ЛЕКЦИЯ 5. Интеллектуальный анализ данных: задача иерархической кластеризации

Иерархические методы кластерного анализа.

Суть иерархической кластеризации состоит в последовательном объединении меньших кластеров в большие или разделении больших кластеров на меньшие.

Иерархические агломеративные методы (Agglomerative Nesting, AGNES).

Эта группа методов характеризуется последовательным объединением исходных элементов и соответствующим уменьшением числа кластеров. В начале работы алгоритма все объекты являются отдельными кластерами. На первом шаге наиболее похожие объекты объединяются в кластер. На последующих шагах объединение продолжается до тех пор, пока все объекты не будут составлять один кластер.

Иерархические дивизимные (делимые) методы (DIvisive ANAlysis, DIANA)

Эти методы являются логической противоположностью агломеративным методам. В начале работы алгоритма все объекты принадлежат одному кластеру, который на последующих шагах делится на меньшие кластеры, в результате образуется последовательность расщепляющих групп.

Принцип работы описанных выше групп методов в виде дендрограммы показан на рис. 1.

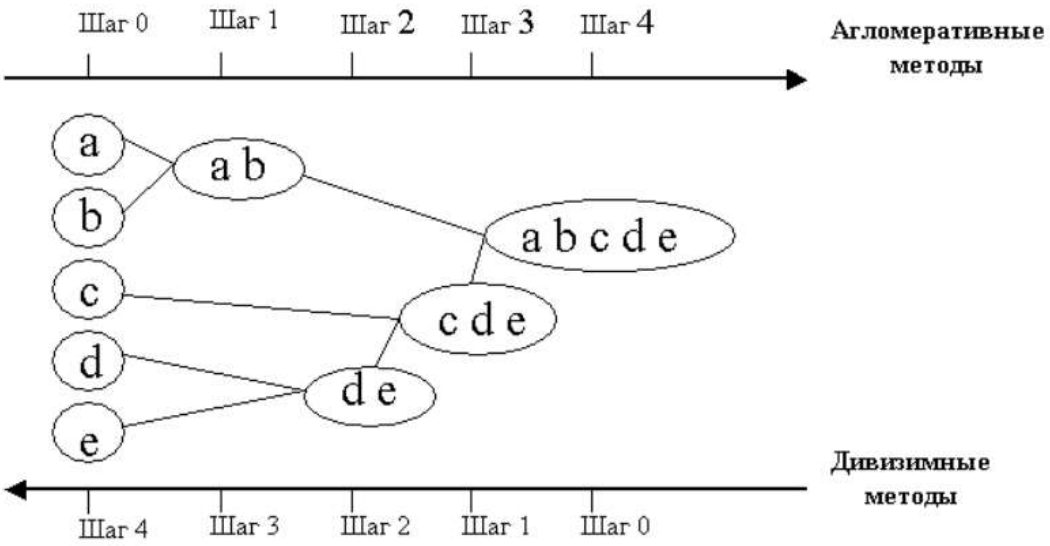

Рис. 1. Дендрограмма агломеративных и дивизимных методов

Программная реализация алгоритмов кластерного анализа широко представлена в различных инструментах Data Mining, которые позволяют решать задачи достаточно большой размерности. Например, агломеративные методы реализованы в пакете SPSS, дивизимные методы - в пакете Statgraf.

Иерархические методы кластеризации различаются правилами построения кластеров. В качестве правил выступают критерии, которые используются при решении вопроса о «схожести» объектов при их объединении в группу (агломеративные методы) либо разделения на группы (дивизимные методы).

Иерархические методы кластерного анализа используются при небольших объемах наборов данных. Преимуществом иерархических методов кластеризации является их наглядность.

Иерархические алгоритмы связаны с построением дендрограмм (от греческого dendron – «дерево»), которые являются результатом иерархического кластерного анализа. Дендрограмма описывает близость отдельных точек и кластеров друг к другу, представляет в графическом виде последовательность объединения (разделения) кластеров.

Дендрограмма (dendrogram) - древовидная диаграмма, содержащая n уровней, каждый из которых соответствует одному из шагов процесса последовательного укрупнения кластеров. Дендрограмму также называют древовидной схемой, деревом объединения кластеров, деревом иерархической структуры. Дендрограмма представляет собой вложенную группировку объектов, которая изменяется на различных уровнях иерархии. Существует много способов построения дендрограмм. В дендрограмме объекты могут располагаться вертикально или горизонтально.

Пример. Предположим, 10 студентам предложили оценить проведенное с ними занятие по двум критериям: увлекательность и полезность. Для оценки использовалась 10 балльная шкала. Полученные данные графически представлены в виде графика двумерного рассеивания (рис.2).

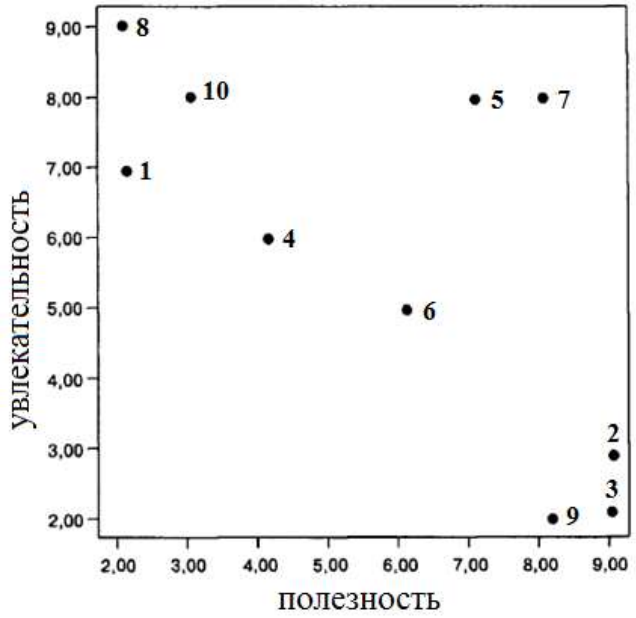

Рис. 2. Данные примера

Конечно, классификация объектов по результатам измерения всего двух переменных не требует применения кластерного анализа: группировки и так можно выделить путем визуального анализа. Так, в данном случае наблюдаются четыре группировки: 9, 2, 3 занятие полезное, но не увлекательное; 1, 10, 8 занятие увлекательное, но бесполезное; 5, 7 занятие и полезное, и увлекательное; 4, 6 занятие умеренно увлекательное и умеренно полезное. Даже для трех переменных можно обойтись и без кластерного анализа. Но для 4 и более переменных визуальный анализ данных практически невозможен. Тем не менее, общий принцип классификации объектов при помощи кластерного анализа не зависит от количества измеренных признаков, так как непосредственной информацией для этого метода являются различия между классифицируемыми объектами.

Основным результатом применения иерархического кластерного анализа является дендрограмма – графическое изображение последовательности объединения объектов в кластеры. На дендрограмме номера объектов следуют по вертикали. По горизонтали отмечены расстояния (в условных единицах), на которых происходит объединение объектов в кластеры.

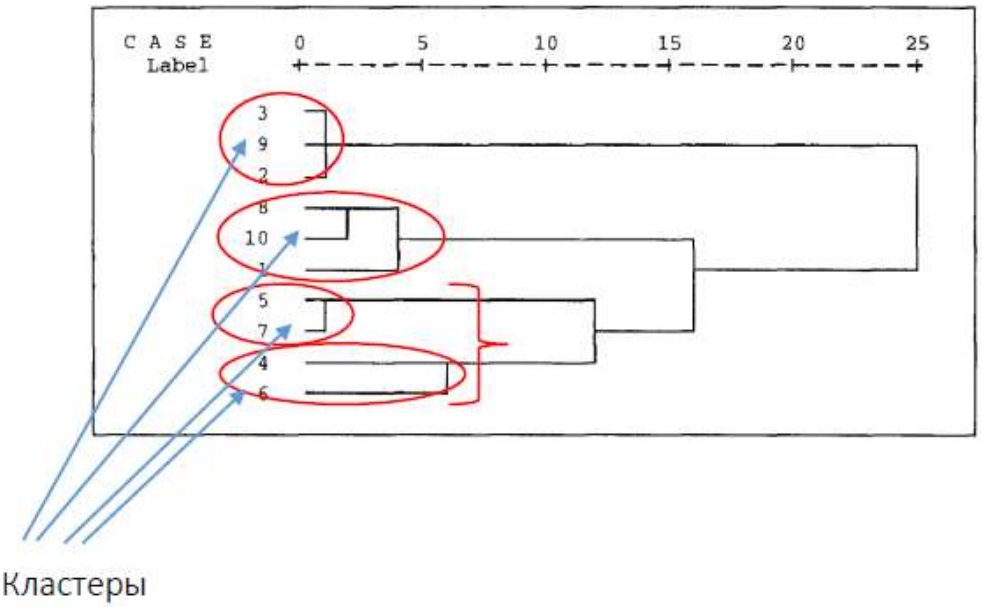

Рис.3. Дендрограмма примера

На первых шагах происходит образование кластеров: (3,9,2) и (5,7). Далее образуется кластер (8, 10, 1) — расстояния между этими объектами больше, чем между теми, которые были объединены на предыдущих шагах. Следующий кластер — (4, 6). Далее в один кластер объединяются кластеры (5, 7) и (4, 6), и т.д. Процесс заканчивается объединением всех объектов в один кластер. Количество кластеров определяет по дендрограмме сам исследователь. Так, судя по дендрограмме, в данном случае можно выделить три или четыре кластера.

Существуют различные методы иерархического кластерного анализа. Каждый метод дает свои результаты кластеризации, но три из них являются наиболее типичными (одиночная связь, полная связь, средняя связь). Проанализируем результаты применения этих методов к одним и тем же данным из рассмотренного выше примера.

Расстояния в иерархической кластеризации

Дерево строится от листьев к корню. В начальный момент времени каждый объект содержится в собственном кластере. Далее происходит итеративный процесс слияния двух ближайших кластеров до тех пор, пока все кластеры не объединятся в один или не будет найдено необходимое число кластеров. На каждом шаге необходимо уметь вычислять расстояние между кластерами и пересчитывать расстояние между новыми кластерами. Расстояние между одноэлементными кластерами определяется через расстояние между объектами: $R(\{x\},\{y\}) = \rho(x,y)$. Для вычисления расстояния $R(U,V)$ между кластерами U и V на практике используются различные функции в зависимости от специфики задачи.

Функции расстояния между кластерами:

- *Метод одиночной связи.* Метод ближнего соседа или одиночная связь. Здесь расстояние между двумя кластерами определяется расстоянием между двумя наиболее близкими объектами (ближайшими соседями) в различных кластерах. Этот метод позволяет выделять кластеры сколь угодно сложной формы при условии, что различные части таких кластеров соединены цепочками близких друг к другу элементов.

$$R_{min}(U,V) = \min_{u \in U, v \in V} \rho(u,v)$$

где $\rho(u,v)$ – расстояяние между $u \in U$ и $v \in V$; U и V различные кластеры

На рисунке 4 приведен результат применения метода.

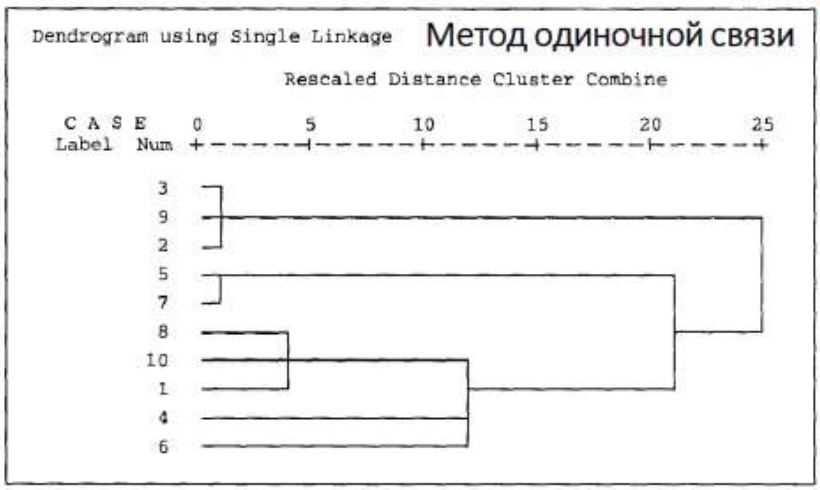

Рис.4. Метод одиночной связи

Сопоставляя эту дендрограмму с рисунком 3, приведенным выше, можно заметить, что объект 4 присоединяется к кластеру (8, 10, 1) и на том же расстоянии — к объекту 6 в связи с тем, что расстояние от объекта 4 до объекта 6 такое же, что и до объекта 1. Из рисунка 4 видно, что метод имеет тенденцию к образованию длинных кластеров «цепочного» вида. Таким образом, метод имеет тенденцию образовывать небольшое число крупных кластеров. К особенностям метода можно отнести и то, что результаты его применения часто не дают возможности определить, как много кластеров находится в данных.

- *Метод полной связи* часто называют методом «дальнего соседа». Правило объединения этого метода подразумевает, что новый объект присоединяется к тому кластеру, самый далекий элемент которого находится ближе к новому объекту, чем самые далекие элементы других кластеров. Это правило является противоположным предыдущему и более жестким. Поэтому здесь наблюдается тенденция к выделению большего числа компактных кластеров, состоящих из наиболее похожих элементов.

$$R_{max}(U,V) = \max_{u \in U, v \in V} \rho(u,v)$$

Сравним результат применения метода полной связи (рис. 5), метода одиночной связи (рис. 4) и фактическую конфигурацию объектов.

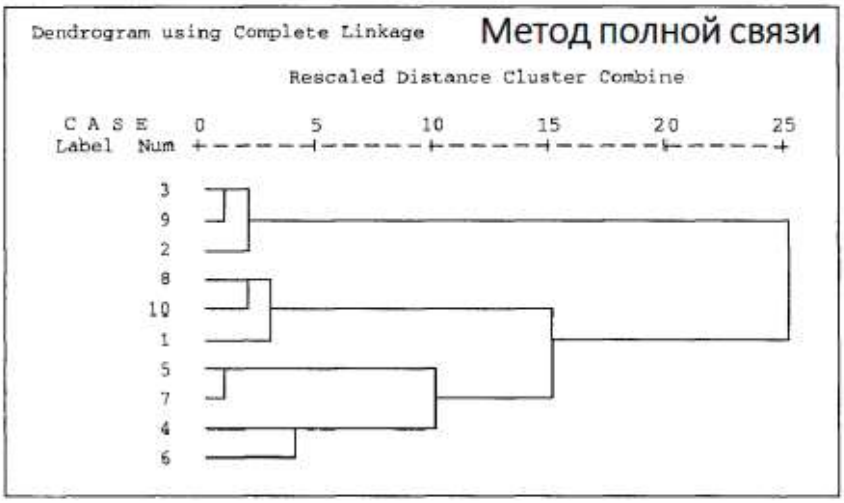

Рис.5. Метод полной связи

Различия в работе методов проявляются прежде всего в отношении объектов 4 и 6. Метод полной связи объединяет их в отдельный кластер и соединяет с кластером (5, 7) раньше, чем с кластером (8, 10, 1) — в отличие от метода одиночной связи. Объект 4 присоединяется сначала к объекту 6, потому что этот последний к нему ближе, чем самый дальний объект кластера (8, 10,1). На этом же основании кластер (4, 6) присоединяется к кластеру (5, 7), потому что

самый дальний объект 6 кластера (4, 6) ближе к самому дальнему объекту 7 кластера (5, 7), чем к самому дальнему объекту 8 кластера (8, 10, 1).

- *Метод средней связи* или межгрупповой связи занимает промежуточное положение относительно крайностей методов одиночной и полной связи. На каждом шаге вычисляется сред нее арифметическое расстояние между каждым объектом из одного кластера и каждым объектом из другого кластера. Объект присоединяется к данному кластеру, если это среднее расстояние меньше, чем среднее расстояние до любого другого кластера. По своему принципу этот метод должен давать более точные результаты классификации, чем остальные методы. То, что объединение кластеров в методе средней связи происходит при расстоянии большем, чем в методе одиночной связи, но меньшем, чем в методе полной связи, и объясняет промежуточное положение этого метода.

$$R_{avg}(U,V) = \frac{1}{|U| \times |V|} \sum_{u \in U} \sum_{v \in V} \rho(u,v)$$

Результат применения метода изображен на рисунке 6 (именно он был использован изначально для визуализации результатов кластерного анализа). Поскольку объектов в нашем примере немного, результаты применения методов полной и средней связи различаются не значительно.

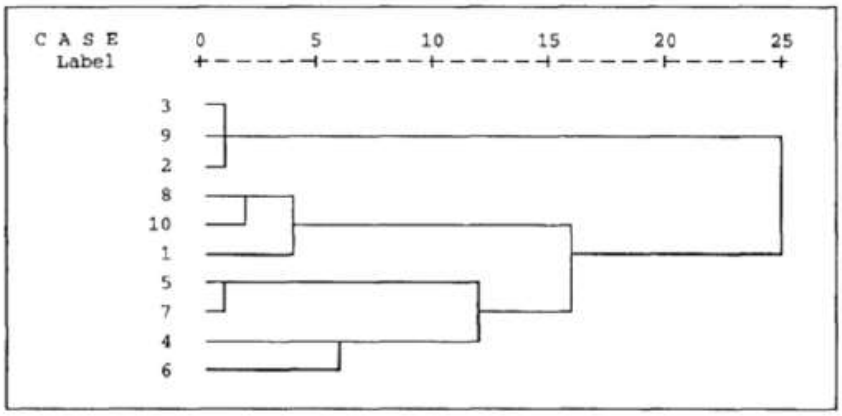

Рис.6. Метод полной связи

- *Центроидный метод.* Расстояние между двумя кластерами определяется как евклидово расстояние между центрами (средними) этих кластеров:

$$R_c(U,V) = \rho^2 \left(\sum_{u \in U} \frac{u}{|U|} \sum_{v \in V} \frac{v}{|V|} \right)$$

Существуют: Невзвешенный центроидный метод. В этом методе расстояние между двумя кластерами определяется как расстояние между их центрами тяжести. Взвешенный центроидный метод (медиана). Этот метод идентичен

предыдущему, за исключением того, что при вычислениях используются веса для учёта разницы между размерами кластеров (т.е. числами объектов в них). Поэтому, если имеются (или подозреваются) значительные отличия в размерах кластеров, этот метод оказывается предпочтительнее предыдущего.

- *Метод Уорда.* В этом методе в качестве целевой функции применяют внутригрупповую сумму квадратов отклонений, которая есть ни что иное, как сумма квадратов расстояний между каждой точкой (объектом) и средней по кластеру, содержащему этот объект. На каждом шаге объединяются такие два кластера, которые приводят к минимальному увеличению целевой функции, т.е. внутригрупповой суммы квадратов. Этот метод направлен на объединение близко расположенных кластеров.

$$R_{ward}(U,V) = \frac{|U| \times |V|}{|U| + |V|}\rho^2\left(\sum_{u\in U}\frac{u}{|U|}\sum_{v\in V}\frac{v}{|V|}\right)$$

Формула Ланса-Уильямса

В наиболее общем виде способы задания расстояния между кластерами даются формулой Ланса-Уильямса. На каждом шаге необходимо уметь быстро подсчитывать расстояние от образовавшегося кластера $W = U \cup V$ до любого другого кластера S, используя известные расстояния с предыдущих шагов. Это легко выполняется при использовании формулы, предложенной Лансом и Уильямсом в 1967 году.

$$R(W,S) = \alpha_U \cdot R(U,S) + \alpha_V \cdot R(V,S) + \beta \cdot R(U,V) + \gamma \cdot |R(U,S) - R(V,S)|$$

где $\alpha_U, \alpha_V, \beta, \gamma$ – числовые параметры.

- Метод одиночной связи

$$\alpha_U = \frac{1}{2}, \alpha_V = \frac{1}{2}, \beta = 0, \gamma = -\frac{1}{2}$$

- Метод полной связи

$$\alpha_U = \frac{1}{2}, \alpha_V = \frac{1}{2}, \beta = 0, \gamma = -\frac{1}{2}$$

- Метод средней связи

$$\alpha_U = \frac{|U|}{|W|}, \alpha_V = \frac{|V|}{|W|}, \beta = 0, \gamma = 0$$

- Центроидный метод

$$\alpha_U = \frac{|U|}{|W|}, \alpha_V = \frac{|V|}{|W|}, \beta = -\alpha_U \cdot \alpha_V, \gamma = 0$$

- Метод Уорда

$$\alpha_U = \frac{|S| + |U|}{|S| + |W|}, \alpha_V = \frac{|S| + |V|}{|S| + |W|}, \beta = \frac{-|S|}{|S| + |W|}, \gamma = 0$$

Агломеративные методы

Алгоритм кластеризации с использованием представителей CURE

Clustering Using Representatives

Чтобы избежать проблем с неоднородными размерами или формами кластеров, CURE использует алгоритм иерархической кластеризации, который принимает компромиссное решение между центром тяжести и всеми крайностями. В алгоритме CURE выбирается постоянная с точек кластера с хорошим распределением и эти точки стягиваются к центру тяжести кластера на некоторое значение. Точки после стягивания используются как представители кластера. Кластеры с ближайшей парой представителей объединяются на каждом шаге алгоритма иерархической кластеризации CURE. Это даёт возможность алгоритму CURE правильно распознавать кластеры и делает его менее чувствительным к выбросам.

За счет использования точек-представителей алгоритм CURE устойчив к выбросам и может выделять кластеры сложной формы и различных размеров.

Описание шагов

Шаг 1. Если все данные использовать сразу как входные для CURE, то эффективность алгоритма будет низкая, а время выполнения большим. Поэтому на первом шаге мы случайным образом выбираем часть точек, которые помещаются в память, затем группируем наиболее похожие с помощью иерархического метода в заранее заданное число кластеров. Дальше работаем с кластерами.

Шаг 2. Для каждого кластера выбираем с точек-представителей, максимально удаленных друг от друга. Чисто с остается постоянным.

Шаг 3. Объединяем кластеры с наиболее похожими наборами точек-представителей. Если не достигнуто нужное число кластеров, то перейти на шаг 2.

Выбранные точки сдвигаются на следующем шаге на α к центроиду кластера. Алгоритм становится основанным на методе поиска центроида при $\alpha = 1$, и основанным на всех точках кластера при $\alpha = 0$ (рис. 7).

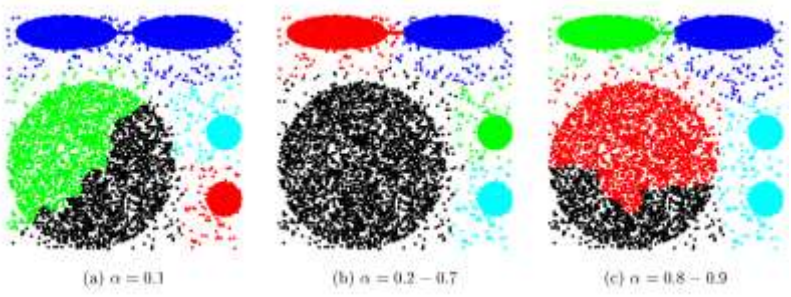

Рис.7. Пример работы алгоритма для разных параметров α [1].

Таким образом, для каждого i-го кластера вычисляется центроида μ_i по формуле:

$$\mu_i = \frac{1}{s_i} \sum_{j \in S_i} x^j$$

где S_i – номера строк матрицы x, входящие в i-й кластер; s_i – количество элементов в множестве S_i; $x^j - j$-я строка матрицы x.

Сдвиг представителей к центроиде происходит следующим образом.

Для каждого представителя x^y из k-го кластера выполняется сдвиг к его центроиде μ_k в α раз:

$$\hat{x}^y = \alpha \mu_k + (1 - \alpha) x^y.$$

Получаем множество $\hat{R}_k$ из с точек, которые будут далее использоваться как представители этого кластера.

Алгоритм устойчивой кластеризации с использованием связей ROCK (RObust Clustering using linKs)

Данный алгоритм принадлежит классу алгоритмов кластеризации, целью которых является разбиение данных на некоторое заранее заданное число групп. Подобные методы часто используются для анализа данных в различных областях, в том числе в маркетинге. Основная особенность алгоритма ROCK заключается в использовании связей между точками (количество общих соседей), в отличие от методов, базирующихся на различных метриках, таких как расстояние между точками (Евклидово и прочие). Такой подход улучшает определение глобальных зависимостей, а также наиболее эффективен при рассмотрении данных, свойства которых принимают достаточно малое конечное количество значений.

Одним из основных понятий для алгоритма ROCK является соседство двух точек. Пусть нам дана функция схожести $sim(p_i, p_j)$, принимающая значения от 0 до 1, которая выражает схожесть или близость объектов(точек) p_i и p_j. Предполагается, что 1 соответствует абсолютной близости, и 0 – наоборот. Тогда при некоторой границе θ между 0 и 1, если $sim(p_i, p_j) \geq \theta$, то p_i и p_j будут соседними точками. Выбор Функции $sim(p_i, p_j)$ и граничного значения θ зависит входных данных и особенности реализации.

Функция, определяющая соседство двух точек может быть:

$$neib(p_i, p_j) = \begin{cases} 1, \text{sim}(p_i, p_j) \geq \theta \\ 0, \text{sim}(p_i, p_j) < \theta \end{cases}$$

Вторым ключевым понятием являются связи. Функция связи $link(p_i, p_j)$ определяется как количество общих соседей у p_i и p_j. Из такого определения

сразу видно, что чем больше значение связи, тем больше вероятность, что эти точки принадлежат одному и тому же кластеру.

Функция, определяющая количество связей между точками:

$$link(p_i, p_j) = \sum_{s \in S} neib(p_i, s) neib(p_j, s)$$

Такой подход является более глобальным, по сравнению с использованием только близости двух точек, что позволяет снизить число ошибок, особенно в тех случаях, когда кластеры имеют несколько близких точек.

Алгоритм состоит из двух основных этапов. Изначально имеется n точек и k - желаемое число кластеров. На первом этапе вычисляются значения связей $link(p_i, p_j)$ между всеми парами точек, каждая точка объявляется отдельным кластером. Для каждого кластера i создается локальная куча $q[i]$ (это специализированная структура данных типа дерево), которая содержит все такие кластеры j, что связь между ними не нулевая. Кроме этого, создается глобальная куча Q, содержащая все кластеры. После этого алгоритм переходит ко второму этапу. Вторая часть представляет из себя цикл, на каждом шаге которого объединяются два кластера с максимальным значением функции полезности $g(i, j)$, после чего вносятся соответствующие изменения в кучи. Алгоритм завершает работу в двух случаях: когда осталось k кластеров, или когда все связи между оставшимися кластерами равны нулю.

Определим функцию связи для двух подмножеств C_i, C_j:

$$link[C_i, C_j] = \sum_{p_q \in C_i, p_r \in C_j} link(p_i, p_j)$$

Введем функцию полезности, выражающую то, насколько выгодно объединить подмножества C_i, C_j:

$$g(C_i, C_j) = \frac{link[C_i, C_j]}{(n_i + n_j)^{1+2f(\theta)} - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}}$$

В данном алгоритме сначала все элементы разбиваются на n подмножеств, затем делается $n - k$ шагов, на каждом из которых объединяются два подмножества, для которых значение функции полезности $g(i, j)$ наибольшее.

Рассмотрим алгоритм в виде информационного графа и введем обозначения для рис. :

CN – вершина построения соседей: заполнение $nbrlist[i]$.

CLM – вершина построения матрицы связей: заполнение $link[i][j]$.

CLH – вершина построения локальных куч: заполнение q[i].

CGH – вершина построения глобальной кучи: заполнение Q.

GC – вершина поиска и слияние топ 2 кластеров: построение w.

RL – пересчет матрицы связей: заполнение link[x,w].

UPDATE – обновление глобальной кучи: обновление Q.

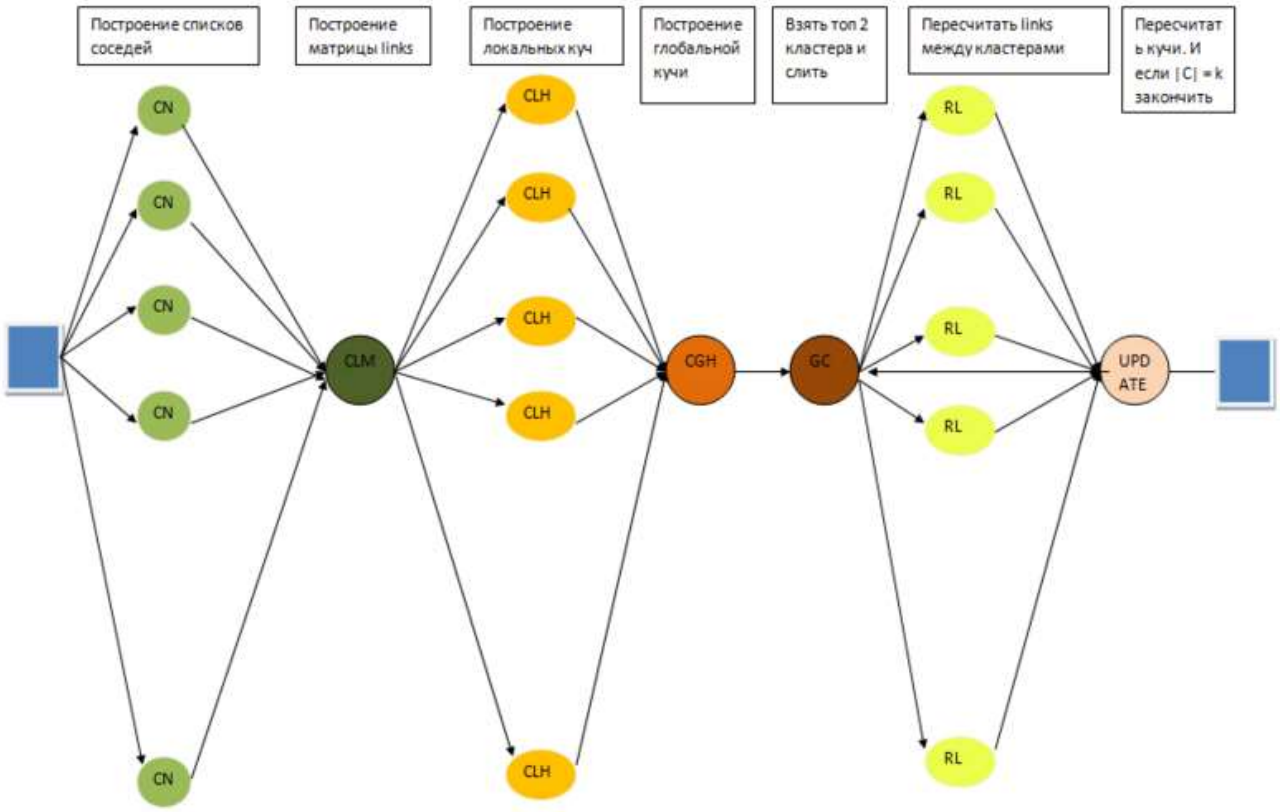

Рис 8. Граф в общем виде для n точек и k кластеров.

Графо-ориентированный алгоритм кластеризации CHAMELEON (Хамелион)

Иерархические алгоритмы кластеризации строят не одно разбиение множества данных на кластеры, а систему вложенных разбиений. CHAMELEON является агломеративным методом (снизу вверх), когда после первого разбиения на мелкие кластеры пары близлежащих кластеров итеративно объединяются в общий кластер. На первом этапе CHAMELEON использует алгоритм разделения графа для получения набора относительно малых кластеров. На втором этапе, для объединения кластеров, полученных на первом этапе, в настоящие кластеры, используется иерархическая агломеративная кластеризация. Таким образом, алгоритм, фактически, является гибридным, построенным комбинацией графо-ориентированных и классических иерархических методов.

На первом этапе, согласно графо-ориентированному подходу, происходит построение графа на матрице сходства объектов по принципу k ближайших соседей. Две вершины такого графа соединяет ребро, если объект,

соответствующий любой из этих вершин попадает в число k наиболее близких объектов для объекта, соответствующего другой вершине из данной пары.

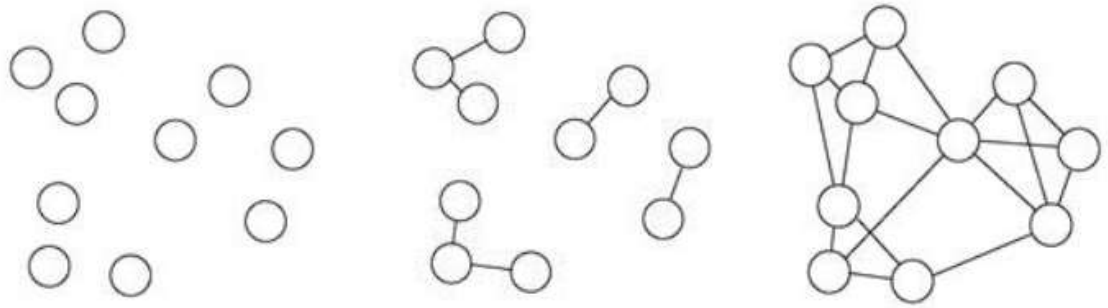

Рис 9. Объекты в пространстве – слева, граф по принципу 1-го ближайшего соседа – в середине, граф по принципу 3-х ближайших соседей – справа.

Далее, алгоритм разделяет полученный граф на множество сравнительно малых подграфов. Разделение происходит последовательно. На каждом шаге выбирается подграф, содержащий наибольшее число вершин. Этот граф разделяется на два подграфа, так, что разделитель ребер графа минимален и каждый из получаемых подграфов содержит не менее некоторого процента вершин исходного графа. Процесс разделения останавливается, когда наибольший подграф содержит меньше некоторого заданного числа вершин. Полученное множество связных графов считается множеством начальных кластеров, на котором требуется провести последовательное иерархическое объединение.

На втором этапе, алгоритм последовательно объединяет кластеры, используя значение их относительной взаимной связности и относительного взаимного сходства. Только если оба эти значения достаточно высоки у двух кластеров, они объединяются.

Относительная взаимная связность пары кластеров определяется абсолютной взаимной связностью кластеров, нормализованной с учетом внутренней связности каждого кластера (подграфа). Нормализация используется для того, чтобы исключить тенденцию к преимущественному объединению крупных кластеров, у которых, определенно, значение взаимной связности будет больше, вследствие их размера.

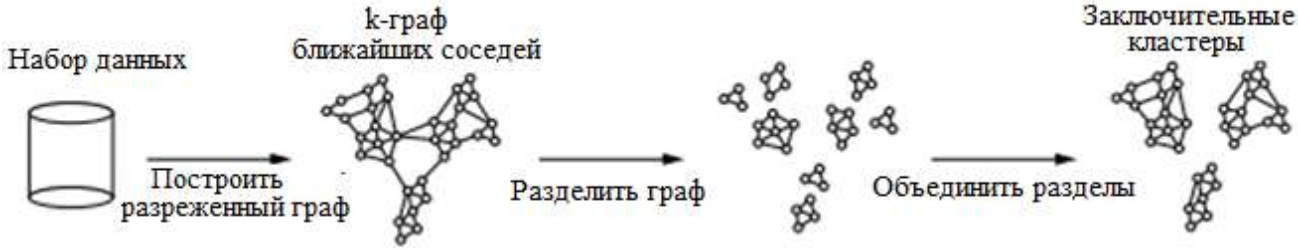

Рис 9. Алгоритм CHAMELEON

Дивизимные (делимые) методы

Сбалансированное итеративное сокращение и кластеризация с помощью иерархий BIRCH

Это алгоритм интеллектуального анализа данных без учителя, используемый для осуществления иерархической кластеризации на наборах данных большого размера. Преимуществом BIRCH является возможность метода динамически кластеризовать по мере поступления многомерных метрических точек данных в попытке получить кластеризацию лучшего качества для имеющегося набора ресурсов (памяти и временных рамок). В большинстве случаев алгоритм BIRCH требует одного прохода по базе данных.

Каждое решение кластеризации локально и осуществляется без просмотра всех точек данных и существующих на текущий момент кластеров. Метод работает на наблюдениях, пространство данных которых обычно не однородно заполнено и не каждая точка данных одинаково важна. Метод позволяет использовать всю доступную память для получения наиболее точных возможных подкластеров при минимизации цены ввода/вывода. Метод является инкрементальным и не требует наличия полного набора данных сразу.

Алгоритм BIRCH берёт в качестве входа набор из N точек данных, представленный как вещественные вектора, и желаемое число кластеров K. Алгоритм разбит на четыре фазы, вторая из которых не обязательна.

Первая фаза строит CF дерево точек данных, высоко сбалансированную древесную структуру, определённую следующим образом:

- Если дан набор N d-мерных точек данных, признак кластеризации CF (Clustering Feature – Функция кластеризации) набора определяется как тройка $CF = (N, LS, SS)$, где $\overrightarrow{LS} = \sum_{i=1}^{N} \overrightarrow{X_i}$ является линейной суммой, а $\overrightarrow{SS} = \sum_{i=1}^{N} \left(\overrightarrow{X_i}\right)^2$ является суммой квадратов точек данных

- Признаки кластеризации организуются в CF-дерево, высоко сбалансированное дерево с двумя параметрами: коэффициентом ветвления B и порогом T. Каждый нелистовой узел состоит максимум из B входов вида $[CF_i, \text{child}_i]$, где child_i является указателем на его i-ого потомка, а CF_i является признаком кластеризации, представляющим связанный подкластер. Лист содержит не более L входов, каждый вида $[CF_i]$. Он также имеет два указателя, prev (предыдущий) и next (следующий), которые используются для соединения в цепь все листы. Размер дерева зависит от параметра порога T. Требуется, чтобы узел A вмещался на страницу размера P. B и L определяются значением P. Таким образом, P может меняться для настройки производительности. Это очень компактное

представление набора данных, поскольку каждый лист не является отдельной точкой данных, а является подкластером.

На втором шаге алгоритм просматривает все листья в начальном CF-дереве, чтобы построить меньшее CF-дерево путём удаления выпадений и группирования переполненных подклассов в бóльшие подклассы. Этот шаг в исходном представлении BIRCH помечен как необязательный.

На третьем шаге используется существующий алгоритм для кластеризации всех листов. Здесь применяется агломерирующий иерархический алгоритм кластеризации непосредственно к подкластерам, представленным их CF-векторами. Это также обеспечивает гибкость, позволяющую пользователю указать либо желаемое число кластеров, либо желаемый порог диаметра кластеров. После этого шага получаем набор кластеров, которые содержат главные схемы распределения в данных. Однако могут существовать небольшие локальные неточности, которые могут быть обработаны необязательным шагом 4. На шаге 4 центры тяжести кластеров, полученных на шаге 3, используются как зародыши и точки перераспределения точек данных для получения нового набора кластеров. Шаг 4 обеспечивает также возможность отбрасывания выбросов. То есть точка, которая слишком далека от ближайшего зародыша, может считаться выбросом.

Вычисление признаков кластеров происходит следующим образом: если дано только $CF = \left(N, \overrightarrow{LS}, \overrightarrow{SS}\right)$, те же измерения могут быть получены без знания истинных значений.

- Центроид:

$$\vec{C} = \frac{\sum_{i=1}^{N} \overrightarrow{X_i}}{N} = \frac{\overrightarrow{LS}}{N}$$

- Радиус:

$$R = \sqrt{\frac{\sum_{i=1}^{N} \left(\overrightarrow{X_i} - \vec{C}\right)^2}{N}} = \sqrt{\frac{N - \vec{C}^2 + \overrightarrow{SS} - 2 \times \vec{C} \times \overrightarrow{LS}}{N}}$$

- Среднее расстояние между кластерами

$CF_1 = \left(N_1, \overrightarrow{LS_1}, \overrightarrow{SS_1}\right)$ и $CF_2 = \left(N_2, \overrightarrow{LS_2}, \overrightarrow{SS_2}\right)$:

$$D_2 = \sqrt{\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \left(\overrightarrow{X_i} - \overrightarrow{Y_i}\right)^2}{N_1 \times N_2}} = \sqrt{\frac{N_1 \times \overrightarrow{SS_2} + N_2 \times \overrightarrow{SS_1} - 2 \times \overrightarrow{LS_1} \times \overrightarrow{LS_2}}{N_1 \times N_2}}$$

Алгоритм поиска минимального остовного дерева
(minimum spanning tree, MST)

Модели графов, в которых с каждым ребром связаны веса или стоимости, используются во многих приложениях. В картах авиалиний, в которых ребрами отмечены авиарейсы, такие веса означают расстояния или стоимости билетов. В электронных схемах, где ребра представляют проводники, веса могут означать длину проводника, его стоимость или время прохода сигнала. В задачах календарного планирования веса могут представлять время или трудоемкость либо выполнения задачи, либо ожидания ее завершения.

Взвешенный неориентированный граф представляет собой множество взвешенных ребер. MST-дерево есть множество ребер минимального общего веса, которые соединяют все вершины (в списке ребер выделены черным, утолщенные ребра на чертеже рис, 10).

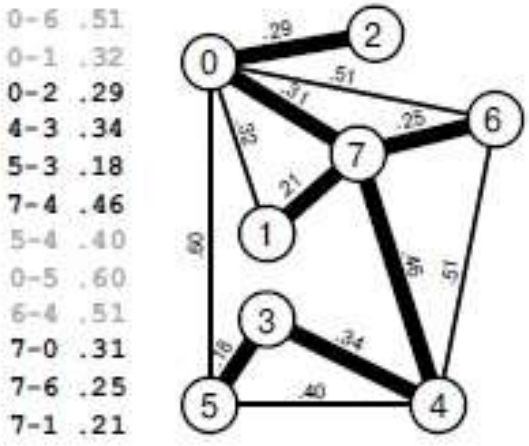

Рис. 10. Взвешенный неориентированный граф и его MST

Определение. Минимальное остовное дерево (minimal spanning tree — MST, другие варианты перевода — минимальный остов, минимальный каркас, минимальный скелет) взвешенного графа есть остовное дерево, вес которого (сумма весов его ребер) не превосходит вес любого другого остовного дерева.

Существует несколько алгоритмов для нахождения минимального остовного дерева.

- Алгоритм Прима,
- Алгоритм Краскала (или алгоритм Крускала),
- Алгоритм Борувки,
- Алгоритм обратного удаления (получение минимального остовного дерева из связного рёберно взвешенного графа).

Алгоритм Прима и поиск по приоритету

Алгоритм Прима, похоже, наиболее прост для реализации из всех алгоритмов поиска MST и рекомендуется для насыщенных графов. В нем используется сечение графа, состоящее из древесных вершин (выбранных для

MST) и недревесных вершин (еще не выбранных в MST-дерево). Вначале мы выбираем в качестве MST-дерева произвольную вершину, затем помещаем в MST минимальное перекрестное ребро (которое превращает недревесную вершину MST-дерева в древесную) и повторяем эту же операцию $V-1$ раз, пока все вершины не окажутся в дереве.

Здесь главное – найти кратчайшее расстояние от каждой недревесной вершины до дерева. Для реализации этой идеи нам потребуются такие структуры данных, которые предоставляют следующую информацию:

- Ребра дерева.
- Самое короткое дерево, соединяющее недревесную вершину с деревом.
- Длина этого ребра.

На рис. 11 показан пример построения MST-дерева с помощью алгоритма Прима.

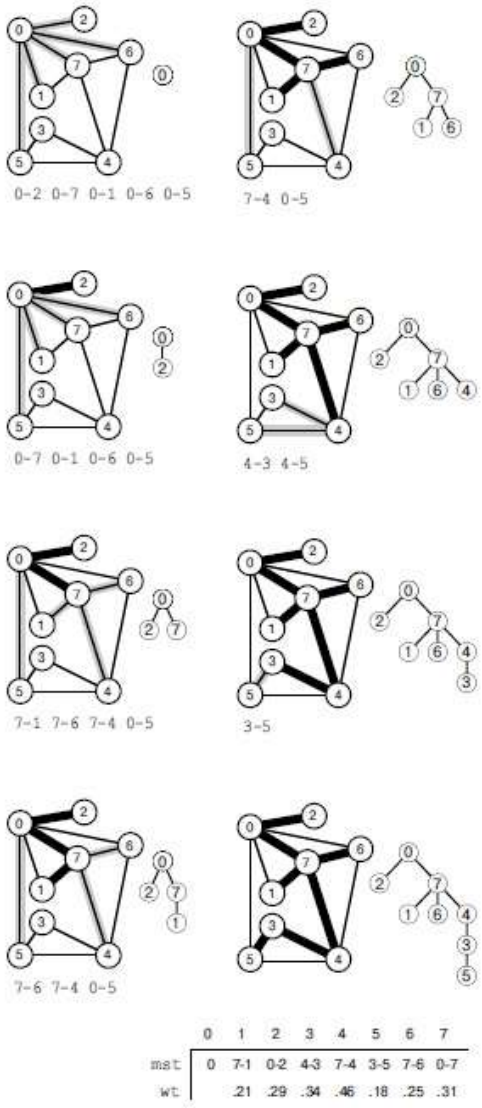

Рис. 11. Алгоритм Прима для вычисления MST

Первым шагом вычисления MST-дерева по алгоритму Прима в это дерево заносится вершина 0. Затем мы находим все ребра, которые соединяют 0 с другими вершинами (еще не включенными в это дерево), и выбираем из них самое короткое (слева вверху). Ребра, соединяющие древесные вершины с недревесными (накопитель), заштрихованы и перечислены под каждым чертежом графа. Для простоты ребра из накопителя перечисляются в порядке возрастания их длины, то есть самое короткое ребро — первое в этом списке. В различных реализациях алгоритма Прима используются различные структуры данных для хранения этого списка и определения минимального ребра. Вторым шагом самое короткое ребро 0-2 переносится (вместе с его конечной вершиной) из накопителя в дерево (вторая диаграмма сверху слева). На третьем шаге ребро 0-7 переносится из накопителя в дерево, в накопителе ребро 0-1 заменяется на 7-1, ребро 0-6 на 7-6 (поскольку включение вершины 7 в дерево приближает к дереву вершины 1 и 6), а ребро 7-4 заносится в накопитель (поскольку добавление вершины 7 в дерево превращает 7-4 в ребро, которое соединяет древесную вершину с недревесной) (третья диаграмма сверху слева). Далее, мы переносим в дерево ребро 7-1 (слева внизу). В завершение вычислений мы исключаем из очереди ребра 7-6, 7-4, 4-3 и 3-5, обновляя накопитель после каждой вставки для отражения обнаруженных более коротких или новых путей (справа, сверху вниз).

Алгоритм Крускала

Алгоритм Прима строит минимальное остовное дерево по одному ребру, находя на каждом шаге ребро, которое присоединяется к единственному растущему дереву. Алгоритм Крускала также строит MST, добавляя к нему по одному ребру, но в отличие от алгоритма Прима, он отыскивает ребро, которое соединяет два дерева в лесу, образованном растущими MST-поддеревьями. Построение начинается с вырожденного леса из V деревьев (каждое состоящее из одной вершины), а затем выполняется операция объединения двух деревьев (самыми короткими ребрами), пока не останется единственное дерево — MST.

На рис. 12 показан пример пошагового выполнения алгоритма Крускала. Разобщенный лес MST-поддеревьев постепенно объединяется в единственное дерево.

Пусть задан список ребер графа в произвольном порядке (левый список ребер). На первом шаге алгоритма Крускала они сортируются по весам (правый список ребер). Затем мы просматриваем ребра этого списка в порядке возрастания их весов, добавляя в MST ребра, которые не создают в нем циклов. Сначала мы добавляем ребро 5-3 (самое короткое ребро), потом 7-6 (слева), затем 0-2 (справа вверху) и 0-7 (справа, вторая диаграмма сверху). Ребро 0-1 со следующим по

величине весом создает цикл и поэтому не добавляется в дерево. Ребра, которые не включаются в MST, выделены в отсортированном списке серым цветом. Затем мы добавляем ребро 4-3 (справа, третья диаграмма сверху). Далее мы отбрасываем ребро 5-4, поскольку оно образует цикл, и потом добавляем 7-4 (справа внизу). Когда MST-дерево готово, любое ребро с большим весом образует цикл и поэтому будет отброшено. В отсортированном списке эти ребра помечены звездочками.

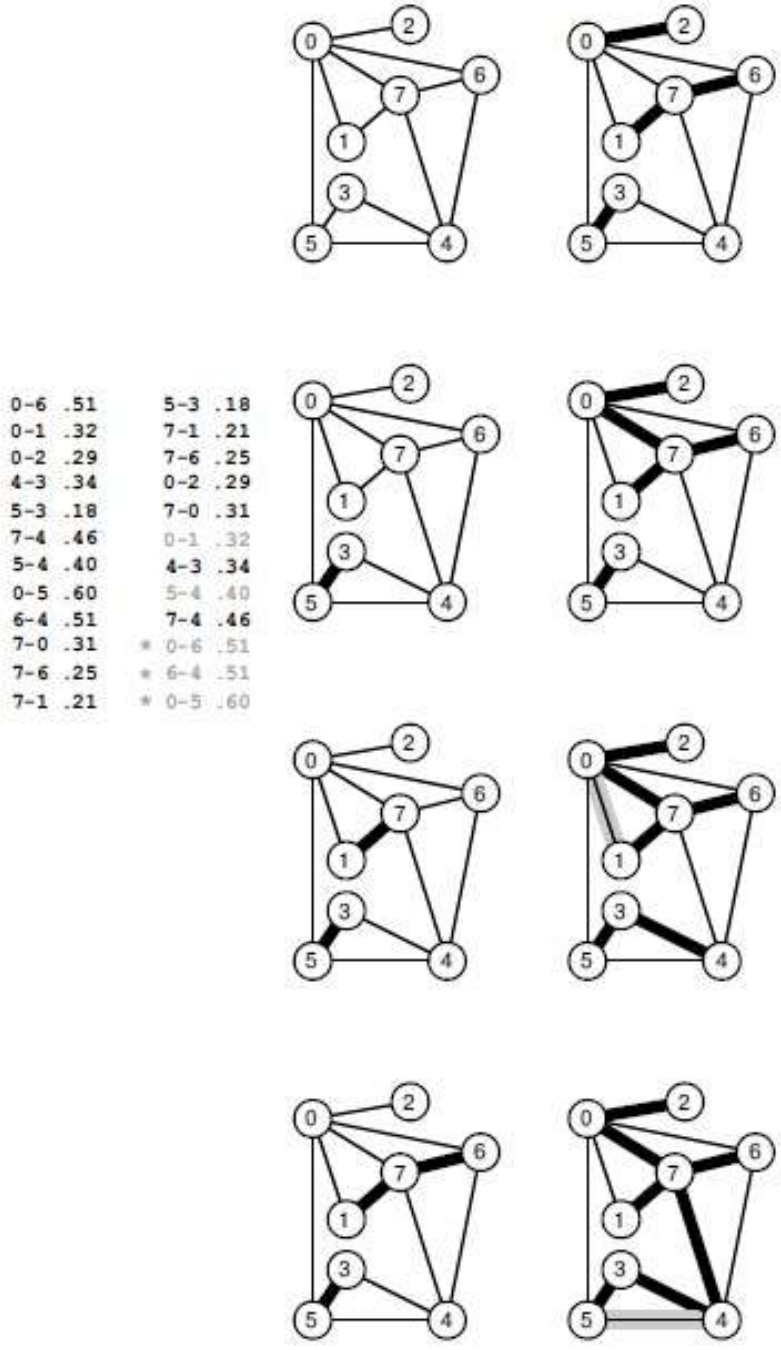

Рис. 12. Алгоритм Крускала вычисления MST