Fuzzification and Reduction of Information-Theoretic Rule Sets

Mark Last and Abraham Kandel

Department of Computer Science and Engineering

University of South Florida

4202 E. Fowler Avenue, ENB 118

Tampa, FL 33620, USA

Abstract. If-then rules are one of the most common forms of knowledge discovered by data mining methods. The number and the length of extracted rules tend to increase with the size of a database, making the rulesets less interpretable and useful. Existing methods of extracting fuzzy rules from numerical data improve the interpretability aspect, but the dimensionality of fuzzy rulesets remains high. In this paper, we present a new methodology for reducing the dimensionality of rulesets discovered in data. Our method builds upon the information-theoretic fuzzy approach to knowledge discovery. We start with constructing an information-theoretic network from a data table and extracting a set of association rules based on the network connections. The set of information-theoretic rules is fuzzified and significantly reduced by using the principles of the Computational Theory of Perception (CTP). We demonstrate the method on a real-world database from semiconductor industry.

Keywords: data mining, association rules, fuzzy rules, information-theoretic networks, computational theory of perception.

1. Introduction

As indicated by (Fayyad et al. 1996), discovery of useful and understandable patterns from data is a major goal in data mining. The basic idea of data mining is a computationally efficient search in the infinite space of patterns possibly existing in a database. Patterns and models can be represented in different forms (e.g., neural networks, mathematical equations, etc.), but if-then rules are known as one of the most expressive and human readable representations (Mitchell 1997). Srikant and Agrawal (1996) have suggested a heuristic method for explicit

enumeration of association (if-then) rules between database attributes. As opposed to the neural network structure, each association rule is easily interpreted in the natural language. However, a long list of association rules is not much more helpful to the user than the weights of a neural network. Due to the random nature of data, the list may include many meaningless interactions. In addition, the most significant associations (having highest support and confidence) are usually the most trivial and uninteresting ones.

Decision-tree algorithms, like C4.5, suggest a more focused approach to rule extraction (Quinlan 1993). The rule extraction method of C4.5 assumes that the user is interested in only one attribute (called "class") as a consequent of every rule. This assumption significantly reduces the search space vs. the problem of finding association rules between *any* database attributes (see above). A set of mutually exclusive and exhaustive if-then (production) rules can be easily extracted from a decision tree. Quinlan (1993) presents a method for generalizing (simplifying) the rules by removing one or more conditions. The method includes setting a preference ordering for conflicting rules and choosing a default rule. However, these are *decision rules* and not *association rules*: the initial rule set includes only one rule for each tree leaf, which represents the predicted class at that leaf.

The vague nature of human perception, which allows the same object to be classified into different classes with different degrees, is utilized for building *fuzzy decision trees* by Yuan and Shaw (1995). Like in C4.5, each path from root to leaf of a fuzzy decision tree is converted into a rule with a single conclusion. However, the same object may have a non-zero membership grade in more than one rule. If a unique classification is required, the class with the highest membership is selected.

There are many techniques described in the literature for extracting fuzzy rules from raw data. Wang and Mendel (1992) present an algorithm for generating fuzzy rules from numerical attributes. The total size of the fuzzy rule base is exponential in the number of input attributes (n), but, under certain assumptions, the number of active rules for a given input is bounded by 2^n. Au and Chan (1999) describe a genetic algorithm based method, called FARM, for discovering fuzzy associations between linguistic terms. The fitness function (i.e. the "goodness" of a rule) used to evaluate the set of rules is based on a probabilistic approach. Both positive (if-then) and negative (if-then-not) association rules can be discovered by FARM. Slawinski et. al. (1999) use a hybrid evolutionary approach. Here, the fitness of a rule is related to its relevance (a rule is considered relevant if the constrained probability of its conclusion exceeds the unconstrained probability). Fuzzy approach to testing a given set of rules (hypotheses) is presented in (Last and Kandel 1999) and (Last, Schenker, and Kandel 1999).

None of the fuzzy-oriented methods mentioned above attempt to interpret rules extracted by another (possibly "crisp") rule induction algorithm. In this paper, we

are applying a fuzzy approach to post-processing a given set of "crisp" association rules extracted from an information-theoretic network (Maimon, Kandel, and Last 1999). The information-theoretic method of knowledge discovery minimizes the dimensionality of extracted rules by using a built-in feature selection procedure. In addition, the algorithm discovers both positive and negative association rules and it is not limited to a single conclusion of each condition. Post-processing of the information-theoretic rules is based on the Computational Theory of Perception (Zadeh 1999) and, as demonstrated by a real-world case study, it results in a small and manageable set of compact linguistic rules.

Section 2 of this Chapter describes the information-theoretic fuzzy approach to knowledge discovery and rule extraction. The process of rule post-processing is presented in Section 3. In Section 4, we proceed with a detailed case study of rule extraction from manufacturing data. The Chapter is concluded by section 5, which reviews the potential of the fuzzy set theory for post-processing of data mining results

2. Information-Theoretic Fuzzy Approach to Knowledge Discovery

The *Info-Fuzzy Network* (IFN) methodology, initially introduced by us in (Maimon, Kandel, and Last, 1999), is a novel and unified approach to automating the process of *Knowledge Discovery in Databases* (KDD). The main stages of the IFN methodology include discretization of continuous attributes, feature selection, extraction of association rules, and data cleaning. The method is aimed at maximizing the *mutual information* (see Cover 1991) between input (predicting) and target (dependent) attributes. The following sub-sections describe the extended data model, used by IFN, the network construction algorithm, and the procedure for extracting information-theoretic association rules from the IFN structure.

2.1 Extended Relational Data Model

We use here the standard notation of the relational data model (see Korth and Silberschatz, 1991). The relational model represents the database as a collection of *relations*, or tables of values. Each table resembles, to some extent, a "flat" file of records.

1) R - a relation schema including n attributes. Each attribute represents a column of the data table. The number of attributes n is called the *degree* of a relation. In our case, $n \geq 2$ (each table is assumed to have at least two columns).

2) A_i - attribute (column) i in the data table. $R = (A_1, ..., A_n)$.

3) D_i - the domain of an attribute A_i. We assume that each domain is a set of M_i discrete values. $\forall i: M_i \geq 2$, finite. For numeric attributes having continuous domains, each value represents an interval between two continuous values. The discretization is performed in the process of network construction (see below).

4) V_{ij}- a value j of domain D_i. Consequently, $D_i = (V_{i1},..., V_{iMi})$.

5) $r(R)$ - a relation instance (table) of the relation schema R. This is a set of *n-tuples* (records). Each n-tuple is an ordered list of n values, which represent a row in the data table.

6) m - number of tuples (records) in a relation r. We assume that $m \geq 2$ (each table has at least two rows).

7) $t_k[A_i]$ - value of an attribute i in a tuple (record) k. Each value represents a cell in the data table. Each value is an element of the attribute domain or is null ($\forall k, i: \quad t_k[A_i] \in \{D_i., Null\}$). A null value may be *empty* (non-existing in the real world) or *missing* (existing in the real world, but not entered into the data table).

To find a set of association rules in a database, we make the following partition of the relation schema:

1) O - a subset of *target* (classification) attributes ($O \subset R$, $|O| \geq 1$). The values of target attributes will be the *consequents* of association rules.

2) C - a subset of *candidate input* attributes ($C \subset R$, $|C| \geq 1$). The values of candidate input attributes *can be* used as conditions in the *antecedent part* of association rules.

3) I_i - a subset of *input* attributes (features) selected by the algorithm for the target attribute i ($\forall i: I_i \subset C$). The antecedent of every rule will include at least one input attribute.

Assumptions:

- $\forall i : I_i \cap O = \varnothing$. An attribute cannot be both an input and a target. This implies that cyclic dependencies cannot be detected by the IFN method.
- $\forall i : I_i \cup O \subseteq R$. Some attributes in a database may be neither input, nor target. These may include identifying (key) attributes and candidate input attributes that were not chosen by the algorithm for the target attribute i.

2.2 Info-Fuzzy Network Structure

An Info-Fuzzy Network (IFN) has the following components:

- $|I_i|$ - total number of hidden layers in a network. Each hidden layer is uniquely associated with an input attribute by representing the interaction of that attribute and the input attributes of the previous layers. The first layer (layer 0) includes only the root node and is not associated with any input attribute. The number of conditions in the antecedent of an association rule cannot exceed the number of network layers.
- L_l - a subset of nodes in a hidden layer l. Each hidden node represents a conjunction of rule conditions.
- *At (l)* - an input attribute corresponding to the layer l in the network
- K - distinct target nodes V_{ij} for each value j in the domain of the target attribute i. Continuous target attributes are discretized to a pre-defined set of intervals. Each target node represents a consequent of association rules.
- w_z^{ij} - a connection weight between a hidden node z and a target node V_{ij}. Each node-target connection is related to a distinct association rule. As we show below, the calculation of the rule weights is based on the information theory.

The network structure, described above, differs from the structure of a standard decision tree (see Quinlan, 1986 and 1993) in two aspects. First, it is restricted to the same input attribute at all nodes of each hidden layer. Second, its node-target connections represent *association rules* between input and target attributes unlike the standard decision trees, which are used to extract *prediction rules* only (e.g., see Quinlan 1993).

2.3 Network Construction Procedure

Without loss of generality, we present here a search procedure for constructing a multi-layered network of a *single* target attribute A_i. In a general case, the network should be re-built (starting with Step 4 below) for every target attribute defined in a database.

Step1 - Given a relation schema and available domain knowledge, partition the schema into a subset of *candidate input* and a subset of *target* attributes (see the extended relational model above).

Step 2 - Enter a minimum significance level α for splitting a network node (default: $\alpha = 0.001$). High significance levels cause the random ("noisy") rules to be excluded from the network.

Step 3 - Read tuples (records) of a relation. Tuples with non-valid or missing target values are ignored by the algorithm. Missing (null) values of candidate input attributes are ignored too, but without ignoring the other, non-empty attributes in the same tuple. The domain of every attribute may be restricted by the user to a set of pre-defined values or learned by the algorithm from the data itself.

Step 4 - Estimate unconditional (*a priori*) probability of each value of the target attribute by: $P(V_{ij}) = O_{ij}/n$,

where

O_{ij} - number of occurrences of the value j of a target attribute i in the relation

n - number of complete tuples in the relation

Step 5 - Calculate the estimated unconditional entropy of the target attribute (see Cover, 1991) by:

$$H(A_i) = -\sum_{j=1}^{M_i} P(V_{ij}) \cdot \log P(V_{ij}) \quad (1)$$

Where

M_i - domain size of an attribute i (number of distinct values taken by the attribute)

The entropy is a *metric-free* measure of uncertainty. It reaches its highest value ($logM_i$), when the probability of all values is distributed uniformly. If an attribute takes a single value with the probability of 1.0, its entropy is equal to zero. The formula (1) above calculates *unconditional* entropy, since it is not based on the knowledge of values of any other attribute.

Step 6 - Initialize the info-fuzzy network (single root node associated with all tuples, no input attributes, and a target layer for values of the target attribute). An example of the initial network structure for a three-valued target attribute is shown in Figure 1.

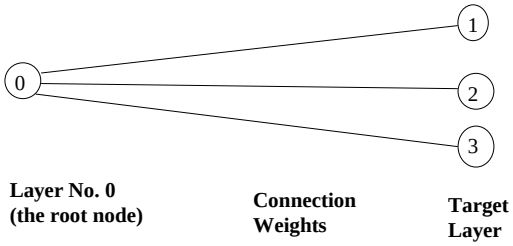

Figure 1 Info-Fuzzy Network: Initial Structure

Step 7 - While the maximum number of layers (equal to the number of candidate input attributes) is not exceeded, Do:

Step 8- Repeat for every candidate input attribute $A_{i'}$, *which is not in the network*:

Step 8.1 – Initialize to zero the degrees of freedom and the estimated conditional mutual information of the candidate input attribute and the target attribute, given the final layer of hidden nodes. Conditional mutual information is defined as a decrease in the *conditional entropy*, which represents uncertainty of a random attribute, given values of other attributes. According to (Cover, 1991), the information on more attributes can never increase the entropy. Thus, conditional mutual information is a non-negative variable. As shown below, conditional mutual information can be estimated by using the frequency estimators of conditional and unconditional probabilities of the target attribute values.

Step 8.2 – If $A_{i'}$ is a continuous attribute, then Do:

Step 8.2.1 - Define the boundaries of the interval *S*, to be partitioned, as the first and the last distinct values of $A_{i'}$.

Step 8.2.2 – Repeat for every distinct value included in the interval *S* (except for the last distinct value):

Step 8.2.2.1 – Define the distinct value as a partitioning threshold (*T*). All distinct values below or equal to *T* belong to the first sub-interval S_1 (sub-interval *1*). Distinct values above *T* belong to the second sub-interval S_2 (sub-interval *2*).

*Step 8.2.2.*2 – Repeat for every node *z* of the final hidden layer:

Step 8.2.2.2.1 – Calculate the estimated conditional mutual information between the partition of the interval *S* at the threshold *T* and the target attribute A_i, given the node *z*, by the following formula (based on Cover (1991)):

$$MI\ (T; A_i / S, z) = \sum_{j=0}^{M_{i'}-1} \sum_{y=1}^{2} P(S_y; V_{ij}; z) \cdot \log \frac{P(S_y; V_{ij} / S, z)}{P(S_y / S, z) \bullet P(V_{ij} / S, z)} \quad (2)$$

where

$P(S_y / S, z)$ - an estimated conditional (*a posteriori*) probability of a sub-interval S_y, given the interval *S* and the node *z*

$P(V_{ij} / S, z)$ - an estimated conditional (*a posteriori*) probability of a value *j* of the target attribute *i* given the interval *S* and the node *z*.

$P(S_y ; V_{ij} / S, z)$ - an estimated joint probability of a value j of the target attribute *i* and a sub-interval S_y, given the interval *S* and the node *z*.

$P(S_y; V_{ij;} z)$ - an estimated joint probability of a value j of the target attribute *i*, a sub-interval S_y, and the node *z*.

Step 8.2.2.2.2 - Calculate the likelihood-ratio test for the partition of the interval *S* at the threshold *T* and the target attribute A_i, given the node *z*, by the following formula (based on Rao and Toutenburg, 1995):

$$G^2(T; A_i / S, z) = 2\sum_{j=0}^{M_i-1}\sum_{y=1}^{2} N_{ij}(S_y, z) \cdot \ln \frac{N_{ij}(S_y, z)}{P(V_{ij} / S, z) \bullet E(S_y, z)} \qquad (3)$$

where

$N_{ij}(S_y, z)$ - number of occurrences of a value *j* of the target attribute *i* in sub-interval S_y and the node *z*.

$E(S_y, z)$ - number of tuples in sub-interval S_y and the node *z*

$P(V_{ij}/ S, z)$ - an estimated conditional (*a posteriori*) probability of a value *j* of the target attribute *i*, given the interval *S* and the node *z*.

$P(V_{ij}/ S, z) \bullet E(S_y, z)$ - an estimated number of occurrences of a value *j* of the target attribute *i* in sub-interval S_y and the node *z*, under the assumption that the conditional probabilities of the target attribute values are identically distributed, given each sub-interval.

Step 8.2.2.2.3- Calculate the degrees of freedom of the likelihood-ratio statistic by:

$$DF(T, A_i / S, z) = (NI_{i'}(S, z) - 1) \cdot (NT_i(S, z) - 1) =$$

$$(2-1) \cdot (NT_i(S, z) - 1) = NT_i(S, z) - 1 \qquad (4)$$

Where

$NI_{i'}(S, z)$ - number of sub-intervals of a candidate input attribute i' at node *z* (2)

$NT_i(S, z)$ - number of values of a target attribute i in the interval S at node *z*.

Step 8.2.2.2.4 - If the likelihood-ratio statistic is significant at the level defined in *Step 2* above, mark the node as "split" by the threshold *T* and increment the estimated conditional mutual information of the candidate input attribute and the target attribute, given the threshold *T*; else mark the node as "unsplit" by the threshold *T*.

Step 8.2.2.2.5 - Go to next node.

Step 8.2.2.3 – Go to next distinct value.

Step 8.2.3 – Find the threshold T_{max} maximizing the estimated conditional mutual information between a partition of the candidate input attribute $A_{i'}$ and the target attribute A_i, given the interval S and the set of input attributes I by:

$$T_{\max} = \arg\max_{T} MI(T; A_i / I_i, S) \tag{5}$$

and increment the estimated conditional mutual information between the candidate input attribute $A_{i'}$ and the target attribute A_i by the value calculated in the formula (2) above.

Step 8.2.4 – If the maximum estimated conditional mutual information is greater than zero, then do:

Step 8.2.4.1 - Repeat for every node *z* of the final hidden layer:

Step 8.2.4.1.1 – If the node *z* is splitted by the threshold T_{max}, mark the node as splitted by the candidate input attribute $A_{i'}$

Step 8.2.4.2 - Partition each sub-interval of *S (*go to step *8.2.2)*. If the threshold T_{max} is the first distinct value in the interval *S*, T_{max} is marked as a new encoding interval and only the second sub-interval is partitioned.

Step 8.2.4.3 - EndDo

Else (if the maximum estimated conditional mutual information is equal to zero) Do:

Step 8.2.5 - Create a new encoding interval *S* and increment the domain size of $A_{i'}$ (number of encoding intervals).

Step 8.2.6 - EndIf

Step 8.2.7 – EndDo

Step 8.3 – Else (if the attribute $A_{i'}$ is discrete), Do

Step 8.3.1 - Repeat for every node z of the final hidden layer:

Step 8.3.1.1 - Calculate the estimated conditional mutual information of the candidate input attribute i and the target attribute *i*, given the node *z*, by

$$MI(A_{i'}; A_i / z) = \sum_{j=0}^{M_i-1} \sum_{j'=0}^{M_{i'}-1} P(V_{ij}; V_{i'j'}; z) \cdot \log \frac{P(V_{i'j'}^{ij} / z)}{P(V_{i'j'} / z) \cdot P(V_{ij} / z)} \tag{6}$$

Where

$P(V_{i'j'}/ z)$ - an estimated conditional (*a posteriori*) probability of a value j' of the candidate input attribute *i'*, given the node *z*.

P (V_{ij}/ *z)* - an estimated conditional (*a posteriori*) probability of a value j of the target attribute *i*, given the node *z*.

P ($V_{i'j'}^{ij}$/ *z)* - an estimated conditional (*a posteriori*) probability of a value j' of the candidate input attribute *i'* and a value *j* of the target attribute *i*, given the node *z*.

P (V_{ij}; $V_{i'j'}$; *z)* - an estimated joint probability of a value *j* of the target attribute *i*, a value *j'* of the candidate input attribute *i'* and the node *z*.

Step 8.3.1.2 - Calculate the statistical significance of the estimated conditional mutual information, by using the likelihood-ratio statistic (also based on Rao and Toutenburg, 1995):

$$G^2(A_{i'}; A_i / z) = 2\sum_{j=0}^{M_i-1}\sum_{j'=0}^{M_{i'}-1} C_{i'j'}^{ij}(z)\cdot \ln \frac{C_{i'j'}^{ij}(z)}{P(V_{ij}/z)\bullet E_{i'j'}(z)} \quad (7)$$

Where

$C_{i'j'}^{ij}$ *(z)* - number of joint occurrences of value *j* of the target attribute *i* and value *j'* of the candidate input attribute *i'* in the node *z*.

$E_{i'j'}$*(z)* - number of occurrences of value *j'* of the candidate input attribute *i'* at the node *z*.

P (V_{ij}/ *z)*$\bullet$ $E_{i'j'}$*(z)* - an estimated number of joint occurrences of value *j* of the target attribute *i* and value *j'* of the candidate input attribute *i'* under the assumption that the attributes *i'* and *i* are conditionally independent, given the node *z*.

Step 8.3.1.3 - Calculate the degrees of freedom of the likelihood-ratio statistic by:

$$DF(A_{i'}; A_i / z) = (NI_{i'}(z) - 1)\cdot(NT_i(z) - 1) \quad (8)$$

where

$NI_{i'}$ *(z)* - number of values of a candidate input attribute *i'* at node *z*.

NT_i *(z)* - number of values of a target attribute *i* at node *z*.

Step 8.3.1.4 - If the likelihood-ratio statistic is significant, mark the node as "split" and increment the conditional mutual information of the candidate input attribute and the target attribute, given the final hidden layer of nodes (*MI* ($A_{i'}$; A_i / I_i)) by the value calculated in the formula (6) above; else mark the node as "terminal".

Step 8.3.1.5 - Go to next node.

Step 8.3.2 - Go to next candidate input attribute.

Step 8.3.3 - EndDo

Step 8.3.4 - EndIf

Step 9 - Find a candidate input attribute maximizing the estimated conditional mutual information ("the best candidate attribute").

Step 10 - If the maximum conditional mutual information is zero**,** go to Step 13. Otherwise, go to the next step.

Step 11 - Add a new hidden layer to the network: make the best candidate attribute a new input attribute and define a new layer of nodes for a Cartesian product of splitted hidden nodes in the previous layer and the values of the best candidate attribute. A new hidden node is defined if the relation (data table) has at least one tuple associated with it.

Step 12 - EndDo

Step 13 - **Stop** the network construction

Step 14 - Output the network structure which includes the names of the attributes associated with each hidden layer, the ID numbers of hidden nodes related to every value of an input attribute, and the connections between the terminal hidden nodes and the target nodes.

In Figure 2, a structure of a two-layered network (based on two selected input attributes) is shown. The first input attribute has three values, represented by nodes 1,2, and 3 in the first layer, but only nodes 1 and 3 are split due to the statistical significance testing in Step *8.3* above. The second layer has four nodes standing for the combinations of two values of the second input attribute with two splitted nodes of the first layer. Like in Figure 1, the target attribute has three values, represented by three nodes in the target layer. The network in Figure 2 has five terminal (unsplit) nodes: 2; 1, 1; 1, 2; 3, 1; and 3, 2. The total number of input-target connections in this network is 5*3 = 15.

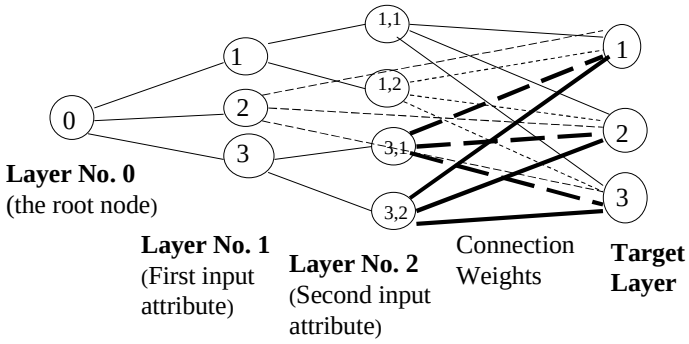

Figure 2 Info-Fuzzy Network: Two-Layered Structure

2.4 Rule Extraction

Each terminal node in an info-fuzzy network represents a conjunction of input attribute values. Thus, a connection between a terminal node and a node of the target layer may be interpreted as a rule of the form *if conjunction of input values, then the value of the target attribute is likely / unlikely to be…* An information-theoretic *weight* is associated with every input-target connection. The general algorithm for extracting association rules from the network connections and evaluating their information-theoretic weights is given below.

Step 1 – Initialize the number of rules *r* to zero

Step 2 - Repeat for every terminal node *z*:

Step 2.1 – Repeat for every value *j* of the target attribute A_t:

Step 2.1.1 – Initialize the hidden layer index *l* to zero

Step 2.1.2 – While *l* < {number of layers associated with the node *z*} Do:

Step 2.1.2.1 - Add new condition to the antecedent part of the rule *r*, based on the value of the input attribute corresponding to the layer *l*

Step 2.1.2.2 – Increment *l*

Step 2.1.2.3 - EndDo

Step 2.1.3 - Make value *j* the consequent of the rule *r*

Step 2.1.4 – Calculate the connection weight w_z^j associated with the rule *r* by:

$$w_z^j = P(V_j; z) \cdot \log \frac{P(V_j / z)}{P(V_j)} \qquad (9)$$

Where

P (V_j; *z*) - an estimated joint probability of the value V_j and the node *z*

P (V_j/ *z*) - an estimated conditional (*a posteriori*) probability of the value V_j, given the node *z*

P (V_j) - an estimated unconditional (*a priori*) probability of the value V_j

Step 2.1.5 – Increment the number of rules *r* by one

Step 2.1.6 – Go to next target value *j*

Step 2.2 - Go to next terminal node *z*

Step 3 - End

Each connection weight represents a contribution of a node-pair to the total mutual information between the input attributes and the target attribute. The weight will be positive if the conditional probability of a target attribute value, given the node, is higher than its unconditional probability and negative otherwise. A zero weight means that the target attribute value is independent of the node value. Thus, each positive connection weight can be interpreted as an *information content* of an appropriate rule of the form *if node, then target value.* Accordingly, a negative weight refers to a rule of the form *if node, then* ***not*** *target value*. Connections with zero weights can be ignored, since they do not change the conditional probability of the target attribute.

The most informative rules can be found from sorting the rules by their information-theoretic connection weights. Both the rules having the highest positive and the lowest negative weights are of a potential interest to a user. As shown in the proposition below, the sum of connection weights is equal to the estimated mutual information between a set of input attributes and a target attribute. According to the well-known Pareto rule, a small number of informative rules are expected to explain a major part of the total mutual information.

Proposition. The sum of connection weights at all unsplitted and final layer nodes is equal to the estimated mutual information between a set of input attributes and a target attribute:

$$MI(A_i; I_i) = \sum_{z \in F} \sum_{j=0}^{M_i - 1} P(V_{ij}; z) \cdot \log \frac{P(V_{ij} / z)}{P(V_{ij})} \qquad (10)$$

Where

A_i – target attribute i

I_i - set of input attributes

z – hidden node in the information-theoretic network

F - subset of terminal (unsplit) nodes

$P(V_{ij}; z)$ - an estimated joint probability of the target value V_{ij} and the node z

$P(V_{ij}/ z)$ - an estimated conditional (a posteriori) probability the target value V_{ij}, given the node z

$P(V_{ij})$ - an estimated unconditional (a priori) probability of the target value V_{ij}.

Proof. This proposition is directly derived from the definition of mutual information between random variables X and Y (Cover, 1991):

$$MI(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \cdot \log \frac{p(y/x)}{p(y)} \qquad (11)$$

In the above expression, we have replaced Y with the target attribute A_i and X with the set of input attributes I_i. A node $z \in F$ represents a conjunction of input attribute values. Since the information-theoretic network represents a disjunction of these conjunctions, each conjunction is associated with one and only one node $z \in F$. Consequently, the summation over all unsplitted and final nodes covers all possible values of the input attributes. This completes the proof.

2.5 Computational Complexity of the Algorithm

The computational complexity of the network construction for a single target attribute is calculated by using the following notation:

m - total number of records in a training data set

$|C|$ - total number of candidate input attributes

p - portion of candidate input attributes, selected as inputs by the network construction procedure, $0 \le p \le 1$

$|I|$ - number of hidden layers (input attributes), $|I| \le |C|$

M_C - maximum domain size of a candidate input attribute

M_T - domain size of the target attribute

The computational "bottleneck" of the algorithm is calculating the estimated conditional mutual information between every binary partition of a continuous candidate-input attribute and a target attribute, given a hidden node (*MI (T; A_i / S, z)*). Since each node of l-th hidden layer represents a conjunction of values of *l* input attributes, the total number of nodes at a layer *l* is apparently bounded by $(M_c)^l$. However, we restrict defining a new node by the requirement that there is at least one record associated with it (see Step 11 in sub-section 2.3 above). Thus, the total number of nodes at any hidden layer cannot exceed the total number of records (*m*). In most cases, the number of nodes will be much smaller than *m*, due to records having identical values of input attributes and the statistical significance requirement of the likelihood-ratio test when splitting a hidden node.

The calculation of the conditional mutual information is performed at each hidden layer of the information-theoretic network for all candidate input attributes at that layer. The number of possible partitions of a continuous attribute is bounded by $m\log_2 m$ (Fayyad and Irani, 1993). For every possible partition, the term *MI (T; A_i / S, z)* is summed over all nodes of the final layer. This implies that the total number of calculations is bounded by:

$$m \cdot m \cdot \log_2 m \cdot M_T \cdot \sum_{l=0}^{p|C|} (|C| - l) \leq$$

$$\frac{m^2 \cdot \log_2 m \cdot M_T \cdot |C^2| \cdot p \cdot (2-p)}{2} \qquad (12)$$

The actual number of calculations will usually be much smaller than this bound, since the number of tested partitions may be less than the number of distinct values (resulting from the likelihood-ratio test). The number of distinct values, in turn, may be much lower than the total number of records (*m)* and some candidate input attributes may not require discretization due to their discrete nature (e.g., nominal attributes). Thus, the run time of the search procedure is quadratic-logarithmic in the number of records and quadratic polynomial in the number of initial candidate input attributes. Moreover, it is reduced by the factor of *p(2-p)*.

3. Post-processing of Association Rules

The number of rules extracted from an information-theoretic network may be quite large. It is bounded by the product of the number of terminal nodes and the

number of target nodes (see the algorithm in 2.4 above), and the previous applications of the algorithm show that this bound is sharp. Although the rules are important for the predictive accuracy of the network, the user may find it difficult to comprehend the entire set of rules and to interpret it in natural and actionable language. As we show in this section, the *fuzzification* of the information-theoretic rules provides an efficient way for reducing the dimensionality of the rule set, without losing its actionable meaning. The process of rule reduction includes the following stages:

Stage 1 - Fuzzifying crisp rules

Stage 2 – Reducing the set of fuzzified rules by conflict resolution

Stage 3 – Merging rules from the reduced set

Stage 4 - Pruning the merged rules

3.1 Fuzzifying Association Rules

Although, the boundaries of the discretized intervals are determined by the algorithm of sub-section 2.3 above to minimize the uncertainty of the target attribute, the user may be more interested in the linguistic descriptions of these intervals, rather in their precise numeric boundaries. Thus, we start with expressing "linguistic ranges" of continuous attributes as lists of terms that the attributes can take (“high”, “low”, etc.). Then we define membership functions representing the user perception of each term. According to (Zadeh 1999), this is the first stage in an automated reasoning process, based on the Computational Theory of Perception (CTP), which can directly operate on perception-based, rather than measurement-based, information. Subsequent CTP stages include constructing the initial constraint set (ICS), goal-directed propagation of constraints, and creating a terminal constraint set, which is the end result of the reasoning process.

As indicated by (Shenoi 1993), fuzzification of numeric attributes in a real-world database may be used for an additional purpose: *information clouding*. The user may be unwilling to disclose the actual values of some critical performance indicators associated with marketing, sales, quality, and other areas of business activity. In many cases, data security considerations prevent results of successful data mining projects from being ever published. The application part of this chapter also deals with highly sensible data obtained from a semiconductor company. Direct presentation of rules extracted from this data could provide valuable information to the company competitors. However, we are going to “hide” the confidential context of the rules by presenting them in their fuzzified form only.

The terms assigned to each simple condition and to the target (consequence) of the association rule are chosen to maximize the membership function at the middle point of the condition / consequence interval. Thus, we convert a crisp rule into a *fuzzy relation* (Wang 1997). Since a complex condition is a conjunction of simple conditions, an algebraic product is used to find the fuzzy intersection of the simple conditions. Fuzzy implication of Mamdani type (see below) is applied to each rule. Mamdani implication is more appropriate for the fuzzification of the information-theoretic rules due to the local nature of these rules. The informativeness of each fuzzified rule is represented by weighting the implication by the information-theoretic weight of the corresponding crisp rule (see sub-section 2.4 above). If the weight is positive, the rule is stated as "*If <conjunction of terms assigned to rule conditions>, then <term assigned to the rule target >*". If the weight is negative, the rule will be of the form "*If <conjunction of terms assigned to rule conditions>, then* ***not*** *<term assigned to the rule target >*". The expression for calculating the weighted membership grade of an association rule is given below.

$$\mu_R = w \bullet [\prod_{i=1}^{N} \max_j \{\mu_{A_{ij}}(V_i)\}] \bullet \max_k \{\mu_{T_k}(O)\} \qquad (13)$$

Where

w – information-theoretic weight of the crisp rule

N – number of simple conditions in the crisp rule

V_i – crisp value of the simple condition i in the crisp rule (middle point of the condition interval)

O – crisp value of the rule target (middle point of the target interval)

$\mu_{A_{ij}}(V_i)$ - membership function of the simple condition i w.r.t. term j

$\mu_{T_k}(O)$ - membership function of the target value O w.r.t. term k

3.2 Removing Inconsistent Rules

An information-theoretic ruleset represents association rules between conjunctions of input values and all possible target values. Hence, several rules may have the same IF parts, but different THEN parts. Fuzzification may even increase the number of distinct rules with identical antecedents, since several adjacent intervals may refer to the same linguistic term. This means that the set of fuzzy rules, produced in sub-section 3.1 above, may be *inconsistent*. To resolve the conflicts, we calculate the grade of each distinct fuzzy rule and choose the target value from a conflict group that has a maximum grade. A similar approach is used by (Wang and Mendel, 1992) for resolving conflicts in fuzzy rules generated from data. The reduced set of distinct fuzzy rules is constructed by the following procedure:

Algorithm RESOLVE_CONFLICTS (Set_of_Fuzzified_Rules)

- *Initialize total number of distinct fuzzy rules to zero.*
- *Repeat for every fuzzified rule:*
  - *Find a distinct fuzzy rule with identical linguistic values of input attributes*
  - *If Rule Found,*
    - *Find identical linguistic value of the target attribute in the distinct rule*
    - *If Value Found,*
      - *Increment the grade of the target linguistic value by the grade of the fuzzified rule*
    - *Else (if value not found),*
      - *Update the set of target linguistic values in the distinct rule*
      - *Initialize the grade of the new target value to the grade of the fuzzified rule*
  - *Else (if rule not found)*
    - *Increase number of distinct fuzzy rules*
    - *Update the linguistic values of input attributes in the new rule*
    - *Update the first target linguistic value in the new rule*
    - *Initialize the grade of the first target value in the new distinct rule to the grade of the fuzzified rule*
  - *Next fuzzified rule*
- *For each distinct fuzzy rule do*
  - *Find the target linguistic value providing the maximum membership grade for the rule*
  - *Make it the single target value of the rule*

In the above procedure, there is no explicit distinction between positive and negative rule grades. For example, the fuzzified rules of the form *If A then B* and *If A then not B* are associated with the same target value in the same distinct rule. However, their combined grade will be equal to the *difference* of their absolute grades. Eventually the target value with the maximum *positive* grade will be chosen by the above procedure. This closely agrees with the interests of most users, who need to estimate *positively* the expected outcome of each condition.

The computational complexity of the RESOLVE_CONFLICTS algorithm is proportional to the square of the number of fuzzified rules times the average number of rule conditions. This is because the algorithm compares the antecedent conditions of every *fuzzified* rule to the corresponding conditions of every *distinct fuzzy* rule. If the two antecedents are found identical, the grade of the corresponding target linguistic value is updated. If no matching rule is found, a new distinct fuzzy rule is created. Thus, the number of distinct fuzzy rules is bounded by the number of fuzzified rules.

3.3 Merging Reduced Rules

In the previous sub-section, we have shown a method for handling rules having *identical antecedents* and *distinct consequents*. However, the resulting set of conflict-free rules may be reduced by merging the rules having *distinct antecedents* and *identical consequents*. Thus, any two rules (I) and (II) having the form:

I.If a is A and b is B and c is C, then t is T

II.If d is D and e is E and f is F, then t is T

can be merged into a single rule (III) of the following *disjunctive* form:

III. *If a is A and b is B and c is C* ***or*** *d is D and e is E and f is F, then t is T*

Using the above approach, we can create a rule base of a minimal size, limited by the number of target values. However, this approach may produce a small number of long and hardly useable rules (like the rule III above). Therefore, we perform the merging of disjunctive values *for the last rule condition only*. The procedure of merging fuzzy conjunctive rules is given below. It is based on the assumption that each fuzzy rule is using the same partial sequence of input attributes, which is true for any rule base extracted from an information-theoretic network (see sub-section 2.3 above).

Algorithm MERGE_RULES (Consistent_Set_of_Fuzzy_Rules)

- *Initialize total number of merged fuzzy rules to zero.*
- *Initialize number of conditions (l) to zero*
- *While (l < total number of input attributes) do*
  - *Repeat for every fuzzy rule having l conditions*
  - *If there are no merged rules having l conditions*
    - *Define the first merged rule with l conditions*
    - *Initialize the rule grade*

- *Else*
  - *Try to merge with an existing rule (having the same target value and the same input values for (l-1) conditions)*
- *If merged,*
  - *Update disjunctive condition no. l by a new term*
  - *Update the rule grade by using a fuzzy union ("max" operation).*
- *Else*
  - *Define a new merged rule with l conditions*
  - *Initialize the rule grade*
- *Increment l*

The computational complexity of the MERGE_RULES algorithm is proportional to the square of the number of distinct fuzzy rules (bounded by the number of information-theoretic rules) times the average number of rule conditions (minus the last condition). This is because the algorithm performs pairwise comparison of all conditions, except for the last one. If the partial antecedents of two rules are found identical and their target values are identical too, the rules are merged. In the end of the process, only the rules, which do not match any other rule, are left unmerged.

3.4 Pruning Merged Rules

The rules merged by the algorithm of sub-section 3.3 above may include several values in the last (disjunctive) condition. The number of values is bounded by the number of fuzzy terms in the attribute corresponding to the last condition. However, if the number of values in a disjunctive condition is *equal* to the number of attribute terms, the condition can be eliminated, since a complete linguistic domain of an attribute represents the entire universe of discourse. In other words, we can *prune* the rule by removing the last condition. The formal algorithm for pruning merged rules is given below.

Algorithm PRUNE_RULES (Set_of_Merged_Rules)

- *Repeat for each layer l in the Info-Fuzzy Network*
  - *Repeat for each merged rule r having l conditions*
    - *If the number of values in the last condition (condition l) is equal to the domain size of the attribute At (l) corresponding to the layer l in the network:*
      - *Decrement the number of conditions in rule r by one*

- *Remove rule r from the set of rules having l conditions*
- *Add rule r to the set of rules having (l-1) conditions*
- *End If*

- *Next rule r at the layer l*

- *Next layer l*

The computational complexity of the PRUNE_RULES algorithm is proportional to the number of merged rules. The number of merged rules is bounded by the number of information-theoretic rules (see previous sub-sections).

4. Case Study

In this section, we are applying the process of rule extraction, fuzzification, and reduction to a real-world data set provided by a semiconductor company. The semiconductor industry is a highly competitive sector, and the data included in our analysis is considered highly sensitive proprietary information. Consequently, we are forced to omit or change many details in the description of the target data and the obtained results. As indicated in sub-section 3.1 above, fuzzification of continuous attributes has helped us to "hide" the proprietary information from the unauthorized (and, probably, curious) reader.

4.1 The Problem Domain

The Information-Fuzzy Network (IFN) methodology is applied to a real-world database containing typical data from a semiconductor plant. The basic measure of profitability in semiconductor industry is the outgoing *yield* of manufactured batches. Overall, or line yield of a manufacturing process is defined as the ratio between the number of good parts (chips) in a completed batch and the initial number of chips in the same batch. Since capitalization costs constitute the major part of manufacturing costs in semiconductor industry, the *cost* of producing a single batch is almost fixed. However, the *income* from a given batch is equal to the price of one chip times the number of good chips. Thus, there is a direct relationship between the yield and the profits of semiconductor companies, who treat their yield data as "top secret" information.

Controlling and preserving the yield is a complex engineering problem. Both new and mature semiconductor products suffer from variability of yield within and between individual batches and even on specific wafers of the same batch. Improved understanding of this variability can save significant manufacturing

costs by focusing on problematic processes and taking appropriate actions, whenever excursion of yield is expected for a given batch, wafer, etc.

Although the amount of manufacturing data collected by semiconductor companies is constantly increasing, it is still hard to identify the most important parameters for yield modeling and prediction. In this study, we are trying to find relationships between the batch yield and two types of available data:

- *Batch-based data* including information about product and process type, batch priority, etc. Different processes are expected to have different yields, depending on their maturity, tool condition, and other factors.
- *WIP (Work-in-Process) data* showing the batch routing (sequence of fabrication steps), the date of completing each fabrication step, quantity transferred to the next step, and other parameters. Multiple records (based on different fabrication steps) may be related to the same batch. The batch yield may depend on the *flow time*, which is the amount of time spent at the same fabrication step. Certain materials used in semiconductor industry are known to be sensitive to the time difference between succeeding operations.

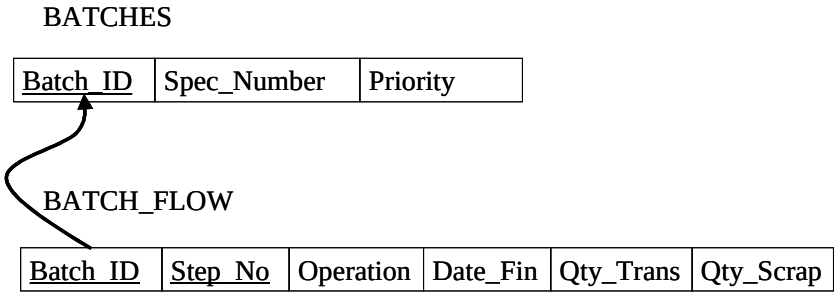

Figure 3 Relational Schema of the Semiconductor Database

The relational schema of the database provided to us by the company is shown in Figure 3 above. Here is a short explanation about each attribute in the schema:

- Table BATCHES
  - *Batch_ID*. This is the identification number of each batch and the primary key of the table.
  - *Spec_Number*. This is a specification (part) number of a batch. It specifies the manufacturing parameters of the batch, like voltage, frequency, chip size, etc.
  - *Priority*. This is the priority rank of a batch, usually assigned by the marketing department.

- Table BATCH_FLOW
  - *Batch_ID*. This is the identification number of a batch. It is a foreign key, since it is related to the primary key of the table BATCHES, but it is also a part of the primary key of this table.
  - *Step_No*. This is the serial number of a fabrication step in the manufacturing process of a given batch. A completed batch has several steps. The attribute *Step_No* is a part of the primary key. Each record in the BATCH_FLOW table is uniquely identified by a combination of values of two attributes: *Batch_ID* and *Step_No*.
  - *Operation*. The code of the operation applied to the batch no. *Batch_ID* at the fabrication step no. *Step_No*.
  - *Date_Fin*. The date when the fabrication step was completed. After completion of a step, the batch is transferred automatically to the next step on its routing list.
  - *Qty_Trans*. The quantity of good chips transferred to the next step. If a batch consists of wafers, the number of good chips is calculated automatically from the number of wafers.
  - *Qty_Scrap*. This is the number of chips scrapped at the current fabrication step. It is equal to the difference between the number of chips transferred from the previous step and the number of chips transferred to the next step. If entire wafers are scrapped, the number of scrapped chips is calculated automatically.

4.2 Data Preparation

4.2.1 Data Selection

In the original dataset provided by the company, the table BATCHES included 3,129 records. Since the company is manufacturing a variety of semiconductor products, the batches represented by the table records had different electric characteristics and different routings. Consequently, we have decided to focus our analysis on a group of 816 batches related to a single product family. The products of this family have two main parameters (chip size and electric current) and their manufacturing process includes about 30 fabrication steps.

4.2.2 Feature Extraction

The extended relational data model (see sub-section 2.1 above) assumes that the values of all candidate input and target attributes are given in the same record of a relational table. However, the table BATCHES does not include some candidate input attributes (product parameters and flow times between succeeding steps), as well as the target attribute (yield). The product parameters (size and current) were extracted from the attribute *Spec_Number* by using metadata on the attribute's encoding schema. The flow times at each fabrication step (except for the first one) were calculated by taking the difference between the completion dates of the current step and the previous step. The completion dates were given by the attribute *Date_Fin* in the table BATCH_FLOW. The line yield of each batch was found from dividing the value of the attribute *Qty_Trans* in the last fabrication step by its value in the first step. These feature extraction operations have resulted in a new schema of the table BATCHES, which is shown in Figure 4 below.

BATCHES

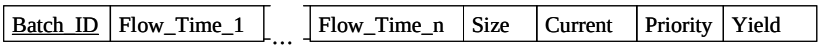

Figure 4 New Relational Schema of the BATCHES Table

4.2.3 Discretization of Target Attribute

The structure of the Info-Fuzzy Network introduced in sub-section 2.2 above requires the target attribute to be a discrete variable. However, yield is a continuous attribute: it can take any value between zero and one. Thus, we have discretized the attribute *yield* to 10 intervals of approximately equal frequency. The resulting entropy of *yield* was 3.32 (very close to $log_2 10$).

4.3 Extraction of Information-Theoretic Rules

The Info-Fuzzy Network extracted from the BATCHES table is shown in Figure 5 below. The network includes three hidden layers related to three input attributes selected by the algorithm of sub-section 2.3 above: *Size*, *Current*, and *Flow_Time_29* (flow time at operation 29). *Size* (chip size) was defined as a nominal attribute, since the given product is manufactured in three different sizes only (represented by the three nodes of the first hidden layer). *Current* is a continuous attribute, which was discretized by the algorithm into four intervals resulting in four nodes of the second layer. Another continuous attribute, *Flow_Time_29*, was discretized into two intervals. Hence, the third hidden layer has two nodes. The network has seven terminal (unsplitted) nodes: 1, 2, 5, 6, 7, 8,

and 9. Full connections between the terminal and the target nodes are not shown in Figure 5 due to space limitations.

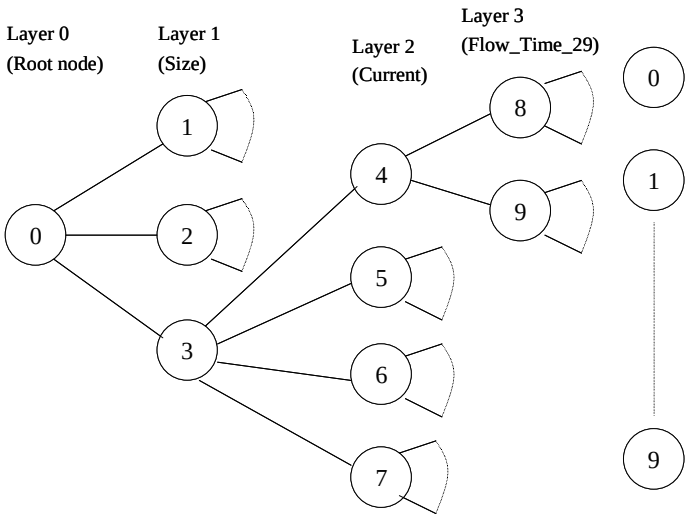

Figure 5 Info-Fuzzy Network (BATCHES Table)

The relative importance of each selected attribute is shown in Table 1 below. The column "Mutual Information" shows the cumulative association between a subset of input attributes, selected up to a given iteration inclusively, and the target attribute. Since the mutual information is defined as the difference between unconditional and conditional entropy (Cover 1991), it is bounded by the unconditional entropy of *yield*, which is 3.32. The estimated net increase in the mutual information, due to adding each input attribute, is presented in the column "Conditional MI". The last column "Conditional Entropy" is the difference between the unconditional entropy (3.32) and the estimated mutual information.

Table 1 Selected Attributes (BATCHES Table)

Iteration	Attribute Name	Mutual Information	Conditional MI	Conditional Entropy
0	Size	0.102	0.102	3.218
1	Current	0.204	0.102	3.116
2	Flow_Time_29	0.255	0.051	3.065

The network of Figure 5 above can have up to 7*10=70 connections between its seven terminal nodes and ten nodes of the target layer. The number of connections having non-zero information-theoretic weights is 58. Each connection represents an association rule of the form

If Size = V_1 and Current = V_2, and Flow_Time_29 = V_3 then Yield is [not] V_4

where V_1, V_2, and V_3 either represent valid values from the domains of the corresponding attributes, or are equal to "don't care". The consequent V_4 represents one of discretization intervals of the target attribute (*Yield*). The rules having the highest positive and the smallest negative connection weights are given below (confidential information was replaced by meaningless letters).

- **Rule No. 28**: If Size is *Z* and Current is between *C* and *D* then Yield is between *A* and *B* (weight = 0.0737).
- **Rule No. 6**: If Size is *Y* then Yield is not between *E* and *F* (weight = -0.0233).

Though the above rules are expressed in accurate, "crisp" terms defining the exact boundaries of each underlying interval, their representation power is quite limited for the following reasons:

1) The user is more interested in the rules of the form "If current is high, then the yield is low", which is closer to the human way of reasoning. People tend to "compute with words" rather than with precise numbers.

2) The total number of rules, extracted from this dataset, is 58, which is larger than the number of rules generally used by people in their decisions.

3) The rules cannot be presented to outsiders (e.g., representatives of a rival company) without revealing some sensitive information. This may be an obstacle to open exchange of technological information in forums like professional conferences, multi-company consortia, etc.

In the next sub-section, we are going to use the Computational Theory of Perception (Zadeh 1999) for converting the set of "crisp" numeric rules into a reduced set of fuzzy (linguistic) rules.

4.4 Rules Fuzzification and Reduction

We have chosen the following terms (words in natural language) for each type of numerical attribute in the BATCHES table:

- Flow Time: *short, long*.
- Current: *low, medium, high*.
- Yield: *low, normal, high*.

To convert the above attributes into linguistic variables, we have defined triangular membership functions associated with each term (see Figures 6-8 below). Triangular functions are frequently used in the design of fuzzy systems (Wang 1997). To protect the confidentiality of the original data, the membership functions are shown here without the values of the X-axis. The nominal attribute *Size* was not fuzzified.

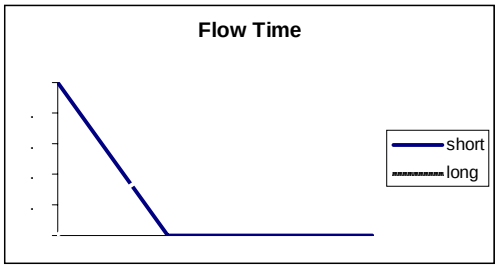

Figure 6 Membership Functions of *Flow Time*

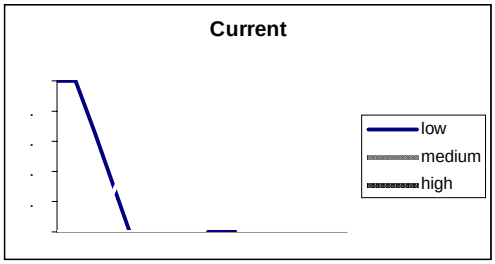

Figure 7 Membership Functions of *Current*

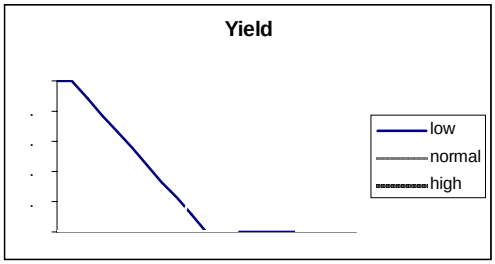

Figure 8 Membership Functions of *Yield*

Applying the fuzzification procedure to the "crisp" rules, shown in the previous sub-section, results in the following fuzzy rules:

- **Rule No. 28**: If Size is *Z* and Current is medium then Yield is normal (grade = 0.0326)
- **Rule No. 6**: If Size is *Y* then Yield is not low (grade = -0.0216)

In Table 2 below, we present the consistent set of fuzzy rules, extracted from the BATCHES table by using the conflict resolution procedure of sub-section 3.2 above. The last column represents the number of original rules (crisp / fuzzified), associated with a given fuzzy rule. As one can see, the size of the fuzzy rule base has been significantly reduced from 58 original rules to six rules only (a decrease of nearly 90%).

Table 2 The Set of Consistent Fuzzy Rules

Rule No	Rule Text	Grade	Number of Crisp Rules
0	If Size is *X* then Yield is low	0.0522	5
1	If Size is *Y* then Yield is normal	0.0226	10
2	If Size is *Z* and Current is medium then Yield is normal	0.0395	17
3	If Size is *Z* and Current is high then Yield is normal	0.0097	8
4	If Size is *Z* and Current is low and Flow_Time_29 is short then Yield is normal	0.0077	10
5	If Size is *Z* and Current is low and Flow_Time_29 is long then Yield is low	0.0176	8

All the rules in Table 2 above are *conjunctions* of fuzzy and "crisp" conditions. However, rules 2 and 3 can be merged into a *disjunction*, since they have the same consequent (*Yield is normal*). The formal algorithm for merging fuzzy rules was presented in sub-section 3.3 above and the resulting set of 5 merged fuzzy rules is shown in Table 3 below. The merged rule (no. 2) does not include all the terms associated with the attribute *Current*, and, thus, it cannot be pruned by the algorithm of sub-section 3.4.

The users (process engineers) would be particularly interested in the rules describing problematic situations, where the yield is below normal. Thus, Rule 0 indicates that the chips of the size *X* are more problematic, since their yield tends

to be low. Corrective actions may include changes of the manufacturing process, purchase of new equipment, and adjustment of chips' prices. Rule 4 says that batches having a different size (*Z*) and low current suffer from low yield, if the flow time at Operation 29 is long. In this case, the engineers should find the reason why long waiting times at this operation cause the yield to be low. Anyway, the delays may be decreased by the proper changes of the working procedures (assigning higher priority to low-current batches of size *Z*).

Table 3 The Set of Merged Fuzzy Rules

Rule No	Rule Text	Grade
0	If Size is *X* then Yield is low	0.0522
1	If Size is *Y* then Yield is normal	0.0226
2	If Size is *Z* and Current is medium or high then Yield is normal	0.0395
3	If Size is *Z* and Current is low and Flow_Time_29 is short then Yield is normal	0.0097
4	If Size is *Z* and Current is low and Flow_Time_29 is long then Yield is low	0.0077

5. Conclusions

In this paper, we have presented a new approach to extracting a compact set of linguistic rules from relational data. The approach is based on the *Information-Fuzzy Network* (IFN) methodology, which is aimed at maximizing the mutual information between input and target attributes. Post-processing of the IFN output includes information-theoretic fuzzification of numeric association rules, removal of conflicting rules, merging of consistent rules, and pruning of merged rules. As demonstrated by the case study of a semiconductor database, the process results in a small set of interpretable and actionable rules. If necessary, the fuzzification of the rules can also be helpful for hiding confidential information from unauthorized users of the rule set.

The full potential of the fuzzy set theory for efficient post-processing of data mining results has yet to be studied. Future research includes integration of the Computational Theory of Perception with other rule extraction systems like C4.5 (Quinlan 1993) and Quest (Agrawal et al. 1996). Application of the same approach to non-relational data (e.g., time series databases and multi-media documents) is another important topic.

Acknowledgment

This work was partially supported by the USF Center for Software Testing under grant no. 2108-004-00.

References

R.Agrawal, M. Mehta, J. Shafer, and R. Srikant (1996). The Quest Data Mining System. Proc. of KDD-96, pages 244-249. AAAI Press.

W.-H. Au and K. C. C. Chan (1999). FARM: A Data Mining System for Discovering Fuzzy Association Rules. Proc. of IEEE International Fuzzy System Conference, pages 1217-1222. IEEE Press.

T. M. Cover (1991). Elements of Information Theory. Wiley.

U. Fayyad and K. Irani (1993). Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. Proc. of the 13th International Joint Conference on Artificial Intelligence, pages 1022-1027. Morgan Kaufmann.

U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth (1996a). From Data Mining to Knowledge Discovery: An Overview. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Editors, Advances in Knowledge Discovery and Data Mining, , pages 1-30. AAAI/MIT Press.

H.F. Korth and A. Silberschatz (1991). Database System Concepts. McGraw-Hill, Inc.

M. Last and A. Kandel (1999). Automated Perceptions in Data Mining. Proc. of 1999 IEEE International Fuzzy Systems Conference, pages 190-197. IEEE Press.

M. Last, A. Schenker, and A. Kandel (1999). Applying Fuzzy Hypothesis Testing to Medical Data. Proc. of RSFDGrC'99, pages 221-229. Springer-Verlag.

O. Maimon, A. Kandel, and M. Last (1999). Information-Theoretic Fuzzy Approach to Knowledge Discovery in Databases. In R. Roy, T. Furuhashi and P.K. Chawdhry, editors, Advances in Soft Computing - Engineering Design and Manufacturing, , pages 315-326.

T.M. Mitchell (1997). Machine Learning. McGraw-Hill.

J.R. Quinlan (1986). Induction of Decision Trees. Machine Learning, 1 (1): 81-106.

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

C.R. Rao and H. Toutenburg (1995). Linear Models: Least Squares and Alternatives. Springer-Verlag.

S. Shenoi (1993). Multilevel Database Security Using Information Clouding. Proc. of IEEE International Conference on Fuzzy Systems, pages 483-488. IEEE Press.

T. Slawinski, et. al. (1999). A Hybrid Evolutionary Search Concept for Data-based Generation of Relevant Fuzzy Rules in High Dimensional Spaces. Proc. of IEEE International Fuzzy System Conference, pages 1432-1437. IEEE Press.

R. Srikant and R. Agrawal (1996). Mining Quantitative Association Rules in Large Relational Tables. Proc. of ACM-SIGMOD 1996 Conference on Management of Data, pages 1-12.

L.-X. Wang and J.M. Mendel (1992). Generating Fuzzy Rules by Learning from Examples. IEEE Transactions on Systems, Man, and Cybernetics, 22 (6): 1414-1427.

L.-X. Wang (1997). A Course in Fuzzy Systems and Control. Prentice-Hall.

Y. Yuan, M.J. Shaw (1995). Induction of Fuzzy Decision Trees. Fuzzy Sets and Systems, 69 (0): 125-139.

L. A. Zadeh (1999). A New Direction in System Analysis: From Computation with Measurements to Computation with Perceptions. In N. Zhong, A. Skowron, S. Ohsuga, Editors, New Directions in Rough Sets, Data Mining, and Granular-Soft Computing , pages 10-11.