The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014)

Espresso for Rule Mining

Elhoussini F. Ashmounia, Rabie A. Ramadanc, Ali. A. Rashedc

a,c Systems and Computers Department, Alazhar University, Cairo, Egypt
b Computer Engineering Department, Cairo University, Cairo, Egypt

Abstract

The Rule-based systems generate many of the redundant rules. Such rules are expensive especially in online systems. Currently, there are many of the available rule minimization techniques; however, they still suffer from many challenges in exploiting parallelism, load balancing, efficient memory usage, minimization of communication cost, efficient data, task decomposition and others. This paper introduces a new approach for minimizing association rules based on the adaptation of Espresso algorithm, used in reducing Boolean expressions. We believe that our proposed method is a simple and efficient method that supports a large number of input and output variables. The proposed method starts by the usage of binary encoding followed by the minimization. In the last step, data decoding is utilized generating the final rules. Such rule minimization could be used in many applications including the Wireless Sensor Networks collected data. For testing purposes, a car data set has been used and the results seem promising compared to the original rules

Keywords: rule association mining; Car evaluation dataset; WSNs

1. Introduction and Overview

There is a dramatic exponential increase in the amount of information or data being manipulated, in which mining for data becomes essential to support decision making. Data mining is a key step in the knowledge discovery process, where the main tasks are divided into Predictive or Descriptive. Predictive in terms of predicting the value of a particular attribute based on the values of other attributes or extracting previously unknown and useful

* Corresponding author. Tel.: +2-012-276-19419; +2-010-676-43227; fax: +2-02-353-70273.
E-mail address: rabie@rabieramadan.org; housainy@gmail.com
information such as patterns, associations, changes, anomalies and significant structures, from large datasets. There are several techniques satisfying these objectives of data mining such as clustering, classification, association rule mining, sequential pattern discovery and analysis. This paper focuses on Association Rule Mining for knowledge discovery where a large number of current systems are based on association rules including the WSNs. Rules are usually generated from different sets of items and can be expanded dramatically where the increase of the item set frequency increases the number of the association rules in a proportional or exponential relationship.

The mining for association rules between items in large datasets is an important area of data mining research. Rules can be manipulated to discover unknown relationships and produce the results that provide a basis for forecasting and decision making. A number of association rule mining algorithms and techniques have been developed in the last few years. Such algorithms can be classified into two classes: 1) candidate generation/test approach such as Apriori and 2) pattern growth approach. However, both approaches suffer from the following drawbacks:

1) Inefficiency in reading large datasets (where it was designed in forms of several passes so that the whole dataset needs to be read several times); consequently the performance will be affected.
2) A “try and error” approach may be used to get suitable number of rules where there is no clear guide for the suitable constraints’ settings like the support and the confidence. This is very time consuming and inefficient.
3) The algorithms generate a large number of association rules. Most of the time it is not easy to understand or validate such large number of complex association rules. Many of these rules are redundant and/or irrelevant and thereby limiting the usefulness of the data mining results. Again, this is very time consuming and inefficient.
4) Some of the algorithms are based on constructing a decision tree in which it is not efficient to be constructed for large data.

Therefore, one of the main challenges in mining association rules is to reduce such number of rules. This paper introduces a new approach for association rule minimization based on the basics of Boolean algebra and the digital logic minimization. The approach starts by using encoding to compress the data; then it adapts the Espresso heuristic logic minimizer simplification method for the rule minimization. The paper proposes an efficient binary encoding method for data/database compression. Thus, the data size is reduced and its scanning time is minimized. Moreover, our proposal handles multiple output states with high performance.

Due to the limited space, the following subsections contain an overview on the most related association rules techniques to our proposal including Quine-McCluskey and Espresso as well as some other heuristics.

1.1. Quine–McCluskey

Quine-McCluskey is a tabular method to find all prime implicants systematically. The method starts with the truth table for a set of logic functions. By combining the minterms for which the functions are active (the ON-cover) or for which the function value is irrelevant (the Don’t-Care-cover or DC-cover), a set of prime implicants is composed. Finally a systematic procedure is followed to find the smallest set of prime implicants that the output functions can be realized with. However, Quine-McCluskey is considered as an exact method for logic minimization which takes a very long time when large number of input and outputs are used. Certainly, such computation time will not be suitable for real time machines that use such method. On the other hand, the method calculates all prime implicants to derive the optimal solution(s) and derives all covers to determine minimum cover set(s). Unfortunately the number of prime implicants grows quickly in which the solution space becomes huge. Finding the minimum cover set in a class is NP-Hard problem where determining optimal solution is difficult. For majority of application, there is a need for moving to heuristics. In, Quine-McCluskey method is introduced for rule minimization. New encoding technique is used allowing rule reduction. However, due to the method restrictions, it is not suitable to be used in large data. In order to avoid the restriction to the number of variables, output functions and product terms of a combinational function, Espresso algorithm could be a suitable heuristic to do so which is our proposal in this paper.
1.2. Espresso

The Espresso algorithm was developed by Brayton et al. at the University of California, Berkeley\textsuperscript{15}. It has been included as one of the standard logic function minimization step in many logic synthesis tools. For implementing a function in multi-level logic, the minimization result is optimized by factorization. The results are mapped onto the available basic logic cells in the target technology including Field Programmable Gate Array (FPGA) or Application Specific Integrated Circuit (ASIC). The Espresso manipulates "cubes" that represent the product terms in the ON-, DC- and OFF-covers iteratively instead of expanding a logic function into minterms. Practically, when using espresso the minimization result is not guaranteed to be global minimum. However, it is very closely approximated and solution is always free from redundancy. The advantages here, compared with other methods are the provided efficiency, reduction of memory usage and computation time by several orders of magnitude. In addition Espresso has the ability to deal efficiently with many of input variables and many of output functions as well as product terms of a combinational function block.

Espresso is the most popular heuristic two-level logic minimization\textsuperscript{21}. Firstly, it finds complement of the original cover that is eligible to be used in the EXPAND operation. Secondly, it applies the EXPAND and IRREDUNDANT operation to obtain an ISOP. Thirdly, it extracts the set of EPIs, and it iterates the REDUCE, the EXPAND and the IRREDUNDANT operations until no product can be reduced any more. Fourthly, it attempts to REDUCE and EXPAND by using different heuristics. Finally, ESPRESSO tries to reduce the number of connections in the output part\textsuperscript{14}. As can be seen in Figure 1, there are many functions explained as follows:

- **EXPAND**: In Espresso, the goal of EXPAND is to enlarge each implicant in turn into a prime implicant. Once the implicant is expanded, it may contain other implicants that can be removed; hence the cover cardinality is reduced. If the current implicant cannot be expanded to contain another implicant completely, then, the implicant is expanded to overlap as many other implicants of the current cover as possible. In Espresso, an implicant is expanded until no further expansion is possible, i.e. when the implicant is prime. Two steps are used: (i) expansion to overlap a maximum number of cubes still covered by the over-expanded cube; and (ii) raising of entries to find the largest prime implicant covering the cube\textsuperscript{17}.

- **Essentials**: Essential prime implicants are prime implicants that need to be included in any cover of prime implicants. Therefore, it is desirable to identify them as soon as possible making the resulting problem size smaller. On the one hand, there is no efficient solution for identifying the essential primes using the unate recursion paradigm as in Espresso.

- **Reduce**: The goal of the REDUCE operator is to set up a cover that is likely to be made smaller by the following EXPAND step. To achieve this, each cube \(c\) in a cover \(F\) is maximally reduced in turn to a cube \(\tilde{C}\), such that the resulting set of cubes, \(\{F-C\} \cup \tilde{C}\) is still a cover. Espresso uses the unate recursive paradigm to maximally reduce each cube.

- **Irredundant**: Espresso uses the unate recursive paradigm to find an irredundant cover.

![Fig. 1: Espresso Method](image-url)
The details of the Espresso algorithm can be described as follows:

**Algorithm 1:**

```
Procedure ESPRESSO ( F, D, R)  /* F is ON set, D is don’t care, R OFF */  
R = COMPLEMENT(F+D);  /* Compute complement */  
F = EXPAND(F, R);  /* Initial expansion */  
F = IRREDUNDANT(F,D);  /* Initial irredundant cover */  
F = ESSENTIAL(F,D) /* Detecting essential primes */  
F = F - E; /* Remove essential primes from F */  
D = D + E; /* Add essential primes to D */  
WHILE Cost(F) keeps decreasing DO  
    F = REDUCE(F,D);  /* Perform reduction, heuristic which cubes */  
    F = EXPAND(F,R);  /* Perform expansion, heuristic which cubes */  
    F = IRREDUNDANT(F,D);  /* Perform irredundant cover */  
ENDWHILE;  
F = F + E;  
RETURN F;  
END Procedure;
```

The following table Table.1 shows a comparison between Quine-McCluskey and Espresso algorithms.

<table>
<thead>
<tr>
<th>Quine-McCluskey</th>
<th>Espresso</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact Method for Logic Minimization</td>
<td>Heuristic Method for Logic Minimization</td>
</tr>
<tr>
<td>Compute Time is prohibitive for minimizations (Doubly Exponential)</td>
<td>Some sacrifice in optimization for a large reduction in run time</td>
</tr>
<tr>
<td>Times-out for medium complexity benchmarks, Too slow on large problems</td>
<td>No restriction to the number of variables, output functions and product terms of a combinational function</td>
</tr>
</tbody>
</table>

**1.3. Other Algorithms**

Various researches were done to improve the performance of the rule minimization. The studies have manipulated multiple ideas to improve the performance like the speed of finding large itemsets with hash table, map, and tree data structures. Also there were studies to improve the logic minimization such as Genetic Algorithms (GAs) and Boolean representation.

Optimized association rule mining using genetic algorithm have been discussed from different perspectives. By using GAs, the system will be able to predict rules that have negative attributes in the generated rules along with more than one attribute in its consequent part. The GA algorithm can also be used to search a solution space for an optimal solution to a problem. The algorithm creates a “population” of possible solutions to the problem and lets them “evolve” over multiple generations to find better and better solution. The major advantage of using GAs in the discovery of prediction rules is that they perform global search and its complexity is less compared to other algorithms as the genetic algorithm is based on the greedy approach.

The authors in introduced approach uses Boolean representation for an input of database D, which is scanned only once to build a Boolean matrix M. M has N columns and K rows, where N represents number of items and K represents number of transactions in D. A position (i,j) in M is 1 iff in transaction I, item j exists, and 0 otherwise. Boolean algebra operations are applied on M to generate all frequent itemsets, frequent items or frequent sequential patterns has been represented by a logical expression that could be minimized by using a suitable logic function minimization technique.
2. Espresso for Association Rule Minimization

The Association Rules are extracting from the system database by counting all the possible combination of attributes. With a large number of attributes, the performance goes down, where the system’s computational time increases exponentially. Many algorithms have been developed to reduce the system search space including \(^{11}\). In this paper, we introduce Espresso as a rule mining algorithm. Espresso reduction uses a heuristic logic reduction algorithm to reduce combinational functions. Although it is not guaranteed to always produce optimally minimal expressions, however, its results are close to optimal. In addition, Espresso is able to handle large functions that would be impossible to minimize manually or even optimally. The developed approach adopts the mechanism of Logic Minimization approach with modifications to reduce the processing time of Rule Minimization algorithms. The idea of encoding and data compression is used, following by engaging Espresso heuristic logic during processing, as shown in following Figure 2. The steps of the proposed approach are as follows:

1. Data Preparation
   a. Identifying attributes and outputs
   b. Mapping the rules according to the application
   c. Encoding and Data Compression
2. Utilization Espresso heuristic logic minimizer simplification method for the rule minimization, and
3. Re-Mapping the results

![Fig. 2: Steps of Minimizing the Association Rules Mining Using Espresso for Logic Minimization](image)

In the data preparation step, attributes and outputs are identified. Then, the rules are mapped according to the application. Espresso for rule mining does not work directly on raw data; the manipulated dataset should be filtered out to determine the attributes and the output functions to be encoded based on the Boolean algebra concepts and its basic operations to the form of Binary representation technique (0 and 1). In addition, it is important to map this rules according to the area of the application used. The approach starts by counting the number of support factors in each item. This could be done by the dataset owner; otherwise an automatic approach should be considered to do one pass over the dataset records to extract the variations in the items. There are techniques to improve the performance like the one in \(^{20}\). Based on the number of supports, a number of bits are identified for each item.

After completing the encoding step, the Espresso heuristic logic minimizer algorithm is used to minimize the number of rules in the data set. The reduction of Boolean functions is an efficient way of minimization in a rule-based system. After classifying the attributes according to its decision values, the input files will be constructed for each decision value to be input into Espresso logic minimizer.

Finally, the output retrieved after running the Espresso Logic Minimizer represents the simplified rules in a mapped format in which needs to be decoded in order to be readable for other applications.

3. Experimental Results

As a proof of concept, Espresso and the encoding method are implemented along with the re-mapping after all. The efficiency of the proposed method is tested over Car Evaluation Dataset \(^{13}\). Basically, this dataset is used for classification. The dataset is a car evaluation database that was derived from a simple hierarchical decision model originally developed for the demonstration of DEX model. The dataset contains of 1728 record. The following steps are used for rule minimization and extraction.

1) Data Preparation
After analyzing the dataset contents, the evaluation attributes have been extracted and mapped in a binary form as shown in Table 2. In this dataset, six attributes are found, provided by the UCI \(^{13}\). The proposed approach in his paper has the capability of handling multi-state output along with data compression by converting the data from
string data to binary. For example, a car evaluation’s dataset has six attributes (buying cost, cost of maintenance, number of doors, capacity in terms of persons to carry, the size of luggage boot and the estimated safety of the car). There are four attributes of decision making for the car evaluation (unacceptable, acceptable, good and very good) which is the output decision/state.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Attribute values</th>
<th>Mapping Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buying</td>
<td>v-high, high, med, low</td>
<td>11,10,01,00</td>
</tr>
<tr>
<td>Maint</td>
<td>v-high, high, med, low</td>
<td>11,10,01,00</td>
</tr>
<tr>
<td>Doors</td>
<td>2,3,4,5-more</td>
<td>11,10,01,00</td>
</tr>
<tr>
<td>Persons</td>
<td>2,4,more</td>
<td>11,10,01,00</td>
</tr>
<tr>
<td>Lug_boot</td>
<td>Small, med, big</td>
<td>11,10,01,00</td>
</tr>
<tr>
<td>Safety</td>
<td>Low, med, high</td>
<td>11,10,01,00</td>
</tr>
<tr>
<td>output</td>
<td>vgood, good, acc, unacc</td>
<td>11,10,01,00</td>
</tr>
</tbody>
</table>

2) **Espresso heuristic logic minimizer Utilization**

After completing the encoding step, the Espresso heuristic logic minimizer algorithm to minimize the number of rules in the data set comes to scene. In order to use Espresso and as given in Table 2, the attributes (buying cost, cost of maintenance and number of doors) have four different values while the others have three different values. Therefore, such attributes are encoded in the form of 12 variables (A, B, C,…,K and L). The first three attributes are encoded by 2-bits in which (low, med, high and v-high) values for buying are encoded as (00, 01, 10 and 11), respectively. In addition, the values of doors attribute (2, 3, 4, 5-more) values are encoded by (00, 01, 10 and 11), respectively. The last three attributes are also encoded by 2-bits with three combinations (00, 01, and 10). The variables after attributes encoding is named (A, B, C, D, E, F, G, H, I, J, K and L) which the number of variables reaches 12 to restrict the problem space. Each row has a decision value of five (unacc, acc, good, v-good and d) which (d) refers to “Don’t care” combination. When we look at the whole table, we can discover that the states (unacc, acc, good and v-good) have 1210, 384, 69 and 65 rows, respectively. While the “Don’t care” state takes the rest of rows (2386 rows). The proposed method, as described above, is used to separately reduce each state.

After classifying the attributes according to its decision values we can implement the following steps

- Construct an input file for each decision value.
- Run the Espresso Logic Minimizer.
- Read the output file for each decision value.

When we look at the whole output after running the Espresso Logic Minimizer for the given input, we can discover that the states (unacc, acc, good and v-good) have 38, 58, 20 and 10 with a reduction percentage of (3.14%, 15.10%, 28.99% and 15.38%) respectively with an average of 126 output function produced from 1728 record dataset; the output of each state is given as follows:


As can be seen, this can be implemented in a PLA in an easy way. Here both of the input and output have been arranged according to the information gain. We have to refer to the fact that output will change when selecting different sequence of attributes.
3) Output Decoding

As show in Fig 3, the simplification process produces 126 product terms. When such terms are decoded, we can obtain 126 rules. A sample from the car evaluation’ rules after decoding is illustrated in Fig 4. As an example, the associated rule of an output product term like ‘¬A ¬C ¬D G ¬HIJ ¬KL’ can be deduced as “if (Buying low or med) and (maint, low) and (persons,4) and (lug_boot, small) and (safety, high) THEN good”:

- IF (buying, low) and (main, med or low) and (doors, greater than 2) and (persons, more than 4) and (lug_boot, small) and (safety,high)) THEN good.
- IF (buying, low) and (maint, low) and (persons,4) and (lug_boot,small) and (safety,high) THEN good.
- IF (buying, low or med) and (maint,low) and (persons,4) and (lug_boot,small) and (safety,high) then good.
- IF (buying, low) and (maint, low) and (persons,4) and (lug_boot,small) and (safety,medium) THEN good.
- if (buying, low or med) and (maint, low or med) and (doors,5 more) and (persons,more) and (lug_boot, med) and(safety, high) THEN v-good.
- if (buying, low or med) and (maint, low or med) and (persons, more) and (lug_boot,med) and(safety, high) THEN v-good.
- if (buying, low) and (maint, low) and (doors,5 more) and (persons,more) and (lug_boot,big) and(safety, high) THEN v-good.
- if (buying, low) and (maint, low) and (persons,more) and (lug_boot,big) and(safety, high) THEN v-good.
- if (buying, v-high) and (maint, high) and (persons, more) and (lug_boot, small) and(safety,high) THEN acc.
- if (buying, high) and (maint, high) and (persons, more) and (lug_boot, small) and(safety,high) THEN acc.
- if (buying, v-high) and (maint, high) and (persons, more) and (lug_boot, small) and(safety,high) THEN acc.
- if (doors,2) and (persons,2) and (lug_boot,small) and(safety,low) THEN unacc.
- if (maint, low) and (persons,2) and (lug_boot,small) and(safety,low or med).
- if (buying, high) and (maint,v-high) THEN unacc.
- if (buying, v-high) and (maint, high) THEN unacc.
- if (buying, v-high) and (maint, high) THEN unacc.

Fig. 4- Sample of car evaluation rules

4. Conclusion

The aim of this paper was to improve the performance of minimizing association rules mining by providing a way to adapt a heuristic algorithm used in reducing Boolean expressions. The approach is to attain the desired improvement by using heuristic methods like espresso which is more suitable for fast simplification of larger dataset problems and supports many items, in addition to its flexibility to deal with a large number of input variables and output functions. The approach can be used with all type of applications and the high benefits can be obtained for those application types where exact solutions are not necessary, however, solutions that are near to the optimum in a relatively short time are preferred like in different WSN applications. The minimization method is very simple and supports many items and can be used with a large number of variables. The introduced method has four benefits over the current rule minimization methods which are:

1) The encoding phase allows the data set reduction,
2) Could be used with any type of data set with multiple states input or output
3) Compared to the other methods, this one is essentially more efficient, reducing memory usage and computation time by several orders of magnitude. There is no restriction to the number of input variables, output functions and product terms of a combinational function block. In general, e.g. tens of variables with tens of output functions are readily dealt with
4) It is very simple to be implemented in a software package; therefore no user supervision is required as well as setting certain thresholds.

5. References


7. Espresso , Minimization software , can be found at http://www.physics.dcu.ie/~bl/digi/unitd17.pdf _


