Алгоритмы кластеризации

Гиршов Евгений*

1 Введение

В данной статье собран материал из 4-х последних статей по кластеризации, в каждой из которых описываются проблемы кластеризации с разных точек зрения и с помощью разных подходов. В статьях анализируется достоинства и недостатки алгоритмов, и показано, какой алгоритм пользователь должен выбрать при использовании конкретных приложений с кластеризацией.

Кластеризация может быть неформально определена как: процесс организации объектов в группы по схожим признакам. Есть 2 основных метода кластеризации: *декомпозиция* (разделение, k-кластеризация) - в данном случае каждый объект связан только с одной группой, *иерархическая кластеризация* - в этом случае каждая группа большего размера состоит из групп меньшего размера. Оба метода активно исследовались в середине 70-х годов и сравнительно немного в 80-х. В последние годы, в связи с появлением WWW поисковых систем (и особенно проблемы организации огромных кол-ва информации) и концепцией 'информационная проходка' (способ анализа информации в базе данных с целью отыскания аномалий и трендов без выяснения смыслового значения записей) снова возник интерес к алгоритмам кластеризации.

Огромные объёмы информации в сети Internet зачастую приводят к тому, что количество объектов, выдаваемых по запросу пользователя, очень велико. Это затрудняет процесс обзора результатов и выбора наиболее подходящих материалов (статей, обзоров, отчетов, др.) из множества найденных. Однако, в большинстве случаев огромные объемы информации можно сделать доступными для восприятия, если уметь разбивать источники информации (например, WEB-страницы) на тематические группы. Тогда, пользователь сразу может отбрасывать множества документов из мало-релевантных групп. Такой процесс группи-

*Законспектировал лекцию Шаров Михаил

ровки данных (в нашем случае текстовых) осуществляется с помощью кластеризации или классификации корпуса текстов.

ОПИСАНИЕ МЕТОДОВ

Прежде чем описывать алгоритмы важно различать следующие свойства, присущие алгоритмам. Во-первых, необходимо чётко понять разницу между классификацией и кластеризацией документов. *Классификация* это отнесение каждого документа в определённый класс с заранее известными параметрами, полученными на этапе обучения. Число классов строго ограничено. *Кластеризация* - разбиение множества документов на кластеры - подмножества, параметры которых заранее неизвестны. Количество кластеров может быть произвольным или фиксированным. Во-вторых, все методы можно разделить на числовые и нечисловые. Числовые - это те методы, которые используют числовые характеристики о документах. Нечисловые - методы, которые для работы используют непосредственно слова и фразы, составляющие текст.

Цель данной статьи - оценить предыдущие работы по кластеризации. В частности, ударение сделано на алгоритмы кластеризации, чей смысл выявить группы, которым могут принадлежать объекты из входного множества. Это противопоставление известной проблеме 'подобию кластеризации', к которым относятся разбиение графа или проблемы сегментации. Также эта статья делает акцент на фундаментальные идеи, не следя за построением кластеров и пытается избежать конкретных деталей приложений кластеризации таких как: выявление хороших единиц измерения для подобных текстовых документов.

2 k - кластеризация

K-means.Относится к не - иерархическим алгоритмам. Кластеры представлены в виде центройдов, являющихся 'центром массы' всех документов, входящих в кластер.

В основном, алгоритмы k - кластеризации берут на вход множество S и число k. И на выход отдают разделение множества S на подмножества $S1, S2, ..., Sk$. Наиболее общий k - алгоритм это *алгоритм оптимизации (оптимизационный алгоритм)*. Оптимизационные алгоритмы обычно предполагают, что элементы S принадлежат множеству R^d, и определяется оценочная функция $c : \{X : X \leq S\} \rightarrow R^+$. Функция - стоимость кластера. Смысл алгоритма заключается в том, чтобы минимизировать сумму стоимостей кластера, т.е. $min \sum_i c(Si)$, где $i = 1...k$.

- Наиболее хорошо известный критерий оптимизации - критерий суммы квадратов. Допустим $x(i,r)$ - является r-ым элементом S_i. $|S_i|$ - кол-во элементов S_i, и $d(x(i,r),x(i,s))$ - расстояние между $x(i,r)$ и $x(i,s)$. Тогда функция стоимости критерия суммы квадратов вычисляется следующим образом: $c(Si) = \sum_r \sum_s d(x(i,r),x(i,s))$, где $r = 1..|S_i|, s = 1..|S_i|$. Это NP - сложная задача.

- Однако, рассмотрим усовершенствованный алгоритм МакКвина. Этот алгоритм основан на возможности вычислить центройд каждого кластера S_i. (x^i - центройд).

$$c(S_i) = \sum_r d(x^i, x(i,r)), \text{где } r = 1...|S_i|$$

  Этот алгоритм приведёт точно к тем же результатам, что и предыдущий. В основе K-means лежит итеративный процесс стабилизирования центроидов кластеров. Основной характеристикой кластера является его центроид и вся работа алгоритма направлена на стабилизирование или, в лучшем случае, полное прекращение изменения центроида кластера. Теоретическая скорость работы алгоритма $O(n)$.

- В связи с тем, что много типов данных не принадлежат пространствам, в которых определены их значения, был разработан ещё один подобный алгоритм 'k - medioids'. (Kaufman, Rousseeuw) Алгоритм назван PAM (Partitioning Around Medioids) основан на возможности найти медиану каждого S_i. (x^i - медиана).

$$c(S_i) = \sum_r d(x^i, x(i,r)), \text{где } r = 1...|S_i|$$

- Альтернативный оптимизационный критерий был предложен Gonzalez. Вместо минимизации $\sum_i c(S_i)$ было предложено $\min\max_{(1\le i\le k)} c(S_i)$, где $c(S_i) = \max d(x(i,r),x(i,s))d(x(i,r),x(i,s)) \le S_i$.

3 Иерархическая кластеризация

Single Link, Complete Link, Group Average

Одни из старейших алгоритмов кластеризации данных. Относятся к методам кластерного анализа. Особенностью этих методов, является то, что они разбивают документы на кластеры путем разбиения их на

иерархичные группы, т.е. получаемое множество кластеров имеет иерархичную структуру. Они называются еще методами иерархической агломеративной кластеризации. Принцип работы иерархических агломеративных процедур состоит в последовательном объединении групп элементов, сначала самых близких, а затем всё более отдалённых друг от друга [20].

Основная суть этих методов заключается в выполнении следующих шагов ([13]):

- вычисление значений близости между элементами и получении матрицы близости;
- определение каждого элемента в свой отдельный кластер;
- слияние в один кластер наиболее близких пар элементов;
- обновление матрицы близости путем удаления колонок и строк для кластеров, которые были слиты с другими и дальнейшего пересчета матрицы;
- переход на шаг 3 до тех пор, пока не сработает остановочный критерий.

Эти три метода отличаются между собой в 4-м шаге. И именно благодаря способам обновления матрицы близости разные алгоритмы имеют разную точность. Проверка точности алгоритмов была проведена на специальных тестовых наборах и показала, что алгоритм Single Link имеет наименьшую точность, а остальные два - примерно одинаковую, более высокую, чем у Single Link. В качестве остановочного критерия выбирается максимальное количество документов в кластере.

Скорость работы алгоритмов Single Link и Group Average - $O(n^2)$, а в Complete Link - $O(n^3)$, где n - количество элементов. Количество занимаемой памяти алгоритмом Single Link - $O(n)$. Достоинства методов:

- алгоритмы не нуждаются в обучении;
- использование матрицы близости между элементами;
- алгоритмы инкрементены.

Недостатки методов:

- необходимо задавать порог - максимальное количество элементов в кластере;

- для получения хороших результатов кластеризации значения близости между парами элементов должны приходить в определённом порядке, т.е. работа алгоритма не детерминированная;

- кластеры не пересекаются.

Метод	Функция стоимости
Single - link	$mind(x_i, x_j), x_{i,j} \leq S_{i,j}$
Average - link	$\frac{1}{(\lvert S_i \rvert \lvert S_j \rvert)} * \sum_{x_i} \sum_{x_j} d(x_i, x_j)$
Complete - link	$maxd(x_i, x_j), x_{i,j} \leq S_{i,j}$

4 Banfield and Raftery: Mixture Models

Этот раздел описывает статью 'Модели основанные на гауссовой и негауссовой кластеризации' Banfield и Raftery. Их подход основан на *статической неоднородной (смешанной) модели* для кластеризации. Идея такой модели в том, что входное множество векторов $x_1, x_2, ..., x_n$ - наблюдения из множества из к неизвестных распределений $E_1, E_2, ..., E_k$ Допустим, плотность распределения x_r связанного с E_i задана с помощью функции $f_i(x_r, \theta)$ для некоторого неизвестного множества параметров θ. Также, предположим что для каждого x_r и распределения $E_i t(r, i)$ - вероятность, что x_r принадлежит E_i. $\sum t(r, i) = 1$, где $i = 1...k$. Согласно определению, цель данной схемы - найти параметры θ и t, которые увеличат вероятность

$$L(\theta, t) = \prod_{r=1}^{n} \sum_{i=1}^{k} t(r, i) f_i(x_r|\theta).$$

Рассматриваем упрощенную модель, где каждый элемент точно принадлежит одному распределению. Таким образом $t(r, i)$ заменяется на $\gamma = (\gamma_1, \gamma_2, ..., \gamma_n)$, т.к. $\gamma_r = i$, если x_r принадлежит E_i.

$$L(\theta, \gamma) = \prod_{r=1}^{n} f(\gamma_r)(xr|\theta).$$

Предыдущие работы по смешанным моделям моделировали распределение как многомерные нормальные функции. Таким образом, неизвестные параметры θ - собственный вектор и матрица ковариации каждого распределения. Допустим μ_i и Σ_i означают собственный вектор и матрицу ковариаций соответствующую E_i. Эта структура очень важна: к примеру, пусть $\Sigma_i = \sigma^2 I$ для всех i - тогда максимум $L(\theta, \gamma)$ - эквивалентен оптимизации критерия суммы квадратов. С помощью такой

параметризации, стандартный итеративный процесс EM (expectation - maximization) может быть использован для поиска μ_i, Σ_i и γ. Основная проблема - если μ_i и Σ_i сильно различаются для каждого распределения, то нахождения общего оптимума для $L(\theta, \gamma)$ займёт много времени и памяти. К тому же, мы накладывали ограничение - $\Sigma_i = \sigma^2 I$. Чтобы достичь большей гибкости в характеристиках каждого распределения применили репараметризацию распределений. Основная идея -декомпозиция

$$\Sigma_i = D_i \Lambda_i D_i^T$$

где D_i - матрица собственных векторов, а $\Lambda_i = diag(\lambda_1, \lambda_2, ..., \lambda_n)$ - собственные числа. Если матрица ковариаций положительно определённая - тогда такая декомпозиция возможна. Матрица Λ_i - в свою очередь может быть представлена в виде $\lambda_i A_i$, где λ_i главные собственные числа.

Было предложено 2 модели:

- $A_i = I$ для всех A_i. Тогда разложения сферические, но разного размера.
- Сделать все A_i одинаковыми, но допустить λ_i и D_i - разные для каждого распределения. (т.е. одинаковой формы, но разного размера и ориентации).

5 Динамические системы

В данной модели подразумевается, что данные были разбиты по категориям, т.е. каждый элемент имеет набор атрибутов, взятых из каких-то наборов. Входные данные - множество $T = \{\tau_1, ..., \tau_n\}$ - кортежи, где τ_i - набор d полей, каждый из которых может принимать значение из небольшого набора. Удобно представлять входные данные как граф, в котором вершины - все значения фигурирующие во входных картежах. Таким образом получаем множество узлов $V = \{v_1, v_2, ..., v_m\}$. Каждый картеж может быть представлен как путь через граф, и множество граней - коллекция граней в пути. С каждым узлом v ассоциируем вес w_v и вектор w, содержащий веса всех узлов - рассматриваемый как *конфигурация* графа. Внутри данного подхода - функция f, которая отображает данную конфигурацию в следующую. Идея в том, чтобы повторять применять f пока фиксированная точка не будет достигнута, т.е. $f(w)$ w.

- **1. For each node v update wv as follows: For each tuple** $\tau = \{v, u_1, \ldots, u_{d-1}\}$

Containing v do

$$x_\tau < -(w_{u_1}, ..., w_{u_{d-1}}), \quad w_v < -\sum_\tau x_\tau$$

- **2. Normalize the configuration such that the sum of the squares of the weights in each field is 1.**

The product operator $\prod : \oplus(\omega_1, ..., \omega_d) = \omega_1 * ... * \omega_d$.
The S_p operator: $\oplus(\omega_1, ..., \omega_d) = ((\omega_1)^2 + ... + (\omega_d)^p)^{1/p}$.
The S_∞ operator:$\oplus(\omega_1, ..., \omega_d) = max\{|\omega_1|, ..., |\omega_d|\}$.

Дополнительно, к выбору оператора объединения пользователь должен выбрать начальную конфигурацию. Предлагается 2 метода. Можно рассматривать данные без ввода центра притяжения, т.е. использовать случайный или одинаковый вес узлов для начальной конфигурации. С другой стороны, можно взять множество определённых значений для значений весов.

Результат 1: Для каждого множества картежей T и каждой начальной конфигурации w, $f^i(w)$ сходится в фиксированную точку при $i \to \infty$ когда S_i берётся как правило сложения (объединения).

Результат 2: Если T_a и T_b - 2 множества картежей без значений и входное множество - объединение T_a и T_b, тогда шанс что итеративный процесс сойдётся к весам,, которые удовлетворяют значениям в T_a и этим же в T_b возрастает, если множества неравны по размерам.

Результат 3: Если динамические системы имеют объединенную (сложную) входное множество - они могут быть определены с помощью начала итерации с относительно малым множеством начальных конфигураций.

Такие результаты подсказывают методологию для использования этого типа динамических систем для исследования данных. Можно запускать итеративный процесс к множеству картежей пока они не сойдутся (Результат 1). Если у нас уесть узлы с легким весом и большим весом, то разделяя картежи (содержащие узлы) можно найти разделение входного множества, которое имеет семантическое значение (Результат 2). Запуская разные начальные конфигурации - получаем разные группирования данных (Результат 3).

Есть также ещё один способ основанный на спектральном делении графа.

- Update $w^{(i)} \leftarrow f(w^{(i)})$ for $i = 1, 2, ..., m$.
- Update the set of vectors $\{w^{(1)}, ..., w^{(m)}\}$ to be orthonormal.

6 Clique Graphs

Основная идея данной модели в том, что в идеале все элементы в рамках кластера должны быть подобны всем элементам другого кластера. Графы также должны быть подобны. Было представлено 2 алгоритма - теоретический, практический. Для начала несколько определений, чтобы описать теоретический алгоритм. Представим, что V показывает множество вершин во входном графе, и V - объединение независимых множеств так, то $V = \bigcup E_k$, где каждый $_i$ - представляет кластер. (v) - метка кластера, содержащего v. $C(v) = i$, если и только если v из $_i$. Центральная часть - подмножество $V^{'} = E_1^{'} \bigcup ... \bigcup E_k^{'}$.

$$C_{V'}(v) = \max_{1 \leq i \leq k} deg(v, E_i^{'})/|E_i^{'}|.$$

где $v \epsilon V - V^{'}$.

Результат1: Пусть m определяет размер самого маленького кластера на входе. Если a не слишком большое и m не слишком мало, тогда с большой вероятностью случайное маленькое подмножество V содержит центральную часть, которая корректно классифицирует все входные значения. Практический алгоритм называется $CAST$, работает немного подругому. Он требует не предшествующего знания количества кластеров, и избегает грубого установления количества центральных кандидатов. Как входной параметр он берет функцию подобия s: $V \times V \rightarrow [0, 1]$ и коэффициент t и определяет сходство элемента v кластеру E:

$$a(v, E) = \sum_{u \epsilon E} s(u, v)$$

Говорят, что элемент t - сильно схож с , если $a(v, E) \geq t|E|$. Смысл алгоритма - добавлять сильно похожие элементы в кластер , удаляя слабо похожие. Когда процесс стабилизируется - кластер считается завершённым, и стартует новый кластер. Когда все элементы отнесены к кластерам - процесс останавливается. Были представлены результаты работы алгоритма на модельных и на реально-биологических данных. Результат на моделированных данных поразителен в термах точности. Но не представлены результаты времени обработки и требования алгоритма к памяти.

7 Subspace clustering

Данный алгоритм относится к Clique алгоритмам, применяется для работы с большими базами данных. Выделяются три цели для данных используемых на практике.

- Эффективная обработка высокой размерности: Каждый элемент БД может иметь большое количество атрибутов, каждый из которых может иметь отношение к формированию кластера.
- Интерпретируемость результатов: Многие приложения требуют, чтобы алгоритм поддерживал простое резюмирование каждого кластера.
- Масштабируемость и практичность: алгоритмы кластеризации должны быть быстрыми и легко используемыми в больших базах данных, нечувствительными к шумам и порядку, в котором читаются данные.

Clique системы берут три действия приближения к кластеризации. Первое, выбирается множество подпространств, в котором надо кластеризовать данные. Затем кластеризация выполняется независимо в каждом подпространстве. В конце, генерируется компактное резюме по каждому кластеру в виде выражения в дизъюнктивной нормальной форме форме (ДНФ). Предположим, что входные данные - множество n d-размерных числовых? векторов из пространства $A = A_1 * A_2 * ... * A_d$, где каждый $_i$ - интервал. Кроме того, предположим, что каждый $_i$ разделён на ξ подинтервалов, где ξ - параметр определённый пользователем. Применяя такое деление в d- пространстве получим, что пространство разделится на множество d-размерных координатно-расположенных квадратов, которые называют *регионами(пространствами)*. Регион плотный - если доля входящих векторов содержит регион превышающий τ - другой параметр, определённый пользователем. Теперь рассмотрим граф, в котором вершины - плотные регионы, и ребро между вершинами. Кластер определяется как соединенные компоненты этого графа. Определения дают возможность выполнять кластеризацию в полно размерном пространстве. Во-первых, определяют плотные регионы, затем строят граф для поиска кластеров. Вторым шагом генерируют ДНФ форму для каждого кластера с помощью дизъюнкции описания каждого региона в кластере и затем эвристически упрощая результирующие выражения.

8 Rock

Rock - иерархический алгоритм кластеризации. Представляется он следующим образом:

Procedure cluster (S, k)
Begin
1. link := compute-links (S)
2. for each s from S do
3. q[s] := build-local-heap (link,S)
4. Q:= build-global-heap (S,q)
5. while size (Q) > k do {
6. u := extract-max (Q)
7. v := max (q[u])
8. delete (Q,v)
9. w:= merge (u,v)
10. for each x from (q[u] or q[v]) do {
11. link [x,w] := link [x,u] + link [x,v]
12. delete (q[x],u); delete (q[x],v)
13. insert (q[x],w,g(x,w)); insert (q[w],x,g(x,w));
14. update (Q,x,q[x])
15. }
16. insert (Q,w,q[w])
17. deallocate (q[u]); deallocate (q[v]);
18. }
end.

Этот алгоритм берёт в качестве входных параметров множество S из n пробных точек для кластеризации, и k - количество требуемых кластеров. Сначала считается количество ссылок между парами точек. Изначально, каждая точка - отдельный кластер.

Подсчет ссылок:
Procedure compute-links (S)
Begin
1. Compute nbrlist [I] for every point i in S
2. Set link [i,j] to be zero for all i,j
3. for i:=1 to n do {
4. N:= nbrlist[I]
5. for j:=1 to |N| -1 do
6. for l:= j+1 to |N| do
7. link[N[j], N[l]] := link [N[j],N[l]]+1
8. }
end.

Подробнее об этом алгоритме можно прочитать:
http://citeseer.ist.psu.edu/guha00rock.html

9 Дополнительные материалы

- http://logic.pdmi.ras.ru/ics/papers/aca.pdf
- http://citeseer.ist.psu.edu/guha00rock.html