Chapter 1

DETECTING FRAUD IN THE REAL WORLD

Michael H. Cahill
Lucent Technologies
535 Mountain Ave
New Providence, NJ 07974
mikecahill@lucent.com

Diane Lambert
Bell Labs, Lucent Technologies
600 Mountain Ave
Murray Hill, NJ 07974
dl@bell-labs.com

José C. Pinheiro
Bell Labs, Lucent Technologies
600 Mountain Ave
Murray Hill, NJ 07974
jcp@bell-labs.com

Don X. Sun
Bell Labs, Lucent Technologies
600 Mountain Ave
Murray Hill, NJ 07974
dxsun@bell-labs.com

Abstract Finding telecommunications fraud in masses of call records is more difficult than finding a needle in a haystack. In the haystack problem, there is only one needle that does not look like hay, the pieces of hay all look similar, and neither the needle nor the hay changes much over time. Fraudulent calls may be rare like needles in haystacks, but they are much more challenging to find. Callers are dissimilar, so calls that look like fraud for one account look like expected behavior for another, while all needles look the same. Moreover, fraud has to be found repeatedly, as fast as fraud calls are placed, the nature of fraud changes over time, the extent of fraud is unknown in advance, and fraud may be spread over more than one type of service. For example, calls placed on a stolen wireless telephone may be charged to a stolen credit card. Finding fraud is like finding a needle in a haystack only in the sense of sifting through masses of data to find something rare. This paper describes some issues involved in creating tools for building fraud systems that are accurate, able to adapt to changing legitimate and fraudulent behavior, and easy to use.

Keywords: Customer Profiles, Customer Relationship Management, Dynamic Databases, Incremental Maintenance, Massive Data, Sequential Updating, Transaction Data, Thresholding.

1. BACKGROUND

Fraud is a big business. Calls, credit card numbers, and stolen accounts can be sold on the street for substantial profit. Fraudsters may subscribe to services without intending to pay, perhaps with the intention of re-selling the services, or even the account itself, at a low cost until shut down. Call sell operations may extend their lives by subverting regulatory restrictions that are in place to protect debtors. Gaining access to a telephone or telephone line by physical intrusion still accounts for some fraud. Fraudsters also focus on the people who use and operate the network by applying “social engineering” to instruct an unsuspecting subscriber or operator to unknowingly agree to carry fraudulent traffic. Large profits have justified the growth of a well-organized and well-informed community of fraudsters who are clever and mobile. Fraud is also important to shady organizations that want to communicate without leaving records of their calls that can be traced back to them. Domestically, *Telecom and Network Security Review* (Vol. 4(5), April 1997) estimates that fraud losses in the U.S. telecommunications industry amount to between 4% and 6% of revenue. Internationally, the figures are generally worse, with several new service providers reporting losses over 20%.

Many service providers respond by building fraud control centers. They acquire multimillion dollar network and operations systems, hire and train staff for 24-by-7 operations, educate customers, require the use of Personal Identification Numbers, partner with competitors and law enforcement agencies, perform internal audits, and constantly tune their operations. Automated fraud detection systems may detect calls to certain “hot numbers”, simultaneous use of calling cards in distant locations, which is unlikely except in the case of fraud, or other patterns of usage that are known to be associated with fraud. Such efforts have helped to reduce fraud, but the set of fraudsters is continually replenished and fraudsters have been able to continue to operate.

Detecting fraud is hard, so it is not surprising that many fraud systems have serious limitations. Different systems may be needed for different kinds of fraud (calling card fraud, wireless fraud, wireline fraud, subscription fraud), each system having different procedures, different parameters to tune, different database interfaces, different case management tools and different quirks and features. Fraud systems may be awkward to use. If they are not integrated with billing and other databases, then the fraud analyst may waste time on simple tasks, such as pulling relevant data from several disparate databases. Many systems have high false alarm rates, especially when fraud is only a small percentage of all traffic, so the chance of annoying a legitimate customer with a false alarm may be much higher than the chance of detecting fraud. More elaborate systems, such as those based on hidden Markov models, may promise more accuracy, but be useless for realtime detection for all but the smallest service provider. Finally, fraud and legitimate behavior constantly change, so systems that cannot evolve or "learn" soon become outdated. Thus, there is a need for tools for designing accurate and user-friendly systems that can be applied to detecting fraud on different kinds of telecommunications services, that can scale up or down, and that can adapt to the changing behavior of both legitimate customers and fraudsters.

Overwhelming data complicates each step of the design and implementation of a fraud management system, where overwhelming is defined not in terms of an absolute standard but relative to the available computing resources. There can be hundreds of millions of call records available for designing the detection algorithms, but there is no need for real-time performance during the design stage and researchers and developers often have access to powerful computing. In production, there may be only millions of calls per day, but each call has to be screened for signs of fraud quickly, faster than calls are being placed, or else the system may fall behind the traffic flow. Moreover, the computing environment may be designed for processing bills rather than for complicated numerical processing, limiting the kinds of algorithms and models that can be used to detect fraud. Once an account is flagged for fraud, all the calls for the case may need to be re-analyzed to prioritize the cases that require human intervention or analysis. There may not be a huge number of calls in an account with suspicious activity, but specialized algorithms for fitting complex models that take call history and account information into account may be needed to pinpoint fraud accurately. If the case is opened for investigation, then thousands of historical and current call records and other kinds of business and account history may need to be considered to determine the best response. Case management tools are needed to help the analyst sift through that data. All these stages are important, all involve data that can overwhelm the resources available, but the data requirements of each are very different.

This paper begins by considering the heart of a fraud management system: the fraud detection algorithm. Simply stated, a fraud detection algorithm has two components: (1) a summary of the activity on an account that can be kept current and (2) rules that are applied to account summaries to identify accounts with fraudulent activity. Section 2 describes these components for threshold based fraud detection. Section 3 describes our approach, which is based on tracking each account's behavior in realtime.

Identifying possible cases of fraud automatically is usually not the last step in fraud detection. Often, the fraud cases need to be prioritized to help a supervisor determine which possible case of fraud should be investigated next. The performance of a fraud management system then depends on both the detection step and the prioritization step. The latter step tends to be ignored by fraud system designers, which can lead to unrealistic estimates of performance. Realistic performance analysis is discussed in Section 4. Final thoughts on fraud detection are given in Section 5.

2. FRAUD DETECTION BASED ON THRESHOLDING

Summarizing account activity is a major step in designing a fraud detection system because it is rarely practical to access or analyze all the call records for an account every time it is evaluated for fraud. A common approach is to reduce the call records for an account to several statistics that are computed each period. For example, average call duration, longest call duration, and numbers of calls to particular countries might be computed over the past hour, several hours, day, or several days. Account summaries can be compared to thresholds each period, and an account whose summary exceeds a threshold can be queued to be analyzed for fraud. Summaries over fixed periods resemble the aggregations of calls used in billing, so software for threshold based fraud detection is not difficult to write or manage. The summaries that are monitored for fraud may be defined by subject matter experts, and thresholds may be chosen by trial and error. Or, decision trees or machine learning algorithms may be applied to a training set of summarized account data to determine good thresholding rules.

Systems based on thresholding account summaries are popular, perhaps because they are easy to program and their logic is easily understood. Thresholding has several serious disadvantages, however. First, thresholds may need to vary with time of day, type of account, and type of call to be sensitive to fraud without setting off too many false alarms for legitimate traffic. Consequently, multivariate rather than univariate statistics must be thresholded. The need for sensitivity and specificity can easily lead to thousands of thresholds that interact with each other and need to be initialized, tuned, and periodically reviewed by an expert to accommodate changing traffic patterns. Accounts with high calling rates or unusual, but legitimate, calling patterns may regularly exceed the thresholds, setting off false alarms. Raising the thresholds reduces the false alarm rate but increases the chances of missing fraud cases. Classifying accounts into segments and applying thresholds to each segment separately may improve performance, but at the expense of multiplying the number of thresholds that have to be managed. Additionally, rules applied to summaries over fixed periods cannot react to fraud until the period is over nor consider calls from previous periods. Such arbitrary discontinuities in account summaries can impair the performance of the system. Perhaps most importantly, sophisticated fraudsters expect thresholds to apply and restrict their activity on any one account to levels that cannot be detected by most thresholding systems. Thus, there are both too many false alarms for legitimate calling and too many missed cases of fraud.

Thresholding can be improved, though. For example, Fawcett and Provost, 1997 develop an innovative method for choosing *account-specific thresholds* rather than universal thresholds that apply to all accounts or all accounts in a segment. Their procedure takes daily traffic summaries for a set of accounts that experienced at least 30 days of fraud-free traffic before being hit by fraud and applies a machine learning algorithm to each account separately to develop a set of rules that distinguish fraud from non-fraud for the account. Thus, each account has its own set of rules at this point. The superset of the rules for all accounts is then pruned by keeping only those that apply to or *cover* many accounts, with possibly different thresholds for different accounts. For example, the rule may specify that long international calls indicate fraud, where long might be interpreted as more than three standard deviations above the mean duration for the account during its period of fraud-free traffic. Pruning then proceeds sequentially: a candidate rule is added to the current set of rules if it applies to a minimum number of accounts that have not already been covered by a specified number of rules. The final set of rules, therefore, covers "most" accounts, with the understanding that most of the final rules may be irrelevant for most accounts, but all the final rules are relevant for at least some accounts.

A fraud detection system based on account-specific thresholds is straightforward to implement for established accounts. The calls for the period of interest are separated by account, account summaries are computed, and then account summaries are compared to account-specific thresholds that were previously computed from training data. This process is similar to billing, which also requires account aggregation and access to account information. The account-specific thresholds can be updated periodically by re-fitting trees and sequentially selecting the account summaries to threshold. Re-training requires more resources than running the detection algorithm does, but re-training may be needed infrequently. Fawcett and Provost, 1997 describe an application of their methods to a set of fewer than 1,000 accounts, each of which had at least 30 days of fraud-free activity followed by a period of wireless cloning fraud.

Account-specific thresholding has limitations, though. Perhaps most importantly, a procedure that requires a fixed period, such as 30 days, of uncontaminated traffic for training does not apply to accounts that experience fraud before the training period is over. In subscription fraud, for example, all the calls for an account are fraudulent, so there is no fraud-free period. Moreover, rules that are good for one time period may not be relevant for future time periods because account calling behavior, both fraudulent and legitimate, changes over time. And the basic limitations of thresholding—looking for fraud only at the end of a period and basing fraud detection on calls in only the current period—still apply. Nonetheless, a method that automatically derives account specific thresholds is clearly an important advance in threshold-based fraud detection.

3. FRAUD DETECTION BASED ON TRACKING ACCOUNT BEHAVIOR

3.1 ACCOUNT SIGNATURES

Like Fawcett and Provost, 1997, we believe that fraud detection must be tailored to each account's own activity. Our goals for fraud detection are more ambitious, though. First, fraud detection should be *event-driven*, not time-driven, so that fraud can be detected as it is happening, not at fixed points in time that are unrelated to account activity. Second, fraud detection methods should have *memory* and use all past calls on an account, weighting recent calls more heavily but not ignoring earlier calls. Third, fraud detection must be able to *learn* the calling pattern on an account and *adapt* to legitimate changes in calling behavior. Fourth, and perhaps most importantly, fraud detection must be *self-initializing* so that it can be applied to new accounts that do not have enough data for training.

The basis of our approach to fraud detection is an account summary, which we call an *account signature*, that is designed to track legitimate calling behavior for an account. An account signature might describe which call durations, times-between-calls, days-of-week and times-of-day, terminating numbers, and payment methods are likely for the account and which are unlikely for the account, for example. That is, given a vector of M *call variables* $\mathbf{X}_n = (X_{n,1}, X_{n,2}, \ldots, X_{n,M})$ for each call n, the likely (and unlikely) values of $\mathbf{X}_n$ are described by a multivariate probability distribution P_n, and an *account signature* is an estimate of P_n for the account. Because fraud typically results in unusual account activity, P_n is the right background to judge fraud against.

Estimating the full multivariate distribution P_n is often impractical, both in terms of statistical efficiency and storage space, so a major task in signature design is to reduce the complexity of P_n. To do that, we rely on the law of iterated probability (Devore, 2000), which states that $P_n(\mathbf{X}_n)$ can be expressed as the product

$$\begin{aligned} &P_n(X_{n,1} = x_1)P_n(X_{n,2} = x_2 | X_{n,1} = x_1) \cdots \\ &\quad P_n(X_{n,M} = x_M | X_{n,1} = x_1, \ldots, X_{n,M-1} = x_{M-1}). \end{aligned} \tag{1.1}$$

The first term in the product represents the marginal distribution of $X_{n,1}$, and each successive term is conditional on all the variables that were entered before it. (The order of the variables is arbitrary.) The last term, for example, implies that the distribution of $X_{n,M}$ depends on the outcomes of the other $M - 1$ variables. Thus, there is a different conditional distribution of $X_{n,M}$ for each possible combination of outcomes of all the other variables.

For example, suppose $X_{n,1}$ represents call duration discretized into 10 different intervals, $X_{n,2}$ represents the period of day (peak or off-peak), and $X_{n,3}$ represents direction (incoming or outgoing). Then equation (1.1) requires 31 terms: one for the marginal distribution of duration, 10 for the conditional distributions of time of day given duration, and 20 for the conditional distribution of direction given each possible combination of duration and time of day. Some of these terms might be redundant, however. For example, if the probability that a call is incoming rather than outgoing is different for peak and off-peak hours but independent of call duration, then there are only 2 conditional distributions for call direction, not 20. Then the account signature $\mathcal{A}_n$, which is an estimate of P_n, would be the product of 13 estimated distributions, rather than 31 estimated distributions. Each term in the product $\mathcal{A}_n$ is called a *signature component*. Each signature component has a *signature variable* X_m and, possibly, a set of *conditioning variables*. The set of all signature components summarizes our knowledge about calling behavior, ignoring only those relationships among variables that are unimportant.

3.2 SIGNATURE DESIGN

In the applications that we have seen, fast processing has depended on allocating the same, small amount of space to each account signature. Controlling the size of a signature can also contribute to its accuracy and precision. For example, suppose the duration of a call is the same for peak and off-peak hours, but a separate signature component is (needlessly) reserved for each. Then a peak call will not be used to update the signature component for off-peak duration, even though it would be statistically appropriate to do so. Consequently, the signature component for off-peak duration will be estimated from a smaller sample size than it should be, leading to statistically inefficient estimates. Designing a signature amounts to eliminating conditioning variables that do not matter and controlling the amount of space devoted to each remaining term in the product (1.1).

Like most fraud detection systems, we assume that there is a set of *priming data* that consists of all calls for a large number of accounts over a fixed period (say 60 days) that can be used to design signatures. We also assume that we have call records for a second set of accounts that experienced fraud during the period. For subscription fraud, all the calls on the account are fraudulent. For other kinds of fraud, some calls are fraudulent and some are not. Ideally, the fraudulent calls are labelled; otherwise they have to be labelled "by hand". In any case, because one fraudster often affects more than one account and we are interested in characterizing fraud, not a particular fraud user, we collapse all the fraud records for any particular kind of fraud into one set of *target data* and, ultimately, into one *fraud signature*. There may be separate fraud signatures for wireless subscription fraud, calling card fraud, and cloning fraud, but not a separate signature for each case of subscription fraud, for example. Each fraud signature has the same structure as an account signature.

The first design step is to choose the type of representation to be used for each signature variable. A continuous variable, such as call duration or time between calls, might be described by a parametric distribution, reducing each signature component for duration to a vector of estimated parameters, such as a mean and a standard deviation. But often no parametric family is both flexible enough to fit all accounts well and tractable enough to estimate quickly (say, during call setup or teardown). Therefore, we generally take all signature components to be nonparametric. In particular, a continuous variable or ordered categorical variable with many possible values can be discretized, so its signature components are vectors of probabilities over a set of fixed intervals. Alternatively, a continuous variable can be represented as the coefficients of a fixed set of knots for a spline fit to the log density or as a vector of quantiles. A signature variable with many possible unordered categories, such as terminating number or area code, can be represented by the labels and probabilities of the most likely categories. This kind of representation resembles a histogram, but one in which the labels of the bins are not fixed. For illustration, all signature components are assumed to be represented by histograms in this paper.

There are many criteria for defining histogram bins; see Gibbons, Matias and Poosala, 1997 and Ioannidis and Poosala, 1999, for example. Many of these criteria are designed to give good performance over all possible values of the signature variable, but in fraud detection only small probability events are important because only small probability events are able to indicate fraud. Since our goal is fraud detection, we choose the bins of the histogram so that, on average, it is as easy as possible to distinguish legitimate calls from fraud calls. For a given signature variable, this can be accomplished by maximizing the average weighted Kullback-Leibler (AWKL) distance from the histogram for an account in the priming data ($p_{i,k}$) to the histogram for the fraud data (f_k), where

$$\mathrm{AWKL} = -\frac{1}{N}\sum_{i=1}^{N}\left(w\sum_{k=1}^{K} p_{i,k}\log\frac{p_{i,k}}{f_k} + (1-w)\sum_{k=1}^{K} f_k \log\frac{f_k}{p_{i,k}}\right)$$

for some w, $0 \leq w \leq 1$, K is the number of bins in the histogram, and N is the number of accounts in the priming data (Chen, Lambert, Pinheiro and Sun, 2000). The weight w controls the balance between more informative description of legitimate behavior and better separation of fraud and legitimate behavior. When $w = 0$, the criterion ignores the fraud training data and emphasizes only the ability to represent the behavior of legitimate accounts well. When $w = 1$, the criterion considers only the ability to avoid false alarms. Intermediate values of w balance these two concerns.

The cutpoints $d_1, \ldots, d_{k-1}$ that maximize the AWKL criterion can be found by numerical search, if feasible. If exhaustive search is not feasible, then searching can be limited by requiring minimum widths for the K final bins. For example, call duration might be measured to the nearest second, but each bin might be required to be at least one minute long and endpoints might be restricted to integer minutes.

The AWKL distance can also be used when a signature variable X is represented by something other than a histogram. For example, if the signature represents a continuous distribution with probability density $p_i(x)$ for account i and probability density $f(x)$ for fraud, then the sum over i in the AWKL distance is replaced by an integral over x. An appropriate set of parameters maximizes the integral form of AWKL. If X is represented by a vector of quantiles, then a change of variables shows that $\int_x p(x)\log(p(x))dx = \int_q \log(p(P^{-1}(u))du$, where P^{-1} is the quantile function defined by $P^{-1}(u) = q$ if $\int_{-\infty}^{q} p(x)dx = u$, so the AWKL criterion still applies. The quantiles to be used in the signature are again those that maximize AWKL.

Methods for deciding which conditional distributions to keep in a signature are discussed in detail in Chen, Lambert, Pinheiro and Sun, 2000. The basic idea involves computing a *p*-value for a χ^2 test for each account in the training data and keeping only the conditioning variables that are statistically significant for a majority of accounts and highly statistically significant for at least some accounts. The other possible conditioning variables add only noise rather than predictive power to the signature of most accounts. Conditioning variables are added sequentially until the incremental benefit from any of the remaining variables is too small to be statistically significant for a majority of accounts. Loosely stated, a conditioning variable is kept only if it is important for describing many accounts, and very important for at least some accounts.

3.3 KEEPING A SIGNATURE CURRENT

A key feature of a signature is that it can be updated sequentially, so that it evolves in time (measured in calls) rather than abruptly changing at the end of an arbitrary period. This enables event-driven fraud detection because there is always an up-to-date standard against which fraud can be assessed. Sequential updating is also computationally efficient because it does not require access to a data warehouse, which is often slow, but only access to a data structure that is short enough to store in main memory. In the wireline, calling card and wireless fraud detection systems in which we have applied signatures, each signature can be stored in about the amount of space required to store one call, with careful quantization of probabilities.

Most signature components, such as duration or method of payment, can be considered to be randomly sampled. Thus, they can be updated by exponentially weighted moving averaging. For example, suppose the signature component is a vector of probabilities, call $n + 1$ is represented by a vector $\mathbf{X}_{n+1}$ of 0's except for a 1 in the bin that contains the observed value of the call, and A_n is the account's signature component after the previous call n. Then the updated signature component based on call $n + 1$ is

$$A_{n+1} = (1 - w)A_n + w\mathbf{X}_{n+1},$$

where w determines the rate at which old calls are "aged out" of the signature component and the effect of the current call on the component. If $w = .05$, the probability assigned to the observed bin increases by the constant amount $.05$ and the probability of any other bin decreases by a factor of .95. Also, call $n - 10$, which was 10 calls earlier than call n, has about 60% the weight of call n at the time of call n if $w = .05$, and about 82% the weight of call n if $w = .02$. The smaller w, the more stable the signature component. Of course, some care has to be taken to avoid incorporating fraud into the signature; this is discussed in Section 3.4 below.

A variant of exponentially weighted stochastic approximation, which is similar in computational effort to exponentially weighted moving averaging, can be used to update signature components that are vectors of quantiles (Chen, Lambert and Pinheiro, 2000). Signature components that are represented as tail probabilities rather than as complete distributions can be updated by using a move-to-the-front scheme to update the labels of the top categories and a variant of exponentially weighted moving averaging to update their probabilities. (See Gibbons, Matias and Poosala, 1997 for the details of one possible approach.) Timing variables, like day-of-week and hour-of-day, are not randomly observed because they are observed "in order"—all the calls for Tuesday of this week are observed before all the calls for Wednesday of this week. Nonetheless, timing distributions can be updated in a way that is similar in spirit and in amount of computation to exponentially weighted moving averaging (Lambert, Pinheiro and Sun, 1999).

Of course, sequential updating requires a starting point or *initial signature* for a new account. One way to do that is to segment the signatures for the accounts in the priming data, basing the segmentation criteria on information in the first one or few calls in an account. Details of one procedure based on statistical testing are given in Chen, Lambert, Pinheiro and Sun, 2000; details of another procedure that uses multivariate regression trees are given in Yu and Lambert, 1999. These procedures initialize each signature component separately. This gives a huge number of possible initial signatures (products of initial signature components), expressing a huge number of possible calling patterns. That is, a newly active account is assigned to a different segment of customers for each signature component, and each assignment depends only on the first few calls on the account. Finally, note that it is the initialization of signatures from the calls for a set of legitimate accounts, rather than from the calls on the account alone, that enables us to detect subscription fraud, in which every call on an account is fraudulent. Moreover, call-by-call updating ensures that an account with many calls soon evolves to its own "segment", allowing for personalized customer relationship management.

3.4 USING SIGNATURES TO DETECT FRAUD

Scoring a call for fraud is a matter of comparing its probability under the account signature to its probability under a fraud signature. Suppose the account signature after call n is $\mathcal{A}_n$, the fraud signature is $\mathcal{F}$, which is independent of the number of calls on the account, and call $n+1$ with signature variables $\mathbf{x}_{n+1}$ is observed. Then the *call score* for call $n+1$ is defined by

$$C_{n+1} = log(\mathcal{F}(\mathbf{x}_{n+1})/\mathcal{A}_{n+1}(\mathbf{x}_{n+1})).$$

Because $\mathcal{A}_{n+1}$ and $\mathcal{F}$ are products of signature components, the call score is a sum of contributions from the signature components. Note that standard statistical theory implies that the log likelihood ratio $\mathcal{C}_{n+1}$ is the best discriminator of fraud ($\mathcal{F}$) from legitimate activity on the account ($\mathcal{A}_n$) when $\mathbf{x}_{n+1}$ is observed (Bickel and Doksum, 1976).

The higher the call score, the more suspicious the call. For a call to obtain a high score, it has to be unexpected for the account, so $\mathcal{A}_{n+1}(\mathbf{x}_{n+1})$ must be small. Calls that are not only unexpected under the account signature but also expected under the fraud signature score higher, and are considered more suspicious, than calls that are unexpected for both the account and fraud. Thus, some departures from the signature are more interesting than others from the perspective of fraud management. As constructed, each signature component contributes equally to the fraud score because the multivariate distribution that predicts call $n+1$ is a product of the marginal and conditional distributions represented by the signature components. As a result, some signature components can counteract others. This is not unreasonable. An hour long wireless call may be less suspicious if it originates from a region that is often used by the account. It is, however, possible to weight different signature components differently. Weights might depend on the reliability of the estimated distribution, some subjective information about the value of the signature component for fraud detection, or tuning that optimizes performance on training data. Note that the choice of the fraud distribution also affects the call score. In particular, a uniform fraud distribution implies that all unexpected call characteristics are equally good indicators of fraud.

Call scores serve two purposes. One is to give information that can be used to identify *accounts* that may have fraudulent activity. The other is to identify *calls* that may be suspicious and so should not be used to update the signature. For example, negative scores suggest that the call is not fraudulent so it should be used to update the signature. Calls with high positive scores raise concerns about fraud and, to be safe, should not be used to update the signature. Small positive scores are ambiguous. They might suggest a slight change in calling pattern that resembles fraud, or they might suggest that there is a subtle case of fraud. This ambiguity suggests the following procedure: update the signature if the call score is negative, do not update the signature if the call score is high, and act probabilistically if the call score is positive but small. In the latter case, a signature is updated with a probability that depends on the call score, varying from probability one for a call score that is less than or equal to zero to probability zero for a call score that is sufficiently high.

Egregious cases of fraud may generate calls with scores so high that a service provider may be willing to declare that fraud has occurred with only one call. Usually, however, the evidence from one call alone is not sufficient to identify fraud. If not, it is important to monitor the *score rate*, where score rate is either the average score over the last several calls that have a score above a pre-determined threshold or the average score of calls above a threshold over a specified time period. Some minimal information about previous calls needs to be kept in the signature to calculate the score rates. Accounts with high score rates can then be examined for fraud. The thresholds are not applied to all aggregated calls at the end of a period but rather to aggregated *high scoring* calls at the time of a possibly fraudulent call. We say that an account that exceeds the thresholds on score rates is *flagged*. Flagging filters the set of all accounts, producing a smaller set of accounts that have some evidence of fraud, just as standard account thresholding filters.

Both call scoring and account flagging have parameters that can be thought of as thresholds, but there is a major difference with the kinds of thresholds discussed in Section 2. In standard account thresholding, an account is marked as suspicious if it passes any one of several thresholds. In signature-based thresholding, different thresholds are not developed for different call characteristics or different kinds of accounts, such as business accounts or residential accounts. Instead, thresholds are applied to log-likelihood ratios that are a combination of all call characteristics. In other words, taking log-likelihood ratios converts all the different call characteristics and types of accounts to a standard measurement scale with one common set of thresholds.

Nonetheless, there are thresholds that need to be tuned in a signature-based fraud detection system. These parameters may be chosen to minimize the chance of flagging a legitimate account as fraudulent, subject to a constraint on the probability of missing a legitimate case of fraud in a set of training data. Because there are only a limited number of parameters to tune, the optimal parameters can be found by searching over a grid.

In summary, signatures are the basis of event-driven, adaptive, self-initializing fraud detection. It is event-driven in the sense that it is applied at the time of each call, which is the only time that fraud is active. It is self-learning, in the sense that the basis for comparison, the signature, is updated with each call that is not judged to be suspicious. Moderate changes in behavior are learned more slowly, on average, because the rate at which shifts in behavior are incorporated into a signature depends on the size of the shift. Signature-based fraud detection is also self-initializing in the sense that the first call or two on the account is used to assign signature components to new accounts. Because the signature components are initialized with calling patterns for previous accounts without fraud, it is possible to detect subscription fraud for new accounts. In a sense, the automatic initialization step allows us to start with a procedure for new accounts that is akin to universal thresholding with a huge number of segments. The procedure then naturally evolves to a procedure that is akin to customer specific thresholding for established accounts. Moreover, the threshold limits are placed on likelihood ratios and hence are the same for all segments, thus greatly reducing the number of parameters that have to be tuned in advance.

4. PERFORMANCE METRICS

The performance of a fraud system is ultimately determined by the losses a service provider is able to prevent, but measuring averted losses, which never occur, is clearly difficult if not impossible. So, instead, service providers use metrics like the detection rate, false alarm rate, average time to detection after fraud starts, and average number of fraud calls or minutes until detection. An ideal fraud detection system would have 0% false alarms and 100% hits with instantaneous detection. But, finding all cases of fraud as soon as fraud starts requires mislabeling many (if not most) legitimate accounts as fraudulent at least once. A realistic, practical fraud system strikes a satisfactory balance of the performance criteria.

First, however, it is important to define the metrics carefully. Traditionally, the false alarm rate has been defined to be the percentage of legitimate accounts mislabelled as fraud. (False alarm rates, like type II errors, are usually quoted as percents, rather than as rates.) If there are 1,000,000 legitimate accounts in the population and 100 of these accounts are falsely labeled as fraud, then the false alarm rate is .01%. False alarms are important at the flagging stage because the goal of that step is to reduce the set of accounts in the population that have to be considered for fraud to just those that had fraud. False alarms are also important if account activity is restricted for flagged accounts because then the goal is to keep the number of legitimate accounts in the population that are needlessly restricted as small as possible.

If the evidence for fraud is ambiguous or attracting customers is costly, the service provider may require that a fraud analyst investigate the case before activity on the account is restricted. In that case, there is at least one queue of flagged accounts and the highest priority case in the queue is investigated whenever a fraud analyst becomes available. A queue may prioritize accounts by the number of fraudulent minutes accumulated to date or by the time of the most recent high scoring call, for example. Performance can then be evaluated after flagging or after prioritization. For example, the *flagging detection rate* is the fraction of compromised accounts in the population that are flagged. The *system detection rate*, which includes the rules used to decide which flagged accounts to open, is the fraction of compromised accounts in the population that are investigated by a fraud analyst. The system and flagging detection rates are equal only when every flagged case of fraud is investigated. Otherwise, the system detection rate is smaller than the flagging detection rate because both detection rates are computed relative to the number of accounts with fraud in the population.

Service providers are also keen to know that their analysts are working on fraud cases, not investigating legitimate accounts. Thus, a different question is what fraction of investigated cases have fraud? The *flagging hit rate* is the fraction of flagged accounts that have fraud, and the *system hit rate* is the fraction of investigated cases that have fraud. One minus the system hit rate is often a good measure of the service provider's perception of the "real false alarm rate," especially since this is the only error rate that the service provider can evaluate easily from experience. That is, only the cases that are acted upon may be of interest to the service provider, not the legitimate cases in the population that were never judged to be suspicious. If 20 cases of fraud are investigated and only 8 turn out to be fraud, then a service provider may feel that the "real false alarm rate" is 60%, even if only .01% of the legitimate accounts in the population are flagged as fraud.

Typically, the flagging false alarm rate, which is computed relative to all legitimate accounts, is much smaller than one minus the system hit rate because so many of the accounts in the population never experience fraud and so few accounts are investigated. Note that the hit rate after the flagging step should be larger than the fraction of accounts in the population that have fraud. Otherwise, flagging is no better than randomly labeling accounts as fraudulent. The difference between the fraction of fraud in the population and the fraction of fraud in flagged accounts is a measure of the efficiency of the fraud detection algorithm. Similarly, the system hit rate should be larger than the flagging hit rate, or else the analyst can find as much fraud by randomly selecting one of the flagged accounts to investigate.

The flagging false alarm rate, detection rate, and hit rate can be estimated by applying the flagging algorithm to a set of training accounts, some of which have fraudulent activity. The only subtlety is that these rates vary over time and should be investigated as a function of time. If there are L_t legitimate accounts on day t and $L_{1,t}$ of these are flagged, then the flagging false alarm rate for day t is $L_{1,t}/L_t$. If there are X_t active cases of fraud on day t and $X_{1,t}$ of these cases are flagged, then the flagging detection rate for day t is $X_{1,t}/X_t$. The flagging hit rate is then $X_{1,t}/(L_{1,t} + X_{1,t})$. Because legitimate and fraudulent behavior is learned over time in a signature-based fraud detection system, each of these performance statistics should improve quickly when the system is still new and then stabilize if the number of new accounts added stabilizes over time.

Some care needs to be taken in counting accounts when estimating system performance metrics. After an account is flagged and put into a priority queue, it can either be opened for investigation by a fraud analyst, it can remain in the queue without exceeding flagging thresholds again, or it can remain in the queue and continue to cross flagging thresholds. If the case is opened, the account is removed from the queue and the analyst decides, perhaps after contacting the account owner, if fraud has been committed. If an account remains in the queue unopened and it is not flagged again, then eventually it is considered uninteresting and reaped from the queue. The simplest rule is to reap an account if it has not been opened and has not been flagged again for a specified a number of days. If an account that is already queued is flagged again, then its priority needs to be re-computed to reflect the continuing suspicious activity.

Simulating system performance, then, requires simulating the prioritization and reaping processes. In practice, re-prioritization may occur whenever another account is flagged, but for simulation purposes it is enough to re-prioritize accounts once a day, for example. Then, for each day, the number of accounts opened, the number of accounts with fraud opened, and the number of active fraud cases can be counted to compute the system hit rate and system detection rate, which are typically the most important parameters to service providers.

Finally, note that realistic performance assessment requires assuming realistic levels of fraud. Performance often appears better for larger fraud rates, but if typically 4% to 6% of all accounts are infected by fraud annually, then assuming that 20% to 30% of all accounts are infected by fraud in three months is not realistic and overstates system performance.

5. FURTHER THOUGHTS

This paper describes an approach to fraud detection that is based on tracking calling behavior on an account over time and scoring calls according to the extent that they deviate from that pattern and resemble fraud. Signatures avoid the discontinuities inherent in most threshold-based systems that ignore calls earlier than the current time period. In the applications that we have encountered, a signature can be designed to fit in about the space needed to store one call. Consequently, they can be stored in main memory that is quick to access, rather than in a data warehouse that is slow to access. In one application to wireless fraud detection, we had 33 gigabytes of raw wireless call records for about 1.5 million customers. Each signature required 200 bytes, so the signature database required only about 300 megabytes to store. In an application to domestic wireline calling, there were 90 gigabytes of call records for about 1 million customers. Each signature required only 80 bytes to store, so the signature database required only about 80 megabytes. Because signatures are updated call-by-call, there is no need for offline processing or for using out-of-date customer profiles. Thus, signatures are able to avoid two common complaints about fraud detection systems that profile customers: they cannot scale up and they cannot keep up with the volume of data. Moreover, signatures are initialized from legitimate data from a huge number of possible initializations, so they can detect subscription fraud. This is in contrast to profiling systems, which use only the calls on the account itself to profile customers and, therefore, cannot detect subscription fraud. Finally, the signature evolves with each new call that is not considered fraudulent, so each established customer eventually has its own signature, not a signature designed for a segment of customers. Evolving the customer from an initial segment to a personalized segment is painless—no additional processing, such as re-clustering a database, is necessary.

It is possible to put other kinds of fraud detection algorithms in the signature framework. For example, many service providers keep lists of “hot numbers” that are associated with fraud. These can also be used in a signature-based system, by giving each call a high score when a hot number is called. More importantly, perhaps, it is possible to have scores for warm numbers that are often but not exclusively associated with fraud. These numbers can be assigned a contribution to the log-likelihood that is smaller than that for hot numbers, for example. It is also possible to keep account characteristics that can be derived from call records and that might be useful for fraud detection in the signature.

For example, the fact that an account is a coin phone might change the rate at which it is attacked by fraud or the kind of fraud it is likely to be subjected to.

There is much more to fraud detection than the algorithms that score calls and label accounts as suspicious, though. For example, the fraud detection system needs to be able to access calls at the switch, before they are sent to a data warehouse, in order to be real-time or nearly real-time. Putting hooks into a telecommunications network to pull call data from a switch can be extremely difficult. After an account is flagged, investigators need sophisticated tools for case management. These tools must be integrated with billing systems so that the investigator can access payment history. The case management tools must also be integrated with service provisioning systems so that an investigator can enable service restrictions on the account, if appropriate. Putting hooks into these systems is non-trivial at best. Supervisors then need case management tools that allow them to track the performance of the system and to spot new trends in fraud. Tools are needed to ensure that if one account for a customer is closed for fraud, then all accounts for that customer are closed for fraud, for example.

Signatures can be adapted to any kind of fraud in which transactions are made on accounts. This includes credit card fraud, in which the transaction is a purchase, and medical fraud, where the account may be a medical practitioner and the transaction an interaction with a patient or the account may be a patient and the transaction a visit to a medical practitioner. More generally, signatures can be used to predict transaction behavior on accounts. For example, signatures can be used in to track the behavior of visitors at web sites to identify those who are about to make purchases. While further research is needed to work out the details for particular applications, the concept of signature-based predictive tracking is broad and potentially valuable for a wide range of applications.

ACKNOWLEDGEMENTS

At various times, our thoughts on fraud detection have been influenced by discussions with many people, especially Jerry Baulier (Lucent Technologies), Rick Becker (AT&T Labs), Fei Chen (Bell Labs, Lucent Technologies), Linda Clark (Bell Labs, Lucent Technologies), Virginia Ferrara (Lucent Technologies), Al Malinowski (AT&T), Daryl Pregibon (AT&T Labs) and Allan Wilks (AT&T Labs).

References

Bickel, P. J. and Doksum, K. A. (1976). *Mathematical statistics*. Holden-Day, San Francisco, CA.

Chen, F., Lambert, D., and Pinheiro, J. C. (2000). Sequential percentile estimation. Technical memorandum, Bell Labs, Lucent Technologies.

Chen, F., Lambert, D., Pinheiro, J. C., and Sun, D. X. (2000). Reducing transaction databases, without lagging behind the data or losing information. Technical memorandum, Bell Labs, Lucent Technologies.

Devore, J. L. (2000). *Probability and Statistics for Engineering and the Sciences*. Wadsworth, Belmont, CA, 5th edition.

Fawcett, T. and Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1:291–316.

Gibbons, P. B., Ioannidis, Y. E., and Poosala, V. (1997). Fast incremental maintenance of approximate histograms. In *Proceedings of the 23rd VLDB Conference*, pages 466–475.

Ioannidis, Y. E. and Poosala, V. (1999). Histogram-based approximations to set-valued query answers. In *Proceedings of the 25th VLDB Conference*, pages 174–185.

Lambert, D., Pinheiro, J. C., and Sun, D. X. (1999). Updating timing profiles for millions of customers in real-time. Technical memorandum, Bell Labs, Lucent Technologies.

Yu, Y. and Lambert, D. (1999). Fitting trees to functional data, with an application to time-of-day patterns. *Journal of Computational and Graphical Statistics*, 8(4):749–762.