
Artificial Intelligence Review **16:** 177–199, 2001.

Understanding the Crucial Role of Attribute Interaction in Data Mining

ALEX A. FREITAS
Pontificia Universidade Catolica do Parana, Postgraduate Program in Computer Science, Rua Imaculada Conceicao, 1155, Curitiba, PR, 80215-901, Brazil
(E-mail: alex@ppgia.pucpr.br; URL: http://www.ppgia.pucpr.br/~alex)

Abstract. This is a review paper, whose goal is to significantly improve our understanding of the crucial role of attribute interaction in data mining. The main contributions of this paper are as follows. Firstly, we show that the concept of attribute interaction has a crucial role across different kinds of problem in data mining, such as attribute construction, coping with small disjuncts, induction of first-order logic rules, detection of Simpson's paradox, and finding several types of interesting rules. Hence, a better understanding of attribute interaction can lead to a better understanding of the relationship between these kinds of problems, which are usually studied separately from each other. Secondly, we draw attention to the fact that most rule induction algorithms are based on a greedy search which does not cope well with the problem of attribute interaction, and point out some alternative kinds of rule discovery methods which tend to cope better with this problem. Thirdly, we discussed several algorithms and methods for discovering *interesting* knowledge that, implicitly or explicitly, are based on the concept of attribute interaction.

Keywords: attribute interaction, classification, constructive induction, data mining, evolutionary algorithms, inductive logic programming, rule induction, rule interestingness, Simpson's paradox, small disjuncts

1. Introduction

Although the basic idea of extracting some sort of knowledge from data is quite old, we can consider that – at least from an academic viewpoint – the birth of modern data mining and knowledge discovery as a self-described scholarly, interdisciplinary field was the IJCAI-89 Workshop on Knowledge Discovery in Real Databases (see Piatetsky-Shapiro (1991)).

Before this workshop was held, most research on data mining consisted of mining small data sets that can be called "databases" only in the loosest sense of the word. A major research goal set forth by this workshop was to extract high-level knowledge from real-world database systems. It is well-known that real-world database systems are very large. Hence, in order to meet that important goal, several researchers have focused on designing algorithms that

are scalable with respect to the size of the data being mined (see e.g. Provost and Kolluri (1999), Freitas and Lavington (1998)).

However, real-world databases also offer other significant challenges to data mining, such as the ones mentioned by Frawley et al. (1991): (a) real-world databases tend to contain quite noisy data, which requires cost-effective data cleaning methods; (b) real-world databases contain many irrelevant attributes, which requires cost-effective attribute selection methods; (c) relevant attributes may be missing in real-world databases.

It should be noted that the above challenges are not strongly related to the size of real-world databases, but rather strongly related to the nature of the data, more precisely the fact that the data stored in real-world databases were collected for purposes other than data mining. This fact has another implication, which is relatively less investigated in the literature and is the focus of this paper.

In this paper we argue that one of the major challenges associated with real-world databases is that they tend to have a large degree of attribute interaction. Note that this is *not* the case in many data sets often used in machine learning and data mining research. For instance, one of the reasons why, in general, medical domains are so appropriate for knowledge discovery is that considerable medical expertise goes into selecting which attributes are included in a medical database, as pointed out by Piatetsky-Shapiro (1991). Medical doctors usually do not select attributes that are redundant or have strong interaction with other attributes, as noted by Michie et al. (1994).

The assumption that there is a relatively small degree of attribute interaction in the data being mined is implicit in most rule induction algorithms (see e.g. Rendell and Seshu (1990), Nazar and Bramer (1999), Dhar et al. (2000)). Unfortunately, this assumption does not usually hold when mining real-world database systems, and this is the basic problem addressed in this paper.

Although throughout the paper we discuss attribute interaction in several pattern-discovery tasks, in most of the paper we discuss attribute interaction in the context of a supervised prediction task, typically classification. In this context attribute interaction can be defined as follows. Consider three attributes Y, X_1 and X_2, where Y is the goal (class) attribute to be predicted and X_1 and X_2 are predictor attributes. X_1 and X_2 interact when the direction or magnitude of the relationship between Y and X_1 depends on the value of X_2. Actually, this can be called a two-way interaction. Higher-order attribute interactions can be defined in a similar way.

The goal of this paper is to significantly improve our understanding of the crucial role of attribute interaction in data mining. We believe this is an important goal to be pursued for at least three reasons. First, attribute interactions are the rule, and not the exception, in real-world database systems, as

will be discussed in section 2. Second, attribute interaction issues are rarely explicitly discussed in the literature. Third, the understanding of attribute interaction as a key concept in data mining can lead to the design of new kinds of data mining algorithms and methods specifically designed from scratch to take into account (and sometimes even take advantage of) the large degree of attribute interaction found in real-world database systems. Indeed, section 4 of this paper will discuss some recent algorithms and methods for discovering *interesting* knowledge that, implicitly or explicitly, are based on the concept of attribute interaction.

The rest of this paper is organized as follows. Section 2 discusses the importance of attribute interaction to support our argument that attribute interaction is a key concept in data mining. Section 3 presents a critical review of how most current rule induction systems cope with attribute interaction. Section 4 will show how attribute interaction can be used as the key concept for discovering interesting patterns. Finally, section 5 concludes the paper.

2. The Importance of Attribute Interaction in Data Mining

2.1. *Psychological evidence for the importance of attribute interaction*

Since the goal of data mining is to discover knowledge that is not only accurate but also comprehensible for human decision makers, the field of cognitive psychology is clearly relevant for data mining, as pointed out by Pazzani (2000).

The problem of classification has long been studied by psychologists. In their terminology a class is called a category, or a concept. Roughly speaking, until a few decades ago the field of psychology was dominated by a classical view of concepts, but in the last decades this view has been replaced by a natural view of concepts (see Gardner (1984)). A major difference between these two views has to do with attribute interaction.

In the classical view, categories are defined by a small set of attributes. All members of a category share these defining attributes, and no non-member shares them. In general the defining attributes are supposed to be largely independent (uncorrelated) of each other – i.e. there is little or no attribute interaction.

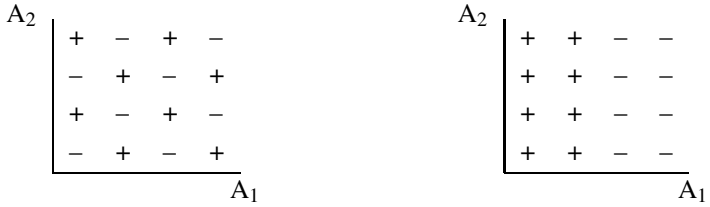
For instance, consider Bruner et al.'s classical psychological study of how individuals learn to form concepts, as summarized by Gardner (1984). In this study, subjects were asked to recognize instances of geometric concepts like *large red triangle* or *tall blue cylinder*. A concept was arbitrarily defined and each object was unambiguously considered a member or a non-member of that category. The subject was exposed to one card at a time, asked in each

case whether that card belonged to the target concept, and then told whether his/her response was correct. The subject's task was to identify the attributes defining the target concept, in order to select all, and only, those cards exhibiting the defining features of the concept. The researchers found that the most foolproof method used by subjects was *conservative focusing*, where one finds a positive instance of the concept and then makes a sequence of choices, each of which alters only a single attribute value of this first "focus" card and tests whether the change yields a positive or negative instance of the concept. This psychological study was quite influential, and it seems that at that time no one realized that the use of such artificial concepts might invalidate the findings.

In any case, we find a striking parallel between conservative focusing and the greedy search performed by some rule induction algorithms, particularly AQ-style algorithms – see Michalski (1983), which start with a seed example and then slowly generalize it by removing one-attribute-at-a-time. (The issue of greedy rule induction will be discussed later.)

By contrast, in the natural view of concepts, highly correlated (non-independent) attributes are the rule, not the exception. For instance, consider the classification of animals into birds and non-birds in the real world. Wings occur with feathers much more often than they do with furs. Hence, there is considerable interaction between predictor attributes, and this kind of interaction is exploited by human concept-recognition mechanisms. This situation is very different from the above example of classical view, where independent attributes are used to define artificial, arbitrary concepts. In the natural view, concepts are much fuzzier and are motivated by real-life considerations, rather than being arbitrarily defined.

To summarize, in the natural view of concepts, which is currently much more accepted in psychology than the classical view, attribute interaction is the rule, and not the exception. Evidence for this natural view of concepts is provided, in the context of data mining, by projects that found a significant degree of attribute interaction in real-world data sets. An example is the large number of small disjuncts found by Provost and Danyluk (1993) in telecommunications data, as mentioned in section 2.2. Another example are the several instances of Simpson's paradox discovered in real-world data sets by Fabris and Freitas (1999), as discussed in section 2.3. Yet another example is the existence of strong attribute interactions in a typical financial data set, as discussed by Dhar et al. (2000) – see also section 3.2.



(a) hard classification problem (b) easy classification problem

Figure 1. Large degree of attribute interaction makes a concept harder to learn.

2.2. *The influence of attribute interaction on concept hardness and small disjuncts*

There are, of course, many factors that make a concept (class description) difficult to be learned, including unbalanced class distributions, noise, missing relevant attributes, etc. However, in some cases even if all relevant information for class separation is included in the data – i.e. all relevant attributes are present, there is little noise, etc. – many rule induction algorithms achieve poor predictive accuracy when learning some concepts. These kinds of concepts can be called *hard* (see Rendell and Seshu (1990), Rendell and Cho (1990), Rendell and Ragavan (1993), Nazar and Bramer (1999)) and the main reason for their hardness is the problem of concept dispersion.

We summarize, in the following, the main arguments of Rendell et al. To simplify our discussion, let us assume a two-class problem where the concept to be learned consists of the positive-class examples. In essence, we can say that a concept is dispersed when we need a large number of small disjuncts (rules covering a few examples) to cover the concept. This point is illustrated in Figure 1, which shows graphical representations of the data space for two hypothetical data sets, each with two attributes. Figure 1(a) shows an extreme case where the positive-class ("+") examples are maximally spread across the data space. Each "+" example is surrounded by "–" examples, so one cannot induce reliable, generalized rules predicting the "+" class. Figure 1(b) shows the other extreme, where the "+" concept can be easily learned.

Note that the kind of concept dispersion shown in Figure 1(a) is mainly caused by attribute interaction. Indeed, in that Figure, in order to determine the class of an example we need to know the values of both predictor attributes, since a minor variation in the value of one attribute leads to an entirely different prediction. Knowing the value of a single attribute is useless. Although the above example is hypothetical and simplified for pedagogical reasons, data sets with a large degree of concept dispersion – i.e. with many small disjuncts – do exist in the real-world.

One of the earliest machine learning projects to cope with a large degree of attribute interaction in a real-world data set was Samuel's checker player developed during the 1950's by Samuel (1959). This project was very influential and it anticipated many ideas rediscovered later in machine learning research. (For instance, Samuel's adaptive polynomial learning, which adjusted the weights of a polynomium of attributes for evaluating a board position, was quite similar to perceptron neural networks.)

This project showed that to learn the concept of 'winning' well, it was necessary to use high-level attributes such as piece advantage, which are usually a good indicator of the strength of a position. This kind of representation effectively helps to group together examples of the same class in the data space, which greatly reduces the number of small disjuncts. By contrast, consider a low-level representation where each attribute indicates whether a board square was occupied by a black king, black man, red king, red man, or vacant. This kind of representation would lead to a much larger number of small disjuncts, since now there is no small set of attributes that can be considered a good indicator of the strength of a position. With this low-level representation one would need to consider the value of virtually all the attributes to evaluate the strength of a position, and a minor variation in the value of one of the many attributes might lead to a completely different evaluation of the position.

More recently, a number of data mining projects have had to cope with a large number of small disjuncts, due to a large degree of attribute interaction. A good example is a real-world application reported by Danyluk and Provost (1993), where small disjuncts cover roughly 50% of the training examples.

This fact has probably influenced the design of some rule induction algorithms. For instance, Domingos (1995)'s RISE algorithm tends to cope with small disjuncts better than most rule induction algorithms. The reason is that, in a nutshell, this algorithm starts with a rule set where each rule is a training example, and it generalizes a rule only if this generalization increases the global predictive accuracy on the training set. Hence, small-disjunct rules will not be unduly generalized by RISE.

In addition, several methods for coping with small disjuncts have been proposed, see Holte et al. (1989), Ting (1994), Weiss (1995, 1998), Weiss and Hirsh (2000), Carvalho and Freitas (2000a,b). Despite these advances, the problem of small disjuncts (related to a high degree of attribute interaction) is still an open problem in data mining.

To summarize, in general the more dispersed the positive examples of a concept are, the more difficult that concept is to be learned. Low-level representations, such as the one illustrated in Figure 1(a) are associated with many small disjuncts and much concept dispersion due to a high degree of attribute

Table 1. Simpson's paradox in data about tuberculosis deaths.

	New York		Richmond	
Total population	4766883		127682	
no. of deaths	8878		286	
percentage	0.19%		0.22%	
	New York		Richmond	
	white	coloured	white	coloured
Total population	4675174	91709	80895	46733
no. of deaths	8365	513	131	155
percentage	0.18%	0.56%	0.16%	0.33%

interaction, which tends to lead to poor predictive accuracy. By contrast, high-level representations, such as the one illustrated in Figure 1(b), greatly reduce the degree of attribute interaction, which tends to lead to a less dispersed concept and so to a higher predictive accuracy.

2.3. *Detecting occurrences of Simpson's paradox*

Attribute interaction is at the core of an occurrence of Simpson's paradox. This paradox can be briefly described as follows (see Simpson (1951), Wagner (1982), Newson (1991)). Suppose that a population is partitioned into two, Pop_1 and Pop_2, and that the probability of a given situation of interest – i.e. the value of a given goal attribute – *increases (decreases)* from Pop_1 to Pop_2. Suppose now that both Pop_1 and Pop_2 are further partitioned into several categories. We say that Simpson's paradox occurs if the probability of the situation of interest *decreases (increases)* from Pop_1 to Pop_2 in *each* of those categories. The idea is better illustrated with an example.

Table 1 shows an occurrence of the paradox in a comparison of tuberculosis deaths in New York City and Richmond, Virginia, during the year 1910 (see Newson (1991)). Overall, the tuberculosis mortality rate of New York was lower than Richmond's one. However, the opposite was observed when the data was further partitioned according to two racial categories: white and coloured. In both these racial categories Richmond had a lower mortality rate. In other words, if we consider only one attribute (*City*) we draw a conclusion, while if we consider the interaction between two attributes (*City* and *Racial Category*) we draw the opposite conclusion.

Simpson's paradox occurs more often than one might think at first glance. For instance, several real-world instances of the paradox are described by

```
Procedure for Rule Set Induction
Specify the training set of examples;
DO
   induce a rule covering as many "+" examples and as few "–" examples as possible
   remove the "+" examples covered by the induced rule from the training set
UNTIL there are uncovered "+" examples in the training set
```

(a) building a rule set on a one-rule-at-a-time basis

```
Top-Down Procedure for Rule Induction
Rule = ∅;
DO
   /* specialize the rule */
   add the "best" condition to the rule;
UNTIL (stopping criterion)
```

(b) top-down rule construction

```
Bottom-Up Procedure for Rule Induction
Rule = a conjunction of conditions;
DO
   /* generalize the rule */
   remove "worst" condition from the rule;
UNTIL (stopping criterion)
```

(c) bottom-up rule construction

Figure 2. High-level view of greedy search in rule induction.

Newson (1991) and Wagner (1982). In addition, Fabris and Freitas (1999) have shown that the paradox occurs in several data sets from the UCI repository, as will be seen in section 4.4.

3. A Critical Review of Rule Induction Systems Concerning How They Cope with Attribute Interaction

3.1. *The myopia of greedy rule induction*

In general a rule induction algorithm is said to be greedy if: (1) it constructs a rule in an incremental fashion by considering one-attribute-at-a-time; (2) at each step the best possible local choice is made. While there are many different kinds of rule induction algorithm, the majority of them is greedy, performing a local search as described in Figure 2. Figure 2(a) shows that this kind of search discovers one rule at a time. The procedure performed to build a single rule can be top-down or bottom-up, as shown in Figures 2(b) and 2(c). In both cases, rules are typically built by considering one attribute at a time and choosing the best attribute to be included/removed in/from the rule in a greedy fashion.

It should be noted that the greedy search performed by most rule induction algorithms makes them quite sensitive to attribute interaction problems. A very simple example of the danger of greedily constructing a rule by selecting one-attribute-at-a-time is shown in the hypothetical data set of Figure 3. The

credit_limit	*c/a_balance*	*is_credit_abnormal?*
low	low	no
low	high	yes
high	low	yes
high	high	no

Figure 3. Attribute interaction in a kind of logic XOR (eXclusive OR) function.

first column indicates whether the credit limit granted to a customer is low or high. The second column indicates whether the balance of the customer's current account is low or high. The third column, corresponding to the goal (class) attribute, indicates whether or not the credit limit is abnormal, in the sense of being incompatible with the current account balance of the customer. The value of this goal attribute is given by a kind of logic XOR (eXclusive OR) function, which is true ('yes') if and only if the two predictor attributes have different values, and is false ('no') otherwise.

Suppose that we try to induce a rule from the data in Figure 3 by selecting one attribute-value condition at a time. Suppose we include in our rule the condition <*credit_limit* = low>. This condition is not useful for predicting the value of *is_credit_abnormal?*, since the class distribution for the examples satisfying the condition is 50–50%, which is the same class distribution as in the entire data set. Actually, any of the four rule conditions (attribute-value pairs) which could be included in a rule leads to the same problem, so a greedy rule induction algorithm would conclude that these rule conditions are irrelevant for predicting the goal attribute. However, this is not the case. The problem is that we need to know the value of both predictor attributes at the same time to accurately predict the value of the goal attribute.

Note that one of the most popular kind of data mining algorithm, namely decision-tree algorithms, would be fooled by the simple attribute interaction problem shown in Figure 3. For instance, assume that the data set shown in Figure 3 includes not only attributes *credit_limit* and *c/a_balance*, but also another attribute, say *gender*, which is irrelevant for predicting *is_credit_abnormal?* but turns out to be slightly correlated with *is_credit_abnormal?* by sheer chance (i.e. a spurious correlation). No matter how small this spurious correlation is, the irrelevant attribute *gender* would be chosen to label the root node of a decision tree, since the two relevant attributes *credit_limit* and *c/a_balance* (when considered separately, i.e. one-at-a-time) have no correlation at all with *is_credit_abnormal?*.

In passing, we note that the above-discussed kind of logical XOR problem is in reality a particular a case of parity problems, where the target function returns true if and only if an odd number of predictor attributes is true. The

complexity of attribute interaction in parity problems increases very fast with the number of predictor attributes, which makes this kind of problem very difficult for greedy rule induction algorithms, as shown by Schaffer (1993).

Similar arguments hold for the bottom-up rule induction procedure shown in Figure 2(c). This is the kind of search performed, for example, by AQ-style rule induction algorithms (see Michalski (1983)), which start with a seed example and then generalize it, in a greedy fashion, by removing one-condition-at-a-time. In other words, both rule induction procedures in Figure 2(b) and Figure 2(c) are greedy, they are simply greedy in "opposite directions".

The above discussion was somewhat simplified for pedagogical reasons, but the basic idea is that, in order to cope well with attribute interactions, we need data mining methods which consider several-attributes-at-a-time.

There are other knowledge discovery paradigms that cope better with attribute interaction than rule induction. An example is neural nets. However, despite the progress in the area of extracting comprehensible rules from neural nets (see Taha and Ghosh (1999), Vaughn (1996)) this extraction still remains a difficult, cumbersome task. Hence, we do not discuss neural networks here. Rather, we discuss below other knowledge discovery paradigms that not only tend to cope better with attribute interaction than the classical rule induction paradigm but also lend themselves more naturally to the discovery of comprehensible rules.

3.2. *The global search of evolutionary algorithms*

An attempt to cope better with attribute interaction is to avoid the idea of greedy search altogether and perform a more global search in the rule space. This can be achieved with evolutionary algorithms such as genetic algorithms (see Michalewicz (1996)) and genetic programming (see Banzhaf (1998)).

An important characteristic of evolutionary algorithms is that they perform a global search. This is due to several factors. First of all, evolutionary algorithms work with a population of candidate solutions, rather than working with a single candidate solution at a time. These solutions concurrently explore different parts of the search space. Second, the major genetic operator, crossover, modifies individuals on a several-genes (attributes)-at-a-time basis, rather than on a single-gene (attribute)-at-a-time basis. Third, the fitness function evaluates an individual as a whole. In the context of data mining, this corresponds to evaluating a candidate rule (or a candidate rule set) as a whole, rather than one condition at a time (or a rule at a time).

Finally, evolutionary algorithms use stochastic search operators, which contributes to make them more robust and less sensitive to noise.

Intuitively, the above factors make evolutionary algorithms cope better with attribute interaction than the greedy search algorithms often used in rule induction. Some evidence that this is the case is provided by Greene and Smith (1993), who show that, as the amount of interaction increases, the relative performance between a genetic algorithm and two rule induction algorithms (CN2 and NewId, the latter a decision tree algorithm) becomes increasingly larger. Other evidence for the fact that evolutionary algorithms cope well with attribute interaction is provided by Dhar et al. (2000). They have compared a genetic algorithm with two rule induction algorithms, namely RL and CART (a well-known decision-tree algorithm), on a financial data set involving a considerable amount of attribute interaction. The GA outperformed the other two algorithms due to its ability to perform a more global, thorough search in rule space. To quote from the conclusions of Dhar et al. (2000), p. 278:

> We claim that for hard problems, genetic algorithms, appropriately 'fixed up', are more thorough in their search than other rule learning algorithms. Our results support this claim. GLOWER indeed is less restricted by greedy search biases, and for problems with weak structure or variable interactions, it is precisely the subtle relationships that are useful discoveries.

Recent collections of papers on data mining with evolutionary algorithms can be found in Zytkow (1999), Freitas (1999), Freitas (2000). Section 3.3.1 will revisit evolutionary algorithms in the context of constructive induction.

It should be noted that evolutionary algorithms also have some disadvantages in the context of data mining. Arguably, one of the most serious problems, at least in the context of very large databases, is the fact that they tend to take much longer to run than greedy rule induction algorithms. This disadvantage is partially mitigated by their potential for parallel processing (see e.g. Neri and Giordana (1995), Anglano et al. (1997), Araujo et al. (1999), Freitas and Lavington (1998)), but scalability with respect to the size of the database being mined is still an open problem for evolutionary algorithms.

3.3. *The crucial role of attribute interaction in attribute construction*

With respect to the autonomy of a rule induction algorithm, one can make an important distinction between two kinds of algorithms. Roughly speaking, some algorithms discover rules by just selecting attribute values among the original set of input attributes and their corresponding domains, while other algorithms discover rules by not only selecting but also automatically constructing new attributes.

The former kind of algorithm includes many well-known rule induction algorithms, such as Quinlan's (1993) C4.5. These algorithms carefully choose attribute-value pairs to include in rule conditions. However, they are incapable of constructing relatively simple new attributes, which can be quite useful in some situations. For instance, suppose we try to predict whether the shares of a company will go up or down in the financial market, based on a set of predictor attributes that includes both the company's total income and the company's total expenditure in the last 12 months. Most rule induction algorithms are not capable of discovering rules of the form:

IF (*Income* > *Expenditure*) AND ... THEN (*Shares* = *up*)

IF (*Income* < *Expenditure*) AND ... THEN (*Shares* = *down*),

because those algorithms do not have the autonomy to create attribute-attribute (rather than attribute-value) rule conditions. Clearly, this limitation would be removed if the rule induction algorithm was able to automatically construct the new boolean attribute "*Income* > *Expenditure?*".

More generally, an attribute construction algorithm might face attribute interaction problems of arbitrary complexity in its search for high-level, effective new attributes. The better the algorithm can cope with attribute interaction, the more useful the new constructed attributes will probably be.

A review of constructive induction algorithms is beyond the scope of this paper. The interested reader is referred to Liu and Motoda (1998) and an online bibliography at http://liinwww.ira.uka.de/bibliography/Ai/feature.engineering.html.

It is important to note, however, that most constructive induction methods follow the greedy, local search paradigm of rule induction discussed in section 3.1. Hence, we briefly discuss in subsection 3.3.1 an alternative approach for constructive induction based on genetic programming, with the aim of performing a more global search in the space of new candidate attributes.

3.3.1. *Constructive induction with genetic programming*

As mentioned above, the major problem in attribute construction is that the search space tends to be huge. In addition, the construction of a good attribute often requires that the attribute construction algorithm generates and evaluates combinations of several original attributes, rather than just two attributes. This is where evolutionary algorithms can be useful. In particular, genetic programming (GP), due to its characteristic of performing a very open-ended, stochastic, global search (see Banzhaf (1998)) seems particularly suitable for efficiently searching a huge space of new candidate attributes.

In GP an individual is usually represented by a tree, with rule conditions and/or attribute values in the leaf nodes and functions (e.g. logical, relational

or mathematical operators) in the internal nodes. An individual's tree can grow in size and shape in a very dynamic way. In general, in order to apply GP to data mining one must define a terminal set, containing the attributes and their corresponding domain values, and a function set, containing the operators to be applied to the terminal set elements.

In the context of rule induction and attribute construction, the important point is that we can include virtually any kind of operator in the function set of GP. In the first iteration (generation), operators will be applied to original attributes in a kind of random way. However, as the population of individuals (new candidate attributes) evolves, the system will automatically discover which operator must be applied to which combination of original attributes in order to create good new attributes (the best evolved individuals).

Evidence that GP is an effective method for constructive induction is presented e.g. by Hu (1998) and Kuscu (1999). Hu's results are particularly interesting, because they involved more data sets and more algorithms being compared. More precisely, Hu compares CPCI, a GP-based constructive induction method, with two other constructive induction methods, namely LFC and GALA. The comparison is made across 12 data sets. Overall, the predictive accuracy of GPCI was considerably better than LFC and somewhat better than GALA.

3.4. *Coping with attribute interaction via Inductive Logic Programming (ILP)*

Although it is rarely described this way, Inductive Logic Programming (ILP) can be regarded as a way to cope with attribute interaction.

One of the basic ideas of ILP is to use a richer, more expressive knowledge representation language, based on first-order logic (FOL) (see e.g. Lavrac and Dzeroski (1994), Quinlan (1990)). This representation is more expressive than the traditional propositional ("zero-th order"), attribute-based representation used by most rule induction algorithms. A very simple example shows the point. Let A and B be two boolean predictor attributes. A concept defined by the equality of these two attributes would be represented in propositional logic as: ((A = true) AND (B = true)) OR ((A = false) AND (B = false)). The same concept would be represented in FOL as: $A = B$, which is obviously a much more compact representation. Another example of the representational power of first-order logic is the fact that it allows the discovery of rule conditions such as "*Income* > *Expenditure?*", as mentioned in section 3.3. Note: assuming that the *Income* and *Expenditure* attributes are real-valued, we would need an infinite number of propositional rule conditions to express a rule set equivalent to the FOL rule condition "*Income* > *Expenditure?*".

Another basic idea of ILP is that the rule induction algorithm accepts as input not only the training set of examples, but also a set of domain (background) knowledge predicates. Actually, FOL is a useful tool to cope with attribute interaction because it allows us to express in a single term (predicate) arbitrarily complex relationships among attributes, rather than attribute values. In essence, a predicate in FOL corresponds to a relation in relational databases, while the predicate arguments (variables) correspond to attributes in relational databases. An example is the predicate *better_investment(A, B)*, which means that an investment in *A* is better than one *B* if a set of arbitrarily complex conditions are satisfied. These predicate conditions can be intentionally defined by logic clauses such as "*higher_interest_rate(A, B) AND smaller_risk(A, B)*", which is analogous to a view definition in relational databases, or extensionally defined by a set of grounded facts satisfying the condition, which is analogous to enumeration of the set of tuples belonging to the relation.

Hence, we can say that the use of a FOL background predicate allows ILP systems not only to capture an arbitrarily complex attribute interaction among attributes but also to express that interaction in a compact, abstracted form. However, it should be noted that, despite this advantage, most ILP systems still have two limitations for coping with attribute interaction, as follows.

First, background predicates are usually manually predefined. They correspond either to relations (or views) predefined in the database or to logical clauses (or sets of grounded facts) manually specified specifically for the target data mining task. In other words, most ILP systems lack the autonomy to build background predicates. Some exceptions will be seen in the next subsection, where we discuss ILP-based constructive induction.

Second, most ILP systems use the same kind of greedy, local search strategy as used by most rule induction algorithms. Indeed, a high level description of many ILP algorithms could consist of the pseudo-code in Figure 2 with a simple modification: replace the attribute-based words *condition* and *rule* by the FOL words *literal* and *clause*, respectively (see e.g. a similar pseudo-code in Quinlan (1990)).

3.4.1. *ILP-based constructive induction*

An interesting approach is to combine the search autonomy of constructive induction procedures with the more expressive representation language of FOL typically used in ILP systems. Here we briefly mention a couple of examples of this approach.

LINUS converts all possible combinations of background predicates into attributes, which are then given as input to a propositional, attribute-based

rule induction algorithm (see Dzeroski and Lavrac (1993)). The discovered rules are then converted back into FOL predicates. This transformation of background predicates into attributes can be regarded as a form of constructive induction.

Another ILP-based constructive induction method is proposed by Srinivasan and King (1999). This method essentially works as follows. An ILP algorithm is used to induce clauses. Then the algorithm identifies subsets of literals of the discovered clauses that can be used as attributes. Each identified subset of literals consists of a conjunctive sequence of literals, and each of these conjunctive sequences is considered a new attribute.

Note that both of the above methods essentially generate *all* possible combinations of background predicates that can be used as predictor attributes by a rule induction algorithm. Hence, the new-attribute generation procedure performs a kind a exhaustive-search procedure. There is no heuristics to limit the number of candidate attributes generated by the system. Heuristics are used only in the form of evaluation functions to select the best new attributes, among all generated candidates. This kind of exhaustive generation of new candidate attributes does not seem to be scalable to databases with a high number of background predicates.

This is in contrast with alternative approaches for constructive induction that perform a more heuristic search in the space of new candidate attributes, such as the approach discussed in section 3.3.1.

4. Attribute Interaction as a Key Concept for Discovering Interesting Patterns

The concept of attribute interaction can be a valuable tool to detect surprising knowledge. Indeed, as argued above, human analysts usually analyze data on an one-attribute-at-a-time basis (see Gardner (1984), Brazdil and Henery (1994), Michie et al. (1994)) and they have difficulty in analyzing data on a several-attributes-at-a-time basis. Hence, the kind of rule deemed interesting by a user is probably a non-compositional rule, where – due to attribute interactions – the relationship expressed in the rule as a whole is quite different from the relationship that is expressed in separate parts of the rule. This section will address this issue in detail, discussing four methods to discover interesting patterns based on the key concept of attribute interactions.

4.1. *Attribute interaction as the key concept in pruning and summarizing discovered rules*

Liu et al. (1999) proposed a method for summarizing and pruning discovered rules. Their method is implicitly based on detecting attribute interaction, even though this point is not explicit in their work. In essence, the method works as follows. Let X → Y be a rule, where X is a conjunction of attribute-value conditions and Y is a single attribute-value pair. X and Y are said to be correlated if: (a) the rule support exceeds a user-defined minimum support; and (b) X and Y are deemed correlated by using a chi-squared test at a user-defined significance level.

If these two conditions are met, the system determines whether X and Y have a positive or a negative correlation. The correlation is considered positive (negative) if the observed frequency of examples satisfying both X and Y is greater (smaller) than the expected frequency assuming statistical independence between X and Y.

The system then finds all direction setting rules, which are to be included in the summary of rules to be reported to the user. Note that a direction setting rule can be considered an interesting rule, for the purposes of our discussion. A precise definition of a direction setting rule can be found in the original paper. For the purpose of our discussion it is enough to say that: (a) a k-condition rule r is viewed as a combination of two rules, a 1-condition rule r_1 and a (k-1)-condition rule r_{rest} with the same consequence (there are k such combinations); (b) if the correlation of the rule r is expected with respect to any of the k combinations, then it is not a direction setting rule.

These ideas are better explained with an example. Assume that the rule r given by *Job=yes AND Own_house=yes* → *Loan=approved* is positively correlated. Suppose that both the rule *Job=yes* → *Loan=approved* and the rule *Own_house=yes* → *loan=approved* are positively correlated. Then the rule r is not a direction setting rule, since it is not interesting (two positive correlations are expected to lead to a positive correlation). Now suppose instead that both the rule *Job=yes* → *Loan=approved* and the rule *Own_house=yes* → *Loan=approved* are negatively correlated. Then the rule r is a direction setting rule, since it is interesting (two negative correlations are unexpected to lead to a positive correlation).

In effect, a rule r is considered interesting (direction setting) when attribute interaction makes the correlation of the rule to be different from the correlations of the rule combinations formed from r.

At first glance perhaps one might argue that this method is greedy because the k rule combinations are generated by picking one condition at a time to form rule r_1. However, note carefully that *all* rule combinations must pass the test of having a correlation different from that of the original rule. Therefore,

the method does incorporate a "global" (rule-wide) test of correlation reversal and effectively detects attribute interactions.

4.2. *Attribute interaction as the key concept in finding surprising rules*

Freitas (1998) proposed that a rule be considered as surprising knowledge to the extent that it predicts a class different from the classes predicted by its minimum generalizations. In essence, this method works as follows. Let a rule antecedent be a conjunction of m conditions, of the form cond_1 AND cond_2 AND ... cond_m. A rule has m minimum generalizations, one for each of its m conditions. The k--th minimum generalization of the rule, $k = 1, \ldots,$ m, is obtained by removing the k-th condition from the rule.

Note that a minimum generalization g of a rule r covers a superset of the examples covered by r. As a result, the class distribution of the examples covered by g can be significantly different from the class distribution of the examples covered by r. Therefore, assuming that a rule predicts the majority class of the examples covered by the rule, after creating the generalized rule g the system has to re-compute the class predicted by g, which can be different from the class predicted by the original rule r. Let C be the class predicted by the original rule r and let C_k be the class predicted by the k-th minimum generalization g_k of r. Then the system compares C with each C_k, $\text{k} = 1, \ldots,$ m, and counts the number of times that C differs from C_k. The higher the value of this count, the higher the degree of surprisingness (interestingness) assigned to the original rule r.

In other words, the system effectively considers that a rule r has a large degree of surprisingness when attribute interaction makes r cover a set of examples whose majority class is different from the majority class of the sets of examples covered by most of the minimum generalizations of r.

4.3. *Attribute focusing*

Attribute Focusing is a technique designed for detecting interesting attribute values, in the sense that the values differ from an expected value. Bhandari (1993), Bhandari and Biyani (1994) proposed two methods for detecting interesting attribute values. The first method consists of finding interesting values of a given attribute by comparing the observed frequency of that value with its expected frequency assuming a uniform probability distribution. Since this is a one-dimensional method, analyzing just one attribute at a time, it involves no attribute interaction and so will not be further discussed here.

The second method, however, is very relevant for our discussion. It analyzes one pair of attributes at a time. An interestingness function I_2 is

used to detect an interesting pair of attribute values, where each of the values belong to a different attribute of a given pair of attributes.

The function I_2 measures how much the observed joint frequency of a pair of attribute values deviates from the expected frequency assuming that the two attributes are statistically independent. More precisely,

$$I_2(A = V_i, B = V_j) = |Pr(A = V_i, B = V_j) - Pr(A = V_i) \times Pr(B = V_j)|, \quad (1)$$

where A is one of the attributes being analyzed, V_i is i-th value of the domain of A, $\Pr(A = V_i)$ is the probability of attribute A having value V_i; B, V_j and $\Pr(B = V_j)$ are defined in the obviously analogous way; $\Pr(A = V_i, B = V_j)$ is the probability that both A has value V_i and B has value V_j; and |x| denotes the absolute value of x. $\Pr(A = V_i)$ is computed as the ratio of the number of records in which $A = V_i$ over the number of records in which A has some value. $\Pr(B = V_j)$ is computed in analogous way. $\Pr(A = V_i, B = V_j)$ is computed as the ratio of the number of records in which both $A = V_i$ and $B = V_j$ over the number of records in which both A and B have some value. A pair of attribute values is considered interesting if its I_2 value is greater than a user-specified threshold.

Hence, the essence of Attribute Focusing (using the interestingness function I_2) is precisely to detect attribute values whose interactions produce unexpected observed joint frequency. Note that this basic idea is similar to the basic idea of the method discussed in section 4.1 – though the latter is considerably more elaborated and discovers high-level rules, rather than just pairs of attribute values.

Although the basic idea of attribute focusing is quite simple, it has been effectively used to discover interesting knowledge in real-world data by Bhandari (1993), Bhandari and Biyani (1994) and it has been used as the basis for additional research in data mining. We briefly mention two examples of this additional research.

Goil and Choudhary (1997) have extended Attribute Focusing for multi-dimensional databases (data cubes). A contribution of this work was to introduce a parallel algorithm to compute the above-discussed interestingness function I_2. This research addressed the problem of making Attribute Focusing more computationally efficient, which is important in the context of large data cubes. However, it did not adapt Attribute Focusing to one of the major characteristics of data cubes, namely the fact that dimensions contain *hierarchical* attributes.

This characteristic of data cubes introduces new opportunities and requirements for adapting the computation of the interestingness function I_2. For

instance, suppose we use Attribute Focusing to analyze sales of a product and find interesting combinations of values of two attributes, say *Store*, with hierarchy: store → city → state, and *Time*, with hierarchy: day → month → year. Should the function I_2 for combinations of states and years (highest hierarchical level of both attributes) be computed in the same way as for the combinations of stores and days (lowest hierarchical level of both attributes)? What about combinations of stores and years (mixed hierarchical levels)? These questions are addressed by Fabris and Freitas (2000), where the computation of the function I_2 takes into account a correction factor based on the current hierarchical levels of the two attributes being analyzed by Attribute Focusing.

4.4. *Discovering surprising patterns by detecting occurrences of Simpson's paradox*

Freitas (1998) has designed an algorithm that searches for all occurrences of Simpson's paradox in a data set. The motivation for this algorithm is that Simpson's paradox, due to its paradoxical nature, can be deemed a surprising pattern. Hence, intuitively, occurrences of the paradox are a potentially interesting output for a data mining algorithm.

Fabris and Freitas (1999) have applied this algorithm to seven data sets of the UCI repository (http://www.ics.uci.edu/~mlearn/MLRepository.html). They have discovered in total 13 instances of the paradox in four data sets, namely seven instances in Voting Records, two instances in Hepatitis, two instances in Australian Credit and two instances in Nursery.

Note that, as mentioned in section 2.3, attribute interaction is at the core of an occurrence of Simpson's paradox. Hence, this work can be regarded as a direct approach to make the detection of attribute interaction the central goal of a data mining algorithm. In addition, it can be regarded as an algorithm designed from scratch to discover interesting patterns, rather than first discover many patterns and then pass them through a filter that selects the most interesting patterns.

5. Conclusion

We have argued that attribute interaction is a key concept of data mining that has been relatively little investigated in the literature, at least in an explicit form. Hence, the general goal of this paper was to significantly increase our understanding of this concept, in order to eventually support the design of better data mining systems.

The main contributions of this paper are as follows. Firstly, we showed that the concept of attribute interaction has a crucial role across different kinds of problem in data mining, such as attribute construction, coping with small disjuncts, induction of first-order logic rules, detection of Simpson's paradox, and finding several types of interesting rules. Hence, a better understanding of attribute interaction can lead to a better understanding of the relationship between these kinds of problems, which are usually studied separately from each other.

Secondly, we drew attention to the fact that most rule induction algorithms are based on a greedy search which does not cope well with the problem of attribute interaction, and pointed out some alternative kinds of rule discovery methods which tend to cope better with this problem.

Thirdly, we discussed several algorithms and methods for discovering interesting knowledge that, implicitly or explicitly, are based on the concept of attribute interaction.

We hope that the insights provided by this paper can guide the design of more effective data mining algorithms, which take into account the large degree of attribute interaction typically found in real-world database systems.

References

Anglano, C., Giordana, A., Lo Bello, G. & Saitta, L. (1997). A Network Genetic Algorithm for Concept Learning. *Proc. 7th Int. Conf. Genetic Algorithms*, 434–441. Morgan Kaufmann.

Araujo, D. L. A., Lopes, H. S. & Freitas, A. A. (1999). A Parallel Genetic Algorithm for Rule Discovery in Large Databases. *Proc. 1999 IEEE Systems, Man and Cybernetics Conf., v. III*, 940–945. Tokyo.

Banzhaf, W., Nordin, P., Keller, R.E. & Francone, F.D. (1998) *Genetic Programming, an Introduction: on the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann.

Bhandari, I. (1993). Attribute Focusing: Machine-assisted Knowledge Discovery Applied to Software Production Process Control. *Proc. 1993 Workshop on Knowledge Discovery in Databases*, 61–69. AAAI Press.

Bhandari, I. & Biyani, S. (1994). On the role of statistical significance in exploratory data analysis. *Proc. AAAI-94 Workshop on Knowledge Discovery in Databases*, 61–72. AAAI Press.

Brazdil, P. B. & Henery, R. J. (1994). Analysis of Results. In Michie, D., Spiegelhalter, D.J. & Taylor, C.C. (eds.) *Machine Learning, Neural and Statistical Classification*, Chapter 10. Ellis Horwood.

Carvalho, D. R. & Freitas, A. A. (2000a). A Hybrid Decision Tree/Genetic Algorithm for Coping with the Problem of Small Disjuncts in Data Mining. *Proc. Genetic and Evolutionary Computation Conf. (GECCO-2000)*, 1061–1068. Las Vegas, NV, USA.

Carvalho, D. R. & Freitas, A. A. (2000b). A Genetic Algorithm-based Solution for the Problem of Small Disjuncts. *Principles of Data Mining and Knowledge Discovery (Proc. 4th*

European Conf., PKDD-2000). Lecture Notes in Artificial Intelligence 1910, 345–352. Springer-Verlag.

Danyluk, A. P. & Provost, F. J. (1993). Small Disjuncts in Action: Learning to Diagnose Errors in the Local Loop of the Telephone Network. *Proc. 10th Int. Conf. Machine Learning*, 81–88.

Dhar, V., Chou, D. & Provost, F. (2000). Discovering Interesting Patterns for Investment Decision Making with GLOWER – A Genetic Learner Overlaid with Entropy Reduction. *Data Mining & Knowledge Discovery* **4**(4): 251–280.

Domingos, P. (1995). Rule Induction and Instance-based Learning: a Unified Approach. *Proc. 14th Int. Joint Conf. on Artif. Intel. (IJCAI-95)*, 1226–1232.

Dzeroski, S. & Lavrac, N. (1993). Inductive Learning in Deductive Databases. *IEEE Trans. Knowledge and Data Engineering* **5**(6): 939–949.

Fabris, C. C. & Freitas, A. A. (1999). Discovering Surprising Patterns by Detecting Occurrences of Simpson's Paradox. In Bramer, M. et al. (eds.) *Research and Development in Intelligent Systems XVI*, 148–160. Springer-Verlag.

Fabris, C. C. & Freitas, A. A. (2000). Incorporating Deviation-detection Functionality into the OLAP Paradigm. *Unpublished manuscript*.

Frawley, W. J., Piatetsky-Shapiro, G. & Matheus, C. J. (1991). Knowlege Discovery in Databases: An Overview. (1991) In Piatetsky-Shapiro, G. & Frawley, W.J. (eds.) *Knowledge Discovery in Databases*, 1–27. AAAI/MIT Press.

Freitas, A. A. (1998). On Objective Measures of Rule Surprisingness. *Principles of Data Mining & Knowledge Discovery (Proc. PKDD'98) – Lecture Notes in Artif. Intel. 1510*, 1–9. Springer-Verlag.

Freitas, A. A. (ed.) (1999). *Data Mining with Evolutionary Algorithms: Research Directions – Papers from the AAAI Workshop*. Technical Report WS-99-06. AAAI.

Freitas, A. A. (ed.) (2000). Data Mining with Evolutionary Algorithms Workshop. In Wu, A. S. (ed.) *Proc. of the 2000 Genetic and Evolutionary Computation Conf. Workshop Program*, 69–92. Las Vegas, NV, USA.

Freitas, A. A. & Lavington, S. H. (1998). *Mining Very Large Databases with Parallel Processing*. Kluwer.

Greene, D. P. & Smith, S. F. (1993). Competition-based Induction of Decision Models from Examples. *Machine Learning* **13**, 229–257.

Gardner, H. (1984). *The Mind's New Science: A History of the Cognitive Revolution.* Basic Books.

Goil, S. & Choudhary, A. (1997). High Performance OLAP and Data Mining on Parallel Computers. *Data Mining and Knowledge Discovery* **1**(4): 391–417.

Holte, R. C., Acker, L. E. & Porter, B. W. (1989). Concept Learning and the Problem of Small Disjuncts. *Proc. Int. Joint Conf. Artif. Intel. (IJCAI-89)*, 813–818.

Hu, Y-J. (1998). A Genetic Programming Approach to Constructive Induction. *Genetic Programming 1998: Proc. 3rd Annual Conf.*, 146–151. Morgan Kaufmann.

Kuscu, I. (1999). A Genetic Constructive Induction Model. *Proc. Congress on Evolutionary Computation (CEC-99)*, 212–217. Washington D.C., USA.

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

Liu, H. & Motoda, H. (1998). *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Kluwer.

Liu, B., Hsu, W. & Ma, Y. (1999). Pruning and Summarizing the Discovered Associations. *Proc. 4th Int. Conf. Knowledge Discovery and Data Mining*, 125–134. ACM.

Michalewicz, Z. (1996). *Genetic Algorithms + Data structures = Evolution Programs*, 3rd Ed. Springer-Verlag.

Michalski, R. W. (1983). A Theory and Methodology of Inductive Learning. *Artificial Intelligence* **20**: 111–161.

Michie, D., Spiegelhalter, D. J. & Taylor, C. C. (1994). Conclusions. In Michie, D., Spiegelhalter, D. J. & Taylor, C. C. (eds.) *Machine Learning, Neural and Statistical Classification*, Chapter 11, 213–227. Ellis Horwood.

Nazar, K. & Bramer, M. A. (1999). Estimating Concept Difficulty with Cross Entropy. In Bramer, M. A. (ed.) *Knowledge Discovery and Data Mining*, 3–31. London: The Institution of Electrical Engineers.

Neri, F. & Giordana, A. (1995). A Parallel Genetic Algorithm for Concept Learning. *Proc. 6th Int. Conf. Genetic Algorithms*, 436–443. Morgan Kaufmann.

Newson, G. (1991). Simpson's Paradox Revisited. *The Mathematical Gazette* **75**(473): 290–293. Oct. 1991.

Pazzani, M. J. (2000). Knowledge Discovery from Data? *IEEE Intel. Systems, March/April 2000*, 10–12.

Piatetsky-Shapiro, G. (1991). Knowledge Discovery in Real Databases: A Report on the IJCAI-89 Workshop. *AI Magazine*, Vol. 11, No. 5, 68–70, Jan. 1991.

Provost, F. & Kolluri, V. (1999). A Survey of Methods for Scaling up Inductive Algorithms. *Data Mining and Knowledge Discovery* **3**(2): 131–195.

Quinlan, J. R. (1990). Learning Logical Definitions from Relations. *Machine Learning* **5**(3): 239–266.

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

Rendell, L. & Cho, H. (1990). Empirical Learning as a Function of Concept Character. *Machine Learning* **5**(3): 267–298.

Rendell, L. & Seshu, R. (1990). Learning Hard Concepts Through Constructive Induction: Framework and Rationale. *Computational Intelligence* **6**: 247–270.

Rendell, L. & Ragavan, H. (1993). Improving the Design of Induction Methods by Analyzing Algorithm Functionality and Data-based Concept Complexity. *Proc. 13th Int. Joint Conf. on Artif. Intel. (IJCAI-93)*, 952–958.

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM journal of Research and Development* **3**: 211–229. Reprinted in E. A. Feigenbaum (ed.) *Computers and Thought*. McGraw-Hill, 1963.

Schaffer, C. (1993). Overfitting Avoidance as Bias. *Machine Learning* **10**: 153–178.

Simpson, E. H. (1951). The Interpretation of Interaction in Contingency Tables. *Journal of the Royal Statistical Society, Series B* **13**: 238–241.

Srinivasan, A. & King, R. D. (1999). Feature Construction with Inductive Logic Programming: a Study of Quantitative Predictions of Biological Activity Aided by Structural Attributes. *Data Mining and Knowledge Discovery* **3**(1): 37–57.

Taha, I. A. & Ghosh, J. (1999). Symbolic Interpretation of Artificial Neural Networks. *IEEE Trans. Knowledge and Data Engineering* **11**(3):, 448–463. May/June 1999.

Ting, K. M. (1994). The Problem of Small Disjuncts: Its Remedy in Decision Trees. *Proc. 10th Canadian Conf. Artif. Intel.*, 91–97.

Vaughn, M. L. (1996). Interpretation and Knowledge Discovery from the Multilayer Perceptron Network: Opening the Black Box. *Neural Comput. & Appl.* **4**: 72–82.

Wagner, C. H. (1982). Simpson's Paradox in Real Life. *The American Statistician* **36**(1): 46–48. Feb. 1982.

Weiss, G. M. (1995). Learning with Rare Cases and Small Disjuncts. *Proc. 12th Int. Conf. Machine Learning (ML-95)*, 558–565. Morgan Kaufmann.

Weiss, G. M. (1998). The Problem with Noise and Small Disjuncts. *Proc. Int. Conf. Machine Learning (ICML-98)*, 574–578. Morgan Kaufmann.

Weiss, G. M. and Hirsh, H. (2000). A Quantitative Study of Small Disjuncts. *Proc. 17th Nat. Conf. on Artificial Intelligence (AAAI-2000)*, 665–670. AAAI Press.

Zytkow, J. (ed.) (1999). Special Session on Data Mining. In: Angeline, P. (ed.), *Proc. 1999 Congress on Evolutionary Computation (CEC-99)*, 1307–1345.