УДК 519.8

А.Ю. Завгородний

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков

МЕТОД БЫСТРОГО ПОИСКА АССОЦИАТИВНЫХ ПРАВИЛ

За счет изменения традиционного подхода к хранению данных был получен эффективный метод поиска ассоциативных правил. Доказаны утверждения о кодировании исходных данных «без потерь».

данные, метод быстрого поиска, ассоциативные правила, кодирование без потерь

Введение

С развитием искусственного интеллекта и, в частности, экспертных систем особенно актуальной стала задача получения знаний в формализованном, ориентированным на машинную обработку виде. Практика создания экспертных систем показывает, что успешность проекта ориентированного на машинную обработку знаний в значительной мере зависит от эффективности процесса приобретения таких знаний. Более того, классический путь приобретения знаний путем взаимодействия разработчиков с экспертами зачастую становится причиной провала таких проектов. В то же время, сегодня для многих предметных областей огромные объемы детализированных данных стали доступны в электронном виде. Во многих случаях анализ таких данных позволяет выявить скрытые закономерности, т.е. получить новые знания о предметной области без участия эксперта. Вместе с тем, поскольку классические методы анализа данных не позволяют обрабатывать большие объемы данных, появилась потребность в новых эффективных машинно-ориентированных алгоритмах и методах анализа данных [1].

Одним из видов знаний, извлекаемых при анализе данных, являются ассоциативные правила, представляющие собой новые, зачастую нетривиальные, импликации связывающие атрибуты предметной области. Как и продукционным правилам, ассоциативным правил присуща простота интерпретации, возможность создания простого механизма их использования, что явилось причиной широкой востребованности методов поиска таких правил в различных приложениях: анализе потребительской корзины [2, 3], анализе содержимого Интернет-ресурсов [4], системах повышения качества данных информационных систем [5].

1. Существующие методы поиска ассоциативных правил

Формальная постановка задачи поиска ассоциативных правил заключается в следующем. Пусть дано множество $I = \{i_1, i_2, ..., i_q\}$ всех возможных значений атрибутов предметной области. Каждый элемент $i \in I$ является атрибутом, определенным на бинарном домене $\{0, 1\}$, где значение 1 показывает, что определенный кортеж содержит соответствующее значение атрибута. Каждая сущность предметной области представляется кортежем отношения R, определенного на I. Ассоциативным правилом называется импликация $X \Rightarrow Y$, где $X \subset I, Y \subset I$ и $X \cap Y = \varnothing$. Ассоциативное правило $X \Rightarrow Y$ обладает достоверностью c в отношении R, если c% кортежей в R содержащих X, также содержат Y. Правило $X \Rightarrow Y$ обладает поддержкой s, если s% кортежей содержат $X \cup Y$. Необходимо найти все ассоциативные правила поддержка и достоверность, которых не меньше некоторых заранее заданных s_{min} и c_{min} соответственно.

Несмотря на кажущуюся простоту, автоматический поиск всех искомых правил не является тривиальной задачей. Так, вычислительная сложность «наивного» подхода, предполагающего перебор всех возможных правил, будет экспоненциально зависеть от количества элементов множества I.

В работе [2] впервые было отмечено, что поддержка набора элементов X не может быть большей, чем поддержка любого подмножества из X. Использование такого утверждения позволило создать метод AIS [2], позволяющий находить ассоциативные правила за приемлемое время. Суть этого метода заключается в последовательном выполнении следующих двух этапов. На первом этапе последовательно находятся все часто встречающиеся наборы элементов в соответствии с системой

$$\begin{cases} C[k] = \{c_k \,|\, (\forall c_{k-1}, c_{k-1} \subset c_k \rightarrow c_{k-1} \in L_{k-1})\}; \\ L[k-1] = \{l_k \,|\, l_k \in C_k \wedge S(l_k) \geq s_{min}\}, \end{cases} \quad (1)$$

где C[k] – множество кандидатов в k-элементные часто встречающиеся наборы, получаемое с использованием упомянутого утверждения на основании множества L[k–1] – (k–1)-элементных часто встречающихся наборов.

На втором этапе, на основании множества найденных часто встречающихся наборов все правила, достоверность которых выше заданного порога. Вычислительная сложность данного этапа невелика в связи с тем, что выполняется следующее

условие:

$$c(X \Rightarrow Y) = \frac{s(X \cup Y)}{s(X)}, \qquad (2)$$

где $c(X \Rightarrow Y)$ – достоверность правила $X \Rightarrow Y$; $s(X \cup Y)$ – поддержка набора элементов $X \cup Y$; $s(X)$ – поддержка набора элементов X.

Идеи представленные в [2] получили дальнейшее развитие при создании наиболее известного на сегодняшний день метода поиска ассоциативных правил Apriori [3]. Как и ранее, метод поиска правил в [3] состоял из двух этапов, однако была существенно улучшена методика подсчета поддержки множества наборов путем использования т.н. хэш-деревьев.

Несмотря на то, что Apriori обладал линейной вычислительной сложностью относительно количества кортежей исходного отношения, попытки использования данного метода в практических приложениях показали его недостаточное быстродействие. Как показано в работе [6] основной причиной невысокого быстродействия Apriori следует назвать большое количество операций ввода/вывода (I/O) при работе с вторичной памятью, что в свою очередь вызвано большим объемом исходного отношения R и невозможностью разместить все данные из R в первичной памяти. Таким образом, при поиске часто встречающихся наборов Aprori требовалось (k + 1) * n "дорогих" операций или (k + 1) проходов по всем кортежам отношения R, где к – наибольшее возможное количество элементов в часто встречающихся наборах.

Для уменьшения количества I/O операций в методе Partition [6] было предложено разбивать исходное отношение R на отношения, помещающиеся в первичной памяти ($R = r_1 \cup r_2 \cup ... \cup r_v$). Затем для каждой «суботношения», таким же образом, как и в Apriori, выполняется поиск часто встречающихся наборов. Показано, что если набор часто встречается в R, то он будет часто встречающимся хотя бы в одном из отношений $r \in \{r_1, r_2, ..., r_v\}$. Далее для всех наборов найденных для $\{r_1, r_2, ..., r_v\}$ за один проход по R производится проверка, является ли он часто встречающимся в R. Таким образом, количество проходов сокращается до двух.

Помимо упомянутых ранее, создавались и другие методы. Однако, большинство из них обладают и существенными недостатками. Так, например, не умаляя достоинств методов созданных в работах [4, 7, 8], следует отметить, что методы в [4, 7] ориентированы на нахождение только 2-х элементных правил, а метод использующий выборку из R [8] в худшем случае даже уступает по быстродействию методу Partition.

2. Структура для представления исходных данных

Анализ существующих исследований в области поиска ассоциативных правил позволяет указать на основную причину невысокого быстродействия созданных методов – чрезвычайно большой объем исходных данных и, как следствие, большое количество выполняемых операций для работы с вторичной памятью на этапе поиска часто встречающихся наборов. Следует отметить, что такая проблема не является новой для компьютерных наук. Так, например, в реляционных системах управления базами данных (СУБД) широко применяются т.н. индексы – дополнительные массивы данных, объем которых меньше исходных, а структура оптимизирована для решения распространенных задач СУБД [9]. В связи с этим подход к решению задачи поиска ассоциативных правил, предполагающий введение более компактного отображения исходных данных представляется перспективным.

Для повышения скорости поиска ассоциативных правил такое отображение должно отвечать следующим требованиям: 1) объем памяти необходимый для хранения такого отображения, должен быть существенно меньшим относительно исходного отношения; 2) вычислительная сложность алгоритмов обеспечивающих актуальность отображения должна быть невелика с тем, чтобы производить соответствующие изменения отображения в режиме реального времени при вводе данных в исходное отношение R; 3) в случае, когда как отображение, так и исходное отношение R, помещаются в первичную память, скорость поиска часто встречающихся наборов по отображению не должна быть существенно меньше скорости поиска по исходному отношению; 4) метод поиска часто встречающихся наборов по отображению должен обеспечивать отсутствие ошибок 1-го рода (т.е. все часто встречающиеся наборы должны быть найдены) и незначительное количество ошибок 2-го рода (т.е. найденные методом наборы, не являющиеся часто встречающимися в исходном отношении R).

Учитывая тот факт, что в реальных приложениях плотность единиц в исходном отношении R чрезвычайно невелика [3], в настоящей работе предлагается для поиска часто встречающихся наборов использовать отображение, получаемое следующим образом.

Исходное отношение R произвольным образом разбиваем на v отношений (см. рис. 1) таким образом, чтобы выполнялись следующие условия:

$$R = r_1 \cup r_2 \cup ... \cup r_v;\ v = \left\| |R| / (2^m - m - 1) \right|;$$
$$\forall i, i = \overline{1, v} : |r_i| \le (2^m - m - 1) \forall i;$$
$$j, i = \overline{1, v};\ j = \overline{1, v};\ i \ne j : |r_i \cap r_j| = 0,$$

где $\cup, \cap$ – операции объединения и пересечения реляционной алгебры; $|R|$ – кардинальность отношения R; m – некоторое целое число (вопрос выбора конкретного значения u рассматривается далее).

На следующем шаге рассматриваем каждый атрибут каждого отношения r_i. Каждый такой атрибут можно представить, как цепочку w длиной u в би-

нарном алфавите $\Sigma = \{0, 1\}$, в которой каждая j-я позиция определяет значение соответствующего атрибута кортежа r_i с условным номером j. Тогда результирующее отображение можно представить как набор данных, в котором последовательно для каждой такой цепочки w содержится следующая информация (рис. 2): а) количество позиций цепочки w равных единице; б) значение хэш-функции H(w) от цепочки w, определяемое как проверочные разряды кодового слова кода Хэмминга в котором информационные разряды соответствуют цепочке w; в) отсортированная последовательность значений функции h(j), где j – все номера позиций цепочки w равные единице, h(j) проверочные разряды кодового слова кода Хэмминга, в котором информационные разряды соответствуют цепочке, в которой только j-я позиция равна единице.

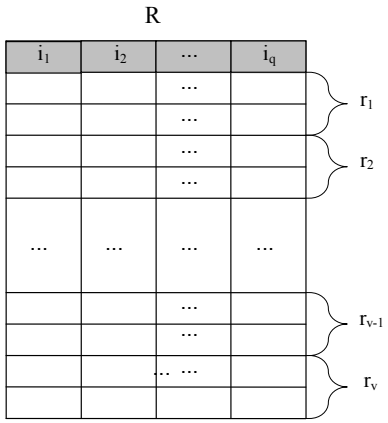

Рис. 1. Разбиение исходного отношения

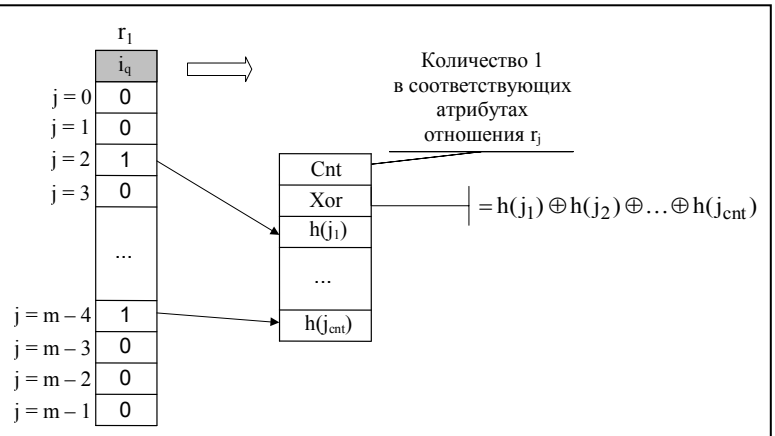

Рис. 2. Отображение атрибута «суботношения»

Введем для определенных ранее H(w) и h(j) следующие утверждения, доказательство которых следует непосредственно из свойств систематических кодов [10].

Утверждение 1. Для любого значения x из области значений функции h(j), существует единственное значение j такое что h(j) = x.

Следствием утверждения 1 является факт, что процесс получения предлагаемого отображения является кодированием без потерь и, следовательно, возможно создание корректного алгоритма поиска всех часто встречающихся наборов.

Утверждение 2. $H(w) = h(j_1) \oplus h(j_2) \oplus \ldots \oplus h(j_n)$, где $\{j_1, j_2, ..., j_n\}$ – множество всех номеров позиций, в которых w равно 1; $\oplus$ – символ сложения по модулю 2.

Оценим объем памяти, необходимый для хранения такого отображения. Пусть при разбиении исходного отношения R было выбрано некоторое целое число m. Тогда количество “суботношений”, на которые разбивалось R, равно $v = \lceil |R| / u \rceil$, где u – количество кортежей в v – 1 “суботношениях” равное $u = 2^m - m - 1$. Тогда количество проверочных разрядов в используемом коде Хэмминга равно m [10]. Количество цепочек, для которых информация сохраняется в отображении, равно $v * p_R$, где p_R – количество атрибутов в заголовке отношения R. Для каждой такой цепочки требуется $\lceil \log_2 u \rceil$ бит для хранения количества единиц в ней и m бит для хранения значения хэш-функции от цепочки. Кроме того, в отображении отводится m бит для каждой единицы из R. Таким образом, если предположить, что плотность единиц в R равна ρ, то объем памяти необходимый для хранения полученного из R отображения равен:

$$V_o(R) = v * p_R * (\lceil \log_2 u \rceil + m) + |R| * p_R * \rho * m . \quad (3)$$

Теперь, учитывая тот факт, что объем памяти необходимый для хранения исходного отношения R равен $V(R) = |R| * p_R$, путем тривиальных математических преобразований, мы можем оценить во сколько раз объем памяти необходимый для хранения предлагаемого отображения меньше относительно объема требуемого для исходного отношения

$$K_{сж} = \frac{V(R)}{V_o(R)} = \frac{u}{\lceil \log_2 u \rceil + m + m * \rho * u} . \quad (4)$$

Значения такого коэффициента сжатия при различной плотности единиц показаны на рис. 3.

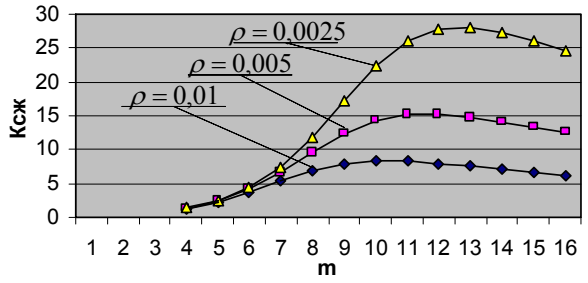

Рис. 3. Коэффициент сжатия при различной плотности единиц в исходном отношении R

Такие показатели позволяют сделать вывод о том, что при $m \geq 7$ предлагаемое отображение будет отвечать первому из выдвинутых требований.

Рассмотрим второе выдвинутое требование. Ал-

горитмы, обеспечивающие актуальность отображения, должны выполняться в тех случаях, когда производятся операции добавления, удаления, обновления кортежа в исходном отношении R. Каждая из перечисленных операций сводится к изменению одной позиции в от 1 до p_R цепочек-атрибутов одного суботношения r_i. В свою очередь, изменение одной такой цепочки w в одном разряде потребует от алгоритма выполнения следующих изменений в отображении: а) изменения на 1 суммарного количества позиций цепочки w равных единице; б) изменение значения хэш-функции H(w) в соответствии со следующим равенством, вытекающим из утверждения 2, $H(w_k) = H(w_{k-1}) \oplus h(j_k)$, где w_{k-1}, w_k – значения цепочки w до и после изменения соответственно; j_k – номер изменившегося разряда; в) вставка или удаление элемента $h(j_k)$ из отсортированной последовательности значений h(j), соответствующих единицам цепочки w_{k-1}. Учитывая тот факт, что в количество единиц в цепочках w невелико и в среднем равно u * ρ, можно утверждать, что среднее количество машинных операций необходимых для отображения одной изменившейся цепочки крайне невелико и равно u * ρ + 2, и, следовательно, предлагаемое отображение будет отвечать второму из выдвинутых требований.

3. Поиск часто встречающихся наборов атрибутов в памяти

Суть предлагаемого решения задачи поиска всех часто встречающихся наборов атрибутов по отображению в памяти, аналогична решению представленному в [3] и также может быть представлена соотношением (1).

Также как и ранее здесь на основании (k – 1)-элементных наборов находятся кандидаты в часто встречающиеся k-элементные наборы. Далее для найденных кандидатов строится хэш-дерево и с помощью дерева производится подсчет поддержки каждого набора. В Apriori для подсчета поддержки каждого набора последовательно рассматривался каждый кортеж и по хэш-дереву находились кандидаты, поддерживаемые данным кортежем. В предлагаемом нами решении последовательно рассматривается отображения каждого суботношения r_i. Поиск всех кандидатов поддерживаемых в данном суботношении r_i выполняется по его отображению следующим образом. Пусть N – корень хэш-дерева. Если N является листом (т.е. содержит ссылки на кандидатов), то ищем поддержку каждого кандидата упомянутого в N. Иначе (т.е. если N содержит ссылки не на кандидатов, а на другие узлы дерева) для каждого атрибута r_i количество единиц в котором больше нуля ищется узел дерева, на который необходимо спуститься. Описанная выше процедура выполняется рекурсивно, до тех пор, пока не будет рассчитана поддержка каждого кандидата из L_k присутствующего в r_i. Следует отметить, что как при расчете поддержки кандидата, так и при прохождении по узлам дерева выполняется операция поиска общих элементов для некоторого множества атрибутов r_i представленных в отображении, которая, в свою очередь, сводится к многократному применению операции поиска общих элементов в двух атрибутах r_i. Учитывая тот факт, что все элементы каждого атрибута r_i в отображении хранятся как отсортированная последовательность значений функции h(j) в худшем случае алгоритмическая сложность выполнения операции CMP_e сравнения двух таких последовательностей с целью нахождения общих элементов будет равна $O(cnt_1 + cnt_2)$, где cnt_1, cnt_2 – количество элементов в первой и второй последовательностях соответственно. Такой подход к выполнению этой операции в соответствии с утверждением 1 обеспечивает «точный» расчет поддержки каждого кандидата. В то же время иногда «точный» расчет не является необходимым и возможно применение «грубого» расчета:

$$CMP_r(vw_1, vw_2) = \begin{cases} CMP_e(vw_1, vw_2), \text{ если } F_r(vw_1, vw_2) = 0; \\ vw_1, \text{ если } F_r(vw_1, vw_2) = 1, \end{cases} \quad (5)$$

где CMP_r, CMP_e – обозначения операций «точного» и «грубого» поиска общих элементов в двух атрибутах из r_i представленных в отображении как vw_1, vw_2; $Fr(vw_1, vw_2) = \{((cnt_1 = cnt_2) \wedge (H(w_1) = H(w_2)))\}$. Очевидно, что применение операции CMP_r не приводит к появлению ошибок 1-го рода. Вместе с тем следует оценить вероятность появления ошибок 2-го рода. В качестве оценочной характеристики может служить вероятность возникновения ситуации, когда $Fr(vw_1, vw_2) = 1$ и $vw_1 \neq vw_2$.

Введем следующие утверждения.

Утверждение 3. Число $N_p(j)$ кодовых слов веса j кода Хэмминга таких, что все проверочные разряды равны 0 не больше чем коэффициент многочлена

$$f(x) = \frac{1}{u+m+1}\left((1+x)^{u+m} + (u+m)(1+x)^{\frac{u+m-1}{2}}(1-x)^{\frac{u+m+1}{2}}\right),$$

стоящий при x^j.

Утверждение 4. Количество случаев, когда две неравные цепочки w1, w2, содержащих a единиц каждая будут иметь равные H(w) равно

$$B(a) = \sum_{i=0}^{a-2} C^i_{u-2*(a-i)} * C^2_{2*(a-i)} * N_p(2*(a-i)). \quad (6)$$

Теперь представляется возможным получить вероятность возникновения ситуации, когда $Fr(vw_1, vw_2) = 1$ и $vw_1 \neq vw_2$ по следующей формуле

$$p_n(a) = \frac{2B(a)}{C^a_u(C^a_u - 1)}. \quad (7)$$

Результаты, получаемые с помощью (7) убедительно показывают, что вероятность неверного расчета поддержки набора в «суботношении» невелика и, следовательно, количество ошибок 2-го рода при поиске часто встречающихся наборов с помощью CMP_r также будет невелико, таким образом, приходим к выводу, что предлагаемое отображение удовлетворяет 4-му выдвинутому требованию.

Для оценки быстродействия предложенного метода поиска в памяти часто встречающихся наборов по отображению относительно метода предложенного в [3] было проведено ряд экспериментов на основе реальных данных о сотрудниках университета «ХАИ». Все эксперименты проводились на персональном компьютере с процессором CELERON 566 МГц и 196 Мб ОЗУ. Операционная система – Windows NT 4 Server, компилятор – Borland Delphi 6. Объем исходных данных составил 978 Кб. Объем полученного отображения составил 105 Кб. Результаты исследования упомянутых методов представлены в табл. 1.

Таблица 1

Результаты исследования

Поддержка, %	«точный» поиск в отображении, мс	«грубый» поиск в отображении, мс	Apriori поиск в отношении, мс
50	260	230	922
40	561	490	1803
20	1582	1432	4697
10	3886	3635	11096
5	24485	10976	44114

Анализ полученных результатов позволяет сделать заключение о превосходстве по быстродействию предлагаемых методов и, следовательно, о том, что отображение отвечает 3-му требованию.

4. Предлагаемый метод поиска ассоциативных правил

Предлагаемое решение базируется на идее показанной в [6], и также предполагает два прохода по данным при поиске часто встречающихся наборов. Однако, в отличие от [6] поиск производится по отображению. Рассмотрим предлагаемый метод в виде последовательности выполняемых шагов.

На первом шаге уже существующее отображение разбивается на части, помещающиеся по объему в память. В каждой из частей ищутся часто встречающиеся наборы элементов. Поскольку на этом этапе мы получаем только кандидатов в часто встречающиеся в R наборы, при расчете поддержки наборов используется «грубый» метод. После выполнения первого шага мы получаем множество кандидатов в часто встречающиеся в R наборы, каждый из которых является таковым хотя бы в одной из частей.

На втором шаге за один проход по отображению, используя «точный» метод, рассчитываем поддержку каждого из полученных ранее кандидатов, получая в результате множество часто встречающихся наборов.

Наконец, на третьем шаге находим по множеству $L = \{L_1, L_2,\ldots, L_k\}$ как и в [3] (2) находим все ассоциативные правила поддержка и достоверность которых не меньше некоторых заранее заданных s_{min} и c_{min} соответственно.

Количество «дорогих» I/O операций в данном случае будет соответствовать количеству операций необходимых для последовательного прочтения 2 * Vo(R) бит из внешней памяти. При использовании метода Partition количество таких операций будет определяться необходимостью прочтения 2 * V(R) бит из внешней памяти. Учитывая вышеприведенные оценки, а также тот факт, что методы для работы с отображением не уступают по быстродействию методам, применявшимся для работы с исходным отношением, можно сделать вывод о превосходстве по быстродействию предлагаемого метода над Partition в $\frac{2*V(R)}{2*V_o(R)} = K_{сж}$ раз.

Экспериментальные исследования полученного метода. Экспериментальные исследования описанного метода были проведены на основе отношения, содержащего данные о 1596849 клиентах одной из украинских компаний. Объем исходных данных составил 7,3 Гб. Объем полученного отображения составил 483 Мб. Все эксперименты проводились на персональном компьютере с процессором AMD Athlon XP 1700 и 768 Мб ОЗУ. Операционная система – Windows XP Professional, компилятор – Borland Delphi 6. Результаты экспериментальных исследований описанного метода проиллюстрированы на рис. 4.

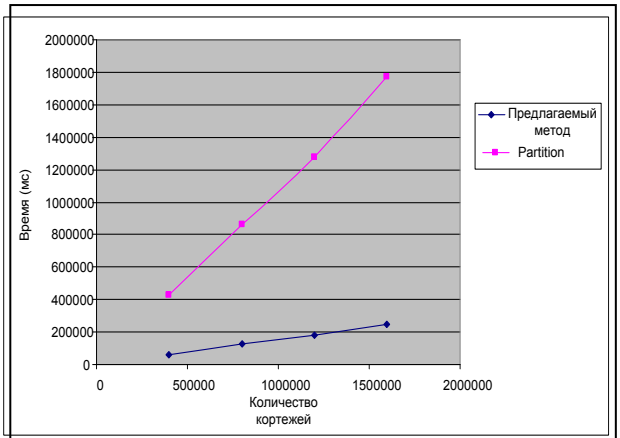

Рис. 4. Результаты экспериментальных исследований метода поиска ассоциативных правил

Таким образом, полученные результаты подтвердили сделанные ранее аналитические оценки и продемонстрировали превосходство предлагаемого метода относительно Partition для выбранного набора данных в 7 раз.

Заключение

Таким образом, за счет изменения традиционного подхода к хранению данных был получен эффективный метод поиска ассоциативных правил. Доказаны утверждения о кодировании исходных данных «без потерь». Показано, что при распространенной ситуации низкой плотности единиц, возможно, достичь существенного сокращения объема детализированных данных. Результаты экспериментов позволяют судить о высоком быстродействии предложенного метода в сравнении с другими подходами.

Список литературы

1. *Agrawal R., Imielinski T., Swami A. Database Mining: A Performance Perspective // IEEE Transactions on Knowledge and Data Engineering. – December 1993. – P. 914-925.*
2. *Agrawal R., Imielinski T., Swami A. Mining association rules between sets of items in large databases // Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93). – May 1993. – P. 207-216.*
3. *Agrawal R., Srikant R. Fast Algorithms for Mining Association Rules // Proc. 20th Int. Conf. Very Large Data Bases, 1994. – P. 487-499.*
4. *Cohen E., Datar M., Fujiwara S., Gionis A., Indyk P., Motwani R., Ullman J.D., Yang C. Finding interesting associations without support pruning // IEEE Transactions on Knowledge and Data Engineering. – January 2001. – P. 64-78.*
5. *Maletic J., Marcus A. Data Cleansing: Beyond Integrity Analysis // Proceedings of The Conference on Information Quality (IQ2000). – Massachusetts Institute of Technology, Boston, MA, USA. – 2000. – P. 200-209.*
6. *Savasere A., Omiecinski E. and Navathe S. An efficient algorithm for mining association rules in large databases // Proceedings 21st International Conference on Very Large Data Bases (VLDB'95). – Zurich, Switzerland, 1995. – P. 432-444,*
7. *Park J.S., Chen M.S., Yu P.S. An effective hash-based algorithm for mining association rules // New York: SIGMOD ACM Press, 1995. – P. 175-186.*
8. *Toivonen H. Sampling Large Databases for Association Rules. In Proceedings of the 22th International Conference on Very Large Data Bases. – Morgan Kaufmann Publishers, 1996. – P. 134-145***,**
9. *Дейт К. Дж. Введение в системы баз данных. – М.: Издательский дом «Вильямс», 2001. – 1072 с.*
10. *Морелос-Сарагоса Р. Искусство помехоустойчивого кодирования. – М.: Техносфера, 2005. – 320 с.*

Поступила в редакцию 11.07.2006

Рецензент: д-р техн. наук, проф. А.С. Кулик, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.