A Frequent Patterns Tree Approach for Rule Generation with Categorical Septic Shock Patient Data

Jürgen Paetz[1,2] and Rüdiger Brause[1]

[1] J.W. Goethe-Universität Frankfurt am Main,
Fachbereich Biologie und Informatik,
Institut für Informatik, AG Adaptive Systemarchitektur
Robert-Mayer-Straße 11-15, D-60054 Frankfurt am Main, Germany
[2] Klinikum der J.W. Goethe-Universität,
Klinik für Allgemein- und Gefäßchirurgie
Theodor-Stern-Kai 7, D-60590 Frankfurt am Main, Germany
{brause,paetz}@cs.uni-frankfurt.de
http://www.medan.de

Abstract In abdominal intensive care medicine letality of septic shock patients is very high. In this contribution we present results of a data driven rule generation with *categorical septic shock patient data*, collected from 1996 to 1999. Our descriptive approach includes preprocessing of data for rule generation and application of an efficient algorithm for frequent patterns generation. Performance of generated rules is rated by frequency and confidence measures. The best rules are presented. They provide new quantitative insight for physicians with regard to septic shock patient outcome.

1 Introduction

A septic shock during a stay in an intensive care unit affects outcome in a negative manner [1], [2]. This phenomenon is related to mechanisms of the immune system [3]. Our approach to reduce letality of septic shock patients is the automated, intelligent search of information in already documented patient records without looking at additional costly measurements of immune system reactions and markers. We analysed the data of 362 patients by 30 boolean variables, including almost all the usually documented categorical data like relevant diagnoses, medicaments and therapies. For technical reasons operations were not included in this analysis. Data was collected in a german hospital from 1996 to 1999. 14.9% of all the patients are deceased. Our analysis of categorical data carries on the analyses already done for metric septic shock patient data with another data base in [4] and [5]. To find interesting rules within the high number of all the rules coming from subsets of 30 variables we used a tree search algorithm based on [6] which is described in Sect. 2. Subsequently, in Sect. 3 achieved results are presented.

2 Frequent and Confident Patterns by FP-Trees

If you have medical – or economical, geological, biological, etc. – categorical variables (in database language called *attributes*) with different possible values like {high, middle, low}, {yes, no} or {A, B, C, ...}, i.e. n different variables $v_1, \ldots, v_n$, each variable v_i takes one of the m_i different values $a_{i,1}, \ldots, a_{i,m_i}$, it could be interesting to find out which combinations of attribute values could be observed with respect to one class c like the class of deceased patients. Therefore, all the patients P_j are listed with their attribute values and their outcome, e.g. P_1 := (high,yes,...,C,no;deceased), P_2 := (low,yes,...,D,no;survived), etc. Now we are looking for rules of the form "**if** $v_{l_1} = a_{l_1,k_1}$ **and** ... **and** $v_{l_s} = a_{l_s,k_s}$ **then** class c" that contain a small number s of variables. Rule R is better than a rule S if it is valid for more samples (patients), and R is also better than S if it produces less misclassifications. For this purpose the *frequency* and *confidence* of a rule are defined in Sect. 2.1.

There exist a lot of different approaches to solve the combinatorical problem of finding appropriate rules efficiently, e.g. [7], [8]. Ref. [7] combines the elementary attributes to more complex combinations, evaluating the frequencies of the emerged patterns (*association rules*). [8] starts using the complex patterns P_j, melting them to less complex rules (*generalization rules*). Both algorithms are based on *frequent patterns*, i.e. frequent occurences of equal attribute values in different samples. So, here we base our algorithm on a frequent patterns approach to generate rules, improving it by additional confidence calculations, see Sect. 2.2.

2.1 Frequency and Confidence of Rules

Now, we define the common rule performance measures *frequency* and *confidence*.

Definition 1: (Frequency and Confidence)
Let N be the number of all the samples and let R be a generated rule of class c with $\leq n$ attribute values b_k for variables v_k. Let $\sharp$ denote the number of elements of a set.
a) The **frequency** $\mathrm{freq}(R) \in [0,1]$ of R is the number of samples $P_j = (d_1, \ldots, d_n; \tilde{c})$ that induces rule R divided by N, i.e.

$$\mathrm{freq}(R) := \frac{\sharp\{P_j \,|\, \forall\, k \;\; (b_k \text{ is attribute value of } R \text{ and } b_k = d_k)\}}{N} \quad . \tag{1}$$

b) The **confidence with respect to c** $\mathrm{conf}(R,c) \in [0,1]$ of R is defined as the number of all the samples of class c inducing R divided by the number of samples P_j of any class that induces rule R,[1] i.e.

$$\mathrm{conf}(R,c) := \frac{\sharp\{P_j = (d_1, \ldots, d_n; c) \,|\, \forall\, k \;\; (b_k \text{ is attrib. value of } R \text{ and } b_k = d_k)\}}{\sharp\{P_j \,|\, \forall\, k \;\; (b_k \text{ is attribute value of } R \text{ and } b_k = d_k)\}} \; . \tag{2}$$

[1] Multiplied by 100 the measures could be interpreted as a percentage.

The aim of our rule generation process is now to find all the sufficient *frequent* and *confident* rules R, those with $\text{freq}(R) \geq \min_{\text{freq}}$ and $\text{conf}(R, c) \geq \min_{\text{conf}}$ using predefined thresholds $\min_{\text{freq}}$ and $\min_{\text{conf}}$. These thresholds must be high enough to provide interesting, significant rules and low enough to generate a sufficient number of rules. So an expert of the application area – in medical applications a physician – has to be involved in order to design proper thresholds for useful results.

To extract all the frequent patterns fast without scanning the database several times we use the FP-tree structure.

2.2 The FP-Tree Approach

For convenience of the reader we repeat shortly the ideas behind the basic algorithm with an example. We refer to [6] for more formal, extensive definitions, proofs and explanations. Basic knowledge of data structures for algorithms [9] is required. For our purpose we changed the algorithm slightly and added the last step 7.

1. Let us assume, our database D consists of three patient records, each with 3 variables: $P_1 = (\text{yes}, \text{high}, \text{low}; \text{deceased})$, $P_2 = (\text{yes}, \text{middle}, \text{low}; \text{deceased})$ and $P_3 = (\text{no}, \text{high}, \text{middle}; \text{survived})$. For technical reasons, encode the attribute values so that no attribute value of one variable is equal to one of another variable. For our example, we encode the attribute values with letters: $P_1 = (A, G, L; \text{deceased})$, $P_2 = (A, H, L; \text{deceased})$ and $P_3 = (B, G, M; \text{survived})$. We choose $\min_{\text{freq}} := 2/3$ and $\min_{\text{conf}} := 0.6$ for class "deceased".
2. For generating a FP-tree for class "deceased" in step 3, we have to count the frequency of all attribute values related to this class and order the attributes in a descending list. For our small database of three patients we get: (A:2), (L:2), (G:1), (H:1), (B:0), (M:0). Then we re-order the attributes in the samples with respect to this list: $P_1 = (A, L, G; \text{deceased})$, $P_2 = (A, L, H; \text{deceased})$ and $P_3 = (G, B, M; \text{survived})$.

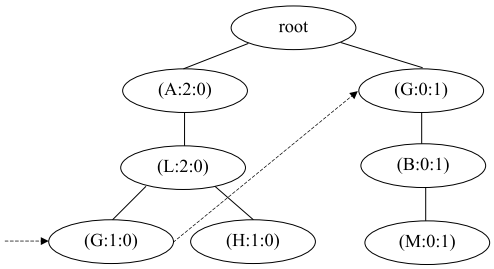

Figure 1. FP-tree for database D with node information. Here, only G's link list is shown (dotted arrows).

3. We build up a prefix-tree with an additional link list pointer for every attribute and attached basic information (attribute value, frequency count "deceased", frequency count "survived") to every node, see Fig. 1. This prefix-tree is called a **FP-tree**. So, the data base is very efficiently stored, avoiding redundancy (*database compression*). You can efficiently reconstruct all the samples and their frequencies for the class "deceased".

4. For every attribute construct a **conditional database**, i.e. walk through the link lists of the items and build up (from attribute node to root) lists of all prefix paths with maximal possible frequency for the classes "deceased" and "survived" in the path. For our example, this means for item A the path (<root>), for item L the path (<(A:2:0), root>), for G the paths: (<(L:1:0), (A:1:0), root>, <root>), for H: (<(L:1:0), (A:1:0), root>), for B: (<(G:0:1), root>) and for M: (<(B:0:1), (G:0:1), root>). After step 4 we have a set of conditional databases.

5. We chose $\min_{\text{freq}} = 2/3$ for class "deceased" in step 1, so without the trivial path <root> we only have to consider the path L: (<(A:2:0), root>), because all the other paths do not fulfill the frequency threshold condition (*frequency pruning*).

6. For all conditional databases – generated in step 4 – that contain more than one single path, build a (sub-)FP-tree using items and frequencies from the paths. Then, repeat steps 4 and 5 separately for every (sub-)FP-tree (recursion). For conditional databases with only a single path go on with step 7. In our simple example we need no recursion for the single conditional database L: (<(A:2:0), root>).

7. Calculate the confidence for combinations – that represent the rules – in each single path, i.e. for the resulting rules. For this purpose in our example we set $R_1 := (A, L)$, $R_2 := (A)$ and $R_3 := (L)$. Then, we have conf(R_1,deceased) = 1, conf(R_2,deceased) = 1, conf(R_3,deceased) = 1. Select confident rules with respect to $\min_{\text{conf}}$. Here, all three rules are confident. However, we generate longer rules only if the confidence is better, but this depends on the application. So only $R_2 = (A)$ and $R_3 = (L)$ - meaning "yes" resp. "low" – are given out, i.e. we have generated two rules R_2 "**if** var_1 = yes **then** class deceased" and R_3 "**if** var_3 = low **then** class deceased", both with confidence 1 for class "deceased" and frequency 2/3.

Of course, the whole procedure could be applied for class "survived". The algorithm performs efficiently if and only if there are a lot of common long subpatterns in most of the samples of the database, so that there is no explosion of recursive calls of the steps 4 and 5. Due to the fact that we have only binary variables and patients with a very individual behaviour the algorithm is not very performant, but until now it is one of the few algorithms that can find *all* frequent and confident rules in acceptable time. Other algorithms could be more inefficient due to a *combinatorical explosion* in the search space.

3 Results

Before we will present the results of our application of the FP-tree approach, we give a short description of the database with respect to preprocessing steps.

3.1 Data Preprocessing

Our database consists of 362 septic shock patients. Because it is often difficult to get a verification of an infection in time, this criteria was not presumed in the septic shock definition. The data of each patient was given as admission data (e.g. chronic diagnoses) and daily measurements (e.g. acute diagnoses, medicaments and therapies). We extracted binary values from *time series* in the following manner: If someone developed an organ failure during a stay at the hospital we set this binary variable to "true" for this patient. So the dynamical behaviour of the time series is lost; the analysis of dynamical behaviour of 30 synced variables is not yet possible. – Another problem are *missing values*: A maximum of 30, a minimum of 12 and a mean of 25 variables was available for every patient. For this reason some technical adaptations were necessary in the FP-tree algorithm. In fact, preprocessing of multivariate time series with missing values – the usual case in medical databases – is very time consuming but although very important [5], [10].

3.2 Generated Rules

Now, let us give examples of frequent and confident rules. We chose $\min_{\text{freq}} = 0.020$ and $\min_{\text{conf}} = 0.750$ for class "deceased". Because the database of survived patients is easier to describe with rules we set $\min_{\text{freq}} = 0.165$ and $\min_{\text{conf}} = 0.980$ for class "survived". We generated 1284 rules for class "deceased" and 9976 rules for class "survived" that are frequent and confident with respect to the thresholds. Two of the best rules for survival and death are listed below.

1) "**if** peritoneal lavage = no **and** thrombocyte concentrate = no **and** haemodialysis = no **then** class survived **with** confidence 0.980 **and** frequency 0.420"
2) "**if** haemofiltration = no **and** reoperation = no **and** acute renal failure = no **and** liver cirrhosis = no **then** class survived **with** confidence 0.990 **and** frequency 0.290"
3) "**if** minimal use of three different antibiotics = yes **and** artificial respiration = yes **and** tube feeding = no **then** class deceased **with** confidence 0.818 **and** frequency 0.030"
4) "**if** organ failure = yes **and** antiarrythmics = yes **and** haemodialysis = yes **and** peritoneal lavage = yes **then** class deceased **with** confidence 0.800 **and** frequency 0.028"

Although the rules are frequent and confident enough to indicate survival and death of patients and the rules have mostly conditions for only a few of the 30 variables, the number of rules is surely too high. A physician can not consider

all the rules for practical use. So future work have to be done to find a smaller rule basis that describes the patient data sufficiently.

4 Conclusion

Our aim was the extraction of information from categorical septic shock patient data. For this purpose we applied an efficient improved frequent patterns algorithm to generate frequent and confident rules. We obtained a lot of performant rules for the classes of deceased and survived patients. Such rules give good hints for physicians. The remaining problem is the high number of interesting rules – due to the individual behaviour of the patients – that have to be reduced technically with the support of experts to obtain a human understandable rule basis. Also it is desirable to combine an approach for categorical data with an approach for metric rule generation to build up a warning system. Finally, we plan to analyse multicenter data to get more representative results.

Acknowledgement: Our work was done within the DFG-project MEDAN (Medical Data Analysis with Neural Networks). The authors like to thank Mr. Slomka who implemented the FP-tree algorithm and all the participants of the MEDAN working group especially Prof. Hanisch.

References

1. Wade, S., Büssow, M., Hanisch, E.: Epidemiologie von SIRS, Sepsis und septischem Schock bei chirurgischen Intensivpatienten. Der Chirurg **69** (1998) 648–655
2. Schoenberg, M.H., Weiss, M., Radermacher, P.: Outcome of Patients with Sepsis and Septic Shock after ICU Treatment. Arch Surch **383** (1998) 44–48
3. Fein, A.M. et al. (eds.): Sepsis and Multiorgan Failure, Williams & Wilkins Baltimore (1997)
4. Hamker, F., Paetz, J., Thöne, S., Brause, R., Hanisch, E.: Erkennung kritischer Zustände von Patienten mit der Diagnose „Septischer Schock“ mit einem RBF-Netz. Interner Bericht 04/00, FB Informatik, J.W. Goethe-Univ. Frankfurt am Main, Germany (2000)
5. Paetz, J., Hamker, F., Thöne, S.: About the Analysis of Septic Shock Patient Data. 1st Int. Symp. on Medical Data Analysis (ISMDA). Frankfurt am Main, Germany. LNCS Vol. 1933. Springer-Verlag (2000) 130–137
6. Han, J., Pei, J., Yin, Y.: Mining Frequent Patterns Without Candidate Generation. ACM SIGMOD Int. Conf. on Management of Data. Dallas, USA (2000) 1–12
7. Agrawal, R., Skrikant, R.: Fast Algorithms for Mining Association Rules. 20th Int. Conf. on Very Large Databases (VLDB). Santiago de Chile, Chile (1994) 487–499
8. Brause, R., Langsdorf, T., Hepp, M.: Neural Data Mining for Credit Card Fraud Detection. 11th IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI). Chicago, USA (1999) 103–106
9. Sedgewick, R.: Algorithms in C. Addison Wesley (1992)
10. Tsumoto, S.: Clinical Knowledge Discovery in Hospital Information Systems: Two Case Studies. 4th European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD). Lyon, France. LNAI Vol. 1704. Springer-Verlag (2000) 652–656