Parameter-free association rule mining with *yacaree*

José L Balcázar

Departamento de Matemáticas, Estadística y Computación
Universidad de Cantabria, Santander, Spain
joseluis.balcazar@unican.es

1 Introduction

Association rule mining is one of the most central aspects of data mining. There are many implementations of association miners: Borgelt (2003); Witten and Frank (2005); Zaki and Hsiao (2005). The problems faced in association rule mining are twofold. First, the quantity of candidate itemsets potentially leading to rules grows exponentially with the often already large universe of items. The introduction of a support parameter was a key advance that allowed for the design of efficient frequent set miners and for the computation of association rules in large datasets: there, exploration is limited to those itemsets that appear "often enough" as subsets of the transactions, that is, their relative frequency exceeds a certain ratio of the transactions (Agrawal et al. (1996)). Then, the second problem is that, often, the set of rules provided as output is huge, specially if we consider that its purpose is to be read by a human.

Many studies of notions of redundancy exist that limit the output to "irredundant" rules, according to several existing notions of redundancy (e.g. Zaki (2004); Kryszkiewicz (2001)); but even taking redundancies into account, the results are, in many cases, unsatisfactory: high implication thresholds leave out many interesting rules, whereas lower ones let pass far too many rules to be manually inspected. Many alternative quality measures exist for association rules: Geng and Hamilton (2006); Lenca et al. (2008); Tan et al. (2004).

Our system *yacaree* (Yet Another Closure-based Association Rule Experimentation Environment) processes transactional datasets, each transaction being an itemset, and obtains irredundant rules $X \rightarrow Y$, where both X and Y can be arbitrary disjoint itemsets; yet, it does not require the user to select the value of any threshold parameter. As in most current proposals, we mine only frequent closed itemsets, and apply known redundancy filters. Our current closure mining algorithm is a simplified variant of ChARM (see Zaki and Hsiao (2005)), rather close to a depth-first search. As in some of the associators of Weka, we mine closures in order of decreasing support; see Witten and Frank (2005). However, our algorithm is very different: it requires no "delta" parameter for stepwise support threshold decrease as Weka's "Apriori" does, and does not relate support to confidence as "predictive Apriori" does (Scheffer (2005), also present in Weka). Instead, we self-adjust the internal effective support bound on the basis of technological limitations: it starts at an almost trivial level, and grows, if necessary, as the monitorization of the mining process reveals that the memory consumption surpasses internal thresholds. The closures are passed on to a "border" algorithm which computes the lattice structure, and then irredundant rules are extracted.

Second, we implement a novel implication quality process. Our option is as follows: we impose a very mild confidence threshold that remains fixed, letting large quantities of rules pass; but we control the number of rules to be provided to the user via a relative, rather than absolute, confidence bound, measured by the parameter called "closure-based confidence boost" (Balcázar (2010)), related both to the lift and to the "support ratio" (introduced in Kryszkiewicz (2001), without giving it a name, as one of the properties used to accelerate a computation of the representative basis of association rules); these connections are crucial for our system: we use the approximation to the confidence boost provided by the support ratio to push the confidence boost constraint into the mining process, and we use the lift, applied to particular cases where we can prove a strong connection with the boost, to self-adjust the boost threshold.

All our components are joined together through lazy evaluation using the functional programming facilities of Python: closures, Hasse diagram edges, predecessors of a closure, and rules are obtained from iterators. Closures and candidate rules are either discarded, if we can guarantee that future threshold adjustments will never recover them; or processed, if they obey the thresholds; or maintained separately on hold, if they fail the current thresholds but might obey them after later adjustments. Thus, the memory expense is controlled by maintaining only those pieces of information that can be relevant for the current or future threshold values.

The result is a functional preliminary system, where ample room still remains for efficiency and algorithmic improvements, which shows that it is possible to find interesting association rules in a fully autonomous manner: the user simply selects a dataset and launches the process, which takes just one to five minutes in many easy datasets, and up to ten to twenty minutes on a modern laptop for a few difficult, highly dense datasets. The output is a set of rules which, in most cases, is reasonably small and shows independent and sensible associations. The screenshot provided in Figure 1 shows the simple interface (button "Run" is disabled as the system has been just run) and the two text files generated: the log, where we can see that the process took a bit over five minutes, and the start of the file containing the rules found. Both the console and the log indicate the self-adjustments of the support; no adjustment was performed on the boost threshold, as enough high-boost rules were found for its initial value.

See `http://sourceforge.net/projects/yacaree/` for further information.

References

Agrawal, R., H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo (1996). Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pp. 307–328. AAAI/MIT Press.

Balcázar, J. L. (2010). Closure-based confidence boost in association rules. In *Workshop on Applications of Pattern Analysis*, Volume 11, pp. 74–80. JMLR Workshop and Conference Proceedings.

Borgelt, C. (2003). Efficient implementations of apriori and eclat. In B. Goethals and M. J. Zaki (Eds.), *FIMI*, Volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Geng, L. and H. J. Hamilton (2006). Interestingness measures for data mining: A survey. *ACM Comput. Surv. 38*(3).

Kryszkiewicz, M. (2001). Closed set based discovery of representative association rules. In F. Hoffmann, D. J. Hand, N. M. Adams, D. H. Fisher, and G. Guimarães (Eds.), *Proc. of the*

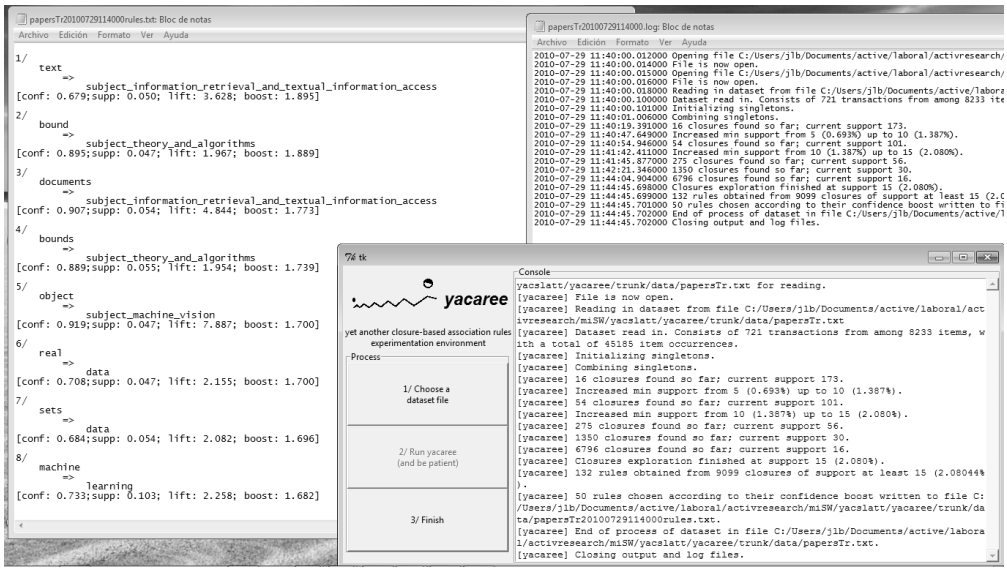

FIG. 1 – *A screenshot of* yacaree *with the rules and log output files*

4th International Symposium on Intelligent Data Analysis (IDA), Volume 2189 of *Lecture Notes in Computer Science*, pp. 350–359. Springer-Verlag.

Lenca, P., P. Meyer, B. Vaillant, and S. Lallich (2008). On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid. *European Journal of Operational Research 184*(2), 610–626.

Scheffer, T. (2005). Finding association rules that trade support optimally against confidence. *Intelligent Data Analysis 9*, 293–313.

Tan, P.-N., V. Kumar, and J. Srivastava (2004). Selecting the right interestingness measure for association patterns. *Inf. Syst. 29*(4), 293–313.

Witten, I. H. and E. Frank (2005). *Data Mining: Practical Machine Learning Tools and Techniques (2ed)*. Morgan Kaufmann.

Zaki, M. J. (2004). Mining non-redundant association rules. *Data Min. Knowl. Discov. 9*(3), 223–248.

Zaki, M. J. and C.-J. Hsiao (2005). Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering 17*(4), 462–478.

Résumé

We describe our program *yacaree* that mines irredundant association rules, in usually reasonable quantities, yet it does not require the user to choose any value for any parameter. Instead, it fixes values for some parameters and runs a dataset-driven self-adjustment of the two main ones: support and a variant of relative confidence studied recently by the present author.