Data Mining and Knowledge Discovery 1, 291–316 (1997)

Adaptive Fraud Detection

TOM FAWCETT fawcett@nynexst.com
FOSTER PROVOST foster@nynexst.com
NYNEX Science and Technology, 400 Westchester Avenue, White Plains, New York 10604

Editor: Usama Fayyad

Received April 2, 1997; Revised July 10, 1997; Accepted July 11, 1997

Abstract. One method for detecting fraud is to check for suspicious changes in user behavior. This paper describes the automatic design of user profiling methods for the purpose of fraud detection, using a series of data mining techniques. Specifically, we use a rule-learning program to uncover indicators of fraudulent behavior from a large database of customer transactions. Then the indicators are used to create a set of monitors, which profile legitimate customer behavior and indicate anomalies. Finally, the outputs of the monitors are used as features in a system that learns to combine evidence to generate high-confidence alarms. The system has been applied to the problem of detecting cellular cloning fraud based on a database of call records. Experiments indicate that this automatic approach performs better than hand-crafted methods for detecting fraud. Furthermore, this approach can adapt to the changing conditions typical of fraud detection environments.

Keywords: fraud detection, rule learning, profiling, constructive induction, intrusion detection, applications

1. Introduction

In the United States, cellular fraud costs the telecommunications industry hundreds of millions of dollars per year (Walters and Wilkinson, 1994; Steward, 1997). One kind of cellular fraud called *cloning* is particularly expensive and epidemic in major cities throughout the United States. Cloning fraud causes great inconvenience to customers and great expense to cellular service providers. Existing methods for detecting cloning fraud are *ad hoc* and their evaluation is virtually nonexistent. We have embarked on a program of systematic analysis of cellular call data for the purpose of designing and evaluating methods for detecting fraudulent behavior.

Cloning fraud is one instance of *superimposition fraud*, in which fraudulent usage is superimposed upon (added to) the legitimate usage of an account. Other examples are credit card fraud, calling card fraud and some forms of computer intrusion. Superimposition fraud typically occurs when a non-legitimate user gains illicit access to the account or service of a legitimate user. Superimposition fraud is detectable if the legitimate users have fairly regular behavior that is generally distinguishable from the fraudulent behavior.

This paper presents a framework, and a corresponding system, for automatically generating detectors for superimposition fraud. We have applied the system in the domain of cellular cloning fraud. Under the framework, massive amounts of cellular call data are analyzed in order to determine general patterns of fraud. These patterns are then used

to generate a set of monitors, each of which watches customers' behavior with respect to one discovered pattern. A monitor profiles each customer's typical behavior and, in use, measures the extent to which current behavior is abnormal with respect to the monitor's particular pattern. Each monitor's output is provided to a neural network, which weights the values and issues an alarm when the combined evidence for fraud is strong enough.

This article is organized as follows. We first describe the problem of cellular cloning fraud and some existing strategies for detecting it. We then describe the framework in detail using examples from the implemented system. We present experimental results comparing the system against other known methods for detecting fraud. Finally, we discuss the evaluation and describe issues in the future of automatic fraud detection.

2. Cellular communications and cloning fraud

Whenever a cellular phone is on, it periodically transmits two unique identification numbers: its *Mobile Identification Number* (MIN) and its *Electronic Serial Number* (ESN). These two numbers together specify the customer's account. These numbers are broadcast unencrypted over the airwaves, and they can can be received, decoded and stored using special equipment that is relatively inexpensive.

2.1. Cloning fraud

Cloning occurs when a customer's MIN and ESN are programmed into a cellular telephone not belonging to the customer. When this second telephone is used, the network sees the customer's MIN and ESN and subsequently bills the usage to the customer. With the stolen MIN and ESN, a cloned phone user (whom we shall call a *bandit*) can make virtually unlimited calls, whose charges are billed to the customer. The attraction of free and untraceable communication makes cloned phones very popular in major metropolitan areas.

If the fraudulent usage goes undetected, the customer's next bill will include the corresponding charges. Typically, the customer then calls the cellular service provider (the *carrier*) and denies the usage. The carrier and customer then determine which calls were made by the "bandit" and which were legitimate calls. The fraudulent charges are credited to the customer's account, and measures are taken to prohibit further fraudulent charges. In certain cases, the fraudulent call records will be referred to a law enforcement agency for prosecution.

There are two primary motivations for cloning fraud. Obviously, cloning fraud allows *low-cost* communications. A bandit is not charged for calls, which are usually worth far more (in retail dollars) than the cost of the cloned phone. Less obviously, cloning fraud allows *untraceable* communications because the bandit's identity cannot be tied to the cloned account. This second aspect is very important to criminals (DeMaria and Gidari, 1996).

Cloning fraud is detrimental in many ways. First, fraudulent usage congests cell sites, causing service to be denied to legitimate customers. Second, most cellular calls are to non-cellular destinations, so fraud incurs land-line usage charges. Third, cellular carriers must pay costs to other carriers for usage outside the home territory. Because these are

retail costs, they constitute a considerable financial burden to the customer's carrier. Fourth, the crediting process is costly to the carrier and inconvenient to the customer; the customer is more likely to switch to another carrier ("customer churn") if the other is perceived to be less susceptible to fraud. For these reasons, cellular carriers have a strong interest in reducing cloning fraud.

2.2. Strategies for dealing with cloning fraud

There are two classes of methods for dealing with cloning fraud. *Pre-call* methods try to identify and block fraudulent calls as they are made. *Post-call* methods try to identify fraud that has already occurred on an account so that further fraudulent usage can be blocked.

2.2.1. Pre-call methods. Pre-call detection methods involve validating the phone or its user when a call is placed. A common method is requiring that a Personal Identification Number (PIN) be entered before every call. A PIN serves as a password that is validated by the switch prior to allowing the call into the network. PINs are in use throughout many metropolitan areas in the United States. Unfortunately, like MIN-ESN pairs, PINs are broadcast over the airwaves unencrypted. For technical reasons, PINs are more difficult to receive and decode, but with more sophisticated equipment PIN cracking is possible (Herzog, 1995). Although PINs make cloning fraud more difficult, they do not prevent it.

Other methods of prevention include RF Fingerprinting and Authentication (Redden, 1996). RF Fingerprinting is a method of identifying cellular phones by their transmission characteristics. Authentication is a reliable and secure private-key encryption method that imposes no inconvenience on the customer. It has been predicted that authentication will eliminate cloning fraud eventually. However, authentication requires changes in hardware: both phones and switches must be capable of processing authentication requests. Currently about thirty million non-authenticatable cell phones are in use in the United States alone, and their replacement will not be immediate (Steward, 1997). In the meantime, cloning fraud will continue to be a problem and the industry will rely on *post-call* fraud detection methods.

2.2.2. Post-call methods. Post-call methods periodically analyze call data on each account to determine whether cloning fraud has occurred. One such method, *collision detection*, involves analyzing call data for temporally overlapping calls. Since a MIN-ESN pair is licensed to only one legitimate user, any simultaneous usage is probably fraudulent. A closely related method, *velocity checking* (Davis and Goyal, 1993), involves analyzing the locations and times of consecutive calls to determine whether a single user could have placed them while traveling at reasonable speeds. For example, if a call is made in Los Angeles 20 minutes after a call is made on the same account in New York, two different people are likely using the account.

Collisions and velocity checks are both believed to be accurate, but they share the disadvantage that their usefulness depends upon a moderate level of legitimate activity. Low-usage subscribers (for example, people who only use cellular phones in emergencies) will rarely cause collisions or velocity alarms with bandits.

Another post-call method, *dialed digit analysis*, mines call data to build up a database of telephone numbers called by bandits during periods of fraudulent activity. For detection, this database is matched against the numbers called by customers, and alarms are produced when the number of hits is above a threshold ("dialed digit hits").

2.2.3. User profiling. User profiling methods constitute a special class of post-call methods. They involve analyzing calling behavior in order to detect usage anomalies suggestive of fraud. Profiling often works well with low-usage subscribers because unusual behavior is very prominent. For this reason, profiling is a good complement to collision and velocity checking because it covers cases the others might miss.

Figure 1 shows some chronological call data from an example (fabricated) frauded account (the fields shown are just a sample; our call data contain many more attributes than shown here). The column at the far right indicates whether the call is fraudulent or not; that is, whether the call was placed by the customer or the bandit. A fraud analyst looking at this account would quickly be able to recognize the two classes of calls:

1. The legitimate user calls from the metro New York City area, usually during working hours, and typically makes calls lasting a few minutes.
2. The bandit's calls originate from a different area (Boston, Massachusetts, about 200 miles away), are made in the evenings, and last less than a minute.

Ideally, a fraud detection system should be able to learn such rules automatically and use them to catch fraud.

This paper addresses the automatic design of user profiling methods. User profiling methods are attractive because they do not depend upon any special hardware capability, as authentication does, nor do they require that the customer replace or upgrade existing equipment. Moreover, the ability to generate such detectors is domain independent: such a system should be able to generate fraud detectors for any domain with superimposition fraud.

Date & Time	Day	Duration	Origin	Destination	Fraud
1/01/95 10:05:01	Mon	13 mins	Brooklyn, NY	Stamford,CT	
1/05/95 14:53:27	Fri	5 mins	Brooklyn, NY	Greenwich,CT	
1/08/95 09:42:01	Mon	3 mins	Bronx, NY	White Plains, NY	
1/08/95 15:01:24	Mon	9 mins	Brooklyn, NY	Brooklyn, NY	
1/09/95 15:06:09	Tue	5 mins	Manhattan, NY	Stamford, CT	
1/09/95 16:28:50	Tue	53 sec	Brooklyn, NY	Brooklyn, NY	
1/10/95 01:45:36	Wed	35 sec	Boston, MA	Chelsea, MA	BANDIT
1/10/95 01:46:29	Wed	34 sec	Boston, MA	Yonkers, NY	BANDIT
1/10/95 01:50:54	Wed	39 sec	Boston, MA	Chelsea, MA	BANDIT
1/10/95 11:23:28	Wed	24 sec	White Plains, NY	Congers, NY	
1/11/95 22:00:28	Thu	37 sec	Boston, MA	East Boston, MA	BANDIT
1/11/95 22:04:01	Thu	37 sec	Boston, MA	East Boston, MA	BANDIT

Figure 1. Call records of a sample frauded account.

2.3. The need to be adaptive

There are a number of commercially available expert systems for fraud detection that include user profiling. Fraud analysts or system administrators can tune the techniques by adjusting parameters, or by entering specific patterns that will trigger alarms. Unfortunately, determining which potential patterns will be useful is a time-consuming process of trial-and-error. Moreover, the patterns of fraud are dynamic. Bandits constantly change their strategies in response to new detection techniques or new cellular hardware capabilities. By the time a system is manually tuned, the fraudulent behavior may have changed significantly.

The environment is dynamic in other ways as well. The level of fraud changes dramatically month-to-month because of modifications to work practices (both the carrier's and the bandits'). Also, the costs of missing fraud or of dealing with false alarms change with intercarrier contracts, and because of fraud analyst workforce issues.

For all these reasons, it is important that a fraud detection system adapt easily to new conditions. It should be able to notice new patterns of fraud. It should also be able to modify its alarm generation behavior, for example, as the level of fraud or the cost of dealing with a false alarm changes. Such adaptability can be achieved by generating fraud detection systems automatically from data, using data mining techniques.

3. Automatic construction of profiling fraud detectors

One approach to building a fraud detection system is to classify individual transactions, calls in our case, as being fraudulent or legitimate. Classification has been well explored, e.g., in machine learning and statistics, so this would seem to be a straightforward application of existing techniques.

We have not had success using standard machine learning techniques to construct such a classifier. Some specific results are discussed in Section 6.3. In general, there are two problems that make simple classification approaches infeasible.

- *Context*: A call that would be unusual for one customer would be typical for another. For example, a call placed from Brooklyn is not unusual for a subscriber who lives there, but might be very strange for a Boston subscriber. Thus, it is necessary (i) to discover indicators corresponding to *changes* in behavior that are indicative of fraud, rather than absolute indicators of fraud, and (ii) to *profile* the behavior of individual customers to characterize their normal behavior.
  If there were available substantial information about an account's context, we could possibly ameliorate this problem. Context information would comprise behavior information such as what the phone is used for, what areas it is used in, what areas/numbers it normally calls, what times of day, and so on. Context information is not available[1], so our solution is to derive it from historical data specific to each account. The discovery of context-sensitive fraud indicators and the profiling of individual accounts comprise two of the three major elements of the learning problem.
- *Granularity*: At the level of the individual call, the variation in calling behavior is large, even for a particular user. Legitimate subscribers occasionally make calls that look

suspicious. As far as we have been able to determine, it is not possible to achieve simultaneously the high degree of accuracy and high level of coverage necessary to classify individual calls effectively. Any classifier that fires on a significant number of defrauded accounts produces an unacceptably large number of false alarms. Therefore, decisions to take corrective action cannot be made with confidence on the basis of individual calls. Instead, it is necessary to aggregate customer behavior, smoothing out the variation, and watch for coarser-grained changes that have better predictive power. This is the third major element of the learning problem; in the experiments we describe later, we aggregate customer behavior into *account-days*.

In sum, the learning problem comprises three questions, each of which corresponds to a component of our framework.

1. *Which call features are important*? Which features or combinations of features are useful for distinguishing legitimate behavior from fraudulent behavior?
2. *How should profiles be created*? Given an important feature, how should we characterize/profile the behavior of a subscriber with respect to the feature, in order to notice important changes?
3. *When should alarms be issued*? Given the results of profiling behavior based on multiple criteria, how should they be combined to be effective in determining when fraud has occurred?

Each of these issues corresponds to a component of our framework.

4. The Detector Constructor framework

Our *Detector Constructor* framework is illustrated in figure 2. Under the framework, a system first learns rules that serve as indicators of fraudulent behavior. It then uses these rules, along with a set of templates, to create profiling monitors (M_1 through M_n). These monitors profile the typical behavior of each account with respect to a rule and, in use, describe how far each account is from its typical behavior. Finally, the system learns to weight the monitor outputs so as to maximize the effectiveness of the resulting fraud detector.

Figure 3 shows how such a detector will be used. The monitors are provided with a single day's calls from a given account, and each monitor generates a number indicating how unusual that account-day looks for the account. The numeric outputs from the monitors are treated as evidence and are combined by the detector. When the detector has enough evidence of fraudulent activity on an account, based on the indications of the monitors, it generates an alarm.

We now discuss each step of the framework in detail, illustrated by the particular choices made in our first implemented system, as applied to the problem of cloning fraud detection. The first Detector Constructor system is called DC-1. The call data used for detecting cloning fraud are chronological records of calls made by each subscriber, organized by

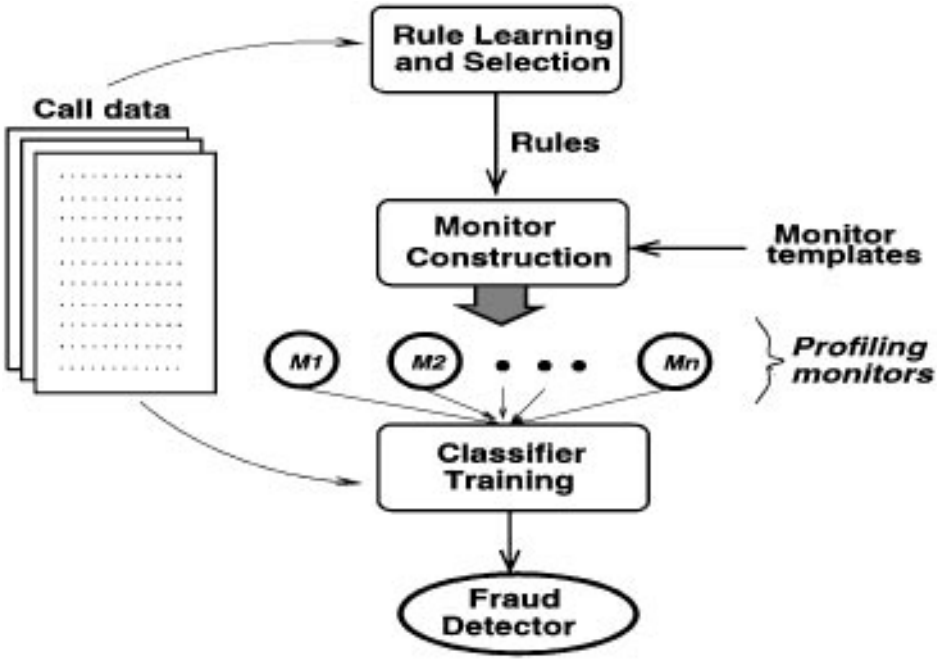

Figure 2. The framework for automatically constructing fraud detectors.

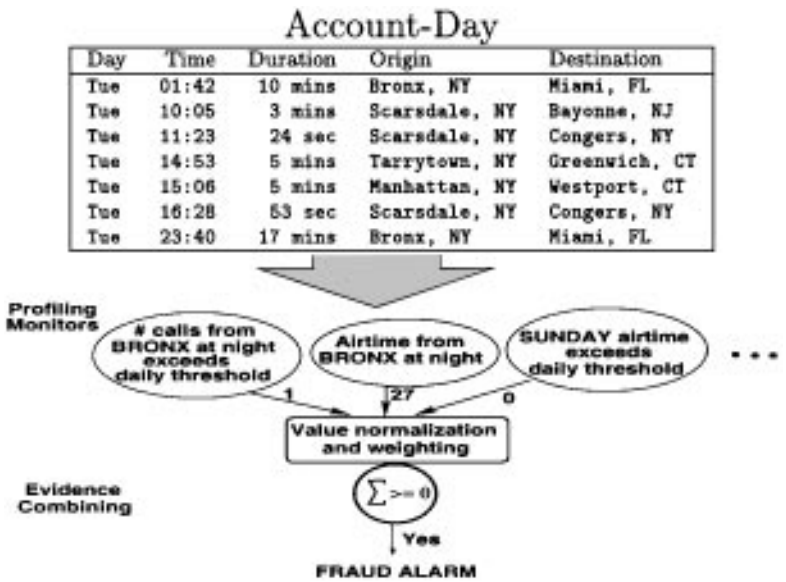

Figure 3. A DC-1 fraud detector processing a single account-day of data.

account. These data describe individual calls using attributes such as DATE, FROM-STATE, DURATION and CELL-SITE.

4.1. Learning fraud rules

The first stage of detector construction, *rule learning*, involves searching the call data for indicators of fraud. In the DC-1 system, the indicators are conjunctive rules discovered by a standard rule-learning program.

As discussed above, an obvious way of mining fraud indicators is to create an example set consisting of all frauded calls and all legitimate calls, and apply to a rule learning algorithm to the example set. However, this approach loses context information about the normal behavior of the account in which the fraud occurred. To illustrate the importance of context, consider a situation in which half the subscribers live in New York and half in Los Angeles. When cloned, the New York accounts are used in Los Angeles and the Los Angeles accounts are used in New York. Applying rule learning to the combined set of call records would uncover no fraud rules based on call origin; in other words, knowing that a call originated in New York says nothing about how likely it is to be fraud. In fact, this conclusion would be wrong: in this scenario a New York account with Los Angeles calls is much more likely to have been cloned than if it had only New York calls. This fact is missed when using a combined example set, because in combining examples all account context information is lost.

In light of the need to maintain context information, rule learning is performed in two steps. Rules are first generated locally based on differences between fraudulent and normal behavior for each account, then they are combined in a rule selection step.

4.1.1. Rule generation. DC-1 uses the RL program (Clearwater and Provost, 1990; Provost and Aronis, 1996) to generate indicators of fraud in the form of classification rules. Similar to other MetaDENDRAL-style rule learners (Buchanan and Mitchell, 1978; Segal and Etzioni, 1994; Webb, 1995), RL performs a general-to-specific search of the space of conjunctive rules. This type of rule-space search is described in detail by Webb (1995). In DC-1, RL uses a beam search for rules with certainty factors above a user-defined threshold. The certainty factor we used for these runs was a simple frequency-based probability estimate, corrected for small samples (Quinlan, 1987). In order to deal with the very large numbers of values for some of the attributes used to describe calls (more than 10,000 values in total), RL also used breadth-first marker propagation techniques, so that the algorithm's time complexity does not depend on the number of attribute values (Aronis and Provost, 1997). (RL's time complexity is linear in the number of attributes and the number of examples.)

The call data are organized by account, with each call record labeled as fraudulent or legitimate. When RL is applied to an account's calls it produces a set of rules that serve to distinguish, within that account, the fraudulent calls from the legitimate calls. As an example, the following rule would be a relatively good indicator of fraud:

```
(TIME-OF-DAY = NIGHT) AND (LOCATION = BRONX) ==> FRAUD
                Certainty factor = 0.89
```

This rule denotes that a call placed at night from The Bronx (a Borough of New York City) is likely to be fraudulent. The `Certainty factor = 0.89` means that, within this account, a call matching this rule has an 89% probability of being fraudulent.

From each account, RL generates a "local" set of rules describing the fraud on that account. Each rule is recorded along with the account from which it was generated. The covering heuristic typically used by RL was disabled, so that all of the (maximally general) rules with probability estimates above threshold would be generated. This option was chosen because rule generation in DC-1 is local and decisions about coverage should not be made locally. The next step, rule selection, incorporates information about coverage and generality.

4.1.2. Rule selection. After all accounts have been processed, a rule selection step is performed. The purpose of this step is to derive a set of rules that will serve as fraud indicators.

A rule selection step is necessary because the rule generation step typically generates tens of thousands of rules in total, most of which are specific only to single accounts. The system cannot know *a priori* how general each rule will be. For example, from one account RL may generate the rule:

```
(LOCATION = FREEHOLD) AND (DAY-OF-WEEK = SATURDAY) AND
        (CALL-DURATION < 47 SECS) ==> FRAUD
```

This rule is probably specific only to the account from which it was generated, but there is no *a priori* way to know a rule's generality in the generation step, which processes a single account at a time. If this rule is found in ("covers") many accounts, it is probably worth using; if it was only found in a single account, it is probably not a general indicator of fraudulent behavior. Even if the same account is cloned again, it will not be defrauded in exactly the same way. Note that DC-1's notion of coverage is slightly different from the standard notion, because of the multiple levels of granularity; in particular, DC-1 selects a set of rules that covers the *accounts*, as opposed to typical classifier learning, in which a set of rules is selected that covers the examples (in this case, the calls).

The rule selection algorithm is given in figure 4. The algorithm identifies a small set of general rules that cover the accounts. Two parameters control the algorithm. T_{rules} is a threshold on the number of rules required to "cover" each account. In the selection process, if an account has already been covered by T_{rules} rules, it will not be examined. T_{accts} is the number of accounts a rule must have been found in (i.e., mined from) in order to be selected at all.

For each account, the list of rules generated by that account is sorted by the frequency of occurrence in the entire account set. The highest frequency unchosen rule is selected. An account is skipped if it is already sufficiently covered. The resulting set of rules is used in construction of monitors.

4.2. Constructing profiling monitors

Rule learning produces a set of rules characterizing changes that commonly occur when an account is cloned. These rules are not universal: for a given account, we do not know to what extent that account's normal behavior already satisfies the rule. For example, the

```
Given:
    Accts: set of all accounts
    Rules: set of all fraud rules generated from Accts
    T_rules: (parameter) Number of rules required to cover each account.
    T_accts: (parameter) Number of accounts in which a rule must have been found.
Output:
    S: Set of selected rules.

 1. /* Initialization */
 2. S = {};
 3. for (a ∈ Accts) do Cover[a] = 0;
 4. for (r ∈ Rules) do
 5.       Occur[r] = 0; /* Number of accounts in which r occurs */
 6.       AcctsGen[r] = {}; /* Set of accounts generating r */
 7. end for
 8. /* Set up Occur and AcctsGen */
 9. for (a ∈ Accts) do
10.       R_a = set of rules generated from a;
11.       for (r ∈ R_a) do
12.            Occur[r] := Occur[r] + 1;
13.            add a to AcctsGen[r];
14. end for; end for
15. /* Cover Accts with Rules */
16. for (a ∈ Accts) do
17.       R_a = list of rules generated from a;
18.       sort R_a by Occur;
19.       while (Cover[a] < T_rules) do
20.            r := highest-occurrence rule from R_a
21.            Remove r from R_a
22.            if (r ∉ S and Occur[r] ≥ T_accts) then
23.                 add r to S;
24.                 for (a_2 ∈ AcctsGen[r]) do
25.                      Cover[a_2] = Cover[a_2] + 1;
26.                 end for; end if
27.       end while; end for
```

Figure 4. Rule selection and covering algorithm used by DC-1.

"Bronx-at-night" rule, mentioned above, may be very useful for someone living in Hartford, Connecticut, but it may cause many false alarms on a subscriber living in the Bronx. A fraud detection system should distinguish the two. In the latter case the system should inhibit the rule from firing, or at least require a much higher level of activation.

Sensitivity to different users is accomplished by converting the rules into *profiling monitors*. Each monitor has a *Profiling* step and a *Use* step. Prior to being used on an account, each monitor profiles the account. In the Profiling step, the monitor is applied to a segment of an account's typical (non-fraud) usage in order to measure the account's normal activity.

Statistics from this profiling period are saved with the account. In the monitor's Use phase, the monitor processes a single account-day at a time. The monitor references the normalcy measures calculated in Profiling, and generates a numeric value describing how abnormal[2] the current account-day is.

Profiling monitors are created by the monitor constructor, which employs a set of templates. The templates are instantiated by rule conditions. Given a set of rules and a set of templates, the constructor generates a monitor from each rule-template pair. Two monitor templates are shown in figure 5. At the top is a template that creates *threshold* monitors.

Threshold monitors

- **Given:** *Rule conditions* from a fraud rule.
- **Profiling:** On a daily basis, count the number of calls that satisfy *rule conditions.* Keep track of the maximum as *daily threshold.*
- **Use:** Given an account-day, let C be the set of all calls on that day that satisfy *rule conditions.*

$$Output = \begin{cases} 1 & \text{if } |C| > \textit{daily threshold} \\ 0 & \text{otherwise} \end{cases}$$

Standard deviation monitors

- **Given:** *Rule conditions* from a fraud rule.
- **Profiling:** On a daily basis, sum the airtime of all calls satisfying *rule conditions.* At the end of the training period, record the mean (μ) and standard deviation (σ) of the samples.
- **Use:** Given an account-day, let C be the set of all calls on that day that satisfy *rule conditions.* Let

$$\text{Airtime} = \sum_{\text{Call} \in C} airtime(Call)$$

$$Output = \begin{cases} \text{Airtime} & \text{if } \sigma = 0 \\ \frac{\text{Airtime} - \mu}{\sigma} & \text{if Airtime} > \mu \\ 0 & \text{otherwise} \end{cases}$$

Figure 5. Two templates for creating monitors from rules. A threshold monitor learns a threshold on maximum use and outputs a 1 whenever daily usage exceeds the threshold. A standard deviation monitor outputs the number of standard deviations over the mean profiled usage.

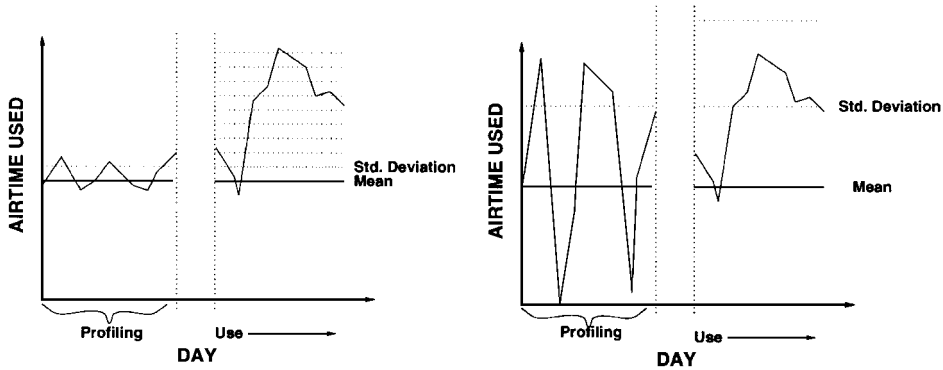

Figure 6. Using mean and standard deviation in profiling.

Such a monitor yields a binary feature corresponding to whether the user's behavior was above threshold for the given day. The bottom of figure 5 shows a template for a *standard deviation* monitor. In the Profiling period, such monitors measure the mean (μ) and standard deviation (σ) of typical usage; in the Use period, they produce a continuous output representing how many standard deviations above the mean an account-day is.

As an example, assume the Bronx-at-night rule mentioned earlier was used with the template shown in figure 5(b). Assume that, on some account, the subscriber called from the Bronx an average of five minutes per night with a standard deviation of two minutes. At the end of the Profiling step, the monitor would store the values (5, 2) with that account. In Use on that account, if the monitor processed a day containing three minutes of airtime from the Bronx at night, the monitor would emit a zero; if the monitor saw 15 minutes, it would emit $(15-5)/2 = 5$. This value denotes that the account is five standard deviations above its average (profiled) usage level.

Standard deviation monitors are sensitive both to the expected amount of activity on an account and to the expected daily variation of that activity. Figure 6 illustrates the difference, showing one monitor applied to two accounts. The account on the left has low variation in the Profiling period so its standard deviation is lower. Consequently the erratic behavior in its Use period will produce large values from the monitor. The account on the right has the same mean but exhibits much larger variation in Profiling period, so the standard deviation is higher. The variations in behavior during the Use period will not produce large values from the monitor.

4.3. Combining evidence from the monitors

The third stage of detector construction learns how to combine evidence from the monitors generated by the previous stage. For this stage, the outputs of the monitors are used as features to a standard learning program. Training is done on account data, and monitors evaluate one entire account-day at a time. In training, the monitors' outputs are presented along with the desired output (the account-day's correct class: fraud or non-fraud). The

evidence combination weights the monitor outputs and learns a threshold on the sum so that alarms may be issued with high confidence.

Many training methods for evidence combining are possible. We chose a simple Linear Threshold Unit (LTU) (Nilsson, 1965; Young, 1984) for the experiments reported below. An LTU is simple and fast, and enables a good first-order judgment of the features' worth.

A feature selection process is used to reduce the number of monitors in the final detector. Some of the rules do not perform well when used in monitors, and some monitors overlap in their fraud detection coverage. We therefore employ a sequential forward selection process (Kittler, 1986) which chooses a small set of useful monitors. Empirically, this simplifies the final detector and increases its accuracy.

The final output of DC-1 is a detector that profiles each user's behavior based on several indicators, and produces an alarm if there is sufficient evidence of fraudulent activity. Figure 3 shows an example of a simple detector evaluating an account-day.

Before being used on an account, the monitors each profile the account. They are applied to a profiling segment (thirty days in our experiments) during which they measure unfrauded usage. In our study, these initial thirty account-days were guaranteed free of fraud, but were not otherwise guaranteed to be typical. From this initial profiling period, each monitor measures a characteristic level of activity.

4.4. Summary of the detector construction process

In sum, DC-1 begins by examining the call records of defrauded accounts. The call records are expressed in terms of a set of base-level attributes with thousands of possible values. From these data, the system generates rules characterizing fraudulent calls within accounts, then selects a smaller set of general rules as indicators of fraudulent behavior.

These rules are used as the basis from which to build a set of profiling monitors, each of which examines behavior based on one learned rule. A monitor learns the typical behavior of an account by scanning an initial sequence of the account's calls (its "profiling period") and saving some statistics. Subsequently, the monitor examines chunks of calls (in our experiments, account-days). The monitor subsequently examines each chunk (day) of each account's behavior and outputs a number indicating how far away from normal the behavior is.

In order to construct a high-confidence detector, DC-1 must then learn to combine the outputs of the monitors effectively. To do this, it trains a classifier on a sample of account-days. Each account-day is a training instance expressed as a vector of monitor outputs for that day, and labelled as either containing fraud or not. After training, the system has a classifier that is able to combine the monitors effectively.

The final output of the system is a set of monitors and a trained classifier for combining their outputs. In order to be applied to a new account, the monitors must see a profiling period of days from that account.

The next sections describe our cellular call data and the experiments we have performed on the system.

5. The data

The call data used for this study are records of cellular calls placed over four months by users in the New York City area—an area with high levels of fraud. Each call is described by thirty-one attributes, such as the phone number of the caller, the duration of the call, the geographical origin and destination of the call, and any long-distance carrier used. Because of security considerations, we are unable to disclose all the features used in the system.

To these thirty-one attributes are added several derived attributes that incorporate knowledge we judged to be potentially useful. One such attribute is a categorical `TIME-OF-DAY` variable representing the time segment of the day in which a call is placed. Its values are MORNING, AFTERNOON, TWILIGHT, EVENING and NIGHT. Another derived attribute is `TO-PAYPHONE`, a binary flag indicating whether the call terminated at a payphone. Note that any number of additional features could be added to encode relevant domain knowledge.

Each call is also give a class label of legitimate or fraudulent. This is done by cross referencing a database of all calls that were credited as being fraudulent for the same time period.

5.1. Data cleaning

Like all real-world data, our cellular call data contain errors and noise from various sources. For example, calls are marked as fraudulent based on a process called *block crediting*. In this process, the customer and the carrier representative together establish the range of dates during which the fraud occurred, and the calls within the range are credited. The customer is usually not asked about each individual call. The block crediting process uses heuristics to discard obvious non-fraudulent calls from the credited block, but these heuristics are fallible. Also, if there is a disagreement about the fraud span, the customer service representative usually concedes, in the customer's favor, to a wider date span. Any erroneously credited calls constitute noise in our data.

Because of these noise sources, we cleaned the data in several ways.

- Each account's calls were scanned automatically to eliminate credited calls to numbers that had been called outside of the credited block. In other words, the program looked for credited calls made to a phone number that also had been called by the legitimate subscriber; in this case, the crediting may have been a mistake, so the call is discarded completely.
- An account-day was classified as fraudulent only if five or more minutes of fraudulent usage occurred. Days including one to four minutes of fraudulent usage were discarded. This policy eliminated a small number of "gray area" account-days probably mislabelled due to small amounts of noise. For example, the database of credits due to fraud occasionally included credits for other reasons, such as wrong numbers.
- Within any time period there will be fraud that has not yet been detected. We assumed that some genuinely fraudulent calls would not be marked as such because of this time lag. We attempted to minimize this noise by delaying the data retrieval by two weeks.

- In preliminary experiments, rule learning uncovered some unusual attribute values (e.g., `CELLSITE = 0`) that seemed to be very strong indicators of fraud. Discussions with the database providers led us to conclude that these suspicious values were artifacts of the crediting process: in some circumstances, crediting would erase or replace certain fields. Because the values appeared primarily in credited records, data mining had extracted them as high-confidence fraud rules. We found five or six such misleading values and eliminated from the database all records containing them.

In addition, the start times of calls had been recorded in local time with respect to the switch of origin. The calls were normalized to Greenwich Mean Time for chronological sorting.

5.2. Data selection

The call data were separated carefully into several partitions for rule learning, account profiling, and detector training and testing. Once the monitors are created and the accounts profiled, the system transforms raw call data into a series of account-days using the outputs of the monitors as features.

Rule learning and selection used 879 accounts comprising over 500,000 calls. About 3600 accounts were selected for profiling, training, and testing. The only condition used to select these 3600 accounts was that they be guaranteed to have at least thirty fraud-free days of usage before any fraudulent usage. The initial thirty days of each account were used for profiling. The remaining days of usage were used to generate approximately 96,000 account-days. Using randomly selected accounts, we generated sets of 10,000 account-days for training and 5000 account-days for testing. Training and testing accounts were distinct, so their account-days were not mixed between training and testing[3]. Each set of account-days was chosen to comprise 20% fraud and 80% non-fraud days.

6. Experiments and evaluation

Rule learning generated 3630 rules, each of which applied to two or more accounts. The rule selection process, in which rules are chosen in order of maximum account coverage, yielded a smaller set of 99 rules sufficient to cover the accounts. Each of the 99 rules was used to instantiate two monitor templates, yielding 198 monitors. The final feature selection step reduced this to eleven monitors, with which the experiments were performed.

6.1. The importance of error cost

In this domain, different types of errors have different costs. A realistic evaluation should take misclassification costs into account. Classification accuracy, a standard metric within machine learning and data mining, is not sufficient.

A false positive error (a false alarm) corresponds to wrongly deciding that a customer has been cloned. Based on the cost of a fraud analyst's time, we estimate the cost of a false

positive error to be about $5. A false negative error corresponds to letting a frauded account-day go undetected. Rather than using a uniform cost for all false negatives, we estimated a false negative to cost $.40 per minute of fraudulent airtime used on that account-day. This figure is based on the proportion of usage in local and non-local ("roaming") markets, and their corresponding costs.

Because LTU training methods try to minimize errors but not error costs, we employed a second step in training. After training, the LTU's threshold is adjusted to yield minimum error cost on the training set. This adjustment is done by moving the decision threshold from -1 to $+1$ in increments of .01 and computing the resulting error cost. After the minimum cost on training data is found, the threshold is clamped and the testing data are evaluated.

6.2. DC-1 compared with alternative detection strategies

Table 1 shows a summary of results of DC-1 compared against other detectors. The name of each detector is shown in the left-most column. Classification accuracy averages and standard deviations are shown in the second column. The third column shows the mean and standard deviations of test set costs. The right-most column, "Accuracy at cost," is the corresponding classification accuracy of the detector when the threshold is set to yield lowest-cost classifications.

Each detector was run ten times on randomly selected training and testing accounts. For comparison, we evaluated DC-1 along with other detection strategies.

- **Alarm on All** represents the policy of alarming on every account every day. The opposite strategy, **Alarm on None**, represents the policy of allowing fraud to go completely unchecked. The latter corresponds to the maximum likelihood accuracy classification. Note that the cost of **Alarm on None** does not take into account the inhibitory effect of fraud detection, without which fraud levels would likely continue to rise.
- **Collisions and Velocities** is a detector using collision and velocity checks described in Section 2.2.2. DC-1 was used to learn a threshold on the number of collision and velocity alarms necessary to generate a fraud alarm. It is surprising that Collisions and Velocity

Table 1. Accuracies and costs of various detectors.

Detector	Accuracy (%)	Cost (US$)	Accuracy at cost (%)
Alarm on all	20	20000	20
Alarm on none	80	18111 ± 961	80
Collisions + velocities	$82 \pm .3$	17578 ± 749	$82 \pm .4$
High usage	$88 \pm .7$	6938 ± 470	85 ± 1.7
Best individual DC-1 monitor	$89 \pm .5$	7940 ± 313	$85 \pm .8$
State of the art (SOTA)	$90 \pm .4$	6557 ± 541	$88 \pm .9$
DC-1 detector	$92 \pm .5$	5403 ± 507	$91 \pm .8$
SOTA plus DC1	$92 \pm .4$	5078 ± 319	$91 \pm .8$

Checks, commonly thought to be reliable indicators of cloning, performed poorly in our experiments.

The performance of collisions and velocity checks was originally worse than reported here because of false alarms. Manual inspection of false alarms revealed a few synchronization problems; for example, some apparent collisions were caused when a call was dropped then quickly re-established in a neighboring cell whose clock did not agree with the first cell's. Some such conditions could be caught easily, so we patched the detection algorithms to check for them. The results in this paper are for the improved detectors.

Investigation of confusion matrices revealed that the collision and velocity check detectors' errors were due almost entirely to false negatives. In other words, when the detectors fired they were accurate, but many fraud days never exhibited a collision or velocity check.

- Some fraud analysts believe that cloning fraud is usually accompanied by large jumps in account usage, and sophisticated mining of fraud indicators is probably unnecessary since most fraud could be caught by looking for sudden increases in usage. We created the **High Usage** detector to test this hypothesis. It generates alarms based only on amount of usage. It is essentially a standard deviation monitor (see figure 5) whose rule conditions are always satisfied. The threshold of this detector was found empirically from training data.

  Note that the evaluation of cost for the high usage detector may be overly optimistic, due to inadequacies in our cost model. In particular, a trained high usage detector learns to optimally "skim the cream," without regard to the fact that the errors it makes will involve annoying the best customers. In these cases, the cost of a false alarm may be much higher than the fixed cost we assigned.
- The **Best Individual DC-1 Monitor** was used as an isolated detector. This experiment was done to determine the additional benefit of combining monitors. The best individual monitor was generated from the rule:

  ```
  (TIME-OF-DAY = EVENING) ==> FRAUD
  ```

  Rule learning had discovered (in 119 accounts) that the sudden appearance of evening calls, in accounts that did not normally make them, was coincident with cloning fraud. The relatively high accuracy of this one monitor reveals that this is a valuable fraud indicator.

  Our `TIME-OF-DAY` attribute has five possible values: MORNING, AFTERNOON, TWILIGHT, EVENING and NIGHT. Although EVENING is by far the most frequent value implicated in fraud, rule learning generated fraud rules involving each of these values. This suggests that any time-of-day change in a subscriber's normal behavior may be indicative of fraud, though the other shifts may not be predictive enough to use in a fraud monitor.
- The **DC-1** detector incorporates all the monitors chosen by feature selection. We used the weight learning method described earlier to determine the weights for evidence combining.
- The **SOTA** ("State Of The Art") detector incorporates thirteen hand-crafted profiling methods that were the best individual detectors identified in a previous study. Each method profiles an account in a different way and produces a separate alarm. Weights

for combining SOTA's alarms were determined by our weight-tuning algorithm. Details on the detectors comprising SOTA are given in the appendix.

The results in Table 1 demonstrate that DC-1 performs quite well. In fact, DC-1 outperforms SOTA in terms of both accuracy and cost[4]. In our experiments, lowest cost classification occurred at an accuracy somewhat lower than optimal. In other words, some classification accuracy can be sacrificed to decrease cost. More sophisticated methods could be used to produce cost sensitive classifiers, which would probably produce better results.

Finally, the monitors of SOTA and DC-1 were combined into a hybrid detector. The resulting detector (**SOTA plus DC-1**) exhibits no increase in classification accuracy, but does show a slight improvement in fraud detection cost.

In this work we have dealt with differing costs of false positive and false negative errors. However, we have still glossed over some complexity. For a given account, the only false negative fraud days that incur cost to the company are those prior to the *first* true positive alarm. After the fraud is detected, it is terminated. Thus, our analysis overestimates the costs slightly; a more thorough analysis would eliminate such days from the computation.

6.3. Fraudulent call classifiers

Section 4.1 asserted that account context is important in the rule learning step: a global example set taken from all accounts would lose information about each account's normal behavior. In order to test this hypothesis, two such call classifiers were created from global example sets.

Applying standard classification algorithms to the call data was difficult for several reasons. First, the description language is very detailed because many thousands of attribute values appear in the data. Because of this, the volume of data necessary was relatively large for desktop platforms; the use of fewer than 100,000 examples led to erratic classification behavior. Furthermore, in order to achieve high coverage of calls, massively disjunctive concept descriptions had to be learned—there were no simple classifiers that performed well.

After trying many approaches, we chose two classifiers learned in the following manner. A set of 100,000 training examples was sampled from the accounts set aside for rule learning. The sample was random, but stratified to achieve a 50/50 class distribution. RL was applied to these data, with parameters set so that it learned massively disjunctive rule sets. The two classifiers, CC 1054 and CC 1861, comprise 1054 and 1861 rules, respectively[5]. Each of these rule sets covered around 60% of the calls in a 92212 example test set, with an accuracy of about 75% on the calls it covered. We observed a clear and graceful tradeoff between accuracy and coverage: as the coverage increased, the accuracy decreased.

In order to achieve a competitive comparison, the call classifiers were then given the advantage of profiling and monitoring. A standard deviation airtime monitor was created from each. Specifically, instead of instantiating the monitor template with a single rule, the template was instantiated with the entire classifier. The resulting monitor profiled each account's normal behavior with respect to the classifier's output. The call classifier

Table 2. A comparison of DC-1 to two global call classifiers.

Detector	Accuracy (%)	Cost (US$)	Accuracy at cost (%)
CC 1054	$88 \pm .4$	8611 ± 531	$88 \pm .6$
CC 1861	$88 \pm .5$	8686 ± 804	$88 \pm .6$
DC-1 detector	$92 \pm .5$	5403 ± 507	$91 \pm .8$

monitor learns if a particular customer's legitimate behavior typically triggers a positive output. Furthermore, each call classifier monitor was inserted into the DC-1 weight-training framework in order to find an "optimal" output threshold for accuracy maximization or cost minimization.

The results are shown in Table 2. The two call classifiers perform similarly, and DC-1 outperforms both by a considerable margin. Indeed, we were surprised that the the call classifier monitors perform as well as they do.

6.4. Shifting distributions of fraud

As discussed in Section 2.3, a fraud detection system should be able to adapt to shifting fraud distributions. For example, each month the relative amount of fraud changes slightly, and it is rarely possible to predict the level of fraud far into the future. Thus, unless it is adaptive, even a well-tuned detection system will begin to lose its edge.

To illustrate this point, we simulated the effects of changing fraud distributions on detector performance. One DC-1 detector was trained on a fixed distribution of account-days (80% non-fraud, 20% fraud) and tested against several other distributions (ranging from 75% to 99% non-fraud account-days), to simulate a well-tuned but non-adaptive detection system. Another DC-1 detector was allowed to adapt to each distribution; its LTU threshold was re-trained for minimum predicted cost on a training set with the new distribution.

The results are shown in figure 7. The X-axis is the percentage of non-fraud account-days, and the Y-axis is the cost per account day. This figure shows that the second detector, which is allowed to adjust itself to each new distribution, is consistently more cost effective than the fixed detector. This difference increases as the testing distribution becomes more skewed from the distribution upon which the fixed detector was trained.

We close by noting that these experiments illustrated changes in fraud detection performance with respect to fairly simple changes in fraud distribution (changing fraud volume). The patterns of fraud also change, particularly in reponse to detection methods. Thus the ability to use data mining to discover new patterns amplifies the benefit of adaptability.

6.5. Discussion

It is difficult to evaluate DC-1 against existing expert systems for fraud detection. Fraud detection departments carefully protect information about how much fraud they have and how effective their detection strategies are. Likewise, vendors of fraud detection systems

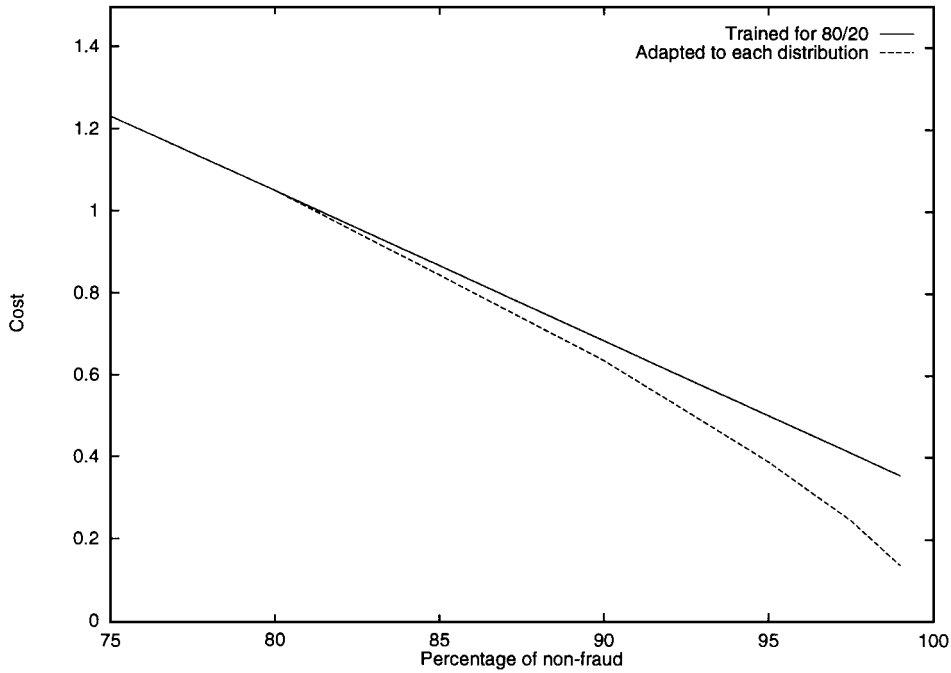

Figure 7. The effects of changing fraud distributions.

protect details of their systems' operation that may constitute trade secrets. Little performance data on fielded systems are available, and what data do exist are insufficient for careful evaluation.

For these reasons, we evaluated DC-1 against individual known fraud detection techniques, as well as against a collection of techniques representing the state of the art as we understand it. Results in the previous sections show that the DC-1 detector performs better than the high-usage alarm and the collision/velocity alarm. DC-1 also out-performs the SOTA detector, consisting of a collection of the best fraud detection techniques known to us, trained by DC-1's evidence combining method.

DC-1's framework has three main components, and is more complex than other approaches. Our experiments were designed not only to evaluate the overall performance of the system, but also to analyze the contribution of the individual components. In particular:

- The High Usage detector profiles with respect to undifferentiated account usage. Comparison with DC-1's performance demonstrates the benefit of using rule learning to uncover specific indicators of fraudulent calls.
- The Call Classifier detectors represent rule learning without the benefit of account context. Comparison with DC-1's performance demonstrates the value of DC-1's rule generation step, which does preserve account context.
- Comparison of DC-1 with the single best individual DC-1 monitor demonstrates the benefit of combining evidence from multiple monitors.

- Experiments with shifting fraud distributions indicate the benefit of making evidence combination sensitive to fraud distributions.

In each of these cases, the composite DC-1 system out-performed the detector in which a significant piece was missing. These results suggest that each component contributes critically to the performance of the entire detector.

Our system uses a Linear Threshold Unit to combine evidence from the monitors. Other methods of evidence combination are possible. We performed some experiments with multi-layer neural networks, but found that adding units to a hidden layer did not improve performance. These networks produced higher training accuracies, but lower accuracies on the test sets; such behavior is symptomatic of data overfitting. Additional experimentation might yield better-performing networks, but we have not pursued this. It is possible that, because the neural network is applied far along in the fraud detection process as a means of combining evidence, non-linear combinations of the evidence contribute little to fraud detection performance.

By increasing the expressiveness of the language used for inductive learning, it may be possible to learn more general patterns of fraudulent behavior, reducing the need for highly disjunctive class descriptions. The caveats mentioned earlier about the inability to procure background knowledge for context notwithstanding, it may be possible to provide additional context by linking call and account data to geographic and demographic databases. Furthermore, it may be possible to learn context in one stage, and then apply relational learning approaches in a later stage.

One such possibility is to make use of inductive learners that learn concept descriptions in first-order logic, such as FOIL (Quinlan, 1990) or ILP methods (Džeroski, 1996). Given the appropriate context information, it is possible that more expressive methods could learn general relational rules such as the following, which indicates fraud when a user calls from an abnormal location.

$$\begin{aligned} &\texttt{CALL\_ORIGIN(X,ORIG)} \ \& \\ &\texttt{NORMAL\_CALL\_LOCS(USER,LOCS)} \ \& \\ &\texttt{ORIG} \notin \texttt{LOCS} \Longrightarrow \texttt{FRAUD} \end{aligned}$$

Another possibility is to use a learner that forms propositional rules, but can take advantage of relational background knowledge in the process (Aronis et al., 1996). We have not explored the use of relational learning to any great extent. We have linked the data to knowledge about the geographic locations of telephone numbers ((Aronis and Provost, 1997), which does produce useful generalizations of the areas to which calls are placed.

Finally, it is important to point out additional limitations to the evaluation of learned classifiers on real-world problems. For the work described in this paper, we made use of techniques to deal with skewed class distributions, viz., stratified sampling, and to deal with nonuniform misclassification costs, viz., empirical threshold adjustment. We also ensured that our evaluation include cost effectiveness in addition to accuracy. However, because of the complexity and dynamics of real-world domains, determining *precisely* the target cost and class distributions is often impossible. As noted above, levels of fraud and costs change monthly. It is important to be able to compare competing classification methods

under imprecision in these distributions. The investigation and design of such techniques is an important area for future research (Provost and Fawcett, 1997).

7. Related work

Fraud detection is related to *intrusion detection*, a field of computer security concerned with detecting attacks on computers and computer networks (Frank, 1994; Sundaram, 1996; Kumar, 1995). Many forms of intrusion are instances of superimposition fraud, and thus candidates for systems built with our framework. Within the intrusion detection community, *anomaly detection* systems try to characterize behavior of individual users in order to detect intrusions on that user's account via anomalies in behavior. Existing anomaly detection systems typically examine audit trails of user activities, which fill the same roll as cellular call records in DC-1. DC-1 would be considered a statistical anomaly detection system. Sundaram (1996) writes:

> An open issue with statistical approaches in particular, and anomaly detection systems in general, is the selection of measures to monitor and the choice of metrics. It is not known exactly what the subset of all possible measures that accurately predicts intrusive activities is.

DC-1's framework directly addresses this problem. Its rule learning step examines large numbers of fraud episodes in order to generate features (measures) that distinguish fraudulent from legitimate behavior. To the best of our knowledge, no published anomaly detection system does this.

Calling card fraud and credit card fraud are other forms of superimposition fraud. A system built by Yuhas (1993, 1995) examines a set of records representing calling-card validation queries to identify queries corresponding to fraudulent card usage. Yuhas transformed the problem into a two-class discrimination task and trained several machine learning models on the data. All three models had comparable performance on the test sets. His system must be provided with appropriate features; it neither mines the data for fraud indicators nor measures typical customer usage. Stolfo et al. (1997) address credit card fraud detection. They also transform the problem into a two-class discrimination task, and do not use customer-specific information in detection. Specifically, they predict whether individual transactions are fraudulent. In our domain, we found that DC-1 significantly improves detection performance over systems that use transaction classification alone. It would be interesting to determine whether a system like DC-1 could improve performance on these other superimposition fraud tasks.

Ezawa and Norton (1995, 1996) have addressed the problem of uncollectible debt in telecommunications services. They use a goal-directed Bayesian network for classification, which distinguishes customers who are likely to default from those who are not. As with our work, Ezawa and Norton's work faces problems with unequal error costs and skewed class distributions. However, it does not face the problem of determining the typical behavior of individual customers so as to recognize superimposed fraudulent behavior. Mining the data to derive profiling features is not necessary.

Because fraud happens over time, methods that deal with time series are relevant to this work. However, traditional time series analysis (Chatfield, 1984; Farnum and Stanton, 1989) in statistics strives either to characterize an entire time series or to forecast future events in the series. Neither ability is directly useful to fraud detection.

Hidden Markov Models (Rabiner and Juang, 1986; Smyth, 1994) are concerned with distinguishing recurring sequences of states and the transitions between them. However, fraud detection usually only deals with two states (the "frauded" and "un-frauded" states) with a single transition between them. Yuhas (1995) mentions the possibility of recognizing "at home" and "travel" states in order to distinguish frauded states more effectively. This differentiation could be useful for reducing false alarms. We are aware of no work pursuing this idea.

8. Conclusion

The detection of cellular cloning fraud is a relatively young field. Fraud behavior changes frequently as bandits adapt to detection techniques, and fraud detection systems should be adaptive as well. However, in order to build usage monitors we must know which aspects of customers' behavior to profile. Historically, determining such aspects has involved a good deal of manual work, hypothesizing useful features, building monitors and testing them. Determining how to combine them involves much trial-and-error as well.

We have presented and demonstrated a framework that automates the process of generating fraud detectors. This framework is not specific to cloning fraud, but may be applied to superimposition fraud problems in any domain. Prime candidates are toll fraud, computer intrusion and credit-card fraud. For example, in credit-card fraud, data mining may identify locations that arise as new hot-beds of fraud. The constructor would then incorporate monitors that notice if customers begin to charge more than they usually do from these specific locations.

Even with relatively simple components, DC-1 is able to exploit mined data to produce a detector whose performance exceeds that of the state-of-the-art. The SOTA system took several person-months to build; the DC-1 detector took several CPU-hours. Furthermore, DC-1 can be retrained at any time as necessitated by the changing environment.

Such adaptability is beneficial in many ways. It can save effort in time-consuming manual feature identification and detector tuning. It can save on monetary losses that would occur during the manual identification and tuning process. It can save on less quantifiable damage done due to higher fraud, such as lower customer opinion (or even customer churn). Finally, it can act to *prevent* fraud; a system that quickly adapts to new patterns will be avoided by bandits in favor of easier prey.

Appendix

State of the art (SOTA) detector

The **SOTA** ("State Of The Art") detector incorporates thirteen profiling methods. Each method profiles an account in a different way and produces a separate alarm. Some of

the monitors were designed by hand, but those that employ weights used DC-1's weight tuning methods. Specifically, SOTA contains the following monitors:

- Two collision detectors, which scan for call collisions of greater than 30 seconds and 60 seconds, respectively.
- Two velocity detectors, using velocity thresholds of 400 and 600 miles per hour, respectively.
- Three "dialed digits" monitors. We created a dialed digit database as follows. We scanned through accounts reserved for rule learning and recorded how many distinct accounts called a given number both legitimately and in a fraud period. A phone number was discarded if any legitimate subscriber called it; otherwise, a count was saved of the number of times it was called from a cloned phone.
  Because we did not know an ideal threshold on the number of "hits" required, we created three monitors each with a different threshold.
- Two daily standard deviation usage monitors. One counted the number of calls on that account-day, one measured total airtime on that account-day.
- Four "bad cellsite" indicators. It is commonly believed that certain cellsites are the locus for higher than average amounts of fraud, so calls originating from those cellsites might be suspicious. To test this strategy, we tallied the frauded accounts calling each of the cellsites in our region, then computed the percentage of frauded accounts using each cellsite. The twenty worst cellsites were extracted from this list. Using this cellsite list, we created four detectors that counted hits to these "bad cellsites" each in a different way.

Acknowledgments

This work was sponsored by NYNEX Science and Technology. The views and conclusions in this paper are those of the authors and do not represent official NYNEX policy.

We thank Nicholas Arcuri and the Fraud Control Department at Bell Atlantic NYNEX Mobile for many useful discussions about cellular fraud and its detection. We also thank Usama Fayyad, Andrea Danyluk and our anonymous referees for comments on drafts.

In developing DC-1 we made extensive use of many freely-available software packages. We wish to thank the generous authors, developers and maintainers of the following software: The Perl programming language and many of its user-contributed modules, Donald Tveter's Backprop program, numerous GNU packages including Emacs and GCC, the GMT geographic information system, Gnuplot and the LaTeX text processing system.

Notes

1. In fact, because many cellular phones belong to large corporate accounts, often even basic user information such as home town and work location is unavailable.
2. Technically, the numeric value only describes how much above normal the account is. Behavior levels below normal are not considered.
3. If account-days from a single account appear in both training and testing sets, the performance evaluation can be deceptively optimistic. Fraudulent behavior within a specific cloning episode is more similar than fraudulent

behavior between episodes. When deployed, the monitors will be used to search for previously unseen cloning episodes.

4. Earlier work (Fawcett and Provost, 1996) reported a higher accuracy for SOTA than is shown here. Further development of SOTA revealed that some of its component methods, developed in a prior study, had been built from account data that overlapped data used to test the methods. When a strict separation was enforced, SOTA performance declined slightly to the figures shown here.
5. To learn CC 1861 (1054) RL tried to cover the example set with rules each of which covered at least 50 (100) examples and had a Laplace estimate greater than or equal to 0.95, using a beam width of 5000.

References

Aronis, J. and Provost, F. 1997. Increasing the efficiency of data mining algorithms with breadth-first marker propagation. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), AAAI Press, pp. 119–122.

Aronis, J., Provost, F., and Buchanan, B. 1996. Exploiting background knowledge in automated discovery. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), AAAI Press, pp. 355–358.

Buchanan, B.G. and Mitchell, T.M. 1978. Model-directed learning of production rules. In Pattern-Directed Inference Systems, F. Hayes-Roth (Ed.), New York: Academic Press, pp. 297–312.

Chatfield, C. 1984. The Analysis of Time Series: An Introduction (third edition). New York: Chapman and Hall.

Clearwater, S. and Provost, F. 1990. RL4: A tool for knowledge-based induction. In Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence, IEEE CS Press, pp. 24–30.

Davis, A. and Goyal, S. 1993. Management of cellular fraud: Knowledge-based detection, classification and prevention. In Thirteenth International Conference on Artificial Intelligence, Expert Systems and Natural Language, Avignon, France, vol. 2, pp. 155–164.

DeMaria, R. and Gidari, A. 1996. Uncovering unsavory customers. Cellular Business, 24–30.

Džeroski, S. 1996. Inductive logic programming and knowledge discovery in databases. Advances in Knowledge Discovery and Data Mining, Menlo Park, CA: AAAI Press, 117–152.

Ezawa, K. and Norton, S. 1995. Knowledge discovery in telecommunication services data using bayesian network models. In Proceedings of First International Conference on Knowledge Discovery and Data Mining, U. Fayyad and R. Uthurusamy (Eds.), Menlo Park, CA: AAAI Press, pp. 100–105.

Ezawa, K. and Norton, S. 1996. Constructing Bayesian networks to predict uncollectible telecommunications accounts. IEEE Expert, 45–51.

Farnum, N. and Stanton, L. 1989. Quantitative Forecasting Methods. Boston, MA: PWS-Kent Publishing Company.

Fawcett, T. and Provost, F. 1996. Combining data mining and machine learning for effective user profiling. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), E. Simoudis, J. Han, and U. Fayyad (Eds.), Menlo Park, CA: AAAI Press, pp. 8–13.

Frank, J. 1994. Machine learning and intrustion detection: Current and future directions. In National Computer Security Conference, vol. 1, pp. 22–33. Available via `http://seclab.cs.ucdavis.edu/papers/ncsc.94.ps`.

Herzog, J. 1995. Beware of hurricane clone. Newaves. Available from `http://www.pcia.com/1195l.htm`.

Kittler, J. 1986. Feature selection and extraction. In Handbook of Pattern Recognition and Image Processing, K.S. Fu (Ed.), New York: Academic Press, pp. 59–83.

Kumar, S. 1995. A Pattern Matching Approach to Misuse Intrusion Detection. Ph.D. thesis, Purdue University, Department of Computer Sciences. Available via `ftp://coast.cs.purdue.edu/pub/COAST/kumar-phd-intdet.ps.gz`.

Nilsson, N.J. 1965. Learning Machines. New York: McGraw-Hill.

Provost, F. and Aronis, J. 1996. Scaling up inductive learning with massive parallelism. Machine Learning, 23:33–46.

Provost, F. and Fawcett, T. 1997. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), AAAI Press, pp. 43–48.

Quinlan, J. 1990. Learning logical definitions from relations. Machine Learning, 5:239–266.
Quinlan, J.R. 1987. Generating production rules from decision trees. In Proceedings of the Tenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, pp. 304–307.
Rabiner, L.R. and Juang, B.H. 1986. An introduction to hidden Markov models. IEEE ASSP Magazine, 3(1):4–16.
Redden, M. 1996. A technical search for solutions. Cellular Business, 84–87.
Segal, R. and Etzioni, O. 1994. Learning decision lists using homogeneous rules. In Proceedings of the Twelfth National Conference on Artificial Intelligence, Menlo Park, CA: AAAI Press, pp. 619–625.
Smyth, P. 1994. Hidden Markov models for fault detection in dynamic systems. Pattern Recognition, 27(1):149–164.
Steward, S. 1997. Lighting the way in '97. Cellular Business, 23.
Stolfo, S., Prodromidis, A., Tselepsis, S., Lee, W., Fan, D., and Chan, P. 1997. JAM: Java agents for meta-learning over distributed databases. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), AAAI Press, pp. 74–81.
Sundaram, A. 1996. An introduction to intrusion detection. ACM Crossroads—Special Issue on Computer Security, 2(4). Available from `http://www.acm.org/crossroads/xrds2-4/intrus.html`.
Walters, D. and Wilkinson, W. 1994. Wireless fraud, now and in the future: A view of the problem and some solutions. Mobile Phone News, 4–7.
Webb, G. 1995. OPUS: An efficient admissible algorithm for unordered search. Journal of Artificial Intelligence Research, 3:383–417.
Young, P. 1984. Recursive Estimation and Time-Series Analysis. New York: Springer-Verlag.
Yuhas, B.P. 1993. Toll-fraud detection. In Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications, J. Alspector, R. Goodman, and T. Brown (Eds.), Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 239–244.
Yuhas, B.P. 1995. Learning the structure of telecommunications fraud. Technical report, Bellcore.

Tom Fawcett has been a researcher in machine learning and data mining at NYNEX Science and Technology since 1993. He received his B.S. in Computer Science from Worcester Polytech Institute, his Masters from Rutgers University and his Ph.D. from the University of Massachusetts at Amherst. His interests include the effect of representation on induction, time series problems and applications of machine learning to real-world problems.

Foster Provost has been a researcher in machine learning and data mining at NYNEX Science and Technology since 1994. Previously he worked on automated knowledge discovery for scientific domains at the University of Pittsburgh. He received his B.S. in Physics and Mathematics from Duquesne University in 1986, and his Ph.D. in Computer Science from the University of Pittsburgh in 1992. Recently, his research has concentrated on weakening the simplifying assumptions that prevent inductive algorithms from being applied successfully to real-world problems.