Алгоритм кластеризации *k*-means [M.123]

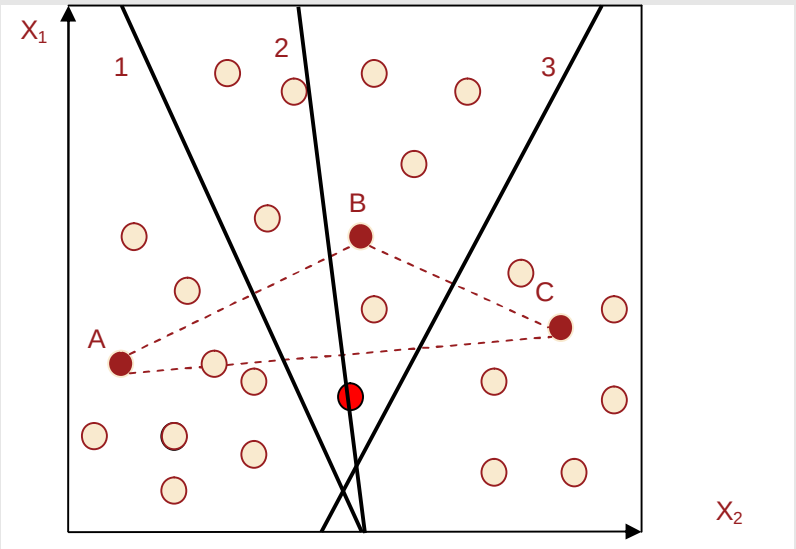
Одной из широко используемых методик кластеризации является *разделительная кластеризация* (англ.: *partitioning clustering*) в соответствии с которой для выборки данных, содержащей n записей (объектов), задается число кластеров k, которое должно быть сформировано. Затем алгоритм разбивает все объекты выборки на k разделов ($k < n$), которые и представляют собой кластеры.

Одним из наиболее простых и эффективных алгоритмов кластеризации является алгоритм *k-means* (Mac-Queen, 1967) или в русскоязычном варианте *k*-средних (от англ. *mean* – среднее значение). Он состоит из четырех шагов.

1. Задается число кластеров *k*, которое должно быть сформировано из объектов исходной выборки.
2. Случайным образом выбирается *k* записей исходной выборки, которые будут служить начальными центрами кластеров. Такие начальные точки, из которых потом «вырастает» кластер, часто называют «семенами» (от англ. *seeds* – семена, посевы). Каждая такая запись будет представлять собой своего рода «эмбрион» кластера, состоящий только из одного элемента.
3. Для каждой записи исходной выборки определяется ближайший к ней центр кластера.

Пример. Для лучшего понимания методики воспользуемся геометрической интерпретацией определения, к какому из центров кластеров ближе всего расположена та или иная запись. Границей между двумя кластерами является точка, равноудаленная от четырех центров. Из геометрии известно, что множество точек, равноудаленных от двух точек *A* и *B*, будут образовывать прямую, перпендикулярную отрезку *AB* и пересекающую его по середине. Принцип поиска границ будущих кластеров поясняется с помощью рисунке. Для простоты рассматривается 2-мерный случай, когда пространство признаков содержит всего два измерения (X_1,X_2). Пусть на первом шаге были отобраны три точки *A*, *B* и *C*. Тогда линия 1, проходящая перпендикулярно отрезку *AB* через его середину, будет состоять из точек, равноудаленных от точек *A* и *B*. Аналогично строится прямая 2 для точек *A* и *C*, а также прямая 3 для точек *B* и *C*. Используя данные линии можно определить, к какому из центров оказывается ближе та или иная точка.

Однако, такая «геометрическая» интерпретация определения того, в «сферу влияния» какого центра кластера входит та или иная запись, полезна только для лучшего понимания. На практике для этого вычисляется расстояние от каждой записи до каждого центра в многомерном пространстве признаков, и выбирается то «семя», для которого данное расстояние минимально (более подробно это рассмотрено ниже).



4. Производится вычисление *центроидов* – центров тяжести кластеров. Это делается путем простого определения среднего для значений каждого признака для всех записей в кластере. Например, если в кластер вошли три записи с наборами признаков

$(x_1, y_1), (x_2, y_2), (x_3, y_3)$, то координаты его центроида будут рассчитываться следующим образом:

$$(x, y) = \left(\frac{(x_1 + x_2 + x_3)}{3}, \frac{(y_1 + y_2 + y_3)}{3} \right).$$

Затем старый центр кластера смещается в его центроид. На рис. 1 центры тяжести кластеров представлены в виде крестиков.

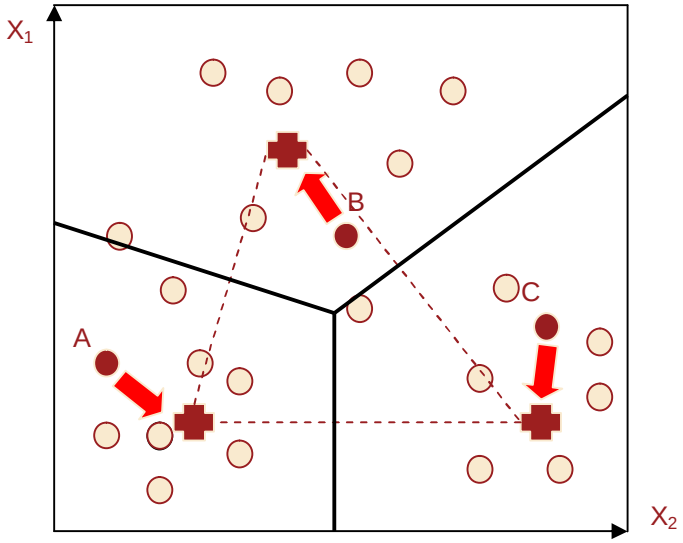


Рис. 1. Определение центров тяжести кластеров (центроидов) и новых границ кластеров

Таким образом, центроиды становятся новыми центрами кластеров для следующей итерации алгоритма.

Шаги 3 и 4 повторяются до тех пор, пока выполнение алгоритма не будет прервано либо не будет выполнено условие в соответствии с некоторым критерием сходимости.

Остановка алгоритма производится тогда, когда границы кластеров и расположения центроидов не перестанут изменяться от итерации к итерации, т.е. на каждой итерации в каждом кластере будет оставаться один и тот же набор записей. На практике алгоритм *k-means* обычно находит набор стабильных кластеров за несколько десятков итераций.

Что касается критерия сходимости, то чаще всего используется критерий суммы квадратов ошибок между центроидом кластера и всеми вошедшими в него записями, т.е.

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} (p - m_i)^2 ,$$

где $p \in C_i$ – произвольная точка данных, принадлежащая кластеру C_i, m_i – центроид данного кластера. Иными словами, алгоритм остановится тогда, когда ошибка *E* достигнет достаточно малого значения.

Меры расстояний

Ключевым моментом алгоритма *k-means* является вычисление на каждой итерации расстояния между записями и центрами кластеров, что необходимо для определения, к какому из кластеров принадлежит данная запись. Правило, по которому производится вычисление расстояния в многомерном пространстве признаков, называется *метрикой*. Наиболее часто в практических задачах кластеризации используются следующие метрики.

- Евклидово расстояние (метрика L_2). Использует для вычисления расстояний следующее правило:

$$d_E(x, y) = \sqrt{\sum_i (x_i - y_i)^2} , \quad (1)$$

где $\mathbf{x} = (x_1, x_2, ..., x_m)$, $\mathbf{y} = (y_1, y_2, ..., y_m)$ – наборы (векторы) значений признаков двух записей. Поскольку множество точек, равноудаленных от некоторого центра при использовании евклидовой метрики будет образовывать сферу (или круг в двумерном случае), то и кластеры, полученные с использованием евклидова расстояния, также будут иметь форму, близкую к сферической.

- Расстояние Манхэттена (англ.: *Manhattan distance*) или метрика L_1:

$$d_M(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i| .$$

Фактически это кратчайшее расстояние между двумя точками, пройденное по линиям, параллельным осям координатной системы (рис. 2).

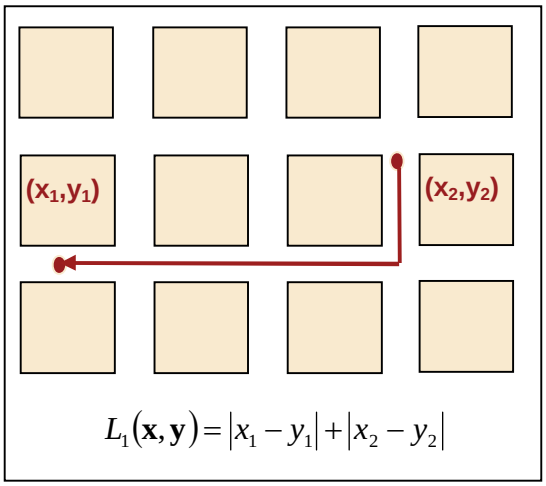


Рис. 2. Расстояние Манхэттена (метрика L_1)

Преимущество метрики L_1 заключается в том, что ее использование позволяет снизить влияние аномальных значений на работу алгоритмов. Кластеры, построенные на основе расстояния Манхэттена, стремятся к кубической форме. Иногда расстояние Манхэттена называют «расстояние городских кварталов» (англ.: *city-block distance*), из-за ассоциаций, возникающих с прямоугольными формами застройки, которая характерна для современных городов.

Для категориальных признаков в качестве меры расстояния можно использовать функцию отличия (англ: *different function*), которая задается следующим образом:

$$d(x_i, y_i) = \begin{cases} 0, & \text{если } x_i = y_i \\ 1, & \text{в остальных случаях} \end{cases},$$

где x_i и y_i – категориальные значения.

Свою популярность алгоритм *k-means* приобрел благодаря следующим свойствам. Во-первых, это умеренные вычислительные затраты, которые растут линейно с увеличением числа записей исходной выборки данных. Вычислительная сложность алгоритма определяется как $k \times n \times l$, где k – число кластеров, n – число записей и l – число итераций. Поскольку k и l заданы, то вычислительные затраты возрастают пропорционально числу записей исходного множества. Полезным свойством *k-means* является то, что результаты его работы не зависят от порядка следования записей в исходной выборке, а определяются только выбором исходных точек.

Одним из основных недостатков, присущих этому алгоритму, является отсутствие четких критериев выбора числа кластеров, целевой функции их инициализации и модификации. Кроме этого, он является очень чувствительным к шумам и аномальным значениям в данных, поскольку они способны значительно повлиять на среднее значение, используемое при вычислении положений центроидов. Чтобы снизить влияние таких факторов, как шумы и аномальные значения, иногда на каждой итерации используют не среднее значение признаков, а их медиану. Данная модификация алгоритма называется *k-mediods* (*k*-медиан).

Пример работы алгоритма k-means, разбивающего объекты на 4 кластера, приведен на рис. 3. Центры кластеров отмечены знаком «x», а каждому кластеру соответствует свой цвет.

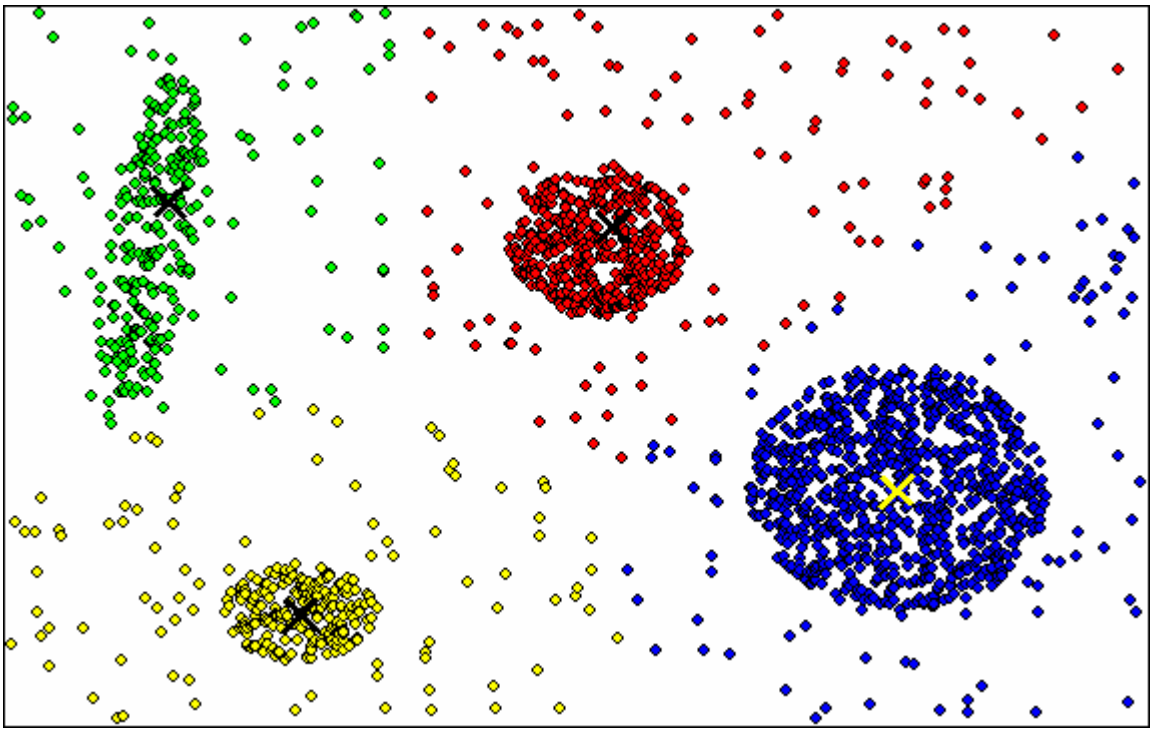


Рис. 3. Пример работы алгоритма k-means

Пример работы алгоритма ***k-means***

Пусть имеется набор из 8 точек данных в двумерном пространстве, из которого требуется получить два кластера. Значения точек приведены в табл. 1 и на рис. 4.

Таблица 1.

A	B	C	D	E	F	G	H
(1,3)	(3,3)	(4,3)	(5,3)	(1,2)	(4,2)	(1,1)	(2,1)

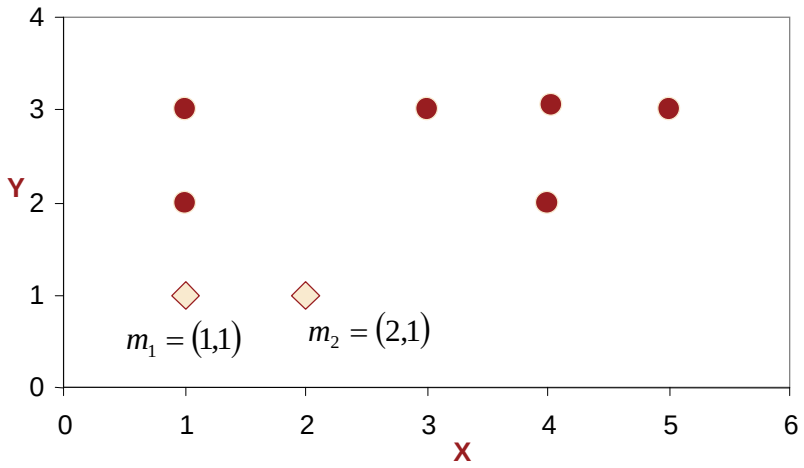


Рис. 4. Начальная инициализация

Шаг 1. Определим число кластеров, на которое требуется разбить исходное множество k=2.

Шаг 2. Случайным образом выберем две точки, которые будут начальными центрами кластеров. Пусть это будут точки m_1=(1;1) и m_2=(2;1). На рис. 3 они представлены ромбами.

Шаг 3, проход 1. Для каждой точки определим к ней ближайший центр кластера с помощью расстояния Евклида. В табл. 2 представлены вычисленные с помощью формулы (1) расстояния между центрами кластеров m_1=(1;1), m_2=(2;1) и каждой точкой исходного множества, а также указано, к какому кластеру принадлежит та или иная точка.

Таблица 2.

Точка	Расстояние от m_1	Расстояние от m_2	Принадлежит кластеру
A	2,00	2,24	1
B	2,83	2,24	2
C	3,61	2,83	2
D	4,47	3,61	2
E	1,00	1,41	1
F	3,16	2,24	2
G	0,00	1,00	1
H	1,00	0,00	2

Таким образом, кластер 1 содержит точки A, E, G, а кластер 2 – точки B, C, D, F, H. Как только определятся члены кластеров, может быть рассчитана сумма квадратичных ошибок:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} (p - m_i)^2 = 2^2 + 2{,}24^2 + 2{,}83^2 + 3{,}61^2 + 1^2 + 2{,}24^2 + 0^2 + 0^2 = 36.$$

Шаг 4, проход 1. Для каждого кластера вычисляется его центроид, и центр кластера перемещается в него. Центроид для первого кластера вычисляется как

[(1+1+1)/3, (3+2+1)/3]=(1; 2).

Центроид для кластера 2 будет равен

[(3 + 4 + 5 + 4 + 2)/5, (3 + 3 + 3 + 2 + 1)/5] = (3,6; 2,4).

Расположение кластеров и центроидов после первого прохода алгоритма представлено на рис. 5.

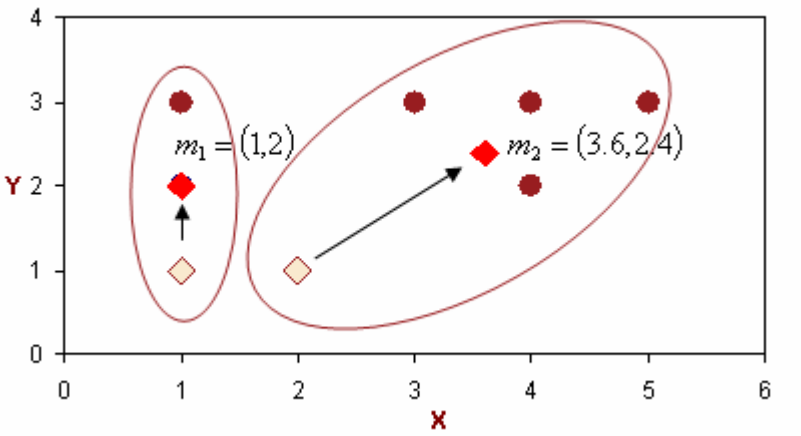


Рис. 5. Расположение кластеров и центроидов после первого прохода алгоритма

На рис. 5 начальные центры кластеров представлены светлыми ромбами, а центроиды, вычисленные при 1-м проходе алгоритма, – красными ромбами. Они и будут являться новыми центрами кластеров, к которым будет определяться принадлежность точек данных на втором проходе.

Шаг 3, проход 2. После того, как найдены новые центры кластеров, для каждой точки снова определяется ближайший к ней центр и ее отношение к соответствующему кластеру. Для это еще раз вычисляются евклидовы расстояния между точками и центрами кластеров. Результаты вычислений приведены в табл. 3.

Таблица 3.

Точка	Расстояние от m_1	Расстояние от m_2	Принадлежит кластеру
A	1,00	2,67	1
B	2,24	0,75	2
C	3,16	0,72	2
D	4,12	1,52	2
E	0,00	2,63	1
F	3,00	0,57	2
G	1,00	2,95	1
H	1,41	2,13	1

Относительно большое изменение m_2 привело к тому, что запись *H* оказалась ближе к центру m_1, что автоматически сделало ее членом кластера 1. Все остальные записи остались в тех же кластерах, что и на предыдущем проходе алгоритма. Таким образом, кластер 1 будет A, E, G, H, а кластер 2 – B, C, D, F. Новая сумма квадратов ошибок составит

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} (p - m_i)^2 = 1^2 + 0{,}85^2 + 0{,}72^2 + 1{,}52^2 + 0^2 + 0{,}57^2 + 1^2 + 1{,}41^2 = 7{,}88,$$

что показывает уменьшение ошибки относительно начального состояния центров кластеров (которая на первом проходе составляла 36). Это говорит об улучшении качества кластеризации, т.е. более высокую «кучность» объектов относительно центра кластера.

Шаг 4, проход 2. Для каждого кластера вновь вычисляется его центроид, и центр кластера перемещается в него. Новый центроид для 1-го кластера вычисляется как [(1+1+1+2)/4, (3+2+1+1)/4]=(1.25, 1.75). Центроид для кластера 2 будет [(3 + 4 + 5 + 4)/4, (3 + 3 + 2 + 4)/4] = (4, 2.75). Расположение кластеров и центроидов после второго прохода алгоритма представлено на рис. 6.

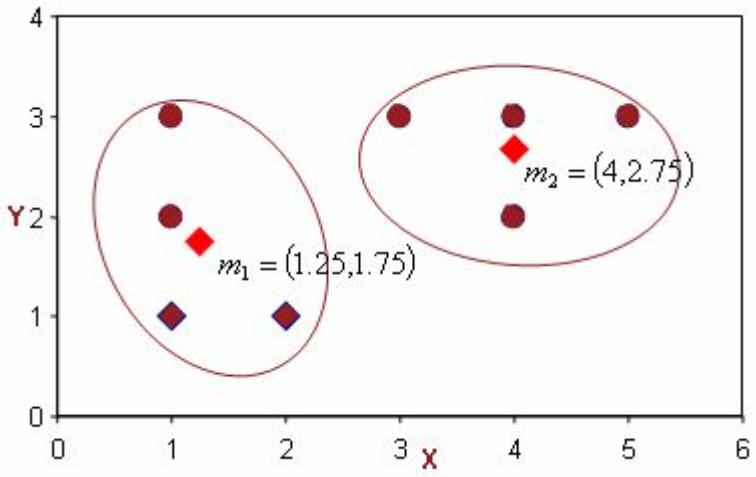


Рис. 6. Расположение кластеров и центроидов после второго прохода алгоритма

По сравнению предыдущим проходом центры кластеров изменились незначительно.

Шаг 3, проход 3. Для каждой записи вновь ищется ближайший к ней центр кластера. Полученные на данном проходе расстояния представлены в табл. 4.

Таблица 4.

Точка	Расстояние от m_1	Расстояние от m_2	Принадлежит кластеру
A	1,27	3,01	1
B	2,15	1,03	2
C	3,02	0,25	2
D	3,95	1,03	2
E	0,35	3,09	1
F	2,76	0,75	2
G	0,79	3,47	1
H	1,06	2,66	1

Следует отметить, что записей, сменивших кластер на данном проходе алгоритма, не было. Новая сумма квадратов ошибок составит

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} (p - m_i)^2 = 1{,}27^2 + 1{,}03^2 + 0{,}25^2 + 1{,}03^2 + 0{,}35^2 + 0{,}75^2 + 0{,}79^2 + 1{,}06^2 = 6{,}25.$$

Таким образом, изменение суммы квадратов ошибок является незначительным по сравнению с предыдущим проходом.

Шаг 4, проход 3. Для каждого кластера вновь вычисляется его центроид, и центр кластера перемещается в него. Но поскольку на данном проходе ни одна запись не изменила своего членства в кластерах, то положение центроида не поменялось, и алгоритм завершает свою работу.