The Application of K-medoids and PAM to the Clustering of Rules

A. P. Reynolds, G. Richards, and V. J. Rayward-Smith

School of Computing Sciences,
University of East Anglia,
Norwich

Abstract. Earlier research has resulted in the production of an 'all-rules' algorithm for data-mining that produces all conjunctive rules of above given confidence and coverage thresholds. While this is a useful tool, it may produce a large number of rules. This paper describes the application of two clustering algorithms to these rules, in order to identify sets of similar rules and to better understand the data.

1 Introduction

Data mining often involves the production of accurate yet simple rules that describe records of interest within a database. For example, in a motor insurance database, the records of interest may be those motorists who have claimed on their insurance. Often, rules produced are conjunctions of simple clauses, where each clause applies to one field within the database (e.g. [2]). For example,

`if age` $<$ `25 and car_type = sports then claim = yes`

is such a rule.

Although this approach produces useful rules, these rules may describe only small subsets of the records of interest. To describe larger subsets accurately, collections of such rules are required. B. de la Iglesia et al. [3] have researched the application of multi-objective genetic algorithms to data mining, using confidence and coverage as two separate objectives. This approach leads to the production of a collection of Pareto-optimal rules. Richards et al. [1] have created software that produces *all* the rules, above given confidence and coverage thresholds, that apply to the data.

This all-rules algorithm can produce many thousands of rules. For example, on the 'adult' database [4], seeking rules of greater than 15% coverage and greater than 60% confidence results in 4785 rules. Examining these rules by eye immediately reveals that many are very similar. Others, while appearing to be different, may in fact match similar sets of records. In order to determine which rules match similar sets of records, we applied two clustering algorithms to the rules produced by the all-rules algorithm: k-medoids and Partitioning About Medoids (PAM).

2 Rule Dissimilarity

Let D be the database from which rules of the form $\alpha \rightarrow \beta$ are deduced. If ρ is such a rule then the set of support for ρ, $S(\rho)$, is the set of records in D on which the rule holds, i.e.

$$S(\rho) = \{r \in D : \alpha(r) \wedge \beta(r)\}.$$

If ρ_1 and ρ_2 are two arbitrary rules, we can define a dissimilarity measure

$$\begin{aligned} d(\rho_1, \rho_2) &= |S(\rho_1) \cup S(\rho_2)| - |S(\rho_1) \cap S(\rho_2)| \\ &= |S(\rho_1) - S(\rho_2)| + |S(\rho_2) - S(\rho_1)| \,. \end{aligned}$$

Dividing this measure by the number of records in the database gives the simple matching coefficient [5]. Alternatively, we can use the Jaccard coefficient [6] on the sets of support and define

$$n(\rho_1, \rho_2) = d(\rho_1, \rho_2) / \, |S(\rho_1) \cup S(\rho_2)| \,.$$

Both of these dissimilarity measures are pseudometrics [7].

Intuitively, two rules that are mutually supported by a thousand records and differ over only six are more similar than two rules that are mutually supported by no records and differ over five. For this reason, the results described in this paper are those obtained using the Jaccard dissimilarity coefficient.

3 K-means

The k-means algorithm [8] is a well known technique for performing clustering on objects in $\mathbb{R}^n$. Each cluster is centred about a point called the centroid, where the centroid's coordinates are the mean of the coordinates of the objects in the cluster. K-means can be applied to the clustering of rules but there are two reasons for not doing so. Firstly, although each rule may be represented as a binary string, centroids will not have this structure and will make little sense as rules. More importantly, the k-means algorithm requires that distances to the centroids be calculated for each object at each iteration. Since the original database may have tens of thousands of records, such calculations will be slow.

4 K-medoids

The k-medoids algorithm, as used to produce the results of this paper, is an adaptation of the k-means algorithm. Rather than calculate the mean of the items in each cluster, a representative item, or medoid, is chosen for each cluster at each iteration. Medoids for each cluster are calculated by finding object i within the cluster that minimizes

$$\sum_{j \in C_i} d(i, j),$$

where C_i is the cluster containing object i and $d(i, j)$ is the distance between objects i and j.

There are two advantages to using existing rules as the centres of the clusters. Firstly, a medoid rule serves to usefully describe the cluster. Secondly, there is no need for repeated calculation of distances at each iteration, since the k-medoids algorithm can simply look up distances from a distance matrix.

The k-medoids algorithm can be summarized as follows:

1. Choose k objects at random to be the initial cluster medoids.
2. Assign each object to the cluster associated with the closest medoid.
3. Recalculate the positions of the k medoids.
4. Repeat Steps 2 and 3 until the medoids become fixed.

Step 3 could be performed by calculating $\sum_{j \in C_i} d(i, j)$ for each object i from scratch at each iteration. However, many objects remain in the same cluster from one iteration of the algorithm to the next. Improvements in speed can be obtained by adjusting the sums whenever an object leaves or enters a cluster.

Step 2 can also be made more efficient in terms of speed, for larger values of k. For each object, an array of the other objects, sorted on distance, is maintained. The closest medoid can be found by scanning through this array until a medoid is found, rather than comparing the distance of every medoid.

Figure 1 shows a comparison of the time requirements for each version of the

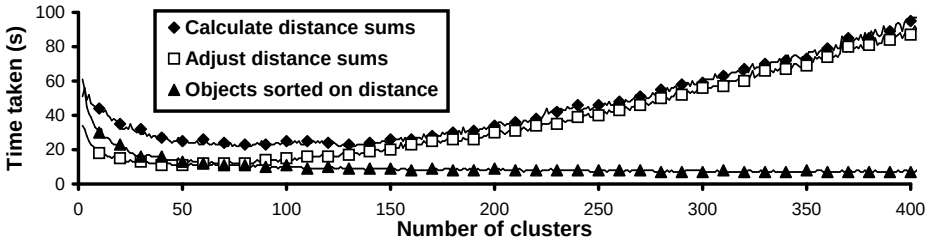


Fig. 1. Time requirements of the k-medoids algorithm.

code. The first series shows the requirements of the naive implementation. The second shows the speed improvements obtained by adjusting the sums required in step 3, rather than recalculating them. The third shows the extra improvements obtained by maintaining, for each object, an array of the other objects sorted on distance. Each point in the graph indicates the time required to run the k-medoids algorithm 100 times on 4785 rules extracted from the 'adult' database, using a AMD Athlon XP 1800+ (1.53GHz) with 480Mb of RAM.

5 Partitioning About Medoids (PAM)

Partitioning about medoids (PAM) also clusters objects about k medoids, where k is specified in advance. However, the algorithm takes the form of a steepest

ascent hillclimber, using a simple swap neighbourhood operation. In each iteration medoid object i and non-medoid object j are selected that produce the best clustering when their rôles are switched. The objective function used is the sum of the distances from each object to the closest medoid.

As this search phase of the algorithm is slow, the initial set of medoids is constructed in a greedy build phase. Starting with an empty set of medoids, objects are added one at a time until k medoids have been selected. At each step, the new medoid is selected so as to minimize the objective function.

6 Silhouettes

Kaufman and Rousseeuw [5] suggest the use of *silhouettes* both to determine which objects lie well within their clusters and which do not and also to judge the quality of the clustering obtained. For each object i, let $a(i)$ be the average distance of i from all other objects in cluster C_i. For every other cluster $C \neq C_i$, let $d(i, C)$ be the average distance of i from the objects in C. After computing $d(i, C)$ for all clusters $C \neq C_i$, let $b(i)$ be the smallest. The cluster for which this minimum is attained is called the *neighbour* of i.

The number $s(i)$ is given by

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

and provides a measure of how well object i fits into cluster C_i rather than the neighbouring cluster. If $s(i)$ is close to one, object i can be said to be 'well classified'. If $s(i)$ is close to zero, it is unclear whether i should belong to cluster C_i or to its neighbour. A negative value suggests that i has been misclassified.

The following summary values are also defined:

- The mean of $s(i)$ for all objects i in a cluster is called the *average silhouette width* of that cluster.
- The mean of $s(i)$ for all the objects is called the *average silhouette width for the entire data set* and is denoted by $\bar{s}(k)$, where k is the number of clusters.

Kaufman and Rousseeuw suggest that $\bar{s}(k)$ can be used for the selection of a best value of k, by choosing that k for which $\bar{s}(k)$ is maximal.

7 Results

We applied these two clustering algorithms, with small values of k, to the 4785 rules derived from the 'adult' database. Figure 2 shows the sum of distances from rules to medoids. Figure 3 shows the silhouette widths for the clusterings. PAM typically outperforms the k-medoids algorithm according to both measures.

The results presented are for small values of k, since we are endeavouring to simplify our understanding of the data. On the basis of silhouette width, PAM obtains the best clusterings when k equals two or seven. In both cases, the

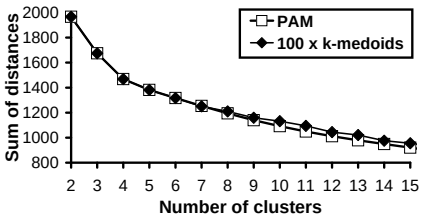


Fig. 2. Sum of the distances of rules to the closest medoids.

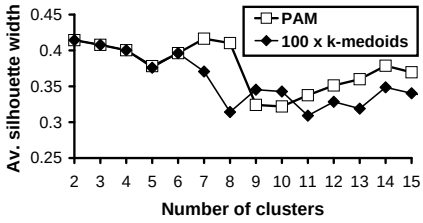


Fig. 3. Silhouette widths of clusterings obtained from PAM and k-medoids.

strongest cluster contains all 1433 rules with a clause of the form '`Cap_gain >=` x'. This cluster has an average silhouette width of 0.74 in both cases and has '`IF Age >= 32 AND Cap_gain >= 3103 THEN *Salary = >50K`' as its medoid. In the two cluster case, the other 3352 rules appear in the second cluster, which has an average silhouette width of 0.27. In the seven cluster case, these rules are split into six clusters, with average silhouette widths from 0.07 to 0.37.

It is unclear from figures 2 and 3 whether better clusterings might be obtained for larger values of k. The time requirements for PAM increase considerably as the number of clusters increases, as shown in figure 4, but figure 1 shows that the

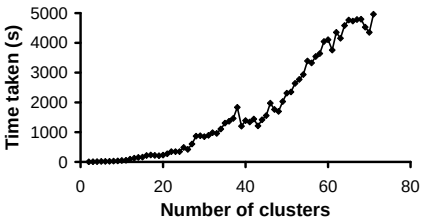


Fig. 4. Time requirements for the PAM algorithm.

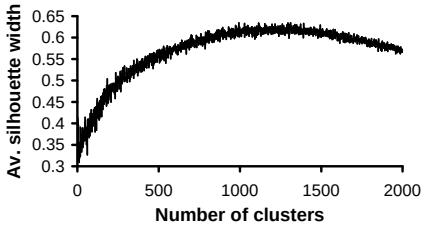


Fig. 5. Results of 100 runs of the k-medoids algorithm for k up to 2000

k-medoids algorithm may be used with much larger values of k. Figure 5 shows the silhouette widths of clusterings obtained for values of k up to 2000. These results suggest that the best clusterings are obtained for values of k around 1200. However, such large values of k do not adequately simplify the data.

8 Conclusions and Further Research

The PAM and k-medoids algorithms have been applied to the clustering of rules obtained from the all-rules algorithm. PAM obtains better results according to both the sum of distances to medoids and the overall silhouette width for most values of k. However, it requires considerably more time than k-medoids for larger values of k. The clusters obtained seem to be reasonable, especially the

cluster of all rules containing a clause of the form '`Cap_gain` $\geq x$', which is the only strong cluster in the two and seven cluster cases.

There are a number of potential areas for further research:

- Experimentation with other medoid based clustering algorithms, such as CLARANS [9], may result in improved clusterings and efficiency. The application of heirarchical clustering algorithms [5] to the clustering of rules might also reveal useful information.
- Better methods for summarizing clusters that give more information than just the cluster medoid are required if we are to improve our understanding of the data via clustering of rules.
- Numeric fields lead to the production of many similar rules differing only in the bounds on these fields. Every rule has the same weight when applying k-medoids or PAM, so if essentially the same rule is produced many times a bias is created towards clusters centred about such rules. More interesting clusters might be found if such rules were given a lesser weight or if the all-rules algorithm were prevented from producing almost identical rules.
- Our difference measures are 'semantic-based', i.e. related to the sets of support. An alternative strategy would be 'syntactic-based', i.e. based on the actual expressions used in the rules. A comparison of the clusters obtained using the two strategies is a matter for further research.

References

1. G. Richards and V. J. Rayward-Smith. Discovery of association rules in tabular data. In *Proceedings of IEEE First International Conference on Data Mining, San Jose, California, USA*, pages 465–473. IEEE Computer Society, 2001.
2. R. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-Based Rule Mining in Large, Dense Databases. In *Proceedings of the 15th International Conference on Data Engineering*, pages 188–197. IEEE, 1999.
3. B. de la Iglesia, M. S. Philpott, A. J. Bagnall, and V. J. Rayward-Smith. Data Mining Rules Using Multi-Objective Evolutionary Algorithms. In *Proceedings of 2003 IEEE Congress on Evolutionary Computation*, 2003.
4. C.L. Blake and C.J. Merz. UCI Repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.
5. Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley series in probability and mathematical statistics. John Wiley and Sons Inc., 1990.
6. P. Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin de la Société Vaudoise de la Sciences Naturelles*, 37:547–579, 1901.
7. J. C. Gower and P. Legendre. Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3:5–48, 1986.
8. J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.
9. Raymond T. Ng and Jiawei Han. CLARANS: A Method for Clustering Objects for Spatial Data Mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003–1016, 2002.