Principles of Data Mining

David Hand
Imperial College

Heikki Mannila
Microsoft Research

Padhraic Smyth
University of California, Irvine

MIT Press, to appear
Draft — October 23, 1999
Do not distribute

Contents

Chapter 13

Finding Patterns and Rules

Contents

13.1 Introduction

In this chapter we consider the problem of finding useful patterns and rules from large data sets. Recall the discussion on patterns versus models from Chapters 1 and 6: a pattern is a local concept, telling us something of a particular aspect of the data, while a model is something that at least attempts to give a full description of the data.

In the image transmission analogue in Section 6.2 a local pattern was a description of a part of the image, such as "there is a pratially obscured circular object in the upper left corner of the image". For the Web log data set in chapter 1, an example pattern could be "if a person visits the CNN Web site, there is a 60% chance they will visit the ABC News Web site in the same month." For a data set about customers of a supermarket, a pattern might be "10 percent of the customers buy wine and cheese," and for a data set about telecommunication alarms a pattern could be "if alarms A and B occur within 30 seconds from each other, then alarm C occurs within 60 seconds with probability 0.5". In each of these cases, the pattern is a potentially interesting piece of information about the data, but such a single pattern does not necessarily give us any global understanding about the data.

How do we find such patterns from data? Given the set of all possible patterns, the trivial method is to try each pattern in turn and see whether it occurs in data. If the number of possible patterns is small, then this method might be applicable, but typically it is completely infeasible. For example in the supermarket example above there is a pattern for each possible subset of the set of all products; for 1000 products this yields 2^{1000} patterns. In the case of images or sequences of alarms there is a potentially infinite number of patterns.

If the patterns were completely unrelated to each other, we would have no other choice than to use the trivial method. However, the set of patterns typically has lots of structure in it. What we have to do is to use this structure of the patterns to guide the search. Typically, there is a generalization/specialization relation between patterns. A pattern α is more general than pattern β, if whenever β occurs in the data, α occurs too. For example, the pattern "at least 10 percent of the customers buy sausage" is more general than the pattern "at least 10 percent of the customers buy sausage and chips". Use of the generalization relationships between patterns leads to simple algorithms for finding all patterns of a certain type that occur in the data.

In this chapter we present some methods for finding local patterns from large classes of data. We start from very simple pattern classes and relatively straightforward algorithms, and discuss some generalizations. The basic theme in the chapter is the discovery of interesting patterns by refining more general ones.

13.2 Rule representations

A *rule* consists of a left-hand side proposition (the antecedent or condition) and a right-hand side (the consequent), e.g., "if it rains then the ground will be wet." Both the left and right-hand sides consist of Boolean (true or false) statements (or propositions) about the world. The rule states that if the left-hand side is true, then the right-hand side is also true. A *probabilistic rule* modifies this definition so that the right-hand side is true with probability p, given that the left-hand side is true — the probability p is simply the conditional probability of the right-hand side given the left-hand side.

Rules have a long history as a knowledge representation paradigm in cognitive modeling and artificial intelligence. Rules can also be relatively easy for humans to interpret (at least relatively small sets of rules are) and, as such, have been found to be a useful paradigm for learning interpretable knowledge from data in machine learning research. In fact, classification tree learning (discussed in Chapters 6 and 10) can be thought of as a special case learning a set of rules, where the conditions at the nodes along the path to each leaf can be considered a conjunction of statements which make up the left-hand side of a rule, and the class label assignment at the leaf node provides the right-hand side of the rule.

Note that rules are inherently discrete in nature, i.e., the left and right-hand sides are Boolean statements. Thus, rules are particularly well-matched to modeling discrete and categorical-valued variables since it is straightforward to make statements about such variables in Boolean terms. One can of course extend the framework to real-valued variables by quantizing such variables into discrete-valued quanta, e.g., "if $X > 10.2$ then $Y < 1$" (this is precisely how classification trees handle real-valued variables for example).

Typically the left-hand sides of rules are expressed as simple Boolean functions (e.g., conjunctions) of variable-value statements about individual variables (e.g., $A = a_1$ or $Y > 0$). For real-valued variables, a left-hand side such as $X > 1 \wedge Y > 2$ is defining a left-hand side region whose boundaries are parallel to the axes of the variables. Again, one could generalize to have statements about arbitrary functions of variables (leading to more complex left-hand side regions), but one loses the interpretability of the simpler form. Thus, for handling real-valued variables in rule learning, simple univariate thresholds are popular in practice because of their simplicity and interpretability.

13.3 Frequent itemsets and association rules

13.3.1 Introduction

Association rules were briefly introduced in Section ??. They are a very simple but useful form of rule patterns for data mining. Consider again an artificial example of 0/1 data shown in Figure 13.1. The rows represent the transactions of individual customers (i.e., a "market basket" of items which were purchased together) and the columns represent the items in the store. A 1 in location (i, j) indicates that customer i purchased item j, and a 0 indicates that that item was not purchased.

basket-id	A	B	C	D	E
t_1	1	0	0	0	0
t_2	1	1	1	1	0
t_3	1	0	1	0	1
t_4	0	0	1	0	0
t_5	0	1	1	1	0
t_6	1	1	1	0	0
t_7	1	0	1	1	0
t_8	0	1	1	0	1
t_9	1	0	0	1	0
t_{10}	0	1	1	0	1

Figure 13.1: An artificial example of basket data.

We are interested in finding useful rules from such data. Given a set of 0,1 valued observations over variables $A_1, \ldots, A_p$, recall from Chapter 6 that an association rule has the form

$$((A_{i_1} = 1) \wedge \cdots \wedge (A_{i_k} = 1)) \Rightarrow A_{i_{k+1}} = 1,$$

where $1 \leq i_j \leq p$ for all j. Such an association rule can be written more briefly as ($(A_{i_1} \wedge \cdots \wedge A_{i_k}) \Rightarrow A_{i_{k+1}}$. Recall also that a pattern such as

$$(A_{i_1} = 1) \wedge \cdots \wedge (A_{i_k} = 1)$$

is called a frequent set. Thus association rules can be viewed as rules of the form $\theta \Rightarrow \varphi$, where θ is a frequent set pattern and φ is a frequent set pattern consisting of a single conjunct. We could also allow conjunctions on the right-hand side of rules, but for simplicity we do not. The framework of association rules was originally developed for large sparse transaction data sets. However, the concept can be directly generalized to non-binary variables taking a finite number of values, although we will not do here (for simplicity of notation).

Given a frequent set pattern θ, its frequency $fr(\theta)$ is the number of observations in the data that satisfy θ. Given an association rule $\theta \Rightarrow \varphi$, its *accuracy* $c(\theta \Rightarrow \varphi)$ is the fraction of rows that satisfy φ among those rows that satisfy θ, i.e.,

$$c(\theta \Rightarrow \varphi) = \frac{fr(\theta \wedge \varphi)}{fr(\theta)}.$$

In terms of conditional probability, the empirical accuracy of an association rule can be viewed as a maximum likelihood (frequency-based) estimate of the conditional probability that φ is true, given that θ is true. Note in passing that instead of a simple frequency-based estimate, one could use a maximum a posteriori estimate to get a more robust estimate of this conditional probability for small sample sizes (see Section ???.???). However, since association rules are typically used in applications with very large data sets and with a large threshold on the size of the frequent sets (i.e., $fr(\theta)$ is usually fairly large), the maximum likelihood estimate will be quite sufficient in such cases.

The frequent itemsets are very simple patterns telling us that a set of variables occurs reasonably often together. Thus knowing a single frequent itemset does not provide us lots of information about the data: it only gives a narrow viewpoint on a certain aspect of the data. Similarly, a single association rule only tells us about a single conditional probability, and does not inform us about the rest of the joint probability distribution governing the variables.

The task in finding frequent itemset patterns is simple: given a frequency threshold s, find all itemset patterns that are frequent, and their frequencies.

In the example of Figure 13.1 the frequent sets for frequency threshold 0.4 are $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$, $\{AC\}$, and $\{BC\}$. From these we could find, e.g., the rule $A \Rightarrow C$ which has accuracy $4/6 = 2/3$ and the rule $B \rightarrow C$ with accuracy $5/5 = 1$.

Algorithms for finding association rules find all rules satisfying the frequency and accuracy thresholds. If the frequency threshold is low, there might be lots of frequent itemsets and thus also lots of rules. Thus, finding association rules is just the beginning in a data mining effort: some of these rules will probably be trivial to the user, while others may be quite interesting. One of the research challenges in using association rules for data mining is to develop methods for selecting potentially interesting rules from the mass of discovered rules.

The rule frequency tells one how often a rule is applicable. In many cases, rules with low frequency are not interesting, and this assumption is indeed built into the definition of the association rule finding problem. The accuracy of an association rule is not necessarily a very good indication of its interestingness, e.g., consider a medical application where the a rule is learned that pregnancy implies that the patient is female with accuracy 1! a rule with accuracy close to 1 could be interesting, but the same is true for a rule with accuracy close to 0. We will return later to this question of measuring how interesting a rule is to a user.

Display 13.1 *Show how the statistical significance of a single association rule can be estimated; talk about the multiple testing problem and discuss the case for sets of association rules.*

A set of association rules does not provide a single model that would enable us to make inferences about the data set. For example, the rules do not provide a direct way of predicting what an unknown entry will be. Different rules might predict different values for a variable, and there is no central structure (as in decision trees) for deciding which rule is in force.

As an example, suppose we have a row with $A = 1$, $B = 1$, $D = 1$, and $E = 1$; then the above set of rules would suggest that $C = 1$ with accuracy $2/3$ (because of the rule $A \Rightarrow C$) and with accuracy 1 (because of the rule $B \Rightarrow C$). Thus the set of rules does not form a global description of the data set in any way. (However, the collection of association rules or frequent sets provides actually a relatively good condensed representation of the original data set, in the sense that a lot of the marginal information about the data can be retrieved from this collection.)

Formulated in terms of the discussion of Chapter 6, the model structure for association rules is the set of all possible conjunctive probabilistic rules. The score function can be thought of as binary: rules with sufficient accuracy and frequency get a score of 1 and all other rules have a score of 0 (only rules with a score of 1 are sought). In the next subsection we discuss search methods for finding all frequent itemsets and association rules, given predefined thresholds on frequency and accuracy.

13.3.2 Finding frequent sets and association rules

In this section we describe methods for finding association rules from large 0/1 matrices. For market basket applications, a typical input data set might have 10^5 to 10^8 data rows, and 10^2 to 10^4 variables. If the rows of the data are documents, and the variables are words occurring in the documents, then the number of columns can be much larger, even millions. These matrices are often quite sparse, though: the number of ones in any given row is typically very small.

The task in association rule discovery is to find all rules fulfilling the frequency and accuracy criteria. This task might seem a little daunting, as there are an exponential number of potential frequent sets in the number of variables of the data, and that number tends to be quite large in, say, market basket applications. Fortunately in real data sets it is the typical case that there will be relatively few frequent sets (e.g., most customers will buy only a small subset of the overall universe of products).

If the data is large enough, it will not fit into main memory. Thus we aim at methods that read the data as few times as possible. The problem of finding association rules is naturally divided into two parts: first finding the frequent itemsets and then forming the rules from the frequent sets.

If the frequent sets are known, then finding association rules is simple. Any rule $X \Rightarrow B$ that has frequency at least s, the set X must by definition have frequency at least s. Thus, if all frequent sets are known, we can generate all rules of the form $X \Rightarrow B$, where X is frequent, and evaluate the accuracy of each of the rules, in a single pass through the data.

We next describe how the frequent sets of variables can be found. A trivial method would be to compute the frequency of all subsets, but obviously that is too slow. The key observation is that a set X of variables can be frequent only if all the subsets of X are frequent. This means that we do not have to find the frequency of any set X that has a nonfrequent proper subset. Therefore, we can find all frequent sets by first finding all frequent sets consisting of 1 variable. Assuming these are known, we build candidate sets of size 2: sets $\{A, B\}$ such that $\{A\}$ is frequent and $\{B\}$ is frequent. After building the candidate sets of size 2, we find by looking at the data which ones of them are really frequent. This gives the frequent sets of size 2. From these, we can build candidate sets of size 3, whose frequency is then computed from the data, and so on. As an algorithm, the method is as follows.

$i = 0$;
$C_i = \{\{A\} \mid A \text{ is a variable } \}$;
while C_i is not empty **do**
    database pass:

```
            for each set in C_i, test whether it is frequent;
            let Li be the collection of frequent sets from C_i;
        candidate formation:
            let C_{i+1} be those sets of size i + 1
                whose all subsets are frequent;
    od.
```

Two issues remain to be solved: how are the candidates formed? and how is the frequency of each candidate computed? The first problem is easy to solve in a satisfactory manner. Suppose we have a collection L_i of frequent sets, and we want to find all sets Y of size $i+1$ that possibly can be frequent, i.e., all sets Y whose all proper subsets are frequent. This can be done by finding all pairs $\{U, V\}$ of sets from L_i such that the union of U and V has size $i+1$, and then testing whether the union really is a potential candidate. There are less than $|L_i|^2$ pairs of sets in L_i, and for each one of them we have to check whether $|L_i|$ other sets are present. The worst-case complexity is approximately cubic in the size of L_i, but in practice the method works in linear time with respect to the size of L_i, since there are often only a few overlapping elements in L_i. Note that candidate formation is independent of the actual data.

Given a set C_i of candidates, their frequencies can be evaluated in a single pass through the database. Simply keep a counter for each candidate and increment the counter for each row that contains the candidate. The time needed for candidate C_i is $O(|C_i|np)$, if the test is implemented in a trivial way. Additional data structure techniques can be used to speed up the method.

The total time needed for the finding of the frequent sets is $O(\sum_i |C_i|np)$, i.e., proportional to the product of the size of the data (np) and the number of sets that are candidates on any level. The algorithm needs k or $k+1$ passes through the data, where k is the number of elements in the largest frequent set.

There exist many variants of the basic association rule algorithm. The methods typically strive towards one or more of the following three goals: minimizing the number of passes through the data, minimizing the number of candidates that have to be inspected, and minimizing the time needed for computing the frequency of individual candidates.

One important way of improving speeding the computation of frequencies of candidates is to use data structures that make it easy to find out which candidate sets in C_i occur for each row in the data set. A possible way of organizing the candidates is to use a tree-like structure with branching factor p (the number of variables). For each variable A, the child of the root of the tree labeled with A contains those candidates whose first variable (according to some ordering of the variables) is A. The child labeled A is constructed in a recursive manner.

Another important way of speeding up the computation of frequent sets is to use sampling. Since we are interested in finding patterns describing large subgroups, i.e., patterns having support higher than a given threshold, it is clear that just using a sample instead of the whole data set will give a fairly good approximation for the collection of frequent sets and their frequencies. A sample can also be used to obtain a method that with high probability needs only two passes through the data. First, compute from the sample the collection of frequent sets $\mathcal{F}$ using a threshold that is slightly lower than the one given by the user. Then, compute the frequencies in the whole data set of each set in $\mathcal{F}$. This produces the exact answer to the problem of finding the frequent sets in the whole data set, unless there is a set Y of variables that was not frequent in the sample but all of whose subsets turned out to be frequent in the whole data set; in this case, one has to make an extra pass through the database.

13.4 Generalizations

The method for finding frequently occurring sets of variables can also be applied to other types of patterns and data since the algorithms described above do not use any special properties of the frequent set patterns. What we used were (1) the conjunctive structure of the frequent sets and the monotonicity property, so that candidate patterns could be formed quickly, and (2) the ability to test quickly whether a pattern occurs in a row, so that the frequency of the pattern can be computed by a fast pass through the data.

Next we formulate the same algorithms in terms of more abstract notions. Suppose, as in Chapter xxx, that we have a class of *atomic patterns* $\mathcal{A}$, and our interest is in finding conjunctions of these atomic patterns that occur frequently. That is, the pattern class $\mathcal{P}$ is the set of all conjunctions

$$\alpha_1 \wedge \cdots \wedge \alpha_k,$$

where $\alpha_i \in \mathcal{A}$ for all i.

Let the data set D consist of n objects $d_1, \ldots, d_n$, and assume we can test whether a pattern α is true about an object d. A conjunction $\theta = \alpha_1 \wedge \cdots \wedge \alpha_k \in \mathcal{P}$ is true about d if all conjuncts α_i are true about d. The goal is to find those conjunctions of patterns that occur frequently:

$$\{\theta \in \mathcal{P} \mid \theta \text{ is true for for at least } \sigma \text{ objects } d \in D\}.$$

In the case of frequent itemsets, the atomic patterns were conditions of the form $A = 1$, where A is a variable, and the frequent sets like ABC were just shorthand notations for the conjunctions of form $A = 1 \wedge B = 1 \wedge C = 1$.

Suppose we can for each atomic pattern α decide how many times it occurs in the data. Then we can apply the above algorithm for finding all the patterns from $\mathcal{P}$ that occur frequently enough. We simply first find the atomic patterns that occur frequently enough, then build the conjunctions of two atomic patterns that can possibly occur frequently, test to see which of those occur frequently enough, build conjunctions of size three, etc. The method works in exactly the same way as before. If the patterns are complex, it can be that we have to do some clever processing to build the new candidates and to test for occurrence of patterns. We shall see an example of this algorithm in Chapter xxx in the discussion of an application case study from telecommunications.

13.5 Finding highly correlated pairs of variables

A problem in the framework of association rules we have presented so far is that the approach is inherently tied to the finding rules that have a high frequency. This can be motivated in terms of typical applications of data mining. In market basket analysis, for example, the retailer probably is not interested in finding rules that show connections between very rarely bought items. However, if those items are quite expensive, it might still be interesting to know what the most significant correlations between the variables are. Another problem (or strength) of the association rule idea is the asymmetry between 0s and 1s in the definition.

Recently, approaches have been presented for finding the most significant correlations between variables for the cases where there are so many variables that actual examination of each pair is impossible.

There are of course different ways to define and interpret what one means by "significant correlation;" here we present just one possibility. Given p binary variables $A_1, \ldots, A_p$ and a set $r = \{t_1, \ldots, t_n\}$ of n observations with p values, define the correlation of the pair A_i and A_j to be $c(A_i, A_j) = Pr(A_i = A_j)$, i.e., the probability that a randomly chosen row $t \in r$ has the same value for variables A_i and A_j. The task is to find the pair (A_i, A_j) that has the highest correlation. This problem has been called the "lightbulb problem" [**?**].

A straightforward method would be to investigate each pair (A_i, A_j) and to compute an estimate of the probability $Pr(A_i = A_j)$. The complexity of such a method would be $O(p^2 n)$, which is prohibitively large for large values of p. If we have tens of thousands of variables, we simply cannot investigate each pair of variables separately.

A beautiful solution to this problem has been suggested in [**?**]. The algorithm finds, with high probability, a pair of variables whose correlation is not too far from the largest one. The basic idea is to repeatedly sample m rows $\{u_1, \ldots, u_m\}$ from the relation r. Given a variable A_i, denote by $s(A_i)$ the values of A_i for the rows in the sample, i.e., the string $u_1[A_i], u_2[A_i], \ldots, u_m[A_i]$.

If m is not too large, then for a highly correlated pair (A_i, A_j) of variables we have with high probability that $s(A_i) = s(A_j)$. On the other hand, if m is large enough, then the number of pairs of variables (A_i, A_j) such that $s(A_i) = s(A_j)$ is at most linear in the total number of variables.

Repeating the sampling sufficiently many times, it is relatively easy to show that the pair with the highest correlation satisfies the condition $s(A_i) = s(A_j)$ more frequently than all the other pairs.

13.6 Finding episodes from sequences

In this section we describe algorithms for finding episodes from sequences. Again, the methods are based on the same general idea as the association rule algorithms: the frequencies of patterns are computed by starting from the simplest possible patterns. New candidate patterns are built using the information from previous passes through the data, and a pattern is not considered if any of its subpatterns is not frequent enough. The main difference compared to the algorithms outlined in the previous sections is that the conjunctive structure of episodes is not as obvious.

Recalling the notation and discussion in Chapter 6, an episode β is defined as a *subepisode* of an episode α, if all the nodes of β occur also in α and if all the relationships between the nodest in β are also present in α. Using graph-theoretic terminology, we can say that β is an induced subgraph of α. `Query: Check this.`

We write $\beta \preceq \alpha$ if β is a subepisode of α, and $\beta \prec \alpha$ is $\beta \preceq \alpha$ and $\beta \neq \alpha$.

> **Display 13.2** *From Figure* **??** *we see that $\beta \preceq \gamma$ since β is a subgraph of γ. In terms of the definition, there is a mapping f that connects the nodes labeled A with each other and the nodes labeled B with each other, i.e., both nodes of β have (disjoint) corresponding nodes in γ. Since the nodes in episode β are not ordered, the corresponding nodes in γ do not need to be ordered, either.*

We now concentrate on the following discovery task: given an event sequence $\mathbf{s}$, a set $\mathcal{E}$ of episodes, a window width *win*, and a frequency threshold *min_fr*, find the collection $\mathcal{F}\mathbf{s}$*win*, *min_fr* of all episodes from the set that occur at least in a fraction of *min_fr* of all the windows of width *win* on the sequence $\mathbf{s}$. Display xxx gives an algorithm for computing the collection of frequent episodes.

> **Display 13.3** *Given a set E of event types, an event sequence* $\mathbf{s}$ *over E, a set $\mathcal{E}$ of episodes, a window width* win*, and a frequency threshold* min_fr*, the following algorithm computes the collection* $\mathcal{F}\mathbf{s}\mathrm{win}, \mathrm{min\_fr}$ *of frequent episodes.*

```
C_1 := {α ∈ E | |α| = 1};
l := 1;
while C_l ≠ ∅ do
        /* Database pass: */
        compute F_l := {α ∈ C_l | fr(α, s, win) ≥ min_fr};
        l := l + 1;
        /* Candidate generation: */
        compute C_l := {α ∈ E | |α| = l and for all β ∈ E such that β ≺ α and |β| < l
                                we have β ∈ F_|β|};
for all l do output F_l;
```

The algorithm performs a levelwise (breadth-first) search in the class of episodes following the subepisode relation. The search starts from the most general episodes, i.e., episodes with only one event. On each level the algorithm first computes a collection of candidate episodes, and then checks their frequencies from the event sequence.

The algorithm does at most $k+1$ passes through the data, where k is the number of edges and vertices in the largest frequent episode. Each pass evaluates the frequency of $|\mathcal{C}_l|$ episodes. The computation of the frequency of an episodes requires that we find which windows on the sequence the episode occurs. This can in practice be done in time linear in the length of the sequence and the size of the episode. The running time of the episode discovery algorithm is thus $O(n \sum_{l=1}^{k} |\mathcal{C}_l| l)$, where n is the length of the sequence.

13.7 Selective discovery of patterns and rules

13.7.1 Introduction

The previous sections discussed methods that are used to find all rules of a certain type that fulfill simple frequency and accuracy criteria. While this task is useful in several applications, there are also simple and important classes of patterns for which we definitely do not want to see all the patterns.

Consider for example a data set having variables with continuous values. Then, as in Chapter 1, we can look at patterns such as

if $X > x_1$ then $Y > y_1$ with probability p, and the frequency of the rule is q.

Such a rule is a fine partial description of the data. The problem is that if in the data we have k different values of X and h different values of Y, there are kh potential rules, and lots of these will have sufficient frequency to be interesting from that point of view. For example, from a dataset with variables Age and Income we might discover rules

α: if Age>40 then Income>62755 (probability 0.34)
β: if Age>41 then Income>62855 (probability 0.33)

First, the user will not be be happy seeing two rules that express more or less the same general pattern. Thus, even if we found both of the rules above, we should avoid showing them to the user, Second, the pattern α is more general than β, and there are very long sequences $\alpha_1, \alpha_2, \ldots$ such that α_i is more general than α_{i+1}. The basic algorithmic idea in the previous sections of starting from the most general pattern, looking at the data, and expanding the qualified patterns in all possible ways does not work, as there are many specializations of any single pattern and the pattern space is too large.

Thus the search for patterns has to be pruned in addition to using frequency criteria. The pruning is typically done by using two criteria:

1. interestingness: whether a discovered pattern is sufficiently interesting so that it should be output;

2. promise: whether a discovered pattern has a potentially interesting specializations.

Note that a pattern can be promising even though it is not interesting. A simple example is a trivial pattern **true** that is true of any data object: it is not interesting, but some specialization of it can be. Interestingness can be quantified in various ways using the frequency and accuracy of the pattern as well as background knowledge. We will discuss various definitions of interestingness and promise in Section xxx.

`Query: there has to be a way of deciding which rules are interesting and which are not.`

13.7.2 Heuristic search for finding patterns

Assume we have the a way of defining the interestingness and promise of a pattern, and a way of pruning Then a generic heuristic search algorithm for finding interesting patterns can be formulated as follows.

```
C = { the most general pattern };
while C ≠ ∅ do
    E = all suitable selected specializations of
        elements of C;
    for p ∈ E do
        if q satisfies the interestingness criteria then output q;
        if q is not promising then discard q else retain q
    od;
    additionally prune E;
    C = E;
od;
```

As instantiations of this algorithm we get several more or less familiar methods:

1. Assume patterns are itemsets, and define the interestingness and promise of an itemset X both by the predicate $fr(X) > \sigma$. Do no additional pruning. Then the above algorithm is in principle the same as the algorithm for finding association rules in Section xxx.

2. Suppose the patterns are rules of the form

$$\alpha_1 \wedge \cdots \wedge \alpha_k \Rightarrow \beta,$$

where α_i andβ are conditions of the form $X = c$, $X < c$, or $X > c$ for a variable X and a constant c. Let the interestingness criterion be that the rule is statistically significant `in which sense` and let the promise criteria be trivially true. The additional pruning step retains only one rule from E, the one with the highest statistical significance. This gives us a hill-climbing search for a rule with the highest statistical significance.

3. Suppose the interestingness criterion is that the rule is statistically significant, the promise test is trivially true, and the additional pruning retains the K rules whose significance is the highest. Above we had the case for $K = 1$; an arbitrary K gives us *beam search*.

13.7.3 Criteria for interestingness

In the previous sections we referred to measures of interestingness for rules. Given a rule $\theta \Rightarrow \varphi$, its interestingness can be defined in many ways. Typically, background knowledge about the variables referred to in the patterns θ and φ have great influence in the interestingness of the rule. For example, in a credit scoring data set we might decide beforehand that rules connecting the month of birth and credit score are not interesting. As another example, in a market basket database we might say that the interest in a rule is directly proportional to the frequency of the rules times the prices of the items mentioned in the product, i.e., we would be more interested in rules of high frequency that connect expensive items. Generally, there is no single method for automatically taking background knowledge into account, and rule discovery systems need to make it easy for the user to use such application-dependent criteria for interestingness.

Purely statistical criteria of interestingness are easier to use in an application-independent way. Perhaps the simplest criteria can be obtained by constructing a 2×2 contingency table by using the presence or absence of θ and φ as the variables, and having as the counts the frequencies of the four different combinations.

	φ	$\neg\varphi$
θ	$fr(\theta \wedge \varphi)$	$fr(\theta \wedge \neg\varphi)$
$\neg\theta$	$fr(\neg\theta \wedge \varphi)$	$fr(\neg\theta \wedge \neg\varphi)$

From the data in this table we can compute different types of measures of association between θ and φ, e.g., the χ^2 score. One particularly useful measure of interestingness of a rule $\theta \Rightarrow \varphi$ is the *J-measure*, defined as

$$J(\theta \Rightarrow \varphi) = fr(\theta)\left(c(\theta \Rightarrow \varphi)\log\frac{c(\theta \Rightarrow \varphi)}{fr(\theta)} + (1 - c(\theta \Rightarrow \varphi))\log\frac{1 - c(\theta \Rightarrow \varphi)}{1 - fr(\theta)}\right).$$

This measure can be viewed as a variation of the cross-entropy between the events θ and φ. The factor $fr(\theta)$ indicates how widely the rule is applicable. The other factor measures how dissimilar our assumptions about φ are with no knowledge and with the knowledge that θ holds. The J-measure has the added advantage that it behaves well with respect to specializations, i.e., it is possible to prove bounds on the value of the J-measure for specializations of a given rule.

In practice it has been found that different score functions for interestingness will often return largely the same patterns, as long as the score functions obey some basic properties (such as the score monotonically increasing as the frequency of a pattern increases, with the accuracy remaining constant). Other measures for interestingness of a pattern were discussed in Chapter 7.4.

13.8 From local patterns to global models

Given a collection of patterns about the data, is there a way of forming a global model using the patterns? In this section we briefly outline two ways of doing this. The first method forms a decision list or rule set for a classification task, and the second method constructs an approximation of the probability distribution using the frequencies of the patterns.

Let B be for simplicity a 0-1 valued variable and suppose that we have a discovered a collection of rules of the form $\theta_i \Rightarrow B = 1$ and $\eta_i \Rightarrow B = 0$. How would we form a decision list for finding out predicting the value of B?

Recall that a decision list for variable B is an ordered list of rules of the form $\theta_i \Rightarrow B = b_i$, where θ_i is a pattern and b_i is a possible value of B. The accuracy of such a decision list can be defined as the fraction of rows for which the list gives the correct prediction. `Query: should this be in the section for score functions`

The optimal decision list could in principle be constructed by considering all possible orderings of the rules and checking for each which produces the best solution. However, this would take exponential time in the number of rules. A relatively good approximation can be obtained by viewing the problem as a weighted set cover task and using the greedy algorithm. `Query: need to write a couple of more sentences about this`

Now we turn to another way of using local patterns to obtain information about the whole data set. If we know that pattern θ_i has frequency $fr(\theta_i)$ for each $i = 1, \ldots, k$, how much information do we have about the joint distribution on all variables $X_1, \ldots, X_p$? In principle, any distribution f that satisfies has the pattern frequencies could be underneath the observations. However, assuming that the underlying distribution in question has the *maximum entropy*, it can actually be constructed it reasonably efficiently by using the *iterative proportional fitting* algorithm. Roughly speaking, this algorithm operates as possible. Start with a random distribution $g(\mathbf{x})$ on the variables X_j, and then enforce the frequency constraint for each pattern θ_i. This is done by computing the sum of g over the states for which θ_i is true, and scaling these probabilities so that the resulting updated version of g has θ_i true in a set of measure $fr(\theta_i)$. The update step is done in turn for each pattern, until the observed frequencies of patterns agree with those provided by g. The method converges under fairly general conditions, and it is widely employed in, e.g., natural language processing. The drawback of the method (at least if applied straightforwardly) is that it requires the construction of each of the states of the joint distribution, i.e., both the space and time complexity of the method are exponential in the number of variables.

13.9 Predictive Rule Induction

In this chapter we have so far focused primarily on association rules. However, we began the chapter with a slightly more general definition of a rule and we now return to this framework. Recall that we can interpret each branch of a classification tree as a rule, where the internal nodes on the path from the root to the leaf define the terms in the conjunctive left-hand side of the rule and the class label assigned to the leaf is the right-hand side. For classification problems the right-hand side of the rule will be of the form $C = c_k$, i.e., a particular value being predicted for the class variable C.

Thus, we can consider our classification tree as consisting of a *set of rules*. This set has some rather specific properties, namely it forms a mutually exclusive (disjoint) and exhaustive partition of the space of input variables. In this manner, any observation $\mathbf{x}$ will be classified by one and only one rule (i.e., the branch defining the region within which it lies). The set of rules is said to "cover" the input space in this manner.

One can see that it may be worth considering more general rule sets other than tree-structured rule sets. The tree representation can be particularly inefficient (for example) at representing disjunctive Boolean functions, e.g., consider the disjunctive mapping defined by $(A = 1 \wedge B = 1) \vee (D = 1 \wedge E = 1) \Rightarrow C = 1$ (and where $C = 0$ otherwise). We can represent this quite efficiently via the two rules $(A = 1 \wedge B = 1) \Rightarrow C = 1$ and $(D = 1 \wedge E = 1) \Rightarrow C = 1$. A tree representation of the same mapping would necessarily involve a specific single root-node variable for all branches (e.g., A) even though this variable is only relevant to part of the mapping.

One technique for generating a rule set is to build a classification tree (using any of the techniques described in Chapter 10) and then to treat each of the branches as individual candidate rules. Visiting each such rule in turn, the rule-induction algorithm determines if each condition on the left-hand side of each rule affects the accuracy of that rule on the data that it "covers." For example, one could assess whether the accuracy of the rule (which is equivalent to the estimated conditional probability) improves for the better (or indeed shows no significant change at all) when a particular condition is removed from the left-hand side. If it improves or shows no change, the condition can be deemed not necessary and can be removed to yield a simpler and potentially more accurate rule. The process can be repeated until all conditions in all rules are examined. In practice this has been found to eliminate a large fraction of the initial rule conditions, conditions which are introduced in the tree-growing process because of their *average* contribution in terms of model improvement, but which are not necessary for some particular branches.

The final rule set produced in this manner is then used for classification. Since the original rules carved up the input space in a disjoint manner, and we have removed a subset of the conditions defining these disjoint regions, the boundaries have been broadened (the rules have been generalized) and these regions may now overlap. Thus, we might have two rules of the form $A = 1 \Rightarrow C = 1$ and $B = 1 \Rightarrow C = 1$. The natural question is how does one use two such rules to classify a new observation vector $\mathbf{x}$ where both $A = 1$ and $B = 1$? One approach would be to view the two rules as constraints on the overall joint distribution of $p(A, B, C)$ and to infer an estimate for $p(C = 1|A = 1, B = 1)$ using the maximum entropy approach described earlier in this chapter. However, since the maximum entropy approach is somewhat complex computationally much simpler techniques tend to be used in practice. For example, one can find all the rules which "fire" (i.e., for which the conditions are satisfied) given the observation vector $\mathbf{x}$. If there is more than one rule, one could simply pick the one with the highest conditional probability. If no rules fire, one can simply pick the class value which is most likely a priori. Other more complex schemes are also possible, such as arranging the rules in an ordered decision list, or voting or averaging among multiple rules.

One might ask why one would start with a classification tree and then produce rules, rather than searching for the rules directly. One advantage of this technique is that it automatically quantizes any real-valued variables in a relatively simple and computationally efficient fashion during the tree-building phase (although of course these quanta need not necessarily be optimal in any sense in the context of the final rule set). Another advantage is the ease of implementing the technique; there are many efficient techniques for producing trees (both for data in main memory and in secondary memory, as discussed in Chapter 10) and, thus, it is relatively straightforward to add the rule selection component as a "postprocessing" step.

Nonetheless, producing rules from trees imposes a particular bias on how we search for rules, and with this in mind, there has also been a great deal of work in machine learning and data mining on algorithms which search for rules directly, particularly for discrete-valued data. It is worth noting once again of course that the number of possible conjunctive rules is immense, $O(m^p)$ for p variables each taking m values. Thus, in searching for an optimal set of such rules (or even just the best single rule) we will usually resort to using some form of heuristic search through this space (as already pointed out in Section 13.7 in the context of finding interesting sets of rules).

Note here that in the context of classification that optimality should be defined as the set of rules which are most accurate on average on new data (or have the minimum average loss when classification costs are involved). However, just as with classification trees, classification accuracy on the training data need not be the best score function to guide in the selection of rules. For example, one could define a specific rule for each training example containing all of the variable values present in that example. Such a specific rule may have high accuracy (e.g., accuracy 1 if all examples with the same variable values have the same class label), but may generalize poorly since it is so specific. Thus, score functions other than simple accuracy are often used in practice, particularly for selecting the next rule to add to the existing set, e.g., some trade-off between the coverage of the rule (the probability of the left-hand side expression) and the rule accuracy, such as the J-measure described earlier.

Having defined a suitable score function, the next issue is how to search for a set of rules to optimize this score on the training data. Many rule induction algorithms utilize a form of "general-to-specific" heuristic, of the same general form described earlier in searching for interesting rules (Section 13.7.2) where now we replace the interestingness score function with a classification-related one. These algorithms start with a set containing the most general rule possible (i.e., the left-hand side is empty) and proceed to add rules to this set in a greedy fashion by successively exploring more specific versions of the rules in the existing set. This can be viewed as a systematic search through the space of all subsets, starting from the null set and using an operator which can only add one condition to a rule at a time. A large variety of search techniques are applicable here, including any of the systematic heuristic search techniques discussed in Chapter 8 (e.g., beam search). The opposite heuristic strategy of starting from the most specific rules and generalizing is also possible, although computationally this tends to be a bit more tricky since it is not so obvious what set of rules to start from. For real-valued data one can either pre-quantize each real-valued variable into bins (for example by using a clustering algorithm on each variable), or quantize as one searches for rules. The latter option can be computationally quite demanding and tricky to implement; an interesting algorithm which operates in this manner is the PRIM algorithm (Friedman, 1998) which gradually "shrinks" the rule regions starting from the full range of the data for each variable.

There is of course a trade-off between the more computationally (and memory) intensive search techniques which search more of the rule space, and the simpler techniques which can only search a smaller fraction of the space. In practice, as with classification trees, relatively simple greedy search techniques with simple operators often seem to perform almost as well empirically as the more complex methods and are quite popular as a result. As with classification trees, there is also the problem of deciding when to stop adding rules to the rule set, i.e., the by now familiar problem of deciding how complex the model should be (here we can interpret our set of rules as a "model" for the data). Once again, the technique of cross-validation can be quite useful in estimating the true predictive accuracy of a set of rules, but again it can be quite computationally intensive, particularly if invoked repeatedly at different stages of the rule search.

We conclude this discussion on predictive rules by mentioning a few notable extensions to the basic classification paradigm above. The first extension is that, just as with classification and regression trees in Chapter 11, one can in principle perform *rule-based regression*. The left-hand side condition of a rule defines a particular region of the input space. Given this region one can then estimate a local regression model on the data in this region (it can be as simple as the best fitting constant for example). If the rules are disjoint one gets a piecewise local regression surface; if the rules overlap one must again decide on how to combine the different rule predictions in the overlapping regions. One particular advantage of the rule-based regression framework is the ease of interpretability, particularly in high-dimensional problems, since only a small fraction of the variables are often selected as being relevant for inclusion in the rules.

The second notable extension to the basic rule induction paradigm is that of using relational logic as the basis for the rules. A discussion in any depth of this topic is beyond the scope of this text, but essentially the idea is to generalize beyond the notion of propositional logic statements ("Variable = value") to what are known as first-order relational logic statements such as "Parent(X, Y) $\wedge$ Male(X) $\Rightarrow$ Father (X, Y)". This type of learning is in principle extremely powerful since it allows a much richer representational language to describe our data. A propositional version of a relational statement is typically quite awkward (and can be exponentially large) since there is no notion in the (simpler) propositional framework of objects and relations among objects. The extra representational power of relational logic comes at a cost of course, both in terms of reasoning with such rules and in terms of learning them from data. Algorithms for learning relational rules exist but to date are largely focused on learning logical relationships and, thus, have yet to have a substantial impact on typical real-world data sets from noisy environments.

13.10 Further Reading

Association rule problem was introduced in [**?**]. The Apriori algorithm is due to [**?**, **?**, **?**]. Extensions to algorithms and uses of the rules are discussed [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**] and lots of other papers.

The accuracy of the association rule is sometimes called confidence, and the frequency support.

Measures of interestingness for rules are discussed in [**?**, **?**]; the J-measure is from [**?**].

The light bulb problem is considered in [**?**].

A large number of different rule induction algorithms have been proposed in the machine learning literature, typically distinguished from each other in terms of the details of how the search is performed. The algorithm C4.5Rules (Quinlan 1987, 1993) is the best-known method for deriving rules from classification trees. The CN2 algorithm (Clark and Niblett, 1989) uses an entropy-based measure to select rules by manner of a beam search. Other more recent rule induction algorithms, with demonstrated ability to provide accurate classification on large data sets, include RL (Clearwater and Stern, 1991), Brute (Segal and Etzioni, 1994) and Ripper (Cohen, 1995) — rule induction designers seem to have a preference for rather obscure names! The RISE algorithm (Domingos, 1996) is an interesting example of a rule-induction algorithm using a specific-to-general search heuristic. Holte (1993) describes an interesting study in which very simple classification rule models provide classification accuracies about as good as other more complex and widely-used classifiers. Aronis and Provost (1997) discuss some practical tips on efficient implementation of rule induction algorithms for massive data sets.

Quinlan's FOIL algorithm (1990) was one of the first relational rule induction programs. More recent work in relational rule learning (also known as inductive logic programming) is summarized in texts such as Lavrac and Dzeroski (1994) and Muggleton (1995).

References

Agrawal, R., Imielenski, T. and Swami, A. (1993) Mining association rules between sets of items in large databases, *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD'98)*, New York, NY: ACM Press, 207–216.

Agrawal, R., and Srikant, R. (1994) Fast algorithms for mining association rules in large databases, *Proceedings of the Twentieth International Conference on Very Large Data Bases (VLDB'94)*, 487–499.

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Inkeri Verkamo, A. (1996), Fast discovery of association rules, in *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurasamy (eds.), Menlo Park, CA: AAAI Press, 307–328.

Aronis, J. M. and Provost, F. J. (1997) Increasing the efficiency of data mining algorithms with breadth-first marker propagation, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, Heckerman, D., Mannila, H., Pregibon, D. (eds.), Menlo Park, CA: AAAI Press, 119–122.

Brin, S., Motwani, R., and Silverstein, C. (1997) Beyond market baskets: generalizing association rules to correlations, *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD'97)*, New York, NY: ACM Press, 265–276.

Clark, P. and Niblett, T. (1989) The CN2 induction algorithm, *Machine Learning*, 3(4), 261–283.

Clearwater, S. and Stern, E. (1991) A rule-learning program in high-energy physics event classification, *Computational Physics Communications*, 67, 159–182.

Cohen, W. (1995) Fast effective rule induction, *Proceedings of the Twelfth International Conference on Machine Learning*, San Mateo: CA, Morgan Kaufmann, 115–123.

Domingos, P. (1996), Unifying instance-based and rule-based induction, *Machine Learning*, ????.

Fisher and Friedman, PRIM paper.

Fukuda, T. et al, (1996) Mining optimized association rules for numeric attributes, *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database and Knowledgebase Systems (PODS'96).*

Han, J. and Fu, Y. (1995) Discovery of multiple-level association rules from large databases, *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, San Mateo, CA: Morgan Kaufmann, 420-431.

Holsheimer, M., Kersten, M., Mannila, H., and Toivonen, H. (1995) A perspective on databases and data mining, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, U. M. Fayyad and R. Uthurasamy (eds.), 150–155.

Holte, R. C. (1993) Very simple classification rules perform well on most commonly used data sets, *Machine Learning*, 11, 63–91.

Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., Inkeri Verkamo, A. (1994) Finding interesting rules from large sets of discovered association rules, *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, New York, NY: ACM Press, 401–407.

Lavrac, N. and Dzeroski, S. (1994) *Inductive Logic Programming: Techniques and Applications*, Ellis Horwood.

Mannila, H., Toivonen, H., and Inkeri Verkamo, A. (1994) Efficient algorithms for discovering association rules, in *Knowledge Discovery in Databases: Papers from the AAAI-94 Workshop (KDD'94)*, Menlo Park, CA: AAAI Press, 181–192.

Meo, R., Psaila, G., and Ceri, S (1996) A new SQL-like operator for mining association rules, *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB'96)*, San Mateo, CA: Morgan Kaufmann.

Muggleton, S. (1995) *Foundations of Inductive Logic Programming*, Englewood Cliffs, NJ: Prentice Hall.

Park, J. S., Chen, M. S., and Yu, P. S. (1995) An effective hash-based algorithm for mining association rules, *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD'95)*, New York, NY: ACM Press, 175–186.

Quinlan, J. R. (1987) Generating production rules from decision trees, *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann, 304–307.

Quinlan, J. R. (1990) Learning logical definitions from relations, *Machine Learning*, 5, 239–266.

Quinlan, J. R. (1993) *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann.

Savasere, A., Omiecinski, E., and Navathe, S. (1995) An efficient algorithm for mining association rules, *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, San Mateo, CA: Morgan Kaufmann, 432–444.

Segal, R. and Etzioni, O. (1994) Learning decision lists using homogenous rules, *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Menlo Park, CA: AAAI Press, 619–625.

Silberschatz, A. and Tuzhilin, A. (1996) What makes patterns interesting in database systems, *IEEE Transactions on Knowledge and Data Engineering*, 8(6), 970–974.

Smyth, P. and Goodman, R. (1992) An information-theoretic approach to rule induction from databases, *IEEE Transactions on Knowledge and Data Engineering*, 4(4), 301–306.

Srikant, R. and Agrawal, R. (1995) Mining generalized association rules, *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, San Mateo, CA: Morgan Kaufmann, 407–419.

Srikant, R. and Agrawal, R. (1996) Mining quantitative association rules in large relational tables, *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD'96)*, New York, NY: ACM Press, 1–12.

Srikant, R., Vu, Q. and Agrawal, R. (1997) Mining association rules with item constraints, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97)*, Heckerman, D., Mannila, H., and Pregibon, D. (eds.), Menlo Park, CA: AAAI Press, 67–73.

Toivonen, H. (1996) Sampling large databases for association rules, *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB'96)*, San Mateo, CA: Morgan Kaufmann, 134–145.

Tsur, D et al (1998) QueryFlocks: a generalization of association rule mining, *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD'98)*, New York, NY: ACM Press, 1–12.