CERIAS Tech Report 2001-96
Protecting Against Data Mining through Samples
by Christopher Clifton
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

To appear in *Proceedings of the Thirteenth Annual IFIP WG 11.3 Working Conference on Database Security*, July 26-28, Seattle, WA.

Chapter 1

PROTECTING AGAINST DATA MINING THROUGH SAMPLES

Chris Clifton

Abstract Data mining introduces new problems in database security. The basic problem of using non-sensitive data to infer sensitive data is made more difficult by the "probabilistic" inferences possible with data mining. This paper shows how lower bounds from pattern recognition theory can be used to determine sample sizes where data mining tools cannot obtain reliable results.

1. INTRODUCTION

The problem of inference has received considerable attention in the database security community. The basic problem is using non-sensitive, or "low", data to infer sensitive, or "high" facts. As an example, we may want to keep the presence of a particular type of equipment (e.g., "super-secret aircraft" (SSA)) at a particular location ("Secret Base", SB) secret. However, to support logistics we want to make information about supplies needed by *all* bases available. The inference problem occurs if parts that are only used for the SSA are ordered by SB – from this we can infer that there must be a SSA at SB.

Most work in preventing inference in multi-level secure databases has concentrated on preventing such "provable" facts [Delugach and Hinke, 1996]. Recent work has extended this to capturing data-level, rather than schema-level, functional dependencies [Yip and Levitt, 1998]. However, data mining provides what could be viewed as *probabilistic* inferences. These are relationships that do not always hold true (are not a functional dependency), but hold true substantially more often than would be expected in "random" data. Preventing this type of inference detection is beyond existing methods.

As an example, what if there are no parts that are used *only* for the SSA? The functional dependency inference problem no longer exists. However, there may be some items that are used *more heavily* by the SSA than other aircraft (e.g., it uses a great quantity of fuel). Data mining could find such a relationship; for

example bases X, Y, and SB use an unusual quantity of fuel *in relation to other supplies*. If we know that bases X and Y support the SSA, we can make a good guess that SB does as well.

Hinke and Delugach [Hinke and Delugach, 1992, Hinke et al., 1997] give a breakdown of inference into seven classes of problems. The first six rely on combining known rules (e.g., Part A is only used on an SSA) with non-sensitive data (Part A was suppled to base SB) to infer sensitive facts. Class 7 is the inference of a sensitive rule. It is noted that this "represents a considerably different target than the previous ones", and as a result has received considerably less attention in the database security community. However, one of the primary focuses of data mining technology is inferring rules, with the rise of data mining technology, this has become a recognizable problem.

What can we do about this, in particular when we don't know what the adversary will be looking for? (In cases where we *know* the inference we must keep secret, like the example above, other techniques are available.[Johnsten and Raghavan, 1999]) We can ensure that *any* results will be suspect. If we can convince the adversary that any "guess" (inferred rule) gained from data mining is no more likely to be a good guess than a random guess, the adversary will not be able to trust any data mining results.

How do we do this? Inferences from data mining have some level of confidence and support. We will show how to ensure that either:

1. The rules "discovered" may be incorrect (the levels of confidence and support could be significantly off); or
2. It is likely that rules exist with higher confidence and support than those found.

We propose to accomplish this by ensuring that the data available to the adversary is only a sample of the data on which the adversary would like the rules to hold (note that in some cases this "sample" may be an entire database, as long as the "inferences" we wish to protect against apply to a larger population than that reflected in the database). We show that we can draw a relationship between the sample size and the likelihood that the rules are correct.

We base this on the fact that inference rules can be used to classify. Vapnik [Vapnik, 1982] has shown error expectations in classification on samples. This is due to expectations of a random sample of a population having a different distribution with respect to any classification information than the population as a whole. If we can show that a "best possible" classifier is likely to be incorrect, we can work back to show that any rules (at best a "best possible" classifier) will also likely be incorrect.

We divide this into two variations:

1. We are concerned with protecting a particular "object" (e.g., any relationship where the target is a single value for a given variable). Note that

Table 1.1 Base Order Database.

Date	*Location*	*Item*
1/1/97	**SB**	**Fuel**
1/1/97	**X**	**Fuel**
1/2/97	Y	Fuel
1/4/97	**A**	**Fuel**
1/6/97	**B**	**Food**
1/10/97	**B**	**Food**
1/18/97	A	Food
1/22/97	A	Food
1/24/97	**B**	**Fuel**
1/31/97	**X**	**Fuel**
2/3/97	**SB**	**Fuel**

we do not require predefining the value. We can reduce the problem of learning a binary classifier to this.

2. We are concerned with a particular class of rules (e.g., inference rules involving a single independent variable). This limits our space of possible classifiers.

As we will show; these two variations lead to different solutions.

2. MOTIVATING EXAMPLE

In this section we will give a more detailed presentation of the "inference through supply orders" example, along with a "proof by example" of how providing only a sample of the data can be used to prevent data mining from making the inference *base SB is likely to support an SSA due to similar ordering patterns to bases X and Y.*

First, assume that the complete order database is as presented in Table 1.1. If we develop an *Item* classifier using this table, we find the rules:

If Location=X then Item=Fuel (confidence 100%, support 29%)
If Location=Y then Item=Fuel (confidence 100%, support 14%)
If Location=SB then Item=Fuel (confidence 100%, support 29%)

Armed with this knowledge, we can search for reasons why X and Y order only Fuel (other reasons that differentiate them from A and B). If we find a common factor (e.g., they support the SSA), we can guess SB has this common factor.

Note that if we only have a sample of the database, we may develop a different set of rules. If we use only the **highlighted** rows from Table 1.1 we get a sample containing 70% of the complete database. The rules generated from this sample are:

If Location=A then Item=Fuel (confidence 100%, support 17%)
If Location=X then Item=Fuel (confidence 100%, support 33%)
If Location=SB then Item=Fuel (confidence 100%, support 33%)

If we looked for common factors between Locations A and X (that differentiated them from others), we would not find the "threatening" evidence that they both support the SSA. Thus we have preserved the secrecy of the SSA at SB.

There are a number of problems with this example:

1. The support of the first rule is low. If the adversary were to ignore rules with support below 20% (in both cases), it might still be possible to obtain the desired information (although with less confidence, as the presence of the base Y supporting the SSA, but not present in the rule, would lessen the impact of this).

2. What if we chose a different sample? Some samples would *improve* the rules, and increase the adversary's ability to find the desired information.

3. Does this sample database still provide the desired information (e.g., an audit of order deliveries, or supporting the improvement of logistics through better prediction of ordering needs)?

Problem 3 is difficult – we must know the intended purpose of the data to determine if the sample is sufficient. Some purposes (such as prediction of ordering needs) are particularly problematic: If we aim to prevent an adversary from learning rules *we don't even know about*, we will necessarily prevent learning any rules. However, if the goal relies on specific data items, rather than inferences among the data items (such as comparing specific orders with their delivery traces), we need only ensure that the desired items are provided. We must be careful, though, to avoid problem 2 by letting the adversary choose the sample. Although not the focus of this paper, we discuss this issue later.

What we address is problem 1. If we can show that the rules with high support or high confidence *on the sample* do not necessarily have high support and confidence *on the complete database*, then the adversary cannot rely on the rules produced from the sample. We don't show that the rules obtained on the sample *are* bad, but that they are *likely* to be bad. The goal of this work is to show that we can convince the adversary of the likelihood of such a failure, *without knowing the problem the adversary wants to solve*. Many data mining techniques (including the production rules shown above) produce rules or knowledge that can be used to classify items in the database. Vladimir Vapnik has shown error limits in classification when the classification is learned from a sample[Vapnik, 1982]. In Section 4., we will use an adaptation of Vapnik's work to show the difficulty of learning as a function of sample size.

To give an idea of how this would work in practice, we will use the results derived later to demonstrate how large a sample would be reasonable for the above example.

Assuming there are at most two food and/or fuel orders per month, and that the adversary attempts to predict using the number of orders of each type per month, e.g., a rule is of the form:

$$January(Fuel, 2)\&January(Food, 0)\&February(\ldots \Rightarrow \text{Supports SSA}$$

Further assume that using a collection of such rules we can develop a "perfect" classifier (one that will always give us the right result). If we are willing to tolerate that with a 50% probability, the learned classifier can be expected to be wrong 40% of the time, we can allow a sample size of over 175 billion where a single sample is all of the orders for a single base for a year (based on Theorem 1; we will discuss how these numbers are derived in Section 4.1). This is large, but the reason is that there are a great many possible rules (3 possible values for each of food and fuel gives 9 possible values per month; or 9^{12} possible values a year). If the sample doesn't contain an *exact* instance of a rule, the classifier won't know if it applies. Thus the need for a large sample size. This is a problem with complex classifiers – they don't generalize well.

If we assume that a simple correct classifier exists, and that the adversary has the sense to look for a simple classifier, we have more serious constraints. In particular, if we assume N (the number of possible classifiers) is 6: fuel $>>$ food, low total order for the year; fuel $\approx$ food, low total order; fuel $<<$ food, low total order; fuel $>>$ food, high total order; fuel $\approx$ food, high total order; fuel $<<$ food, high total order; we can only allow a sample size of $(6-1)/(4*.4) = 3$ (again, a sample is complete information on a base for a year, or 72 tuples from a complete version of Table 1.1.)

This assumes a perfect classifier exists, however with a simple classifier it is unlikely that it *can* perfectly classify the data. If the best possible classifier has some error, it is more difficult to state exactly what we mean by the error of the classifier learned from the sample. We will now give some more detail on error estimation. In Section 4. we discuss the specific theorems that allow us to determine these bounds; we will then return to discussion of this example.

3. BASIC IDEAS OF ERROR ESTIMATION

Our purpose in this section is to show how we can control the expected error of a classifier by limiting sample size. Figure 1.1 gives an idea of the problem. We want to control (by varying the sample size) the error D between the classifier the adversary can expect to learn from the sample and the "best possible" (Bayes) classifier. Note that the space of possible classifiers $\mathcal{C}$ may not include the Bayes classifier. Therefore the error is composed of two components: The approximation error D_a between the best classifier $L_{\mathcal{C}}$ available in $\mathcal{C}$ and the Bayes classifier L^*, and the estimation error D_e between the classifier L_n learned from the sample and $L_{\mathcal{C}}$. The difficulty is that there

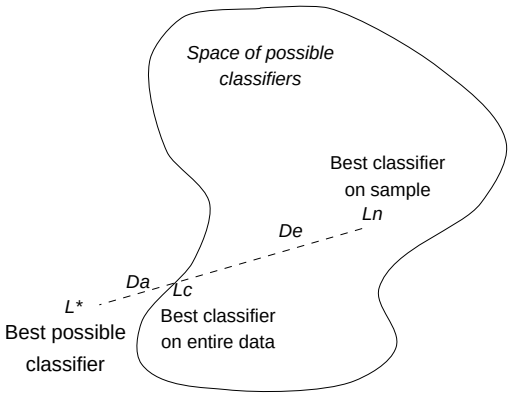

Figure 1.1 Distance between best classifier on sample and best classifier on data.

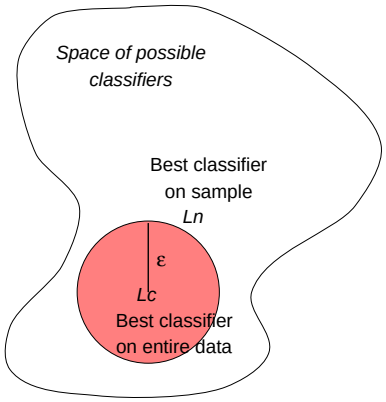

Figure 1.2 Error of best classifier on sample L_n worse than best classifier $L_{\mathcal{C}}$ by more than ϵ.

are many possible "best classifiers" L_n depending on the sample. Thus our goal is to analyze the *expected value* $\mathbf{E}\{D\}$ of the error given the sample size.

There are actually three types of error:

Bayes Error: This is the expected error of the "best possible" classifier on the entire database. If the output is a function of the input, this is 0. This is entirely dependent on the data, and as such we can say little without knowing specifically what we want to protect against.

Approximation Error: This is the difference between the expected error of the best classifier from the types of classifiers we are considering, e.g., decision trees, and the Bayes classifier. The more complex the classifier, the lower the expected approximation error.

Estimation Error: This is the difference in expected error between a classifier learned from a sample and the best classifier available. This is what we can control by varying the sample size.

There are various things we might like to say:

1. Given an error estimate, that error estimate will be off (different from the Bayes error) by some amount ϵ with probability δ.

2. The expected difference between a learned classifier and the Bayes error is at least ϵ with probability δ.

3. Given a sample, the error of the learned classifier can be expected to differ from the best classifier *of that type* by ϵ with probability δ.

4. Given a type of classifier, we can expect the best possible classifier of that type to differ from the Bayes error by ϵ with probability δ.

Item 1 gives a lower bound – it says that even if the adversary "guesses" a great classifier, the adversary's estimate of how good the classifier is (using the sample) will likely be off. However, this is not likely to be a tight bound on the actual problem: what the adversary can learn. Likewise 4 isn't all that interesting; it says how good a classifier is possible, but nothing about what can be learned from the sample.

We will address 3, the estimation error. Figure 1.2 gives an idea of the goal: given a circle of radius ϵ, we want to choose a sample size such that with probability δ, the best classifier on the sample L_n will be outside the circle. If L_n is outside the circle, then at least ϵ percent of the time L_n gives the "wrong" answer, even though a classifier $L_{\mathcal{C}}$ exists that gives the right answer.

We will also see that the formulas for estimation error are dependent on approximation error, giving a way of estimating 2.

4. LIMITS BASED ON SAMPLE COMPLEXITY

The *sample complexity* of a problem is the size of sample needed to ensure that the expected error of a classifier learned from the sample is within given bounds. What we can show is the following: Given a "best *estimation* error" that we will tolerate, and a minimum probability that the adversary will not be able to do better than that error, what is the largest sample we can allow?

Formally, we are determining the sample complexity $N(\epsilon, \delta)$, defined as the smallest integer n such that

$$\sup_{(X,Y)\in\mathcal{X}} \mathbf{P}\{L(g_n) - L_{\mathcal{C}} \geq \epsilon\} \leq \delta$$

where $L_{\mathcal{C}}$ is the best classifier in $\mathcal{C}$:

$$L_{\mathcal{C}} \stackrel{\text{def}}{=} \inf_{\phi\in\mathcal{C}} \mathbf{P}\{\phi(X) \neq Y\}.$$

and $L(g_n)$ is the classifier selected based on the training data:

$$L(g_n) = \mathbf{P}\{g_n(X) \neq Y | ((X_1, Y_1), \ldots, (X_n, Y_n))\}$$

This states that there is a distribution of the data such that if the sample size $n < N(\epsilon, \delta)$, then with probability at least δ, a classifier exists that outperforms the chosen one by at least ϵ. The key factor here is that there is *a distribution* of the data where this holds. This doesn't mean a specific choice of the sample is necessary; the dependence is instead on the characteristics of the data as a whole. There exists a distribution of the data such that the bounds will hold on average across all random samples of the data. (One such distribution is skewed

based on the classifier: most of the data conforms to a single rule left-hand side. This is not an unreasonable distribution to expect in practice. For example, if we assume that the information leading to a sensitive inference is a relatively small part of the database, we are likely to have this type of distribution.)

We begin with two definitions of Vapnik-Chervonenkis theory[Vapnik, 1982] (notation from [Devroye et al., 1996]): First we define the *shatter coefficient* of the classifier:

Definition 1 *Let $\mathcal{A}$ be a collection of measurable sets. For $(z_1, \ldots, z_n) \in \{\Re^d\}^n$, let $N_{\mathcal{A}}(z_1, \ldots, z_n)$ be the number of different sets in*

$$\{\{z_1, \ldots, z_n\} \bigcap A; A \in \mathcal{A}\}.$$

The n-th shatter coefficient of $\mathcal{A}$ is

$$s(\mathcal{A}, n) = \max_{(z_1,\ldots,z_n)\in\{\Re^d\}^n} N_{\mathcal{A}}(z_1, \ldots, z_n).$$

That is, the shatter coefficient is the maximal number of different subsets of n points that can be picked out by the class of sets $\mathcal{A}$.

Definition 2 *Let $\mathcal{A}$ be a collection of sets with $|\mathcal{A}| \geq 2$. The largest integer $k \geq 1$ for which $s(\mathcal{A}, k) = 2^k$ is denoted by $V_{\mathcal{A}}$, and it is called the Vapnik-Chervonenkis dimension (or* VC *dimension) of the class $\mathcal{A}$. If $s(\mathcal{A}, n) = 2^n$ for all n, then by definition, $V_{\mathcal{A}} = \infty$.*

We can see that the *shatter coefficient* of the classifier must be less than or equal to the number of distinct rule sets = $2^{number\ of\ rules}$. (Actually $m^{number\ of\ rules}$, where m is the number of possible output values.) Since the VC-dimension is the largest k such that the shatter coefficient = 2^k, we can see that the VC-dimension for binary decision rules is the number of distinct rules. (For non-binary, it works out to k s.t. $2^k = m^{number\ of\ rules}$, or $k = \log_2(m^{number\ of\ rules}) = number\ of\ rules \log_2(m)$.) Looking back at the example of Section 2., if we say that we are trying to learn if a base supports the SSA based on the number of fuel and food orders in a year, and we assume at most two orders of each type per month, we can see that for a year there are $24 * 24$ possible rules, so the VC dimension = 576. If we were to allow a more complex classifier, say the number of orders per month for each month (e.g., this would be useful if the SSA only flew in good weather), we would have $(\text{possible food orders per month} * \text{possible fuel orders per month})^{\text{number of months}} = 16777216$ possible rules.

We address this in two separate cases: Where a perfect classifier exists in $\mathcal{C}$, and where one does not. In the first case, what we are bounding is the total error (since there is no approximation error). There is a formula for this that holds for all $\delta \leq 1/2$:

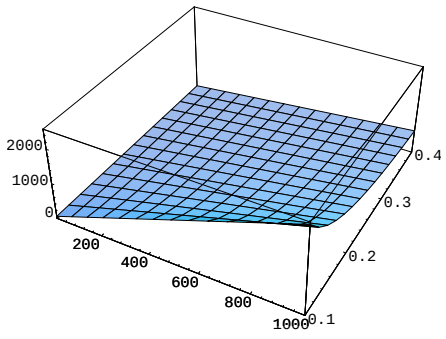

Figure 1.3 Value of n (vertical scale) where probability $\delta = 1/2$ that there is error ϵ, as function of error V (left scale) and ϵ (right scale), when a perfect classifier is available.

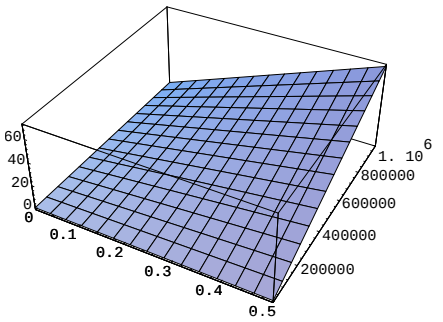

Figure 1.4 Value of n (vertical scale) below which error $\epsilon > .1$ with probability $\delta > .1$ as a function of L (left scale) and V (right scale).

Theorem 1 *[Devroye and Lugosi, 1995, Devroye et al., 1996]. Let $\mathcal{C}$ be a class of discrimination functions with* VC *dimension $V \geq 2$. Let $\mathcal{X}$ be the set of all random variables (X, Y) for which $L_{\mathcal{C}} = 0$. For $\delta \leq 1/2$ and $\epsilon < 1/2$,*

$$N(\epsilon, \delta) \geq \frac{V-1}{4\epsilon}.$$

What this comes down to is the following. If a perfect classifier exists, and we have seen an example to which a rule applies, then we will always get that rule right. If we are asked to classify something where the training data didn't contain a similar sample (similar in the sense that a rule left-hand side matches), we will just be guessing. Thus, as the number of rules (V) goes up, the sample size needed does as well.

Figure 1.3 shows the minimum n needed for various values of ϵ and V. Note that the high values of V are likely to be more relevant in practice, as it is unlikely a perfect classifier will exist if the classifier is simple.

More interesting is what happens when there isn't a perfect classifier (the approximation error is greater than 0).

Theorem 2 *[Devroye and Lugosi, 1995, Devroye et al., 1996]. Let $\mathcal{C}$ be a class of discrimination functions with* VC *dimension $V \geq 2$. Let $\mathcal{X}$ be the set of all random variables (X, Y) for which for fixed $L \in (0, 1/2)$,*

$$L = \inf_{g \in \mathcal{C}} \mathbf{P}\{g(X) \neq Y\}.$$

Then for every discrimination rule g_n based on $X_1, Y_1, \ldots, X_n, Y_n$,

$$N(\epsilon, \delta) \geq \frac{L(V-1)e^{-10}}{32} \times \min\left(\frac{1}{\delta^2}, \frac{1}{\epsilon^2}\right),$$

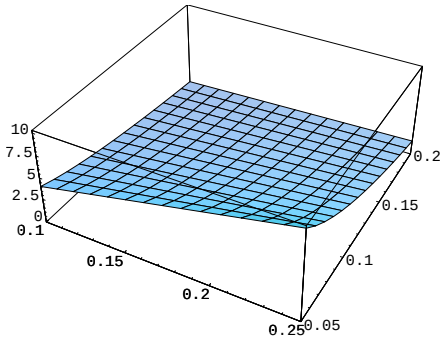

Figure 1.5 Value of n (vertical scale) below which a guarantee of error within 0.1 impossible as a function of L (left scale) and d (right scale).

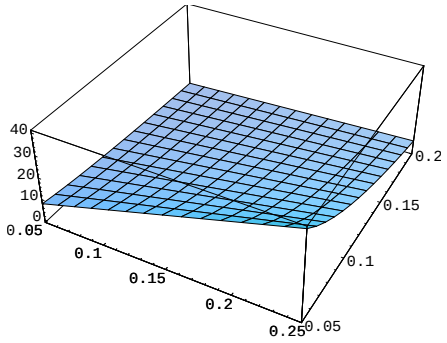

Figure 1.6 Value of n (vertical scale) below which a guarantee of error within 0.05 impossible as a function of L (left scale) and d (right scale).

and also, for $\epsilon \leq L \leq 1/4$,

$$N(\epsilon, \delta) \geq \frac{L}{4\epsilon^2} \log \frac{1}{4\delta}.$$

Note that this gives us two bounds. One is dependent on L, V, ϵ, and δ (although for practical purposes we would let $\epsilon = \delta$ when using this); the second is only dependent on L, ϵ, and δ.

Some sample values for the first, based on $\epsilon = \delta = .1$ (10% of being wrong at least 10% of the time) are given in Figure 1.4. Intuitively, this is based on the probability of choosing the wrong rule to use; this gives a sample size if our primary concern is how the adversary chooses a given outcome rather than their ability to predict the outcome. In other words, the knowledge is in the rule, not in the application of the rule.

The second formula is intuitively based on guessing the wrong outcome for a given rule. Sample values are given in Figures 1.5 ($\epsilon \geq 0.1$) and 1.6 ($\epsilon \geq 0.05$).

Note that these give small sample sizes. However, they allow us to make a strong statement: No matter how good the adversary's data mining technology, there are circumstances under which they can *expect* the results to be poor.

4.1 BACK TO THE EXAMPLE

The figures given in Section 2. are based on Theorem 1. This is appropriate for the complex classifier case (a perfect classifier is likely), and due to the huge VC-dimension (9^{12}) of such a classifier we end up with a sample size $N(.4, .5) = (9^{12} - 1)/(4 * .4) > 175$ billion.

However, the simple classifier had a VC-dimension of 6. This gives a sample size of 3 if a perfect classifier exists. Theorem 2 handles the case where no perfect classifier exists. The first formula depends on large VC-dimension V (it is really only useful when $V > 15,000$). However, the second form gives us something to work with. If we start by assuming that such a simple classifier can be correct at most 75% of the time ($L = .25$), and we want the adversary to be forced to accept an error of 10% (in other words, they can expect to be right only 65% of the time, even though they could do as well as 75%) with probability $\delta = 0.15$, gives us our allowed sample of 3 years of data.

Note that this is the same as the $L_{\mathcal{C}} = 0$ (perfect classifier exists) case with $\epsilon = 0.4$ and $\delta = 0.5$. This seems strange; it appears easier to learn a good classifier when a perfect one doesn't exist. Intuitively, we can say that the *learning problem* is harder if a perfect answer exists.

4.2 EFFECT ON A SINGLE RULE

We have determined what the expected error is for a *set* of rules. The next question is, what confidence can the adversary have in a single inference rule?

The preceding section gives an answer to this question. Since what we have determined is the probability of the learned classifier failing to perform as well as the best possible classifier *on a given input*, it follows that a failure means that the rule that applied gave the wrong output. Thus it is the probability that for any given rule left-hand side (input), the output is "backwards" (since this is a binary classifier).

This, in a sense, is a worst case error: We have a rule that gives exactly the opposite of the proper result. Although the probability of this happening may seem small (.05 or .1), the result is still significant.

5. CONCLUSIONS AND FURTHER WORK

Pattern recognition theory gives us tools to deal with security and privacy issues in data mining. Limiting the sample size that can be mined allows us to state clear limits on what can be learned from the sample. These limits are in the form of expected error on what is learned. What they allow us to do is tell an adversary, "Here is a sample you may mine, but you can expect any result you get will be wrong $\epsilon\%$ of the time with probability δ, *no matter how good your data mining is*". It gives us sample sizes where we can *expect* that the sample may be misleading.

One advantage of this approach is that the *method* can be open. The adversary cannot use knowledge of how we restrict sample size to improve the data mining process. In fact, the knowledge that results from mining the sample cannot be trusted may discourage the adversary from making the attempt.

These sample sizes tend to be small (10s or 100s of tuples). However, for certain purposes this is reasonable. For example, providing samples of actual data to be used for development of new systems to operate in a secured environment. These formulas give us the ability to state "this is a safe amount of data to release", without worrying about the *specific* inferences that may be drawn. This is independent of the external knowledge available to the adversary (except for the database contents not included in the sample).

Another thing we gain is the ability to analyze the effect of a given sample size. This is useful when data is released unintentionally; we can analyze the potential impact of the release both in terms of the direct inferences that can be made, and the "probabilistic inferences" that can be determined by data mining. Again, this is independent of the technology or external knowledge available to the adversary.

There are various ways we can extend this work. One is in the realm of support to a data security administrator; an "operations manual" for determining how much data to release. The primary effort required is determining appropriate parameters for a classifier. One solution would be to use clustering techniques on the database (e.g., self-organizing maps [Kohonen, 1990] with thresholds on nearest neighbor) to give a likely value for the VC-dimension of a reasonable classifier. This idea is based on grouping the potential rule left-hand sides into similar groups, with the idea that similar inputs would likely lead to similar outputs. A classifier on extremely diverse data is likely to be more complex than one on simple data. This needs more work to establish limits on the probabilities involved.

Another area for extension is that this work assumes the sample obtained by the adversary is randomly distributed. However, many applications will produce non-random samples. An example of this would be a system that allows queries to a database (access to individual information), but tries to protect correlations among items through limiting the volume of data available. In such a system *the adversary controls the sample distribution*. What can we say about such an environment?

There are a number of possibilities:

- The sample is random with respect to a correlation discovered. In this case, the fact that the sample is not random with respect to *some* criteria is irrelevant.
- A discovered correlation involves the selection criteria. The problem is that we cannot say if the correlation is *unique* to the selection criteria: It may or may not be independent of the selection criteria.
- A correlation exists between the selection criteria and some other field in the data. The previous case prevents our discovering this correlation, however does the non-randomness of the sample allow us to discover

other correlations between the "other field" and other items? Or does this reduce to the previous case?

- The adversary has multiple samples based on different selection criteria. One obvious sub-case of this is a random sample and a non-random sample. Does this allow us to discover correlations with respect to the selection criteria that we would not expect to discover with a random sample?

This is related to work in privacy problems from data aggregation [Cox, 1996, Chowdhury et al., 1996]. The statistical database inference problem deals with identifying individual data values from one or more summary queries. It is the converse of the problem of this paper; instead of protecting against learning data items from aggregates, we are protecting against learning aggregates from individual items. Although the basic problem is quite different, as we move toward non-random samples the two areas may overlap. Of particular note is work on random sampling queries [Denning, 1980]; this may provide tools to implement policies governing the creation of non-random samples.

Another possible starting point for this is artificial intelligence work on selection of training data [Cohen et al., 1995, Yang and Honavar, 1998]. Preventing the adversary from selecting a "good" set of training data (while still allowing some queries, and those non-random release of data) would support this work.

What we have shown is that for reasonably small random samples, we can be confident that the threat posed by data mining is minor.

References

[Chowdhury et al., 1996] Chowdhury, S. D., Duncan, G. T., Krishnan, R., Roehrig, S., and Mukherjee, S. (1996). Logical vs. numerical inference on statistical databases. In *Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences*, pages 3–10.

[Cohen et al., 1995] Cohen, D. M., Kulikowski, C., and Berman, H. (1995). DEXTER: A system that experiments with choices of training data using expert knowledge in the domain of DNA hydration. *Machine Learning*, 21:81–101.

[Cox, 1996] Cox, L. H. (1996). Protecting confidentiality in small population health and environmental statistics. *Statistics in Medicine*, 15:1895–1905.

[Delugach and Hinke, 1996] Delugach, H. S. and Hinke, T. H. (1996). Wizard: A database inference analysis and detection system. *IEEE Transactions on Knowledge and Data Engineering*, 8(1).

[Denning, 1980] Denning, D. E. (1980). Secure statistical databases with random sample queries. *ACM Transactions on Database Systems*, 5(3):291–315.

[Devroye et al., 1996] Devroye, L., Györfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York.

[Devroye and Lugosi, 1995] Devroye, L. and Lugosi, G. (1995). Lower bounds in pattern recognition and learning. *Pattern Recognition*, 28:1011–1018.

[Hinke and Delugach, 1992] Hinke, T. H. and Delugach, H. S. (1992). Aerie: An inference modeling and detection approach for databases. In Thuraisingham, B. and Landwehr, C., editors, *Database Security, VI, Status and Prospects: Proceedings of the IFIP WG 11.3 Workshop on Database Security*, pages 179–193, Vancouver, Canada. IFIP, Elsevier Science Publishers B.V. (North-Holland).

[Hinke et al., 1997] Hinke, T. H., Delugach, H. S., and Wolf, R. P. (1997). Protecting databases from inference attacks. *Computers and Security*, 16(8):687–708.

[Johnsten and Raghavan, 1999] Johnsten, T. and Raghavan, V. (1999). Impact of decision-region based classification algorithms on database security. In *Proceedings of the Thirteenth Annual IFIP WG 11.3 Working Conference on Database Security*.

[Kohonen, 1990] Kohonen, T. (1990). The self organizing map. *IEEE Transactions on Computers*, 78(9):1464–1480.

[Vapnik, 1982] Vapnik, V. N. (1982). *Estimation of dependences based on empirical data*. Springer-Verlag, New York.

[Yang and Honavar, 1998] Yang, J. and Honavar, V. (1998). Feature subset selection using a genetic algorithm. *IEEE INTELLIGENT SYSTEMS*, 13(2):44–49.

[Yip and Levitt, 1998] Yip, R. and Levitt, K. (1998). The design and implementation of a data level database inference detection system. In *Proceedings of the Twelfth Annual IFIP WG 11.3 Working Conference on Database Security*.