Лекции по алгоритмам кластеризации и многомерного шкалирования

К. В. Воронцов

24 июня 2010 г.

Материал находится в стадии разработки, может содержать ошибки и неточности. Автор будет благодарен за любые замечания и предложения, направленные по адресу `vokov@forecsys.ru`, либо высказанные в обсуждении страницы «Машинное обучение (курс лекций, К.В.Воронцов)» вики-ресурса `www.MachineLearning.ru`.

Содержание

1 Кластеризация и визуализация

Во многих прикладных задачах измерять степень сходства объектов существенно проще, чем формировать признаковые описания. Например, две строки с описанием молекул ДНК или протеинов гораздо легче сравнить непосредственно друг с другом, чем преобразовывать каждый из них в вектор признаков, и затем сравнивать эти векторы. То же самое можно сказать про тексты, временные ряды или растровые изображения.

Задача классификации объектов на основе их сходства друг с другом, когда принадлежность обучающих объектов каким-либо классам не задаётся, называется задачей *кластеризации*. В §1.1 рассматриваются статистические, иерархические и графовые алгоритмы кластеризации.

В §1.2 рассматриваются методы *многомерного шкалирования*, позволяющие восстанавливать признаковые описания объектов по матрице попарных расстояний между ними.

§1.1 Алгоритмы кластеризации

Задача *кластеризации* (или обучения без учителя) заключается в следующем. Имеется обучающая выборка $X^\ell = \{x_1, \ldots, x_\ell\} \subset X$ и функция расстояния между объектами $\rho(x, x')$. Требуется разбить выборку на непересекающиеся подмножества, называемые *кластерами*, так, чтобы каждый кластер состоял из объектов, близких по метрике ρ, а объекты разных кластеров существенно отличались. При этом каждому объекту $x_i \in X^\ell$ приписывается метка (номер) кластера y_i.

Алгоритм кластеризации — это функция $a\colon X \to Y$, которая любому объекту $x \in X$ ставит в соответствие метку кластера $y \in Y$. Множество меток Y в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров, с точки зрения того или иного критерия качества кластеризации.

Решение задачи кластеризации принципиально неоднозначно, и тому есть несколько причин. Во-первых, не существует однозначно наилучшего критерия качества кластеризации. Известен целый ряд достаточно разумных критериев, а также ряд алгоритмов, не имеющих чётко выраженного критерия, но осуществляющих достаточно разумную кластеризацию «по построению». Все они могут давать разные результаты. Во-вторых, число кластеров, как правило, неизвестно заранее и устанавливается в соответствии с некоторым субъективным критерием. В-третьих, результат кластеризации существенно зависит от метрики ρ, выбор которой, как правило, также субъективен и определяется экспертом.

Кластеризация (обучение без учителя) отличается от классификации (обучения с учителем) тем, что метки исходных объектов y_i изначально не заданы, и даже может быть неизвестно само множество Y. В этом смысле задача кластеризации ещё в большей степени некорректно поставленная, чем задача классификации.

Цели кластеризации могут быть различными в зависимости от особенностей конкретной прикладной задачи:

- Понять структуру множества объектов X^ℓ, разбив его на группы схожих объектов. Упростить дальнейшую обработку данных и принятия решений, работая с каждым кластером по отдельности (стратегия «разделяй и властвуй»).

- Сократить объём хранимых данных в случае сверхбольшой выборки X^ℓ, оставив по одному наиболее типичному представителю от каждого кластера.
- Выделить нетипичные объекты, которые не подходят ни к одному из кластеров. Эту задачу называют одноклассовой классификацией, обнаружением нетипичности или *новизны* (novelty detection).

В первом случае число кластеров стараются сделать поменьше. Во втором случае важнее обеспечить высокую степень сходства объектов внутри каждого кластера, а кластеров может быть сколько угодно. В третьем случае наибольший интерес представляют отдельные объекты, не вписывающиеся ни в один из кластеров.

Во всех этих случаях может применяться иерархическая кластеризация, когда крупные кластеры дробятся на более мелкие, те в свою очередь дробятся ещё мельче, и т. д. Такие задачи называются *задачами таксономии* (taxonomy). Результатом таксономии является не простое разбиение множества объектов на кластеры, а древообразная иерархическая структура. Вместо номера кластера объект характеризуется перечислением всех кластеров, которым он принадлежит, от крупного к мелкому. Классическим примером таксономии на основе сходства является систематизация живых существ, предложенная Карлом Линнеем в середине XVIII века. В современном представлении биологическая иерархия имеет около 30 уровней, 7 из них считаются основными: царство, тип, класс, отряд, семейство, род, вид. Таксономии строятся во многих областях знания, чтобы упорядочить информацию о большом количестве объектов. Мы будем рассматривать алгоритмы *иерархической кластеризации*, позволяющие автоматизировать процесс построения таксономий.

Типы кластерных структур. Попытаемся составить реестр различных типов кластерных структур, которые могут возникать в практических задачах.

Различные алгоритмы кластеризации могут быть более или менее успешны в этих ситуациях. Простые алгоритмы, как правило, узко специализированы и дают адекватные результаты только в одной-двух ситуациях. Более сложные алгоритмы, такие как FOREL или агломеративная процедура Ланса-Вильямса, справляются с несколькими типами ситуаций. Однако создание алгоритма, успешно работающего во всех ситуациях без исключения, представляется трудной и едва ли разрешимой задачей [5]. Более современный обзор по методам кластеризации можно найти в [7].

1.1.1 Эвристические графовые алгоритмы

Обширный класс алгоритмов кластеризации основан на представлении выборки в виде графа. Вершинам графа соответствуют объекты выборки, а рёбрам — попарные расстояния между объектами $\rho_{ij} = \rho(x_i, x_j)$.

Достоинством графовых алгоритмов кластеризации является наглядность, относительная простота реализации, возможность вносить различные усовершенствования, опираясь на простые геометрические соображения.

Алгоритм выделения связных компонент. Задаётся параметр R и в графе удаляются все рёбра (i, j), для которых $\rho_{ij} > R$. Соединёнными остаются только наиболее близкие пары объектов. Идея алгоритма заключается в том, чтобы подобрать такое

Алгоритм 1.1. Алгоритм кратчайшего незамкнутого пути (КНП)

1: Найти пару точек (i, j) с наименьшим ρ_{ij} и соединить их ребром;
2: **пока** в выборке остаются изолированные точки
3: найти изолированную точку, ближайшую к некоторой неизолированной;
4: соединить эти две точки ребром;
5: удалить $K-1$ самых длинных рёбер;

значение $R \in [\min \rho_{ij}, \max \rho_{ij}]$, при котором граф развалится на несколько связных компонент. Найденные связные компоненты — и есть кластеры.

Связной компонентой графа называется подмножество его вершин, в котором любые две вершины можно соединить путём, целиком лежащим в этом подмножестве [cite?]. Для поиска связных компонент можно использовать стандартные алгоритмы поиска в ширину (алгоритм Дейкстры) или поиска в глубину [cite?].

Для подбора параметра R обычно рекомендуется построить гистограмму распределения попарных расстояний ρ_{ij}. В задачах с выраженной кластерной структурой эта гистограмма имеет два чётких пика: зона небольших внутриклассовых расстояний и зона больших межклассовых расстояний. Параметр R задаётся как расстояние, соответствующее точке минимума между этими пиками [4].

Отметим два недостатка этого алгоритма.

- Ограниченная применимость. Алгоритм выделения связных компонент наиболее подходит для выделения кластеров типа сгущений или лент. Наличие разреженного фона или «узких перемычек» между кластерами приводит к неадекватной кластеризации.

- Плохая управляемость числом кластеров. Для многих приложений удобнее задавать не параметр R, а число кластеров или некоторый порог «чёткости кластеризации». Управлять числом кластеров с помощью параметра R довольно затруднительно. Приходится многократно решать задачу при разных R, что отрицательно сказывается на временных затратах.

Алгоритм кратчайшего незамкнутого пути строит граф из $\ell-1$ рёбер так, чтобы они соединяли все ℓ точек и обладали минимальной суммарной длиной. Такой граф называется *кратчайшим незамкнутым путём* (КНП), минимальным покрывающим деревом или каркасом. Доказано [cite? Прим, 1967], что этот граф строится с помощью несложной процедуры, соответствующей шагам 1–4 Алгоритма 1.1. На шаге 5 удаляются $K-1$ самых длинных рёбер, и связный граф распадается на K кластеров.

В отличие от предыдущего алгоритма, число кластеров K задаётся как входной параметр. Его можно также определять графически, если упорядочить все расстояния, образующие каркас, в порядке убывания и отложить их на графике. Резкий скачок вниз где-то на начальном (левом) участке графика покажет количество наиболее чётко выделяемых кластеров.

Этот алгоритм, как и предыдущий, очень прост и также имеет ограниченную применимость. Наличие разреженного фона или «узких перемычек» между кластерами приводит к неадекватной кластеризации. Другим недостатком КНП является высокая трудоёмкость — для построения кратчайшего незамкнутого пути требуется $O(\ell^3)$ операций.

Алгоритм FOREL (ФОРмальный ЭЛемент) предложен Загоруйко и Ёлкиной в 1967 году при решении одной прикладной задачи в области палеонтологии. Алгоритм имеет многочисленные вариации, подробно описанные в [2, 1]. В основе всех этих вариаций лежит следующая базовая процедура.

Пусть задана некоторая точка $x_0 \in X$ и параметр R. Выделяются все точки выборки $x_i \in X^\ell$, попадающие внутрь сферы $\rho(x_i, x_0) \leqslant R$, и точка x_0 переносится в центр тяжести выделенных точек. Эта процедура повторяется до тех пор, пока состав выделенных точек, а значит и положение центра, не перестанет меняться. Доказано, что эта процедура сходится за конечное число шагов. При этом сфера перемещается в место локального сгущения точек. Центр сферы x_0 в общем случае не является объектом выборки, потому и называется *формальным элементом*.

Для вычисления центра необходимо, чтобы множество объектов X было не только метрическим, но и линейным векторным пространством. Это требование естественным образом выполняется, когда объекты описываются числовыми признаками. Однако существуют задачи, в которых изначально задана только метрика, а сложение и умножение на число не определены на X. Тогда в качестве центра сферы можно взять тот объект обучающей выборки, для которого среднее расстояние до других объектов кластера минимально. Соответственно, шаг 6 заменяется на

$$x_0 := \arg\min_{x \in K_0} \sum_{x' \in K_0} \rho(x, x').$$

При этом заметно увеличивается трудоёмкость алгоритма. Если в линейном пространстве для вычисления центра требуется $O(\varkappa)$ операций, то в метрическом — $O(\varkappa^2)$, где $\varkappa$ — чисто точек в кластере. Алгоритм можно несколько ускорить, если заметить, что пересчёт центра при добавлении или удалении отдельной точки кластера требует лишь $O(\varkappa)$ операций, а в линейном пространстве — $O(1)$.

Различные варианты алгоритма FOREL отличаются способами объединения сфер в кластеры, способами варьирования параметра R, способами выбора начального приближения для точек x_0. В Алгоритме 1.2 представлен один из вариантов, в котором сферы строятся последовательно. На шаге 9 к центрам этих сфер применяется алгоритм КНП. С одной стороны, это решает проблему низкой эффективности КНП, так как сфер гораздо меньше, чем исходных объектов. С другой стороны, мы получаем более тонкую, двухуровневую, структуру кластеров: каждый кластер верхнего уровня распадается на более мелкие подкластеры нижнего уровня.

Другое преимущество этого алгоритма — возможность описывать кластеры произвольной геометрической формы. Варьируя параметр R, можно получать кластеризации различной степени детальности. Если кластеры близки по форме к шарам, можно сделать R достаточно большим. Для описания кластеров более сложной формы следует уменьшать R.

Алгоритм 1.2 довольно чувствителен к выбору начального положения точки x_0 для каждого нового кластера. Для устранения этого недостатка в [1] предлагается генерировать несколько (порядка 10..20) кластеризаций. Поскольку начальное положение центров выбирается случайным образом, эти кластеризации будут довольно сильно отличаться. Окончательно выбирается та кластеризация, которая доставляет наилучшее значение заданному функционалу качества.

Различные виды функционалов качества рассматриваются далее.

Алгоритм 1.2. Алгоритм FOREL

Инициализировать множество некластеризованных точек:
$U := X^\ell$;
пока в выборке есть некластеризованные точки, $U \neq \varnothing$:
  взять произвольную точку $x_0 \in U$ случайным образом;
  повторять
    образовать кластер — сферу с центром в x_0 и радиусом R:
    $K_0 := \{x_i \in U \mid \rho(x_i, x_0) \leqslant R\}$;
    поместить центр сферы в центр масс кластера:
    $x_0 := \frac{1}{|K_0|} \sum_{x_i \in K_0} x_i$;
  пока центр x_0 не стабилизируется;
  пометить все точки K_0 как кластеризованные:
  $U := U \setminus K_0$;
применить алгоритм КНП к множеству центров всех найденных кластеров;
каждый объект $x_i \in X^\ell$ приписать кластеру с ближайшим центром;

1.1.2 Функционалы качества кластеризации

Задачу кластеризации можно ставить как задачу дискретной оптимизации: необходимо так приписать номера кластеров y_i объектам x_i, чтобы значение выбранного функционала качества приняло наилучшее значение. Существует много разновидностей функционалов качества кластеризации, но нет «самого правильного» функционала. По сути дела, каждый метод кластеризации можно рассматривать как точный или приближённый алгоритм поиска оптимума некоторого функционала.

Среднее внутрикластерное расстояние должно быть как можно меньше:

$$F_0 = \frac{\sum_{i<j}[y_i = y_j]\rho(x_i, x_j)}{\sum_{i<j}[y_i = y_j]} \to \min.$$

Среднее межкластерное расстояние должно быть как можно больше:

$$F_1 = \frac{\sum_{i<j}[y_i \neq y_j]\rho(x_i, x_j)}{\sum_{i<j}[y_i \neq y_j]} \to \max.$$

Если алгоритм кластеризации вычисляет центры кластеров μ_y, $y \in Y$, то можно определить функционалы, вычислительно более эффективные.

Сумма средних внутрикластерных расстояний должна быть как можно меньше:

$$\Phi_0 = \sum_{y \in Y} \frac{1}{|K_y|} \sum_{i:\, y_i = y} \rho^2(x_i, \mu_y) \to \min,$$

где $K_y = \{x_i \in X^\ell \mid y_i = y\}$ — кластер с номером y. В этой формуле можно было бы взять не квадраты расстояний, а сами расстояния. Однако, если ρ — евклидова метрика, то внутренняя сумма в Φ_0 приобретает физический смысл момента инерции кластера K_y относительно его центра масс, если рассматривать кластер как материальное тело, состоящее из $|K_y|$ точек одинаковой массы.

Сумма межкластерных расстояний должна быть как можно больше:

$$\Phi_1 = \sum_{y \in Y} \rho^2(\mu_y, \mu) \to \max,$$

Алгоритм 1.3. Кластеризация с помощью EM-алгоритма

1: начальное приближение для всех кластеров $y \in Y$:
$w_y := 1/|Y|$;
$\mu_y :=$ случайный объект выборки;
$\sigma^2_{yj} := \frac{1}{\ell|Y|}\sum_{i=1}^{\ell}(f_j(x_i) - \mu_{yj})^2,\ \ j = 1, \ldots, n$;
2: **повторять**
3: E-шаг (expectation):
$$g_{iy} := \frac{w_y p_y(x_i)}{\sum_{z\in Y} w_z p_z(x_i)},\ \ y \in Y,\ \ i = 1, \ldots, \ell;$$
4: M-шаг (maximization):
$$w_y := \frac{1}{\ell}\sum_{i=1}^{\ell} g_{iy},\ \ y \in Y;$$
$$\mu_{yj} := \frac{1}{\ell w_y}\sum_{i=1}^{\ell} g_{iy} f_j(x_i),\ \ y \in Y,\ \ j = 1, \ldots, n;$$
$$\sigma^2_{yj} := \frac{1}{\ell w_y}\sum_{i=1}^{\ell} g_{iy}(f_j(x_i) - \mu_{yj})^2,\ \ y \in Y,\ \ j = 1, \ldots, n;$$
5: Отнести объекты к кластерам по байесовскому решающему правилу:
$y_i := \arg\max_{y\in Y} g_{iy},\ \ i = 1, \ldots, \ell$;
6: **пока** y_i не перестанут изменяться;

где μ — центр масс всей выборки.

На практике вычисляют отношение пары функционалов, чтобы учесть как межкластерные, так и внутрикластерные расстояния:

$$F_0/F_1 \to \min, \quad \text{либо} \quad \Phi_0/\Phi_1 \to \min.$$

1.1.3 Статистические алгоритмы

Статистические алгоритмы основаны на предположении, что кластеры неплохо описываются некоторым семейством вероятностных распределений. Тогда задача кластеризации сводится к разделению смеси распределений по конечной выборке.

Снова EM-алгоритм. Напомним основные гипотезы байесовского подхода к разделению смесей вероятностных распределений, см. также **??**.

Гипотеза 1.1 (о вероятностной природе данных). *Объекты выборки X^ℓ появляются случайно и независимо согласно вероятностному распределению, представляющему собой смесь распределений*

$$p(x) = \sum_{y\in Y} w_y p_y(x), \qquad \sum_{y\in Y} w_y = 1,$$

где $p_y(x)$ — функция плотности распределения кластера y, w_y — неизвестная априорная вероятность появления объектов из кластера y.

Конкретизируя вид распределений $p_y(x)$, чаще всего берут сферические гауссовские плотности. Это обычная практика — представлять кластеры в виде шаров.

Мы немного обобщим это представление и будем предполагать, что кластеры скорее похожи на эллипсоиды, оси которых направлены вдоль осей координат. Преимущество эллиптических гауссианов в том, что они обходят проблему выбора нормировки признаков. Нормировать можно немного по-разному, причём результат кластеризации существенно зависит от нормировки. Пока не произведена кластеризация, трудно понять, какая нормировка лучше. При использовании эллиптических гауссианов оптимальная нормировка подбирается самим алгоритмом кластеризации, индивидуально для каждого кластера.

Гипотеза 1.2 (о пространстве объектов и форме кластеров). *Каждый объект x из $X = \mathbb{R}^n$ описывается n числовыми признаками: $x \equiv \big(f_1(x), \ldots, f_n(x)\big)$. Каждый кластер $y \in Y$ описывается n-мерной гауссовской плотностью $p_y(x)$ с центром $\mu_y = (\mu_{y1}, \ldots, \mu_{yn})$ и диагональной матрицей ковариаций $\Sigma_y = \operatorname{diag}(\sigma_{y1}^2, \ldots, \sigma_{yn}^2)$:*

$$p_y(x) = (2\pi)^{-\frac{n}{2}} (\sigma_{y1} \cdots \sigma_{yn})^{-1} \exp\left(-\tfrac{1}{2}\rho_y^2(x, \mu_y)\right),$$

где $\rho_y^2(x, x') = \sum\limits_{j=1}^{n} \sigma_{yj}^{-2} |f_j(x) - f_j(x')|^2$ — взвешенное евклидово расстояние с весами σ_{yj}^{-2}.

При этих предположениях задача кластеризации совпадает с задачей разделения смеси вероятностных распределений, и для её решения можно применить EM-алгоритм **??**. Для оценивания параметров кластеров воспользуемся формулами, полученными в Теореме **??** как раз для случая эллиптических гауссианов. Реализация этой идеи представлена в Алгоритме 1.3.

Напомним, что EM-алгоритм заключается в итерационном повторении двух шагов. На E-шаге по формуле Байеса вычисляются скрытые переменные g_{iy}. Значение g_{iy} равно апостериорной вероятности того, что объект $x_i \in X^\ell$ принадлежит кластеру $y \in Y$. На M-шаге уточняются параметры каждого кластера (μ_y, Σ_y), при этом существенно используются скрытые переменные g_{iy}.

В Алгоритме 1.3 для простоты предполагается, что число кластеров известно заранее. Однако в большинстве практических случаев его лучше определять автоматически, как это было сделано в Алгоритме **??**.

Метод k-средних, представленный в Алгоритме 1.4, является упрощением EM-алгоритма. Главное отличие в том, что в EM-алгоритме каждый объект x_i распределяется по всем кластерам с вероятностями $g_{iy} = \mathsf{P}\{y_i = y\}$. В алгоритме *$k$-средних* ($k$-means) каждый объект жёстко приписывается только к одному кластеру.

Второе отличие в том, что в k-means форма кластеров не настраивается. Однако это отличие не столь принципиально. Можно предложить упрощённый вариант EM, в котором форма кластеров также не будет настраиваться — для этого достаточно взять сферические гауссианы с ковариационными матрицами $\Sigma_y = \sigma_y I_n$. С другой стороны, возможен и обобщённый вариант k-means, в котором будут определяться размеры кластеров вдоль координатных осей. Для этого в Алгоритме 1.3 достаточно убрать шаг 5 и заменить E-шаг жёстким приписыванием объектов кластерам:

$$y_i := \arg\min_{y \in Y} \rho_y(x_i, \mu_y), \quad j = 1, \ldots, n;$$
$$g_{iy} := [y_i = y], \quad j = 1, \ldots, n, \quad y \in Y.$$

Алгоритм 1.4. Кластеризация с помощью алгоритма k-средних

1: сформировать начальное приближение центров всех кластеров $y \in Y$:
μ_y — наиболее удалённые друг от друга объекты выборки;
2: **повторять**
3: отнести каждый объект к ближайшему центру (аналог E-шага):
$y_i := \arg\min_{y \in Y} \rho(x_i, \mu_y)$, $i = 1, \ldots, \ell$;
4: вычислить новое положение центров (аналог M-шага):
$\mu_{yj} := \dfrac{\sum_{i=1}^{\ell} [y_i = y] f_j(x_i)}{\sum_{i=1}^{\ell} [y_i = y]}$, $y \in Y$, $j = 1, \ldots, n$;
5: **пока** y_i не перестанут изменяться;

Таким образом, EM и k-means довольно плавно «перетекают» друг в друга, позволяя строить различные «промежуточные» варианты этих двух алгоритмов.

Заметим, что k-means похож также на поиск центра кластера в алгоритме FOREL. Отличие в том, что в FOREL кластер — это шар заданного радиуса R, тогда как в k-means объекты относятся к кластерам по принципу ближайшего соседа.

Существует два «канонических» варианта алгоритма k-means. Вариант Болла-Холла [5, стр. 110] представлен в Алгоритме 1.4. Вариант МакКина [5, стр. 98] отличается тем, что всякий раз, когда некоторый объект x_i переходит из одного кластера в другой, центры обоих кластеров пересчитываются. Для этого шаг 4 надо перенести внутрь цикла по i, выполняемого на шаге 3. МакКин показал в 1967 году, что этот вариант алгоритма приводит к локальному минимуму функционала Φ_0.

Алгоритм k-means крайне чувствителен к выбору начальных приближений центров. Случайная инициализация центров на шаге 1 может приводить к плохим кластеризациям. Для формирования начального приближения можно выделить k наиболее удалённых точек выборки: первые две точки выделяются по максимуму всех попарных расстояний; каждая следующая точка выбирается так, чтобы расстояние от неё до ближайшей уже выделенной было максимально.

Другая рекомендация — выполнить кластеризацию несколько раз, из различных случайных начальных приближений и выбрать кластеризацию с наилучшим значением заданного функционала качества.

Кластеризация может оказаться неадекватной и в том случае, если число кластеров будет изначально неверно угадано. Стандартная рекомендация — провести кластеризацию при различных значениях k и выбрать то, при котором достигается резкое улучшение качества кластеризации по заданному функционалу.

Кластеризация с частичным обучением. Алгоритмы EM и k-means легко приспособить для решения задач кластеризации *с частичным обучением* (semi-supervised learning), когда для некоторых объектов x_i известны правильные классификации $y^*(x_i)$. Обозначим через U подмножество таких объектов, $U \subset X^\ell$.

Примером такой задачи является рубрикация текстовых документов, в частности, страниц в Интернете. Типична ситуация, когда имеется относительно небольшое множество документов, вручную классифицированных по тематике. Требуется определить тематику большого числа неклассифицированных документов. Сходство документов $\rho(x, x')$ может оцениваться по-разному в зависимости от целей рубрикации

и специфики самих документов: по частоте встречаемости ключевых слов, по частоте посещаемости заданным множеством пользователей, по количеству взаимных гипертекстовых ссылок, или другими способами.

Модификация обоих алгоритмов довольно проста: на E-шаге (шаг 3) для всех $x_i \in U$ полагаем $g_{iy} := [y = y^*(x_i)]$, для всех остальных $x_i \in X^\ell \setminus U$ скрытые переменные g_{iy} вычисляются как прежде. На практике частичная классификация даже небольшого количества объектов существенно улучшает качество кластеризации.

1.1.4 Иерархическая кластеризация

Иерархические алгоритмы кластеризации, называемые также алгоритмами *таксономии*, строят не одно разбиение выборки на непересекающиеся классы, а систему вложенных разбиений. Результат таксономии обычно представляется в виде таксономического дерева — *дендрограммы*. Классическим примером такого дерева является иерархическая классификация животных и растений.

Среди алгоритмов иерархической кластеризации различаются два основных типа. *Дивизимные* или нисходящие алгоритмы разбивают выборку на всё более и более мелкие кластеры. Более распространены *агломеративные* или восходящие алгоритмы, в которых объекты объединяются во всё более и более крупные кластеры. Реализация этой идеи представлена в Алгоритме 1.5.

Сначала каждый объект считается отдельным кластером. Для одноэлементных кластеров естественным образом определяется функция расстояния

$$R(\{x\}, \{x'\}) = \rho(x, x').$$

Затем запускается процесс слияний. На каждой итерации вместо пары самых близких кластеров U и V образуется новый кластер $W = U \cup V$. Расстояние от нового кластера W до любого другого кластера S вычисляется по расстояниям $R(U, V)$, $R(U, S)$ и $R(V, S)$, которые к этому моменту уже должны быть известны:

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

где α_U, α_U, β, γ — числовые параметры. Эта универсальная формула обобщает практически все разумные способы определить расстояние между кластерами. Она была предложена Лансом и Уильямсом в 1967 году [8, 6].

На практике используются следующие способы вычисления расстояний $R(W, S)$ между кластерами W и S. Для каждого из них доказано соответствие формуле Ланса-Вильямса при определённых сочетаниях параметров [5]:

Расстояние ближнего соседа:
$R^{\text{б}}(W, S) = \min\limits_{w \in W, s \in S} \rho(w, s)$; $\quad \alpha_U = \alpha_V = \frac{1}{2},\ \beta = 0,\ \gamma = -\frac{1}{2}$.

Расстояние дальнего соседа:
$R^{\text{д}}(W, S) = \max\limits_{w \in W, s \in S} \rho(w, s)$; $\quad \alpha_U = \alpha_V = \frac{1}{2},\ \beta = 0,\ \gamma = \frac{1}{2}$.

Среднее расстояние:
$R^{\text{с}}(W, S) = \frac{1}{|W||S|} \sum\limits_{w \in W} \sum\limits_{s \in S} \rho(w, s)$; $\quad \alpha_U = \frac{|U|}{|W|},\ \alpha_V = \frac{|V|}{|W|},\ \beta = \gamma = 0$.

Расстояние между центрами:

Алгоритм 1.5. Агломеративная кластеризация Ланса-Уильямса

1: инициализировать множество кластеров C_1:
$t := 1; \quad C_t = \big\{\{x_1\}, \ldots, \{x_\ell\}\big\};$
2: **для всех** $t = 2, \ldots, \ell \quad$ (t — номер итерации):
3: найти в C_{t-1} два ближайших кластера:
$(U, V) := \arg\min\limits_{U \neq V} R(U, V);$
$R_t := R(U, V);$
4: изъять кластеры U и V, добавить слитый кластер $W = U \cup V$:
$C_t := C_{t-1} \cup \{W\} \setminus \{U, V\};$
5: **для всех** $S \in C_t$
6: вычислить расстояние $R(W, S)$ по формуле Ланса-Уильямса;

$$R^{\text{ц}}(W,S) = \rho^2\Big(\sum_{w\in W}\tfrac{w}{|W|}, \sum_{s\in S}\tfrac{s}{|S|}\Big); \qquad \alpha_U = \tfrac{|U|}{|W|},\ \alpha_V = \tfrac{|V|}{|W|},\ \beta = -\alpha_U\alpha_V,\ \gamma = 0.$$

Расстояние Уорда:

$$R^{\text{у}}(W,S) = \tfrac{|S||W|}{|S|+|W|}\rho^2\Big(\sum_{w\in W}\tfrac{w}{|W|}, \sum_{s\in S}\tfrac{s}{|S|}\Big); \quad \alpha_U = \tfrac{|S|+|U|}{|S|+|W|},\ \alpha_V = \tfrac{|S|+|V|}{|S|+|W|},\ \beta = \tfrac{-|S|}{|S|+|W|},\ \gamma = 0.$$

Возможных вариантов слишком много, и на первый взгляд все они кажутся достаточно разумными. Возникает вопрос: какой из них предпочесть? Рассмотрим несколько дополнительных свойств, характеризующих качество кластеризации.

Свойство монотонности. Обозначим через R_t расстояние между ближайшими кластерами, выбранными на t-м шаге для слияния. Говорят, что функция расстояния R обладает свойством *монотонности*, если при каждом слиянии расстояние между объединяемыми кластерами только увеличивается: $R_2 \leqslant R_3 \leqslant \ldots \leqslant R_\ell$.

Свойство монотонности позволяет изобразить процесс кластеризации в виде специального графика, называемого *дендрограммой*. По вертикальной оси откладываются объекты, по горизонтальной — расстояния R_t. Нетрудно доказать, что если кластеризация обладает свойством монотонности, то дендрограмму можно построить так, чтобы она не имела самопересечений. При этом любой кластер из множества C_t представляется сплошной последовательностью точек на вертикальной оси. Если же процесс кластеризации идёт не монотонно, то вместо дендрограммы получается запутанный клубок линий, на котором трудно что-либо разобрать.

Дендрограмма позволяет представить кластерную структуру в виде плоского графика независимо от того, какова размерность исходного пространства. Существуют и другие способы двумерной визуализации многомерных данных, такие как многомерное шкалирование или карты Кохонена, но они привносят в картину искусственные искажения, влияние которых довольно трудно оценить.

Оказывается, не любое сочетание коэффициентов в формуле Ланса-Вильямса приводит к монотонной кластеризации.

Теорема 1.1 (Миллиган, 1979). *Если выполняются следующие три условия, то кластеризация является монотонной:*

1) $\alpha_U \geqslant 0,\ \alpha_V \geqslant 0;$
2) $\alpha_U + \alpha_V + \beta \geqslant 1;$
3) $\min\{\alpha_U, \alpha_V\} + \gamma \geqslant 0.$

Из перечисленных выше расстояний только $R^{\text{ц}}$ не является монотонным. Расстояние Уорда отличается от него мультипликативной поправкой, которая и делает его монотонным.

Свойства растяжения и сжатия. Некоторые расстояния обладают *свойством растяжения*. По мере того, как кластер укрупняется, расстояния от него до других кластеров увеличиваются, как будто пространство вокруг кластера растягивается. На дендрограмме растягивающие расстояния характеризуются повышением плотности линий слева, в области наименьших значений R_t. Свойство растяжения считается желательным, так как оно способствует более чёткому отделению кластеров. С другой стороны, при слишком сильном растяжении возможно найти кластеры там, где их изначально не было. Растягивающими являются расстояния $R^{\text{д}}$ и $R^{\text{у}}$.

Некоторые расстояния, наоборот, обладают *свойством сжатия*. По мере роста кластера расстояния от него до других кластеров уменьшается, и кажется, что пространство вокруг кластера сжимается. Естественная кластеризация при этом может исчезнуть. Для сжимающих расстояний характерно уплотнение дендрограммы справа, в области наибольших значений R_t. Расстояние ближнего соседа $R^{\text{б}}$ является сильно сжимающим.

Свойства сжатия и растяжения определяются через отношение $R_t/\rho(\mu_U, \mu_V)$, где $R_t = R(U, V)$ — расстояние между ближайшими кластерами, объединяемыми на t-м шаге, μ_U и μ_V — центры этих кластеров. Если это отношение на каждом шаге больше единицы, то расстояние R является растягивающим; если оно всегда меньше единицы, то сжимающим. Есть и такие расстояния, которые не являются ни сжимающими, ни растягивающими, например, $R^{\text{с}}$ и $R^{\text{ц}}$. О них говорят, что они *сохраняют метрику пространства*.

На практике часто применяют *гибкое расстояние*, которое представляет собой компромисс между методами ближнего соседа, дальнего соседа и среднего расстояния. Оно определяется одним параметром β вместо четырёх:

$$\alpha_U = \alpha_V = (1 - \beta)/2, \quad \gamma = 0, \quad \beta < 1.$$

Гибкое расстояние является сжимающим при $\beta > 0$ и растягивающим при $\beta < 0$. Стандартная рекомендация: $\beta = -0{,}25$ [6].

Свойство редуктивности. Самой трудоёмкой операцией в Алгоритме 1.5 является поиск пары ближайших кластеров на шаге 3. Он требует $O(\ell^2)$ операций внутри основного цикла. Соответственно, построение всего таксономического дерева требует $O(\ell^3)$ операций. Это ограничивает применимость алгоритма выборками длины в несколько сотен объектов.

Идея ускорения алгоритма заключается в том, чтобы перебирать лишь наиболее близкие пары. Задаётся параметр δ, и перебор ограничивается сокращённым

множеством пар $\{(U,V)\colon R(U,V) \leqslant \delta\}$. Когда все такие пары будут исчерпаны, параметр δ увеличивается, и формируется новое сокращённое множество пар. И так далее, до полного слияния всех объектов в один кластер. Реализация представлена в Алгоритме 1.6.

Доказано, что этот алгоритм приводит к той же кластеризации, что и Алгоритм 1.5, если расстояние R обладает свойством редуктивности:

Опр. 1.1 (Брюинош, 1978). *Расстояние R называется редуктивным, если для любого $\delta > 0$ и любых δ-близких кластеров U и V объединение δ-окрестностей U и V содержит в себе δ-окрестность кластера $W = U \cup V$:*

$$\{S \mid R(U\cup V, S) < \delta,\ R(U,V) \leqslant \delta\} \subseteq \{S \mid R(S,U) < \delta \text{ или } R(S,V) < \delta\}.$$

Теорема 1.2 (Диде и Моро, 1984). *Если выполняются следующие три условия, то расстояние R является редуктивным:*

1) $\alpha_U \geqslant 0,\ \alpha_V \geqslant 0$;
2) $\alpha_U + \alpha_V + \min\{\beta, 0\} \geqslant 1$;
3) $\min\{\alpha_U, \alpha_V\} + \gamma \geqslant 0$.

Сравнение условий теорем 1.2 и 1.1, показывает, что всякое редуктивное расстояние является монотонным, следовательно, позволяет отобразить процесс кластеризации в виде дендрограммы. Из перечисленных выше расстояний только $R^{\text{ц}}$ не является редуктивным.

В Алгоритме 1.6 возможны различные эвристические стратегии выбора параметра δ на шагах 2 и 6. Общие соображения таковы: если δ настолько велико, что $P(\delta)$ содержит практически все пары кластеров, то мы фактически возвращаемся к неэффективному Алгоритму 1.5; если же δ мало́, то приходится слишком часто формировать множество $P(\delta)$. На практике поступают следующим образом. Если число кластеров в C_t не превышает порог n_1, то в качестве $P(\delta)$ берут множество всех возможных пар (U,V) из C_t. В противном случае выбирается случайным образом n_2 расстояний $R(U,V)$, и δ полагается равным наименьшему из них. В случае редуктивности параметры алгоритма n_1 и n_2 влияют только на время выполнения алгоритма, но не на результат кластеризации. Оптимальные значения для них подбираются с помощью калибровочных тестов и, вообще говоря, зависят от компьютера. В качестве начального выбора можно предложить $n_1 = n_2 = 20$.

Определение числа кластеров проще всего производить путём отсечения правого участка дендрограммы. На горизонтальной оси находится интервал максимальной длины $|R_{t+1} - R_t|$, и в качестве результирующей кластеризации выдаётся множество кластеров C_t. Число кластеров равно $K = \ell - t + 1$. При необходимости можно задать ограничение на минимальное и максимальное число кластеров $K_0 \leqslant K \leqslant K_1$ и выбирать t, удовлетворяющие ограничениям $\ell - K_1 + 1 \leqslant t \leqslant \ell - K_0 + 1$.

Во многих прикладных задачах интерес представляет таксономическое дерево целиком, и определять оптимальное число кластеров не имеет особого смысла.

Алгоритм 1.6. Быстрая агломеративная кластеризация на основе редуктивности

1: инициализировать множество кластеров C_1:
$t := 1$; $C_t = \{\{x_1\}, \ldots, \{x_\ell\}\}$;
2: выбрать начальное значение параметра δ;
3: $P(\delta) := \{(U, V) \mid U, V \in C_t,\ R(U, V) \leqslant \delta\}$;
4: **для всех** $t = 2, \ldots, \ell$ (t — номер итерации):
5: **если** $P(\delta) = \varnothing$ **то**
6: увеличить δ так, чтобы $P(\delta) \neq \varnothing$;
7: найти в $P(\delta)$ пару ближайших кластеров:
$(U, V) := \arg\min\limits_{(U,V) \in P(\delta)} R(U, V)$;
$R_t := R(U, V)$;
8: изъять кластеры U и V, добавить слитый кластер $W = U \cup V$:
$C_t := C_{t-1} \cup \{W\} \setminus \{U, V\}$;
9: **для всех** $S \in C_t$
10: вычислить расстояние $R(W, S)$ по формуле Ланса-Уильямса;
11: **если** $R(W, S) \leqslant \delta$ **то**
12: $P(\delta) := P(\delta) \cup \{(W, S)\}$;

Достоинства и недостатки агломеративной кластеризации. Точного ответа на вопрос, какой алгоритм кластеризации лучше, не существует. Каждое из расстояний, перечисленных выше, имеет свои недостатки и подходит не для всех задач.

Метод ближнего соседа обладает *цепочечным эффектом*, когда независимо от формы кластера к нему присоединяются ближайшие к границе объекты. В некоторых случаях это приводит к тому, что кластеры «отращивают щупальца». В зависимости от задачи это свойство может быть как полезным, так и мешающим. Метод ближнего соседа хорошо подходит для выделения кластеров ленточной формы.

Метод дальнего соседа цепочечного эффекта не имеет, но на раннем этапе может объединять довольно несхожие группы.

Расстояние между центрами масс не монотонно и не редуктивно, поэтому редко используется на практике, хотя интуитивно кажется «золотой серединой».

Метод Уорда оказался наилучшим по результатам экспериментального сравнения на представительном наборе модельных задач [5]. Он чаще других методов восстанавливает интуитивно наилучшую кластеризацию.

§1.2 Многомерное шкалирование

Визуализация кластерной структуры заданной выборки объектов X^ℓ — непростая проблема, особенно если выборка содержит тысячи объектов, а пространство объектов существенно многомерно.

Задача *многомерного шкалирования* (multidimensional scaling, MDS) заключается в следующем. Имеется обучающая выборка объектов $X^\ell = \{x_1, \ldots, x_\ell\} \subset X$. Заданы расстояния $R_{ij} = \rho(x_i, x_j)$ для некоторых пар обучающих объектов $(i, j) \in$ $\in D$. Требуется для каждого объекта $x_i \in X^\ell$ построить его признаковое описание — вектор $x_i = (x_i^1, \ldots, x_i^n)$ в евклидовом пространстве $\mathbb{R}^n$ так, чтобы евклидовы рассто-

яния d_{ij} между объектами x_i и x_j

$$d_{ij}^2 = \sum_{d=1}^{n} (x_i^d - x_j^d)^2$$

как можно точнее приближали исходные расстояния R_{ij} для всех $(i,j) \in D$. Данное требование можно формализовать по-разному; один из наиболее распространённых способов — через минимизацию функционала, называемого *стрессом*:

$$S(X^\ell) = \sum_{(i,j)\in D} w_{ij}(d_{ij} - R_{ij})^2 \to \min,$$

где минимум берётся по совокупности ℓn переменных $(x_i^d)_{i=1,\ell}^{d=1,n}$.

Размерность n обычно задаётся небольшая. В частности, при $n = 2$ многомерное шкалирование позволяет отобразить выборку в виде множества точек на плоскости (scatter plot). Плоское представление, как правило, искажено ($S > 0$), но в целом отражает основные структурные особенности многомерной выборки, в частности, её кластерную структуру. Поэтому двумерное шкалирование часто используют как инструмент предварительного анализа и понимания данных.

На практике расстояния чаще бывают известны для всех пар объектов, но мы будем рассматривать более общий случай, когда D — произвольное заданное множество пар индексов объектов.

Веса w_{ij} задаются исходя из целей шкалирования. Обычно берут $w_{ij} = (R_{ij})^\gamma$. При $\gamma < 0$ приоритет отдаётся более точному приближению малых расстояний; при $\gamma > 0$ — больших расстояний. Наиболее адекватным считается значение $\gamma = -2$, когда функционал стресса приобретает физический смысл потенциальной энергии системы ℓ материальных точек, связанных упругими связями; требуется найти равновесное состояние системы, в котором потенциальная энергия минимальна.

Функционал стресса $S(X^\ell)$ сложным образом зависит от ℓn переменных, имеет огромное количество локальных минимумов, и его вычисление довольно трудоёмко. Поэтому многие алгоритмы многомерного шкалирования основаны на итерационном размещении объектов по одному. На каждой итерации оптимизируются евклидовы координаты только одного из объектов, а координаты остальных объектов, вычисленные на предыдущих итерациях, полагаются фиксированными.

Размещение одного объекта методом Ньютона–Рафсона. Рассмотрим аддитивную составляющую функционала стресса, зависящую только от одного объекта $x \in X^\ell$ с координатами $x \equiv (x^1, \ldots, x^n)$, которые и требуется найти:

$$S(x) = \sum_{x_i \in U} w_i \big(d_i(x) - R_i\big)^2 \to \min_x,$$

где $U \subseteq X^\ell$ — все объекты с известными расстояниями $R_i = \rho(x, x_i)$; $d_i(x)$ — евклидово расстояние между векторами $x \equiv (x^1, \ldots, x^n)$ и $x_i \equiv (x_i^1, \ldots, x_i^n)$.

Рассмотрим применение метода Ньютона–Рафсона для минимизации $S(x)$.

Поскольку функционал многоэкстремальный, очень важно выбрать удачное начальное приближение $x^{(0)}$. Вектор x должен быть похож на те векторы $x_i \in U$, для

которых расстояния R_i малы. Неплохая эвристика — взять в качестве $x^{(0)}$ взвешенное среднее всех векторов из U, например, с весами $c_i = R_i^{-2}$:

$$x^{(0)} = \frac{\sum_{x_i \in U} c_i x_i}{\sum_{x_i \in U} c_i}.$$

Затем запускается итерационный процесс

$$x^{(t+1)} := x^{(t)} - h_t \big(S''(x^{(t)})\big)^{-1} S'(x^{(t)}),$$

где $S'(x^{(t)})$ — вектор первых производных (градиент) функционала $S(x)$ в точке $x^{(t)}$, $S''(x^{(t)})$ — матрица вторых производных (гессиан) функционала $S(x)$ в точке $x^{(t)}$, h_t — величина шага, который можно положить равным 1, но более тщательный его подбор способен увеличить скорость сходимости.

Найдём выражения для градиента и гессиана.

$$\frac{\partial d_i}{\partial x^a} = \frac{x^a - x_i^a}{d_i};$$

$$\frac{\partial S}{\partial x^a} = 2 \sum_{x_i \in U} w_i \left(1 - \frac{R_i}{d_i}\right)(x^a - x_i^a);$$

$$\frac{\partial^2 S}{\partial x^a \partial x^a} = 2 \sum_{x_i \in U} w_i \left(\frac{R_i}{d_i}\left(\frac{x^a - x_i^a}{d_i}\right)^2 - \frac{R_i}{d_i} + 1\right);$$

$$\frac{\partial^2 S}{\partial x^a \partial x^b} = 2 \sum_{x_i \in U} w_i \frac{R_i}{d_i}\left(\frac{x^a - x_i^a}{d_i}\right)\left(\frac{x^b - x_i^b}{d_i}\right).$$

Заметим, что в пространствах малой размерности $n \leqslant 3$ обращение гессиана — довольно простая операция, однако с ростом размерности вычислительные затраты растут как $O(n^3)$.

Итерации Ньютона–Рафсона прекращаются, когда значение стресса $S(x)$ стабилизируется или вектор x перестаёт существенно изменяться, то есть стабилизируется норма разности $\|x^{(t+1)} - x^{(t)}\|$.

Будем полагать, что размещение объекта x относительно множества уже размещённых объектов $U \subseteq X^\ell$ реализуется процедурой `НьютонРафсон`(x, U).

Субквадратичный алгоритм многомерного шкалирования. Алгоритм 1.7 начинает с того, что находит две самые удалённые точки выборки x_i, x_j. Достаточно решить эту задачу приближённо. В частности, можно взять произвольную точку, найти самую удалённую от неё, затем для этой точки найти самую удалённую, и так далее. Обычно 3–4 итераций хватает, чтобы найти пару достаточно далёких точек. Найденным точкам приписываются (в двумерном случае) евклидовы координаты $(0, 0)$ и $(0, R_{ij})$. Затем находится третья точка x_k, наиболее удалённая от первых двух, то есть для которой значение $\min\{R_{ki}, R_{kj}\}$ максимально. Евклидовы координаты x_k определяются (в двумерном случае) исходя из того, что треугольник $\triangle ijk$ жёстко задан длинами своих сторон.

Затем начинается поочерёдное добавление точек и их размещение относительно уже имеющихся. После размещения первых K точек их положение уточняется друг

Алгоритм 1.7. Субквадратичный алгоритм многомерного шкалирования

Вход:
R_{ij} — матрица попарных расстояний между объектами, возможно, разреженная;
K — размер скелета;

Выход:
евклидовы координаты всех объектов выборки $x_i \equiv (x_i^1, \ldots, x_i^n)$, $i = 1, \ldots, \ell$;

Инициализировать скелет:
  $U :=$ три достаточно далёкие друг от друга точки;
пока $|U| < K$ — наращивать скелет:
  $x := \arg\max\limits_{x_i \in X^\ell \setminus U} \big(\min\limits_{x_j \in U} R_{ij}\big)$;
  НьютонРафсон(x, U);
  $U := U \cup \{x\}$;
пока координаты точек скелета не стабилизируются:
  найти наиболее напряжённую точку в скелете:
    $x := \arg\max\limits_{x_i \in U} S(x)$;
  НьютонРафсон$(x, U \setminus \{x\})$;
для $x \in X^\ell \setminus U$
  НьютонРафсон(x, U);

относительно друга. Эти точки, размещённые с особой тщательностью, становятся «скелетом», относительно которого размещаются все остальные точки. Расстояния между «не-скелетными» точками в алгоритме вообще не задействуются.

Если $K = \ell$, то все точки будут «скелетными»; в этом случае размещение является наиболее точным, но и наиболее долгим, требуя $O(\ell^2)$ операций. В общем случае число операций алгоритма $O(K^2) + O(K\ell)$. Выбирая размер «скелета» K, можно находить компромисс между точностью и временем построения решения.

Карта сходства отображает результат многомерного шкалирования при $n = 2$ в виде плоского точечного графика. Евклидова метрика и функционал стресса инвариантны относительно произвольных ортогональных преобразований карты сходства — сдвигов, поворотов и зеркальных отражений. Поэтому оси на карте, вообще говоря, не имеют интерпретации.

Для интерпретации карты сходства на ней обозначают *ориентиры* — те объекты выборки, интерпретации которых хорошо известны. Если искажения расстояний на карте не велики, то объекты, близкие к ориентирам, должны иметь схожие интерпретации.

Карта сходства даёт лишь общее представление о взаимном расположении объектов выборки. Делать на её основе какие-либо количественные выводы, как правило, нельзя.

Диаграмма Шепарда позволяет сказать, насколько сильно искажены расстояния на карте сходства. Это точечный график; по горизонтальной оси откладываются исходные расстояния R_{ij}; по вертикальной оси откладываются евклидовы расстояния d_{ij}; каждая точка на графике соответствует некоторой паре объектов $(i, j) \in D$.

Если число пар превышает несколько тысяч, отображается случайное подмножество пар. Иногда на диаграмме изображается сглаженная зависимость $d_{ij}(R_{ij})$, а также сглаженные границы верхних и нижних доверительных интервалов, в которых $d_{ij}(R)$ находится с высокой вероятностью (например, 90%) при каждом значении R.

Идеальной диаграммой Шепарда является наклонная прямая — биссектриса первой четверти. Чем «толще» облако точек, представленное на диаграмме, тем сильнее искажения, и тем меньшего доверия заслуживает карта сходства.

Пример 1.1. Один из надёжных способов тестирования методов многомерного шкалирования, предложенный в [3] — размещение изначально двумерных (или близких к двумерным) выборок. Например, можно взять множество городов Российской Федерации, и задать R_{ij} как евклидовы расстояния между их географическими координатами (долгота, широта). Хороший алгоритм шкалирования должен построить карту сходства, похожую на географическую карту (с тосностью до поворотов и отражений), а диаграмма Шепарда должна оказаться практически диагональной.

Резюме

1. *Задача кластеризации* отличается от классификации тем, что принадлежность обучающих объектов каким-либо классам изначально не задаётся. Требуется классифицировать объекты только на основе их сходства друг с другом. Исходными данными служит матрица попарных расстояний между объектами. Решение задачи кластеризации принципиально неоднозначно. Различные критерии качества и эвристические алгоритмы приводят к различным кластеризациям. Не существует универсального алгоритма кластеризации; каждая эвристика хорошо работает только в определённом классе задач.

2. *Цели кластеризации* могут быть разными, в зависимости от прикладной задачи: выявление структуры выборки, построение таксономического дерева, сокращение объёма хранимых данных путём замены объектов центрами кластеров, выделение нетипичных объектов, визуальный разведочный анализ данных.

3. *Графовые алгоритмы* (связные компоненты, кратчайший незамкнутый путь) очень наглядны, но медленно работают и, как правило, чувствительны к шумовым выбросам и разреженному фону, образованному нетипичными объектами.

4. *Алгоритм FOREL* столь же нагляден, но более устойчив. Кроме того, он строит удобную для интерпретации двухуровневую кластерную структуру. Имеет большое число успешных практических применений.

5. *Функционалы качества кластеризации* часто применяются на практике в роли дополнительных критериев для выбора лучшей кластеризации из нескольких альтернатив. Некоторые алгоритмы кластеризации эквивалентны поиску оптимума того или иного функционала.

6. *EM-алгоритм* является мощным статистическим инструментом кластеризации. Он достаточно эффективен, позволяет определять оптимальное число кластеров, выделять шумовые выбросы и разреженный фон. Недостатком является чувствительность к начальному приближению.

7. *Алгоритм k-средних* является упрощённым вариантом EM-алгоритма. Он сходится ещё быстрее, но ещё более чувствителен к выбору начального приближения, и чаще приводит к плохим кластеризациям. Сходство алгоритма k-средних и EM-алгоритма позволяет конструировать различные «промежуточные варианты». Благодаря своей простоте и наглядности, алгоритм k-средних крайне популярен и встроен во многие пакеты анализа данных.

8. *Алгоритм иерархической агломеративной кластеризации*, в отличие от графовых и статистических алгоритмов, выявляет детальную кластерную структуру множества объектов в виде таксономического дерева или *дендрограммы*. Для многих приложений эта информация крайне полезна. Формула Ланса-Вильямса позволяет пересчитывать межкластерные расстояния после объединения произвольной пары кластеров в один. Формула имеет четыре свободных параметра, причём хорошо известно, какие значения этих параметров гарантируют желательные свойства *монотонности*, *растяжимости* и *редуктивности*. Наиболее предпочтительным вариантом является *формула Уорда*.

9. *Задача многомерного шкалирования* заключается в том, чтобы по матрице попарных расстояний между объектами найти наиболее подходящие признаковые описания объектов, как правило, низкой размерности. В частности, размерность 2 позволяет изобразить изначально многомерную выборку в виде двумерного точечного графика — *карты сходства*.

Список литературы

[1] *Загоруйко Н. Г.* Прикладные методы анализа данных и знаний. — Новосибирск: ИМ СО РАН, 1999.

[2] *Загоруйко Н. Г., Ёлкина В. Н., Лбов Г. С.* Алгоритмы обнаружения эмпирических закономерностей. — Новосибирск: Наука, 1985.

[3] *Кулаичев А. П.* Методы и средства комплексного анализа данных. — М: ИНФРА-М, 2006.

[4] *Лагутин М. Б.* Наглядная математическая статистика. — М.: П-центр, 2003.

[5] *Мандель И. Д.* Кластерный анализ. — М.: Финансы и Статистика, 1988.

[6] *Уиллиамс У. Т., Ланс Д. Н.* Методы иерархической классификации // Статистические методы для ЭВМ / Под ред. М. Б. Малютов. — М.: Наука, 1986. — С. 269–301.

[7] *Jain A., Murty M., Flynn P.* Data clustering: A review // *ACM Computing Surveys*. — 1999. — Vol. 31, no. 3. — Pp. 264–323.
http:// citeseer.ifi.unizh.ch/jain99data.html.

[8] *Lance G. N., Willams W. T.* A general theory of classification sorting strategies. 1. hierarchical systems // *Comp. J.* — 1967. — no. 9. — Pp. 373–380.