A DECISION THEORETICAL BASED SYSTEM FOR INFORMATION DOWNGRADING

Ira S. Moskowitz & LiWu Chang
Center for High Assurance Computer Systems, Mail Code 5540
Naval Research Laboratory
Washington DC, 20375 USA
{moskowitz, lchang}@itd.nrl.navy.mil

Abstract: It is sometimes necessary for the owner of proprietary data to publicize some of it while keeping the rest as private. For example, when releasing census data or corporate financial information, the release must be conducted in a manner consistent with individual privacy. The process of publicly releasing formerly private data is called *downgrading*. However, it may be possible to infer unreleased private information from the downgraded public information—the so-called *inference problem*. Here, we discuss some of the design decisions that we have made, and continue to make, concerning our prototype for a high assurance system that evaluates downgrading decisions based upon the amount of private information that may be deduced through inference. Our software system, the *Rational Downgrader*, is composed of a knowledge-based *decision maker* to determine the rules that may be inferred, a *GUARD* to measure the amount of leaked information, and a *parsimonious downgrader* to modify the initial downgrading decisions. At present, we have restricted the Rational Downgrader to relational databases. Of course, the underlying theories apply to all forms of data. In this paper, we concentrate on design decisions made with the aim of achieving high assurance with respect to an optimality condition.

1. INTRODUCTION

We feel that since downgrading is necessary, it should be done in a high assurance manner. Inference problems must be analyzed and controlled. We propose the Rational Downgrader as a high assurance device to perform downgrading. The goal of the Rational Downgrader is to mitigate the inference of private information from information that is publicly available. The design goal of the Rational Downgrader is to satisfy an *assurance policy---the policy that an unqualified user cannot infer private information.* In practice, total assurance might be an unobtainable Holy Grail of perfection. However, we feel that our tool can be utilized to achieve a pragmatic level of assurance. Our preliminary work in this area is described in [CM98a], [CM98b], [MC99a], and [MC99b].

At present, we focus our attention on relational databases and classification rules (class labels). We are starting experiments with the Rational Downgrader on the UC Irvine machine learning repository [UCI] and plan to use our prototype on other publicly available databases.

2. DOWNGRADING

There exists a relational database DB and initially all rows of DB are considered private. The user or managing authority of all the information in DB is called *High*. Every case in DB is described by its row. A row is specified by its key k. Row k, r_k, is a $(n+1)$-tuple and the tuple entries are the attribute values for r_k. The last attribute value is special and is referred to as the *class label*. To avoid confusion, we will reserve the term "attribute" and "attribute value" for the first n entries of any row. It is possible for an attribute value to be missing, which is denoted by placing a ? in that entry. Class labels are never missing in DB. High determines, based upon reasons of system safety, business decisions, politics, timing, etc., which rows are truly sensitive--the *private* rows--and which rows need no longer be considered private--the *public* rows. We call this *downgrading* the rows. Two new databases, H_{db} and L_{db}, are formed. H_{db} is the same as DB except for the designation of rows as either private or public. The row keys of L_{db} are the same as the row keys of H_{db}. If r_k is public in H_{db}, then r_k in L_{db} is identical to r_k in H_{db}. If r_k is private in H_{db}, then the *n* attribute values of r_k in L_{db} are the same as in H_{db}. However, the class label of r_k in L_{db} is a missing value. In other words, it is the association of a class label with the attributes in a private row that is proprietary. The interested reader is referred to [MC99b] for details.

Downgrading by focusing upon each row as a separate entity, and not on a series of rows in conjunction with each other will not detect many inferences. (Our inference problems are different from the important work done on microdata disclosure problems using contingency tables and statistical databases, e.g., [DL]. We focus on categorical relational databases.)

Our assurance concern is the ability of a user of L_{db}, called *Low*, to infer the missing class label associated with private rows. We restate our *assurance problem* as

the ability of Low to infer an associated private class label.

Note that the assurance problem is not just the class label, but the fact that the class label is associated with a specific private row. The attributes in a private row of L_{db} are not of concern. It is assumed that those attributes alone cannot be used by Low to learn information about the missing class label in a specified private row. Rather, our concern is that knowledge of the public rows can assist Low in learning the missing associated class labels.

3. MODULAR DESIGN

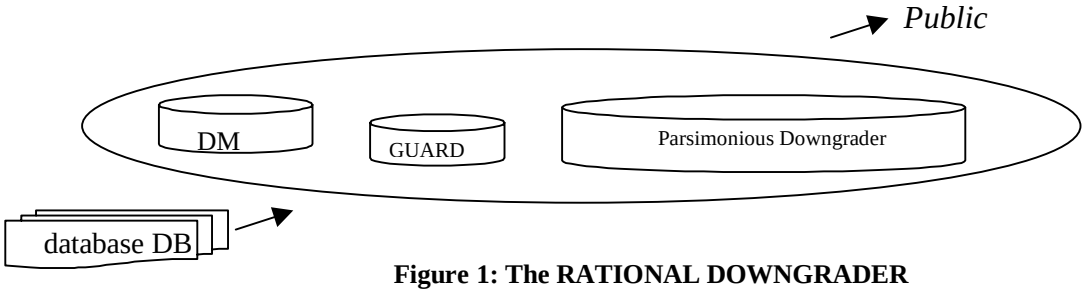

Figure 1: The RATIONAL DOWNGRADER

The RATIONAL DOWNGRADER [MC99b] is comprised of three modular components that may be replaced and upgraded as assurance, performance, and availability of software changes.

The components are the knowledge-based decision maker (DM), the GUARD, and the Parsimonious Downgrader. After it has performed its initial downgrading, High itself, (or some authority supervising High), is the "operator" of the Rational Downgrader.

Let us study the first component of the Rational Downgrader---DM. The database L_{db} can be decomposed into two databases: L^-_{db} and $L^+_{db.}$ The database L^-_{db} consists of the private rows with the class labels missing and L^+_{db} consists of the (downgraded) public rows. L^+_{db} is fed into DM. DM produces classification rules since our concern, in the prototype Rational Downgrader, is the missing associated class labels. We use the standard measurements of rule strength from the field of KDD (e.g. [AIS], [MS]). They are

support = (*number of rows in which the rule is correct ÷ total number of rows*), and

confidence = (*number of rows in which the rule is correct ÷ number of rows where the attribute values agree with rule antecedent).*

DB→ downgrade→ L^+_{db} → DM → Rule Set

The strength of a rule is the 2-tuple (support, confidence). As a research prototype, we use C4.5 [Q] for DM. C4.5 is a very popular decision tree algorithm that produces inferential rules. C4.5 uses L^+_{db} as training data and views L^-_{db} as the test data. Of course, this can be replaced by other sound knowledge-based inference systems and frameworks or combinations thereof. A DM such as CBA [HM] or Bayesian methods e.g., [CM98b] would also be applicable and we are studying their utilization.

4. INFERENCE

In [MC99b] we generalized a definition of inference given in [MC99a] that we discuss here. Let *A* be the categorical random variable representing the distribution of the missing associated class labels in L^-_{db}. Let *B* be the categorical random variable representing the distribution of the missing associated class labels in L_{db}. For both *A* and *B* we are making a *closure assumption* that the set of possible outcomes are known, are the same, and are exhausted by the class labels of L_{db}. Thus, *A* and *B* describe the same outcomes but *A* is based only on the information in the private rows, whereas *B* is based on information in all of L_{db}.

DEFINITION 1: If $A = B$, then we have **perfect noninference**; if $A \neq B$, then we have **inference**.

The type and degree of inference is what we wish to measure. The confidence of the rules goes into the probability calculations and the support of the rules gives a measurement of the strength of the inferences. Keep in mind that the method and parameters used to generate the rules (e.g. C4.5 vs. ID3 vs. CBA, etc.) will influence whether or not we "have" inference. Thus, future variants of the Rational Downgrader might use a mixture of techniques.

We want to minimize inference as much as possible. We accomplish this minimization by changing certain attribute values to missing values in the public rows. The GUARD, which we discuss next, is the component that would allow a benign inference, since, according to our assurance policy, no harm would occur.

5. THE GUARD

Rules learned from DM are applied to $L^-_{db.}$ As noted we view L^+_{db} as training data and L^-_{db} as test data. The information learned from L^-_{db} via the DM rules must now be measured. This is the function of the GUARD.

Rule Set $\oplus$ L^-_{db} → GUARD = pass or fail

The GUARD must determine if the assurance policy is satisfied. It does this based upon the following criteria:
1-Has inference occurred?
2-Is the inference malevolent?
3-Is the malevolent inference associated with rules that have strong support?
These components determine whether we have an assurance problem or not. Item 3 is especially subjective (we will not discuss specific criteria). If High has assurance that private information will not be leaked, our analysis is complete (pass). However, what if Low is able to glean sensitive information from the public rows—the inference problem (fail)? High must reconsider the decisions that it made when it initially downgraded the information. To the best of our knowledge, there is no software tool that accomplishes this. This part---parsimonious downgrading---is an integral part of the Rational Downgrader.

6. PARSIMONIOUS DOWNGRADING

If GUARD = fail, then DB → [Parsimonious Downgrader] → $\mathbf{L}_{db} = \mathbf{L}^{+}_{db} \oplus \mathbf{L}^{-}_{db}$

Parsimonious downgrading was introduced in [CM98b]. Parsimonious downgrading calls the initial downgrading decisions into question and views these decisions as being too liberal if possible malevolent inferences are not taken into account. Parsimonious downgrading is the process of adjusting the downgrading process by making less information publicly available by inserting missing values for some of the attribute values in the public rows. This lessening of the amount of public information causes the DM to produce weaker inference rules. We call this *rule confusion*. The net effect of rule confusion is to lessen Low's ability to infer the private class labels. Of course, hiding additional public information negatively affects the functionality provided by the public rows to Low, so it must be done judiciously. We call the database of adjusted (via the missing values) public rows $\mathbf{L}_{db}$. It is decomposed (as before) into $\mathbf{L}^{+}_{db}$ and $\mathbf{L}^{-}_{db}$; of course, $\mathbf{L}^{-}_{db}$ is just L^{-}_{db}. Future development will also allow the Rational Downgrader to insert missing values into L^{-}_{db}. We stay away from that approach for now since, for a "small" amount of private rows, deletion of attribute values would cause serious performance damage. Also, if the rows are filled in a temporal manner and L^{-}_{db} is instantiated after L^{+}_{db}, the Rational Downgrader could only assure rule confusion ahead of time by manipulating L^{+}_{db}.

We assign a metric to the loss of functionality; i.e., a *penalty*. A missing value in attribute *j* of r_i is given a score of *w(i,j)*. The simplest scenario, called the *Unity scenario*, occurs when *w(i,j)* is set equal to the constant value 1.

Our goal is to weaken the rule set while minimizing the penalty. What does it mean to weaken the rule set? We are only interested in rules that affect L^{-}_{db}. We are tacitly assuming that Low can run whatever knowledge discovery engine that High does. Let us elaborate on this.

We assume that High wants to muddle any rules that can be applied to L^{-}_{db}, and Low is aware of this strategy. Thus, we see that an integral part of parsimonious downgrading is the penalty. With the penalty in mind, we note that one cannot derive any inference from a brick, but neither is a brick of functional use.

$\mathbf{L}^{+}_{db}$ → [DM] → Rule Set, Rule Set $\oplus$ $\mathbf{L}^{-}_{db}$ → [GUARD] = pass or fail

After High has performed parsimonious downgrading within the bounds set by an agreed-upon penalty function, the process must start again with the Low database being sent into DM. After DM extracts the rules and applies them to $\mathbf{L}^{-}_{db}$ and the inferences have been determined, the GUARD determines the pass/fail decision. If "pass," then High is done. High can now have an assurance that Low will not learn things it is not intended to learn. However, if the GUARD determines "fail," then the process must be restarted. One must be careful not to get into complex loops of sending the data through the Rational Downgrader. Trade-offs between assurance and functionality penalties must be evaluated and we allow sub-optimality in this respect. (In the present model, human intervention is allowed to accomplish this.)

7. PARSIMONIOUS DOWNGRADING DESIGN DECISIONS

7.1 A concrete example.

Table 1 represents our database. The database has four attributes and one class label---"sunburn" (the row number identifiers are not considered an attribute). The entire database is designated DB. The first 19 rows are considered public and they make up $L^{+}_{db.}$ The last 9 rows are considered private. Therefore, we see that the database L^{-}_{db} is the last 9 rows with missing class labels. Therefore, the database L_{db} is given by Table 2.

It is our desire to deny Low the ability to infer the missing class labels (with their row associations). Using L^{+}_{db} as training data, and L^{-}_{db} as test data, we run C4.5 (in the default mode with the –u –f options) and obtain the following rules (see Figure 2):

RULE 1: "**hair = brown**" ⇒ **N**
RULE 2: "**hair = red**" ⇒ **S**

RULE 3: "**hair = blonde**" ^ "**lotion = no**" ⇒ **S**
RULE 4: "**hair = blonde**" ^ "**lotion = some**" ⇒ **M**
RULE 5: "**hair = blonde**" ^ "**lotion = yes**" ⇒ **N**

According to the training data, RULE 1 is correct with confidence 3 out of 3 (written 3/3), RULE 2 is correct 6/6, RULE 3 is correct ¾, RULE 4 is correct 2/2, and RULE 5 is correct ¾. Low knows these rules and if Low applies them to L^-_{db}, Low would obtain every missing class label except for row 24, which, by using C4.5, Low would misclassify as M instead of S. The ability for Low to learn 8 correct associated class labels is not acceptable. High is keeping very little away from Low by downgrading rows 20 through 28 with missing class labels.

Now High decides to perform parsimonious downgrading and insert missing values in for some of the attribute values in rows 1 through 19. For simplicity, we assume a penalty function of 1. We further assume a maximum cost of 5. Therefore, only 5 missing values may be inserted (of course if less than 5 will imply the same rule confusion then we should do that.) A naïve approach

row		*hair color*	*height*	*weight*	*lotion use*	*sunburn*
1		blonde	average	light	yes	N
2		blonde	average	heavy	yes	N
3		blonde	short	average	yes	N
4		blonde	tall	heavy	no	N
5		blonde	tall	average	yes	M
6		blonde	short	heavy	some	M
7		blonde	average	light	some	M
8		blonde	short	light	no	S
9		blonde	short	average	no	S
10		blonde	tall	light	no	S
11		brown	tall	heavy	no	N
12		brown	average	light	no	N
13		brown	short	average	some	N
14		red	average	light	some	S
15		red	tall	heavy	no	S
16		red	average	light	no	S
17		red	average	average	no	S
18		red	short	average	no	S
19		red	average	light	some	S
20	**1**	blonde	tall	heavy	yes	N
21	**2**	blonde	short	heavy	some	M
22	**3**	blonde	average	heavy	yes	N
23	**4**	blonde	short	average	no	S
24	**5**	blonde	tall	light	some	S
25	**6**	brown	average	light	no	N
26	**7**	brown	short	average	some	N
27	**8**	red	short	average	no	S
28	**9**	red	short	light	some	S

Table 1

would be to go through all possible ways of putting 5 missing values in for attribute values in L^+_{db}. This approach is computational infeasible since the combinatorial possibilities grows exponentially. Instead, here we make a design decision and attempt an information theoretical approach to inserting missing values. We call this the rule-based approach, since we use the actual rules generated by C4.5 to decide where to insert the missing values.

7.2 Rule-based missing values.

C4.5 generates rules via the Quinlan [Q] gain condition. This is basically the normalized mutual information between attributes and the class label. Of course, C4.5 is rather sophisticated, since rule pruning is performed via various statistical techniques, but the gist is still that of information theory. The rules represent the strongest dependencies between the attributes and the class labels in an efficient manner. We exploit these dependencies in deciding where High should insert missing values. We exploit the decision trees associated with the rules. The rule clauses are generated in descending path order down the decision tree. This order represents the information theoretical interaction between the attributes and the class label.

Step 1: See which rules are needed to classify L^-_{db}. Call these rules the kernel. (In our above example, all of L^+_{db} is the kernel.) From L^+_{db} we delete any cases which do not support a rule in the kernel and from these cases we delete attributes that are not represented in the rules clauses. In the above example, this leaves us with Table 3. Keep in mind the last column of Table 3 is still the class label while the first two columns (we do not include the first "column" which is just a designator for the row, or case, number) are the attribute values. It is from these attribute columns that we change

row		hair color	height	weight	lotion use	sunburn
1		blonde	average	light	yes	N
2		blonde	average	heavy	yes	N
3		blonde	short	average	yes	N
4		blonde	tall	heavy	no	N
5		blonde	tall	average	yes	M
6		blonde	short	heavy	some	M
7		blonde	average	light	some	M
8		blonde	short	light	no	S
9		blonde	short	average	no	S
10		blonde	tall	light	no	S
11		brown	tall	heavy	no	N
12		brown	average	light	no	N
13		brown	short	average	some	N
14		red	average	light	some	S
15		red	tall	heavy	no	S
16		red	average	light	no	S
17		red	average	average	no	S
18		red	short	average	no	S
19		red	average	light	some	S
20	1	blonde	tall	heavy	yes	?
21	2	blonde	short	heavy	some	?
22	3	blonde	average	heavy	yes	?
23	4	blonde	short	average	no	?
24	5	blonde	tall	light	some	?
25	6	brown	average	light	no	?
26	7	brown	short	average	some	?
27	8	red	short	average	no	?
28	9	red	short	light	some	?

Table 2

instantiations into missing values. This is all done to optimize the rule confusion. The GUARD determines the acceptable level of rule confusion. For now we are assuming that the GUARD wishes to maximize the rule confusion (in general, the GUARD will perform a playoff against the functionality).

The agreed number of missing values are now substituted for attribute values in the kernel rules. C4.5 is run on this parsimoniously downgraded L^{+}_{db}. For every test (private) class label that is misclassified, we assign the confidence that the rule has in this misclassification. These values are then added together. (Note that we are not giving any test case preference over another. We can accomplish this by weighting the confidences of the miscalculations.) This score makes up the confusion value assigned to this assignment of missing values. This is done (which is still a huge computational effort) for every possible assignment of the *n* missing values. We collect the set of assignments that results in the maximal confusion value (if there are no misclassifications, a similar approach is used with respect to the confidence values of the correct classifications). The GUARD then randomly picks one of these assignments of missing values as the optimal assignment of missing values.

Our method gives a computationally quicker method than just assigning randomly the missing values. Unfortunately, our method is still quite computationally complex due the factorial nature of assigning the missing values. The second step is an attempt to continue Quinlan's analogy with noisy communication channels by determining an entropy-based approach to assigning missing values. We also will investigate inserting missing values into L^{-}_{db}, instead of just L^{+}_{db}.

With respect to the example given in Tables 1 & 2, we have not gone through all 29 choose 5 (approximately 120 000) ways of assigning missing values. For our example, we conclude with some discussion. If we only have 5 missing values to work with we would not concentrate on rows 8 or 9 of $L^{-}_{db.}$ This is because those rows need RULE 2 to infer their class label. To confuse RULE 2, the Rational Downgrader would need to insert all 5 missing values for any 5 hair color attribute values in rows 14 through 19 of L^{+}_{db}. However, C4.5 is still able to produce a rule that infers that the class labels for rows 8 and 9 of L^{-}_{db} are S. An "intelligent" approach would attempt to achieve the biggest bang for the buck by analyzing how many missing values must be inserted to confuse a rule in conjunction with how many rows of L^{-}_{db} need this rule for classification purposes. By contrast, we see that inserting 3 missing values for the "lotion use" attribute in three of rows 1,2,3, or 5 of L^{+}_{db} would confuse RULE 5 and would not allow Low to learn the class labels of rows 1 or 3.

8. PRESENT SOLUTION AND CONCLUSION

This idea of rule confusion must also take into account the amount of confidence C4.5 generates along with the (hopefully) erroneous rules. In order to bypass an exhaustive search on the attribute values, the Rational Downgrader is presently implemented by using an informative search. It makes suggestions based upon a penalty function as to where to put a missing value, and selects the one with a minimal penalty. Then, after inserting the missing value the Rational Downgrader is run again to determine the next missing value. Globally this may not be the optimal solution but we must have a trade-off against impractical complexity issues. The proposed penalty function takes into account rule confusion, rule confidence, and how many private rows are affected from rule confusion. We are attempting to automate this process and take into account the number of training cases, the number of test cases associated with a leaf, and the classification error of that leaf. Note that our approach is in the area of data mining. Data miners have successfully used simulated annealing [EBM] and genetic algorithms [CAF] to expedite their searches. We are attempting to see if such techniques will work with the Rational Downgrader.

Informative search is a greedy search so, instead of optimality, we are willing to accept a certain level of inference confusion. We hope to fully automate our process and improve our search methods. At this stage we feel that further experimentation is in order and we are presently running additional experiments.

row	*hair color*	*lotion use*	*sunburn*
1	blonde	yes	N
2	blonde	yes	N
3	blonde	yes	N
4	blonde	no	N
5	blonde	yes	M
6	blonde	some	M
7	blonde	some	M
8	blonde	no	S
9	blonde	no	S
10	blonde	no	S
11	brown		N
12	brown		N
13	brown		N
14	red		S
15	red		S
16	red		S
17	red		S
18	red		S
19	red		S

Table 3

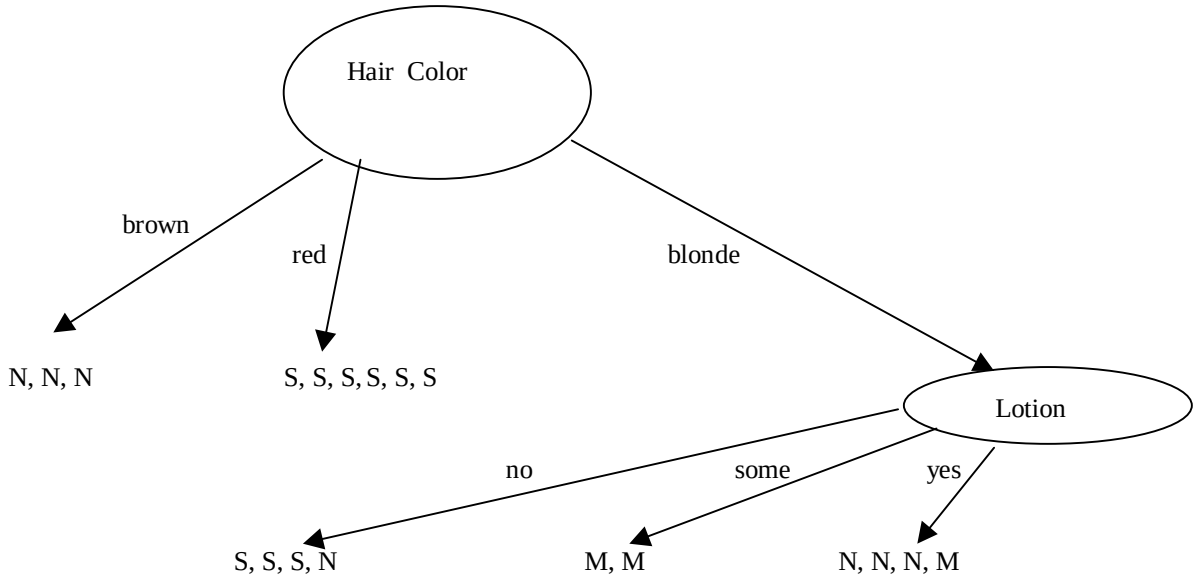

Figure 2

Because of our experimentation, we have come up with an efficient method of parsimoniously downgrading the database in our example (Table 1). Our experiment utilized the fact the decision maker C4.5 is based upon valid statistical and information theoretic principals. Keeping this in mind along with our comments about intelligently inserting missing values (section 7.2) we found that inserting missing values as shown in Table 4 results in Low misclassifying 5 class labels (r_{21}, r_{23}, r_{24},r_{25}, r_{26}). This is an increase of 4 over the original L^{+}_{db}. No other assignment of missing values will cause more than 5 misclassifications or have more inaccuracy than that generated by using Table 4. However, there are other assignments of the 5 missing values that result in the same rule confusion as Table 4. We are not looking for uniqueness, only existence.

Much still has to be done; however, we feel that our early prototyping efforts have shown a proof of concept and the Rational Downgrader will become a useful high assurance tool to assist in privacy and downgrading efforts.

row	*hair color*	*lotion use*	*sunburn*
1	blonde	yes	N
2	blonde	yes	N
3	blonde	yes	N
4	blonde	no	N
5	blonde	yes	M
6	blonde	?	M
7	blonde	?	M
8	blonde	no	S
9	blonde	no	S
10	blonde	no	S
11	?		N
12	?		N
13	?		N
14	red		S
15	red		S
16	red		S
17	red		S
18	red		S
19	red		S

Table 4

9. REFERENCES

[AIS] *Mining Association Rules between Sets of Items in Large Databases*, R. Agrawal, T. Imielinski, & A. Swami, Proc. ACM SIGMOD Conference, Washington DC, May 1993.

[CAF] *A Hybrid Genetic Algorithm/Decision Tree Approach for Coping with Unbalanced Cases*, D.R. Carvalho, B.C. Avila, A.A. Freitas, Proc. PADD'99, pp. 61-70, London, UK, April 1999.

[CM98a] *Parsimonious Downgrading and Decision Trees Applied to the Inference Problem*, L. Chang and I.S. Moskowitz, Proc. NSPW'98, Charlottesville, VA, October 1998.

[CM98b] *Bayesian Methods Applied to the Database Inference Problem*, L. Chang and I.S. Moskowitz, Proc. IFIP WG 11.3 Working Conference on Database Security, Chalkidiki, Greece, July 1998.

[DL] *Disclosure-Limited Data Dissemination*, G.T. Duncan and D. Lambert, Journal of the American Statistical Association, V. 81, No. 393, March 1986.

[EBM] *The Use of Simulated Annealing for Clustering Data in Databases*, F.J. McErlean, D.A. Bell, and S.I. McClean, Information Sciences, 15(2): 223-245, 1990.

[HM] *Integrating Classification and Association Rule Mining*, B. Liu, W. Hsu, and Y. Ma, Proc. 4th International Conference on Knowledge Discovery and Data Mining (KDD 98), NY, NY, August 1998.

[MC99a] *A Formal View of the Database Inference Problem*, I.S. Moskowitz and L. Chang, Proc. CIMCA'99, pp. 254-259, Vienna, Austria, February 1999.

[MC99b] *The Rational Downgrader*, I.S. Moskowitz and L. Chang, Proc. PADD'99, pp. 159-165, London, UK, April 1999.

[MS] *Discovering Predictive Association Rules*, N. Megiddo and R. Srikant, Proc. 4th International Conference on Knowledge Discovery and Data Mining (KDD 98), NY, NY, August 1998.

[Q] *C4.5 Programs for Machine Learning*, J. Ross Quinlan, Morgan Kaufman Publishers, 1993.

[UCI] http://www.ics.uci.edu/~mlearn/MLRepository.html, UC Irvine research database repository.