# Searching for Pattern Rules

Guichong Li and Howard J. Hamilton
*Department of Computer Science, University of Regina,*
*Regina, SK, Canada, S4S 0A2.*
{*liguicho, hamilton*}*@cs.uregina.ca*

## Abstract

*We address the problem of finding a set of pattern rules, from a transaction dataset given a statistical metric. A new data structure, called an incrementally counting suffix tree* (*ICST*), *is proposed for online computation of estimates of the support of any pattern or itemset. Using an ICST, our approach directly generates a set of pattern rules by a single scan of the whole dataset in partitions without the generation of frequent itemsets. Non-redundant rules can be found by removing redundancies from the pattern rules. The PPMCR algorithm first finds pattern rules and then non-redundant rules by generating valid candidates while traversing the ICST. Experimental results show that the PPMCR algorithm can be used for efficiently mining fewer non-redundant rules.*

## 1. Introduction

Correlations are of wide interest in data analysis, because they automatically exclude relations with inadequate statistical significance [4]. An association rule with support higher than a user-specified threshold might be poorly correlated, while a correlation rule might have low support [4]. Poor correlation may occur even for optimal association rules [3]. Thus, mining correlations may be an attractive alternative to mining itemsets.

Previous approaches to tackle this problem include generating non-redundant rules [9], using log-linear models [1], and ranking correlation pairs in the TAPER method [13]. Contrast sets are meaningful itemsets if the corresponding deviations with respect to different groups hold [2]. A more general approach is to consider multi-correlations between itemsets and a single item.

Given a transaction dataset, an itemset $A$ is *correlated* with an itemset $B$ if and only if a suitable statistical test, such as the chi-square ($\chi^2$) test, rejects the null hypothesis of independence at a specified significance level, such as 95% confidence. Itemsets $A$ and $B$ can also be regarded as sets of binary variables. A correlation between $A$ and $B$ is called a *multi-correlation* if and only if $A$ is an itemset containing at least one item, $B$ is a *singleton itemset* (i.e., an itemset containing exactly one item), $A \cap B = \varnothing$, and $A$ is correlated with $B$. Elsewhere, multi-correlations are referred to as *multiple correlations*.

A multi-correlation between an itemset $A$ and a singleton itemset $B$ is denoted $A \Rightarrow B$, which is called a *multi-correlation rule*. In such a rule, $A$ is the *antecedent* and $B$ is the *consequent*. In a multi-correlation rule $A \Rightarrow B$, the correlation is *positive* if $P(AB) > P(A)P(B)$, the correlation is *negative* if $P(AB) < P(A)P(B)$, and $A$ and $B$ are *independent* if $P(AB) = P(A)P(B)$. The *multi-correlation problem* is to find a set of multi-correlation rules from a transaction dataset given a statistical metric.

Without loss of generality, we assume that for each transaction in a dataset, all items are sorted in lexicographical order. All items in the antecedent or consequent of a rule are sorted similarly. In a transaction, an itemset $A$ is a *contiguous pattern* (or simply *pattern*) in the context of a singleton itemset $B$ if the items in either $A$ or $A \cup B$ occur contiguously in the transaction. For example, in the transaction *GHJKL*, *GH* is a pattern in the context of *J, K*, or *L* because *GH* appears contiguously. As well, *GJ* is a pattern in the context of *H* because $A \cup B = GHJ$ occurs contiguously. On the other hand, *GJ* is not a pattern in the context of *K* in *GHJKL,* because neither *GJ* nor *GJK* occur contiguously.

A *pattern rule* $A \Rightarrow B$ holds in a transaction dataset if itemset $A$ (occurring as a pattern in the context of $B$) and the singleton itemset $B$ are positively correlated in terms of some statistical metric. For example, if the pattern *GJ* occurs so often in transactions with *H* that *GJ* and *H* are correlated according to the specified statistical metric and $P(GHJ) > P(GJ)P(H)$, then $GJ \Rightarrow H$ is a pattern rule. Given a transaction dataset and a statistical metric, the restricted multi-correlation problem is to find a set of pattern rules that hold in the dataset.

Proceedings of the Sixth International Conference on Data Mining (ICDM'06)
0-7695-2701-9/06 $20.00 © 2006 IEEE

We propose a new approach to solve this restricted problem, which has four noteworthy features compared to previous research work: (1) pattern rules are found without generating frequent itemsets, (2) by using a partitioning strategy, only a single scan of the dataset is made, (3) the support for a pattern is estimated in time quadratic to the length of an itemset by using an *incrementally counting suffix tree*, abbreviated ICST, and (4) redundancies among positive correlation rules are found and removed by heuristically browsing the ICST using a four-state strategy.

The remainder of this paper is organized as follows. The theoretical foundations of our approach are described in Section 2. In Section 3, we describe the PPMCR algorithm for finding all pattern rules and their non-redundant rules, and in Section 4, we present our experimental results. We draw conclusions and give suggestions for future work in Section 5.

## 2. Theoretical foundations

### 2.1 The $\chi^2$ test and multi-correlations

Given two binary variables $A$ and $B$ and a transaction dataset, their $\chi^2$ value can be simply calculated as follows [12]:

$$\chi^2 = \frac{N\left(P(AB) - P(A)P(B)\right)^2}{P(A)P(B)(1-P(A))(1-P(B))} \qquad (1)$$

where $N$ is the size of sample; $P(AB)$, $P(A)$, and $P(B)$ are the *observed* probabilities of $AB$ and $A$ and $B$, respectively; $P(A)$ and $P(B)$ are not equal to 0 or 1 to allow a safe derivation.

The antecedent $A$ of a multi-correlation $A \Rightarrow B$ can also be regarded as the conjunction of items. Statistical metrics for two categorical variables can be used for checking for multi-correlations [12].

Instead of measuring the strength of linear correlations [12], in this paper, we focus on the $\chi^2$ test for the multi-correlation problem.

We have the following lemma about multi-correlations.

**Lemma 1**. If $A' \Rightarrow B$ and $A \Rightarrow B$ are two multi-correlation rules, where $A' \subseteq A$, and $P(B \mid A') = P(B \mid A)$, then $\chi^2(A' \Rightarrow B) \geq \chi^2(A \Rightarrow B)$, with equality holding when $P(A') = P(A)$.

**Proof**: (omitted). □

More generally, we have the following theorem.

**Theorem 1**. If $r'$:$A' \Rightarrow B$ and $r$:$A \Rightarrow B$ are two multi-correlation rules, where $A' \subseteq A$, and $P(B \mid A') \geq P(B \mid A) > P(B)$, then $\chi^2(r') \geq \chi^2(r)$.

**Proof**: Follows directly from the proof of Lemma 1. □

We define subrules and superrules of a rule as follows.

**Definition 1**. Given two multi-correlation rules $r$: $A \Rightarrow B$ and $r'$: $A' \Rightarrow B$, where $A' \subseteq A$, we say $r'$: $A' \Rightarrow B$ is a *subrule* of $r$, denoted $r' \prec r$. Conversely, $r$ is a *superrule* of $r'$. □

Given two candidate rules $r_1$:$A_1 \Rightarrow B$ and $r_2$: $A_2 \Rightarrow B$, and $r_1 \prec r_2$ and $\text{conf}(r_1) \geq \text{conf}(r_2) > P(B)$, if $r_2$ is a multi-correlation rule, then $r_1$ is also a multi-correlation rule. In such a case, $r_2$ is called a *redundant rule*.

The brute force algorithm for generating all multi-correlation rules requires exponential effort. Another method is to mine a set of frequent itemsets first, and then search for all multi-correlation rules from the resulting set. We propose an alternative method.

### 2.2 Positive multi-correlation rules

Given an itemset $A$ and single item $B$, for any positive multi-correlation rule $A \Rightarrow B$, we know $P(AB) - P(A)P(B) > 0$. We can derive a new tight upper bound for positive multi-correlation rules from previous work [2][11] as follows.

Suppose $A' \supseteq A$, $x(A) = N \times P(A)$, and $y(A) = N \times P(AB)$. According to Equation 1, the upper bound for positive correlation is given by

$$\chi^2(A' \Rightarrow B)$$
$$\leq \max\{\chi^2(y(AB), y(AB)), \chi^2(x(A) - y(AB), 0)\} =$$
$$\max\left(\frac{NP(AB)\left(1-P(B)\right)}{P(B)(1-P(AB))}, \frac{N(P(A)-P(AB))P(B)}{(1-P(A)+P(AB))(1-P(B))}\right)$$

For a positive rule and its subrules, from Theorem 1, we have the following theorem.

**Theorem 2**. Given two positive multi-correlation rules $r_1$:$A_1 \Rightarrow B$ and $r_2$:$A_2 \Rightarrow B$, and $r_1 \prec r_2$ and $\text{conf}(r_1) \geq \text{conf}(r_2)$, then $\chi^2(r_1) \geq \chi^2(r_2)$.

Thus, given two candidates for positive multi-correlation rules $r_1$:$A_1 \Rightarrow B$ and $r_2$: $A_2 \Rightarrow B$, and $r_1 \prec r_2$ and $\text{conf}(r_1) \geq \text{conf}(r_2)$, if $r_2$ is a multi-correlation rule then $r_1$ is certainly a multi-correlation. As well, $r_2$ can be regarded as *redundant* and removed from the set of multi-correlation rules. A rule $r$ is *optimal* if no subrule of $r$ exists such that $r$ is redundant. Optimal rules are preferred for many applications [3]. Another type of non-redundant rules, called basic rules, is stricter than optimal rules and can be used for inference [9]. A *basic rule* is a rule that has confidence greater than or equal to that of any of its superrules, and the same condition is not true for any of its subrules.

A corollary to Theorem 2 identifies redundant rules.

**Corollary 1**. Given a positive multi-correlation rule $r$:$A \Rightarrow B$ where $conf(r) = 1$, then all superrules of $r$ are redundant. □

## 2.3 Incrementally counting suffix trees

An *incrementally counting suffix tree*, abbreviated *ICST*, is an extended version of a suffix tree. A suffix tree, also called a *suffix trie*, is well known for data stream analysis [5]. An example is shown in Figure 1. Suppose we have the dataset shown in Figure 1(a). The corresponding transaction dataset with items in lexicographical order is shown in Figure 1(b). The suffix tree with counts for the first transaction, *ACDE*, is shown in Figure 1(c), where the roots of the subtrees for all suffix transactions are inside the dotted oval.

An ICST can be built in online manner by the process, called getICST, which aggregates all suffix trees of transactions with incrementally counting.

Given an itemset $A_1 \ldots A_k$, the getSupp method for estimating its support from an ICST can be described as follows. From the root of the ICST, the algorithm first searches for a node containing $A_1$. If such a node is found, then the search for subitemset $A_2 \ldots A_k$ is continued in the subtree of the node containing $A_1$ in the same manner. Eventually, the support for itemset $A_1 \ldots A_k$ is obtained by adding all supports in the leaves containing $A_k$. Otherwise, the support equals to 0.

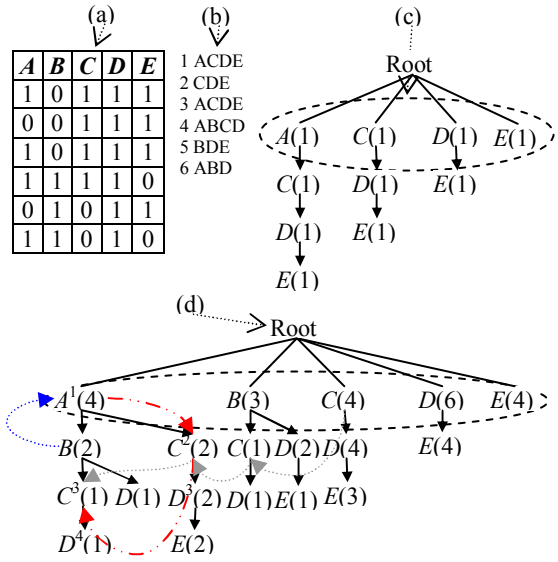


**Figure 1. An ICST.**

A *threaded ICST* is an ICST with extra pointers. Each node has a pointer field called *parent*, which points to its single direct predecessor. For example, in Figure 1(d), in the subtree corresponding to item *A*, the node containing item *B* has a parent field pointing to its predecessor node containing item *A*. However, the root of an ICST has no predecessor. Also in a threaded ICST, all subroots contain a pointer called *head*, which is the head of a linked list linking all nodes containing the same item in other subtrees and a specific field called *tails* referring a set of linked lists. Each linked list in the tails links to all nodes containing the same items in the same subtree. For example, as shown in Figure 1(d), the *head* in the subroot labelled *C* links to four nodes containing item *C* in the corresponding subtrees labelled *C*, *B*, and $A^1$; there is a hashtable *tails* in the subroot labeled $A^1$, in which one element is a head node of a linked list referring the first node containing item *C*. The head node uses its *next* field to point to the next node containing the same item *C* in the linked list. The *next* field of the last node in any linked list is null. For simplicity, all other threads between parents and linked lists have been omitted from Figure 1(d).

The linearity of ICSTs has been proven in the related research [10]. For scalability, the dataset can be partitioned into equal or unequal sized units. A local ICST can be built for each unit. The support of an itemset can be estimated from the local ICST. The final set of multi-correlation rules that is created by combining the sets of multi-correlation rules that are generated from the local ICSTs will be an approximation to the correct set.

## 2.4 Heuristic candidate generation

In this paper, we assume that only patterns recorded in the ICST are of interest. To efficiently generate candidate pattern rules, the ICST is traversed. At each node, we consider both greater rules and lesser rules. For a pattern rule $r$: $A \Rightarrow B$, if the item located at the tail of $A$ is less than $B$ in lexicographic order, then $r$ is a *lesser rule* with respect to $B$; otherwise, $r$ is a *greater rule* with respect to $B$. For example, rule $GH \Rightarrow J$ is a lesser rule and rule $GJ \Rightarrow H$ is a greater rule. Suppose a node corresponds to a pattern $A$, as obtained from getItemsetPrefix. For each item $B$, a candidate rule with antecedent $A - B$ and consequent $B$ is considered: if $A \cup B$ is in a pattern, then $A - B \Rightarrow B$ is a valid candidate.

By heuristically generating candidates for pattern rules, the search space for multi-correlation rules can be reduced, as described by the following theorem.

**Theorem 3**. Given an ICST and a candidate pattern rule $A \Rightarrow B$ generated in a node in the ICST, if $A \Rightarrow B$ is an invalid candidate, then any of its superrules generated in any node of the ICST is also invalid. □

In four cases, all superrules of a candidate can be discarded. First, for an invalid candidate, its antecedent is not a pattern in the context of its consequent. Secondly, the tight upper bound is evaluated to be lower than a critical value given by a statistical metric. Thirdly, the continuity correction threshold is not satisfied [1]. Fourthly, it is identified to be redundant

by a heuristic method, e.g., if the confidence of a pattern rule is already 1.

The pruning method based on Theorem 3 is called *information propagation*. It can often be accomplished my moving from the small subtrees to the large subtrees. Information propagation can be achieved with threads referred to by the head and tails among subtrees and within a subtree, respectively, by finding some special nodes, called *critical nodes*, from which all superrules are known to be invalid candidates or evaluated to be invalid rules. For example, as shown in Figure 1(d), if the candidate $D \Rightarrow B$ generated in the subtree labeled D is invalid, $D$ is identified as a critical node in the subtree labeled C with respect to the item $B$ (called visited by item $B$) while $D^3$ is identified as a critical node in the subtree labeled B for $CD \Rightarrow A$ generated in the subtree labelled C with respect to the item $A$ if it is invalid.

For each node $n$, the following four states are identified for information propagation.

$n$.state = 'C', meaning visited by item C,

$n$.state = 'X' or '', meaning not visited by item C,

$n$.state = 'C–C', meaning a critical node for item C,

$n$.state = 'C+C', meaning a critical node for C that has been propagated from other subtrees.

Other redundant pattern rules can be removed by (1) finding all pattern-based optimal rules from the resulting pattern rules or (2) by finding all pattern-based basic rules from them.

## 3. The algorithm for finding pattern rules

We propose the PPMCR algorithm, given in Figure 2, for searching for a set of pattern rules, given a dataset $D$, assuming that the $\chi^2$ test is the statistical metric. Parameter values are required for the size $W$ of the partitions and the critical value for the $\chi^2$ test.

To begin, PPMCR partitions the transaction dataset into partitions of size at most $W$ in step 1. For each partition, PPMCR creates a local ICST in step 2 using function getICST. In step 3, for each item in the local ICST, the algorithm searches for pattern rules that have the item as their consequents in subtrees of ICST in step 4. In step 5 and step 7, candidates are identified as greater rules or lesser rules such that two procedures getGreaterRules and getLesserRules are executed for evaluating them, respectively.

In step 17 of getGreaterRules, by checking whether the supports of itemsets consisting of all combinations of $A$ and $B$ and their complements are lower than 5, ContinuityCorrection performs the correction for continuity assumed by the $\chi^2$ test [1].

The getLesserRules and other procedures are omitted due to the limited space available.

```
PPMCR algorithm
Input:  D(I, T): a transaction dataset
        W: the size of each partition for D
Output: a set of multi-correlation rules
begin
1     for each partition p# in D of size at most W
2           CST = getCST(p#)
3           for each item B in the root node of a subtree in CST
4                 for each other root node n of a subtree in CST
5                       if (n.item > B)
6                             getGreaterRules(n, B)
7                       else if (n.item < B)
8                             getLesserRules(n, B)
end
Procedure getGreaterRules(n, B)
9     sk.push(n)
10    while (sk ≠ ∅)
11          n = sk.pop();
12          if (n.state == B+B)
13                loop
14          A = getItemsetPrefix(n) – B
15          if (!InPattern(A, B))
16                propagation(A, B); loop
17          if (!ContinuityCorrection(A, B))
18                propagation(A, B) loop
19          if (supp(r) > supp(A) * supp(B))
20                if (upper(A, B) < χ²α)
21                      propagation(A, B); loop
22                output {r:A ⇒ B | χ²(A, B) ≥ χ²α} or
23                getNonRedundantRules(r)
24                if (r.conf ==1)
25                      propagation(A, B); loop
26          for each child c in n
27                s.push(c)
```

**Figure 2. PPMCR algorithm for finding non-redundant pattern rules.**

PPMCR is complete for generation and evaluation of all valid candidates of pattern rules.

## 4. Experimental results

For experiments, several real-world datasets were taken from the UCI repository [6]. The characteristics of these datasets are described in Table 1.

For the small datasets, such as Chess and Mushroom, the partition size was set to the number of transactions in the dataset, i.e., 3196 and 8124, respectively, as shown in Table 1. Larger datasets were partitioned into equal sized units, e.g., 10000 transactions only for the partition 1 of Connect.

Five pruning techniques based on our theoretical results were used: Upper is based on the upper bound, Pattern is based on the validation of pattern rules, Correction is based on continuity correction, Redundant is based on Corollary 1, and Propagation is based on information propagation. For example, in Table 1, for Chess with respect to the critical value 3.841 of $\chi^2$ test and the critical value 0.6 of Pearson's correlation coefficient, Upper prunes 0 and 2792 times, Pattern prunes 983 and 1218 times, Correction prunes

3864 times, Propagation prunes 332830 and 352833 times, and PPMCR generates 4101 optimal rules (or 8155 basic rules) and 1774 basic rules. The elapsed time for generating optimal rules is 363 seconds.

A comparison between our approach and three previous approaches is shown in Table 2. Apriori is an early method for mining association rules. We call the method for generating non-redundant association rules GNRAR [14] and the method for mining non-derivable association rules, MNDAR [7]. Experiments on the same benchmark datasets were reported by the original authors. For MNDAR, we chose the version called Id-cc, which generates subrules with identical conditions or consequent rules for comparison. PPMCR generates fewer optimal rules than the other approaches in most cases, but GNRAR generates 475 rules for Mushroom when minsup = 0.4. PPMCR generates all pattern rules with a high level of statistical significance regardless of their support values. Thus, rules with low support values are included in the resulting set.

## 5. Conclusions and future work

In this paper, we defined the problem of finding all pattern rules from a transaction dataset given a statistical metric and proposed the PPMCR algorithm to solve this problem. Experiments conducted on several real-world datasets show that our approach efficiently generates fewer rules than previous approaches in most cases. We identified four salient advantages of our approach compared to previous approaches for mining frequent itemsets, association rules, and correlation rules [3][4][8][14]. In the future, a compressed version of the ICST might be devised for the PPMCR algorithm. As well, other types of patterns might be generated from the ICST.

**Table 1. Synthetic and real-world datasets.**

| Datasets | # Items | # Trans | *L* | *W* | # Upper | # Pattern | # Correction | # Redundant | # Propagation | # Optimal | # Basic | #0.6 Basic | Elapsed Time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Chess | 76 | 3196 | 37 | 3196 | 0 / 2792 | 983 / 1218 | 3864 | 0 / 8 | 332830 / 352833 | 4101 | 8155 | 1774 | 363 |
| Connect | 130 | 67557 | 43 | 10000 | 0 / 6278 | 3083 / 2480 | 6975 | 0 / 250 | 1441825 / 1427964 | 9445 | 29051 | 5091 | 2458 |
| Mushroom | 120 | 8124 | 23 | 8124 | 0 / 4275 | 5769 / 4748 | 5936 | 0 / 88 | 138394 / 122740 | 5629 | 20159 | 870 | 318 |

**Table 2. Comparison of the number of rules.**

| Dataset (minsupp) | Apriori | MNDAR (Id-cc) | GNRAR | PPMCR |
|---|---|---|---|---|
| Chess(0.8) | 552564 | 65978 | 27711 | **4101** |
| Chess(0.7) | 8171198 | | 152074 | |
| Connect(0.97) | 8092 | 11231 | 1116 | **9445** |
| Connect(0.9) | 3640704 | | 18848 | |
| Mushroom(0.4) | 7020 | 94860 | 475 | **5629** |
| Mushroom(0.2) | 19191656 | | 5741 | |

## 6 References


[1] A. Agresti. *Categorical Data Analysis*. New York: John Wiley, 1996.

[2] S. Bay and M. Pazzani. Detecting Change in Categorical Data: Mining Contract Sets. In *Proc. KDD'99*, 302-306, San Diego, CA, 1999.

[3] R. J. Bayardo and R. Agrawal. Mining the Most Interesting Rules. In *Proc. KDD'99*, 145-154, 1999.

[4] S. Brin, R.Motwani, and C. Silverstein. Beyond market baskets: generalizing association rules to correlations. *Proc. SIGMOD'97*, Tucson, AZ, 1997.

[5] J. Fischer, V. Heun, and S. Kramer. Fast frequent string mining using suffix arrays. In *Proc. 5th IEEE International Conference on Data Mining*, 609-612, 2005.

[6] Frequent Itemset Mining Dataset Repository. http://fimi.cs.helsinki.fi/data/.

[7] B. Goethals, J. Muhonen, and H. Toivonen. Mining Non-derivable Association rules. *Proc. 2005 SIAM International Conference on Data Mining* (*SDM05*).

[8] J. Han, J. Pei, Y. Yin, and R. Mao. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Mining and Knowledge Discovery*, 8:53–87, 2004.

[9] G. Li and H. J. Hamilton. Basic Association Rules. In *Proc. 2004 SIAM International Conference on Data Mining* (*SDM04*), 166-177, Orlando, FL, April, 2004.

[10] G. Li and H. J. Hamilton. *Searching for Pattern Rules: Extended Report*. Technical Report CS-2006-09, Department of Computer Science, University of Regina.

[11] S. Morishita and J. Sese. Traversing Itemset Lattices with Statistical Metric Pruning. In *Proc. 19th ACM SIGACT-SIGMOD-SIGART Symposium on Database System (PODS)*, 226-236, 2000.

[12] H. T. Reynolds. *The Analysis of Cross-classifications*. The Free Press, New York, 1997.

[13] H. Xiong, S. Shekhar, P. Tan, and V. Kumar. Exploiting a support-based upper bound of Pearson's correlation coefficient for efficiently identifying strongly correlated pairs. *Proc. ACM SIGKDD'04*, August 2004, Seattle, Washington, USA.

[14] M. Zaki, Mining non-redundant association rules. *Data Mining and Knowledge Discovery*, 9:223–248, 2004.