Лабораторная работа №8.

Ассоциативные правила.

Цель работы: Поиск ассоциативных правил в заданном наборе транзакций

Нужно сделать

С помощью алгоритма Apriory для заданного набора транзакций найти наиболее значимые ассоциативные правила. На каждом шаге строить диаграммы поддержки найденных наборов. Для найденных правил вычислить лифт, левередж и улучшение. Объяснить найденные правила с точки зрения предметной области.

К отчету

Документ с реализованным алгоритмом поиска ассоциативных правил для заданного примера.

Методы поиска ассоциативных правил

Ассоциация - одна из задач Data Mining. Целью поиска ассоциативных правил (association rule) является нахождение закономерностей между связанными событиями в базах данных.

Очень часто покупатели приобретают не один товар, а несколько. В большинстве случаев между этими товарами существует взаимосвязь. Так, например, покупатель, приобретающий макаронные изделия, скорее всего, захочет приобрести также кетчуп. Эта информация может быть использована для размещения товара на прилавках.

Часто встречающиеся приложения с применением ассоциативных правил:

- розничная торговля: определение товаров, которые стоит продвигать совместно; выбор местоположения товара в магазине; анализ потребительской корзины; прогнозирование спроса;
- перекрестные продажи: если есть информация о том, что клиенты приобрели продукты А, Б и В, то какие из них вероятнее всего купят продукт Г?
- маркетинг: поиск рыночных сегментов, тенденций покупательского поведения;
- сегментация клиентов: выявление общих характеристик клиентов компании, выявление групп покупателей;
- оформление каталогов, анализ сбытовых кампаний фирмы, определение последовательностей покупок клиентов (какая покупка последует за покупкой товара А);
- анализ Web-логов.

Приведем простой пример ассоциативного правила: покупатель, приобретающий банку краски, приобретет кисточку для краски с вероятностью 50%.

Введение в ассоциативные правила

Впервые задача поиска ассоциативных правил (association rule mining) была предложена для нахождения типичных шаблонов покупок, совершаемых в супермаркетах, поэтому иногда ее еще называют анализом рыночной корзины (market basket analysis).

Рыночная корзина - это набор товаров, приобретенных покупателем в рамках одной отдельно взятой транзакции.

Транзакции являются достаточно характерными операциями, ими, например, могут описываться результаты посещений различных магазинов.

Транзакция - это множество событий, которые произошли одновременно.

Регистрируя все бизнес-операции в течение всего времени своей деятельности, торговые компании накапливают огромные собрания транзакций. Каждая такая транзакция представляет собой набор товаров, купленных покупателем за один визит.

Полученные в результате анализа шаблоны включают перечень товаров и число транзакций, которые содержат данные наборы.

Транзакционная или операционная база данных (Transaction database) представляет собой двумерную таблицу, которая состоит из номера транзакции (TID) и перечня покупок, приобретенных во время этой транзакции.

TID - уникальный идентификатор, определяющий каждую сделку или транзакцию.

Пример транзакционной базы данных, состоящей из покупательских транзакций, приведен в таблице 15.1. В таблице первая колонка (TID) определяет номер транзакции, во второй колонке таблицы приведены товары, приобретенные во время определенной транзакции.

Таблица 15.1. Транзакционная база

TID	Приобретенные покупки
100	Хлеб, молоко, печенье
200	Молоко, сметана
300	Молоко, хлеб, сметана, печенье
400	Колбаса, сметана
500	Хлеб, молоко, печенье, сметана

На основе имеющейся базы данных нам нужно найти закономерности между событиями, то есть покупками.

Ассоциативное правило состоит из двух наборов предметов, называемых условием и следствием, записываемых в виде X→Y.

Допустим, имеется транзакционная база данных D. Присвоим значениям товаров переменные (таблица 15.2).

Хлеб = a

Молоко = b

Печенье = c

Сметана = d

Колбаса = e

Конфеты = f

Таблица 15.2. Часто встречающиеся наборы товаров

TID	Приобретенные покупки	→	TID	Приобретенные покупки
100	Хлеб, молоко, печенье		100	a, b, c
200	Молоко, сметана		200	b, d
300	Молоко, хлеб, сметана, печенье		300	b, a, d, c
400	Колбаса, сметана		400	e, d
500	Хлеб, молоко, печенье, сметана		500	a, b, c, d
600	Конфеты		600	f

Рассмотрим набор товаров (Itemset), включающий, например, {Хлеб, молоко, печенье}. Выразим этот набор с помощью переменных:

abc={a,b,c}

Этот набор товаров встречается в нашей базе данных три раза, т.е. поддержка этого набора товаров равна 3:

SUP(abc)=3.

При минимальном уровне поддержки, равной трем, набор товаров abc является часто встречающимся шаблоном.

min_sup=3, {Хлеб, молоко, печенье} - часто встречающийся шаблон.

Поддержкой называют количество или процент транзакций, содержащих определенный набор данных.

Для данного набора товаров поддержка, выраженная в процентном отношении, равна 50%.

SUP(abc)=(3/6)*100%=50%

Поддержку иногда также называют обеспечением набора.

Таким образом, набор представляет интерес, если его поддержка выше определенного пользователем минимального значения (min support). Эти наборы называют часто встречающимися (frequent).

Характеристики ассоциативных правил

Ассоциативное правило имеет вид: "Из события A следует событие B": A→B

В результате такого вида анализа мы устанавливаем закономерность следующего вида: "Если в транзакции встретился набор товаров (или набор элементов) A, то можно сделать вывод, что в этой же транзакции должен появиться набор элементов B)" Установление таких закономерностей дает нам возможность находить очень простые и понятные правила, называемые ассоциативными.

Основными характеристиками ассоциативного правила являются поддержка и достоверность правила.

Рассмотрим правило "из покупки молока следует покупка печенья" для базы данных, которая была приведена выше в таблице 15.1. Понятие поддержки набора мы уже рассмотрели. Существует понятие поддержки правила.

Поддержка ассоциативного правила – это число транзакций, которые содержат как условие, так и следствие.

Например, для ассоциации A→B можно записать

S(A→B)=P(A∩B)= (количество транзакций, содержащих A и B)/(общее число транзакций)

Правило имеет поддержку s, если s% транзакций из всего набора содержат одновременно наборы элементов A и B или, другими словами, содержат оба товара.

Молоко - это товар A, печенье - это товар B. Поддержка правила "из покупки молока следует покупка печенья" равна 3, или 50%.

Достоверность правила показывает, какова вероятность того, что из события A следует событие B.

Достоверность ассоциативного правила A→B представляет собой меру точности правила и определяется как отношение количества транзакций, содержащих и условие и следствие, к количеству транзакций, содержащих только условие:

C(A→B)=P(A|B)=P(A∩B)/P(A)=

= (количество транзакций, содержащих A и B)/(количество транзакций, содержащих только A)

Правило "Из A следует B" справедливо с достоверностью c, если c% транзакций из всего множества, содержащих набор элементов A, также содержат набор элементов B.

Число транзакций, содержащих молоко, равно четырем, число транзакций, содержащих печенье, равно трем, достоверность правила равна (3/4)*100%, т.е. 75%.

Достоверность правила "из покупки молока следует покупка печенья" равна 75%, т.е. 75% транзакций, содержащих товар A, также содержат товар B.

Границы поддержки и достоверности ассоциативного правила

При помощи использования алгоритмов поиска ассоциативных правил аналитик может получить все возможные правила вида "Из A следует B", с различными значениями поддержки и достоверности. Однако в большинстве случаев, количество правил

необходимо ограничивать заранее установленными минимальными и максимальными значениями поддержки и достоверности.

Если значение поддержки правила слишком велико, то в результате работы алгоритма будут найдены правила очевидные и хорошо известные. Слишком низкое значение поддержки приведет к нахождению очень большого количества правил, которые, возможно, будут в большей части необоснованными, но не известными и не очевидными для аналитика. Таким образом, необходимо определить такой интервал, "золотую середину", который с одной стороны обеспечит нахождение неочевидных правил, а с другой - их обоснованность.

Если уровень достоверности слишком мал, то ценность правила вызывает серьезные сомнения. Например, правило с достоверностью в 3% только условно можно назвать правилом.

Аналитики могут отдавать предпочтение правилам, которые имеют только высокую поддержку или только высокую достоверность, или оба этих показателя. Правила, для которых значения поддержки или достоверности превышают определенный, заданный пользователем порог, называются сильными правилами. Например, аналитика может интересовать, какие товары, покупаемые вместе в супермаркете, образуют ассоциации с минимальной поддержкой 20% и минимальной достоверностью 70%. А при анализе с целью обнаружения мошенничеств может потребоваться уменьшить поддержку до 1%, поскольку с мошенничеством связано сравнительно небольшое число транзакций.

Значимость ассоциативных правил

Методики поиска ассоциативных правил обнаруживают все ассоциации, которые удовлетворяют ограничениям на поддержку и достоверность, наложенным пользователем. Это приводит к необходимости рассматривать десятки и сотни тысяч ассоциаций, что делает невозможным обработку такого количества данных вручную. Число правил желательно уменьшить таким образом, чтобы проанализировать только наиболее значимые из них. Значимость часто вычисляется как разность между поддержкой правила и в целом и произведением поддержки только условия и поддержки только следствия.

Если условие и следствие независимы, то поддержка правила примерно соответствует произведению поддержек условия и следствия, то есть $S_{AB} \approx S_A S_B$. Это значит, что хотя условие и следствие часто встречаются вместе, не менее часто они встречаются и по отдельности. Например, если товар A встречался в 70 транзакциях из 100, а товар B – в 80, и в 50 транзакциях из 100 они встречаются вместе, то несмотря на высокую поддержку ($S_{AB}=0.5$), это не обязательно правило. Просто эти товары покупаются независимо друг от друга, но в силу их популярности часто встречаются в одной транзакции. Так как $S_A S_B=0.7\ 0.8=0{,}56$ отличается от S_{AB} всего на 0.06, то предположение о независимости товаров A и B достаточно обоснованно.

Субъективные меры значимости ассоциативных правил

Лифт вычисляется следующим образом:

$$L(A \rightarrow B)=C(A \rightarrow B)/S(B).$$

Лифт – это отношение частоты появления условия в транзакциях, которые также содержат и следствие, к частоте появления следствия в целом. Значения лифта, большие, чем 1,

показывают, что условие чаще появляется в транзакциях, содержащих следствие, чем в остальных. Можно сказать, что лифт является обобщенной мерой связи двух предметных наборов: при значениях лифта больше 1 связь положительная, при 1 она отсутствует, при значениях меньше 1 – отрицательная.

Левередж вычисляется следующим образом:

$$T(A \rightarrow B)=S(A \rightarrow B)-S(A)S(B).$$

Левередж – это разность между наблюдаемой частотой, с которой условие и следствие появляются совместно (т.е. с поддержкой ассоциации), и произведением частот появления (поддержек) условия и следствия по отдельности. Из ассоциаций с одинаковым лифтом ассоциация с большим левереджем представляет больший интерес, так как это говорит о том, что данное правило встречаются чаще.

Улучшение вычисляется следующим образом:

$$I(A \rightarrow B)=S(A \rightarrow B)/(S(A)S(B)).$$

Улучшение показывает, полезнее ли правило случайного угадывания. Если $I(A \rightarrow B)>1$, это значит, что вероятнее предсказать наличие набора B с помощью правила, чем угадать случайно.

Методы поиска ассоциативных правил

В процессе поиска ассоциативных правил может производится обнаружение всех ассоциаций, поддержка и достоверность для которых превышают заданный минимум. Простейший алгоритм поиска ассоциативных правил рассматривает все возможные комбинации условий и следствий, оценивает для них поддержку и достоверность, а затем исключает все ассоциации, которые не удовлетворяют заданным ограничениям. Число возможных ассоциаций с увеличением числа предметов растет экспоненциально. Поэтому в процессе генерации ассоциативных правил широко используются методики, позволяющие уменьшить количество ассоциаций, которое требуется проанализировать.

Алгоритм AIS. Первый алгоритм поиска ассоциативных правил, называвшийся AIS, (предложенный Agrawal, Imielinski and Swami) был разработан сотрудниками исследовательского центра IBM Almaden в 1993 году. С этой работы начался интерес к ассоциативным правилам; на середину 90-х годов прошлого века пришелся пик исследовательских работ в этой области, и с тех пор каждый год появляется несколько новых алгоритмов.

В алгоритме AIS кандидаты множества наборов генерируются и подсчитываются "на лету", во время сканирования базы данных.

Алгоритм SETM. Создание этого алгоритма было мотивировано желанием использовать язык SQL для вычисления часто встречающихся наборов товаров. Как и алгоритм AIS, SETM также формирует кандидатов "на лету", основываясь на преобразованиях базы данных. Чтобы использовать стандартную операцию объединения языка SQL для формирования кандидата, SETM отделяет формирование кандидата от их подсчета.

Неудобство алгоритмов AIS и SETM - излишнее генерирование и подсчет слишком многих кандидатов, которые в результате не оказываются часто встречающимися. Для улучшения их работы был предложен алгоритм **Apriori.**

Алгоритм Apriori

В основе алгоритма Apriori лежит понятие частого набора. Под частотой понимается простое количество транзакций в которых содержится данный предметный набор.

Частый предметный набор – предметный набор с поддержкой больше заданного порога либо равной ему. Этот порог называется минимальной поддержкой.

Работа данного алгоритма состоит из нескольких этапов, каждый из этапов состоит из следующих шагов:

- формирование кандидатов;
- подсчет кандидатов.

Формирование кандидатов (candidate generation) - этап, на котором алгоритм, сканируя базу данных, создает множество i-элементных кандидатов (i - номер этапа). На этом этапе поддержка кандидатов не рассчитывается.

Подсчет кандидатов (candidate counting) - этап, на котором вычисляется поддержка каждого i-элементного кандидата. Здесь же осуществляется отсечение кандидатов, поддержка которых меньше минимума, установленного пользователем (min_sup). Оставшиеся i-элементные наборы называем часто встречающимися.

Чтобы сократить пространство поиска ассоциативных правил, алгоритм Apriori использует свойство антимонотонности. Свойство утверждает, что если предметный набор Z не является частым, то добавление некоторого нового предмета A к набору Z не делает его более частым. Данное полезное свойство позволяет значительно уменьшить пространство поиска ассоциативных правил.

На первом этапе алгоритма Apriori формируются частые однопредметные наборы – множество F_1.

Для поиска F_k, то есть k-предметных наборов, алгоритм Apriori сначала создает множество F_k кандидатов в k-предметные наборы путем связывания множества F_{k-1} с самим собой. Затем F_k сокращается с использованием свойства антимонотонности. Предметные наборы множества F_k, которые остались после сокращения, формируют F_k.

После того, как все частые предметные наборы найдены, можно переходить к генерации на их основе ассоциативных правил. Для этого к каждому частому предметному набору s можно применить процедуру, состоящую из 2 шагов.

1. Генерируются все возможные поднаборы s
2. Если поднабор ss является непустым поднабором s, то рассматривается ассоциация R:ss→(s-ss), где s-ss представляет собой набор s без поднабора ss. R считается ассоциативным правилом, если удовлетворяет условию заданного минимума поддержки и достоверности. Данная процедура повторяется для каждого подмножества ss из s.

Рассмотрим работу алгоритма Apriori на примере базы данных D. Иллюстрация работы алгоритма приведена на рис. 15.1. Минимальный уровень поддержки равен 3.

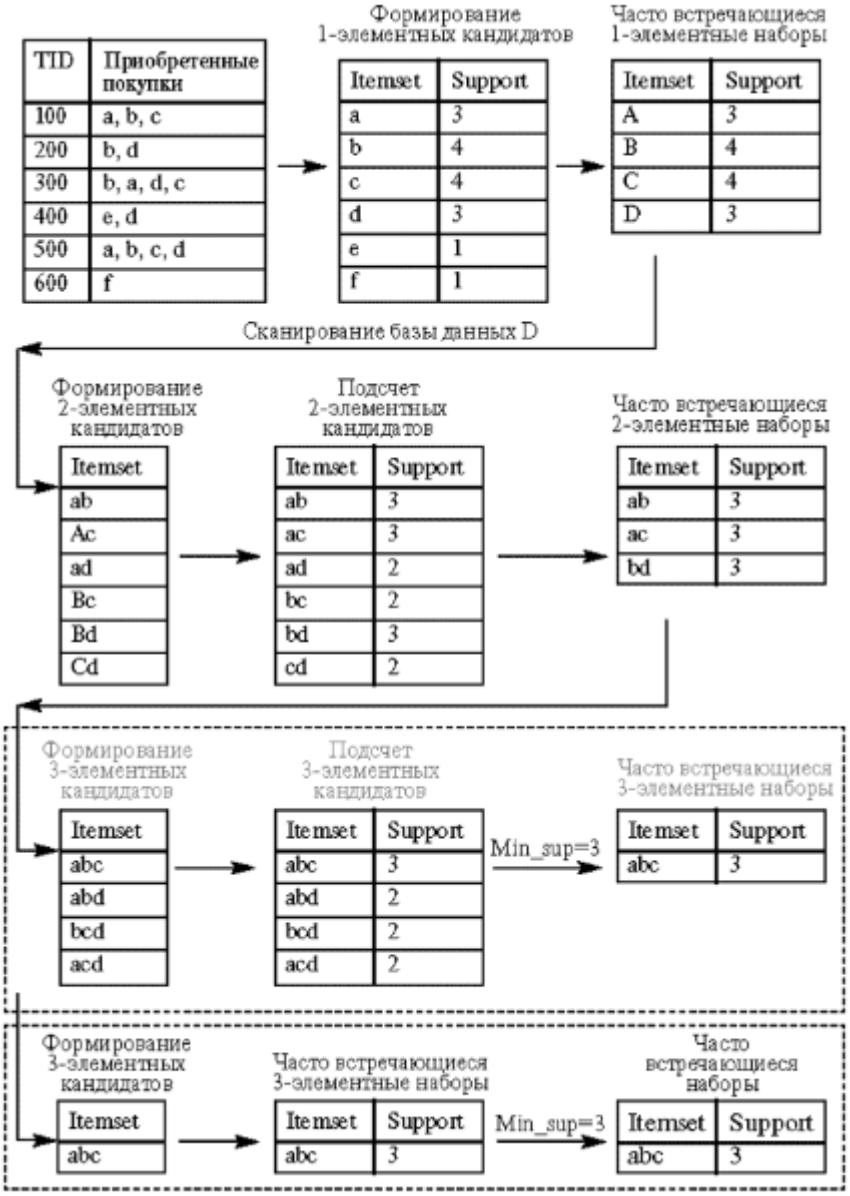

Рис. 15.1. Алгоритм Apriori

На первом этапе происходит формирование одноэлементных кандидатов. Далее алгоритм подсчитывает поддержку одноэлементных наборов. Наборы с уровнем поддержки меньше установленного, то есть 3, отсекаются. В нашем примере это наборы e и f, которые имеют поддержку, равную 1. Оставшиеся наборы товаров считаются часто встречающимися одноэлементными наборами товаров: это наборы a, b, c, d.

Далее происходит формирование двухэлементных кандидатов, подсчет их поддержки и отсечение наборов с уровнем поддержки, меньшим 3. Оставшиеся двухэлементные наборы товаров, считающиеся часто встречающимися двухэлементными наборами ab, ac, bd, принимают участие в дальнейшей работе алгоритма.

Если смотреть на работу алгоритма прямолинейно, на последнем этапе алгоритм формирует трехэлементные наборы товаров: abc, abd, bcd, acd, подсчитывает их

поддержку и отсекает наборы с уровнем поддержки, меньшим 3. Набор товаров abc может быть назван часто встречающимся.

Однако алгоритм Apriori уменьшает количество кандидатов, отсекая - априори - тех, которые заведомо не могут стать часто встречающимися, на основе информации об отсеченных кандидатах на предыдущих этапах работы алгоритма.

Отсечение кандидатов происходит на основе предположения о том, что у часто встречающегося набора товаров все подмножества должны быть часто встречающимися. Если в наборе находится подмножество, которое на предыдущем этапе было определено как нечасто встречающееся, этот кандидат уже не включается в формирование и подсчет кандидатов.

Так наборы товаров ad, bc, cd были отброшены как нечасто встречающиеся, алгоритм не рассматривал товаров abd, bcd, acd.

При рассмотрении этих наборов формирование трехэлементных кандидатов происходило бы по схеме, приведенной в верхнем пунктирном прямоугольнике. Поскольку алгоритм априори отбросил заведомо нечасто встречающиеся наборы, последний этап алгоритма сразу определил набор abc как единственный трехэлементный часто встречающийся набор (этап приведен в нижнем пунктирном прямоугольнике).

Алгоритм Apriori рассчитывает также поддержку наборов, которые не могут быть отсечены априори. Это так называемая негативная область (negative border), к ней принадлежат наборы-кандидаты, которые встречаются редко, их самих нельзя отнести к часто встречающимся, но все подмножества данных наборов являются часто встречающимися.

Разновидности алгоритма Apriori

В зависимости от размера самого длинного часто встречающегося набора алгоритм Apriori сканирует базу данных определенное количество раз. Разновидности алгоритма Apriori, являющиеся его оптимизацией, предложены для сокращения количества сканирований базы данных, количества наборов-кандидатов или того и другого. Были предложены следующие разновидности алгоритма Apriori: AprioriTID и AprioriHybrid.

AprioriTid

Интересная особенность этого алгоритма - то, что база данных D не используется для подсчета поддержки кандидатов набора товаров после первого прохода.

С этой целью используется кодирование кандидатов, выполненное на предыдущих проходах. В последующих проходах размер закодированных наборов может быть намного меньше, чем база данных, и таким образом экономятся значительные ресурсы.

AprioriHybrid

Анализ времени работы алгоритмов Apriori и AprioriTid показывает, что в более ранних проходах Apriori добивается большего успеха, чем AprioriTid; однако AprioriTid работает лучше Apriori в более поздних проходах. Кроме того, они используют одну и ту же процедуру формирования наборов-кандидатов. Основанный на этом наблюдении, алгоритм AprioriHybrid предложен, чтобы объединить лучшие свойства алгоритмов

Apriori и AprioriTid. AprioriHybrid использует алгоритм Apriori в начальных проходах и переходит к алгоритму AprioriTid, когда ожидается, что закодированный набор первоначального множества в конце прохода будет соответствовать возможностям памяти. Однако, переключение от Apriori до AprioriTid требует вовлечения дополнительных ресурсов.

Некоторыми авторами были предложены другие алгоритмы поиска ассоциативных правил, целью которых также было усовершенствование алгоритма Apriori.

Один из них - **алгоритм DHP**, также называемый алгоритмом хеширования (J. Park, M. Chen and P. Yu, 1995 год). В основе его работы - вероятностный подсчет наборов-кандидатов, осуществляемый для сокращения числа подсчитываемых кандидатов на каждом этапе выполнения алгоритма Apriori. Сокращение обеспечивается за счет того, что каждый из k-элементных наборов-кандидатов помимо шага сокращения проходит шаг хеширования. В алгоритме на k-1 этапе во время выбора кандидата создается так называемая хеш-таблица. Каждая запись хеш-таблицы является счетчиком всех поддержек k-элементных наборов, которые соответствуют этой записи в хеш-таблице. Алгоритм использует эту информацию на этапе k для сокращения множества k-элементных наборов-кандидатов. После сокращения подмножества, как это происходит в Apriori, алгоритм может удалить набор-кандидат, если его значение в хеш-таблице меньше порогового значения, установленного для обеспечения.

К другим усовершенствованным алгоритмам относятся: PARTITION, DIC, алгоритм "выборочного анализа".

PARTITION алгоритм (A. Savasere, E. Omiecinski and S. Navathe, 1995 год). Этот алгоритм разбиения (разделения) заключается в сканировании транзакционной базы данных путем разделения ее на непересекающиеся разделы, каждый из которых может уместиться в оперативной памяти. На первом шаге в каждом из разделов при помощи алгоритма Apriori определяются "локальные" часто встречающиеся наборы данных. На втором подсчитывается поддержка каждого такого набора относительно всей базы данных. Таким образом, на втором этапе определяется множество всех потенциально встречающихся наборов данных.

Алгоритм DIC, Dynamic Itemset Counting (S. Brin R. Motwani, J. Ullman and S. Tsur, 1997 год). Алгоритм разбивает базу данных на несколько блоков, каждый из которых отмечается так называемыми "начальными точками" (start point), и затем циклически сканирует базу данных.