Reducing Transaction Databases, Without Lagging Behind the Data or Losing Information

Fei Chen, Diane Lambert, José C. Pinheiro, and Don X. Sun*

Abstract

This paper describes a way to reduce a database of transaction records when interest lies in the behaviors of the people making the transactions. A fixed length data structure, called a signature, is used to summarize the complete, multidimensional behavior of each individual, giving a database that is linear in the number of individuals instead of the number of transactions. The signature structure can be used with both static and dynamic databases. With dynamic databases, a signature can be updated with every transaction that the individual makes, so that the signature adapts reliably and efficiently to changing behavior. The structure of a signature and the initial set of signatures to be assigned to individuals with no past transaction history are derived automatically from a set of past transactions for a set of individuals. An application to wireless calling patterns, in which each signature is limited to about the size of one call record and must be updated at the end of each call, is used to motivate the ideas.

1 Background

Increasingly, large transaction databases are used to understand customer behavior [1]. A business may periodically seek "interesting" associations in customers' patterns of purchases, or it may segment customers into homogeneous clusters for marketing, for example. Or, a business may analyze transaction data in nearly real-time to influence the behavior of customers. An e-commerce web site may want to offer a visitor help from a live agent to increase the chances of a large sale, or it may want to limit a customer's purchases if there is evidence of fraud. Accessing and analyzing continually updated transaction data fast enough to influence customer behavior is challenging, though, unless the data are scaled down to fit in main memory.

The idea of data reduction is not new, of course. See, for example, the review article [2], the notion of sampling from the leaves of a classification tree in [3], the use of univariate frequency distributions, quantiles and "backing samples" in [4] and [5], and "data squashing" or sophisticated weighted sampling for the purposes of logistic regression in [6]. But all these techniques are designed to preserve *global* characteristics of the data, rather than to preserve information at the *customer* level. Also, with the exception of [4], these methods are aimed at static databases rather than dynamic databases that must be updated with each new transaction.

In contrast, this paper describes an approach to reducing dynamic transaction databases when interest centers on the "customers" or individuals who are making transactions. A fixed-length data structure that we call a *signature* summarizes each customer's behavior, giving a database that is linear in the number of individuals rather than the number of transactions. A signature database is easy to maintain because it can be easily updated transaction-by-transaction, tracking customer

*The authors are in the Statistics and Data Mining Research Department of Bell Labs, Lucent Technologies.

behavior reliably and efficiently. Signatures are also optimal, in the sense that they minimize the loss of information about individual customers, under a certain criterion that requires one level of accuracy for most customers and a high level of accuracy for a specified fraction of customers. Methods for determining the structure of a signature, the set of signatures that can be assigned to new customers, and for updating signatures as a customer makes new transactions are also discussed in this paper. An application to wireless calling records motivates the ideas throughout.

2 A Model for Data Reduction

To be concrete, consider a dynamic database of wireless calls. Each transaction or call record contains over twenty variables on a call, such as its date and time, its duration, its direction (incoming or outgoing), the location of the caller, the service provider (roaming or home based), which, if any, special features, such as call waiting, were used, and the number called. These variables contain information about each customer's transaction pattern, and the goal is to represent that pattern succinctly and accurately. The summarization might then be used to answer questions about all customers, such as what fraction of calls occur on weekends, or questions about a particular customer, such as is it unusual for this customer to make a one hour call at 11 p.m. on a Sunday?

Histograms are an easy way to summarize transaction behavior dynamically – just discretize the variable if necessary and update bin counts with each new call that a customer makes. But there are complications. An obvious problem is that the discretization intervals or *buckets* must be chosen, and what is "right" for one customer may be "wrong" for another. Another complication is that standard histograms give equal importance to recent and old transactions, so they cannot represent recent behavior unless they are re-computed periodically. A more challenging and important difficulty is that a customer's pattern of behavior for any one variable may depend on other variables, so a multidimensional histogram may be needed. If the duration of a customer's call is different for incoming and outgoing calls and for peak and off-peak calls, for example, then a three dimensional histogram is needed to represent the customer's call durations accurately. On the other hand, some variables, like the duration of a customer's calls and the location of the caller, may be independent, so storing the full multidimensional histogram for each caller is likely to waste space. Moreover, the full multivariate histogram is likely to be sparsely populated for most customers unless all the variables are severely discretized.

To address issues like choice of buckets, staleness, and multidimensionality, we cast the problem of customer-centric data reduction as estimation of a huge set of multivariate probability models, one for each customer. Each probability distribution describes the likely transaction behavior of a customer and the objective of data reduction is to estimate this probability distribution, quickly and reliably. The estimated distributions can then be used to understand the likely behavior of each customer separately or the likely behavior of a population of customers as a whole. For example, the probability that a customer makes a local call that lasts at least an hour at 11 p.m. Sunday night can be computed from that customer's probability distribution, and the fraction of customers who make at least an hour long call at 11 p.m. Sunday night can be computed from the full set of probability distributions for all customers. Thus, knowledge of the probability distributions for all customers provides answers to many questions about the customer base.

More formally, let $\boldsymbol{X}_n = (X_{n,1}, X_{n,2}, \ldots, X_{n,M})$ represent the vector of M transaction variables for a particular customer at the time of transaction n. Some transaction variables, such as call duration, are explicit in the transaction record; others, such as time from last call, may need to be computed from information in the current and previous transaction records. Let P_n represent the joint probability distribution of $\boldsymbol{X}_n$ at the time of transaction n. Then P_n describes which values of the transaction variables are likely and which are unlikely. In that sense, P_n describes the customer's pattern of behavior at the time of transaction n. The goal of data reduction, then, is to keep enough information in the reduced data to estimate P_n well. Note that the goal is not to give an exact accounting of the customer's past transaction records but rather to keep enough information about P_n to make predictions about the customer and answer statistical questions about the population as a whole. Loosely speaking, we are more interested in predicting the future than in reproducing the past data.

The first task is to build a model for P_n. The law of iterated probability [7] states that P_n can be written as a product of one dimensional distributions:

$$P_n(\boldsymbol{X}_n) = P_n(X_{1,n})P_n(X_{2,n}|X_{1,n}) \ldots P_n(X_{M,n}|X_{1,n}, \ldots, X_{M-1,n}), \tag{1}$$

where $P_n(X_{1,n})$ is the marginal distribution of the first variable for transaction n and $P_n(X_{2,n}|X_{1,n})$ is the conditional distribution of the second variable for transaction n given the first variable for transaction n. The last term in the product implies that the behavior of $X_{M,n}$ depends on all the other variables for transaction n, so there is a one dimensional distribution of $X_{M,n}$ for each combination of possible values of the other variables.

A *signature* is a fixed-length estimate of P_n. To simplify the discussion, we assume that each variable either has at most several possible values or it is discretized into buckets. Consequently, from equation (1), the signature is an estimate of a set of one dimensional discrete distributions. Thus, it is natural to write a signature as a product of *signature components*, each of which is a one dimensional histogram. For example, if only the duration and direction of the call are recorded, then the signature components might be a histogram for call direction (incoming or outgoing), a histogram for the duration of incoming calls, and a histogram for the duration of outgoing calls. Alternatively, if one of the distributions in equation (1) is parametrized, then the corresponding signature component might be the estimated parameter(s) of the distribution rather than a histogram. (Finding a parametric distribution that is appropriate for all or even most customers has proven impossible in the applications that we have encountered, however.) Other nonparametric possibilities include representing each distribution by a set of percentiles or representing the density of each continuous distribution by a set of coefficients from a spline fit for the density [8]. In this paper, we consider only histogram representations, where histograms are understood to be normalized to have a total weight of 1, but extending the ideas and methods in this paper to these other kinds of representations is not difficult.

Generally, using a full set of histograms for each customer, one for each term in the product representation (1) of P_n, takes too much space and is statistically inefficient. Obviously, storing separate histograms for durations of incoming and outgoing calls wastes space if the two histograms are identical. But it is also statistically inefficient because only the incoming calls would be used to update the incoming histogram, even though the outgoing calls in this case would also have information about the distribution of the duration of incoming calls. "Wasting" the outgoing calls in this way is particularly bad for infrequent callers. Finally, it may also be necessary to coarsen a variable to save space. For example, duration may be recorded to the nearest second, but a much coarser representation may be adequate for estimation. The next section discusses signature design to meet space constraints when statistical efficiency needs to be taken into account.

3 Signature Design

The set of *priming data* is a collection of transaction records collected from a representative set of customers during a fixed time period that can be used to design signatures. If the signatures are to be used primarily for one application, such as identification of big spenders or detection of fraud, the priming data is assumed to include transaction records for a set of customers in the target group (customers who are big spenders or who commit fraud, for example). In applications that we have seen, the target characteristic is rare, so there are not many target customers in the priming data. Moreover, in some applications, such as fraud, the relevant (fraudulent) records for each target customer are not labelled, so the relevant and irrelevant records for the target behavior are not distinguished in advance. Given all the uncertainties surrounding the target data, we do not construct separate signatures for each customer with the target behavior but instead compute one *target signature* $\boldsymbol{t}$ from the data for all the customers with the target behavior. Thus, $p_{i,k}$ refers to the fraction of transactions for customer i for which variable X falls in bin k, and t_k refers to the fraction of target transaction records for which variable X falls in bin k. The target signature is assumed to have the same structure as any other signature. Also, any customer known to have the target behavior is removed from the priming data, so the priming data represents customers without that behavior (perhaps including a few misidentified individuals) and the target set includes customers with the target behavior (perhaps including some irrelevant records.)

3.1 Binning Signature Variables

Because a signature is a set of components, it is tempting to consider the resolution needed for each component X conditional on Y separately. But that would require knowledge of the signature structure (choice of conditioning variables), which might depend on the resolution of the variables. Thus, there is a chicken and egg dilemma. Should the variables be binned first, or should the signature structure be defined first? The approach that we have taken is to choose the binning of the signature variables first, ignoring the conditioning, and then to choose the conditioning variables for the coarsened variables. The argument is that it is simpler to use the same resolution for each conditioning variable. The process of the coarsening variables and choosing conditioning variables could be iterative, choosing the binning, then choosing the set of histograms, then re-evaluating the binning and so on, but we have not done that in our applications. Also, we suppose that the number of bins to be allocated to each variable is determined in advance, so the only task is to define the bins. Again, the process could be iterative, so that the number of bins could also be optimized, but we have never done that.

If X is numeric, then it is natural to think of the bins as intervals with endpoints $d_0 < d_1 < \ldots < d_K$, where d_0 is the smallest value of X that can be observed and d_K is the largest (possibly infinity). The task is to choose $d_1, \ldots, d_{K-1}$ so that the coarsened distribution is as close as possible to the original distribution for *each* customer i. Many other authors have considered optimal *global* binning (e.g., [9] and the review paper [2]) in the context of representing the entire database of calls, not *local* binning, in the context of choosing one set of bins for representing each individual customer well.

Suppose first that there is no target group. Then, the global binning problem is solved by the equi-depth histogram in which each of the K bins has about $1/K$ of the transactions, in the sense that that histogram maximizes the empirical information entropy $IE_i = -\sum_{k=1}^{K} p_{i,k} \log p_{i,k}$ in the coarsened distribution [2]. For local binning, we choose the K bins for X that maximize the *average information entropy*

$$\overline{IE} = \frac{1}{N} \sum_{i=1}^{N} IE_i, \tag{2}$$

where N is the number of customers in the priming data. The average information entropy assigns equal weights to all individuals in the training data, but it can be easily adapted to weight different customers differently if some classes of customers are more important to track than others.

If there is a target behavior, then the bins should be chosen to increase the chances of finding it. Standard statistical theory [10] implies that when $X = k$ is observed, the log-likelihood ratio $log(t_k/p_{i,k})$ is the best discriminator based on X for testing whether customer i belongs to the target group. Large positive values of the log-likelihood ratio imply that bin k is more likely under the target signature, while large negative values imply that bin k is more likely under the customer's signature. The more extreme the log-likelihood ratio, the easier it is to identify whether a customer belongs to the target group. Thus, the goal is to make the log-likelihood ratio as positive as possible when the customer belongs to the target group and as negative as possible otherwise. More precisely, using the priming data, we choose the d_k's to maximize

$$KL(\boldsymbol{t}) = \frac{1}{N}\sum_{i=1}^{N}\left(w\sum_{k=1}^{K} p_{i,k}\log\frac{p_{i,k}}{t_k} + (1-w)\sum_{k=1}^{K} t_k \log\frac{t_k}{p_{i,k}},\right) \tag{3}$$

for some weight w, $0 \leq w \leq 1$. When $w = 0$, the criterion considers only the ability to identify members of the target group. When $w = 1$, the criterion considers only the ability to avoid misclassifying a customer who does not belong to the target group. Intermediate values of w balance these two concerns. Note that this procedure is equivalent to maximizing the average, weighted, symmetrized Kullback-Leibler distance [11] from $\boldsymbol{p}_i$ to $\boldsymbol{t}$, averaging over all individuals.

The cutpoints $d_1, \ldots, d_{k-1}$ that maximize the average information entropy (2) or the Kullback-Leibler distance (3) are found by exhaustive search when the original variable X has only several possible values. If exhaustive search is not feasible, then the search is constrained by specifying minimum widths for the K final bins. For example, transaction duration might be measured to the nearest second, but all bins might be forced to be at least 2 minutes long and to have endpoints that are integer minutes.

3.2 Choosing Conditioning Variables

The question is can the product (1) be simplified by ignoring some of the conditioning variables in some of the terms? For example, perhaps a variable X_k is independent of all the other variables, so only its marginal distribution, not its conditional distributions, need to be considered. Or perhaps X_3 depends on X_1 but not X_2 so only the conditional distributions X_3 given X_1 need to be considered. Because each possible combination of the conditioning variables gives another term in the product, reducing the number of conditioning variables for one signature variable X for each customer can save significant space.

Generally, X should be conditioned on Y if the distribution of X varies significantly with Y, which happens if the conditional distributions of X given Y differ from the distribution of X that does not condition on Y. For example, consider the coarsened duration X of wireless calls and the candidate conditioning variables "service provider" Y_1, which has values "local" and "roaming", and "direction," Y_2 which has values "incoming" and "outgoing". The top panel in Figure 1 shows the marginal (unconditional) histogram of duration for all calls for one customer in a set of priming data, and the four bottom panels show the conditional histograms of duration for the same customer, split according to the four possible combinations of service provider and direction. For this customer, stratifying call duration by provider and direction would be useful. If the histograms in the four strata had been similar, though, then stratifying duration by either service provider or direction would waste space (and be statistical inefficient) because the same information on the customer would need to be stored (and estimated) four times. Similarly, if the histograms in the two rows in Figure 1 were similar but the two columns were not, then keeping separate histograms for the durations of incoming and outgoing calls would be useful for this customer, but also separating roaming and local calls would not.

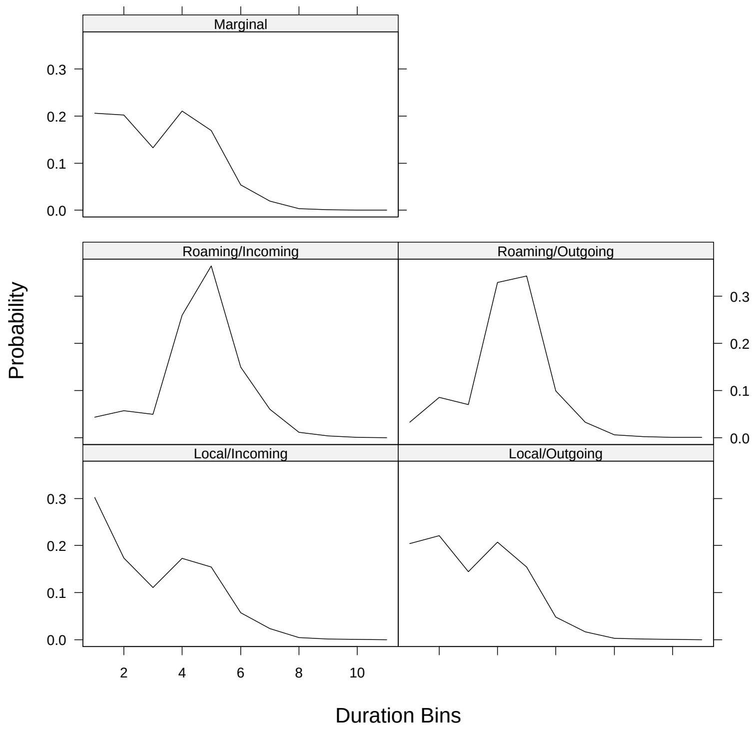

Figure 1: The marginal distribution (top panel) and conditional distributions (bottom four panels) of the the signature variable call duration for one customer.

Of course, finding conditioning variables that are appropriate for *all* customers simultaneously is likely to be impossible. Instead, we strive for conditioning variables that are *"good for most, best for some"*. That is, the selected conditioning variables for X must be "useful" for most customers and "important" for some customers, where useful means that the differences in the histograms corresponding to the different values of Y are statistically significant for the customer. Thus, we have a statistical model selection for each customer. There are many variants of model selection (see, for example, the textbook [12]); here we adopt forward model selection in which variables are added to the model sequentially until the gain from adding another conditioning variable is unimportant.

Forward model selection proceeds as follows. First, take a candidate conditioning variable Y in the set $\mathcal{C}$ of all conditioning variables. For each customer i in the priming data and each level y_l of Y, $l = 1, \ldots, L$, compute the number of transactions $a_{k,l}$ for which $X = x_k$ and $Y = y_l$. This gives a $K \times L$ table for customer i that looks like

$$\begin{array}{|cccc|c|} a_{1,1} & a_{1,2} & \cdots & a_{1,L} & r_1 \\ a_{2,1} & a_{2,2} & \cdots & a_{2,L} & r_2 \\ \vdots & & & & \vdots \\ a_{K,1} & a_{K,2} & \cdots & a_{K,L} & r_K \\ \hline c_1 & c_2 & \cdots & c_L & n \\ \hline \end{array} \tag{4}$$

where $c_l = \sum_{k=1}^{K} a_{k,l}$ is a column sum, $r_k = \sum_{l=1}^{L} a_{k,l}$ is a row sum, and $n = \sum_{k=1}^{K} r_k = \sum_{l=1}^{L} c_l$ is the total number of transactions in the priming data for customer i. A standard χ^2 statistic [12] can then be computed from the table to test whether the conditioning variable Y is important for signature variable X for customer i, as long as the counts in the table are not too small. The text book answer to "how small is too small" is five, but this is often too conservative [12]. Here we just assume that there is a threshold T for too small and apply the following algorithm to each customer's table separately.

1. Remove rows with $r_k = 0$ and columns with $c_l = 0$.
2. Compute the *expected table* with entry $E_{k,l} = (r_k \times c_l)/n$ for cell (k, l).
3. If the smallest expected cell count, say $E_{k',l'}$, is smaller than T, then compare $r_{k'}$ and $c_{l'}$. If $r_{k'}$ is less than $c_{l'}$, compare $r_{k'-1}$ and $r_{k'+1}$. If $r_{k'-1}$ is smaller, collapse row k' with row $k'-1$ in the table. If $r_{k'-1}$ is larger, collapse row k' with row $k'+1$. If $r_{k'-1} = r_{k'+1}$, collapse row k' with either row $k'-1$ or row $k'+1$. An analogous procedure, substituting columns for rows, is applied when $r_{k'} \geq c_{l'}$.
4. Re-compute the expected cell counts for the collapsed table from Step 3.
5. Repeat Steps 3 and 4 until there are no expected cell counts below the threshold T.
6. If the final collapsed table from Step 5 for customer i has only one row or one column, then discard the table and do not continue with the test for customer i.
7. Suppose the table from Step 5 has $K' > 1$ rows and $L' > 1$ columns. Then compute the test statistic
$$\chi^2 = \sum_{k=1}^{K'} \sum_{l=1}^{L'} \frac{(r_k c_l / n - a_{k,l})^2}{r_k c_l / n}.$$
8. The p-value for customer i is defined to be the probability that a chi-squared random variable with $(K' - 1) \times (L' - 1)$ degrees of freedom exceeds the computed value of the test statistic χ^2.

Following standard statistical practice, the smaller the p-value, the stronger the evidence that the signature of X should be conditioned on Y for customer i.

Because any conditioning variable Y for X is either used for everyone or used for no one, we require some evidence that Y is useful for a majority of customers and strong evidence that it is useful for some customers. For example, the p-value should be less than 0.05 for at least 50% of all customers and less than 0.01 for at least 10% of all customers. Exactly how small the p-value should be and exactly what fraction of customers should have a p-value that small to justify conditioning on Y depends on the application. The general form of this "good for most, best for some" criterion is that at least $100\delta_1\%$ of the p-values must be smaller than α_1 and at least $100\delta_2\%$ of the p-values must be smaller than α_2, with $\alpha_2 < \alpha_1$ to condition X on Y. We represent this criterion by $(\alpha_1, \delta_1, \alpha_2, \delta_2)$.

To illustrate the selection procedure, let us consider again the wireless calls database example. Suppose we want to find conditioning variables for call duration and the candidate set includes call direction (incoming or outgoing) and an indicator of peak/off-peak calls. Figure 2 presents the cumulative percentage of p-values for the chi-square tests of independence between call duration and the candidate conditioning variables, based on a priming dataset with about 20,000 customers.

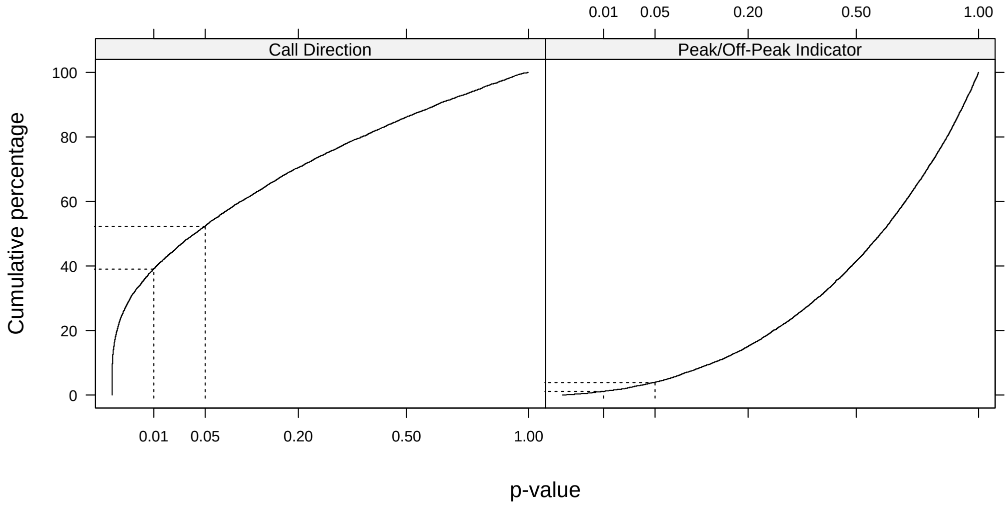

Figure 2: Cumulative percentage of p-values for the chi-square test of independence between call duration and call direction and call duration and peak/off-peak indicator. A square-root scale is used for the p-values.

The plots indicate that call direction has a very significant association (p-value ≤ 0.01) with call duration for about 40% of the customers and a significant association (p-value ≤ 0.05) for about 55% of the customers. The peak/off-peak indicator, however, has a very significant association with call duration only for 1.2% of the customers and a significant association only for 4% of the customers. Assuming a $(0.05, 0.50, 0.01, 0.10)$ "good for most, best for some" criterion, the conclusion is that call direction should be chosen as the single conditioning variable for call duration.

The above selection procedure is applied to each candidate conditioning variable for X in turn. If no candidate conditioning variable satisfies the $(\alpha_1, \delta_1, \alpha_2, \delta_2)$ criterion, then the marginal distribution of X is used as a signature component. If exactly one candidate variable Y satisfies the criterion, then X is conditioned Y and each possible value of Y gives a different signature component for X. If more than one candidate Y satisfies the criterion, then the one with the largest fraction of customers with p-values below α_1 is selected. At this point we have the first conditioning variable (if any), say Y_1.

The next step is to decide whether conditioning on Y_1 alone is sufficient or conditioning on an additional variable Y_2 would be helpful. To do that, each of the conditional histograms for X given one of the possible values of Y is further partitioned according to each of the remaining candidate variables in $\mathcal{C}$. For example, suppose the conditioning variable Y_1 that has already been selected has L_1 possible values and the candidate conditioning variable Y_2 under consideration has L_2 possible values. Then a table analogous to Table 4 is built with $L_1 \times L_2$ columns, one for each of the possible combinations of Y_1 and Y_2 and K rows, one for each of the possible values of X. The steps outlined above are then applied to the table, giving a χ^2 p-value. Again, the candidate variables, if any, that satisfy the $(\alpha_1, \delta_1, \alpha_2, \delta_2)$ criterion are found, and of those the one with the largest value of α_1 is chosen. The selection procedure is iterated until no further conditioning variable satisfies the selection criterion, or all candidate conditioning variables are chosen. Note that the choice of $(\alpha_1, \delta_1, \alpha_2, \delta_2)$ controls the amount of conditioning, and thus the number of histograms in the signature for representing X.

Note that the decomposition (1) constrains the set of candidate conditioning variables for any signature variable X. For example, the variable X_k can only be conditioned on variables that are derived from $X_1, \ldots, X_{k-1}$ or are unrelated to $X_1, \ldots, X_K$. Also, the procedure here can easily be modified to handle signatures that are not based on histograms by replacing the χ^2 test with another test that is more appropriate for the signature component.

3.3 Quantizing Histogram Cells

If each signature component is represented by a histogram, and each cell of the histogram contains a probability, then the amount of space allocated to each probability has a major effect on the amount of space needed to store the signature and a major effect on the ability to represent small probabilities accurately. Suppose, for example, that each cell probability is allocated one byte, so 256 different cell entries are possible. If the interval $[0, 1]$ is cut into 256 equal segments and the midpoint of each of the 256 intervals is used to represent the probability, then the smallest nonzero probability that can be represented is $1/512$. This may not be small enough for applications, such as fraud detection, in which the target behavior is rare.

The smallest probability that can be monitored with a byte can be nearly halved as follows. For a given signature component, find the cell with the largest probability (or one of the cells with the largest probability in case of a tie.) The probability for that cell can be found by subtracting the sum of the probabilities of the other cells from 1, so there is no need to store it. The probability of each of the other cells cannot exceed 0.5, so now it is enough to quantize the half interval $[0, 0.5)$ into 255 subintervals labelled 0 through 254, reserving the value 255 to mark the cell with the largest probability. Thus, the smallest quantized probability is now $1/1020$ instead of $1/512$. The quantized probability for the maximum probability cell is then 1 minus the sum of the quantized probabilities for the other cells.

Even $1/1020$ may be too big when the target behavior is rare. For those applications, the interval $[0, 0.5]$ must be broken into unequal pieces. For example, the log probabilities to some base, instead of the raw probabilities, can be quantized. A minimum quantized probability of p_* can then be achieved by taking a log-uniform quantization on $[0, 0.5)$ to a base $b_* = (2p_*)^{-1/254.5}$. The labels 0 through 254 are then assigned according to

$$q(p) = 255 + \begin{cases} 0, & 0.5 \leq p \leq 1 \\ \lfloor \log_{b_*} 2p \rfloor, & p_* < p < 0.5 \\ -255, & p \leq p_*. \end{cases} \quad \text{with} \quad b_* = (2p_*)^{-1/254.5},$$

where $q(p)$ is the quantized value for p and $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. Quantization is reversed by computing

$$p(q) = \begin{cases} b_*^{q-254.5}/2, & 0 \leq q < 255 \\ 1 - \sum_{k:q_k<255} p(q_k), & q = 255,\ n_{\max} = 1 \\ 0.5, & n_{\max} = 2. \end{cases},$$

where $n_{\max}$ is the number of cells with the largest probability. Note that the log-based quantization gives more precise small probabilities than uniform quantization does, but at the expense of less precise large probabilities. For example, if the logarithm base is chosen so that the minimum probability that can be recovered from the quantized values is $p_* = 10^{-6}$, then the maximum absolute difference between raw and quantized probabilities is 2.54×10^{-5} when $p \leq 10^{-3}$, but the maximum absolute difference is 0.0127 when $0.1 \leq p \leq 0.5$. If necessary, the error in p over several intervals can be controlled by breaking the interval $[0, 0.5]$ into subintervals and bounding the error in each subinterval.

4 Initializing a Signature for a New Customer

Our goal is to track customers dynamically, starting as soon as the customer begins to make transactions. To do that, we need to assign an *initial signature* to a customer who has made only one or a few transactions. This task resembles customer segmentation, in the sense that there is a set of initial signatures (or customer segments) and the customer is assigned to one of these sets based on the information available at the time of the assignment. In our approach, however, the customer is assumed to belong to multiple segments, one for *each* signature component. For example, the initial signature component for duration of incoming calls and the initial signature component for duration of outgoing calls are assigned independently in our model. This allows a huge number of different initial signatures – namely, the product of the number of initializations for each signature component. Having a rich set of initial signatures is important for expressing nuances in customer behavior. Again, the product representation (1), along with the elimination of some conditional terms in the product during signature design, enable this richness.

To illustrate the ideas, suppose that the signature component for X that is conditional on $Y = y$ is represented by a normalized histogram (cells summing to one) $\boldsymbol{p}_i = (p_{i,1}, \ldots, p_{i,K})$ for each customer i and that the initial values of $\boldsymbol{p}_i$ are "filled in" according to rules that depend on a set of *index variables* Z that are computed from customer i's first two transaction records. Thus, each value of Z points to a *representative signature component* $\boldsymbol{r}_Z = (r_{Z,1}, \ldots, r_{Z,K})$ for X conditional on Y. The goal is to find rules for assigning representative signature components so that $\boldsymbol{r}_Z$ describes a new customer accurately. More precisely, given a set of priming data, as described in Section 3, the goal is to find rules for which the initial assignment $\boldsymbol{r}_z$ is close to the summarized transaction history $\boldsymbol{p}_i$ for each customer i.

As in signature design, there is a list of *candidate index variables*, each of which takes on at most several values. (So candidate variables that have many possible values need to be coarsened, for example.) The candidate index variables do not have to be a subset of the signature and conditioning variables, and they may be binned differently from the signature and conditioning variables. For example, day-of-week may have seven values as a signature variable, three values (weekday, Saturday, Sunday) as a conditioning variable, and only the two values (weekday, weekend) as an index variable. The only requirement is that the index variables must be able to be evaluated when a representative signature is assigned.

Our procedure for choosing index variables for a signature component X conditional on Y is sequential, adding candidate index variables one at a time until the incremental benefit from adding any remaining candidate is insignificant. To start, suppose the histogram for X conditional on Y has K bins, the candidate index variable Z has J levels labelled $1, \ldots, J$, and no index variable has been chosen yet. Then index selection has the following steps.

1. For the i^{th} customer in the priming data with $Z = j$, find the histogram $\boldsymbol{p}_{i,j} = (p_{i,j,1}, \ldots, p_{i,j,K})$

of X conditional on $Y = y$. Suppose there are N_j such customers.

2. Find the average signature $\bar{\boldsymbol{p}}_j = (\bar{p}_{j,1}, \dots, \bar{\boldsymbol{p}}_{j,K})$ for the customers found in Step 1, where

$$\bar{p}_{j,k} = \frac{1}{N_j} \sum_{i=1}^{N_j} p_{i,j,k}.$$

3. Compute the average signature component $\bar{\boldsymbol{p}}_0$ that would be used ignoring Z; namely,

$$\bar{p}_{0,k} = \frac{1}{\sum_{j=1}^{J} N_j} \sum_{j=1}^{J} \sum_{i=1}^{N_j} p_{i,j,k}.$$

4. Compute the bin distances

$$b_{i,j,k} = \begin{cases} |\log\left(p_{i,j,k}/\bar{p}_{j,k}\right)| & p_{i,j,k} > 0 \\ -\log(\bar{p}_{j,k}) & otherwise. \end{cases}$$

$$b_{i,0,k} = \begin{cases} |\log\left(p_{i,j,k}/\bar{p}_{0,k}\right)| & p_{i,j,k} > 0 \\ -\log(\bar{p}_{0,k}) & otherwise. \end{cases}$$

Then define the effect of using index variable Z for the signature component X conditional on Y for a customer i with $Z = j$ as

$$D_{i,j} = \sum_{k=1}^{K} \left(b_{i,j,k} - b_{i,0,k}\right).$$

The effect of $Z = j$ is positive if $\bar{\boldsymbol{p}}_j$ is closer to the customer's signature than the average signature $\bar{\boldsymbol{p}}_0$, which does not depend on Z. The average effect of Z over all customers with $Z = j$ is then

$$\bar{D}_j = \frac{\sum_{i=1}^{N_j} D_{i,j}}{N_j}.$$

5. Because Z is known when a representative signature is needed, we can choose to use Z only when $Z = j$ for some j for which $\bar{D}_j > 0$; that is, only when Z is, averaging all customers with $Z = j$, useful. Therefore, define the candidate representative signature r_Z for X conditional on Y by

$$r_Z = \begin{cases} \bar{\boldsymbol{p}}_j & \text{if } Z_i = j \text{ and } \bar{D}_j > 0 \\ \bar{\boldsymbol{p}}_0 & \text{if } Z_i = j \text{ and } \bar{D}_j \leq 0. \end{cases}$$

6. Let $\mathcal{J}^+$ be the set of j for which $\bar{D}_j > 0$; that is, the set of customers with a value $Z = j$ for which the representative signature is not just the initial average signature $\boldsymbol{p}_0$. Let N_j^+ be the number of customers i for which $Z = j$, $\bar{D}_j > 0$, and $D_{i,j} > 0$; that is, the number of customers in the priming data that would have representative signature $\boldsymbol{p}_j$ and for which that signature would be better than $\boldsymbol{p}_0$. The *coverage* of Z is defined as the fraction of customers in the priming data for which indexing by Z is useful, so

$$C_Z = \frac{\sum_{j \in \mathcal{J}^+} N_J^+}{\sum_{j=1}^{J} N_j}.$$

Finally, the *average value* for a customer that is indexed by Z is

$$V_Z = \frac{\sum_{j \in \mathcal{J}^+} \sum_{i=1}^{N_j} D_{i,j}}{\sum_{j \in \mathcal{J}^+} N_j}.$$

Whether Z is useful for indexing depends on its coverage and average value. Ideally, we would like the indexing to be useful for all customers and to have a large effect on all customers, but often the more customers an index applies to the smaller its average effects. Thus, it is necessary to balance coverage and average value.

First, only consider indexing variables that exceed a minimum threshold (e.g., 30%) on coverage. If no candidate index variable exceeds the coverage threshold, then index selection stops, without adding an index variable. Otherwise, the relative importance of coverage and average value needs to be balanced. For example, what does the average value of an index variable with 40% coverage need to be to be equivalent to an index variable with 80% coverage and an average value of 0.2? More generally, suppose Z_1 has better coverage but worse average value than Z_2 so $c_1/c_2 \geq 1$ and $v_2/v_1 \geq 1$. More importantly, suppose that there are constants c_* and v_* so that Z_1 and Z_2 would be considered equivalent if $c_1/c_2 = c_* \geq 1$ and $v_2/v_1 = v_* \geq 1$. For example, Z_1 and Z_2 might be equivalent if Z_1 has twice the coverage of Z_2 but Z_2 has four times the average value of Z_1, so $c_* = 2$ and $v_* = 4$. Then we require a function that is increasing in C and V and that has the same value at $(C_1, v_* V_1)$ and $(c_* C_2, V_2)$. One such function is

$$g(C, V) = V \times C^{\log(v_*)/\log(c_*)}.$$

Thus, the first index variable, if any, that is chosen is the one that maximizes $g(C, V)$ among all candidate index variables that exceed the minimum requirement for coverage.

For a concrete example, consider the indexing of the signature component corresponding to duration of outgoing calls in the wireless calls database application. Assume that four candidate index variables *based on the first two calls* are available: the average call duration, the number of peak calls, the number of roaming calls, and the number of outgoing calls. A minimum coverage threshold of 30% is used. Figure 3 presents the coverage, average value, and $g(C, V)$ value, calculated using $c_* = 2$ and $v_* = 4$, for each candidate index variable.

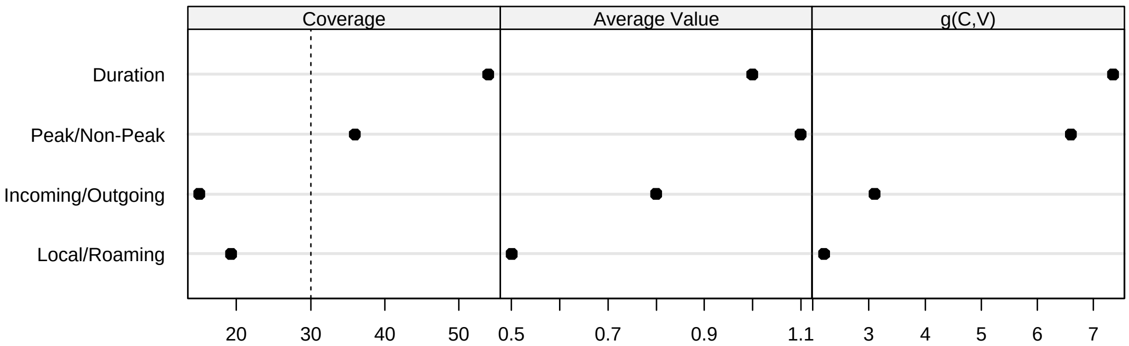

Figure 3: Coverage, average value, and $g(C, V)$ value – calculated using $c_* = 2$ and $v_* = 4$ – corresponding to the duration of outgoing calls signature component indexed by average call duration, number of peak calls, number roaming calls, and number of outgoing calls of the first two calls.

Only two of the candidate index variables, average call duration and number of peak calls, satisfy the minimum coverage threshold of 30%. Average duration has larger coverage and smaller average value than number of peak calls, and is the index variable selected by the $g(C, V)$ criterion.

Index variables are added sequentially, by making the following changes in the above procedure: replace the average signature component $\bar{\boldsymbol{p}}_0$ with the current r_z and take $\bar{\boldsymbol{p}}_j$, to be the average signature component over all customers for which $\boldsymbol{Z} = \boldsymbol{z}_j$ for the j^{th} possible combination of the current and candidate index variables.

The final representative signature corresponding to index variables $\boldsymbol{Z}$ for the signature component for X conditional on Y is then computed as follows.

1. Find all customers in the priming set for which $Y = y$ and $\boldsymbol{Z} = \boldsymbol{z}$.

2. Find $\boldsymbol{p}_i$, the signature component of X conditional on $Y = y$, for each customer i identified in Step 1.

3. Set $\boldsymbol{r}_{\boldsymbol{Z}} = (r_{\boldsymbol{Z},1}, \ldots, r_{\boldsymbol{Z},K})$ equal to the average of the components found in Step 2. If there are $n_{\boldsymbol{Z}}$ such customers, then

$$r_{\boldsymbol{Z},k} = \frac{1}{n_{\boldsymbol{Z}}} \sum_{i=1}^{n_{\boldsymbol{Z}}} p_{i,k}.$$

4. The representative signature component is then

$$\boldsymbol{r}_Z = \begin{cases} \bar{\boldsymbol{p}}_j & \text{if } \boldsymbol{Z}_i = j \text{ and } \bar{D}_j > 0 \\ \bar{\boldsymbol{p}}_0 & \text{if } \boldsymbol{Z}_i = j \text{ and } \bar{D}_j \leq 0, \end{cases}$$

where the subscript j now denotes the j^{th} possible value of the index variables $\boldsymbol{Z}$.

5 Updating Signature Components

A key feature of a signature is that it can be updated sequentially from the current transaction record – no past record needs to be accessed – by methods that are no more difficult to compute than weighted averaging. This ability to avoid looking at past transactions is important for both processing and analysis. For example, reading and writing a transaction to a data warehouse and retrieving it for analysis or summarization later is often much less efficient than updating a summary or analysis when the record is written. If a signature is sufficiently short to be stored in main memory, then nearly real-time applications, such as fraud detection, are possible using signatures. Moreover, allowing a signature to evolve in time (measured in transactions) rather than abruptly changing at the end of an arbitrary period, enables up-to-the minute data mining because the data summaries are always current.

Histograms for record variables, like call duration or direction, that can be considered to be randomly varying according to P_n, the probability distribution that is used to predict behavior at transaction $n + 1$, are particularly easy to update. If $\widehat{\boldsymbol{p}}_n$ are the histogram probabilities after transaction n, and $\boldsymbol{X}_{n+1}$ is a vector of 0's except for a 1 in the histogram cell that contains the observed value, then the sequentially updated histogram probabilities are

$$\widehat{\boldsymbol{p}}_{n+1} = (1 - w_{n+1})\widehat{\boldsymbol{p}}_n + w_{n+1}\boldsymbol{X}_{n+1}, \tag{5}$$

where $w_{n+1} = 1/(n+1)$. The initial estimate of the signature component, $\widehat{\boldsymbol{p}}_0$, can either be set to a vector of zeroes, or initialized with a representative signature, using the methodology described in

Section 4. Updating thus requires only the most recent histogram, the number of transactions made so far and the current transaction. We call this kind of updating *unweighted averaging*, because $\widehat{\boldsymbol{p}}_n$ weights each observed transaction $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ equally.

Unweighted averaging is appropriate for static databases, or dynamic databases in which a customer's behavior does not change over time. If behavior changes over time, then unweighted averaging is inappropriate because recent transactions have no more influence on the histogram than old transactions do. Evolving behavior is tracked better by *exponentially weighted moving averaging (EWMA)*. The updated EWMA vector $\widehat{\boldsymbol{p}}_{n+1}$ is given by equation (5) with $w_{n+1} = w$ for a fixed weight w, $0 < w < 1$, that controls the extent to which $\widehat{\boldsymbol{p}}_{n+1}$ is affected by a new transaction and the speed with which a previous transaction is "aged out." For example, if $w = 0.05$, then the probability assigned to the observed histogram bin increases by a constant amount w and the probability of any other bin decreases by a factor of 0.95. Also, transaction $n - 10$, which was 10 transactions earlier than transaction n, has about 60% the weight of transaction n if $w = 0.05$, but about 82% the weight of transaction n if $w = 0.02$. Thus, smaller w lead to more stable estimated distributions. The initial probability estimate $\widehat{\boldsymbol{p}}_0$ must be specified, as in the unweighted averaging procedure. (See [13] for more information about EWMA.) Under some conditions, the EWMA approximates the posterior mean under a Bayesian dynamic model [14].

Timing variables, like day of week and hour of day, are not randomly observed because they are observed "in order." For example, all the transactions for Tuesday of this week are observed before all the transactions 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 (see [15]). Signature components that are represented as top categories plus "others", rather than as complete distributions, are only slightly more difficult to update sequentially. For example, a move-to-the-front scheme can be used to update the labels of the top categories and a variant of exponentially weighted moving averaging can be used to update the probabilities $\widehat{p}_n$. (See [16] for the details of one possible approach.) 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 (see [17] for details).

Although updating is conceptually simple, there can be challenges when probabilities are quantized. For example, suppose the quantized probability 0.01 represents the interval from 0.005 to 0.015 and the updating weight is 0.005. Then an observation in that bin causes the bin probability 0.01 to be updated to 0.015, but the quantized value of 0.015 is again 0.01. Thus, the bin probability in this example can never be updated by observations that fall in the bin unless the updating scheme is modified. Obviously, similar problems can arise for bins with large probabilities. Note that this problem cannot necessarily be avoided by choosing weights carefully because bin probabilities can vary dramatically over a large customer base.

Probabilistic updating solves the problem of stagnant bins due to a mismatch in updating weights and cell probabilities. Simply stated, we toss a coin with a certain probability of heads every time that an observation falls in a stagnant cell and change the cell probability to an adjacent quantized value if the coin turns up heads. Thus, the cell probability in the above example would be moved to the quantized value below 0.01 if the coin showed heads, even though the quantized probability by the standard updating rule should be 0.01 again. If the coin has the right probability of heads, then *on average* the quantized probabilities and the probabilities that would have been computed without any quantizing both take about the same number of updates to move to a neighboring cell.

Let q_n be the quantized level (e.g., a number between 0 and 254 or 255) after transaction n. Suppose $p_n = p(q_n)$ is the probability that is quantized to q_n, and let $u(q_n)$ be the minimum number of updates needed to move the raw probability p_n to a value that either has quantization level $q_n - 1$ or $q_n + 1$. A little work shows that the number of updates needed to move p_n that much (assuming no quantization) is

$$u(q_n) = \frac{\log\left[p(q_n - 1)/p(q_n)\right]}{\log(1-w)}, \; q_n \geq 1$$

when q_n has to be moved to $q_n - 1$, and

$$u(q_n) = \frac{\log\left\{\left[1 - p(q_n + 1)\right] / \left[1 - p(q_n)\right]\right\}}{\log(1-w)},$$

when q_n has to move to $q_n + 1$. Probabilistic updating then proceeds as follows. At each transaction, the updated quantized level is moved to the appropriate neighboring bin with probability $1/u(q_n)$ and stays at the current value with probability $1 - 1/u(q_n)$. The number of transactions until a quantized value is moved has a geometric$(1/u(q_n))$ distribution, so on average $u(q_n)$ transactions are needed to move the bin probability, as desired.

6 Conclusions and Further Thoughts

This paper introduces a fixed-length data structure called a *signature* that represents massive datasets of records on customers efficiently. Signatures do not attempt to give an exact accounting of the past records, but rather to describe current customer behavior accurately. Thus, they are appropriate for targeted analyses, such as detecting fraud or big spenders, especially when the goal is to influence customer behavior in nearly real-time.

Designing a signature requires pre-processing some priming data to decide how much detail is needed on each customer, in light of the need to save space but still represent customer behavior accurately. There are three kinds of ways that we save space. First, signature variables are coarsened or binned. Second, for each signature variable, some of the possible conditioning variables are eliminated, keeping only those that are statistically significant for a majority of customers and very important for some customers. These first two tasks are accomplished by applying methods discussed in Sections 3.1 and 3.2 to the records for a representative set of customers. Third, each histogram probability is represented by one byte, taking into account that small probabilities often need special care because they are associated with the most interesting behaviors.

This paper also gives a statistically sound procedure for initializing a signature for a new customer, after the customer makes at most a few transactions. The initialization rules are derived from a set of priming data. The rules are such that they can be applied in real-time, so initializing a signature is event-driven rather than time-driven. Initialization is also flexible, in the sense that each signature component is initialized separately. Thus, the set of initial behaviors is rich.

Signatures can be updated sequentially, using fast procedures that require only the current signature and the current transaction. Sequential updating based on exponentially weighted moving averaging is particularly important with dynamic databases, in which individuals may change their behavior over time. The adaptive updating allows the signature to track evolving behavior.

Signatures generally take little space to store, so a signature database takes only a fraction of the space of the raw data. For example, in one application we had 33 gigabytes of raw wireless calling records for about 1.5 million customers. The signature for each customer used about 200 bytes (including the space required to store the wireless phone number), so the signature database required only about 300 megabytes to store. In an application to wireline calling, there were 90 gigabytes of transaction records for about 1 million customers. Each signature required about 80 bytes to store, so the signature database required only about 80 megabytes.

Because signatures take so much less space than the original records, it is even possible to maintain them in main memory in some applications. This can enable fast, approximate answers to queries at the database and customer level. Signatures can also be used for more sophisticated analyses, including classification (fraud detection, credit scoring) and clustering (customer segmentation). Furthermore, many statistical algorithms, including most types of regression analysis, can be easily adapted to work with signatures, opening the possibility for more in-depth analyses of big databases. Again, reducing the raw data to signatures saves space, making some of the procedures that would normally too costly to run on the raw data possible to apply to the signature data.

Signatures can also be thought of as a way to do real-time, individualized customer segmentation. A new customer is assigned to a segment (given an initial signature) that is determined from past data using a procedure that is applied off-line, as segmentation methods typically are. But as the customer makes transactions, this initial segment evolves until eventually the customer has a personalized segment. The process for this is painless – no additional off-line processing is required to change the customer's segment. In particular, the database does not have to be re-clustered in order to change a customer's segment. Moreover, even the initial segments are richer than those that would normally be provided by clustering or fitting trees, because each signature component is initialized separately. Thus, the number of possible segments is the product of the number of possible initializations for each segment.

Further research is needed to more formally determine the information loss associated with data reduction using signatures. This information loss may be evaluated in the context of a specific type of analysis (e.g., results of a regression analysis using the whole data and the individual signatures), or, more generally, by estimating the differences between the true joint distribution and the signature representation. Methods for efficiently and meaningfully displaying massive datasets using individual signatures also require further investigation.

References

[1] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining.* AAAI Press, Menlo Park, CA, 1996.

[2] D. Barbará, W. DuMouchel, C. Faloutsos, P. J. Haas, J. M. Hellerstein, Y. Ioannidis, H. V. Jagadish, T. Johnson, R. Ng, V. Poosala, K. A. Ross, and K. C. Sevcik. The New Jersey data reduction report. *Bulletin of the Technical Committee on Data Engineering*, 20(4):3–42, 1997.

[3] L. Breiman and J. Friedman. *Statistical Signal Processing*, chapter Tools for Large Data Set Analysis, pages 191–197. Marcel Dekker, New York, 1984.

[4] P. B. Gibbons and Y. Matias. Synopsis data structures for massive data sets. Technical Memorandum BL0112300-980928-01TM, Bell Labs, Lucent Technologies, September 1998.

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

[6] W. DuMouchel, C. Volinsky, T. Johnson, C. Cortes, and D. Pregibon. Squashing flat files flatter. In *Proceedings of the Fifth ACM Conference on Knowledge Discovery and Data Mining*, pages 6–15, 1999.

[7] J. L. Devore. *Probability and Statistics for Engineering and the Sciences.* Wadsworth, Belmont, CA, 4th edition, 1995.

[8] M.H. Hansen, Z. H. Huang, C. L. Kooperberg, C. J. Stone, and Y. Truong. *Data Mining with Splines.* Springer-Verlag, New York, 2000.

[9] I. Grabec. Modelling of chaos by a self-organizing neural network. In K. Makisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks, Proceedings of ICANN*, volume 1, pages 151–156. Elsevier Science Publishers, 1991.

[10] P. J. Bickel and K. A. Doksum. *Mathematical statistics.* Holden-Day, San Francisco, CA, 1976.

[11] S. Kullback and A. Leibler. On information and sufficiency. *Annals of Matehmatical Statistics*, 22:79–86, 1951.

[12] A. Agresti. *Categorical data analysis.* John Wiley & Sons, New York, NY, 1990.

[13] B. Abraham and J. Ledolter. *Statistical Methods for Forecasting.* John Wiley & Sons, New York, NY, 1983.

[14] M. West and J. Harrison. *Bayesian forecasting and dynamic models.* Springer-Verlag, 1989.

[15] D. Lambert, J. C. Pinheiro, and D. X. Sun. Updating timing profiles for millions of customers in real-time. Technical Memorandum Statistics and Data Mining Research, Bell Labs, Lucent Technologies, 1999.

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

[17] F. Chen, D. Lambert, and J. C. Pinheiro. Sequential percentile estimation. Technical Memorandum Statistics and Data Mining Research, Bell Labs, Lucent Technologies, 2000.