Годишник на секция "Информатика" **Annual of "Informatics" Section**
Съюз на учените в България **Union of Scientists in Bulgaria**
Том 6, 2013, 46–53 **Volume 6, 2013, 46–53**

ОБОБЩЕНОМРЕЖОВ МОДЕЛ НА ПРОЦЕСА НА ИЗВЛИЧАНЕ НА АСОЦИАТИВНИ ПРАВИЛА ЧРЕЗ FREQUENT PATTERN-GROWTH АЛГОРИТЪМ

Веселина Бурева[1], Евдокия Сотирова[1], Красимир Атанасов[1,2]

[1] Университет „Проф. д-р Асен Златаров", гр. Бургас
[2] Институт по биофизика и биомедицинско инженерство, БАН
e-mail: esotirova@btu.bg, vesito_ka@abv.bg, krat@bas.bg

Резюме: Чрез извличането на знания се откриват предварително недефинирани зависимости и скрити закономерности в големи масиви и потоци от данни. За целта се използват различни техники, например асоциативни правила, дървета на решенията, невронни мрежи, клъстерен анализ [4]. В настоящата работа се разглежда генерирането на асоциативни правила чрез алгоритъм *Frequent Pattern-Growth,* който компресира максимално входните данни. За моделирането на процеса е използван апарата на обобщените мрежи.
Ключови думи: Обобщени мрежи, Асоциативен анализ, Извличане на знания от данни

1. Въведение

Асоциативните правила представят зависимостите между елементите в едно множество. Те се записват във формата $A \Rightarrow B$, където A и B са два независими елемента. Най-използваните критерии за избор на асоциативни правила са *подкрепа (support), доверие (confidence)* и *лифт (lift)* [5]:

$$support\ (A \Rightarrow B) = P(A \cup B)$$

$$confidence\ (A \Rightarrow B) = P(B|A) = \frac{P(A \cup B)}{P(A)}$$

$$lift\ (A \Rightarrow B) = \frac{confidence(A \Rightarrow B)}{P(B)} = \frac{P(A \cup B)}{P(A)P(B)}$$

Алгоритъмът, конструиращ *Frequent pattern tree*, не се нуждае от създаването на всички възможни кандидат-множества на елементите, за да извлече асоциативните

правила. Той представя входните данни в компресиран вид [3]. Конструира се чрез сканиране на транзакциите една по една и записването им като пътища в дървото. За конструирането му са необходими само две преминавания през данните. Алгоритъмът обработва дървото по рекурсивен начин. При нанасянето на транзакциите по дървото, ако са налични едни и същи елементи в транзакциите, то техните пътища се припокриват. По този начин се получава по-кратко представяне на входната информация.

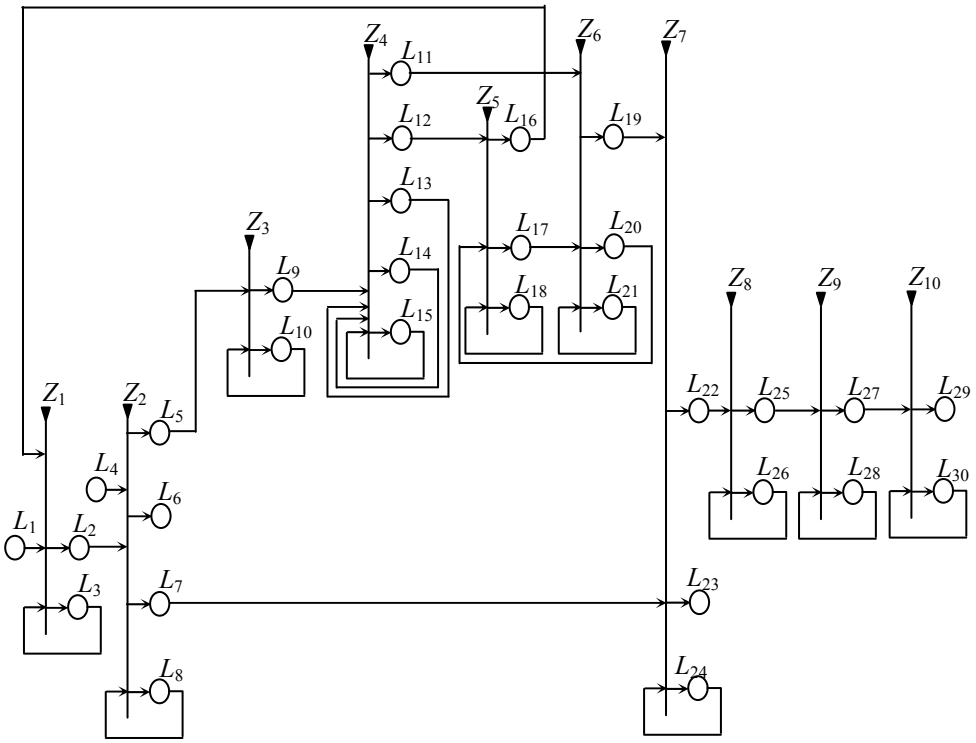
2. Обобщена мрежа на процеса на създаване на асоциативни правила

Теорията на обобщените мрежи е описана в [1, 2]. Чрез нея са конструирани редица модели от областта на извличането на знания от данни (Data Mining) [6, 7, 8, 9, 10]. Обобщеномрежовият модел, описващ процеса на генериране на асоциативни правила чрез *Frequent Pattern-Growth*, е показан на Фиг. 1. Алгоритъмът конструира *Frequent Pattern Tree* (*Fp-Tree*), като значително намалява обема на входните данни. Конструирането на *Fp-Tree* е разделено на три по-важни стъпки:

1. Сканират се входните данни, за да се определи поддръжката за всеки елемент, отхвърлят се нечестите елементи, а честите се сортират в низходящ ред (във вид на списък);
2. За втори път се сканират входните данни, като на тази стъпка транзакциите се нанасят една по една като пътища на *Fp-Tree* като:
    - Сортират се елементите в транзакцията в съответствие със списъка с честите елементи;
    - Ако има възел, съответстващ на обработвания елемент, тогава се увеличава броячът на възела и се изгражда пътят;
    - Ако няма такъв възел (уникална транзакция), се формира нов и се задава брояч със стойност 1.
3. Това се повтаря докато всяка транзакция бъде изобразена на дървото.

Възлите на *Fp-Tree* отговарят на елементите от транзакциите и имат броячи, равни на броя срещания на елемента в транзакциите. Възможно е дървото да има свързващи връзки, които посочват еднаквите елементи в различните пътища. Колкото повече пътища се припокриват, толкова по-висока е компресията на данните. *Fp-Growth* извлича честите елементи "отдолу-нагоре" по дървото. Първоначално се извличат пътищата до поддърветата, завършващи на определен елемент. Всеки извлечен път се обработва рекурсивно до извличането на честите единични елементи. От дървото с пътищата се съставя условно дърво за елемента, като за целта се актуализира подкрепата на елементите и съответно брояча на възлите. Ако са налични пътища с елементи, неудовлетворяващи вече тази подкрепа, то те се изрязват. Премахват се и възлите съдържащи определения елемент. От условното дърво се извличат честите подмножества на елемента.

Обобщената мрежа има 10 прехода и 30 позиции (Фиг. 1). Преходите описват следните процеси:



Фиг. 1 Обобщеномрежов модел, описващ генерирането на асоциативни правила чрез алгоритъм *Fp-Tree*

- Z_1 – постъпване на транзакции в хранилището,
- Z_2 – задаване на поддръжка (честота),
- Z_3 – сортиране на елементите в транзакциите по низходящ ред,
- Z_4 – конструиране на *Fp-Tree*,
- Z_5 – обхождане на *Fp-Tree* "отдолу - нагоре",
- Z_6 – съставяне на дърво с възможните пътища до елемент,
- Z_7 – записване на списък с пътищата до елемент (*F-List*),
- Z_8 – съставяне на условно дърво (*Fp-Growth*),
- Z_9 – рекурсивно извличане на честите подмножества от условното дърво (*Fp-Growth*),
- Z_{10} – запис на честите елементи и техните подмножества.

Първоначално в обобщената мрежа в позиция L_3 има α_1-ядро с характеристика „*Хранилище с данни*“. То ще стои там по време на работата на обобщената мрежа, като в определени времеви моменти може да генерира нови α-ядра, които ще се придвижат до прехода Z_2.

През позиция L_1 в обобщената мрежа постъпват α_0-ядра с характеристика „*Транзакции*“.

Преходът Z_1 има следната форма:

$$Z_1 = \langle \{L_1, L_3, L_{16}\}, \{L_2, L_3\}, R_1, \vee(L_1, L_3, L_{16})\rangle,$$

където:

$$R_1 = \begin{array}{c|cc} & L_2 & L_3 \\ \hline L_1 & false & true \\ L_3 & W_{3,2} & true \\ L_{16} & false & true \end{array},$$

и $W_{3,2}$ = „определени са транзакции за намиране на чести и нечести елементи“.

α_0-ядрата, постъпващи от позиция L_1 се слива с α_1-ядрото в позиция L_3. α_1-ядрото в позиция L_3 генерира ново α_2-ядро, което постъпва в позиция L_2 с характеристика: „*Транзакции за намиране на чести (удовлетворяващи поддръжката) и нечести (неудовлетворяващи поддръжката) елементи*“.

През позиция L_4 в обобщената мрежа постъпва β-ядро с характеристика „*поддръжка (честота) за транзакции*“.

Преходът Z_2 има следната форма:

$$Z_2 = \langle \{L_2, L_4, L_8\}, \{L_5, L_6, L_7, L_8\}, R_2, \vee(\wedge(L_2, L_4), L_8)\rangle,$$

където:

$$R_2 = \begin{array}{c|cccc} & L_5 & L_6 & L_7 & L_8 \\ \hline L_2 & false & false & false & true \\ L_4 & false & false & false & true \\ L_8 & W_{8,5} & W_{8,6} & W_{8,7} & W_{8,8} \end{array},$$

- $W_{8,5}$ = „определени са честите елементи за сортиране в транзакциите“;
- $W_{8,6}$ = „определени са елементите, неудовлетворили поддръжката (нечестите елементи)“;
- $W_{8,7}$ = „определени са елементите, удовлетворили поддръжката (честите елементи) за проверка при записване на списъка с пътищата до чест елемент (F-list)“;
- $W_{8,8} = \neg(W_{8,5} \wedge W_{8,6} \wedge W_{8,7})$.

α_2- и β-ядрата, постъпващи от позиции L_2 и L_4 не получават нови характеристики в позиция L_8. α_3-ядрото в позиция L_8 генерира α_4-, α_5- и α_6-ядра, които постъпват в позиции L_5, L_6 и L_7 с характеристики, съответно: „*Чести елементи (елементи, удовлетворили минималната поддръжка)*“ в позиция L_5, „*Нечести елементи (елементи, неудовлетворили минималната поддръжка)*“ в позиция L_6, и „*Чести елементи (елементи, удовлетворили минималната поддръжка) за проверка при записване на списъка с пътищата до чест елемент (F-list)*“ в позиция L_7.

Преходът Z_3 има следния вид:

$$Z_3 = \langle \{L_5, L_{10}\}, \{L_9, L_{10}\}, R_3, \vee(L_5, L_{10})\rangle,$$

където:

$$R_3 = \begin{array}{c|cc} & L_9 & L_{10} \\ \hline L_5 & \mathit{false} & \mathit{true} \\ L_{10} & W_{10,9} & W_{10,10} \end{array},$$

- $W_{10,9}$ = „сортирани са елементи, удовлетворили минималната подкрепа в низходящ ред“;
- $W_{10,10} = \neg W_{10,9}$.

α_4-ядрото (от позиция L_5) не получава нова характеристика в позиция L_{10}. То генерира α_5-ядро, което постъпва в позиция L_9 с характеристика „*Сортирани елементи (удовлетворили минималната поддръжка) в низходящ ред*“.

Преходът Z_4 има следната форма:

$$Z_4 = \langle \{L_9, L_{13}, L_{14}, L_{15}\}, \{L_{11}, L_{12}, L_{13}, L_{14}, L_{15}\}, R_4, \vee(L_9, L_{13}, L_{14}, L_{15})\rangle,$$

където:

$$R_4 = \begin{array}{c|ccccc} & L_{11} & L_{12} & L_{13} & L_{14} & L_{15} \\ \hline L_9 & \mathit{false} & \mathit{false} & \mathit{false} & \mathit{false} & \mathit{true} \\ L_{13} & \mathit{false} & \mathit{false} & \mathit{false} & \mathit{false} & \mathit{true} \\ L_{14} & \mathit{false} & \mathit{false} & \mathit{false} & \mathit{false} & \mathit{true} \\ L_{15} & W_{15,11} & W_{15,12} & W_{15,13} & W_{15,14} & W_{15,15} \end{array},$$

- $W_{15,11}$ = $W_{15,12}$ = „Съставено е *FP-tree*“;
- $W_{15,13}$ = „построен е пътя за транзакция, увеличен е брояча и са построени връзките между еднаквите обекти“;
- $W_{15,14}$ = „позицията L_{15} не е празна и е построен главен възел Null“;
- $W_{15,15} = \neg W_{15,11}$.

При първото активиране на преход Z_4 в позиция L_{15} (от позиция L_9) постъпва α_5-ядро без да получава нова характеристика. При следващото активиране на прехода Z_4 α_5-ядрото в позиция L_{15} генерира α_6-ядро, постъпващо в позиция L_{14} с характеристика „*Главен възел Null*“.

При следващото активиране на прехода Z_4 α_5-ядрото от позиция L_{15} генерира α_7-ядро, постъпващо в позиция L_{13} с характеристика „*Построен път за транзакция, увеличен брояч, създадени връзки между еднаквите елементи*“.

При следващото активиране на прехода Z_4 α_5-ядрото от позиция L_{15} генерира α_8- и α_9-ядра, които постъпват в позиции L_{11} и L_{12} с характеристика „*FP-tree*“.

Преходът Z_5 има следната форма:

$$Z_5 = \langle \{L_{12}, L_{18}, L_{20}\}, \{L_{16}, L_{17}, L_{18}\}, R_5, \vee(L_{12}, L_{18}, L_{20})\rangle,$$

където:

$$R_5 = \begin{array}{c|ccc} & L_{16} & L_{17} & L_{18} \\ \hline L_{12} & \mathit{false} & \mathit{false} & \mathit{true} \\ L_{18} & W_{18,16} & W_{18,17} & W_{20,18} \\ L_{20} & \mathit{false} & \mathit{false} & \mathit{true} \end{array},$$

- $W_{18,16}$ = „Списъкът с елементи на текущото *FP-tree* е празен“;
- $W_{18,17}$ = „Избран е елемент“;
- $W_{19,19} = \neg W_{19,16}$.

α_8-ядрото, постъпващо в позиция L_{18}, не получава нова характеристика. То генерира α_{10}- и α_{11}-ядра, постъпващи в позиции L_{16} и L_{17} със следните характеристики: „*Връщане към хранилището с данни*“ в позиция L_{16}, и „*Избран елемент*“ в позиция L_{17}.

Преходът Z_6 има следната форма:

$$Z_6 = \langle \{L_{11}, L_{17}, L_{21}\}, \{L_{19}, L_{20}, L_{21}\}, R_6, \vee(L_{11}, L_{17}, L_{21})\rangle,$$

където:

$$R_6 = \begin{array}{c|ccc} & L_{19} & L_{20} & L_{21} \\ \hline L_{11} & false & false & true \\ L_{17} & false & false & true \\ L_{21} & W_{21,19} & W_{21,20} & W_{21,21} \end{array},$$

- $W_{21,19}$ = „съставено е дървото с възможните пътища за избрания елемент“;
- $W_{21,20}$ = „необходим е избор на следващ елемент“;
- $W_{21,21} = \neg W_{21,19}$.

α_9-ядрото, постъпващо в позиция L_{21}, не получава нова характеристика. То генерира α_{12}- и α_{13}-ядра, постъпващи в позиции L_{19} и L_{20} със следните характеристики: „*Дърво с пътищата за избрания елемент*“ в позиция L_{19} и „*Избор на следващ елемент*“ в позиция L_{20}.

Преходът Z_7 има следната форма:

$$Z_7 = \langle \{L_{19}, L_7, L_{24}\}, \{L_{22}, L_{23}, L_{24}\}, R_7, \vee(L_{24}, \wedge(L_7, L_{19}))\rangle,$$

където:

$$R_7 = \begin{array}{c|ccc} & L_{22} & L_{23} & L_{24} \\ \hline L_7 & false & false & true \\ L_{19} & false & false & true \\ L_{24} & W_{24,22} & W_{24,23} & W_{24,24} \end{array},$$

- $W_{24,22}$ = „Елементът удовлетворява минималната поддръжка“;
- $W_{24,23} = \neg W_{24,22}$;
- $W_{24,24}$ = „има още елементи за проверка на поддръжката“.

α_6- и α_{12}-ядрата, постъпващи в позиция L_{24} се сливат в α_{14}-ядро с характеристика „*Главен възел Null, дърво с пътищата за избрания елемент*“.

α_{14}-ядрото генерира α_{15}- и α_{16}-ядра, постъпващи в позиции L_{22} и L_{23} със следните характеристики: „*Елемент, удовлетворяващ минималната поддръжка (чест елемент)*“ в позиция L_{22} и „*Елемент, неудовлетворяващ минималната поддръжка (нечест елемент)*“ в позиция L_{23}.

Преходът Z_8 има следната форма:

$$Z_8 = \langle \{L_{22}, L_{26}\}, \{L_{25}, L_{26}\}, R_8, \vee(L_{22}, L_{26}))\rangle,$$

където:

$$R_8 = \begin{array}{c|cc} & L_{25} & L_{26} \\ \hline L_{22} & false & true \\ L_{26} & W_{26,25} & W_{26,26} \end{array},$$

- $W_{26,25}$ = „съставено е условно дърво за елемент“;
- $W_{26,26} = \neg W_{26,25}$.

α_{15}-ядрото, постъпващо в позиция L_{26}, не получава нова характеристика. То генерира α_{16}-ядро, постъпващо в позиция L_{25} с характеристика „*Условно дърво за елемент*“.

Преходът Z_9 има следната форма:

$$Z_9 = \langle \{L_{25}, L_{28}\}, \{L_{27}, L_{28}\}, R_9, \vee(L_{25}, L_{28}))\rangle,$$

където:

$$R_9 = \begin{array}{c|cc} & L_{27} & L_{28} \\ \hline L_{25} & false & true \\ L_{28} & W_{28,27} & W_{28,28} \end{array},$$

- $W_{28,27}$ = „извлечени са подмножествата на чест елемент“;
- $W_{28,28} = \neg W_{28,27}$.

α_{16}-ядрото, постъпващо в позиция L_{28}, не получава нова характеристика. То генерира α_{17}-ядро, постъпващо в позиция L_{27} с характеристика „*Подмножества на чест елемент*“.

Преходът Z_{10} има следната форма:

$$Z_{10} = \langle \{L_{27}, L_{30}\}, \{L_{29}, L_{30}\}, R_{10}, \vee(L_{27}, L_{30}))\rangle,$$

където:

$$R_{10} = \begin{array}{c|cc} & L_{29} & L_{30} \\ \hline L_{27} & false & true \\ L_{30} & W_{30,29} & W_{30,30} \end{array},$$

- $W_{30,30}$ = „има необработени елементи“;
- $W_{30,29} = \neg W_{30,30}$.

α_{17}-ядрото, постъпващо в позиция L_{30}, не получава нова характеристика. То генерира α_{18}-ядро, постъпващо в позиция L_{29} с характеристика „*Асоциативно правило*“.

3. Заключение

Обобщеномрежовият модел за процеса на съставяне на асоциативни правила чрез метода *Frequent Pattern-Growth* е вторият, който е разработен след модела за извличане на асоциативни правила чрез алгоритъм *Apriori* [7]. Предимството от генерирането на асоциативни правила чрез *FP- Growth* е неговото бързодействие, тъй като не се нуждае от съставяне на всички възможни кандидати (кандидат-множества на елементите). Недостатък на алгоритъма е необходимостта от голямо количество памет, поради рекурсивното обхождане на дървото.

Литература

[1] Atanassov, K. *Generalized Nets*, World Scientific. Singapore, 1991.

[2] Atanassov, K. *On Generalized Nets Theory*, “Prof. M. Drinov” Academic Publishing House, Sofia, 2007.

[3] Kantardzic, M. *Data Mining: Concepts, Models, Methods, and Algorithms*, John Wiley & Sons, 2003.

[4] Sumathi, S., S. N. Sivanandam. *Introduction to Data Mining and Its Applications*, Springer, 2006.

[5] Zhao Y., *R and Data Mining: Examples and Case Studies 1*, Elsevier Inc., 2012.

[6] Bureva, V., P. Chountas, K. Atanassov, A generalized net model of the process of decision tree construction, *Proc. of 13th Int. Workshop on Generalized Nets*, London, 29 October 2012, 1–7.

[7] Бурева В., Обобщеномрежов модел на процеса на създаване на асоциативни правила, *Годишник на секция "Информатика", Съюз на учените в България*, Том 5, 2012, 73–83.

[8] Orozova, D., E. Sotirova, Generalized net model of the applying data mining tools, *Proc. of the Tenth International Workshop on Generalized Nets*, Sofia, 2009, 22–26.

[9] Sotirova, E., D. Orozova, Generalized net model of the phases of the data mining process, *Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics. Vol. II: Applications*, Warsaw, Poland, 2010, 247–260.

[10] Sotirova. E., K. Dimitrova, R. Papancheva, A Generalized Net Model for Analysis of a Student's Evaluations by Data Mining Techniquesmin the e-Learning university, *Proc. of the Tenth International Workshop on Generalized Nets*, Sofia, 2009, 41–46.