Mining Clusters with Association Rules

Walter A. Kosters, Elena Marchiori and Ard A.J. Oerlemans*

Leiden Institute of Advanced Computer Science
Universiteit Leiden
P.O. Box 9512, 2300 RA Leiden, The Netherlands
{*kosters, elena, aoerlema*}*@cs.leidenuniv.nl*

Abstract

In this paper we propose a method for extracting clusters in a population of customers, where the only information available is the list of products bought by the individual clients. We use association rules having high confidence to construct a hierarchical sequence of clusters. A specific metric is introduced for measuring the quality of the resulting clusterings. Practical consequences are discussed in view of some experiments on real life datasets.

1 Introduction

The essence of clustering in databases is to identify homogeneous groups of objects based on the values of their attributes. Various clustering techniques have been proposed (e.g., [6, 16]). In particular, in the database community several systems have been introduced for clustering of data in large databases, see [8]. One can distinguish two main classes of clustering techniques: partitional and hierarchical clustering. In partitional clustering ([11, 13, 17]) objects are grouped into disjoint clusters such that objects in a cluster are more similar to each other than to objects in other clusters. For instance, the

*Published in Advances in Intelligent Data Analysis (IDA-99) (D.J. Hand, J.N. Kok and M.R. Berthold, Eds.), pp. 39–50, Lecture Notes in Computer Science 1642, Springer, 1999.

well-known K-means and K-medoid methods determine K cluster representatives and assign each object to the cluster with its representative closest to the object in such a way that the sum of the distances between the objects and their representatives is minimized. Hierarchical clustering on the other hand is a nested sequence of partitions. In the bottom-up method, larger and larger clusters are built by merging smaller clusters, starting from atomic clustering where each object forms a cluster on its own. In the top-down method however one starts with one cluster containing all objects and constructs a subdivision of the cluster into smaller pieces, e.g., [10].

In this paper we introduce a top-down hierarchical clustering method for finding clusters in a population of customers, where the only information available is the list of products bought by the individual clients. The technique as well as the metric we use are tailored for this specific class of data. However, the technique gives satisfactory results also when applied to the more general problem of finding clusters in a set of itemsets consisting of a sequence of binary attributes. In [11] some theoretical questions for this setup are addressed, such as computational complexity (NP-completeness) of possible embeddings in k-dimensional spaces, and the associated clusterings.

The idea is to use association rules for mining clusters. These rules relate groups of customers, and are of the form "80% of the customers that buy products A, B and C, buy product D too". Association rules (cf. [2]) are formalized by means of implication rules augmented with two parameters which describe their quality: the support which measures the frequencies of the products occurring in the rule, and the confidence which denotes the strength of the implication. A hierarchical clustering can be built using the following top-down method. First, association rules having support above a certain threshold are generated, using the efficient `Apriori` technique from [3]. Next, the "best" association rule is selected, where the selection criterion may depend on the number of products occurring on the left-hand side of the rules, as well as on the confidence and support of the rules. Finally, a cluster is constructed consisting of all the customers buying the products that occur in the left-hand side of the rule. The data set is then modified by removing the elements of that cluster. This procedure is iterated until a suitable stopping criterion is satisfied. Note that a small threshold for the support may bias the search towards clusters containing few, but strongly related clients, whereas a high support threshold allows one to construct larger clusters sharing less products.

In order to assess the effectiveness of this clustering method, we have conducted experiments on benchmark data sets from the literature, as well as on two real life datasets. The real life data sets contain different kinds of itemsets: in the first data set, items describe a small number of products, whereas in the second one items describe many products. The results of the experiments indicate that the success of this technique depends on the structure of the items in the data sets. If the set of possible products is large, whereas customers buy relatively few products, the association rules tend to be of lower confidence. The corresponding clustering may not be very informative in this case; however, the clusterings still make sense. On the other hand, if the customers buy (relatively) more products, for instance if the number of products is small, association rules tend to be more reliable—which holds for the implied clustering too. For measuring the quality of a cluster and comparing clusters, we introduce a metric for the space of customers which takes into account the specific structure of the itemsets.
The paper is organized as follows. In Section 2 we give some terminology on association rules and introduce an appropriate metric. Section 3 is devoted to a description of our method. In Section 4 we present some results from experiments. We conclude with a discussion.

2 Preliminaries

In this section we define the terminology and concepts that are used throughout the paper.

2.1 Discovering Association Rules

Suppose that we have n customers and a set $\mathcal{S}$ consisting of m products. Every customer i buys a subset $\mathcal{S}_i \subseteq \mathcal{S}$ of these products. The only information that is used for extracting regularities about the customers is the collection $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$. An *association rule* is a rule of the form $\mathcal{R} \Rightarrow \mathcal{T}$ for disjoint subsets $\mathcal{R}$ and $\mathcal{T}$ of $\mathcal{S}$, where $\mathcal{T} \neq \emptyset$. Customer i is said to *satisfy* this rule if and only if $\mathcal{R} \cup \mathcal{T} \subseteq \mathcal{S}_i$. The *support* of this rule is the number of customers that satisfy the rule, divided by the total number of customers. The *confidence* of a rule is the number of customers that satisfy it, divided by the number of clients i with $\mathcal{R} \subseteq \mathcal{S}_i$ (the confidence is set to zero if the denominator is zero). If $\mathcal{R} \cup \mathcal{T}$ has k elements, we say that the association rule has *order* k. In this paper we restrict ourselves to association rules whose right-hand side contains only one element.

Association rules having both large support and confidence can be constructed using simple algorithms. However, if the database is very large, efficient methods are necessary (see [2, 3, 4, 14]), like the well-known `Apriori` algorithm. This algorithm is based on the construction of subsets of $\mathcal{S}$ that are present in many customers, joining them to find ever larger subsets. These subsets are called large k-itemsets where k denotes their cardinality. Given minimum threshold $s\%$ for the support, the algorithm starts by constructing 1-itemsets having support greater than or equal to $s\%$. Then a large $(k+1)$-itemset is generated by merging two k-itemsets having exactly $(k-1)$ elements in common, and checking that its support is greater than or equal to $s\%$.

Once all large itemsets are generated, association rules can be easily derived as follows: suppose $\{A, B, C, D\}$ and consequently $\{A, B, C\}$ are large itemsets, then the rule $A, B, C \Rightarrow D$ (note that for simplicity we omit $\{$ and $\}$) can be derived. Clearly, any k-itemset gives rise to k association rules of order k.

2.2 A Metric for the Space of Customers

We turn the space of customers (i.e., subsets of $\mathcal{S}$) into a metric space by means of the following distance measure. For subsets $\mathcal{R}$ and $\mathcal{T}$ of $\mathcal{S}$ we define

$$d(\mathcal{R}, \mathcal{T}) = \frac{|\mathcal{R} \backslash \mathcal{T}| + |\mathcal{T} \backslash \mathcal{R}|}{|\mathcal{R} \cup \mathcal{T}| + 1}$$

In this formula $\backslash$ denotes the set-theoretic difference ($\mathcal{X} \backslash \mathcal{Y}$ consists of those elements from $\mathcal{X}$ that are not in $\mathcal{Y}$), and $|\mathcal{X}|$ denotes the number of elements of $\mathcal{X}$. So the numerator is the number of elements in the symmetric difference of $\mathcal{R}$ and $\mathcal{T}$. This is the well-known Hamming distance when the list of products bought by a client is characterized by means of a string of bits whose length is equal to m, and where the i-th entry is 1 if the customer bought the i-th product, and 0 otherwise. The denominator in the definition of d is added in order to compensate for the size of the two sets. If for instance two customers differ in exactly one product, their distance is $2/(k+3)$, where k is the number of products bought in common. So their distance decreases as the number of common purchases increases. This allows for judging the distance between customers also in terms of the number of products they

bought. The $+1$ in the denominator of the formula for the distance d is added to deal with the case $\mathcal{R} = \mathcal{T} = \emptyset$, but may also be omitted if one defines $d(\emptyset, \emptyset) = 0$. This approach leads to almost the same metric.
Note that $0 \leq d(\mathcal{R}, \mathcal{T}) \leq 1 - 1/(n+1) < 1$ for all subsets $\mathcal{R}$ and $\mathcal{T}$ of $\mathcal{S}$. (If necessary the measure can be renormalized by multiplying it by a suitable fixed factor.) Of course $d(\mathcal{R}, \mathcal{T}) = d(\mathcal{T}, \mathcal{R})$ for all subsets $\mathcal{R}$ and $\mathcal{T}$ of $\mathcal{S}$, and $d(\mathcal{R}, \mathcal{R}) = 0$. Finally the triangular inequality holds:

$$d(\mathcal{R}, \mathcal{T}) \leq d(\mathcal{R}, \mathcal{U}) + d(\mathcal{U}, \mathcal{T})$$

for all subsets $\mathcal{R}$, $\mathcal{U}$ and $\mathcal{T}$ of $\mathcal{S}$; this can be verified by some tedious calculations. Indeed, put $|\mathcal{R} \cap \mathcal{U} \cap \mathcal{T}| = a$, $|\mathcal{R} \cap \mathcal{U} \cap \overline{\mathcal{T}}| = b$, and so on; here $\overline{\mathcal{T}}$ denotes the complement of the subset $\mathcal{T}$ in $\mathcal{S}$. Now substitute these numbers in the inequality to be proved, remove the denominators and carefully check the remaining abundance of terms. We may conclude that d is a metric on the space of customers.
A cluster can be defined as a set of customers that are more near to each other than to clients in other clusters with respect to the distance d. Note that the construction of clusterings is biased towards customers buying many products. Therefore, when we encode the information of a customer as a string of bits, strings containing many ones are more liable to form a cluster. Also notice that the measure can be used for classification tasks, where the class of an item is defined to be the nearest cluster.

3 Association Rules Infer Clusters

Suppose that we have association rules of order 1, 2, 3 and so on. Now fix a minimum support threshold of say $s\%$. We consider the association rules having highest possible order, since they represent dependency among a larger set of products. These rules can be obtained by considering only the largest k-itemsets generated by the `Apriori` algorithm. The minimum support s has to be rather small for ensuring the existence of these rules. However, s should not be too small in order to avoid the generation of many rules which are satisfied by few customers. In the experiments we have conducted, the values of s have been chosen after tuning the algorithm on each specific dataset.
Once the association rules of highest order have been generated, we select the one with the highest confidence; if there exist more rules attaining this

maximum, we choose one of them in a random way. We refer to the selected rule as $rule_1 = (\mathcal{R}_1 \Rightarrow \mathcal{T}_1)$. Now all customers that bought products in $\mathcal{R}_1$ constitute $cluster_1$. This means to include into the cluster not only the customers that satisfy the rule, but also those customers that bought all products from $\mathcal{R}_1$ but not those from $\mathcal{T}_1$. Because we consider rules having high confidence, it is expected that these extra customers are similar to those satisfying the rule. Next, we remove the customers occurring in $cluster_1$ from the original dataset. The process is iterated a suitable number of times, leading to a hierarchical clustering $cluster_1$, $cluster_2$, $cluster_3$, ... The termination condition we have used in the experiments consists of stopping when either a maximum number of clusters is generated (this maximum is given as an input parameter) or when the generated association rules do not reach the minimum support threshold.
The algorithm is illustrated below:

```
1   s := minimum support; i := 1;
2   Data := { all objects in the dataset }; Clust := ∅ ;
3   while (not termination-condition) do
4       H := { association rules over Data
              having maximum order and support ≥ s };
5       best := rule from H with highest confidence;
6       cluster_i := { objects containing products in LHS(best) };
7       Data := Data \ cluster_i ; Clust := Clust ∪ {cluster_i};
8       i := i + 1;
9   od
```

Here LHS(*best*) denotes the set of elements on the left-hand side of the association rule *best*. The core of the algorithm consists of the statements in lines 4 and 5. In line 4 the association rules of maximum order are generated, by considering the objects in the dataset *Data*. Note that *Data* is initially equal to the original dataset, but it becomes smaller and smaller at each iteration (line 7). The generated cluster is inserted into the actual clustering *Clust* (assuming that this is an ordered set with respect to insertion) and the process is repeated on the smaller dataset *Data* obtained by removing the objects within the cluster.
In this way we obtain a partitioning of the set of customers into clusters. As the results of the experiments will show, the sequence of clusters has the property that the first clusters that are built are of good quality, while clusters that are generated later on may become less informative. The quality

of the cluster is here only determined by considering the average distance of its elements and the confidence of the corresponding association rule. In this study we do not take into account other measures like the cluster diameter, i.e., the maximum distance between any two points of the cluster.

4 Experiments

For the experiments we used the so-called "Zoo Database", the artificial "LED Database" (both available from the Internet, see [15]), and two real life datasets generated from actual shop sales. The first two data sets are used for illustrating the effectiveness of our simple method in the case of classification problems. The other two datasets are used for illustrating the usefulness of the method for finding regularities in larger real life datasets—our original goal.

4.1 The Zoo Database

The zoo database contains 15 boolean attributes for $n = 101$ animals. In addition, there is one six-valued numeric attribute: the number of legs. For our algorithm to work it was necessary to turn this attribute into a boolean one, either by stating that "legs > 2 is equivalent to True" (or to False) or by introducing a boolean attribute for every possible numeric value. Here we choose the second option, the first one leading to almost identical results. So we have $m = 21$ "products": the terminology customer-product has to be interpreted as animal-attribute in this section. The original dataset also contains a classification of the animals into seven classes, referred to as A (41 mammals), B (20 birds), C (5 reptiles), D (13 fishes), E (4 amphibians), F (8 insects) and G (10 molluscs). The mean distance within the entire dataset is 0.577, which indicates that the attributes have several dependencies. In our experiments we discard information about the class to which an animal belongs.

Figure 1 shows the results of some experiments. We considered three runs with different minimum threshold for the support: 4%, 10% and 40%. The maximum number of clusters to search for was set to 10. In the column "class contents" we mention the classes of the animals in the clusters (between brackets the number of animals of each class within the cluster). We also included the mean Hamming distance, since it might be a better measure for

support	cluster	number of animals	mean distance	mean Hamming distance	class contents	confidence
4%	1	12	0.030	0.333	$A(12)$	100%
	2	5	0.040	0.400	$A(5)$	100%
	3	6	0.000	0.000	$A(6)$	100%
	4	5	0.095	1.000	$B(5)$	100%
	5	9	0.074	0.778	$D(9)$	100%
	6	5	0.290	3.600	$A(5)$	100%
	7	11	0.123	1.200	$B(11)$	100%
	8	5	0.167	1.600	$A(5)$	100%
	9	5	0.149	1.400	$A(5)$	100%
	10	6	0.250	3.000	$A(2), D(4)$	100%
10%	1	12	0.030	0.333	$A(12)$	100%
	2	15	0.120	1.200	$A(15)$	100%
	3	16	0.158	1.608	$B(16)$	100%
	4	13	0.121	1.231	$D(13)$	100%
	5	11	0.345	4.036	$A(11)$	100%
	6	13	0.416	4.692	$A(1), B(4), C(4), E(4)$	100%
	7	21	0.614	5.895	$A(2), C(1), F(8), G(10)$	80%
40%	1	41	0.253	2.971	$A(41)$	100%
	2	42	0.451	5.573	$B(20), C(5), D(13), E(4)$	97%

Figure 1: Experimental results for the zoo database.

this database—the number of ones being relatively high.
The first and second run show that the classes are well separated; if the mean distance within the clusters gets higher, more classes may occur in the same cluster. The hierarchical nature of the clustering is also apparent. Note that when the support threshold is high, not all items are clustered in the end: not even rules of type "$\Rightarrow$ attribute" obtain this threshold anymore, just because the number of remaining animals is too small. The animals not clustered are exactly those from classes F and G. It would of course be possible to lower

the threshold during the run, thus giving smaller clusters the opportunity to be discovered.
The rules found were of high order and confidence; for cluster 1 in the first and second run (it happens to be the same rule) the rule has order 9, for cluster 2 it has order 8 and 7, respectively. For the third run the first rule has order 3: "milk, breathes $\Rightarrow$ backbone"; the second rule is: "backbone $\Rightarrow$ eggs", so the animals not having a backbone remain unclustered. The mammal in cluster 6 of the second run is a platypus, the two mammals in cluster 7 are a dolphin and a porpoise; both classifications make some sense, at least for a non-biologist. The mean distances within the clusters are very small, see for instance cluster 3 in the first run. This also reveals some of the nature of the database.

4.2 The LED Database

The LED database is an artificial database, where each item corresponds with the "seven bit LED encoding" of one of the ten numbers $0, 1, 2, \ldots, 9$. Noise is introduced by appending at the end of the original string (with seven bits) a sequence of fixed length consisting of randomly chosen bits, as well as by corrupting some of the bits in the original string. The LED encoding shows which LED's out of seven are on in each case, for instance the topmost horizontal LED is activated for the numbers 0,2,3,5,7,8 and 9. This database is particularly interesting, since the noise may be added in such a way that there are lots of zeroes—a property also present in the real life data sets we use in the sequel.
As a typical example (see Figure 2) we added 93 random bits, giving a total of $m = 100$ bits; these extra bits had a 90% probability of being zero. We generated $n = 1000$ strings. The first experiment shows a situation where the original seven bits are not corrupted, whereas in the second run these bits had a 7% chance of being toggled (so on average 50% of the encoded strings contains a flaw). In the column "class contents" the classes of the strings, i.e., their numbers, in the clusters are given (between brackets the number of strings of each class within the cluster); a $*$ denotes elements different from the majority within the class. Observe that some encodings are quite similar (for instance those of 0 and 8), giving understandable faults in case of noise. The mean distance within the entire dataset is 0.764 and 0.778, respectively. The mean distance within the clusters is easily understood for the first experiment: the clustered strings may only differ in the 93 last bits, 9.3 of which

support	cluster	number of strings	mean distance	class contents	confidence
7%	1	101	0.630	8(101)	100%
	2	120	0.654	9(120)	100%
	3	99	0.643	0(99)	100%
	4	117	0.656	6(117)	100%
	5	79	0.689	5(79)	100%
	6	92	0.693	2(92)	100%
	7	93	0.690	3(93)	100%
	8	95	0.716	4(95)	100%
	9	96	0.750	7(96)	100%
	10	108	0.800	1(108)	100%
7%	1	79	0.641	$8(58), *(21)$	91%
	2	83	0.669	$6(78), *(5)$	93%
	3	88	0.660	$9(67), *(21)$	88%
	4	90	0.703	$2(70), *(20)$	95%
	5	91	0.679	$0(67), *(24)$	94%
	6	90	0.716	$5(58), *(32),$	96%
	7	81	0.721	$4(72), *(9)$	90%
	8	81	0.712	$3(58), *(23),$	81%
	9	127	0.774	$7(79), *(48)$	84%
	10	122	0.814	$1(93), *(29)$	90%

Figure 2: Experimental results for the LED database.

are 1 on average. The association rules found are of high order, due to the exact nature of the database. The ordering of the clusters is as expected: note that the fact that our algorithm is biased towards ones implies that encodings with many zeroes (e.g., that for the number 1) are only detected in the end. We may conclude that the algorithm is capable of discovering clusters that respect the original classification, as it did for the zoo database.

4.3 Two Real Life Databases

In contrast with the previously discussed databases, the real life datasets considered here contain more customers, who may choose from many products.

They are expected to show a less regular behaviour. Also, the division into classes is not known in advance. In all cases two or three of the biggest selling products were removed, since they do not contribute to the generation of interesting association rules. For example, in one of the datasets about 50% of the customers bought one particular product; this product is very likely to occur in a good association rule, but probably has not much discriminating ability.
The first database has $m = 7,500$ products, whereas the number of customers is of moderate size; we experimented with a subset consisting of $n = 800$ customers, buying 50 to 95 products each (sample **D1**), and a subset consisting of $n = 1,400$ customers, buying 8 to 10 products each (sample **D2**). For **D1** the mean number of products bought was 57.05, and the mean distance

sample	cluster	number of customers	mean distance	rule	confidence
D1	1	26	0.947	$s_1, s_2 \Rightarrow s_3$	69%
	2	23	0.952	$s_4, s_5 \Rightarrow s_3$	69%
	3	35	0.952	$s_6, s_7 \Rightarrow s_2$	51%
	4	41	0.955	$s_3, s_8 \Rightarrow s_9$	43%
	5	22	0.962	$s_{10} \Rightarrow s_{11}$	72%
	6	32	0.965	$s_{12} \Rightarrow s_{13}$	50%
	7	41	0.964	$s_{14} \Rightarrow s_9$	46%
	8	46	0.964	$s_{15} \Rightarrow s_3$	45%
	9	55	0.964	$s_{16} \Rightarrow s_{17}$	43%
	10	62	0.966	$s_{18} \Rightarrow s_{19}$	38%
D2	1	14	0.832	$t_1 \Rightarrow t_2$	85%
	2	11	0.841	$t_3 \Rightarrow t_2$	72%
	3	10	0.841	$t_4 \Rightarrow t_5$	70%
	4	24	0.823	$t_5 \Rightarrow t_2$	87%
	5	14	0.855	$t_2 \Rightarrow t_6$	50%
	6	25	0.865	$t_7 \Rightarrow t_8$	40%
	7	50	0.868	$t_9 \Rightarrow t_{10}$	34%
	8	32	0.872	$t_{11} \Rightarrow t_{12}$	28%
	9	1220	0.939	$\Rightarrow t_{13}$	5%

Figure 3: Experimental results for the first real life database.

was 0.975, which is very high. For **D2** these numbers were 8.36 and 0.939, respectively.
The second database has $m = 100$ products, and we considered $n = 10,000$ customers (sample **D3**). It also contained purer data, i.e., there were less flaws present, probably because the products involved were more expensive. The mean number of products per customer was 2.11, with mean distance 0.720. Sample **D4** consisted of $n = 10,000$ other customers from the same database; here the mean number of products per customer was 2.35, with mean distance 0.733.
In all cases the support threshold was taken to be 2%, and the number of clusters to find was bounded by 10. Only for **D2** the support threshold was 0.5; larger values did not provide any significant clustering in that case. The results of the experiments are reported in Figure 3 and Figure 4. Except for the computation of the mean distance the runs took only a few minutes on a Pentium-based PC. The products that occur in the rules are arbitrarily named s_1, s_2, $\ldots$, s_{19} and t_1, t_2, $\ldots$, t_{13} for the first database, and p_1, p_2, $\ldots$, p_{13} for the second one.
Note that the products in the rule for $cluster_3$ (sample **D3**) form a subset of those from $cluster_1$. In fact, in this database the rule $p_2 \Leftrightarrow p_3$ holds with high reliability. Since the algorithm tries to find association rules of the highest possible order first, having fixed support threshold, this rule is superseded by the rule $p_1, p_2 \Rightarrow p_3$ that constitutes $cluster_1$. In this case two separate clusters are found. If the support threshold were such that it was not met by $p_1, p_2 \Rightarrow p_3$, only one cluster, based on $p_2 \Rightarrow p_3$, would have resulted. This shows that human interference plays a crucial role in the process: the choice of the support threshold influences the clustering. In contrast, the triples $\{p_1, p_2, p_3\}$ and $\{p_4, p_5, p_6\}$ show some differences; the rule $p_5 \Rightarrow p_6$ has low confidence, and the clustering concerning these three products is not as clear as the one for $\{p_1, p_2, p_3\}$.
The last clusters do not seem to be of any importance. This holds in particular for $cluster_7$, resp. $cluster_8$, which resulted from a rule with empty left-hand side and consequently very low confidence. Also note the relatively high mean distance. No association rules of order 2 were present anymore, and the algorithm clusters all the remaining customers (remember that all customers buying the left-hand side, which is empty here, are clustered). Domain experts were capable of interpreting the most significant clusters. The experiments with many customers show higher coherence within the clusters, reflected by lower mean distance and higher confidence. In all cases the rules

sample	cluster	number of customers	mean distance	rule	confidence
D3	1	253	0.484	$p_1, p_2 \Rightarrow p_3$	98%
	2	337	0.443	$p_4, p_5 \Rightarrow p_6$	89%
	3	431	0.411	$p_2 \Rightarrow p_3$	99%
	4	320	0.372	$p_7 \Rightarrow p_8$	91%
	5	370	0.473	$p_9 \Rightarrow p_{10}$	72%
	6	2102	0.388	$p_5 \Rightarrow p_6$	67%
	7	6187	0.679	$\Rightarrow p_{11}$	20%
D4	1	216	0.455	$p_4, p_5, p_{12} \Rightarrow p_6$	96%
	2	202	0.557	$p_2, p_{13} \Rightarrow p_3$	100%
	3	312	0.428	$p_1, p_2 \Rightarrow p_3$	98%
	4	485	0.424	$p_4, p_5 \Rightarrow p_6$	90%
	5	376	0.450	$p_7 \Rightarrow p_8$	93%
	6	392	0.426	$p_9 \Rightarrow p_{10}$	85%
	7	1253	0.446	$p_6 \Rightarrow p_5$	63%
	8	6764	0.676	$\Rightarrow p_{11}$	26%

Figure 4: Experimental results for the second real life database.

found had low order, also due to the abundance of products to choose from. But even for the case with fewer customers and less products per customer the clusters found made sense.

5 Conclusions

In this paper we have proposed a simple method for mining clusters in large databases describing information about products purchased by customers. The method generates a sequence of clusters in an iterative hierarchical fashion, using association rules for biasing the search towards good clusters. We have tested this method on various datasets. The results of the experiments indicate that the technique allows one to find informative clusterings.
As already mentioned, due to the hierarchical strategy employed to generate the clustering, it may happen that the cluster generated in the last iteration contains objects sharing few regularities. In this case, one can discard the last cluster and consider its objects as not belonging to the clustering, because

they do not present enough regularities. Alternatively, one can redistribute the elements of the last cluster among the other clusters. For instance, a possible redistribution criterion can be the distance of the objects from the clusters, where an object is inserted in the cluster having minimal distance. More sophisticated techniques for redistributing objects of the last cluster can also be applied (e.g., [7, 9]).

Clustering techniques have been studied extensively in the database community, yielding various systems such as `CLARANS` ([13]) and `BIRCH` ([17]). These systems are rather general: they apply techniques imported from clustering algorithms used in statistics, like in `CLARANS`, or sophisticated incremental algorithms, like in `BIRCH`. It is not our intention to advocate the use of our clustering algorithm as an alternative for such systems. Nevertheless, our clustering algorithm provides a simple tool for mining clusters in large databases describing sales data.

Several techniques based on association rules have been proposed for mining various kinds of information. However, to the best of our knowledge, our method provides a novel use of association rules for clustering. Some related techniques based on association rules are the following. In [1] an algorithm for finding profile association rules is proposed, where a profile association rule describes associations between customer profile information and behaviour information. In [12] association rules containing quantitative attributes on the left-hand side and a single categorical attribute on the right-hand side are considered. A method for clustering these rules is introduced, where rules having adjacent ranges are merged into a single description. This kind of clustering provides a compact representation of the regularities present in the dataset. Finally, in [5] the use of association rules for partial classification is investigated, where rules describing characteristics of some of the data classes are constructed. The method generates rules which may not cover all classes or all examples in a class. Moreover, examples covered by different rules are not necessarily distinct.

In this paper, we restricted ourselves to a specific type of datasets where the objects are vectors of binary attributes. We intend to investigate the effectiveness of the clustering method when multivalued attributes as well as quantitative ones are used: this amounts to considering more expressive forms of rules, like for instance the so-called profile association rules (see [1]). An interesting topic for future work is the analysis of the integration of our technique into more sophisticated clustering systems. For instance, we would like to analyze the benefits of our clustering algorithm when used for generating a "good" initial clustering of the data that could be subsequently refined, either by means of iterative methods in the style of those from [7], or by means of methods based on evolutionary computation like genetic algorithms.

References

[1] C.C. Aggarwal, Z. Sun and P.S. Yu, Online Generation of Profile Association Rules, in: R. Agrawal, P.E. Stolorz and G. Piatetsky-Shapiro (editors), *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pp. 129–133, AAAI Press, 1998.

[2] R. Agrawal, T. Imielinski and A. Swami, Mining Association Rules between Sets of Items in Large Databases, in: P. Buneman and S. Jajodia (editors), *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp. 207–216, ACM Press, 1993.

[3] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A.I. Verkamo, Fast Discovery of Association Rules, in: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy (editors), *Advances in Knowledge Discovery and Data Mining*, pp. 307–328, AAAI/MIT Press, 1996.

[4] R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules in Large Databases, in: J.B. Bocca, M. Jarke and C. Zaniolo (editors), *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, pp. 478–499, Morgan Kaufmann, 1994.

[5] K. Ali, S. Manganaris and R. Srikant, Partial Classification Using Association Rules, in: D. Heckerman, H. Mannila, D. Pregibon and R. Uthurusamy (editors), *Proceedings of the Third International Conference on Knowledge Discovery in Databases and Data Mining (KDD-97)*, pp. 115–118, AAAI Press, 1997.

[6] P. Arabie and L.J. Hubert, An Overview of Combinatorial Data Analysis, in: P. Arabie, L.J. Hubert and G.D. Soete (editors), *Clustering and Classification*, pp. 5–63, World Scientific Pub., New Jersey, 1996.

[7] G. Biswas, J.B. Weinberg and D.H. Fisher, ITERATE: A Conceptual Clustering Algorithm for Data Mining, *IEEE Transactions on Systems, Man, and Cybernetics* 28C (1998), 219–230.

[8] M.S. Chen, J. Han and P.S. Yu, Data Mining: An Overview from a Database Perspective, *IEEE Transactions on Knowledge and Data Engineering* 8 (1996), 866–883.

[9] D. Fisher, Iterative Optimization and Simplification of Hierarchical Clusterings, *Journal of Artificial Intelligence Research* 4 (1996), 147–180.

[10] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[11] W.A. Kosters, J.A. La Poutré and M.C. van Wezel, Understanding Customer Choice Processes Using Neural Networks, in: H.F. Arner Jr. (editor), *Proceedings of the First International Conference on the Practical Application of Knowledge Discovery and Data Mining (PADD97)*, pp. 167–178, The Practical Application Company, London, 1997.

[12] B. Lent, A.N. Swami and J. Widom, Clustering Association Rules, in: A. Gray and P.-Å. Larson (editors), *Proceedings of the Thirteenth International Conference on Data Engineering*, pp. 220–231, IEEE Computer Society Press, 1997.

[13] R.T. Ng and J. Han, Efficient and Effective Clustering Methods for Spatial Datamining, in: J.B. Bocca, M. Jarke and C. Zaniolo (editors), *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, pp. 144–155, Morgan Kaufmann, 1994.

[14] J.-S. Park, M.-S. Chen, P.S.Yu, An Effective Hash Based Algorithm for Mining Association Rules, in: M.J. Carey and D.A. Schneider (editors), *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pp. 175–186, ACM Press, 1995.

[15] The UCI Machine Learning Repository, electronically available at `http://www.ics.uci.edu/~mlearn/MLSummary.html`

[16] M. Zaït and H. Messatfa, A Comparative Study of Clustering Methods, *Future Generation Computer Systems* 13 (1997), 149–159.

[17] T. Zhang, R. Ramakrishnan and M. Livny, BIRCH: An Efficient Data Clustering Method for Very Large Databases, in: H.V. Jagadish and I.S. Mumick (editors), *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pp. 103–114, ACM Press, 1996.