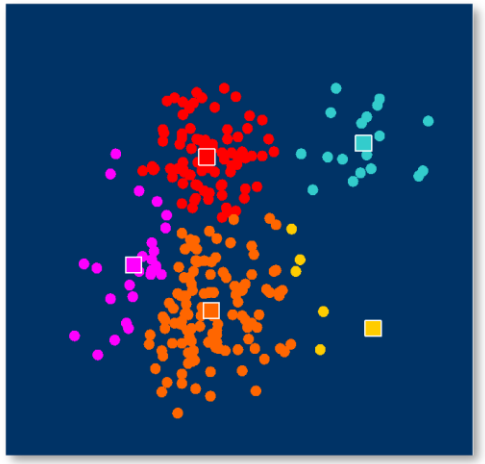
    Отнести каждый x_i к ближайшему центру:

$$a_i := \arg\min_{a\in Y}\|x_i-\mu_a\|$$

    Вычислить новые положения центров:

$$\mu_a := \frac{\sum_{i=1}^{l}[a_i=a]x_i}{\sum_{i=1}^{l}[a_i=a]}, a\in Y;$$

  - Пока a_i не перестанут изменяться.



Метод K-средних (K-means) для кластеризации в векторном пространстве

- Минимизация суммы квадратов внутрикластерных расстояний:

$$\sum_{i=1}^{l}\left\|x_i-\mu_{a_i}\right\|^2 \to \min_{\{a_i\},\{\mu_a\}}, \quad \|x_i-\mu_a\|^2=\sum_{j=1}^{n}\left(f_j(x_i)-\mu_{aj}\right)^2$$

- Алгоритм Ллойда
  - Вход: $X^l, K=|Y|$; выход: центры кластеров $\mu_a, a\in Y$;
  - $\mu_a :=$ начальное приближение центров, для всех $a\in Y$;
  - Повторять

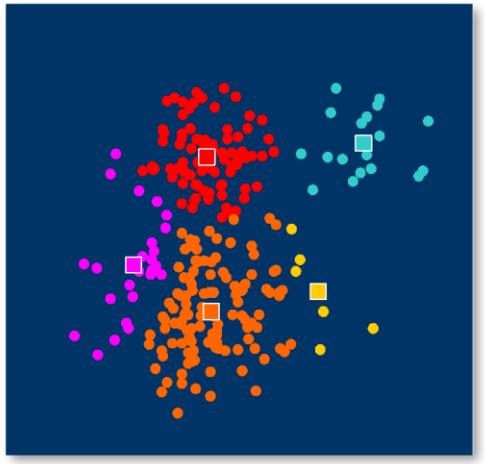
    Отнести каждый x_i к ближайшему центру:

$$a_i := \arg\min_{a\in Y}\|x_i-\mu_a\|$$

    Вычислить новые положения центров:

$$\mu_a := \frac{\sum_{i=1}^{l}[a_i=a]x_i}{\sum_{i=1}^{l}[a_i=a]}, a\in Y;$$

  - Пока a_i не перестанут изменяться.



Метод K-средних (K-means) для кластеризации в векторном пространстве

- Минимизация суммы квадратов внутрикластерных расстояний:

$$\sum_{i=1}^{l}\left\|x_i-\mu_{a_i}\right\|^2 \to \min_{\{a_i\},\{\mu_a\}}, \quad \|x_i-\mu_a\|^2=\sum_{j=1}^{n}\left(f_j(x_i)-\mu_{aj}\right)^2$$

- Алгоритм Ллойда
  - Вход: $X^l, K=|Y|$; выход: центры кластеров $\mu_a, a\in Y$;
  - $\mu_a :=$ начальное приближение центров, для всех $a\in Y$;
  - Повторять

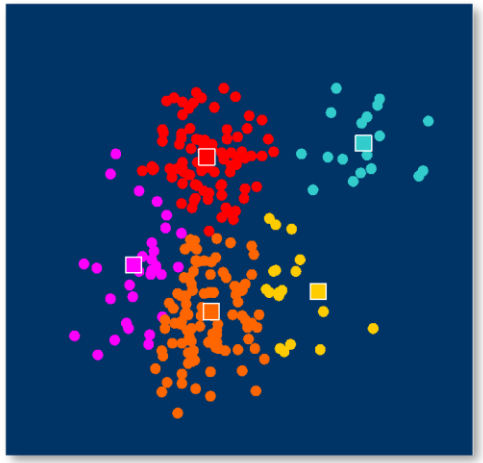
    Отнести каждый x_i к ближайшему центру:

$$a_i := \arg\min_{a\in Y}\|x_i-\mu_a\|$$

    Вычислить новые положения центров:

$$\mu_a := \frac{\sum_{i=1}^{l}[a_i=a]x_i}{\sum_{i=1}^{l}[a_i=a]}, a\in Y;$$

  - Пока a_i не перестанут изменяться.



Метод K-средних (K-means) для кластеризации в векторном пространстве

- Минимизация суммы квадратов внутрикластерных расстояний:

$$\sum_{i=1}^{l}\left\|x_i-\mu_{a_i}\right\|^2 \to \min_{\{a_i\},\{\mu_a\}}, \quad \|x_i-\mu_a\|^2=\sum_{j=1}^{n}\left(f_j(x_i)-\mu_{aj}\right)^2$$

- Алгоритм Ллойда
  - Вход: $X^l, K=|Y|$; выход: центры кластеров $\mu_a, a\in Y$;
  - $\mu_a :=$ начальное приближение центров, для всех $a\in Y$;
  - Повторять

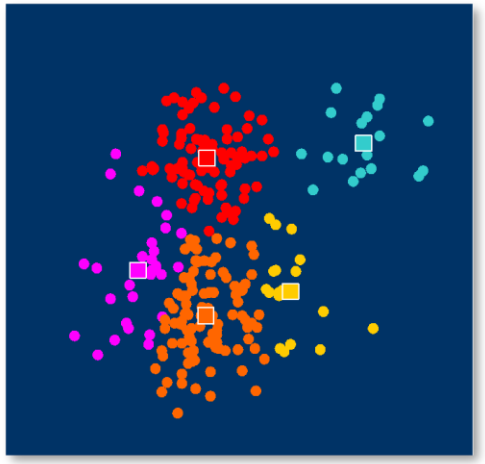
    Отнести каждый x_i к ближайшему центру:

$$a_i := \arg\min_{a\in Y}\|x_i-\mu_a\|$$

    Вычислить новые положения центров:

$$\mu_a := \frac{\sum_{i=1}^{l}[a_i=a]x_i}{\sum_{i=1}^{l}[a_i=a]}, a\in Y;$$

  - Пока a_i не перестанут изменяться.



Метод K-средних (K-means) для кластеризации в векторном пространстве

- Минимизация суммы квадратов внутрикластерных расстояний:

$$\sum_{i=1}^{l}\left\|x_i-\mu_{a_i}\right\|^2 \to \min_{\{a_i\},\{\mu_a\}}, \quad \|x_i-\mu_a\|^2=\sum_{j=1}^{n}\left(f_j(x_i)-\mu_{aj}\right)^2$$

- Алгоритм Ллойда
  - Вход: $X^l, K=|Y|$; выход: центры кластеров $\mu_a, a\in Y$;
  - $\mu_a :=$ начальное приближение центров, для всех $a\in Y$;
  - Повторять

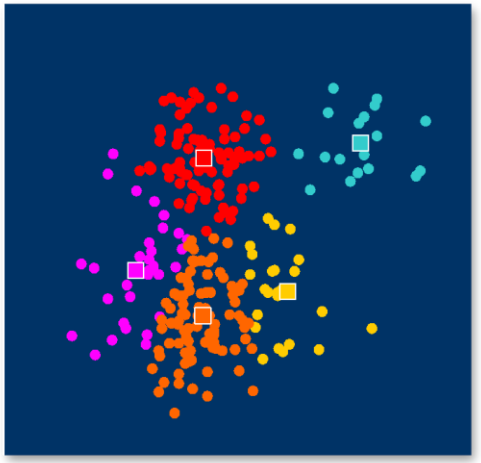
    Отнести каждый x_i к ближайшему центру:

$$a_i := \arg\min_{a\in Y}\|x_i-\mu_a\|$$

    Вычислить новые положения центров:

$$\mu_a := \frac{\sum_{i=1}^{l}[a_i=a]x_i}{\sum_{i=1}^{l}[a_i=a]}, a\in Y;$$

  - Пока a_i не перестанут изменяться.



Метод K-средних (K-means) для кластеризации в векторном пространстве

- Минимизация суммы квадратов внутрикластерных расстояний:

$$\sum_{i=1}^{l}\left\|x_i-\mu_{a_i}\right\|^2 \to \min_{\{a_i\},\{\mu_a\}}, \quad \|x_i-\mu_a\|^2=\sum_{j=1}^{n}\left(f_j(x_i)-\mu_{aj}\right)^2$$

- Алгоритм Ллойда
  - Вход: $X^l, K=|Y|$; выход: центры кластеров $\mu_a, a\in Y$;
  - $\mu_a :=$ начальное приближение центров, для всех $a\in Y$;
  - Повторять

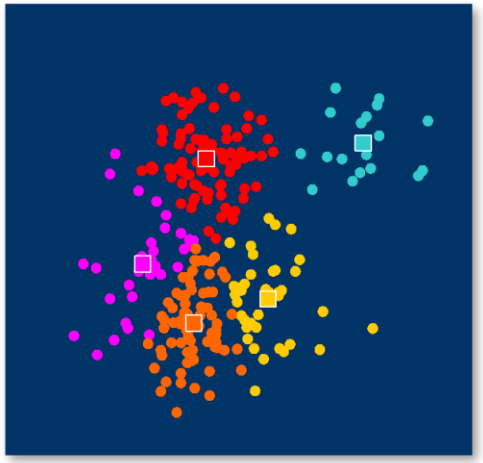
    Отнести каждый x_i к ближайшему центру:

$$a_i := \arg\min_{a\in Y}\|x_i-\mu_a\|$$

    Вычислить новые положения центров:

$$\mu_a := \frac{\sum_{i=1}^{l}[a_i=a]x_i}{\sum_{i=1}^{l}[a_i=a]}, a\in Y;$$

  - **Пока a_i не перестанут изменяться.**



Метод K-средних – упрощение EM-алгоритма для GMM

- EM-алгоритм: максимизация правдоподобия для разделения смеси гауссиан (GMM, Gaussian Mixture Model)
  - Начальное приближение w_a, μ_a, Σ_a для всех $a \in Y$
  - Повторять
    - E-шаг: отнести каждый x_i к ближайшим центрам:

$$g_{ia} := P(a|x_i) \equiv \frac{w_a p_a(x_i)}{\sum_y w_y p_y(x_i)}, a \in Y, i = 1, \dots, l;$$

$$a_i := argmax_{a \in Y} g_{ia}, i = 1, \dots, l;$$

    - M-шаг: вычислить новые положения центров:

$$\mu_{ad} := \frac{1}{l w_a} \sum_{i=1}^{l} g_{ia} f_d(x_i), a \in Y, d = 1, \dots, n;$$

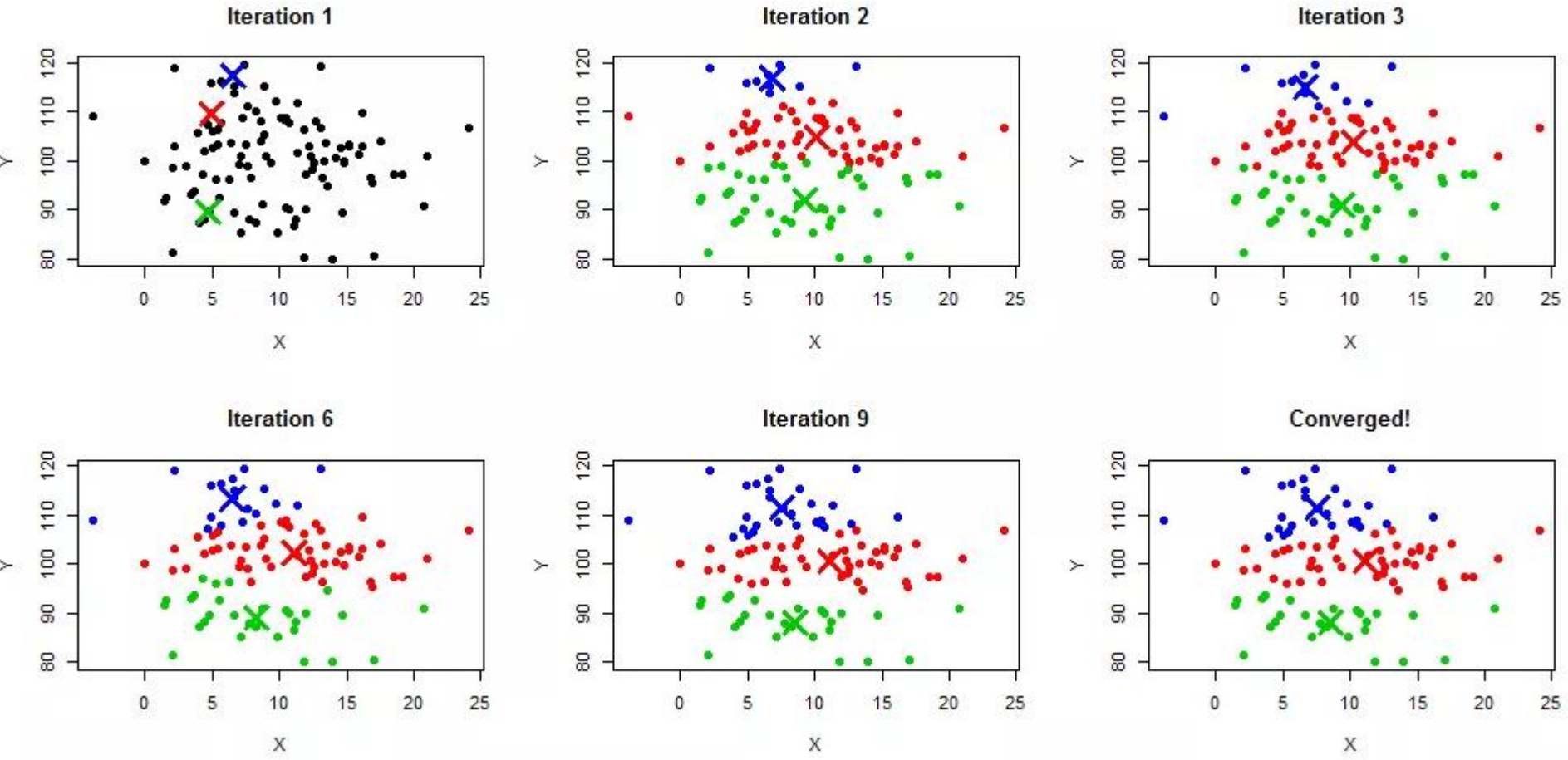
$$\sigma_{ad}^2 := \frac{1}{l w_a} \sum_{i=1}^{l} g_{ia} (f_d(x_i) - \mu_{ad})^2, a \in Y, d = 1, \dots, n; \; w_a := \frac{1}{l} \sum_{i=1}^{l} g_{ia}, a \in Y$$

  - Пока a_i не перестанут изменяться.

Особенности K-Means

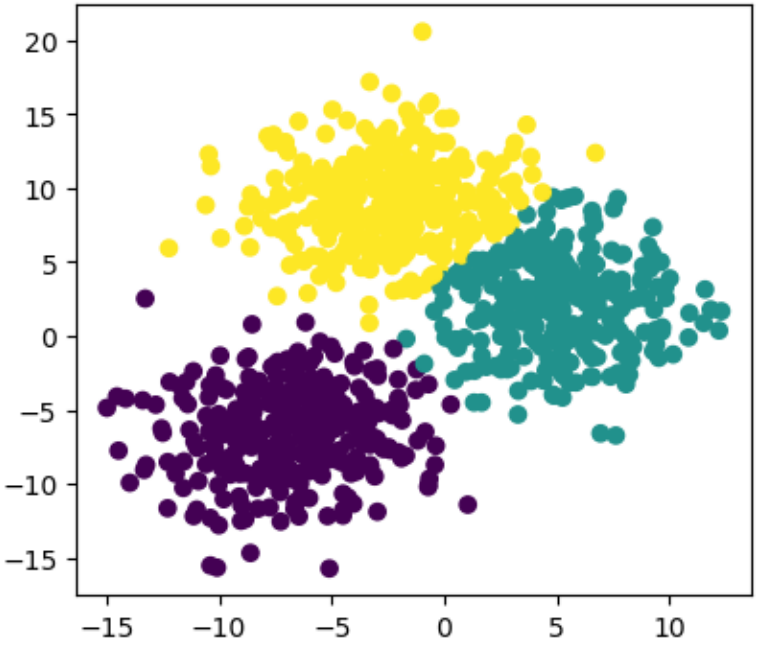
- GMM-EM vs K-means:
  - Принадлежность: GMM-EM – нечеткая (мягкая), K-means – строгая
  - Форма кластера: GMM-EM – эллиптическая (адаптируется) , K-means – сферическая (не адаптируется)
  - Упрощение GMM-EM (усложнение k-means) – сферический или выровненные по осям гауссианы, жесткий E-шаг
- Жадный алгоритм - локальный экстремум:
  - Глобальный можно искать «разумным» перебором: множественная инициализация, имитация отжига, генетические алгоритмы и т.д.
- Недостатки:
  - Числовые признаки (иначе как найти центр?)
  - Необходимо задавать K заранее (есть методы «отбора» K)
  - Чувствительность к шуму и выбросам, кластеры сферической формы

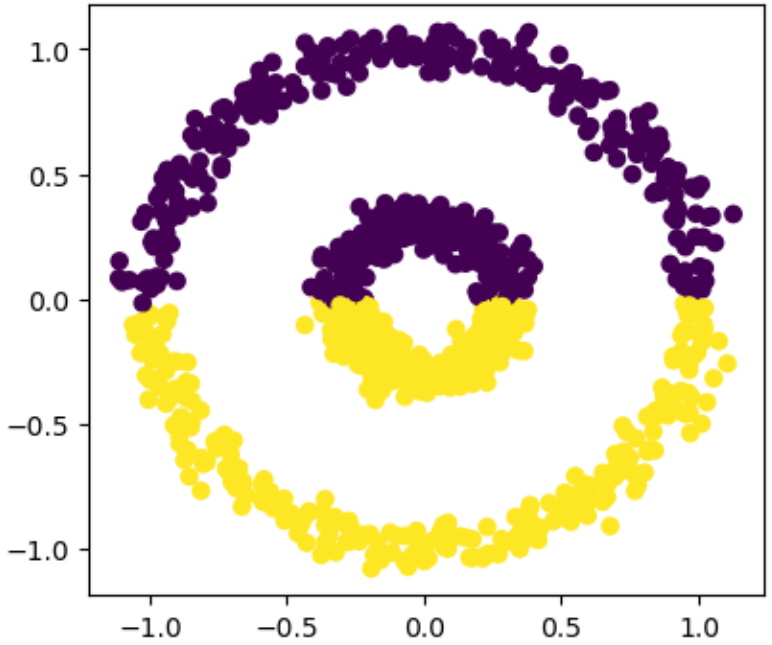
k-Means алгоритм для профилирования (не кластеризации)



Пример использования

```
from sklearn.cluster import KMeans

fig, axes = plt.subplots(ncols=2, figsize=(10,4))
km_bl = KMeans(n_clusters=3)
axes[0].scatter(X_blob[:, 0], X_blob[:, 1],
                c=km_bl.fit_predict(X_blob))

km_cl = KMeans(n_clusters=2)
axes[1].scatter(X_cl[:, 0], X_cl[:, 1],
                c=km_cl.fit_predict(X_cl))
```





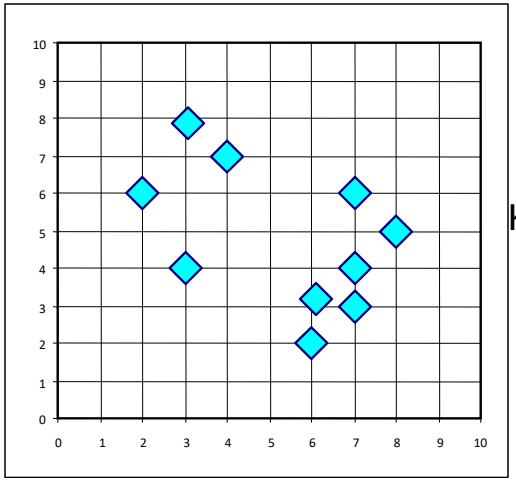
K-Medoids

- K-Medoids:
  - Идея: вместо мат. ожидания кластера ищется представительный («наиболее центральный») объект – medoid
  - Процесс: случайная инициализация, SWAP шаг - перебор и переход к новому медоиду, который максимально улучшает целевую функцию:

$$\min_{C_i \cap C_j = \emptyset, \cup C_k = X} \sum_{k=1}^{K} \sum_{x_j \in C_k} \rho(\mu_k, x_j), \forall k: \boldsymbol{\mu_k} \in \boldsymbol{X}$$

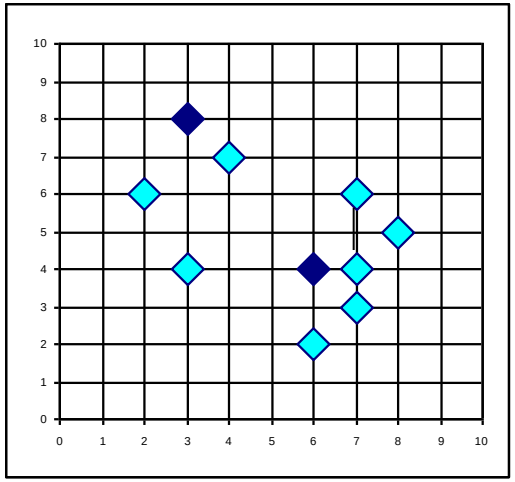
- Достоинства:
  - Робастный, реальные прототипы, не только числовые признаки
- Недостатки:
  - жадный
  - не масштабируется и вычислительно неэффективный $O(K(l-K)^2)$ для каждой итерации
  - все еще сферические кластеры

Алгоритм K-Medoids

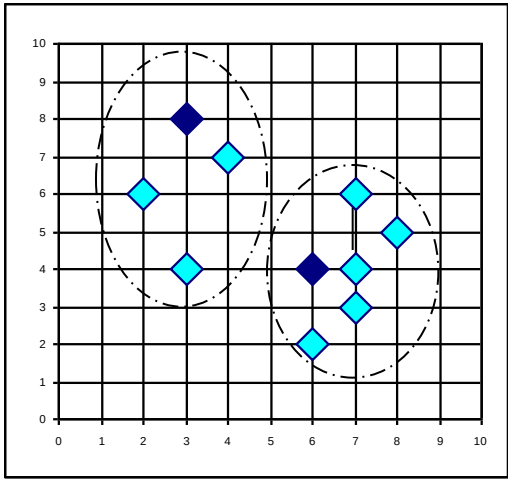


K=2

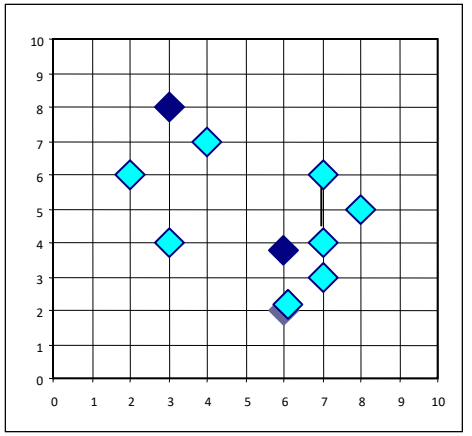
Случайн. K medoids



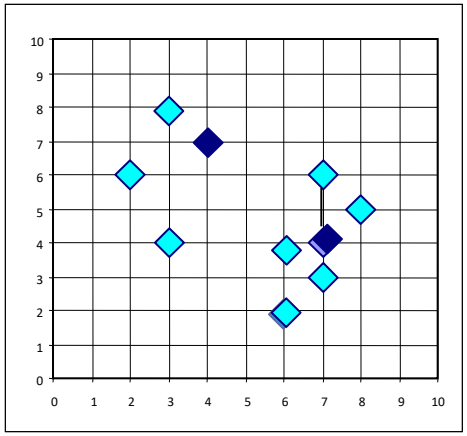
Каждый объект к ближ. medoid



Перебор всех кандидатов на замену medoid



Расчет стоимости перехода (total cost of swapping)

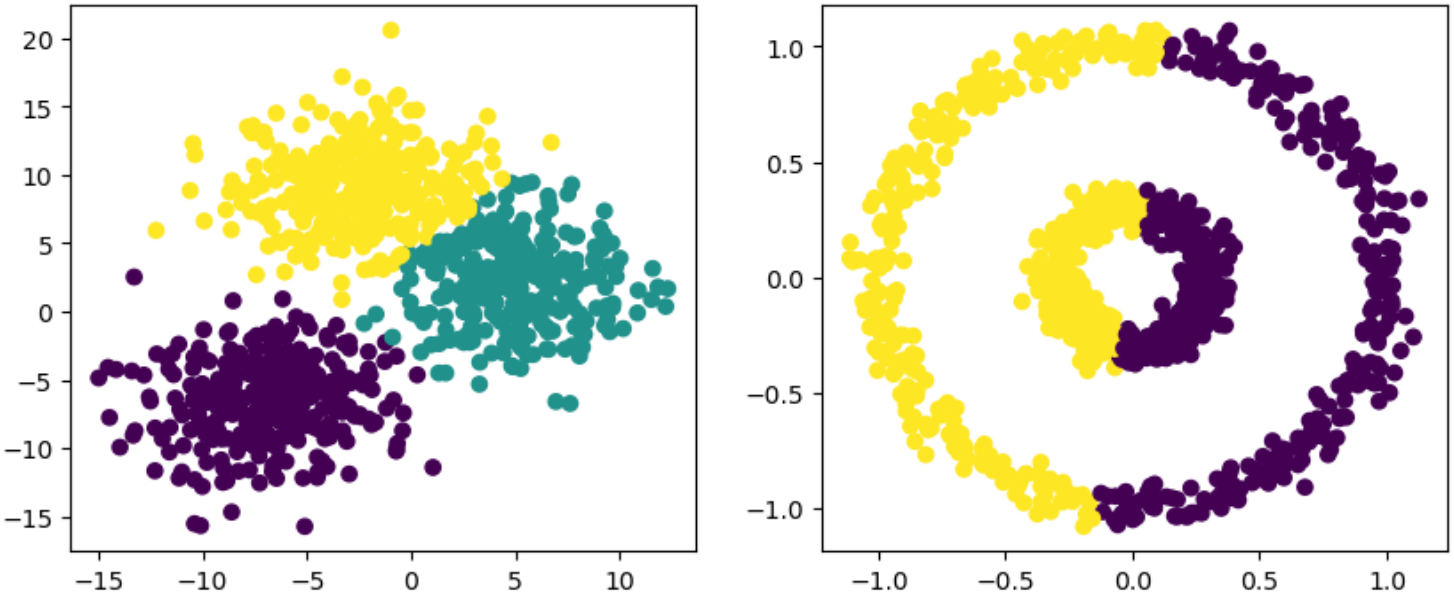


Если качество кластеризац. улучш., то переход к нов. medoid

Пока есть изменения

Пример использования

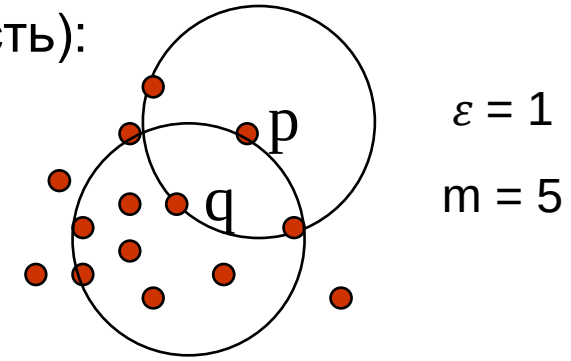
```
# Расширенный функционал scikit-learn
# Установка: "pip install scikit-learn-extra"
from sklearn_extra.cluster import KMedoids

fig, axes = plt.subplots(ncols=2, figsize=(10,4))
km_bl = KMedoids(n_clusters=3)
axes[0].scatter(X_blob[:, 0], X_blob[:, 1],
                c=km_bl.fit_predict(X_blob))

km_cl = KMedoids(n_clusters=2)
axes[1].scatter(X_cl[:, 0], X_cl[:, 1],
                c=km_cl.fit_predict(X_cl))
```



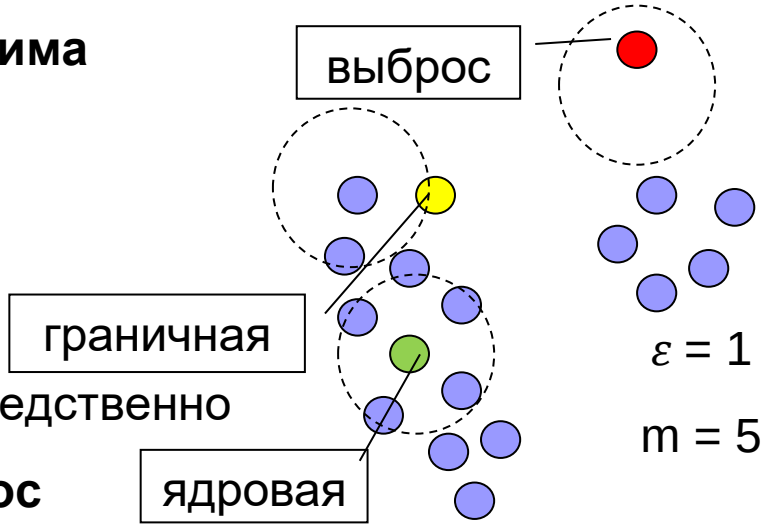
DBSCAN: основные определения

- Важные параметры (что есть «плотная» область):
  - ε: радиус области поиска ближайших соседей
  - m: минимальное число соседей в ε-области
- Множество ближайших соседей точки $p \in U$:

$$U_\varepsilon(p) = \{q \in U | \rho(p,q) < \varepsilon\}$$

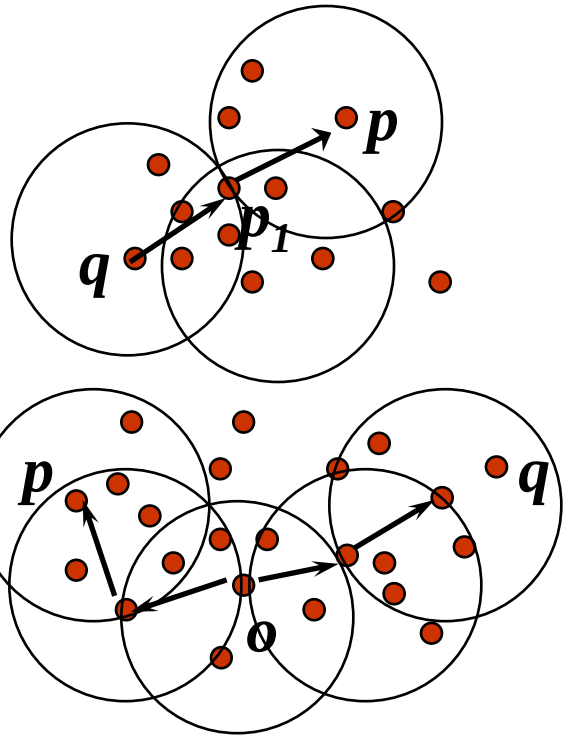


- Непосредственно достижимые точки:
  - Точка p **непосредственно достижима** из q с учетом ε, если $p \in U_\varepsilon(q)$
- Типы точек:
  - **ядровая** точка: $|U_\varepsilon(q)| \geq m$
  - **граничная** - не ядровая, но непосредственно достижима из ядровой, иначе - **выброс**



DBSCAN: основные определения

- Достижимость:
  - Точка *p* достижима из *q* с учетом ε и *m,* если существует путь $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ такой, что p_{i+1} **непосредственно достижима** из p_i
- Связность:
  - Точка *p* **связана** с *q* с учетом ε и *m,* если существует точка *o такая, что* обе точки *p* и *q* достижимы из *o* с учетом ε и *m*.
- **Кластер** определен как максимальное множество связных точек



Процедура:

1. Выбор произвольный точки *p* и всех связанных с *p*.
2. Если *p* – ядровая, то новый кластер сформирован, если *p* граничная или выброс, то обработка следующей точки пока не будут выбраны все

Алгоритм кластеризации DBSCAN

- Вход: выборка $X^l = \{x_1, \dots, x_l\}$; параметры ε и m;
- Выход: разбиение выборки на кластеры и шумовые выбросы;
- $U := X^l$ – **непомеченные**; $a := 0$
- Пока в выборке есть непомеченные точки, $U \neq \emptyset$:
  - Взять *случайную* точку $x \in U$;
  - Если $|U_\varepsilon(x)| < m$, то пометить x как, возможно, шумовой;
  - Иначе

    Создать новый кластер: $K := U_\varepsilon(x); a := a + 1$

    Для всех $x' \in K$, не помеченных или шумовых
    - Если $|U_\varepsilon(x')| \geq m$ то $K := K \cup U_\varepsilon(x')$;
    - Иначе пометить x' как граничный кластера K;

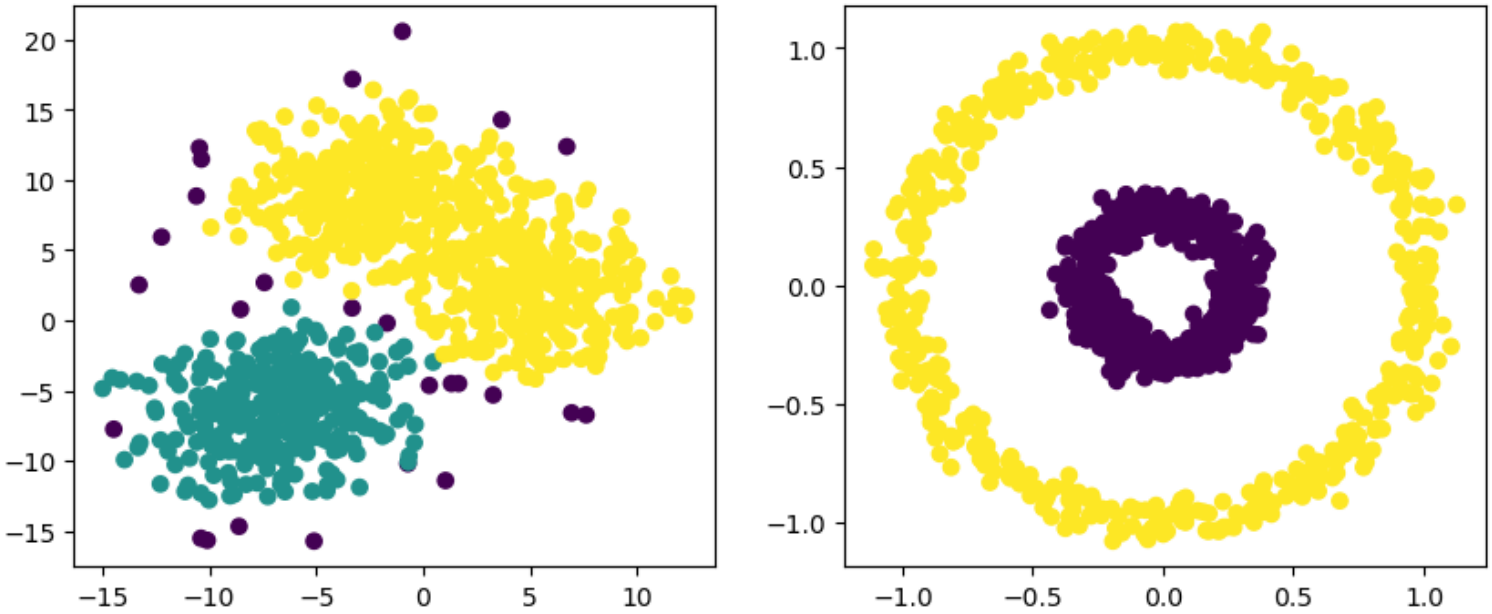
    $a_i := a$ для всех $x_i \in K$; $U := U \backslash \mathrm{K}$

Непараметрическая кластеризация на основе связности DBSCAN

- Density-Based Spatial Clustering of Applications with Noise
- Основные свойства:
  - Произвольная форма кластеров (без прототипов и вероятностных моделей)
  - Работа в условиях шума (объекты делятся на ядровые, шумовые и граничные)
  - Один проход базы, всего $O(l^2)$ операций, а если использовать поисковый индекс для ближайших соседей, то $O(l \cdot \log(l))$
- Недостатки:
  - нужны не интуитивные параметры для «тонкой настройки»
  - жадные, нестабильные, не интерпретируемые модели кластеризации

Пример использования

```
from sklearn.cluster import DBSCAN

fig, axes = plt.subplots(ncols=2, figsize=(10,4))
gbsc_bl = DBSCAN(eps=1.5, min_samples=5)
axes[0].scatter(X_blob[:, 0], X_blob[:, 1],
                c=gbsc_bl.fit_predict(X_blob))

gbsc_cl = DBSCAN(eps=0.1)
axes[1].scatter(X_cl[:, 0], X_cl[:, 1],
                c=gbsc_cl.fit_predict(X_cl))
```

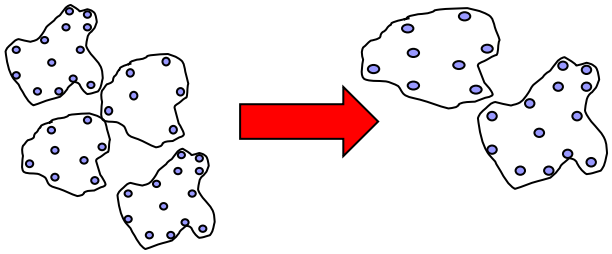


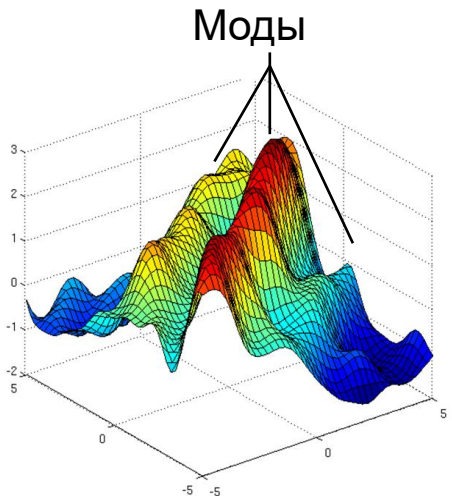
Непараметрическая кластеризация на основе ядерной оценки плотности

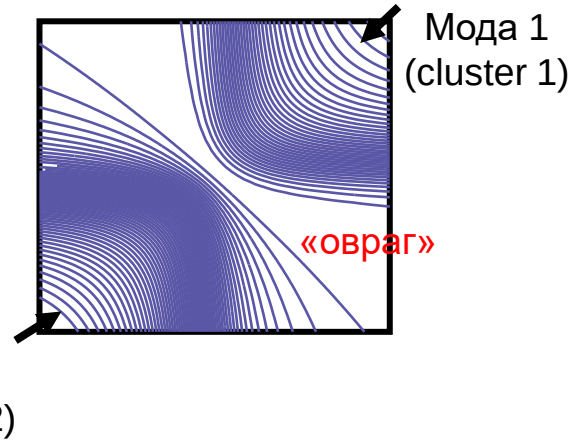
- Восстановление плотности:

$$\hat{p}_h(x) = \frac{1}{lV(h)} \sum_{i=1}^{l} K\left(\frac{\rho(x, x_i)}{h}\right)$$

- Основная идея кластеризации:
  - Локальные пики (моды) плотности – кандидаты в «ядровые точки» кластеров
  - Надо найти пики (градиентом по $\hat{p}_h(x)$)
  - Привязать к ним ближайшие наблюдения (с учетом окрестности) при необходимости «склеить» близкие кластеры





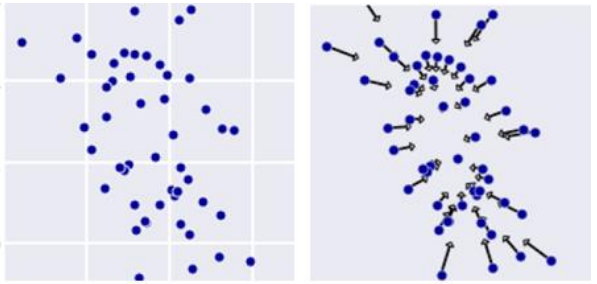


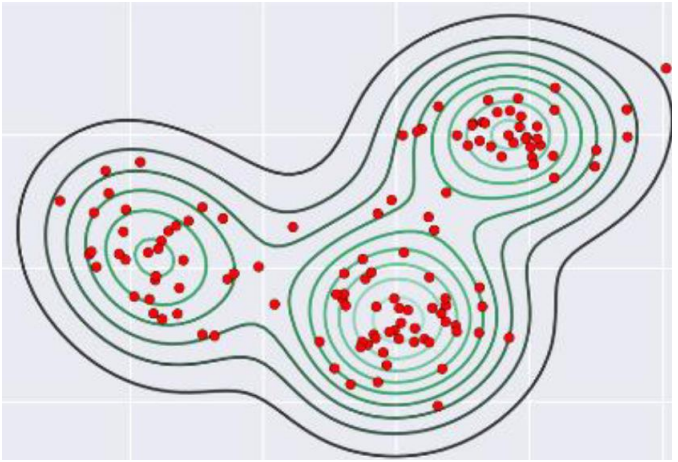
Непараметрическая кластеризация на основе ядерной оценки плотности

- Много методов с разными предположениями и ограничениями:
  - DENsity--based CLUstEring, ModeClus (SAS), Mean shift, …
- Основная идея Mean shift (в цикле):
  - Берем x и ее окрестность

    $U_\varepsilon(x) = \{x \in U | \rho(x, x') < \varepsilon\}$
  - Найдем центр масс окрестности

    $$m(x) = \frac{\sum_{x_i \in U_\varepsilon(x)} K(x - x_i) x_i}{\sum_{x_i \in U_\varepsilon(x)} K(x - x_i)}$$
  - Сдвигаем точку в сторону центра масс окрестности с шагом η: $x \leftarrow x + \eta(m(x) - x)$
- или сразу $x \leftarrow x + \eta \nabla \hat{p}_\varepsilon(x)$



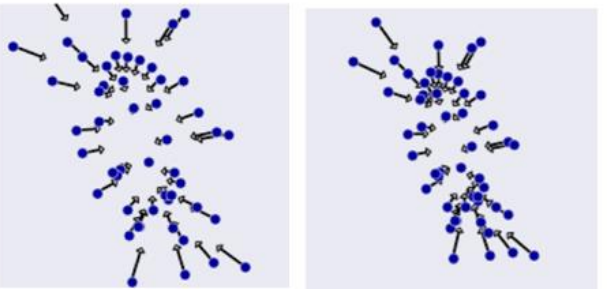


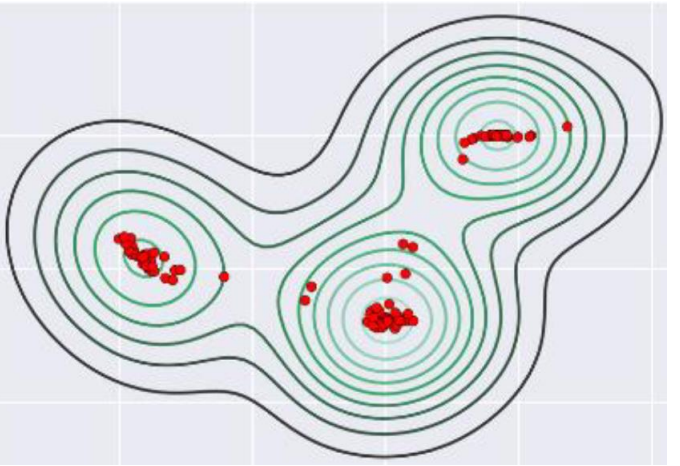
Непараметрическая кластеризация на основе ядерной оценки плотности

- Много методов с разными предположениями и ограничениями:
  - DENsity--based CLUstEring, ModeClus (SAS), Mean shift, …
- Основная идея Mean shift (в цикле):
  - Берем x и ее окрестность

    $U_\varepsilon(x) = \{x \in U | \rho(x, x') < \varepsilon\}$
  - Найдем центр масс окрестности

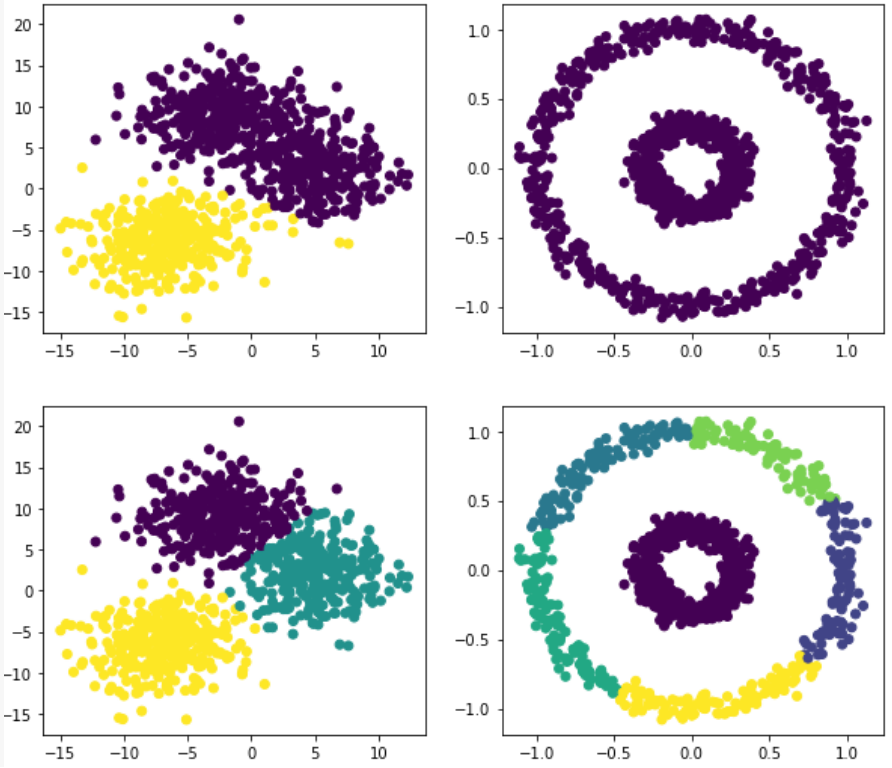
    $$m(x) = \frac{\sum_{x_i \in U_\varepsilon(x)} K(x - x_i) x_i}{\sum_{x_i \in U_\varepsilon(x)} K(x - x_i)}$$
  - Сдвигаем точку в сторону центра масс окрестности с шагом η: $x \leftarrow x + \eta(m(x) - x)$
- или сразу проще $x \leftarrow x + \eta \nabla \hat{p}_\varepsilon(x)$





Пример использования

```
from sklearn.cluster import MeanShift
from sklearn.cluster import estimate_bandwidth

fig, axes = plt.subplots(ncols=2, figsize=(10,4))
gbsc_bl = MeanShift()
axes[0].scatter(X_blob[:, 0], X_blob[:, 1],
                c=gbsc_bl.fit_predict(X_blob))

gbsc_cl = MeanShift()
axes[1].scatter(X_cl[:, 0], X_cl[:, 1],
                c=gbsc_cl.fit_predict(X_cl))

fig, axes = plt.subplots(ncols=2, figsize=(10,4))
gbsc_bl = MeanShift(bandwidth = estimate_bandwidth(X_blob,
        quantile=0.2, n_samples=500))
axes[0].scatter(X_blob[:, 0], X_blob[:, 1],
                c=gbsc_bl.fit_predict(X_blob))

gbsc_cl = MeanShift(bandwidth = estimate_bandwidth(X_cl,
        quantile=0.2, n_samples=500))
axes[1].scatter(X_cl[:, 0], X_cl[:, 1],
                c=gbsc_cl.fit_predict(X_cl))
```



Параметрическая вероятностная кластеризация – обычно EM алгоритм

- Вход: $X^l = \{x_1, \dots, x_l\}$, k – число кластеров, $\varphi(x_i, \theta)$ – параметрическая вероятностная модель в кластере;
- Выход: $(w_j,\ \theta_j)_{j=1}^{k}$ – априорные вероятности кластеров и параметры вероятностных моделей в кластерах;
- Инициализировать $(\theta_j)_{j=1}^{k}$, $w_j := \frac{1}{k}$, и повторять:
  - **E-шаг**: расчет $P(j|x_i)$ ф-ции принадлежности наблюдения x_i кластеру j:
$$g_{ij} := \frac{w_j \varphi(x_i, \theta_j)}{\sum_{s=1}^{k} w_s \varphi(x_i, \theta_s)}$$
  - **M-шаг**: пересчет априорных вероятностей кластеров и параметры вероятностных моделей в каждом:
$$w_j := \frac{1}{l}\sum_{i=1}^{l} g_{ij}\,, \theta_j := \arg\max_{\theta} \sum_{i=1}^{l} \boldsymbol{g_{ij}} \ln \varphi(x_i, \theta)$$

Gaussian Mixture Model (GMM)

- Вход: $X^l = \{x_1, \dots, x_l\}, \quad k;$
- Выход: $\left(w_j, \boldsymbol{\mu_j}, \boldsymbol{\Sigma_j}\right)_{j=1}^{k}$ – параметры смеси гауссиан;
- Инициализировать $\left(\boldsymbol{\mu_j}, \boldsymbol{\Sigma_j}\right)_{j=1}^{k}, w_j := \frac{1}{k}$
- Повторять
  - E-шаг (expectation): для всех $i = 1, .., l, \quad j = 1, \dots, k$

$$g_{ij} := \frac{w_j \boldsymbol{N}(\boldsymbol{x_i}; \boldsymbol{\mu_j}, \boldsymbol{\Sigma_j})}{\sum_{s=1}^{k} w_s \boldsymbol{N}(\boldsymbol{x_i}; \boldsymbol{\mu_s}, \boldsymbol{\Sigma_s})}$$

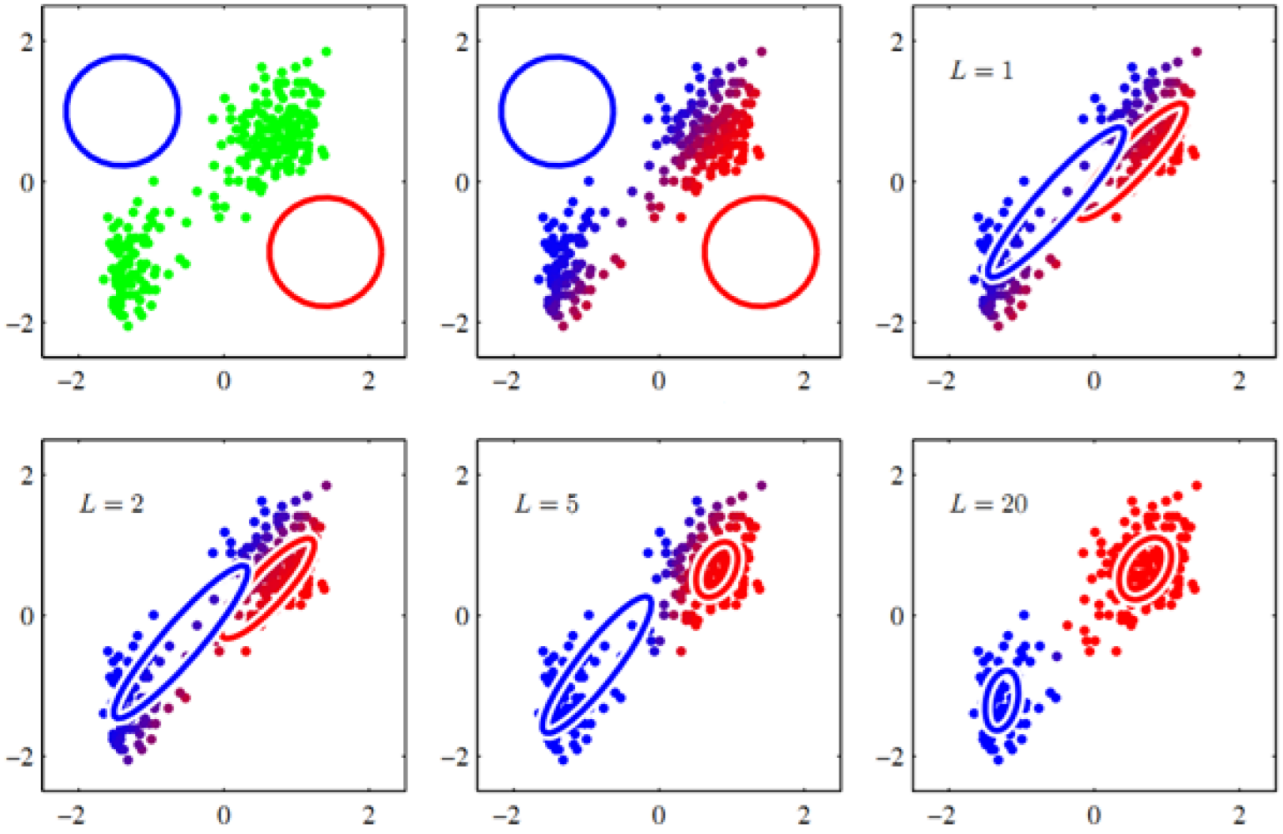
  - M-шаг (maximization): для всех $j = 1, \dots, k$

$$w_j := \frac{1}{l}\sum_{i=1}^{l} g_{ij}\,, \boldsymbol{\mu_j} := \frac{\mathbf{1}}{\boldsymbol{l w_j}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{l}} \boldsymbol{g_{ij} x_i}\,, \boldsymbol{\Sigma_j} := \frac{\mathbf{1}}{\boldsymbol{l w_j}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{l}} \boldsymbol{g_{ij}}\left(\boldsymbol{x_i} - \boldsymbol{\mu_j}\right)\left(\boldsymbol{x_i} - \boldsymbol{\mu_j}\right)^{\boldsymbol{T}}$$

- Пока (w_j, μ_j, Σ_j) и / или g_{ij} не сошлись.

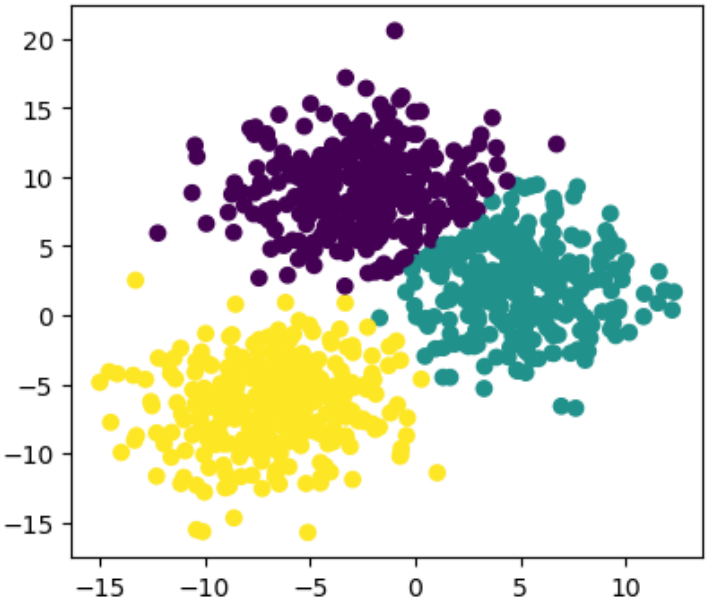
Демо-пример

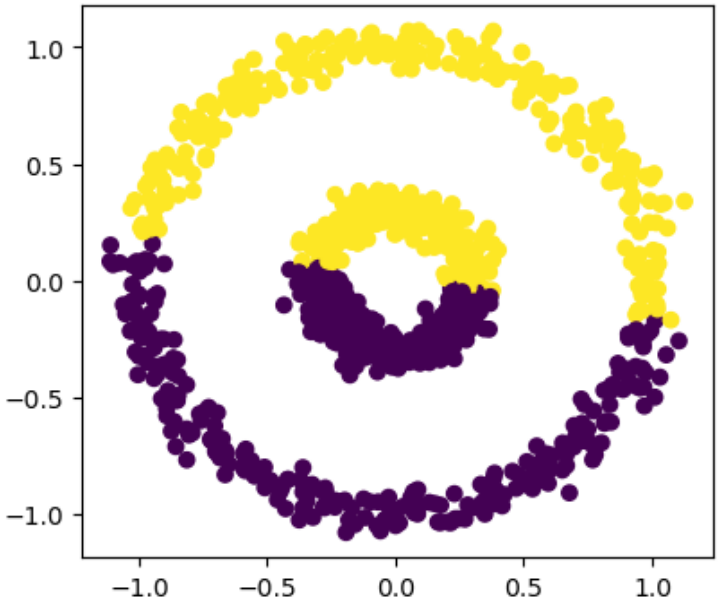
- Две гауссовские компоненты $k = 2$ в пространстве $X = \mathbb{R}^2$.
- Расположение компонент в зависимости от номера итерации L



Пример использования

```
from sklearn.mixture import GaussianMixture

fig, axes = plt.subplots(ncols=2, figsize=(10,4))
gmm_bl = GaussianMixture(n_components=3)
axes[0].scatter(X_blob[:, 0], X_blob[:, 1],
                c=gmm_bl.fit_predict(X_blob))

km_cl = GaussianMixture(n_components=2)
axes[1].scatter(X_cl[:, 0], X_cl[:, 1],
                c=km_cl.fit_predict(X_cl))
```

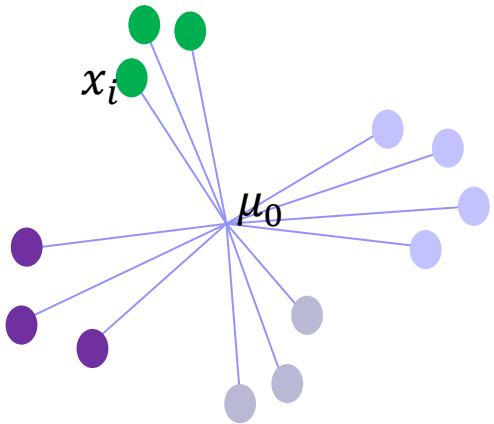


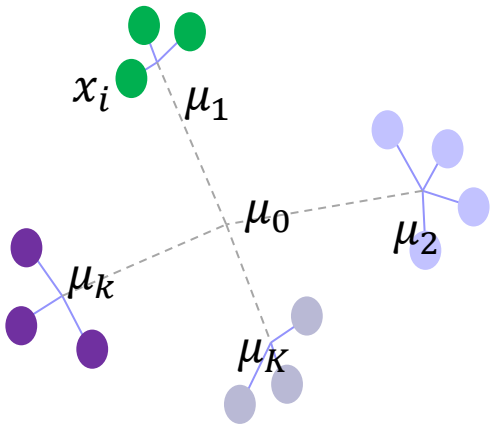


Определение числа кластеров

- Перебор моделей:
  - последовательно меняя число кластеров в исходном алгоритме или по последовательной кластеризации (обычно иерархической)
  - выбор числа кластеров по некоторому критерию, (CCC, Pseudo-F, др.)
- Обозначения:
  - μ_0 - центр масс распределения всей выборки, μ_k- прототип кластера k
  - Total Sum of Squares: $TSS = \sum_{i=1}^{l} \rho(\mu_0, x_i)^2$
  - Between-cluster Sum of Squares: $BSS = \sum_{k=1}^{K} \rho(\mu_k, \mu_0)^2$
  - Within-cluster Sum of Squares: $WSS = \sum_{k=1}^{K} \sum_{x_i \in C_k} \rho(\mu_k, x_i)^2$

$$TSS = WSS + BSS$$



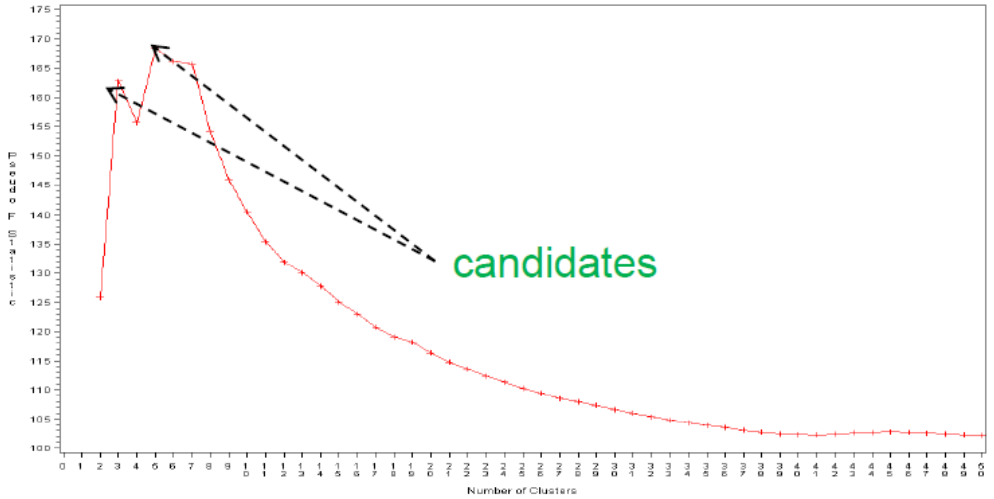


Определение числа кластеров с помощью pseudo-F

- Псевдо критерий Фишера (Calinski and Habarasz, 1974):
  - эвристическое применение статистики Фишера
  - отношение «средней» (деленной на число степеней свободы) общей вариации к «средней» межкластерной

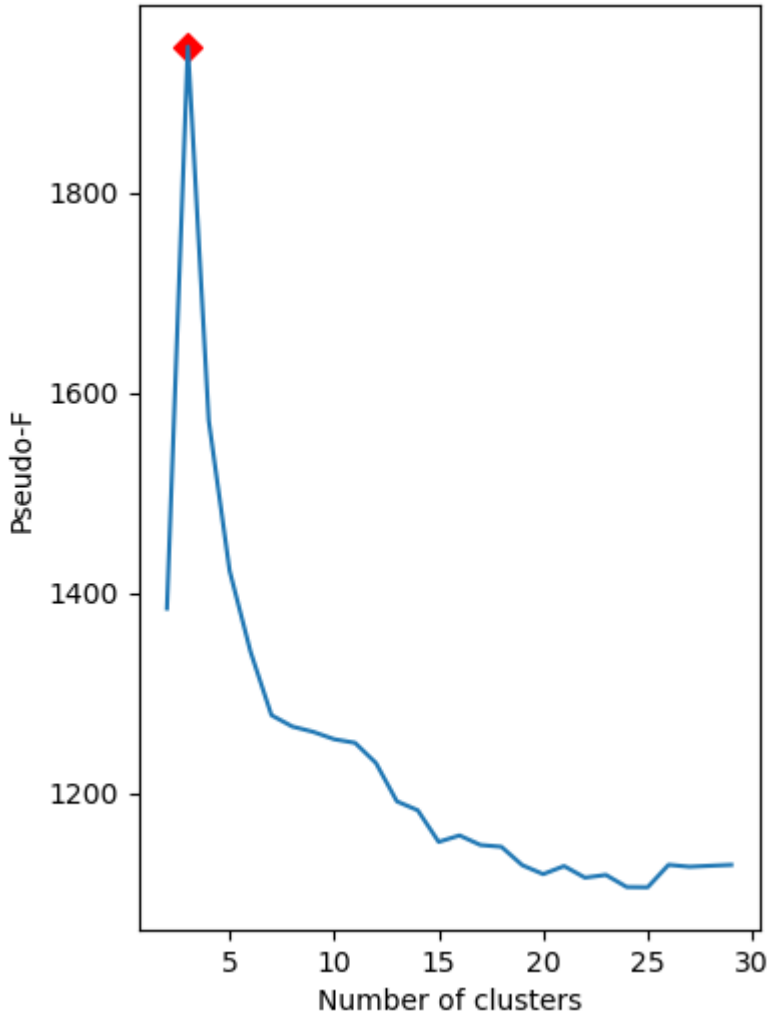
$$F = \frac{BSS/(l-1)}{WSS/(l-K)}$$

  - Исходный статистический тест в дисперсионном анализе применяется для проверки гипотезы о равенстве групповых средних, чем больше F тем больше отличаются кластеры
  - **локальные максимумы** – модели-кандидаты



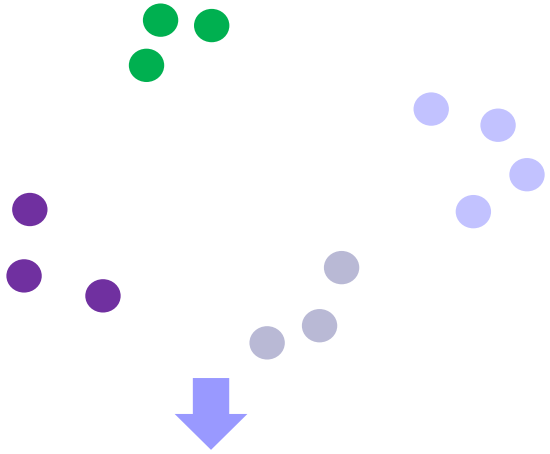
Пример использования

```python
def sum_dist_to_center(X):
    center = np.mean(X, axis = 0)
    return ((X - center)**2).sum()

def choose_num_clusters(X, max_clust = 30):
    N = X.shape[0]
    Q = sum_dist_to_center(X)
    pseudo_f = np.array([])
    for G in range(2, max_clust):
        clustering = KMeans(n_clusters = G).fit(X)
        W = 0
        for l in range(G):
            elems = X[clustering.labels_ == l]
            W += sum_dist_to_center(elems)
        fisher_stat = ((Q - W)/(G - 1))/(W/(N - G))
        pseudo_f = np.append(pseudo_f, fisher_stat)
    plt.plot(range(2, max_clust), pseudo_f)
    ind_best_clust = np.argmax(pseudo_f)
    plt.scatter(ind_best_clust + 2,
                pseudo_f[ind_best_clust],
                color="r", marker="D", s=50)
    plt.xlabel("Number of clusters")
    plt.ylabel("Pseudo-F")
    return ind_best_clust + 2

k = choose_num_clusters(X_blob)
```

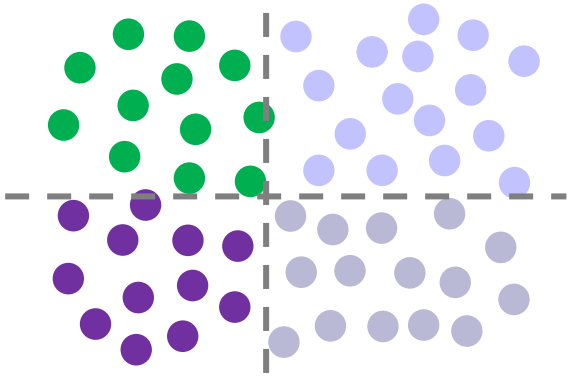


Определение числа кластеров по CCC

Разбиение исходной выборки



Разбиение выборки, из равномерного распределения



$$\boldsymbol{R}^2 = \frac{BSS}{TSS} \Rightarrow CCC = \ln\left[\frac{1-\mathbb{E}(\boldsymbol{R}^2)}{1-\boldsymbol{R}}\right] * k(\mathbb{E}(\boldsymbol{R}^2)) \Leftarrow \mathbb{E}(\boldsymbol{R}^2) = \frac{BSS}{TSS}$$

$\boldsymbol{CCC}$ – **Cubic Cluster Criterion**

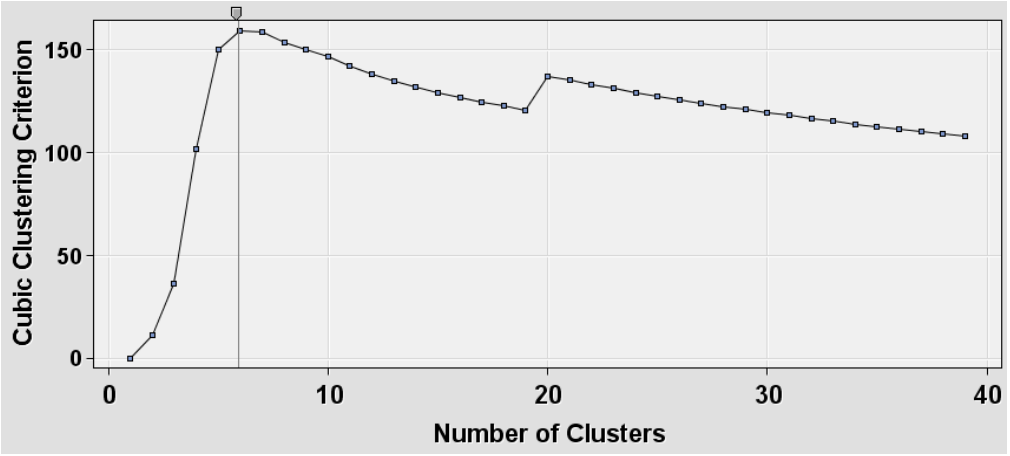
- Значительное упрощение (недостаток):
  - для сравнения всегда используется равномерное распределение, причем используются кластеры одинакового размера с границами параллельными осям. Эти недостатки устранены в ABC (*см. далее*).

Процедура применения CCC

- Процедура:
  - Случайно выбираются центры для большого числа кластеров
  - Все наблюдения объединяются в эти случайные кластеры
  - Решается задач восходящей иерархической кластеризации, на каждом шаге считается CCC
  - Выбирается число кластеров, соответствующее первому локальному пику:

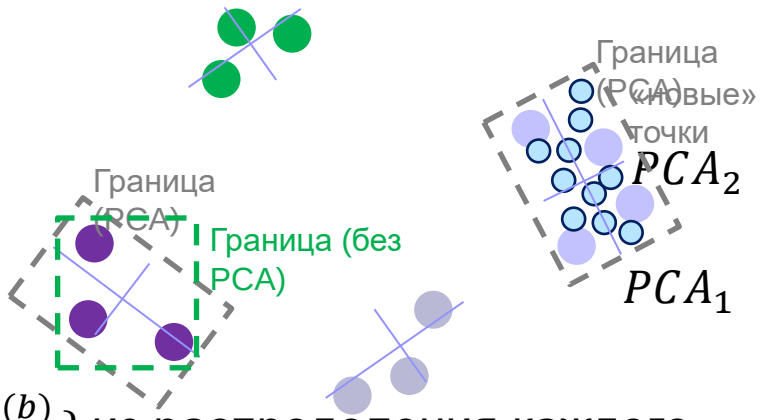
Дополнительные эвристики:

- $CCC > 2$ – решение можно считать хорошим
- $0 < CCC \leq 2$ – решение требует доп. проверки
- $CCC \leq 0$ – в выборке присутствуют выбросы



Определение числа кластеров – ABC

- Процедура:
  - Строим разбиение исходной выборки (на K кластеров) считаем WSS_K
  - Определяем границы кластеров (опционально используем PCA)
  - Методом Монте-Карло делаем



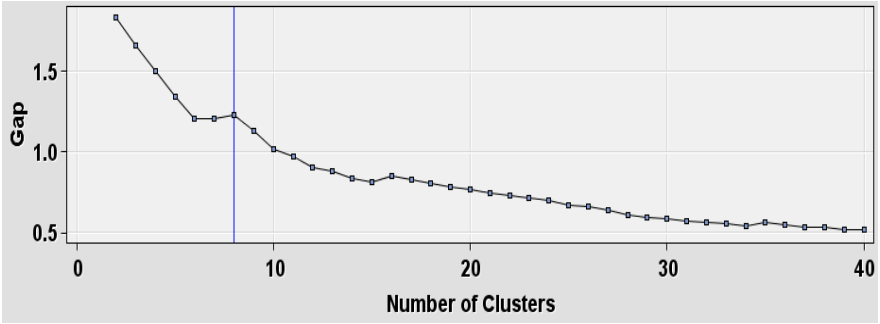
B итераций - генерируем новые точки $\{x_i^{(b)}\}$ из распределения каждого кластера (внутри границ)

  - На «новых» точках строится новое разбиение и считается:

$$WSS_K^{(b)} = \sum_{k=1}^{K} \sum_{x_i \in C_k} \rho\left(\mu_k^{(b)}, x_i^{(b)}\right)^2$$

  - Усредняются по итерациям *B*:

$$WSS_k^* = \frac{1}{B} \sum_{b=1}^{B} WSS_k^{(b)}$$



  - Считаем статистику $Gap(k) = \log\left(WSS_k^*\right) - \log(WSS_k)$ и берем 1 лок. пик

Качество кластеризации - коэффициент силуэта (анализ ошибок кластеризации)

- Распределение качества кластеризации по объектам / кластерам
- Среднее расстояние до объектов своего кластера:

$$r_i = \frac{1}{|X_{a_i}| - 1} \sum_{x \in X_{a_i} \setminus \mathrm{x}_i} \rho(x, x_i)$$

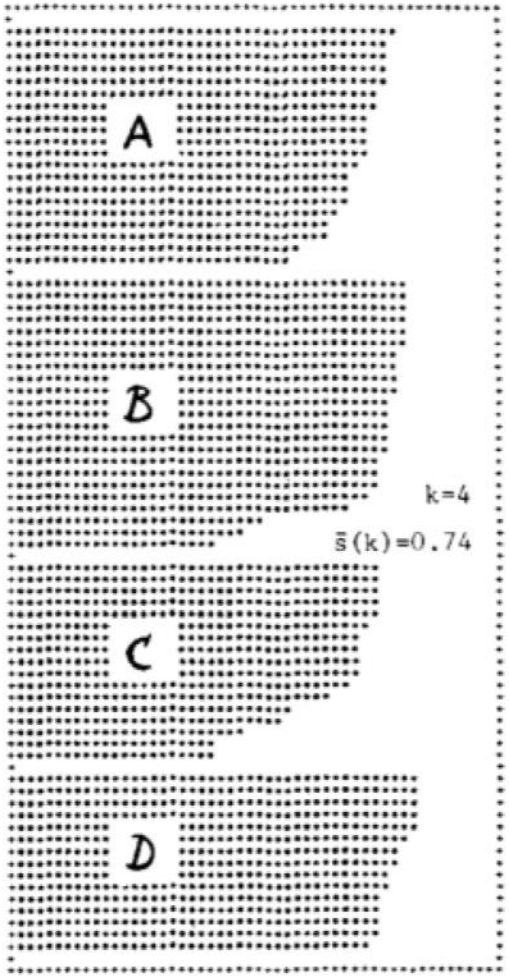
- Минимальное среднее расстояние до чужого кластера:

$$R_i = \min_{a \in Y \setminus \mathrm{a}_i} \frac{1}{|X_a|} \sum_{x \in X_a} \rho(x, x_i)$$

- Коэффициент силуэта объекта:

$$s_i = \frac{R_i - r_i}{\max(R_i, r_i)} \in [-1, +1]$$

- Интерпретация:
  - $s_i \to 1$– «свой», $s_i \to -1$– «чужой», $s_i \to 0$ - «граничный»

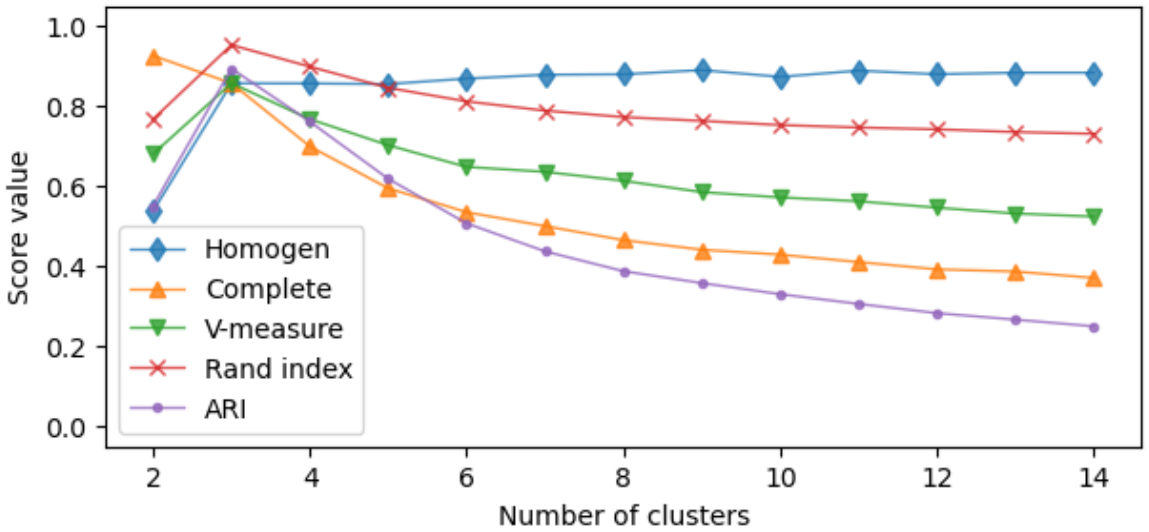


Однородность и компактность кластеризации (если есть эталоны)

- Пусть есть эталоны – истинные группы:
  - n – число наблюдений, *C* – число кластеров- эталонов, *K* – число найденных кластеров, $n_{c,k}$ - число наблюдений эталона c в кластере k
  - $H(C) = -\sum_{c=1}^{C} \frac{n_c}{n} \log(\frac{n_c}{n})$ – энтропия кластеров или эталонов (разнородность)
  - $H(C|K) = -\sum_{c=1}^{C} \sum_{k=1}^{K} \frac{n_{c,k}}{n} \log(\frac{n_{c,k}}{n_k})$ – условная энтропия эталонов или кластеров (оставшаяся разнородность эталонов после кластеризации или наоборот)
  - **Homogeneity** – кластер содержит только примеры одного эталона:
$$h = 1 - \frac{H(C|K)}{H(C)}$$
  - **Completeness** – все одинаковые эталоны относятся к одному и тому же кластеру:
$$\mathrm{c} = 1 - \frac{H(K|C)}{H(K)}$$

Пример использования

```
def apply_clustering(X, y, score_func, cl_range):
    scores = np.zeros(len(cl_range))
    for i, n_clust in enumerate(cl_range):
        cl = KMeans(n_clusters = n_clust).fit(X)
        scores[i] = score_func(y, cl.labels_)
    return scores

cl_range = range(2, 15)
plots, names = [], []

for marker, (name, func) in zip("d^vx.,", score_funcs):
    scores = apply_clustering(X_blob, y_blob, func, cl_range)
    plots.append(plt.errorbar(cl_range, scores, alpha=0.8,
                              linewidth=1, marker=marker)[0])
    names.append(name)
```



Индекс Рэнда:

$$RI = \frac{a+b}{C_2^n}$$

a – кол-во пар где C = K

b – кол-во пар где C != K

C_2^n – кол-во всех пар

Скорректированный *RI*:

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$$

V-measure – гармоническое среднее

$$v = 2\frac{hc}{h+c}$$

Точность и полнота кластеризации в сравнении с эталоном

- Обозначения:
  - $y_i \in Y_0$ – эталонная классификация объектов, $i = 1, \dots, l$,
  - Y_0 может не совпадать с Y по мощности
  - $P_i = \{k: a_k = a_i\}$ – кластер объекта x_i
  - $Q_i = \{k: y_k = y_i\}$ – эталонный класс объекта x_i
- BCubed-меры точности и полноты кластеризации:
  - $Precision = \frac{1}{l}\sum_{i=1}^{l}\frac{|P_i \cap Q_i|}{|P_i|}$ – средняя точность
  - $Recall = \frac{1}{l}\sum_{i=1}^{l}\frac{|P_i \cap Q_i|}{|Q_i|}$ – средняя полнота
  - $F_1 = \frac{1}{l}\sum_{i=1}^{l}\frac{|P_i \cap Q_i|}{|P_i|+|Q_i|}$ – средняя F_1-мера

Выводы по кластеризации

- Кластеризация:
  - Важный инструмент для решения практических задач, предобработки данных, разведочного анализа
  - Задача обучения без учителя, некорректно поставленная, нет объективной меры качества, ориентируются на NGC
- Основные типы алгоритмов кластеризации:
  - Иерархические
  - Основанные на прототипах
  - На основе связности и непараметрической оценки плотности
  - Параметрические вероятностные
- Особенности:
  - Для разных задач подходят разные алгоритмы, надо учитывать требования задачи
  - Все сильно зависит от предобработки данных, метрики сходства или различия, признакового пространства и стратегии выбора мета-параметров

Контрольный опрос