УДК 336.74:004.738.5

ПОИСК АССОЦИАТИВНЫХ ПРАВИЛ С ПОМОЩЬЮ АЛГОРИТМА APRIORI

А.А. Васькевич
Магистрант кафедры математического и информационного обеспечения экономических систем
УО ГрГУ.им. Я.Купалы

Н.В. Марковская
Доцент кафедры математического и информационного обеспечения экономических систем
УО ГрГУ.им. Я.Купалы

УО Гродненский государственный университет имени Янки Купалы
E-mail: vaskevich.artem@gmail.com, n.markovskaya@grsu.by

А.А. Васькевич
Окончил Гродненский государственный университет имени Янки Купалы. Магистрант кафедры математического и информационного обеспечения экономических систем УО ГрГУ им. Я. Купалы.

Н.В. Марковская
Доцент кафедры математического и информационного обеспечения экономических систем УО ГрГУ им.Я.Купалы, кандидат физико-математических наук, доцент.

Аннотация. В данной работе рассмотрены ассоциативные правила. **Associations rules learning** – ARL представляет из себя, простой, но довольно часто применимый в реальной жизни метод поиска взаимосвязей (ассоциаций) в датасетах, или, если точнее, айтемсетах (itemsests). Основным алгоритмом, который применяется для получения ассоциативных правил, является алгоритм apriori. Его автором является Ракеш Агравал (Rakesh Agrawal, сейчас сотрудник Microsoft Research). Алгоритм Apriori предназначен для поиска всех частых множеств признаков. Он является поуровневым, использует стратегию поиска в ширину и осуществляет его снизу-вверх.

Ключевые слова: рыночная корзина, транзакции, ассоциативные правила, алгоритм априори, поддержка, база данных, алгоритм поиска, бизнес операция, поддержка, достоверность.

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 *тран-*

закций, 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*.

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

Рисунок 1. Транзакционная база данных

В 1992 году группа по консалтингу в области ритейла компании Teradata под руководством Томаса Блишока провела исследование 1.2 миллиона транзакций в 25 магазинах для ритейлера Osco Drug (нет, там продавали не наркотики и даже не лекарства, точнее, не только лекартсва. Drug Store – формат разнокалиберных магазинов у дома). После анализа всех этих транзакций самым сильным правилом получилось «Между 17:00 и 19:00 чаще всего пиво и подгузники покупают вместе». К сожалению такое правило показалось руководству Osco Drug настолько контринтуитивным, что ставить подгузники на полках рядом с пивом они не стали. Хотя объяснение паре пиво-подгузники вполне себе нашлось: когда оба члена молодой семьи возвращались с работы домой (как раз часам к 5 вечера), жены обычно отправляли мужей за подгузниками в ближайший магазин. И мужья, не долго думая, совмещали приятное с полезным – покупали подгузники по заданию жены и пиво для собственного вечернего времяпрепровождения.

Итак, мы выяснили, что пиво и подгузники – хороший джентльменский набор, а что дальше?

Пусть у нас имеется некий датасет (или коллекция) D, такой, что D=d0...dj, где d – уникальная транзакция-itemset (например, кассовый чек). Внутри каждой d представлен набор items (i – item), причем в идеальном случае он представлен в бинарном виде: d1 = [{Пиво: 1}, {Вода: 0}, {Кола: 1}, {...}], d2 = [{Пиво: 0}, {Вода: 1}, {Кола: 1}, {...}]. Принято каждый itemset описывать через количество ненулевых значений (k-itemset), например, [{Пиво: 1}, {Вода: 0}, {Кола: 1}] является 2-itemset.

Если изначально датасет в бинарном виде не представлен, можно при желании руками его преобразовать.

Таким образом, датасет представляет собой разреженную матрицу со значениями {1,0}. Это будет бинарный датасет. Существуют и другие виды записи – вертикальный датасет (показывает для каждого отдельного item вектор транзакций, где он присутствует) и транзакционный датасет (примерно как в кассовом чеке).

Данные преобразовали, как найти правила?

Существует целый ряд ключевых (если хотите – базовых) понятий в ARL, которые нам помогут эти правила вывести.

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 существляется *отсечение* кандидатов, *поддержка* которых меньше минимума, установленного пользователем (min_sup). Оставшиеся i-элементные наборы называем часто встречающимися.

Support (поддержка) Первое понятие в ARL – support:

$$supp(X)=\{t\in T;\ X\in t\}|T|$$

где X – itemset, содержащий в себе i-items, а T – количество транзакций. Т.е. в общем виде это показатель «частотности» данного itemset во всех анализируемых транзакциях. Но это касается только X. Нам же интересен скорее вариант, когда у нас в одном itemset встречаются x1 и x2 (например). Ну тут тоже все просто. Пусть x1 = {Пиво}, а x2 = {Подгузники}, значит нам нужно посчитать, во скольких транзакциях встречается эта парочка.

$$supp(x1\cup x2)=\sigma(x1\cup x2)|T|$$

где σ – количество транзакций,содержащих x1 и x2 Разберемся с этим понятием на игрушечном датасете. play_set # микродатасет, где указаны номера транзакций, а также в бинарном виде представлено, что покупалось на каждой транзакции

	Transaction	Кола	Пиво	Подгузники
0	1	1	1	1
1	2	0	0	0
2	3	1	1	0
3	4	1	1	1
4	5	0	1	1

Рисунок 2. Номера транзакций

ТранзакцииспивомиподгузникамиВсетранзакцииПивоПодгузникиsupp=Транзакции с пивом и подгузникамиВсе транзакции=P(Пиво∩Подгузники)

$$supp=3/5=60\%$$

Confidence (достоверность)

Следующее ключевое понятие – confidence. Это показатель того, как часто наше правило срабатывает для всего датасета.

$$conf(x1\cup x2)=supp(x1\cup x2)supp(x1)$$

Приведем пример: мы хотим посчитать confidence для правила «кто покупает пиво, тот покупает и подгузники».

Для этого сначала посчитаем, какой support у правила «покупает пиво», потом посчитаем support у правила «покупает пиво и подгузники», и просто поделим одно на другое. Т.е. мы посчитаем в скольких случаях (транзакциях) срабатывает правило «купил пиво» supp(X), «купил подгузники и пиво»

$$supp(X\cup Y)$$

Lift (поддержка)

Следующее понятие в нашем списке – lift. Грубо говоря, lift – это отношение «зависимости» items к их «независимости». Lift показывает, насколько items зависят друг от друга. Это очевидно из формулы:

$$lift(x1 \cup x2) = supp(x1 \cup x2) / supp(x1) \times supp(x2)$$

Например, мы хотим понять зависимость покупки пива и покупки подгузников. Для этого считаем support правила «купил пиво и подгузники» и делим его на произведение правил «купил пиво» и «купил подгузники». В случае, если *lift = 1*, мы говорим, что items независимы и правил совместной покупки тут нет. Если же *lift > 1*, то величина, на которую lift, собственно, больше этой самой единицы, и покажет нам «силу» правила. Чем больше единицы, тем круче. Если lift < 1, то это покажет, что правило основания x_2 негативно влияет на правило x_1. По-другому lift можно определить, как отношение confidence к expected confidence, т.е. отношение достоверности правила, когда оба (ну или сколько там захотите) элемента покупаются вместе к достоверности правила, когда один из элементов покупался (неважно, со вторым или без).

$$lift = Confidence / Expected\ confidence = P(Подгузники|Пиво) / P(Подгузники)$$
$$lift = (3/4)/(3/5) = 1{,}25$$

Т.е. наше правило, что пиво покупают с подгузниками, на 25% мощнее правила, что подгузники просто покупают.

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 набор товаров abd, *bcd, acd*.

При рассмотрении этих наборов формирование трехэлементных кандидатов происхо-

дило 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. Кратко изложим суть нескольких, для более подробной информации можно рекомендовать.

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.

Возьмем базу Adult, содержащуюся в пакете «arules»

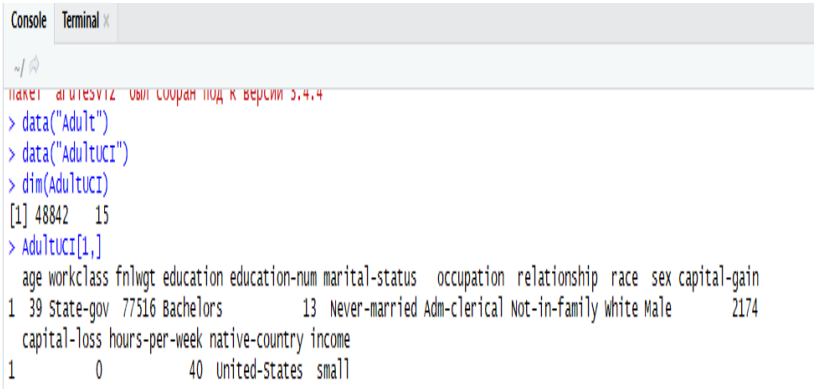

Рисунок 3. Подключение базы Adult

Для анализа данных используем алгоритм Apriori с минимальной поддержкой 0.02, и значимостью 0.6

```
> rules <- apriori(Adult, parameter=list(support=0.02, confidence=0.6))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.6    0.1    1 none FALSE            TRUE       5    0.02      1     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 976

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[115 item(s), 48842 transaction(s)] done [0.02s].
sorting and recoding items ... [59 item(s)] done [0.01s].
creating transaction tree ... done [0.03s].
checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [0.38s].
writing ... [111133 rule(s)] done [0.03s].
creating S4 object  ... done [0.06s].
```

Рисунок 4. Анализ данных

В результате работы алгоритм вывел 111133 правил.

```
> summary(rules)
set of 111133 rules

rule length distribution (lhs + rhs):sizes
    1     2     3     4     5     6     7     8     9    10
    6   368  3253 12503 26051 31595 23283 10733  2929   412

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   5.000   6.000   5.936   7.000  10.000

summary of quality measures:
    support          confidence          lift             count
 Min.   :0.02000   Min.   :0.6000   Min.   : 0.7521   Min.   :  977
 1st Qu.:0.02416   1st Qu.:0.7745   1st Qu.: 1.0070   1st Qu.: 1180
 Median :0.03190   Median :0.9053   Median : 1.0488   Median : 1558
 Mean   :0.04618   Mean   :0.8610   Mean   : 1.2817   Mean   : 2256
 3rd Qu.:0.04908   3rd Qu.:0.9527   3rd Qu.: 1.2573   3rd Qu.: 2397
 Max.   :0.95328   Max.   :1.0000   Max.   :20.0648   Max.   :46560

mining info:
  data ntransactions support confidence
 Adult         48842    0.02        0.6
```

Рисунок 5. Список правил

Изобразим список правил графически:

Рисунок 6. График правил

Выполним выборку правил:

```
> inspect(head(sort(rulesIncomeSmall,by = "support"), n = 5))
    lhs                                rhs             support confidence     lift count
[1] {capital-gain=High}             => {income=large} 0.02319725  0.6704142 4.176045  1133
[2] {capital-gain=High,
     capital-loss=None}             => {income=large} 0.02319725  0.6704142 4.176045  1133
[3] {capital-gain=High,
     native-country=United-States}  => {income=large} 0.02119078  0.6720779 4.186409  1035
[4] {capital-gain=High,
     capital-loss=None,
     native-country=United-States}  => {income=large} 0.02119078  0.6720779 4.186409  1035
[5] {race=White,
     capital-gain=High}             => {income=large} 0.02069940  0.6682089 4.162308  1011
```

Рисунок 7. Выборка правил

Исходя из анализа можно сделать вывод что в первую очередь на уровень дохода влияет, размер капитала, затем страна проживания субъекта и в последнюю очередь его раса.

Литература

[1] Интернет ресурс - https://docplayer.ru/36774167-Associativnye-pravila-algoritm-apriori-m-102.html. Режим доступа – 18.01.2019

[2] Интернет ресурс - http://economics.studio/osnovyi-marketinga/assotsiativnyie-metodyi-13000.html. Режим доступа – 18.01.2019

[3] Интернет ресурс - https://ru.wikipedia.org/wiki/Apriori. Режим доступа – 18.01.2018

[4] R. Srikant, R. Agrawal. "Mining quantitative association rules in large relational tables". In Proceedings of the ACM SIGMOD Conference on Management of Data, Montreal, Canada, June 1996.

[5] J.S. Park, M.-S. Chen, and S.Y. Philip, "An Effective HashBased Algorithm for Mining Association Rules", In Proc. ACM SIGMOD Int'l Conf. Management of Data, ACM Press, New York, 1995.

SEARCH OF ASSOCIATIVE RULES BY USING OF THE ALGORITHM APRIORI

A.A. VASKEVICH
Master student of the Department of Mathematical and Information Support of Economic Systems, YKSUG

N.V. MARKOVSKAYA
Associate professor of the Department of Mathematical and Information Support of Economic Systems, YKSUG

Yanka Kupala State University of Grodno, Republic of Belarus
E-mail: vaskevich.artem@gmail.com, n.markovskaya@grsu.by

Abstract. In this paper, we consider associative rules. Associations rules learning - ARL is a simple, but quite often applicable in real life method of finding relationships (associations) in datasets, or, more precisely, data sets (itemsests). The basic algorithm that is used to obtain association rules is the apriori algorithm.

Keywords: market basket, transactions, association rules, a priori algorithm, support, database, search algorithm, business operation, support, validity.