

Лекция IV

Кластеризация

Т. Е. Абильдаев

Кафедра статистического моделирования
Матмех СПбГУ

2025-04-12

Ранее мы говорили о регрессии и классификации – это наиболее известные и изученные задачи статистического, или же классического машинного, обучения. Их относят к *обучению с учителем (supervised learning)*, то есть к совокупности методов, имеющих дело с размеченными данными.

Существует также и *обучение без учителя (unsupervised learning)*, то есть обучение на неразмеченных данных. Такое обучение, как правило, направлено на обнаружение закономерностей и изучение внутренней структуры данных и включает в себя

- **кластеризацию,**
- понижение размерности,
- обнаружение аномалий,
- обработку пропущенных значений и т.д.

Сегодня мы поговорим о задаче кластеризации.

Задача кластеризации

Кластеризация (clustering) – это задача группировки наблюдений таким образом, чтобы наблюдения из одной группы, или *кластера*, были в некотором смысле более похожи друг на друга, чем на наблюдения из других групп (кластеров).

Формально имеем

- Пространство объектов X и выборку из него X'
- Расстояние между объектами $\rho : X \times X \rightarrow R^+$

Хотим получить

- Множество групп/кластеров Y
- Алгоритм кластеризации $\alpha : X \rightarrow Y$

С какой целью используется

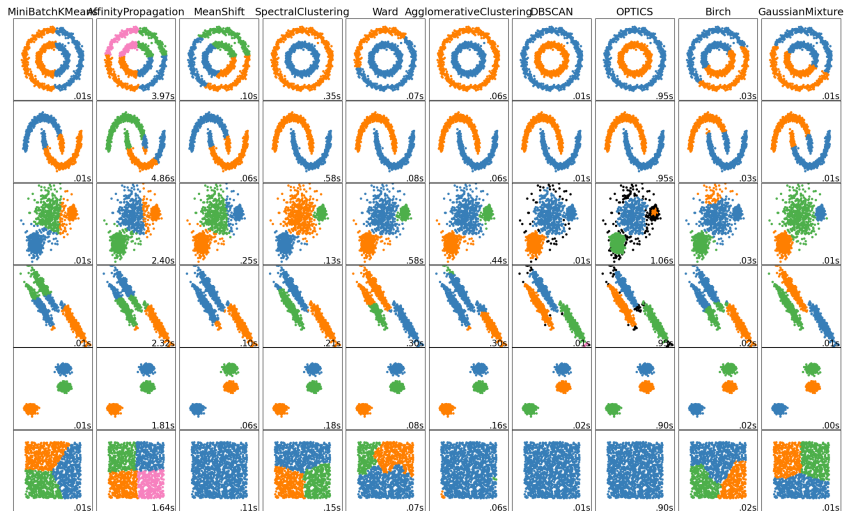
- Разделение на группы с целью упрощения работы (отдельные модели для каждой группы)
- Сокращение объемов наблюдений и сжатие данных (например, квантизация нейронных сетей)
- Выделение новизны/аномалий
- Построение иерархии/таксономии объектов

Некорректность постановки задачи

Сама по себе постановка задачи кластеризации некорректна, а именно

- не существует единого критерия качества (их, скорее, наоборот слишком много)
- число кластеров может быть заранее неизвестно
- сильная зависимость от метрики ρ

Иллюстрация проблем на примерах



И что с этим делать?

- Не полагаться на нее, если не существует альтернативных способов подтверждения адекватности ее результатов
- Ответственно подходить к предварительному изучению данных, отбору обучающей выборки и метрике

Требования к алгоритмам

- Масштабируемость
- Работа с большими размерностями
- Устойчивость к выбросам
- Устойчивость к различным типам кластерных структур
- Интерпретируемость результатов
- Временная сложность

Классификация алгоритмов

В большинстве источников выделяют пять групп алгоритмов

- *Основанные на центроидах* (centroid based): k-means, k-modes, k-medoids, **Meanshift**, FCM, Affinity propagation
- *Иерархические* (hierarchical): **агломеративные** (Ward, single/average/complete linkage), BIRCH, на основе теории графов (выделение связных компонент и минимальное остовное дерево), **Spectral Clustering**, CURE, ROCK, Chameleon, Echidna, SNN, CACTUS, GRIDCLUST
- *Основанные на плотности* (density based): **DBSCAN**, OPTICS, DBCLASD, GDBSCAN, DENCLU, SUBCLU
- *Сеточные* (grid based): STING, Wave cluster, BANG, CLIQUE, OptiGrid, MAFLA, ENCLUS, PROCLUS, ORCLUS, FC, STIRR
- *Основанные на модели данных* (model based): **Expectation Maximization** (EM), COBWEB, CLASSIT, SOM

- Meanshift – это алгоритм кластеризации, использующий *ядерную оценку плотности* (Kernel Density Estimation, KDE), который итеративно назначает наблюдения кластерам, сдвигая точки в сторону моды
- В отличие от (дешевого и сердитого) K-Means, Meanshift не требует заранее указывать количество кластеров, но требует задать параметр окна для KDE.

Meanshift: описание алгоритма

- Алгоритм итеративно назначает каждую точку ближайшему центроиду кластера.
- Направление к ближайшему центроиду скопления определяется тем, где находится большинство ближайших точек.
- Таким образом, на каждой итерации каждая точка данных будет перемещаться ближе к тому месту, где находится большинство точек, которое является или приведет к центру кластера.
- Когда алгоритм останавливается, каждой точке в соответствии ставится номер кластера

Meanshift: подробнее об алгоритме

- Ядерная оценка плотности

$$f_h(x) = \frac{1}{n} \sum_{i=1}^n K_h(x - x_i) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$

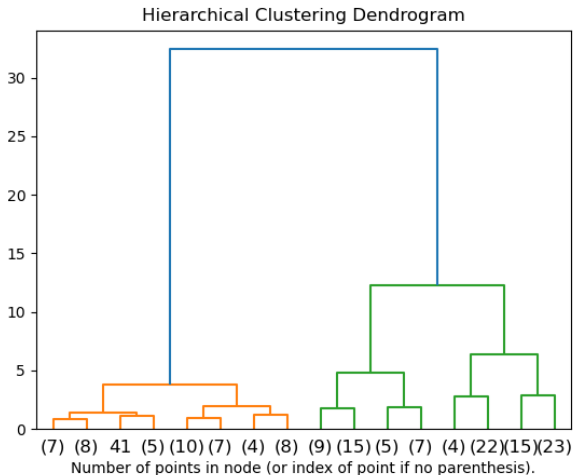
- Итеративное обновление точек

$$x_i^{t+1} = m(x_i^t)$$
$$m(x_i) = \frac{\sum_{x_j \in N(x_i)} K_h(x_j - x_i) x_j}{\sum_{x_j \in N(x_i)} K_h(x_j - x_i)}$$

- Обязательна стандартизация входных данных (привет, bandwidth)
- Есть реализация в scikit-learn'e. Распараллеливание; поддерживает бинаризацию исходных данных; можно даже не задавать h , но тогда страдает производительность; кроме брутфорса для поиска соседей есть KD- и Ball-деревья.

- Иерархическая кластеризация – это метод кластерного анализа, который направлен на построение иерархии кластеров
- Выделяют два подхода
 - Дивизивный (“снизу вверх”)
 - Агломеративный (“сверху вниз”)
- В большинстве реализаций слияния и разбиения происходят жадно
- Результаты иерархической кластеризации обычно представляются в виде дендрограммы.

Агломеративная кластеризация: пример дендрограммы



Агломеративная кластеризация: описание подхода

- Изначально каждая точка – отдельный кластер
- Объединение кластеров происходит путем поиска такой их пары, которая имеет наименьшее значение симметричной метрики близости между кластерами
- Процесс завершается, когда все сливается в один единственный кластер (или до фиксированного числа кластеров)
- Все, что остается, произвести “разрез” в получившейся иерархии кластеров

Агломеративная кластеризация: итеративный алгоритм Ланса-Уильямса

Algorithm 1: Алгоритм Ланса-Уильямса

input : Каждая точка – кластер $C_1 = \{\{x_1\}, \dots, \{x_n\}\}$, подсчитаны расстояния между кластерами-точками $R_{\{x_i\}\{x_j\}} = \rho(x_i, x_j)$

for $t \leftarrow 2$ **to** l **do**

 Найти в C_{t-1} пару кластеров U и V с минимальным R_{UV} ;

$W = U \cup V$;

$C_t = C_{t-1} \cup W \setminus \{U, V\}$;

for $S \in C_t$ **do**

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

Агломеративная кластеризация: частные случаи метрики схожести

- Расстояние до ближайшего соседа/Single/Minimum:

$$R_{WS} = \min_{w \in W, s \in S} \rho(w, s),$$
$$\alpha_U = \alpha_V = \frac{1}{2}, \beta = 0, \gamma = -\frac{1}{2}$$

- Расстояние до дальнего соседа/Complete/Maximum:

$$R_{WS} = \max_{w \in W, s \in S} \rho(w, s),$$
$$\alpha_U = \alpha_V = \frac{1}{2}, \beta = 0, \gamma = \frac{1}{2}$$

- Групповое среднее расстояние/Average:

$$R_{WS} = \frac{1}{|W||S|} \sum_{w \in W, s \in S} \rho(w, s),$$
$$\alpha_U = \frac{|U|}{|W|}, \alpha_V = \frac{|V|}{|W|}, \beta = \gamma = 0$$

Агломеративная кластеризация: частные случаи метрики схожести (прод.)

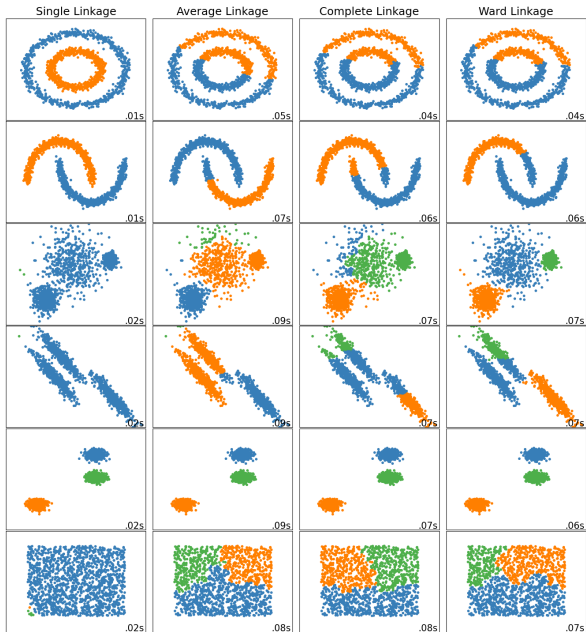
- Расстояние между центроидами/Centroid:

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

- Расстояние Уорда/Ward:

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

Агломеративная кластеризация: пример дендрограммы



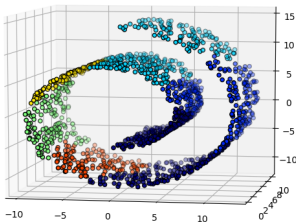
Агломеративная кластеризация: подробнее

- Подбор функции расстояния между точками, очевидно, имеет существенное влияние. Чаще всего выбор между l_2 и l_1
- Стандартизация тоже влияет. Но производить ли ее в случае иерархической кластеризации – вопрос спорный
- Метрика близости: обычно Ward. Ну и тут ограничение на использование только l_2 . В таких случаях в первую очередь смотрите на Average (устойчивее к выбросам)
- Обычно количество кластеров определяют по дендограмме: “отсекают” там, где дельта метрики на двух итерациях имеет наибольшее значение

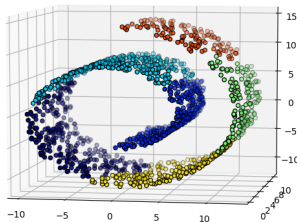
Агломеративная кластеризация: подробнее (прод.)

- Чаще всего применяют в случае, когда данных мало, либо когда хочется построить иерархию/таксономию
- Имплементировано в `scikit-learn`'е. Из интересного: есть возможность докидывать ограничения на локальную структуру в данных

Without connectivity constraints (time 0.05s)



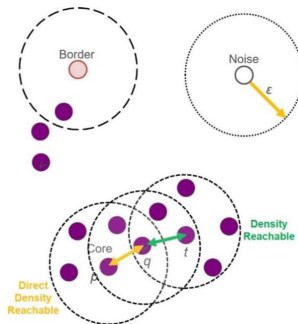
With connectivity constraints (time 0.07s)



- Density-Based Spatial Clustering of Applications with Noise (DBSCAN) – алгоритм кластеризации, основанный на плотности точек, изначально разработанный с целью кластеризации в базах данных, содержащих геометрические представления наблюдений
- Основными преимуществами алгоритма авторы выделили минимальную необходимость понимания предметной области данных при подборе гиперпараметров метода, а также способность обнаруживать кластеры произвольной формы
- Алгоритм достаточно прост, наряду с k-means один из самых популярных

DBSCAN: описание алгоритма

- Алгоритм имеет 2 гиперпараметра: величина окрестности точки ε и минимальное количество наблюдений в окрестности $MinPts$
- При кластеризации точка может быть причислено к 3 типам:
 - корневая: в ε -окрестности не менее $MinPts$ точек
 - граничная: в ε -окрестности меньше $MinPts$ точек, но среди них есть как минимум одна корневая
 - шумовая: не корневая и не граничная



Algorithm 2: DBSCAN

input : Выборка $X = \{x_1, \dots, x_n\}$, параметры ε и $MinPts$

$U = X$, $N = \emptyset$, $a = 0$;

while $U \neq \emptyset$ **do**

 Взять $x \in U$;

if $|U_\varepsilon(x)| < MinPts$ **then**

$N = N \cup x$;

else

$K = U_\varepsilon(x)$, $a = a + 1$;

for $x' \in K$ **do**

if $|U_\varepsilon(x')| \geq MinPts$ **then**

$K = K \cup U_\varepsilon(x')$;

else

 пометить x' как граничную точку кластера K ;

foreach $x_i \in K$ **do** $a_i = a$;

$U = U \setminus K$;

- Подбор гиперпараметров.
 - Общая идея состоит в построении графика, по ординате у которого расстояние до *MinPts*-го соседа, а по абсциссе – точки, отсортированные в порядке увеличения этого расстояния.
 - Существенный скачок в значении идентифицирует выбросы, посему задавая некоторый процент на их число можно определить ε .
 - Обычно строят несколько таких графиков для различных значений *MinPts*.
 - В некоторых источниках значение *MinPts* предлагают выбирать равным $\dim X + 1$, Где-то встречается $2 * \dim X$
- Есть реализация в `sklearn`'е. Кроме того, там же представлена модификация алгоритма под названием OPTICS, фактически отличающаяся от него тем, что задает интервал для значений ε , что позволяет выделять кластеры с различными плотностями

- Expectation Maximization (EM) – итеративный алгоритм для поиска оценок максимального правдоподобия параметров вероятностных моделей, зависящих от некоторых скрытых переменных
- Алгоритм имеет приложения в дискриминатном анализе, кластеризации (разделение смеси распределений), восстановлении пропусков в данных, обработке сигналов и изображений

- Смесь распределений

$$p(x) = \sum_{j=1}^k w_j \varphi(x, \theta_j), \quad \sum_{j=1}^k w_j = 1, \quad w_j \geq 0$$

$\varphi(x, \theta_j) = p(x|j)$ – функция правдоподобия j -ой компоненты,
 $w_j = P(j)$ – априорная вероятность j -ой компоненты

- Задача поиска максимума правдоподобия

$$L(w, \theta) = \ln \prod_{i=1}^n p(x_i) = \sum_{i=1}^n \ln \sum_{j=1}^k w_j \varphi(x_i, \theta_j) \longrightarrow \max_{w, \theta}$$
$$\text{w.r.t } \sum_{j=1}^k w_j = 1, \quad w_j \geq 0$$

- При разделении смеси распределений (обычно, гауссовских, отсюда Gaussian Mixtures Model, GMM-EM) ЕМ используется следующим образом:
 - (expectation) вводится вспомогательный вектор скрытых переменных g , такой что его значения могут быть вычислены, зная параметры распределений θ
 - (maximization) вычислив значения скрытых переменных, задача поиска (локального) максимума правдоподобия существенно упрощается

- В качестве скрытых переменных выберем

$$g = \{g_{ij}\}, \quad g_{ij} = P(j|x_i)$$

- По формуле Байеса

$$g_{ij} = \frac{w_j \varphi(x_i, \theta_j)}{\sum_{s=1}^k w_s \varphi(x_i, \vartheta_s)} \quad (1)$$

- Формулы для θ_j и w_j выводятся из условий Куна–Такера

$$\theta_j = \arg \max_{\theta_s} \sum_{i=1}^n g_{ij} \ln \varphi(x_i, \theta_s) \quad (2)$$

$$w_j = \frac{1}{n} \sum_{i=1}^n g_{ij} \quad (3)$$

Algorithm 3: EM

input : Выборка $X = \{x_1, \dots, x_n\}$, начальные приближения для (w, θ) , количество кластеров k и ε

while *True* **do**

 E: **foreach** $i = 1, \dots, n$, $j = 1, \dots, k$ **do** $g_{ij}^0 = g_{ij}$, вычислить новые значения g_{ij} по формуле (1);

 M: **foreach** $j = 1, \dots, k$ **do** решить задачу (2) для нахождения θ_j , вычислить новые значения w_j по формуле (3);

if $\max_{i,j} |g_{ij} - g_{ij}^0| < \varepsilon$ **then**

 └ Завершить работу

- В случае гауссовских плотностей при условии, что признаки независимы (матрица ковариаций диагональна), задача (2) имеет аналитическое решение
- Кластеризация мягкая, т.е. точкам не ставится в соответствии номер кластера, а лишь вероятность принадлежности. Нужен номер – берите максимум по g_{ij}
- k-means – частный случай ЕМ. На Е-этапе вычисляются скрытые переменные – номера кластеров, а на М-этапе происходит обновление центров.
- Начальное приближение: рандом/k-means

- У EM'а есть масса модификаций: generalized EM (GEM), stochastic EM (SEM), есть вариант с последовательным добавлением компонент
- GaussianMixture в scikit-learn'е. Количество компонент обязательно. Поддерживает различные типы ковариационных матриц

Spectral Clustering: основная теория

- Предварительный этап состоит в формировании из исходной матрицы с наблюдениями X ненаправленный граф похожести (similarity graph). Варианты следующие:
 - Полностью связанный
 - ε -окрестность
 - k -ближайших соседей: если вершина v_i находится среди k -ближайших соседей вершины v_j
- Элементы s_{ij} матрицы схожести графа неотрицательны и равны 0, когда объекты совершенно не похожи (не связаны, нет ребра в графе)
- Обычно матрица формируется на основе радиально-базисных функций: чаще всего Гауссовская $\varphi(x) = \exp(\frac{-d_{ij}^2}{\sigma^2})$

Spectral Clustering: основная теория (прод.)

- Пусть W есть матрица весов (взвешенная матрица смежности), такая что $w_{ij} = s_{ij}$, если существует ребро между вершинами v_i и v_j , и 0 иначе
- Матрица D (degree matrix) есть диагональная матрица с элементами $d_i = \sum_{j=1}^n w_{ij}$

Spectral Clustering: основная теория (прод.)

- Лапласианом графа называется матрица $L = D - W$ с набором интересных свойств
 - $z^T L z = \frac{1}{2} \sum_{i,j=1}^n w_{ij} (z_i - z_j)^2$ для любого $z \in R^n$
 - L симметрична и положительно полуопределена
 - У L n неотрицательных собственных чисел: $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$
 - Наименьшее собственное число λ_1 всегда равно 0. Соответствующий ему собственный вектор состоит из 1, если граф связный. Если у графа есть p компонент связности, то кратность λ_1 равна p , а собственные векторы представляют собой индикаторные векторы

Algorithm 4: Spectral Clustering

input : Матрица схожести $S \in R^{n \times n}$ и количество кластеров k
Построить граф схожести. Пусть W есть его матрица смежности;
Вычислить лапласиан L ;
Найти первые k собственных векторов $u_1, \dots, u_k$. Составить из них матрицу $U \in R^{n \times k}$;
k-means(U, k);

Spectral Clustering: комментарии I

- Подробнее о том, почему это действительно работает, в статье. Спектральная кластеризация по сути является релаксацией задачи о разделении графа на компоненты
- Но нужна грамотная настройка: тут и выбор матрицы схожести, и количества кластеров. Для данного алгоритма есть эвристика для поиска оптимального числа кластеров: построить график собственных векторов по возрастанию и “обрезать” там, где происходит существенный скачок в их значениях
- Что касается поиска собственных векторов, то существует масса численных методов, обернутых в фреймворки. Например, «`arpack`», «`eigen`». `sklearn` подтягивает методы линейной алгебры из `scipy` – подробнее в их доках. С простейшим итеративным методом поиска собственных векторов познакомимся в следующий раз, когда будет говорить о методе главных компонент (PCA)
- Лапласиан не единственный. Существуют “нормализованные” альтернативы

- Есть в `scikit-learn`'е. `n_components` не про количество связанных компонент, а про число собственных векторов, используемых на втором этапе алгоритма (по умолчанию `n_components = n_clusters`)

Сравнение подходов: Centroid based

Преимущества:

- Простые, надежные, обычно масштабируемые
- Не требуют глубокого понимания специфики данных
- Итеративные (пересчитали центроиды и разделение на кластеры изменилось)

Недостатки

- Число кластеров?
- Нормализация/стандартизация данных может существенно изменить расклад

Сравнение подходов: Hierarchical

Преимущества:

- Изначально не требуют фиксации количества кластеров
- Зачастую никаких гиперпараметров
- Просты для понимания и имплементации

Недостатки

- Трудности с интерпретацией
- Чувствительны к выбросам
- Отсутствует итеративность (как только объект отнесен к кластеру, он из него уже не выйдет)

Сравнение подходов: Density based

Преимущества:

- Умеют в кластеры произвольной формы
- Весьма устойчивы к выбросам

Недостатки

- Трудности с очень большими и разреженными данными
- Проклятие размерности

Сравнение подходов: Grid based

Преимущества:

- Разбивают пространство данных на ячейки
- Хороши для датасетов больших размерностей, итеративные

Недостатки

- Плохо работают на данных, являющихся отображением датасета меньшей размерности в пространство большей размерности
- Требуют тщательного выбора проекций, функции оценки плотности и оптимальных разделяющих поверхностей

Сравнение подходов: Model based

Преимущества:

- Полученное разбиение интерпретируемо со статистической точки зрения
- Успешно используются в векторной квантизации
- Позволяют выбирать плотности распределения компонент

Недостатки

- Параметры должны быть оценены, что требует большего количества точек данных в каждой компоненте
- Трудно настраивать

- Разделение простое: либо работа с известной правильной разметкой, либо попытка понять, насколько хорошо произошло разбиение
- К первым относятся Adjusted Rand index, Mutual Information based scores, Homogeneity, completeness and V-measure
- Ко вторым Silhouette, Calinski-Harabasz Index, Davies-Bouldin Index и другие

Adjusted Rand index

- Здесь и далее обозначим через C истинное разбиение данных, а через K – разбиение на основе какого-либо алгоритма
- Пусть a и d есть общее количество пар наблюдений, которые находятся в одном/разных кластерах как в C , так и в K
- Rand Index и Adjusted Rand index

$$RI = 2 \frac{a + d}{n(n-1)} = \frac{a + d}{a + b + c + d}$$

$$\begin{aligned}ARI &= \frac{RI - E[RI]}{\max(RI) - E[RI]} = \\&= \frac{(a + b + c + d)(a + d) - [(a + b)(a + c) + (b + d)(c + d)]}{(a + b + c + d)^2 - [(a + b)(a + c) + (b + d)(c + d)]}\end{aligned}$$

- Очевидно, чем больше, тем лучше. Значения RI в интервале от 0 до 1, ARI – $[-1; 1]$. Если разметка рандомная, то ARI будет близок к 0. Никаких предположений о структуре кластеров – можно сравнивать результаты разных алгоритмов

Mutual Information based scores

- Энтропия C/K

$$H(C) = - \sum_{i=1}^{|C|} P^C(i) \log(P(i)), \quad P^C(i) = \frac{|C_i|}{n}$$

- Взаимная информация (Mutual information)

$$MI(C, K) = \sum_{i=1}^{|C|} \sum_{j=1}^{|K|} P(i, j) \log \left(\frac{P(i, j)}{P^C(i) P^K(j)} \right), \quad P(i, j) = \frac{|C_i \cap K_j|}{n}$$

- Ее нормализованная версия

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

Mutual Information based scores (прод.)

- И скорректированная нормализованная версия

$$\text{AMI}(U, V) = \frac{\text{MI} - E[\text{MI}]}{\text{mean}(H(C), H(K)) - E[\text{MI}]}$$

где

$$E[\text{MI}(U, V)] = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \sum_{n_{ij}=(a_i+b_j-N)^+}^{\min(a_i, b_j)} \frac{n_{ij}}{N} \log \left(\frac{N n_{ij}}{a_i b_j} \right) \times \\ \times \frac{a_i! b_j! (N - a_i)! (N - b_j)!}{N! n_{ij}! (a_i - n_{ij})! (b_j - n_{ij})! (N - a_i - b_j + n_{ij})!}$$

Mutual Information based scores (прод.)

- Чем больше, тем лучше
- MI, NMI лежат в интервале $[0; 1]$. При произвольной разметке не факт что будут близки к 0
- AMI это исправляет, его значения лежат в интервале от -1 до 1

Homogeneity и completeness

- Homogeneity (однородность) – каждый кластер из K содержит наблюдения только из одного класса C

$$h = 1 - \frac{H(C|K)}{H(C)}$$

- Completeness (полнота) – все наблюдения из класса C находятся в каком-либо кластере K

$$c = 1 - \frac{H(K|C)}{H(K)}$$

- Условная энтропия

$$H(C|K) = - \sum_{c=1}^{|C|} \sum_{k=1}^{|K|} \frac{n_{c,k}}{n} \log \left(\frac{n_{c,k}}{n_k} \right)$$

где $n_{c,k}$ есть число наблюдений из класса c в кластере k , а n_k – общее число наблюдений в кластере k

- V-мера – их гармоническое среднее

$$v_{\beta} = \frac{(1 + \beta)hc}{\beta h + c}$$

- Значения всех трех метрик в интервале $[0; 1]$. Чем больше, тем лучше.
- Никаких предположений о структуре кластеров – можно сравнивать результаты разных алгоритмов кластеризации
- Та же проблема с нормализацией. Если мало наблюдений / много кластеров, рекомендуется использовать ARI

- Пусть a и b есть среднее расстояние между наблюдением и всеми другими точками в том же кластере/в следующем ближайшем кластере
- Коэффициент силуэта для наблюдения есть

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

- Для выборки коэффициент силуэта задается средним значением коэффициентов каждого наблюдения
- Значения в интервале $[-1; 1]$. Чем больше, тем лучше. Если коэффициент близок к 0, то это свидетельство в сторону того, что кластеры “накладываются” друг на друга

Calinski-Harabasz Index (Variance Ratio Criterion)

- Индекс Калинского–Харабэса определяется как отношение между средней межкластерной дисперсией и средней дисперсией внутри кластеров

$$s = \frac{\text{tr}(B)}{k-1} / \frac{\text{tr}(W)}{n-k}$$

- Тут k – число кластеров, а матрицы B и W имеют вид

$$W = \sum_{q=1}^k \sum_{x \in C_q} (x - c_q)(x - c_q)^T, \quad B = \sum_{q=1}^k n_q (c_q - c_X)(c_q - c_X)^T$$

где C_q – наблюдения из кластера q , c_q – центр кластера q , c_X – центр всего набора данных X , а n_q – число наблюдений в кластере q

- Значение индекса тем больше, чем разделеннее кластеры и чем более сгруппированы в кластерах наблюдения. Ну и применять для случая сферических/эллиптических кластеров

Davies-Bouldin Index

- Индекс Дэвиса–Болдина оценивает среднее “сходство” между кластерами, где сходство есть мера, которая сравнивает расстояние между кластерами с размером самих кластеров
- Пусть s_i есть среднее расстояние между каждой точкой кластера i и центроида этого кластера, а d_{ij} есть расстояние между центроидами кластеров. Определим схожесть между кластерами следующим образом

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

- Тогда индекс имеет вид

$$DB = \frac{1}{k} \sum_{i=1}^k \max_{i \neq j} R_{ij}$$

- Чем индекс меньше, тем лучше. Близок к 0 – отличное разделение. Недостаток тот же, что и у индекса Калинского–Харабэса

Elbow method

- Вполне естественно полагать, что при хорошей кластеризации (в простом случае сферических/эллиптических кластеров) общее внутрикластерное расстояние

$$W(K) = \sum_{i=1}^k \sum_{j \in K_k} \|x_j - \mu_i\|_2 \longrightarrow \min_K$$

должно быть как можно меньше

- В то же время, минимум метрики достигается тогда, когда каждое наблюдение формирует отдельный кластер
- Идея метода локтя проста: смотрим на график метрики для различного количества кластеров и выбираем то, после которого увеличение числа кластеров не дает существенного снижения значения метрики
- Формализуя, правило следующее:

$$D(k) = \frac{|W(K_k) - W(K_{k+1})|}{|W(K_{k-1}) - W(K_k)|} \longrightarrow \min_k$$

- Рассмотренные ранее коэффициенты Silhouette, Calinski-Harabasz Index, Davies-Bouldin Index также можно использовать для определения числа кластеров
- Всем желающим подробнее изучить эту тему предлагаю взглянуть на библиотеку “NbClust” в R. В ней представлено более 30 различных коэффициентов