Chapter 4

Data Mining

4.1 Introduction

In the first chapter we have mentioned that the database area has played an important role in the growth of data mining. This has been clearly the case in one of the most popular data mining algorithms: *association analysis*.

There is a trend in business to provide a *personalized* service to customers, by means of using barcodes, and many other mechanisms, that identify our transactions. This trend materializes in the storage of large amounts of information in the so-called *transactional databases*.

Transactional databases (see for instance (Han and Kamber, 2001)) store records formed by a unique transaction identifier and a list of items members of that transaction. A traditional example of a transaction is our shopping basket in a supermarket, when we use a customer loyalty card at the moment we pay such that, our identity and the list of products we buy, are registered.

Association analysis (Han and Kamber, 2001) is the discovery of *association rules*. An association rule, as we shall see in depth later, is a set of attribute-value conditions that occur frequently enough that these conditions become an interesting observation about what is stored in the database and, thus, what happens in the store.

This is an instance of one of the main differences between the models used in data mining, and those used in statistics. While in the former tend to describe local patterns also, the latter try to construct a global model only.

The induction of association rules poses several difficult problems regarding the computational complexity of extracting such rules from large databases. Many algorithms have been developed for such purpose and in this chapter we will see how GMMs provide a new perspective for this problem. This new perspective opens new ways of tackling the problem of retrieving association rules from databases. The approach is interesting in the sense that combines the use of a global model to guide the process of finding an interesting local pattern.

In the next section we will describe in detail what association rules are, and existing algorithms for their retrieval, and in section 3 we will introduce a new algorithm for such purpose (Castelo et al., 2001). Finally, we will summarize the important points of this chapter.

4.2 Association Rules

Let $\mathcal{U} = \{I_1, I_2, \ldots, I_m\}$ be a finite set of available items. A transaction is a subset $\mathcal{T}_i \subseteq \mathcal{U}$ uniquely identified by some index i, which may correspond, for instance, to a shopping basket purchased in a supermarket. Given two disjoint non-empty subsets of items $\mathcal{A}, \mathcal{B} \subseteq \mathcal{U}$, an association rule $\mathcal{A} \rightarrow \mathcal{B}$ indicates cross-sell effects on the items in $\mathcal{A}$ and $\mathcal{B}$ across the transactions $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N$.

For each item consider a binary random variable Z_i which, in a given transaction, will take the value 1 if the i_{th} item belongs to the transaction, and 0 otherwise. In this way we can represent a particular set of items by a particular assignment $\mathbf{Z} = \mathbf{z}$, where as in previous chapters, $\mathbf{Z}$ represents a set of variables and $\mathbf{z}$ is some instantiation of the product space $\times \mathcal{Z}_i$, $\mathcal{Z}_i = \{0, 1\}$. Now, we can write an association rule as

$$\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y},$$

where $\mathbf{X}, \mathbf{Y} \subset \mathbf{Z}$. For simplicity, we will neglect the instantiated values $\mathbf{x}$ and $\mathbf{y}$ from the previous notation when their presence is not relevant in the discussion, thus just writing $\mathbf{X} \rightarrow \mathbf{Y}$.

In general terms, the problem of finding *interesting* association rules corresponds to the problem of finding assignments $\mathbf{Z} = \mathbf{z}$, that provide a high probability density in the joint probability distribution $p(\mathbf{Z})$. As pointed out in Hastie et al. (2001), this may be viewed as a problem of "mode finding" or "bump hunting" which reveals that the problem, as formulated, is intractable.

In order to provide a tractable solution to the problem of finding interesting association rules, one does not try to seek these particular assignments $\mathbf{z}_1, \ldots, \mathbf{z}_\mathbf{m}$ that concentrate the density of $p(\mathbf{Z})$. Instead one tries to find *regions* in the product space $\times \mathcal{Z}_i$ that provide a high probability relative to the amount of transactions that fall within those regions.

The *proportion* of transactions that fall within a particular region of the product space $\times \mathcal{Z}_i$, $\mathbf{Z} = \mathbf{z}$, is called the *support* of the items present throughout these transactions, and noted $\sup(\mathbf{Z} = \mathbf{z})$. A set of items, or *itemset*, $\mathbf{X} = \mathbf{x}$ is said to be *frequent* if its support exceeds some predefined support t, i.e. $\sup(\mathbf{X} = \mathbf{x}) \geq t$. Analogously, for a given association rule $\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y}$, one says that the *support of this rule* is the proportion of transactions that contain all items in $\mathbf{X}$ and $\mathbf{Y}$, noted $\sup(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y})$. Using the concept of support, the *confidence* of a rule is defined as:

$$\mathrm{cnf}(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y}) = \frac{\sup(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y})}{\sup(\mathbf{X} = \mathbf{x})}. \tag{4.1}$$

Association rules were introduced by Agrawal et al. (1993) who provided the *Apriori* algorithm to learn them from data. Given some fixed support and confidence thresholds, t and c respectively, $0 < t, c < 1$, the Apriori algorithm works by creating a collection of frequent itemsets which are used later as building blocks to find interesting association rules whose confidence exceeds the corresponding threshold c.

More concretely, it begins by creating all single frequent itemsets. Then, the algorithm creates all frequent itemsets of size two using the single frequent itemsets previously created. The algorithm works in this way systematically, using previously encountered itemsets and discarding those newly created that do not meet the fixed threshold on

support t. This strategy follows from the realization that if an itemset is not frequent, then no superset of this itemset can be frequent.

Once we have a collection $\mathcal{F}$ of frequent itemsets, then the association rules are generated as follows:

1. For every frequent itemset $F \in \mathcal{F}$, create all non-empty subsets of F.
2. For every non-empty subset S of F, let $S = (\mathbf{X} = \mathbf{x})$ and $F \backslash S = (\mathbf{Y} = \mathbf{y})$. If $\text{cnf}(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y}) > c$, then output the rule $\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y}$.

Since the antecedent and the consequent of the rule are derived from a single frequent itemset, it follows immediately that the rule also satisfies the minimum support threshold t. A further measure typically used in this context to rank and filter interesting rules is an estimate of the association between the antecedent and the consequent, which is known as the *lift* and computed as follows:

$$\text{lft}(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y}) = \frac{\text{cnf}(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y})}{\text{sup}(\mathbf{Y} = \mathbf{y})}. \tag{4.2}$$

The lift is a positive quantity that has value 1 if there is no association between the antecedent and the consequent, and we will be interested in those association rules with a high lift.

Examples of the application of the Apriori algorithm, more detailed explanations and enhancements for improving performance may be found in (Agrawal et al., 1995; Han and Kamber, 2001). A different approach for rule extraction, may be found in (Goodman et al., 1992).

4.3 Association Rules based on Conditional Independencies

The support pruning by Apriori poses a serious problem when applying association rules in practice. Rules with high support are in general already known, while using a (very) low support threshold causes two problems. Firstly, it results in very many rules, most of which are uninteresting. The second, as pointed out by Hastie et al. (2001, pg. 444), is that the number of frequent itemsets and their sizes can grow exponentially.

For the first problem, many intestingness measures, such as the lift, have been defined, that can be used to filter or order the resulting association rules. The computational problem, however, has proven to be more difficult. In fact, the support threshold was introduced for computational reasons because a high support threshold substantially reduces the number of possible frequent itemsets.

From a purely data analysis point of view, pruning on support is therefore an artificial device unrelated to what the data may reflect. A sensible approach is to devise algorithms that do not rely on support pruning to find interesting association rules. Two recent works in this direction are (Silverstein et al., 1998; Cohen et al., 2001). In this section we will introduce another approach that does not rely on support pruning.

We propose to use the relationships of conditional independence among the random variables representing the items, to make the discovery of association rules computationally feasible. As we already discussed in previous chapters, conditional independence

plays an important role in the mechanism that generates the data, therefore it is an appealing way to restrict the set of association rules considered.

Recall the definition of conditional independence (see Definition 2.1), under which two non-empty sets of variables $\mathbf{X}$ and $\mathbf{Y}$ are said to be conditionally independent given a third set $\mathbf{Z}$ if for all configurations $\mathbf{x}, \mathbf{y}, \mathbf{z}$ satisfying $p(\mathbf{Z} = \mathbf{z}) > 0$, it holds that

$$p(\mathbf{X} = \mathbf{x} | \mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}) = p(\mathbf{X} = \mathbf{x} | \mathbf{Z} = \mathbf{z}). \tag{4.3}$$

This relationship was noted as $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$, and the notion behind is that if we know $\mathbf{Z}$, any further information about $\mathbf{Y}$ cannot enhance the current state of information about $\mathbf{X}$, i.e. given $\mathbf{Z}$, $\mathbf{Y}$ becomes irrelevant to $\mathbf{X}$.

Since the support for a given itemset $\mathbf{X} = \mathbf{x}$ is defined as the proportion of transactions that contain the items in $\mathbf{X} = \mathbf{x}$, it can be interpreted as an estimate of the probability of $\mathbf{X} = \mathbf{x}$, $p(\mathbf{X} = \mathbf{x}) = \sup(\mathbf{X} = \mathbf{x})$. The support of a rule is then also an estimate of the joint probability of the antecedent and consequent, $p(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) = \sup(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y})$.

Consequently, the confidence (4.1) of a rule is an estimate of the conditional probability $p(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) = \mathrm{cnf}(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y})$. Finally, the lift (4.2) of a rule is therefore an estimate of the following ratio,

$$\mathrm{lft}(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y}) = \frac{p(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x})}{p(\mathbf{Y} = \mathbf{y})}. \tag{4.4}$$

.

If we know that $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$, then by the factorization in (4.3), we know that the previous ratio for the rule $\{\mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z}\} \rightarrow \mathbf{Y} = \mathbf{y}$ can be rewritten as follows

$$\mathrm{lft}(\{\mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z}\} \rightarrow \mathbf{Y} = \mathbf{y}) = \frac{p(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z})}{p(\mathbf{Y} = \mathbf{y})} = \frac{p(\mathbf{Y} = \mathbf{y} \mid \mathbf{Z} = \mathbf{z})}{p(\mathbf{Y} = \mathbf{y})}.$$

In a nutshell, the lift does not rise by adding knowledge about $\mathbf{X}$, $\mathbf{X}$ is irrelevant to $\mathbf{Y}$ given $\mathbf{Z}$. Or, if we have an association rule for $\mathbf{Y}$ with $\mathbf{X}$ and $\mathbf{Z}$ on the lefthand side, we might as well filter $\mathbf{X}$ out.

This is interesting if $\mathbf{Z}$ *shields* $\mathbf{Y}$ from all other variables, i.e. if $\mathbf{Y} \perp\!\!\!\perp \mathbf{U} \backslash (\mathbf{Y} \cup \mathbf{Z}) \mid \mathbf{Z}$ where $\mathbf{U}$ denotes the entire set of variables. Because then we only have to consider association rules whose lefthand side is within $\mathbf{Z}$. All of this is even more interesting if $\mathbf{Z}$ is *minimal*, i.e. if we remove an element from $\mathbf{Z}$ it no longer shields $\mathbf{Y}$ from the rest. Such a minimal set is called a *Markov blanket* of $\mathbf{Y}$ (Pearl, 1988).

To keep our discussions simple, we will assume that the consequent of the association rule is a singleton, as $\mathbf{X} = \mathbf{x} \rightarrow Y = y$. The ideas and procedures explained here are easy to adapt afterwards to association rules with a consequent of an arbitrary cardinality.

We want to build an oracle that can answer queries over the validity of CI statements of the form $Y \perp\!\!\!\perp \mathbf{U} \backslash (Y \cup \mathbf{Z}) \mid \mathbf{Z}$ in a given dataset. Or more precisely, whether $\mathbf{Z}$ is the Markov blanket of Y for a given dataset. The problem that immediately arises is that we do not have the probability distribution but a dataset sampled from it. Thus, there is an implicit uncertainty in the validity of such a CI restriction.

In order to account for this uncertainty we adopt a Bayesian approach and use as oracle a posterior distribution over the set of possible Markov blankets of a given consequent Y for a rule, i.e. CI restrictions of the form $Y \perp\!\!\!\perp \mathbf{U} \backslash (Y \cup \mathbf{Z}) \mid \mathbf{Z}$.

To carry out this idea we will make use of DEC Markov models to handle CI restrictions and the MCMC method to compute the posteriors. For a given vertex v in a DEC Markov model, its Markov blanket corresponds to its boundary $bd(v)$ (see chapter 2). So, we can compute the posterior of a CI statement $I \equiv Y \perp\!\!\!\perp \mathbf{U} \backslash (Y \cup \mathbf{Z}) \mid \mathbf{Z}$ given the data as follows:

$$p(I|D) = \sum_{M \in \mathcal{M}} p(I|M, D) p(M|D),$$

which corresponds to the computation of a quantity of interest Δ as indicated in expression (3.28). In the present context, $\mathcal{M}$ corresponds to the class of DEC Markov models, which we use here as a device to handle CI restrictions.

The general procedure to learn association rules is as follows. First, we build an oracle which consists of pairs (I, p) where I is a CI restriction of the form $Y \perp\!\!\!\perp \mathbf{U} \backslash (Y \cup \mathbf{Z}) \mid \mathbf{Z}$ and p is its posterior probability given the data. These pairs will be obtained from the MCMC output.

Second, for every variable, we consider some maximum number of Markov blankets from the corresponding posterior distribution. For each of the Markov blankets, we generate association rules with a subset of the Markov blanket as lefthand side. There are further details to take into account which we will explain now. The complete algorithm, called Mambo (Maximum A posteriori Markov Blankets using an Oracle), is in Figure 4.1. The Mambo algorithm takes the following four parameters:

- *nmb*: maximum number of Markov blankets.
- $\mathbf{Y}$: set $\{Y_1, \ldots, Y_n\}$ of random variables from the dataset.
- *dc*: minimum improvement in confidence for a superrule.
- *dl*: minimum improvement in lift for a superrule.

It has a main loop that goes through lines 1 to 11. At each iteration it will create all possible association rules with the variable Y_i on the righthand side. The association rules created in each iteration are temporarily stored in *pres*, which is initialized in line 3. In line 4 the function oracle$(\cdot)$ is called which returns a set of, at most *nmb*, Markov blankets for the variable Y_i.

The loop that iterates through lines 5 to 9 considers each of these *nmb* Markov blankets, noted $\mathbf{Z}$. The inner loop iterates through all possible subsets of the Markov blanket $\mathbf{Z}$, i.e. the power set $\mathcal{P}(\mathbf{Z})$ without the empty set. Each of those subsets is noted as $\mathbf{X}$ and is used to generate association rules of the form $\mathbf{X} = \mathbf{x} \rightarrow Y_i = y_i$ for all possible values $\mathbf{x} \in \mathcal{X}$ and $y_i \in \mathcal{Y}_i$. In this context $\mathcal{X}$ is the product space $\times \mathcal{X}_i$ where $\mathcal{X}_i$ are the levels of variable X_i and $\mathcal{Y}_i$ are the levels of variable Y_i. All these association rules are stored in the set *pres*.

The set $\{\mathbf{Z}_1, \ldots, \mathbf{Z}_{nmb}\}$ of *nmb* Markov blankets for a particular righthand side variable Y_i, may contain two Markov blankets $\mathbf{Z}_i = \{X_1, \ldots, X_m\}$ and $\mathbf{Z}_j =$

```
algorithm mambo(int nmb, set Y, flt dc, flt dl) returns set
01  set res = ∅
02  for Y_i ∈ Y do
03    set pres = ∅
04    set O = oracle(Y_i, nmb)
05    for Z ∈ O do
06      for X ∈ P(Z)\{∅} do
07        pres = pres ∪ {(X = x → Y_i = y_i) | x ∈ X, y_i ∈ Y_i}
08      enddo
09    enddo
10    res = res ∪ filter(pres, dc, dl)
11  enddo
12  return res
endalgorithm
```

Figure 4.1: The Mambo algorithm.

$\{X_1, \ldots, X_m, W_1, \ldots, W_k\}$, thus $\mathbf{Z}_i \subset \mathbf{Z}_j$. This may lead to generate the following two association rules:

$$\begin{array}{lcl} R_1 & : & \mathbf{X} = \mathbf{x} \rightarrow Y = y \\ R_2 & : & \{\mathbf{X} = \mathbf{x}, \mathbf{W} = \mathbf{w}\} \rightarrow Y = y \end{array}$$

In this circumstance we will say that R_2 is a superrule of R_1. Given an association rule, not all of its superrules are necessarily of interest. In particular, if they have similar confidence and lift. Then, we will prefer the smaller rule since it is more general. For this reason, in line 10 of the Mambo algorithm, we call the function filter$(\cdot)$ which removes from the initial set *pres* all those superrules that do not improve confidence and lift in the given quantities *dc* and *dl*. The resulting filtered set is added to the final set of association rules *res*, which after all iterations of the main loop is returned. The filtering algorithm is specified in Figure 4.2 and takes three parameters:

- S: set of association rules to filter.
- dc: minimum improvement in confidence for a superrule.
- dl: minimum improvement in lift for a superrule.

The algorithm first copies the input set of association rules into the set F on which the filtering (removal) operations will take place. It has a main loop from line 2 till line 31 that goes through every association rule in F. It starts by storing the confidence and lift of the current association rule $\mathbf{X} = \mathbf{x} \rightarrow Y = y$ in c^* and l^* respectively, which are used later on.

From lines 5 till 15 it checks whether the association rule $\mathbf{X} = \mathbf{x} \rightarrow Y = y$ is a superrule in F that does not improve confidence and lift in dc and dl. In such case the boolean variable *sr* takes the truth value and then in line 17 this rule is removed from the

```
algorithm filter(set S, flt dc, flt dl) returns set
01  set F = S
02  for (X = x → Y = y) ∈ F do
03    flt c* = cnf(X = x → Y = y)
04    flt l* = lft(X = x → Y = y)
05    bool sr = false
06    set Q = {(Z = z → W = w) ∈ F | W = Y ∧ w = y}
07    for (Z = z → W = w) ∈ Q and ¬sr do
08      if Z ⊆ X and z ⊆ x then
09        flt c = cnf(Z = z → W = w)
10        flt l = lft(Z = z → W = w)
11        if c* − c ≤ dc and l* − l ≤ dl then
12          sr = true
13        endif
14      endif
15    endfor
16    if sr then
17      F = F \ {X = x → Y = y}
18    else
19      set R = ∅
20      for (Z = z → W = w) ∈ S do
21        if Z ⊇ X and z ⊇ x then
22          flt c = cnf(Z = z → W = w)
23          flt l = lft(Z = z → W = w)
24          if c − c* ≤ dc and l − l* ≤ dl then
25            R = R ∪ {Z = z → W = w}
26          endif
27        endif
28      endfor
29      F = F \ R
30    endif
31  endfor
32  return F
endalgorithm
```

Figure 4.2: The filtering function for the Mambo algorithm.

set F. Otherwise, the rule will remain in F but all other rules in F that are superrules of this one, and do not improve confidence and lift in dc and dl, should be removed from F. This is implemented in lines 19 through 29 where the set R stores such rules and it is finally removed from F in line 29.

4.3.1 Experimental Results

We have run the Apriori and Mambo algorithms over a dataset in order to show the differences between our approach and a minimum support driven approach. The comparison is not intended to show that one is "better" than the other, but rather to illustrate where the differences lie with our conditional independence driven approach.

We used the insurance company benchmark of the COIL 2000 challenge, available from the UCI KDD archive (`http://kdd.ics.uci.edu`). This dataset contains information about the customers of an insurance company, and consists of 86 variables concerning product ownership and socio-demographic characteristics derived from postal area

codes. For this experiment we ignored the socio-demographic variables (1-43). The remaining variables consist of the target variable of the original challenge (whether or not someone owns a caravan policy) and variables concerning the ownership (number of policies) and contribution (money amount) for 21 other insurance products.

In order to make the data suitable for the analysis with binary association rules, we considered only the product ownership variables coding with a value of 1 if the product is owned and 0 otherwise. This leaves us with a dataset consisting of 22 binary variables for a total of 5822 customers. Table 4.1 provides a description of the variables and the corresponding relative frequencies of product ownership.

Table 4.1: Variables in the insurance dataset.

Var.	Description	Freq.
1	private third party insurance	0.402
2	third party insurance (firms)	0.014
3	third party insurance (agriculture)	0.021
4	car policy	0.511
5	delivery van policy	0.008
6	motorcycle/scooter policy	0.038
7	lorry policy	0.002
8	trailer policy	0.011
9	tractor policy	0.025
10	agricultural machines policy	0.004
11	moped policy	0.068
12	life insurance policy	0.050
13	private accident insurance policy	0.005
14	family accidents insurance policy	0.007
15	disability insurance policy	0.004
16	fire policy	0.542
17	surfboard policy	0.001
18	boat policy	0.006
19	bicycle policy	0.025
20	property insurance policy	0.008
21	social security insurance policy	0.014
22	mobile home insurance policy	0.060

In the analysis with the Apriori algorithm we used a minimum support of $t = 0.01$ that corresponds to 59 records. All rules with 1 item on the righthand side were generated from the frequent itemsets found. This yielded a total of 102 association rules.

The oracle was created by running the MC^3 algorithm on the class of DEC Markov models defined on the 22 variables, through 10^6 iterations which seemed to be sufficient to achieve convergence.

We have run the Mambo algorithm with a number of variations in the parameter values in order to illustrate the effect of these different parameters. In the most basic run we selected for each variable only the Markov blanket with highest posterior probability, i.e.

$nmb = 1$ (see Figure 4.1). Furthermore we only consider positive rules for the moment, i.e. rules of type $\mathbf{X} = \mathbf{1} \rightarrow Y = 1$. In general however, values in the antecedent and consequent of the rule may be either 0 or 1. For an initial comparison with the Apriori results we further selected only those rules with support at least 0.01. It is important to note that this was done afterwards, since the minimal support is not used in Mambo to constrain the search.

In order to consider the effect of filtering the superrules that do not improve confidence and lift, we performed the initial analysis without filtering them. This yielded a total of 72 rules, 30 less than Apriori (since we selected rules with support at least 0.01, all 72 rules were also found by Apriori). To illustrate why some rules generated by Apriori were not considered by Mambo we look to an example.

For variable 9 we found that the Markov blanket with highest posterior probability (0.4) is $\{3, 10\}$. One of the rules found with Apriori is $\{3, 16\} \rightarrow 9$ (we use this notation as a shorthand for $\{3 = 1, 16 = 1\} \rightarrow 9 = 1$) with confidence 0.64, lift 26 and support 0.013. This rule was not found with Mambo since the antecedent contains variable 16 which does not belong to the best Markov blanket. The rule $3 \rightarrow 9$ with confidence 0.62, lift 25 and support 0.013 was found by Mambo because its antecedent is a subset of the best Markov blanket.

If we filter superrules that do not improve confidence by 0.05 and lift by 0.1, the number of rules is reduced from 72 to 60. For example, one of the 72 rules found is $\{4, 3\} \rightarrow 16$ with a lift of 1.83. There exists a subrule $3 \rightarrow 16$ however, with a lift 1.80. Therefore the superrule is filtered out from the ruleset.

Now, as mentioned, with Mambo we can find interesting rules regardless of their support. For example, the best rule for product 4 found with Apriori is $\{1, 6, 22\} \rightarrow 4$ with a confidence of 0.88 and lift 1.7. With Mambo (without any support restrictions) we find the rule $\{21, 1\} \rightarrow 4$, with confidence 0.96, lift 1.9, and support 0.008 (46 records). This rule has higher confidence and lift than the best rule found with Apriori, but does not meet the minimum support requirement of 0.01.

Table 4.2: The number of rules found by Mambo for different values of *nmb*, with removal of superrules. Only rules with support at least 0.005 have been counted. The last row also counts rules with zero values in the antecedent or consequent.

	nmb		
ruletype	1	2	5
positive	78	78	85
all	518	549	750

It is interesting to note that setting the maximum number of Markov blankets *nmb* to 5 instead of 1, does not lead to a dramatic increase of the number of rules (see Table 4.2). For example, if we consider only positive rules and remove superrules that do not meet the minimum improvement requirements previously specified, we obtain a total of 78 rules with *nmb* set to 1 (selecting only the rules that have support at least 0.005). If we use the same settings, but select the 5 most probable Markov blankets, we obtain a total of 85 rules. The 7 additional rules found are of high quality in the sense

that they all have a lift larger than 1. One of the additional rules found with $nmb = 5$ is $\{16, 21\} \rightarrow 1$, with confidence 0.72, lift 1.8 and support 0.007. This rule was not found with $nmb = 1$ since the Markov blanket for 1 with highest posterior probability (0.78) is $\{2, 3, 4, 11, 12, 16\}$, i.e. it does not contain 21. However, the Markov blanket with third highest posterior probability (0.036) is the same set with 21 added (see Figure 4.3). Furthermore, $\{16, 21\} \rightarrow 1$ is not pruned by a subrule since it improves the lift as required on the subrules $16 \rightarrow 1$ (lift 1.6) and $21 \rightarrow 1$ (lift 1.5).

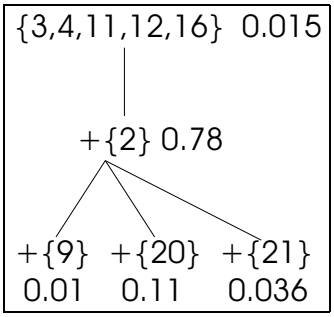

Figure 4.3: The 5 best Markov blankets for variable 1. Only additional elements to the set on top are shown. Posterior probability is shown to the right of or below a set.

Looking at Figure 4.3 we can understand why not very many rules are added when *nmb* is increased from 1 to 5. For example, the set at the top is a subset of the best blanket, and can therefore not generate any additional rules. The sets at the bottom have one element additional to the best set and could generate additional rules. Many of these rules are removed however because they are only slightly better, or even worse, than one of their subrules. For example, $\{4, 16\} \rightarrow 1$ (lift: 1.9) is generated from the best Markov blanket. Later the rule $\{4, 16, 21\} \rightarrow 1$ (lift: 1.9) is considered through the third best Markov blanket but immediately filtered because it has the same lift as the subrule just mentioned.

4.4 Concluding Remarks

Selecting *interesting* association rules is a very difficult problem as in large databases the amount of them, meeting frequency thresholds, can be huge. As Hand et al. (2001) point out, the use of significance testing methods to distinguish interesting rules is problematic.

We have presented an algorithm for the discovery of association rules that is driven by conditional independence relations between the variables rather than minimum support restrictions. We consider this to be an interesting alternative because for some applications minimum support is a rather artificial constraint primarily required for computational reasons.

Making use of conditional independencies is an intuitively appealing way to restrict the set of association rules considered, because CI restrictions are part of the mechanism that generates the data. Furthermore, the information concerning conditional independencies between variables may be of interest in its own right.

We have analyzed an insurance dataset with Mambo and illustrated the differences with a minimum support driven approach. It is ultimately an empirical question whether rules based on CI restrictions or rules based on minimum support restrictions are to be preferred for a particular application. If the rules are used exclusively for descriptive purposes, it makes sense to consult domain experts for the assessment of rule quality. If the rules are used for predictive purposes as well one can compare the predictive accuracy of the rules obtained with the different approaches.