CERIAS Tech Report 2001-79
Using sample size to limit exposure to data mining
by Christopher Clifton
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

To appear in Journal of Computer Security, IOS Press (invited paper).

Using Sample Size to Limit Exposure to Data Mining

Chris Clifton
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730-1420 USA
+1-781-271-2390, Fax: +1-781-271-2352, clifton@mitre.org

December 14, 2000

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 of the work in preventing inference in multi-level secure databases has concentrated on preventing such "provable" facts [DH96]. Recent work has extended this to capturing data-level, rather than schema-level, functional dependencies [YL98]. 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 the reach of 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 [HD92, HDW97] 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?[1] 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 are not strict, but have some "strength" measure (two common ones are *support*, the fraction of database tuples that are examples of the rule; and *confidence*, the ratio of the tuples that are examples of the rule to the number of tuples where the antecedent of the rule holds.) 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[Vap82] 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 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.

[1] In cases where we *know* the inference we must keep secret, like the example above, other techniques are available.[JR99, CM98, ABE+99]

1.1 What this work will lead to

The eventual goal of this work is to provide a tool usable by security administrators. This tool would enable the administrator to determine:

- The allowable size of a sample, given constraints on the quality of what an adversary would be allowed to learn; and
- The quality of what an adversary would be allowed to learn, given the size of a sample.

To make this a usable tool, we must be able to abstract away from technical details of learning/data mining algorithms. We view certain information as clearly reasonable for a security administrator to provide:

|D| The number of tuples in the database (or in the "world" about which we are concerned; if we are concerned about the application of discovered knowledge to the real world, and the database is already only a sample of the real world).

n The number of tuples in the sample (unless this is the goal of our calculation).

Attributes The number and type (e.g., continuous, discrete with k possible values) of attributes for the entities in the data.

Note that this assumes a "single table" view of the data; this matches current data mining technology. For a complete database, independent analyses can be performed for each table (or collection of tables that describe a single type of entity), and the notion of a *universal relation*[Osb79] can be used to analyze "global" relationships.

We need other information that is more difficult to provide, part of the focus of this work is defining this information in a form reasonable for security administrators.

Mining type To obtain tight bounds on error given a sample, we need to know the descriptive power of the adversary's data mining technology. This can be assumed to be "worst case", however if the security administrator knows the *type* of knowledge that needs to be protected against (e.g., inference rules that can classify the data into two groups) we can obtain better bounds.

Expected error To give a recommendation on sample size, we need to know the expected error we plan to force an adversary to live with. This can be difficult; for example for an inference rule that does binary classification, a statement like "The rule must be wrong 25% of the time" would be unreasonable if 90% of the cases belonged to one class. Better descriptions might be "There is a 25% expectation that the actual best rule will be better than the best rule found on the sample", or "The actual confidence of any rule can be expected to have 25% error".

Note that the means for describing expected error will be dependent on mining type; one possibility would be to have a security administrator provide error for one type and have the system determine expected error for other types of mining.

This paper will only provide rudimentary examples of how such a security administration tool would appear. The focus is on establishing sample size/error relationships for a simple class of problems (binary classification), and giving an example of how this may be extended. There is clearly substantial room for further work in this area.

1.2 Applications and problem extensions

Providing a random sample of data has limited utility. A few examples of where this *has* utility are:

- Development: We can provide a sample of real data to a system developer to use in design/testing. This allows the developer to do a better job of testing, without clearing them for access to the entire database. This requires clearing the data *items* and checking for functional-dependency type inferences, but these alone would not be sufficient. The techniques presented here will ensure that such a "sanitized sample" is unlikely to contain *hidden* rules.

- Damage assessment: If a portion of the database *is* released, we can analyze potential effects of that release.

Certain obvious uses of a sample, e.g., for statistical purposes, are unlikely to work on a sample small enough to prevent data mining. However, this is our *goal*: **preventing** an adversary from learning things that apply to the entire population.

However, certain extensions of this work could generate substantial additional utility. For example, we may want to give access to a subset of the database – a non-random sample. This clearly will allow facts to be learned with respect to that subset, but can we use this approach to state limits on what can be learned that is not specific to that subset? Alternatively, can we give query access, but "cut off" access before too much data is released. This requires tracking access over time, as a collection of small independent samples must be treated as a single large sample for our purposes. This is a problem faced with all inference protection mechanisms, Marks[Mar96] solves this for "normal" inference with a mechanism for tracking and limiting what a given user has seen over time.

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, let us assume that the complete order database is as presented in Table 1. If we were to look for a classification of *Item* in this table, we would 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 make a good guess that SB also has this common factor.

Note that if we only have a sample of the database, we may develop a different set of rules. Table 2 gives a database containing 70% of the complete database. The rules generated from this database are:

Table 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

Table 2: Sample of Base Order Database

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

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? It is easy to find samples that 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 will return to this issue in the conclusions.

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. It is our goal to show not that the rules obtained on the sample *are* bad, but that they are *likely* to be bad. The sample may or may not cause unreliable results. If the adversary knows this, the confidence in the rules produced (and in any knowledge gained using those rules) becomes suspect. All we have to do is lower the adversary's confidence in the results to the point where the adversary would view any information gained from data mining on the sample to not be worth the effort required to verify it. In other words, although any knowledge gained using the rules may be correct, it may also be incorrect and thus cannot be trusted. Thus the data is not useful *for learning rules*.

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[Vap82]. In Section 4, we will use an adaptation of Vapnik's work to show the difficulty of learning as a function of

sample size. In Section 4.2 we show how limitations on classification can cause variability in confidence and support expectations for inference rules.

To give an idea of how this would work in practice, we will use the results shown in the next two sections to demonstrate how large a sample would be reasonable for the above example. Assume the adversary wants to develop a classifier separating bases X and Y (known to support the SSA) from A and B. The adversary tries to do this based on the amount of fuel and food ordered each month for a year.

If 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:

$$January(Fuel, 2)\&January(Food, 0)\& \ldots \&December(Food, 1) \Rightarrow \text{Supports SSA}$$

stating that for a given base and year, there are two Fuel and no Food orders in January, etc. 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 10% of the time, we can allow a sample size of over 50 billion where a single sample is all of the orders for a given base for a year (based on Theorem 2; 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.

The problem for the adversary is that there are likely to be too many ways to classify any sample – we are assuming that the adversary will be looking for rules based on *exact numbers* of each type of order. 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:

1. fuel $\gg$ food, low total order for the year
2. fuel $\approx$ food, low total order
3. fuel $\ll$ food, low total order
4. fuel $\gg$ food, high total order
5. fuel $\approx$ food, high total order
6. fuel $\ll$ food, high total order

we can only allow a sample size of $(6-1)/(12*.4) = 1$ (again, a sample is complete information on a base for a year, or 24 tuples from a complete version of Table 1.) The problem is that if a perfect classifier exists, at least one rule will apply to the sample. We then learn that rule – and in this example, we can expect that rule to cover about 1/6 of the data on average, giving us a perfect classifier 1/6 of the time (and a guess the rest of the time.)

This assumes a perfect classifier exists, however with a simple classifier it is unlikely that it *can* perfectly classify the data. In this example, imagine that the given sample is the one sample that the best possible classifier gets wrong – thus the rule learned from that sample is wrong every time except on that sample. Thus we get a guess 5/6 of the time, and a *wrong answer* the other 1/6.

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.

2.1 Related Work

There is a substantial body of work in protecting against rule / inference learning. Much of this has come from the database security community.[DH96, HD92, HDW95, HDW97, YL98] This work has emphasized the problem of strict inferences – learning rules that *always* hold. This is a particular problem in the realm of multi-level secure databases, and much of the work has concentrated on problems in this domain.[Mar96] Data mining produces results that are often, but not always, true. This poses a new challenge.

Perhaps closer in spirit to the work presented here is in statistical inference: Preventing the discovery of individual data values given aggregates.[Den80, Cox96, CDK+96] This is the converse of the problem addressed here. We assume that the individual data values are non-sensitive; it is the aggregates and relationships that must not be discoverable.

Protecting against inference learning in the data mining sense has begun to receive attention. There has recently been work on protecting against the learning of *specific* rules.[JR99, CM98, ABE+99] These approaches start with a goal of protecting against learning a known set of rules, and determine minimal data that must be withheld to prevent learning such rules. These are appropriate for the specific example we have shown – where we know we want to protect the knowledge that Base SB supports the SSA. However, the work presented here drops the assumption that we know the knowledge we want to protect – what can we do if we don't even know what sensitive inferences the data might contain? This closes one end of the spectrum of security challenges posed by data mining.

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 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 primary difficulty is that there 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 with respect to the sample size.

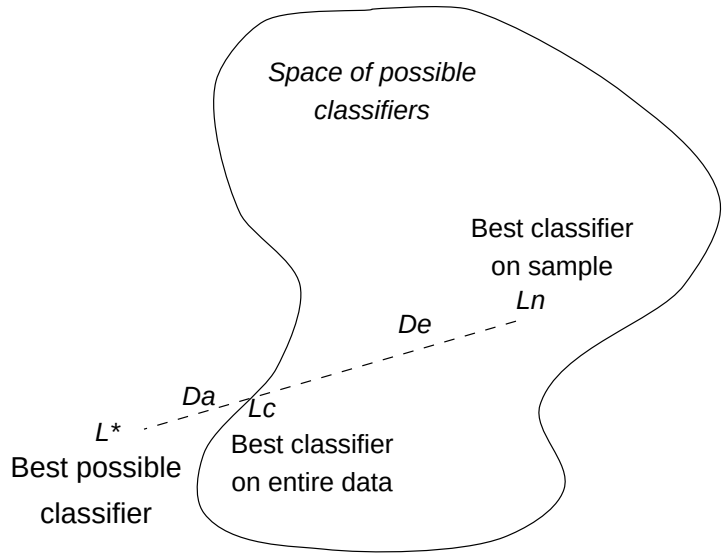

Figure 1: Distance between best classifier on sample and best classifier on data.

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.

Various theorems show that classifiers exist that will learn "perfectly" given a sufficiently large sample, i.e.,

$$\mathbf{E}L_n \to L^*.$$

However, the classifier chosen may be dependent on the data or on n, for example this holds for a nearest neighbor classifier if the number of classes $k \to \infty$ and $k/n \to 0$ ([Sto77]).

Note that this is heavily data dependent; the following theorem states that we can *always* find data where we will do poorly for any given sample size:

Theorem 1 *[DGL96]: Let $\epsilon > 0$ be an arbitrarily small number. For any integer n and classification rule g_n, there exists a distribution of (X, Y) with Bayes risk $L^* = 0$ such that*

$$\mathbf{E}L_n \geq 1/2 - \epsilon.$$

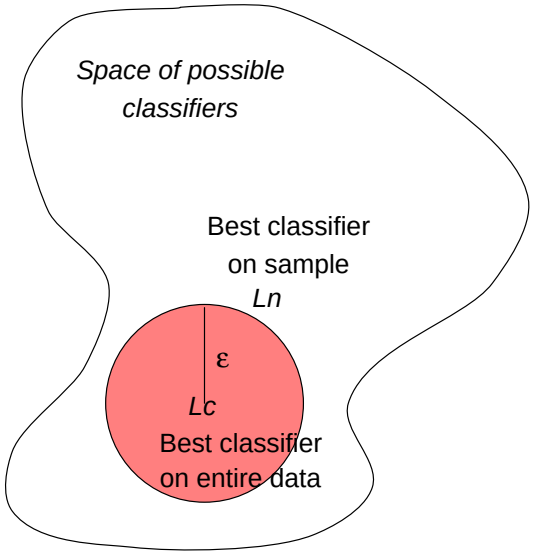

Figure 2: Error of best classifier on sample L_n worse than best classifier $L_{\mathcal{C}}$ by more than ϵ.

However, this only says a bad distribution exists; it *doesn't* say that it is likely in practice. What we need is somewhere in between – to say that given a sample size, we can expect a classifier to do poorly with high probability.

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 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 will give the "wrong" answer, even though a classifier $L_{\mathcal{C}}$ exists that would give 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. The probability that a classifier is outside the given bounds allows us to look at two issues:

1. The likelihood that a different outcome is better for the given left-hand side of the rule (the tuples in the sample that match the left-hand side are not representative of all tuples that match the left-hand side), and

2. The likelihood that a better rule exists for the same outcome; i.e., a better predictor for the same result (lack of a good rule for the given outcome, bases that support the SSA, is the benefit gained from a small sample in Section 2).

What we can show is the following: Given a "best error" that we are willing to 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? In this case, "best error" is a measure of the difference between the classifier (rule set) the adversary chooses, and the best possible classifier *of that type*. Note that this is dependent on the type of classifier; i.e., for a very simple classifier it is easy to get the "best" one (but the best one probably won't be very good). What we are bounding is the estimation error.

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

$$\sup_{(X,Y)\in\mathcal{X}} \mathbf{P}\{L_n - L_{\mathcal{C}} \geq \epsilon\} \leq \delta \tag{1}$$

where $L_{\mathcal{C}}$ is the probability that the best classifier in $\mathcal{C}$ (the one giving the smallest probability of error) will be wrong on any given trial:

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

and L_n is the probability that the best classifier on the training data g_n will be wrong on any given trial on the entire data:

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

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. Knowing this, we can fix ϵ (the difference between the error probability of the learned classifier L_n and the "best" classifier $L_{\mathcal{C}}$) to a suitably large figure (how badly we want the adversary to fail). δ is the probability that the adversary may do as badly as ϵ. We can then find a sample size n such that:

$$\sup_{(X,Y)} \mathbf{P}\{L_n - L_{\mathcal{C}} \geq \epsilon\} \geq \delta$$

The problem is that δ is a worst case of all possible distributions (more precisely, all possible distributions for which the best possible classifier has error $L_{\mathcal{C}}$). For example, if the

data distribution is such that the "real" best classifier is $g(x) = 1$ (e.g. all outputs are 1, regardless of input), the adversary will probably "guess" this with a single sample (provided the sample falls into class 1). However, we will show that there is *some* distribution that will be difficult for the adversary. 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.[2]

We now present an example showing how we can use sample size to control the expected approximation error. This is for a *binary classification* problem: We want a rule (or set of rules) that will divide the input into two classes. We are assuming that the adversaries goal is to learn such a set of rules; our goal is to limit the applicability of the entire set. Section 4.2 shows how this applies to evaluating a single rule, Section 4.3 extends this to non-binary classifiers.

We begin with two definitions of Vapnik-Chervonenkis theory[Vap82] (notation from [DGL96]): 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 $= 2^{number\ of\ rules}$.[3] 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. (Section 4.3 discusses extending this to non-binary decisions.) 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

[2]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.

[3]Actually $m^{number\ of\ rules}$, where m is the number of possible output values.

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 $(possible\ food\ orders\ per\ month * possible\ fuel\ orders\ per\ month)^{number\ of\ months} = 16777216$ 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 that gives us a sample size such that for a given classifier complexity (number of possible rules) V and error ϵ, there is a distribution such that for a sample of size $n < (V-1)/12\epsilon$, there is probability $\geq 1/10$ that the classifier will have error $\geq \epsilon$.

Theorem 2 *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 $\delta < 1/10$ and $\epsilon < 1/2$,*

$$N(\epsilon, \delta) > \frac{V-1}{12\epsilon}.^{4}$$

PROOF. *Follows that of [DGL96].* We will prove this by showing that a distribution exists where $\sup_{(X,Y)\in\mathcal{X}} \mathbf{P}\{L_n \geq \epsilon\} \geq \frac{1}{10}$ if $n \leq \frac{(V-1)}{12\epsilon}$. Since $L_{\mathcal{C}} = 0$ and $\delta < 1/10$, this distribution provides an example (choice of $(X, Y) \in (X)$) where the error probability exceeds the bounds in the definition of $N(\epsilon, \delta)$ (equation 1).

The idea is to construct a family $\mathcal{F}$ of 2^{V-1} "hard" distributions. These are built around a set of V points *shattered* by $\mathcal{C}$. (i.e., each point falls in a different possible set that can be partitioned by the classifier.) Each distribution in $\mathcal{F}$ is concentrated on the set of these points. A member in $\mathcal{F}$ is described by $V-1$ bits, $b_1, \ldots, b_{V-1}$. This is represented as a bit vector b.

Each bit vector corresponds to a possible classifier. The points $x_1, \ldots, x_V$ correspond to the inputs that can be distinguished by the available rules.

Assume $n \geq V-1$. For a particular bit vector, we let each

$$\begin{array}{ll} X = x_i\ (i < V) & \text{with probability } \frac{1}{n}, \\ X = x_V & \text{with probability } 1 - \frac{1}{n}(V-1). \end{array} \tag{2}$$

(Note: This defines a worst-case distribution.)

Then set $Y = f_b(X)$, where f_b is defined as:

$$f_b(x) = \begin{cases} b_i & \text{if } x = x_i, i < V \\ 0 & \text{if } x = x_V \end{cases}. \tag{3}$$

Since Y is a function of X, $L^* = 0$. Since the set $\{x_1, \ldots, x_V\}$ is *shattered* by $\mathcal{C}$, there is a different classifier $g \in \mathcal{C}$ (corresponding to f_b) for every possible combination of outputs for the input $\{x_i\}$, so $L_{\mathcal{C}} = 0$ for our family of distributions.

For a fixed b, the error probability is

$$L_n(b) = \mathbf{P}\{g_n(X, X_1, f_b(X_1), \ldots, X_n, f_b(X_n)) \neq f_b(X) | X_1, \ldots, X_n\}.$$

[4] A tighter bound of $\frac{1}{8\epsilon}\log\left(\frac{1}{\delta}\right)$ exists for some values of ϵ and δ. However, for meaningful probability of error (e.g. $\delta \geq 0.01$), this bound is not as tight, and will not be discussed here.

We can now bound the error as follows:

$$\begin{aligned}
\sup_{(X,Y):L_{\mathcal{C}}=0} \mathbf{P}\{L_n \geq \epsilon\} &\geq \sup_{(X,Y)\in\mathcal{F}} \mathbf{P}\{L_n \geq \epsilon\} \\
&= \sup_b \mathbf{P}\{L_n(b) \geq \epsilon\} \\
&\geq \mathbf{E}\{\mathbf{P}\{L_n(B) \geq \epsilon|B\}\} \\
&\quad \text{(where } b \text{ is replaced by } B\text{, uniformly distributed over } \{0,1\}^{V-1}\text{)} \\
&= \mathbf{P}\{L_n\{B\} \geq \epsilon\}.
\end{aligned}$$

But $L_n(b) = \mathbf{P}\{g_n(X, X_1, f_b(X_1), \ldots, X_n, f_b(X_n)) \neq f_b(X)|X_1, \ldots, X_n\}$ can be viewed as the error probability of a decision function $g_n : \Re^d \times (\Re \times 0, 1)^n \to \{0, 1\}$ in predicting the value of $f_B(X)$ based on the observation $Z_n = (X, X_1, Y_1, \ldots, X_n, Y_n)$. This is bounded from below by the Bayes error

$$L^*(Z_n, f_B(X)) = \inf_{g_n} \mathbf{P}\{g_n(Z_n) \neq f_B(X)\}$$

However, the Bayes error can also be expressed as:

$$L^*(Z_n, f_B(X)) = \mathbf{E}\{\min(\eta^*(Z_n), 1 - \eta^*(Z_n))\}$$

where $\eta^*(Z_n) = \mathbf{P}\{f_B(X) = 1|Z_n\}$ (the a-posteriori probability of the result being 1.)

Since

$$\eta^*(Z_n) = \begin{cases} 1/2 & \text{if } X \neq X_1, \ldots, X \neq X_n \text{ and } X \neq x_V \\ 0 \text{ or } 1 & \text{otherwise (the training data has a sample of the class).} \end{cases}.$$

we can show

$$\begin{aligned}
L_n(B) &\geq \mathbf{E}\{\min(\eta^*(Z_n), 1 - \eta^*(Z_n))|X_1, \ldots, X_n\} \\
&= \frac{1}{2}\mathbf{P}\{X \neq X_1, \ldots, X \neq X_n, X \neq x_V|X_1, \ldots, X_n\} \\
&= \frac{1}{2n}\sum_{i=1}^{V} \mathbf{P}\{x_i \neq X_1, \ldots, x_i \neq X_n\} \\
&\geq \frac{1}{2}\sum_{i=1}^{V-1} \mathbf{P}\{x_i \neq X_1, \ldots, x_i \neq X_n\}
\end{aligned}$$

The probability in the final sum is simply the probability that the given x_i is not represented in the training data. This can be viewed as $1/n$ times the number of cells not represented in the training data. For fixed $X_1, \cdots, X_n$, we denote by J the collection $\{j : 1 \leq j \leq V - 1, \bigcap_{i=1}^n \{X_i \neq x_j\}\}$ of empty cells x_i. This gives us:

$$\begin{aligned}
\sup_{(X,Y):L_{\mathcal{C}}=0} \mathbf{P}\{L_n \geq \epsilon\} &\geq \mathbf{P}\left\{\frac{1}{2}\sum_{i=1}^{V-1} \mathbf{P}\{x_i \neq X_1, \ldots, x_i \neq X_n\} \geq \epsilon\right\} \\
&= \mathbf{P}\left\{\frac{1}{2n}|J| \geq \epsilon\right\} \\
&= \mathbf{P}\{|J| \geq 2n\epsilon\}.
\end{aligned}$$

Assume $12n\epsilon \leq V-1$, $\epsilon < 1/2$. $\mathbf{E}|J| = (V-1)(1-1/n)^n > (V-1)/3$ for $n \geq 6$. Also, since $0 \leq |J| \leq V-1$, we have $\mathbf{Var}|J| \leq (V-1)^2/4$. By the Chebyshev-Cantelli inequality,

$$\begin{aligned}
\mathbf{P}\{|J| \geq 2n\epsilon\} &= (1 - \mathbf{P}\{|J| < 2n\epsilon\}) \\
&\geq (1 - \mathbf{P}\{|J| < (V-1)/6\}) \\
&= (1 - \mathbf{P}\{|J| - \mathbf{E}|J| < -(V-1)/6\}) \\
&= (1 - \mathbf{P}\{|J| - \mathbf{E}|J| > (V-1)/6\}) \\
&\geq \left(1 - \frac{\mathbf{Var}|J|}{\mathbf{Var}|J| + (V-1)^2/36}\right) \\
&\geq \left(1 - \frac{(V-1)^2/4}{(V-1)^2/4 + (V-1)^2/36}\right) \\
&= \frac{1}{10}.
\end{aligned}$$

This completes the proof.

Equation 2 define the worst-case distribution (dependent on n and V.) For $n = V$, this is a uniform distribution across the input categories that could possibly be distinguished by V. As V grows relative to n, the worst-case distribution becomes skewed towards a single input category, with a uniform distribution across the others. Neither are (within limits) unreasonable distributions in practice.

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 doesn'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) go up, the sample size needed does as well.

Figure 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 3 *[DL95, DGL96]. 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 $X_1, Y_1, \ldots, X_n, Y_n$,

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

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

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

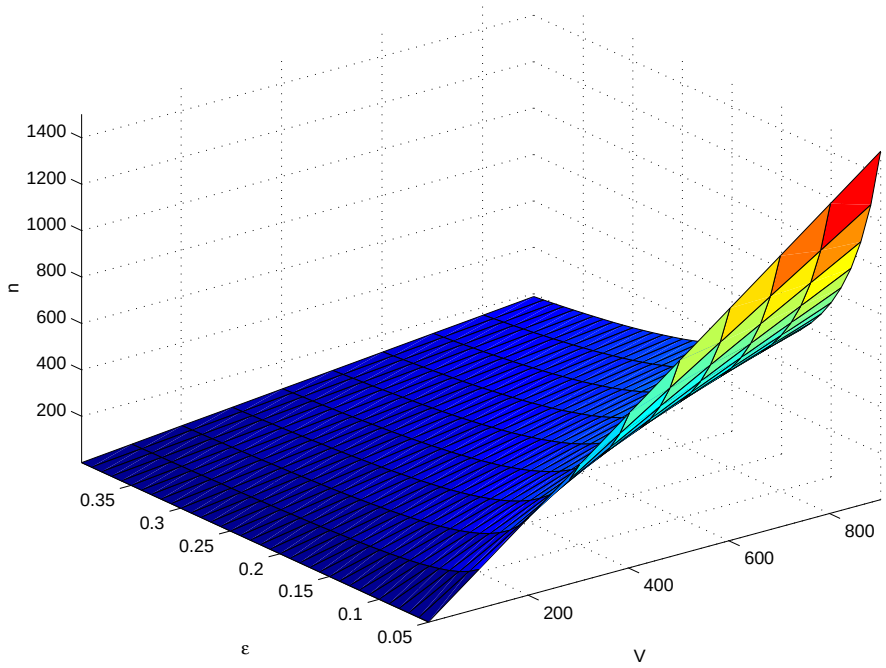

Figure 3: Maximum value of n where probability $\delta < 1/10$ that there is error ϵ, as function of ϵ and classifier complexity V, when a perfect classifier is available.

The proof is similar to that of Theorem 2. For details, readers are referred to [DGL96]. The fundamental difference is that the family of distributions (Equation 2) is constructed differently. Rather than letting $X = x_i$ with probability $1/n$ for $x_1 \ldots x_{V-1}$, let $X = x_i$ with probability p.

The output Y (Equation 3) is also defined differently, as Y no longer need be a function of X (since $L_{\mathcal{C}} \neq 0$.) Instead, U is a uniform $[0, 1]$ random variable independent of X, and

$$Y = \begin{cases} 1 & \text{if } U \leq \frac{1}{2} - \left(\frac{1}{2} - \frac{L}{(V-1)p}\right) + 2\left(\frac{1}{2} - \frac{L}{(V-1)p}\right) b_i, X = x_i, i < V \\ 0 & \text{otherwise} \end{cases}.$$

The proof proceeds similarly to that of Theorem 2, except that p is chosen as a function of L and V to obtain the desired bounds. The most important aspect for our purposes is the nature of these distributions. For the first bound, $p \approx \frac{3L}{V-1}$. For the second bound, $p \approx \frac{2L}{(V-1)(1-\frac{\epsilon}{L+\epsilon})}$, which is between $\frac{2L}{V-1}$ and $\frac{4L}{V-1}$ (since $L \geq \epsilon$.)

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. The second is more useful for our purposes, as it is independent of the complexity of the classifier (except that L, the approximation error, *does* depend on the classifier used.)

Some sample values for the first, based on $\epsilon = \delta = .1$ (10% probability of being wrong at least 10% of the time) are given in Figure 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.

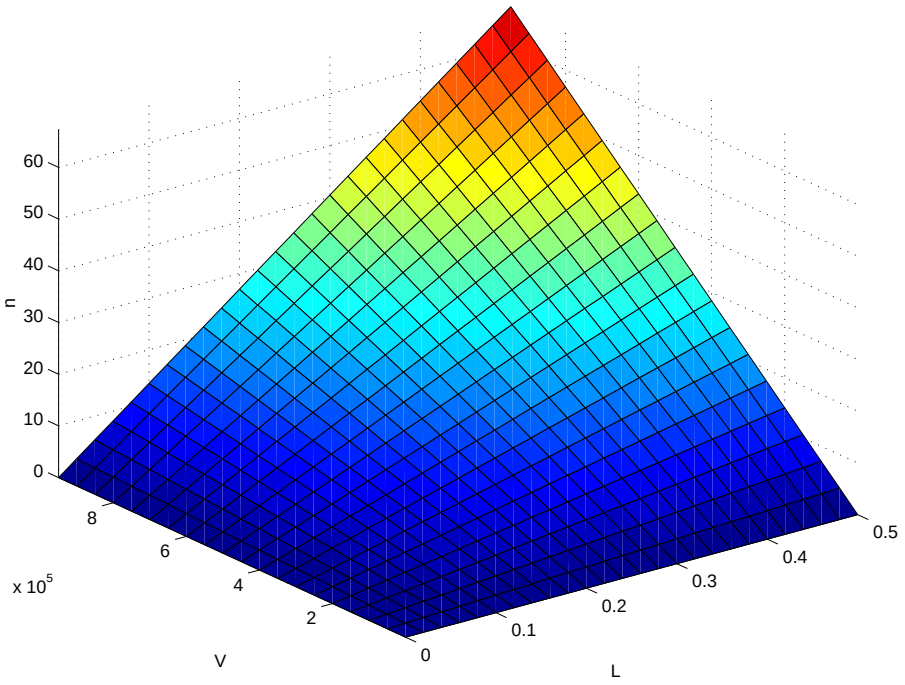

Figure 4: Value of n below which error $\epsilon > .1$ with probability $\delta > .1$ as a function of V and L.

The second formula is intuitively based on guessing the wrong outcome for a given rule. Sample values are given in Figures 5 ($\epsilon \geq 0.1$) and 6 ($\epsilon \geq 0.05$).

Note that all of these are rather 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 2. 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 $N(.4, .1) = (9^{12} - 1)/(12 * .4) > 50$ billion.

However, the simple classifier had a VC-dimension of 6. This gives a sample size of 1 if a perfect classifier exists. Intuitively, we will learn one rule correctly based on the sample, and that will cover on average 1/6 of all cases (for the others, we just guess.) However, it is unlikely that a perfect classifier can be built using such a small set of rules. Theorem 3 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 an allowed sample of 3 years of data.

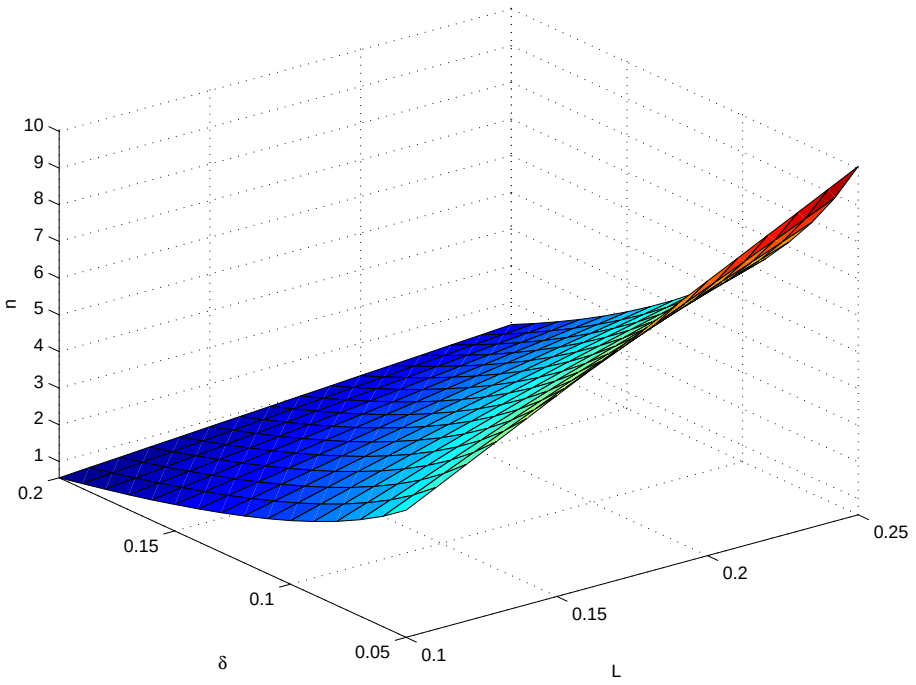

Figure 5: Value of n below which a guarantee of error within 0.1 impossible as a function of δ and L.

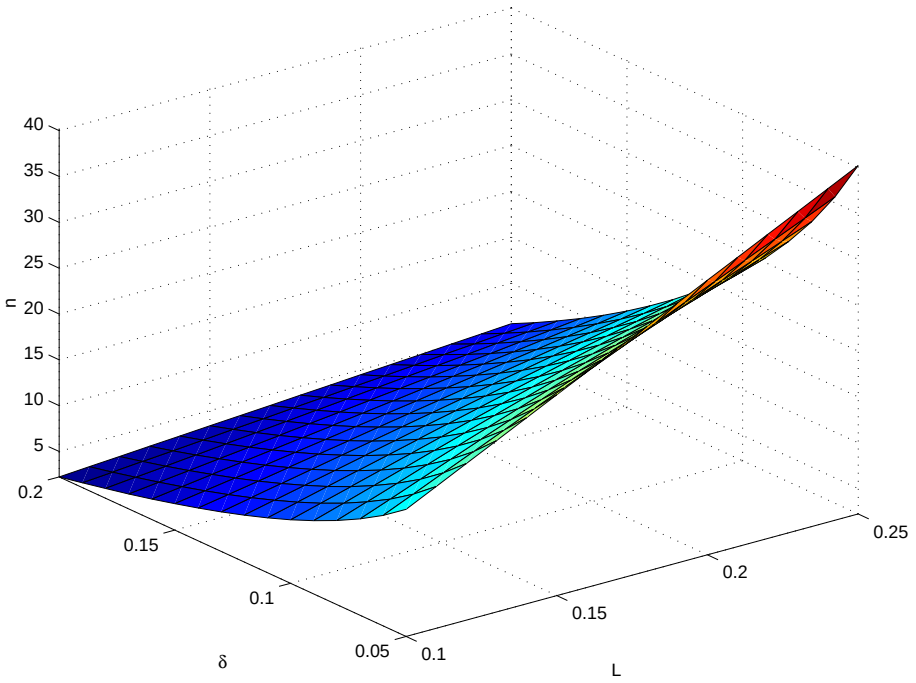

Figure 6: Value of n below which a guarantee of error within 0.05 impossible as a function of δ and L.

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.

4.2.1 Expected deviation of sample

Note that we can do better if we know what we want to protect [JR99, ABE+99]. Here we show how to evaluate expected rule performance based on the expected deviation of the sample from the real data. Given a single category rule, e.g. a rule of the form

$$Ph.D.\ and\ Comp.Sci. \rightarrow Professor$$

our goal is to bound the error estimate for such a single rule; i.e., how far off is the confidence of the rule likely to be. The problem is, this only works where we know what class we want to protect.

First, we can state that such a rule isn't meaningful unless it adds information; for example if the confidence of the above rule is 50%, but 50% of all people are professors, it is a useless rule. Thus if we can say that the estimate of total number of occurrences of the right side may be off, we can limit the "value added" of the rule. This is straightforward – what is the expected variance between the number of occurrences of the right side in n samples and the entire database?

This is a "sampling without replacement" problem. Assume there are a professors in the database, and b non-professors, and our sample consists of a choice of n tuples. It is easy to see that the mean number μ of professor tuples in a sample is n times the probability $p = \frac{a}{a+b}$ that a randomly chosen tuple will be a professor tuple.

The variance gives a measure of how far we can expect a sample to be from this mean: $\mathbf{E}\{(X-\mu)^2\}$. For this example, the variance is

$$np(1-p) \cdot \frac{a+b-n}{a+b-1}.$$

If we assume that the database is large relative to the sample, the last term ($\frac{a+b-n}{a+b-1}$) becomes insignificant, and the variance can be approximated by $np(1-p)$.

Our real concern is in the *prediction* of p from the sample. The best guess for p, from the sample, would be the observed mean divided by the sample size n. Thus, what we desire is the probability that the difference between the observed mean and the actual mean is greater than some constant:

$$\mathbf{P}\left\{\left|\frac{X}{n} - \frac{\mu}{n}\right| \geq c\right\}$$

The difficulty is that this probability is highly dependent on the distribution. What we *can* do is provide an *upper* bound on the probability, in other words, we can estimate the confidence the adversary will have in the observed probability. This uses Chebyshev's Inequality:

Theorem 4 *Chebyshev's Inequality: Let X be a random variable. Then for each $t > 0$,*

$$\mathbf{P}\{|X - \mathbf{E}\{X\}| \geq t\} \leq \frac{\mathbf{Var}\{X\}}{t^2}$$

What we want is:

$$\mathbf{P}\left\{\frac{|X - \mu|}{n} \geq c\right\} = \mathbf{P}\left\{|X - \mu| \geq nc\right\}$$

Thus, once we have picked the sample size n, we can choose $t = nc$ and apply Chebyshev's Inequality, giving:

$$\mathbf{P}\left\{\frac{|X - \mu|}{n} \geq c\right\} \leq \frac{\mathbf{Var}\{X\}}{(nc)^2}$$

Thus, for our example, we can say that the expected deviation of our probability estimate

$$\mathbf{P}\left\{\left|\frac{X}{n} - \frac{\mu}{n}\right| \geq c\right\} \leq \frac{p(1-p)}{nc^2}$$

This gives a reasonable estimate of how confident the adversary will be in the result.

4.2.2 Expected deviation in confidence

Second, what can we say about the confidence of the rule? Here we are interested in the expected number of cases where the rule is true vs. false. This is identical to the above, except that we ignore the cases where the left side doesn't hold (thus requiring a larger sample n).

What we really have is the difference between the expected value of the variance of the full association. For example, $AB \rightarrow C$: we have a set of cases where AB holds. Of these, we have some where C holds. The amount this varies (in the sample) from the true expectation of C holding is our goal.

4.3 Non-binary outputs

Another interesting situation is what happens with multiple output categories. So far we have only discussed binary classifiers; what if their are more than two possible outcomes?

A simple way to model k categories is as $\log_2(k)$ binary classifiers (assuming the output categories are independent). The "combined" classifier will fail if *any* of the binary classifiers fail. Thus we can model the probability as

$$\begin{aligned}
\mathbf{P}\{k\} &= \mathbf{P}\{k_1, \ldots, k_{\log_2(k)}\} \\
&= \mathbf{P}\{k_1\}\mathbf{P}\{k_2\} \cdots \mathbf{P}\{k_{\log_2(k)}\} \\
&\quad \text{(as the "bits" in the outcome are independent).} \\
&= \mathbf{P}\{\text{a binary classifier being correct}\}^{\log_2(k)}
\end{aligned}$$

This is the probability of getting the result *correct*; or $1 - \mathbf{P}\{error\}$. Thus, if the expected error between the classifier (or rule) learned on the sample and the correct rule is .1, the chance of getting a rule right with 16 output categories is $\approx .66$. This is the probability of getting the same result as the best available classifier, which is also likely to have a larger error than in the binary case.

5 Unintentionally Released Data

A related problem is what happens if a sample is released? In other words, what if we know the sample size n? In this case, the goal is to determine "how bad" the loss is. What we can do is state limits on the probability that the error is within a given bound. In other words, we want to determine how confident the adversary can be in any result mined from the data.

This is similar to the results in the preceding section. Formally, the problem is to find lower bounds for

$$\sup \mathbf{E}L_n - L_{\mathcal{C}}.$$

What this states is that there is a distribution of the data where the adversary can expect they will be off by the given amount with a sample of size n randomly chosen over that distribution.

The following theorem gives us a way to make use of this information:

Theorem 5 *[VC74]: Let $\mathcal{C}$ be a class of discrimination functions with* VC *dimension V. Let $\mathcal{X}$ be the set of all random variables (X, Y) for which $L_{\mathcal{C}} = 0$. Then, for every discrimination rule g_n based upon $X_1, Y_1, \ldots, X_n, Y_n$, and $n \geq V - 1$,*

$$\sup_{(X,Y)\in\mathcal{X}} \mathbf{E}L_n \geq \frac{V-1}{2en}\left(1 - \frac{1}{n}\right).$$

This says that if a perfect classifier exists, there is a distribution of the data such that any classifier learned from a random sample of the data will have at least expected error $\mathbf{E}L_n$, where this value is dependent on the VC-dimension V and the sample size n.

Since this function is continually decreasing for $n > 2$, it gives the maximum at the lower limit where the theorem applies: $n = V - 1$. At this point the upper bound is

$$\frac{1}{2e}(1 - \frac{1}{V}) \geq 0.18 \text{ when } V > 50.$$

This means that given a sample of size $n \leq V - 1$, it is possible that any classifier learned from the sample will be wrong 18% of the time (there exists a distribution of the data such that this will hold).

There are similar theorems for the case $L_{\mathcal{C}} > 0$, but they only apply for large n, where the expected error is so small as to be nearly useless (less than 1%). Theorem 3 provides the best approach to understanding potential problems with unintentionally released data in this case.

6 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. The number of possible rules that can be discovered is limited by the number of tuples in the sample. This gives an expected upper limit on the adversary's classifier complexity, but will give a high approximation error. Alternatively, if the classifier complexity is based on the *real* number of possible rules, the estimation error will be a better estimate of the total error. This is an area for further study.

One solution would be to use clustering techniques on the database (e.g., self-organizing maps [Koh90] 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 tuples 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 tuples? 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? As a worst case, this would give us the effect of a sample size as large as the size of a random sample required to give all the selected tuples. As a best case, this would appear as a random sample. The actual bound is probably somewhere in the middle – this needs to be worked out.

This is related to work in privacy problems from data aggregation [Cox96, CDK+96]. The statistical database inference problem deals with identifying individual data values from one or more summary queries. In a sense it is the converse of the problem of this paper; instead of protecting against learning tuple values from aggregates, we are protecting against learning aggregates from individual tuples. 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 [Den80]; 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 [CKB95, YH98]. 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.

Understanding how non-random samples affect learning also provides another possibility: deliberately "skewing" distribution of a sample to lessen the reliability in what is learned. This can be highly effective when the rule to be protected against is known, and may preserve benign rules. However, when we *don't know* what we want to protect, the benefits skewing the distribution are not as well understood. Further work is needed to show how skewing the distribution can be used to change the expected confidence of learning rules when the rule to be learned is unknown, and the type/amount of skew isn't known to the adversary.

Another area is the effect on correlations, rather than inference rules. The example in Section 2 was based on developing a *classifier* to predict bases supporting the SSA. Alternatively, the correlation between "lots of fuel" and "SSA" may be of interest. The problem is similar, however understanding the effect of small samples on the significance of such correlations (e.g., chi-squared measures) is still open.

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

[ABE+99] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules. In *Knowledge and Data Engineering Exchange Workshop (KDEX'99)*, pages 25–32, Chicago, Illinois, November 8 1999.

[CDK+96] Sumit Dutta Chowdhury, George T. Duncan, Ramayya Krishnan, Stephen Roehrig, and Sumitra Mukherjee. Logical vs. numerical inference on statistical databases. In *Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences*, pages 3–10, January 3–6 1996.

[CKB95] Dawn M. Cohen, Casimir Kulikowski, and Helen Berman. DEXTER: A system that experiments with choices of training data using expert knowledge in the domain of DNA hydration. *Machine Learning*, 21:81–101, 1995.

[CM98] LiWu Chang and Ira S. Moskowitz. Bayesian methods to the database inference problem. In *Proceedings of the Twelfth Annual IFIP WG 11.3 Working Conference on Database Security*, Chalkidiki, Greece, July 15–17 1998.

[Cox96] Lawrence H. Cox. Protecting confidentiality in small population health and environmental statistics. *Statistics in Medicine*, 15:1895–1905, 1996.

[Den80] Dorothy E. Denning. Secure statistical databases with random sample queries. *ACM Transactions on Database Systems*, 5(3):291–315, September 1980.

[DGL96] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, 1996.

[DH96] Harry S. Delugach and Thomas H. Hinke. Wizard: A database inference analysis and detection system. *IEEE Transactions on Knowledge and Data Engineering*, 8(1), February 1996.

[DL95] Luc Devroye and Gábor Lugosi. Lower bounds in pattern recognition and learning. *Pattern Recognition*, 28:1011–1018, 1995.

[HD92] Thomas H. Hinke and Harry S. Delugach. Aerie: An inference modeling and detection approach for databases. In Bhavani Thuraisingham and Carl Landwehr, editors, *Database Security, VI, Status and Prospects: Proceedings of the IFIP WG 11.3 Workshop on Database Security*, pages 179–193, Vancouver, Canada, August 19-21 1992. IFIP, Elsevier Science Publishers B.V. (North-Holland).

[HDW95] Thomas H. Hinke, Harry S. Delugach, and Randall Wolf. A framework for inference-directed data mining. In *Proceedings of the Tenth Annual IFIP WG 11.3 Working Conference on Database Security*, pages 187–198, Como, Italy, July 22-24 1995.

[HDW97] Thomas H. Hinke, Harry S. Delugach, and Randall P. Wolf. Protecting databases from inference attacks. *Computers and Security*, 16(8):687–708, 1997.

[JR99] Tom Johnsten and Vijay Raghavan. 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*, Seattle, Washington, July 26–28 1999.

[Koh90] Teuvo Kohonen. The self organizing map. *IEEE Transactions on Computers*, 78(9):1464–1480, 1990.

[Mar96] Donald G. Marks. Inference in MLS database systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(1), February 1996.

[Osb79] Sylvia L. Osborn. Towards a universal relation interface. In *Fifth International Conference on Very Large Data Bases*, 1979.

[Sto77] C. Stone. Consistent nonparametric regression. *Annals of Statistics*, (8):1348–1360, 1977.

[Vap82] Vladimir Naumovich Vapnik. *Estimation of dependences based on empirical data*. Springer-Verlag, New York, 1982.

[VC74] V. Vapnik and A. Chervonenkis. Theory of pattern recognition, 1974. (in Russian).

[YH98] Jihoon Yang and Vasant Honavar. Feature subset selection using a genetic algorithm. *IEEE INTELLIGENT SYSTEMS*, 13(2):44–49, March/April 1998.

[YL98] R. Yip and K. Levitt. 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*, Chalkidiki, Greece, July 15–17 1998.