CHAIN SPLIT AND COMPUTATIONS IN PRACTICAL RULE MINING

Levon Aslanyan, Hasmik Sahakyan

Abstract: A novel association rule mining algorithm is composed, using the unit cube chain decomposition structures introduced in [HAN, 1966; TON, 1976]. [HAN, 1966] established the chain split theory. [TON, 1976] invented an excellent chain computation framework which brings chain split into the practical domain. We integrate these technologies around the rule mining procedures. Effectiveness is related to the intention of low complexity of rules mined. Complexity of the procedure composed is complementary to the known Apriori algorithm which is defacto standard in rule mining area.

Keywords: Data mining, unite cube.

ACM Classification Keywords: 1.5. Pattern recognition, H.2.8 Database applications, Data mining.

Conference: The paper is selected from International Conference "Classification, Forecasting, Data Mining" CFDM 2009, Varna, Bulgaria, June-July 2009

Introduction

Association rule mining (ARM) is a part of data mining theory. Data Mining is known as a non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns or knowledge in data. Existing algorithms are complex computationally, and efficency vs. accuracy issue of algorithms is still open. In association rule mining rules are logical implications of the form $X \to Y$. The mining problem is to generate all implications that have several property estimates greater than the user specified minimum. One of the most used algorithms is *Apriori [KOT, 2006].*

Let we are given a set $I = \{x_1,...,x_n\}$ of n different items. $X \subseteq I$ is itemset and X is k-itemset when $|X| = k$. Given a database D with records (transaction, itemset), and we say that $T \in D$ supports X, if $X \subseteq T$. We consider the standard concepts of support and confidence

$$supp(X) = |\{T \in D \mid X \subseteq T\}| / |D|, \text{ and}$$

$$conf(X \to Y) = supp(X \cup Y) / supp(X).$$

As a rule ARM processes the rule mining in 2 tasks; first is to find frequent subsets (that have transaction support above minimum) and second one is to generate association rules themselves. Several ARM construct the frequent subsets by growing. Theoretically, while growing, it constructs not only the maximal elements of hierarchy but may also construct all their subsets. An alternative approach to accelerate the rule mining is considered in this paper, intending to implement the known research results on n-cube geomrtry and algorithmic recognition of Monotone Boolean Functions to the rule mining area.

Constrained Monotone Boolean Reconstruction

ARM, and its frequent subsets generation (FSG) stage in particular can be described in terms of Monotone Boolean functions. Consider unit cube B^n of dimension n which consists of all binary n-vectors. We apply to n-cube geometry terms – layer, neighbor vertices, chain, etc. [AS, 1979]. Each cube vertex

$\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$ can be viewed as transaction, where $\alpha_{i_j} = 1$ indicates that item i_j involved in the transaction, otherwise $\alpha_{i_j} = 0$. Boolean Function, formed in this way, equals 0 if the vertex-subset is frequent, and 1, if not.

Practically all frequent subsets in a typical application problem are placed on very low layers of B^n. We may suppose that a value k is known so that all frequent subsets belong to layers lower than k. A different question is how precise is the known boundary k, but being given k, FSG applies for effective solutions of reconstruction of monotone Boolean functions with 0' below the k-th layer. Regular ARM starts work from the 0 layer and continues it till some k-th layer. The alternative way of solving FSG uses chain split of B^n. It is that B^n below the k-th layer can be split into the C_n^k disjoint chains providing some special characteristics [HAN, 1966]. Chains are related through the property of conditional complements and as consequence - if values of function are known on $n-2p-1 (0 \le p \le \lfloor n/2 \rfloor)$ chains then applying monotonity on n-2p+1 chains we receive on them at most 2 new undetermined vertices of function. Second valuable component that enforced our algorithm is that the chain system allows calculations in virtue without archiving and search over the chains [TON, 1976]. Being short we formulate a typical result and then explain the algorithm informally:

Theorem 1. Minimal number $\varphi(n)$ of "example" type operations required for recognizing arbitrary monotone Boolean functions $f(x_1, x_2, ..., x_n)$ with 0's only below the k-th layer, $0 \le k \le \lfloor n/2 \rfloor$, equals

$$C_n^k + C_n^{k-1}.$$

The statement considers a specifically constrained set of Boolean functions achieving in this way more precise and lower estimate for complexity of reconstructing algorithms. Theoretically, the use of this concept requires the set of all C_n^k chains of considered area of B^n, which is computationally hard, requiring large memory areas and recursions. Resolving this trade off we engage the [TON, 1976] approach which does not require to keep the chains in memory and calculating, instead, the necessary information having the vertex given by its coordinates.

Chain Computation Algorithm

This part explains the FSG tasks of ARM by the chain technique. The memory and computational resource reductions as mentioned are the results achieved. If the base algorithm - *Apriori* may require $C_n^k + C_n^{k-1} + ... + C_n^0$ steps to restore the Constrained Monotone Boolean Function, then the steps required by alternative algorithm will be not grater than $C_n^k + C_n^{k-1}$.

Large data volumes which appear in data mining applications require low computaional algorithms for composite optimisation problems where data mining is the recurrent task of the total algorithmic solution. *Apriori* alternative algorithm by this work uses the specific theoretical know-how which reduces required computations. The system is developed and applied in solving pratical problems – network intrusion detection by LOG records of application software systems is an example of applications.

Now let us stay on description of chain computation framework. Imagine a set of vertical chains connected to each other through the special set of horizontal passes through sets of vertices. These are the chains splitting B^n. The procedure working on this set of chains produces a knowledge system which finally becomes the result of algorithm. In our case this will be partial values of function on chains. Vertices in which function is yet unknown might occupy some middle intervals of chains because of monotonity. This structure in its size is smaller than the considered area of B^n with its chain split. We intend to generate the same resulting knowledge by computations

which involve the chain split elements and their coordinates. The rules about chains and passes properties are also simply applied. This work style guarantees the minimal possible memory use of algorithm. Chain computation on considered area of B^n is through the following set of procedures:

(1) computation of the consecutive number of a given vertex on its chain;
(2) computation of consecutive numbers of all neighbor vertices for the given one;
(3) characterization of chain lengths adjoint to the neighbor vertices for the given one;
(4) computation of consecutive number of the next upper vertex to the given one on its chain;
(5) computation of consecutive number of the next vertex below to the given one on its chain;
(6) enumeration of all minimal vertices of all chains of given length;
(7) enumeration of all maximal vertices of all chains of given length;
(8) computation of conditional compliment and its parameters;
(9) computation of all down neighbor vertices to the given one.

It is to mention that the set (1)-(9) is just one example set of chain computation style procedures. These are simple computational tasks. The scenario of FSG we consider is not the unique and several modifications and extensions are possible and useful concerning the application problem conditions. Discuss several characteristic fragments of chain computation rule mining algorithm by procedures (1) - (9).

In our approach it is important that the Boolean function describing itemset frequencies equals 1 above the layer k (the best estimate, given by applied problem). Consider all vertices of layer k. For each vertex compute the chain length passing through this vertex and the consecutive number of this vertex on its chain. Working instrument is (1) in this stage and let R_n denotes the chain split of B^n. Firstly, procedure computes some values $K_n(\tilde{\alpha})$ for each vertex $\tilde{\alpha}$ of k-th layer, and then (1) states that vertices $\tilde{\alpha} \in L, L \in R_n$ are $K_n(\tilde{\alpha})$-th consecutive vertices on their chains L. After this, length of chain L is computed taking into consideration properties of chain split. Described fragment is recursive part of total algorithm.

In a later stage, among the vertices of k-th layer we separate all those that are the last vertices of their chains. The chain length of all these vertices equals $n-k-k=n-2k=l$. Ask operator A_f ("example" operator) for the values of considered function on these vertices. After this we apply to the chains of length $l+2$ and extend the results received from A_f to these chains. Determination of all last vertices of the chains of length $l+2$ is by procedure (6), $R(n,l+2,l+3)=\{\tilde{\alpha} \in B^n \| \|\tilde{\alpha}\| = (n-l-2/2)$ and $\tilde{\alpha}$ obeys a property $C\}$, where C is a simple checkable property. Next is to apply $\tilde{\alpha}_{(+1)}$ (down by the chains) $l+2$ times to each vertex $\tilde{\alpha} \in R(n,l+2,l+3)$ which constructs the first vertex $\tilde{\beta}$ of an $l+2$ chain.

On the general step - a chain of length $l+m$ is considered. The first and last vertices of this chain are found and, then the first vertex of a chain of length $l+m-2$ is computed, and the same way the last vertex of this chain, which is the compliment to the pre-final vertex of the chain of length $l+m$. All the values for vertices of chains of length $l+m-2$ are known at this stage, and extension by monotonity to the chains of length $l+m$ and computation on reminder vertices by the operator A_f is to be applied.

Conclusion

Frequent subset generation is always based on computations on monotone Boolean functions. Monotone function domain is known as complex although the optimal algorithms of recognition are known. Monotone recognition in data mining appears with constrains, which helps to construct less complex tasks and the way to this is through a set of simple computational tasks on the chains mentioned above. The concepts were effectively implemented in intrusion detection analysis by the set of LOG files of applied software systems.

Acknowledgement

The paper is partially financed by the project ITHEA XXI of the Institute of Information Theories and Applications FOI ITHEA and the Consortium FOI Bulgaria. www.ithea.org, www.foibg.com.

Bibliography

[AS, 1979] L. Aslanyan. Isoperimetry problem and related extremal problems of discrete spaces, Problemy Kibernetiki, 36, pp. 85-126 (1976).

[HAN, 1966] G. Hansel. Sur le nombre des functions booleennes monotones de n variables, C.R. Acad. Sci. Paris, 262, serie A (1966), 1088.

[TON, 1976] G. P. Tonoyan. Chain decomposition of n dimensional unit cube and reconstruction of monotone Boolean functions, JVM&F, v. 19, No. 6 (1976), 1532-1542.

[KOT, 2006] S. Kotsiantis and D. Kanellopoulos. Association Rules Mining: A Recent Overview, GESTS International Transactions on Computer Science and Engineering, Vol.32 (1), 2006, pp. 71-82.

[AS, 2008] L. Aslanyan and R. Khachatryan. Association rule mining inforced by the chain decomposition of an n-cube, Mathematical Problems of Computer Science, XXX, 2008, ISSN 0131-4645.

Authors' Information

Levon Aslanyan – Head of Department, Institute for Informatics and Automation Problems, P.Sevak St. 1, Yerevan 14, Armenia, e-mail: lasl@sci.am

Hasmik Sahakyan – Leading Researcher, Institute for Informatics and Automation Problems, P.Sevak St. 1, Yerevan 14, Armenia, e-mail: hasmik@ipia.sci.am