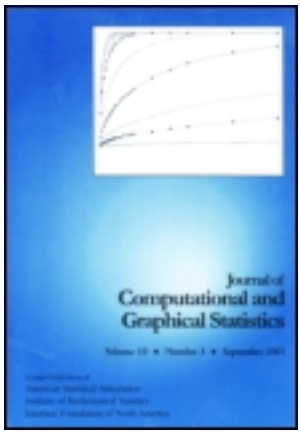

Journal of Computational and Graphical Statistics
Publication details, including instructions for authors and subscription information:
http://amstat.tandfonline.com/loi/ucgs20

Building an Effective Representation for Dynamic Networks

Shawndra Hill, Deepak K. Agarwal, Robert Bell and Chris Volinsky
Shawndra Hill has accepted a position as Assistant Professor, Operations and Information Management Department, Wharton School of the University of Pennsylvania, 3730 Walnut Street, Suite 500, Philadelphia, PA 19104. Deepak K. Agarwal is Senior Technical Specialist, Robert Bell is Senior Technical Specialist, and Chris Volinsky is Director, Statistics Research Department, AT&T Labs-Research, 180 Park Avenue, Florham Park, NJ 07932.
Version of record first published: 01 Jan 2012.

Building an Effective Representation for Dynamic Networks

Shawndra HILL, Deepak K. AGARWAL, Robert BELL, Chris VOLINSKY

A dynamic network is a special type of network composed of connected transactors which have repeated evolving interaction. Data on large dynamic networks such as telecommunications networks and the Internet are pervasive. However, representing dynamic networks in a manner that is conducive to efficient large-scale analysis is a challenge. In this article, we represent dynamic graphs using a data structure introduced in an earlier article. We advocate their representation because it accounts for the evolution of relationships between transactors through time, mitigates noise at the local transactor level, and allows for the removal of stale relationships. Our work improves on their heuristic arguments by formalizing the representation with three tunable parameters. In doing this, we develop a generic framework for evaluating and tuning any dynamic graph. We show that the storage saving approximations involved in the representation do not affect predictive performance, and typically improve it. We motivate our approach using a fraud detection example from the telecommunications industry, and demonstrate that we can outperform published results on the fraud detection task. In addition, we present a preliminary analysis on Web logs and e-mail networks.

Key Words: Approximate subgraphs; Dynamic graphs; Exponential averaging; Fraud detection; Link prediction; Statistical relational learning; Transactional data streams.

1. INTRODUCTION

A graph is one way of representing complex dynamic network phenomena we encounter today. In a *dynamic graph*, nodes represent the transactors, and edges represent (directed) transactions between the transactors. These graphs are built from lists of transactions with time stamps and may include other important information such as the duration of transactions or the physical location of the transactors. What differentiates dynamic graphs from other graphs is that the collection of nodes and edges are subject to discrete changes, such as additions or deletions over time (Eppstein, Galil, and Italiano 1999).

Shawndra Hill has accepted a position as Assistant Professor, Operations and Information Management Department, Wharton School of the University of Pennsylvania, 3730 Walnut Street, Suite 500, Philadelphia, PA 19104. Deepak K. Agarwal is Senior Technical Specialist, Robert Bell is Senior Technical Specialist, and Chris Volinsky is Director, Statistics Research Department, AT&T Labs–Research, 180 Park Avenue, Florham Park, NJ 07932.

Journal of Computational and Graphical Statistics, Volume 15, Number 3, Pages 584–608
DOI: 10.1198/106186006X139162

The notion of a dynamic network appears naturally in a wide range of domains. Perhaps the most obvious examples of dynamic networks are communications networks such as a telephony network or the Internet. In a telephony network, data exist in the form of call detail records, which contain information on phone calls between two network transactors or IDs. The data may include the origination and termination telephone numbers, the date and time of the call, the duration of the call, and any charges for the call. Other examples of data that can be represented by dynamic graphs are author citation networks, social networks, online auctions, and disease transmission data. Although data on dynamic networks are readily available, representing the dynamics in a way that is meaningful for analysis poses a challenge.

There has been a vast amount of recent research on real-world networks and graphs as representations of those networks (Newman 2003). Much theoretical work has been done on the properties of different graph structures including random graphs (Erdős and Renyi 1960) and interval graphs (Scheinerman 1990). However, it has been established that real world networks are not random, and instead follow so-called power laws (Barabasi and Albert 1999). Historically, studies have focused primarily on the global aspects of these networks, such as scale-free or small world properties (Watts 2002; Barabasi and Albert 1999). Global properties allow us to learn about overall characteristics of a graph, such as link structure, average path length, graph diameter, and degrees of separation. Global graph topology has informed the design of robust infrastructures (Albert, Jeong, and Barabasi 2000) and has implications for the rate of infectious disease spread (Anderson and May 1991) as well as the adoption rate of fads (Watts 2002). In addition, global models of network growth such as triad completion (Jin, Girvan, and Newman 2001), polarization (Macy, Kitts, Flache, and Benard 2003), balkanization (Van Alstyne and Brynjolfsson 2005), and cumulative advantage (Solla 1976) have been used to explain empirical data on real world network topologies. In general, global network models rely on centralized evaluation methods that require the entire network to generate useful information.

In many applications global network analysis is not feasible. For example, in the telecommunications industry many business problems rely on complex real-time analysis of large scale call detail record databases. Firms often cannot process the entire call network graph at one time due to its scale. However, global analysis is not always necessary. Often, in analysis, the behavior of individual nodes is the primary interest.

An example using individual node analysis in telecommunications is for determination of *repetitive fraud*, where an individual has perpetrated some type of fraud, perhaps payment related, and has been disconnected. Sometimes the individual will attempt to set up another account, with no intention of valid payment. This individual may use a fake identification, so that we cannot use standard record-linkage techniques to link the old fraudulent account with the new account. However, if the new fraudulent ID has communication patterns similar to the old one (e.g., it may communicate with many of the same people), we may be able to use similarities in the calling patterns to link the old and new IDs. The challenge is to compare the communication patterns of all new IDs on the network to the ones that are known to be fraudulent quickly and efficiently. This suggests a local analysis, focusing on modeling each individual transactor. Nonetheless, comparing local relational features of transactors, against

millions of other network transactors in a short amount of time is nontrivial. To facilitate this, we employ an efficient representation of the dynamic transactional network. Although repetitive fraud is the primary application presented in this article, our representation is applicable to many domains with transactional data.

Two main challenges exist for dynamic network representation. The first challenge is to capture the most relevant features of the dynamic network while eliminating spurious information that does not improve understanding. The second challenge is to represent dynamic graphs efficiently so that the massive volumes of data in these domains can be processed in a reasonable timeframe.

Recently, dynamic network representations have been proposed to address these issues by using node labeling schemes. These schemes allow one to infer the distance between or adjacency of two nodes from their labels (Gavoille and Paul 2003; Breuer 1966). We know of two dynamic local distance labeling schemes that allow for distributed incremental updates, weighted dynamic trees (Korman and Peleg 2003) and communities of interest (COI) graphs (Cortes et al. 2003). Weighted dynamic trees assume a fixed tree graph structure. Therefore, global topological reassignment is needed when nodes and edges are added or deleted, which is costly for dynamic graphs with sufficiently high node and edge growth or decay rates. In contrast, Cortes, Pregibon, and Volinsky (2003)—furthermore referred to as CPV03—used a more general parameterized vector-based approximation that handles distributed incremental revisions to the labels. However, the parameters in CPV03 were selected in an ad-hoc fashion using heuristic arguments. In this article, we argue for an approach which estimates the parameters once through an offline analysis by employing only a sample of the entire dynamic network. The COI are stored in a database to facilitate efficient queries on transaction behavior of individual nodes.

We address the aforementioned challenges with an extension of the COI method. This method compactly represents nodes and their corresponding transactions by summarizing the dynamic nature of transactions between related nodes by the frequency and recency of interaction. We demonstrate our technique on several real-world datasets and show that it performs better than a representation that does not take dynamics into account. This paper makes the following contributions:

- *Approximation technique.* We formalize the COI method, represent it as an approximation parameterized by three key parameters, each with a clear interpretation, and provide an algorithm to set these parameters in a given application.

- *Evaluation technique.* We propose an evaluation technique for parameter selection based on predictive performance on future unobserved data. The evaluation is done on a classification problem where our scores produce better ROC curves compared to those in CPV03.

- *Application of technique.* We apply our technique to different domains to show its generality for a wide range of transactional data applications. We demonstrate that predictions on the approximated graph outperform predictions based on nonapproximated data on several real-world datasets.

The remainder of the article is organized as follows. Section 2 defines the representation of a dynamic graph using the COI representation proposed in CPV03 and introduces the three parameters that define the representation. Section 3 discusses our method of estimating the parameters using telecommunications data with the repetitive fraud application as our illustrative example. Section 4 discusses some other applications followed by a discussion in Section 5.

2. DYNAMIC GRAPH APPROXIMATION

In this article, we propose a framework for representing large dynamic networks for the class of problems where the level of analysis is the transactor. We focus on approximating individual transactors on the network using *entities*. An entity is comprised of both a specific node label (some unique identifier, also referred to as a seed node) and its corresponding local network. For example, the entity for a Web user would be the user herself and all of the Web pages she has visited. An entity's behavior is defined by the seed node's transactions with other nodes on the network over time, and is a subgraph of the entire network.

We know three things about the frequency of transactions between nodes: (1) the frequency of interaction between nodes evolves through time; (2) relationships between nodes may become stale, with the frequency of transactions going to zero; and (3) the occurrence of transactions is sometimes bursty, leading to irrelevant or noisy relationships that exist for short periods of time. When representing the evolution of transactions, we consider three characteristics of change: (1) the lifetime of relationships; (2) the frequency of transactions between related nodes; and (3) the degradation in the relative importance of relationships with time. The frequency of data observation and rate of change of entities is domain dependent, as is the amount and type of information available.

CPV03 captured entity change in a concise representation that changes smoothly through time. The authors used the representation to catch repetitive fraud on telecommunications networks, approximating an entity's behavior over time by a parameterized dynamic graph called the community of interest (COI). However, parameters were chosen in a heuristic, ad hoc fashion, and there was little discussion about model selection, parameter estimation, properties of the approximation, and most importantly, any potential loss associated with the approximation. The COI representation is the only work we know of that directly models the evolution of entity behavior to analyze dynamic graphs. This article extends and generalizes that work by providing a generic framework for modeling any dynamic network where the main focus of analysis is at the transactor level and the central goal is to build an approximate representation which optimizes prediction accuracy. We note that representations optimal for other purposes (e.g., visualization) could be the focus of some applications, but are beyond the scope of this article.

2.1 APPROXIMATION OBJECTIVES

Our approximation of the dynamic network using the COI representation should summarize the essential historical behavior of the entities while being efficient in storage space

and processing time. In addition, we want a parametric representation that is flexible enough to be applied to different domains. Prediction accuracy is our evaluation metric, a measure by which we can evaluate and compare parameter sets for the representation in the context of an application and across applications. Therefore, we have the following objectives when building our representation:

- *Summarization.* Represent historical behavior between two nodes in a concise manner, summarizing the relationship into a single edge with attributes.
- *Simplification.* Prune out noise (both edges and nodes) associated with spurious transactions (such as wrong numbers) or stale relationships.
- *Efficiency.* Handle massive data in a way that supports fast analysis and updating.

These objectives mesh together very well. Summarization supports efficiency; by summarizing multiple transactions into a single edge in a graph, we avoid repeated queries of a massive database. Simplification supports efficiency; by reducing the overall size of the graph, we can update and analyze faster. Simplification may also improve summaries because noisy transactions are eliminated. We represent the edges by an exponentially weighted moving average (parameterized by θ) to summarize the activity on an edge. We prune out noise locally by defining a maximum in- or out-degree (k) for each node, with any overflow going into an aggregator node called *other*. We also prune out edges whose weights are below a threshold (ϵ), since we have high confidence that these are stale edges. We tune the combined parameter set $\phi = (\theta, k, \epsilon)$ to maximize accuracy in predicting future behavior. Specifically, we maximize a similarity score (which may be application dependent) between representations based on data in some pre- and post-periods.

In the next subsections, we will provide a detailed description of the role played by the three parameters. Later we will provide guidance in setting the parameters for specific applications.

2.2 SUMMARIZING HISTORICAL BEHAVIOR

To define a dynamic graph, we will borrow from and extend the notation of CPV03. A graph G is a system of nodes and edges (denoted by $N(G)$ and $E(G)$), where the edges are connections between the nodes. A dynamic graph adds to this a set of time-stamped transactions among members of $N(G)$. In our formulation, each edge $e \in E(G)$ is an aggregation of the transactions between the two nodes for some specified time period. The type of aggregation defines a *weight* on each edge, $\mathrm{w}_G(e)$, such as total number of transactions, or some other relevant metric (e.g., sum of durations of phone calls).

We construct the exponentially weighted graph as follows. For any two graphs A and B, we first define the weighted sum $G = \alpha A \oplus \beta B$, for positive scalars α and β. It satisfies $N(G) = N(A) \cup N(B)$, $E(G) = E(A) \cup E(B)$, and $\mathrm{w}_G(e) = \alpha \mathrm{w}_A(e) + \beta \mathrm{w}_B(e)$, for all $e \in E(G)$. In words, the weighted sum of two graphs contains the union of the nodes and the union of the edges, with the weight on the edges in the combined graph determined

by the weighted sum from the component graphs.

Let the graph corresponding to the transactions during finite time period t be g_t. We define the graph G_t as a weighted collection of all of the transactions in the time periods up to and including g_t:

$$G_t = \omega_1 g_1 \oplus \omega_2 g_2 \oplus \ldots \oplus \omega_t g_t = \bigoplus_{i=1}^{t} \omega_i g_i. \tag{2.1}$$

This definition of G_t includes all historic transactions from the beginning of time, and allows for completely new nodes and edges to enter the graph. This framework includes the special cases where the edges represent the sum weight of all transactions ($\omega_i = 1$), the average weight per time period ($\omega = 1/t$), or a moving window ($\omega_1 \ldots \omega_{t-n} = 0$; $\omega_{t-n+1} \ldots \omega_t = 1/n$).

Often it is desirable to blend network activity in a way that discounts the past in favor of recent behavior. One such form, known as exponential smoothing (Winters 1960), uses weights: $\omega_i = \theta^{t-i}(1-\theta)$, where $0 \leq \theta \leq 1$. This form of weight function is convenient in the sense that Equation (2.1) can be expressed in recurrence form

$$G_t = \theta G_{t-1} \oplus (1-\theta) g_t. \tag{2.2}$$

This weight function provides a smooth dynamic evolution of G_t. In addition, periodic updates do not require accessing transaction data or graphs for all previous time periods. All that is needed is the graph through time period $t-1$ and the new set of transactions defined by g_t.

In the following we adopt Equation (2.2) as the definition of a dynamic graph at time t. The parameter θ determines the relative priority given to recent data. As θ nears 1, G_t approaches a simple average of all time periods, while θ near 0 puts almost all weight on the most recent time period. Therefore, as θ approaches 1, we are blending in more historical data. Another way of stating this is that θ determines the decay of the weight of a given transaction in the aggregated edge. Figure 1 shows the weight over time given to a one hour transaction for different values of θ. At time $t = 1$ day, the weight on the edge (in seconds) is $3{,}600(1-\theta)$, and it decreases multiplicatively by θ every day thereafter. With $\theta = 0.75$, this edge weight has decayed below a threshold of 0.1 in a few weeks, with $\theta = 0.95$ it takes several months to get to the same value, and with $\theta = 0.99$ (not shown in Figure 1), it takes almost two years. As we will see later, the best decay rate will depend on the application and can be set by using an appropriate θ. CPV03 set $\theta = 0.90$ based on heuristic arguments, and later in the article we will revisit the consequences of this choice.

2.3 SIMPLIFICATION BY GLOBAL AND LOCAL THRESHOLDING

The smoothed graph described above decays all edges exponentially over time, but never deletes them. To avoid maintaining a lot of edges with arbitrarily small weights, we define a global threshold parameter ϵ that prunes all edges with weights less than ϵ. This noise reduction step can significantly reduce the size of the graph over time. An edge weight

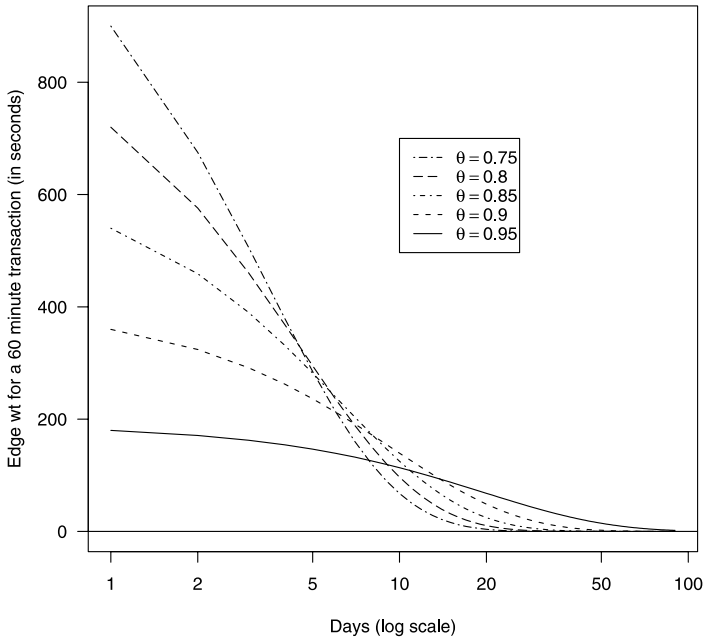

Figure 1. Contribution of a 60-minute (3,600 second) call to edge weight as a function of days. For a small θ, the contribution effectively reaches zero in a few weeks, whereas for a larger θ, it may take 100 days or more.

that is small could mean that the edge had little activity to begin with, or activity has not been observed on that edge for a long time. Both these scenarios are potential indicators of a transient transaction, meaning that it was a connection made once—a wrong number, a call to a store, or a click on a Web site that will never be returned to. Or perhaps it is a stale relationship, where there was once activity but we do not expect any more. If all edges associated with a node are pruned, the node is effectively pruned. We expect to gain efficiency by pruning these noisy edges. However, there is a subtle trade-off here; we want to delete edges if they are transient or stale, because they do not represent an important relationship for an entity (and yet consume storage space) but we do not want to delete information that may be relevant to an entity and can be useful in analysis. The parameter ϵ allows us to tune this pruning.

For most dynamic graph applications we have studied, the graph is extremely sparse. Any single node is typically connected to only a tiny fraction of other nodes in the graph. A growing literature shows that many networks follow the power law, which states that the vast majority of nodes have small in- and out- degrees, while a tiny fraction of nodes have very high in- and out-degrees (Barabasi and Albert 1999). For most nodes with very high degree even after the ϵ thresholding, we question whether it helps to keep track of all its edges. We have found that an entity may have hundreds of edges connected to the seed node, but only a small fraction of these account for a very large fraction of the total weight. Assuming that the edges with relatively small weights might be stale or transient, we assert that aggregating such edges would increase efficiency and create a more relevant summary

of the node's behavior.

To account for this local noise we employ a thresholding parameter, k, which is a maximum in- and out-degree for any single entity. For each entity, we retain the k edges with the largest weight at each update step in Equation (2.2). We also add an aggregator edge, called `other`, which collects the weight associated with edges not in the top k. This aggregator edge effectively replaces a subset of edges of the subgraph in order to preserve the total weight of the edge subset. Aggregating edges that account for a small fraction of the total weight can have a significant impact on computational efficiency.

2.4 EVALUATION CRITERIA: PREDICTIVE PERFORMANCE

We have defined an approximation of a dynamic graph in terms of the parameter $\phi = (\theta, k, \epsilon)$. In order to evaluate a set of parameter values, we compare the results from that set to a default case, $\phi = (\theta = 1, k = \infty, \epsilon = 0)$. Recall that $\theta = 1$ represents the no-smoothing case, where the edge weights are simple averages of transaction weights between pair of nodes, and $k = \infty, \epsilon = 0$ means there is no pruning of edges. Having defined and interpreted our parameters, we explain how to estimate them in the next few paragraphs. In addition, we also evaluate the performance of the estimated parameters relative to the default case.

The COI implementation in CPV03 used this framework with $\phi = (.90, 9, 0.1)$, each parameter being set via heuristic arguments. We present a more principled approach to set the COI parameters based on objective functions that minimize predictive loss between representations of an entity's historical behavior and the entity's future behavior. This is desirable, since it means that entity representations will not change much with typical variation in transaction behavior. Our assumption is that such a model will create entity representations such that entities with different behavior will have distinctly different entity representations.

In a perfect world, we might expect entity behavior to be stable over time. However, in reality there are several factors that cause behavior to change: evolving relationships, seasonal effects, bursty communications due to life events, or simple statistical variance. Because entity behavior varies across time, we will need to optimize our model parameter $\phi = (\theta, k, \epsilon)$ for each application to maximize predictive performance.

We choose two predictive criteria to maximize ϕ. Both criteria are defined for a single node i and our evaluations are based on a score function which is the average of a criterion over the entire node set. We start with a set of time-ordered transactions and split them into pre- and post-period datasets. Entity approximations corresponding to ϕ are built on the pre-set and matched against those built on the post-set for the default parameter setting of $(\theta = 1, k = \infty, \epsilon = 0)$. To avoid sensitivity to the choice of a post-period, we refine our score by replicating with L different pre- and post-periods and taking the average as our objective function. The estimated values maximize the objective function. We further show that our choice makes a difference in a real application relative to the values used by CPV03.

Formally, for the rth ($r = 1, \ldots, R_s$) node in the sth ($s = 1, \ldots, L$) replication and

parameter ϕ, let $A_{rs}(\phi)$ be the entity representation based on the pre-dataset. The edges contained in $A_{rs}(\phi)$ include the top k edges plus the aggregator edge, called `other`. Let B_{rs} denote the representation for the corresponding node based on the post dataset in sth replication and corresponding to the default case of no approximation, that is, $\theta = 1, k = \infty, \epsilon = 0$. Since k=∞, B_{rs} does not have an `other` bin. Then, for a given predictive criterion or similarity score $S(.,.)$, our objective function $F(S, \phi)$ is given by

$$F(S,\phi) = \left(\sum_{s=1}^{L} \sum_{r=1}^{R_s} S(A_{rs}(\phi), B_{rs})/R_s \right) /L. \tag{2.3}$$

For a given S, our estimator is $\hat{\phi} = \text{argmax}_{\phi} F(S, \phi)$.

Next, we discuss our choices of S. For notational simplicity, we replace $A_{rs}(\phi)$ by A and B_{rs} by B. Our first criterion is based on the Dice criterion (Dice 1945), which is commonly used in information retrieval for measuring similarity between documents and queries. The basic Dice Criterion is

$$D(A,B) = \frac{2|A \cap B|}{|A| + |B|}, \tag{2.4}$$

or twice the cardinality of the intersection of the two sets divided by the sum of the cardinalities of the sets. This criterion has the nice property that it is bounded between 0 and 1, with a value of 1 when the two sets have exactly the same node sets.

We extend the Dice criterion to take account of the weights in the pre- and post-period sets. First we normalize the weights for each edge i by defining a normalized edge weight $\mathrm{p}_G(i)$, which normalizes over all edges j in G: $\mathrm{p}_G(i) = \mathrm{w}_G(i)/\sum_j \mathrm{w}_G(j)$. Then let the weighted Dice criterion between A and B equal:

$$\mathrm{WD}(A,B) = \frac{\sum_{[i \in A \cap B]} (\mathrm{p}_A(i) + \mathrm{p}_B(i))}{1 + \sum_{i \neq \texttt{other}} \mathrm{p}_A(i)}. \tag{2.5}$$

Like basic Dice, the weighted Dice criterion is maximized at one when all edges in the predicted pre-set appear in the post-set, and it equals zero if there are no overlaps between the two sets. However, when there is partial overlap, weighted Dice is more sensitive to the edges we are most likely to care about. The term in the denominator is necessary to correct for the case where the predictive set fills up its top-k cases, such that the overflow "`other`" edge is nonzero.

Our second predictive criterion is based on the Hellinger distance (Beran 1977), designed to measure distances between statistical distributions. Applied to our problem, Hellinger distance becomes:

$$\mathrm{HD}(A,B) = \sum_{i \in A \cap B} \sqrt{\mathrm{p}_A(i)\mathrm{p}_B(i)}. \tag{2.6}$$

This sum is also bounded by 0 and 1, and is maximized when all elements of the predicted set appear in the post-period dataset, *with the same normalized weights*.

These two criteria measure related, but slightly different aspects of the validity of the prediction. Both are penalized prediction criteria, designed to penalize predictions incorporating noise edges which do not show up in the post-period dataset. Weighted Dice depends on the proportion of edges in the two periods that belong to the overlap set and does not attempt to minimize the discrepancy between individual weights in the pre- and post-period. Hellinger, on the other hand, gives an added premium if the individual proportions in the pre- and post-sets are similar. Note that a small pre(post) period weight which corresponds to a large post(pre) period weight would contribute more to weighted Dice compared to Hellinger. For cases where the difference between pre- and post-period weights is close to zero, the contribution to both the criteria is approximately the same. This suggests that the performance of weighted Dice should improve more with increasing k and decreasing ϵ relative to Hellinger.

Because the two criteria may imply different parameter values, we need a basis for choosing between criteria. Criterion selection should be application dependent because the goals of different types of network analysis varies. Therefore, when we evaluate parameter sets, we do so within the context of a specific application. For example, in link prediction applications, future links are dependent variables. Thus, the ability to reliably predict the appearance of future relationships is the criterion by which the selection is made. On the other hand, the similarity score between two entities is often used as an attribute for entity classification. When a classification target is the dependent variable, evaluation measures such as classification accuracy and area under the ROC curve may be the application goal. Another application goal may include not only performance evaluation, but also target space and computation requirements. Once the application goal is determined, we select the criterion and optimized parameter set that performs best at the task. In the next section we show how to apply the above construction to the specific application of repetitive fraud, including guidelines on how to set the parameters.

3. INTRODUCTION OF TECHNIQUE IN CONTEXT

This section applies our methods to the repetitive fraud example described in Section 1, where perpetrators of fraud are trying to hide their identities in order to re-establish an account or a presence on the network. Our goal is to identify the fraudulent individuals when they appear as a new identity by analyzing their network behavior. The COI framework allows us to characterize the behavior of fraudulent individuals in a concise manner as entities, and to look for that behavior in new entities appearing on the network.

We need to show that our approximation described in Section 2 is a good representation of an ID's behavior, in that it is a useful predictor of future behavior. In order to show this, we select a random sample of entities from the network, apply our approximation and evaluate it using the two predictive measures that we introduced in Section 2.4, allowing us to fit our model parameter ϕ for this example.

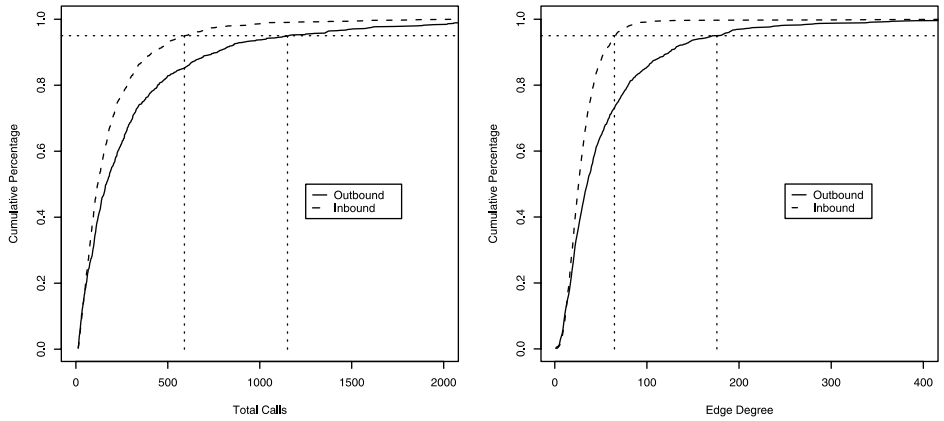

Figure 2. Descriptive statistics of the dataset of randomly selected IDs. Plots show cumulative distributions for total calls (left) and edge degree (right). The distribution for outbound data is shown as a solid line, inbound data is shown as a dotted line. A horizontal line is plotted at the 95th percentile.

3.1 DATA

We collected data on the usage of 1,092 active network IDs over a 12-month period. We see all outbound data from our customers because it is carried on our network, but only a subset of inbound data since calls that do not originate on our network are proprietary data of another firm. Because of this inherent difference, we will analyze the inbound and the outbound portions of entities separately.

Figure 2 shows some descriptive statistics of the 1,092 IDs. The plots show cumulative distributions for total calls and edge degrees for the IDs over a one year period. Both inbound and outbound distributions are plotted. Inbound data has noticeably smaller quantiles for both total calls and edge degree than outbound data at least in part due to the missing data described above. Note that IDs are connected to no more than a few hundred other IDs. The 95th percentile of edge degree is shown as a horizontal line on the plot, and equates to 175 for outbound data and 66 for inbound data.

3.2 TUNING k AND θ

We begin by tuning two of the three parameters: θ, which controls how long transactions endure, and k, which limits how many edges a COI may contain. The parameter ϵ is a tolerance value which we set at the small value of 10^{-5} for now; later we will investigate the robustness of the results to this value.

Although θ and k control different parts of the optimization, optimum values for the two parameters are related. As is true in many networks, most of the overall entity weight lies with a few of the edges with the highest weights. Figure 3 shows this graphically through what we call the *95/95 Plot*, for inbound (left plot) and outbound (right plot). Each line in the plot corresponds to a particular value of θ, from 0.75 up to 0.99. For each of these values, we look at a range of k to see what percentage of the entities have at least 95% of

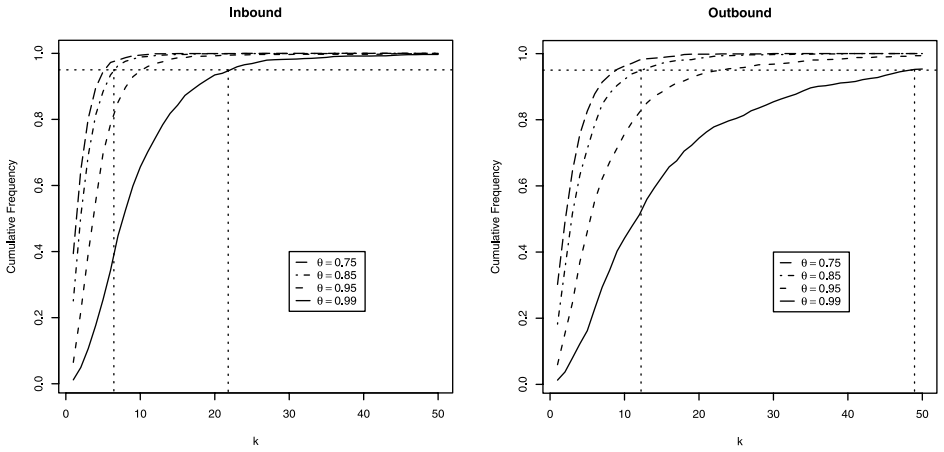

Figure 3. 95/95 plots for inbound (left) and outbound (right). These plots show, for different values of θ, the cumulative distribution of edges needed to reach the 95th percentile of overall weight. The horizontal line at 0.95 and vertical guide lines shows the 95/95 point, the number of edges where 95% of the entities contain at least 95% of their overall weight. For instance, the 95/95 point for $\theta = 0.99$ on the Inbound plot is at $k = 22$.

total weight in their top-k edges. A horizontal line drawn at 0.95 shows where the curve crosses the 0.95 value, what we call the *95/95 point*. The value of k at the 95/95 point shows how many edges need to be included to be assured that 95% of the entities retain 95% of their weight, providing guidance into how many edges we might be able to prune without much harm.

For values of θ closer to one, edge weights decay slower, and more edges contribute substantially to the overall weight of a node, resulting in a higher 95/95 point. For the outbound data with $\theta = 0.85$, the 95/95 point is at 12 edges. With $\theta = 0.99$, that number increases to 48 edges. So, even though some entities may have hundreds of edges (recall from the last section that the 95th percentile of outbound node degree is 175), we need only a few dozen to capture 95% of the weight for 95% of the entities, even with a relatively large θ like 0.99. This exploratory analysis suggests that we might be able to prune quite a few edges from our data, saving in computational storage, while not losing much information.

The 95/95 point also allows us to set ranges for reasonable values of k and θ for investigation. Note that we do not consider values of θ less than 0.75. From Figure 1 we see that for $\theta = 0.75$, weights decay quite quickly, within a week or two . For telecommunications data, we expect that we need to go back much further than that to get a representation that captures the relevant behavior, so it is not desirable for the decay function to be quite so steep. From the 95/95 plot, we see that for $\theta = 0.75$, the 95/95 point is less than 10, which seems inadequate to capture the behavior of many telephone numbers. So we set $\theta = 0.75$ as a lower bound for our investigation. We can also see that with θ near 0.99, we will need to investigate k of at least the 95/95 point of $k = 48$, but values of k much greater than that will probably not add much predictive ability (because the weights on those edges will be small).

To evaluate predictive performance we generated three datasets using a moving window of 10 consecutive months of data. The first nine months of each dataset were used as pre-

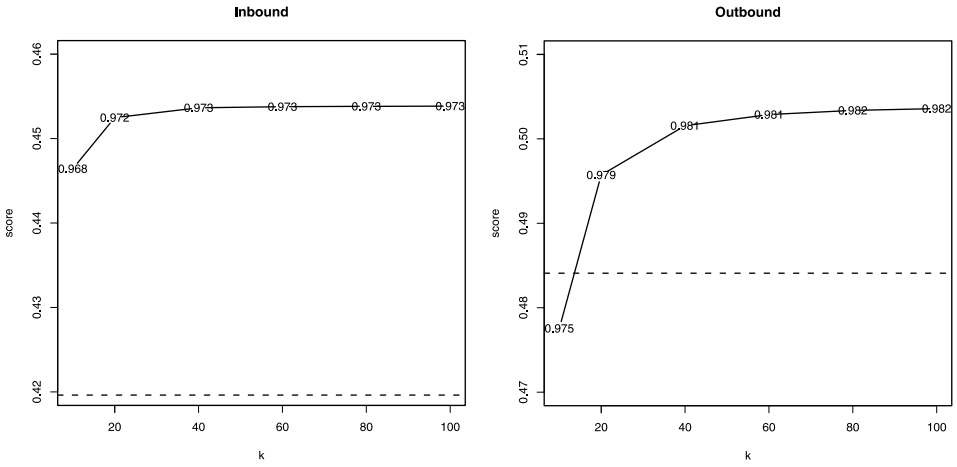

Figure 4. Optimal θ for the Hellinger Distance, inbound (left) and outbound (right). The plot shows the maximized value of θ plotted for a range of values of k, at the point of the score function for that maximized value. The horizontal dotted line shows the default value of $\phi = (1, \infty, 0)$.

period data and the tenth consecutive month as post-period data. Figure 4 shows results of optimizing these two parameters for the Hellinger distance, and Figure 5 shows the same for weighted Dice. In each plot, for selected values of k, we show the value of θ that maximized the predictive score and plot the value of the maximized score. For instance, for the Hellinger plot, if we set $k = 40$, the value of θ that maximized the Hellinger score was 0.973. This resulted in a Hellinger score of 0.454, the y-coordinate for that point.

A key feature of the the plots for both Hellinger and weighted Dice is that the predictive metrics increase monotonically as a function of k, since more overlaps will occur. But there is a point when we get diminishing returns. This point where the curve flattens out is a

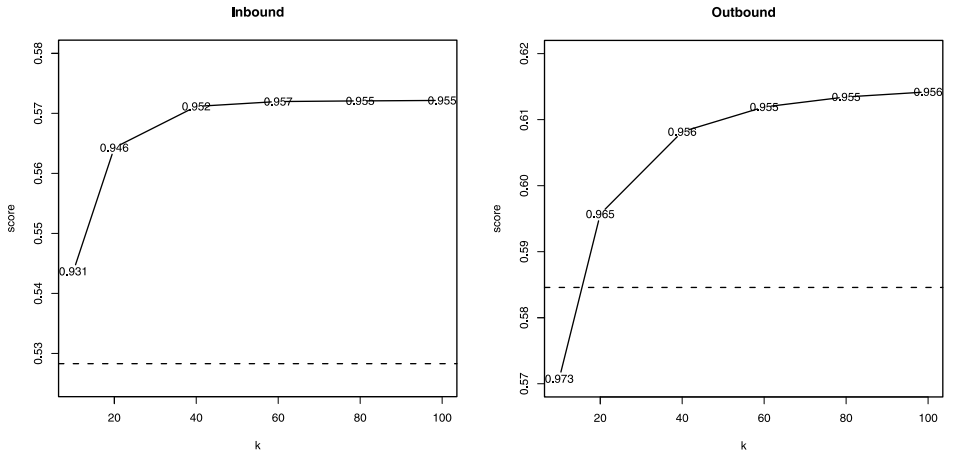

Figure 5. Optimal θ for the weighted Dice Distance, inbound (left) and outbound (right). The horizontal line shows the default value of $\phi = (1, \infty, 0)$.

good candidate for k, since choosing a higher k will not result in any increase in predictive performance. A look at the Hellinger plots shows this point to be at around $k = 20$ for inbound, and $k = 40$ for outbound. The weighted Dice plots seem to suggest slightly larger values of k.

The optimal values of θ show reasonable consistency within a given prediction criterion. For Hellinger, θ is mostly between 0.97 and 0.98, whereas for weighted Dice, it falls between 0.945 and 0.965. We also show how our approximation performs compared to a "baseline" prediction, the arithmetic average of transactions over the pre-period dataset, with no smoothing and no pruning. (This can be represented in our framework as $\phi = (1, \infty, 0)$.) For each plot, we show a dotted horizontal line corresponding to this default case. In each plot, the approximation outperforms the default case for some values of k and θ. This is due to the improvement exponential smoothing provides by increasing the relative importance of recent data over historical data. The improvement over the default case is more pronounced for inbound data than for outbound data, most likely due to our inability to observe all inbound calls.

3.3 TUNING ϵ

For the above section, we set $\epsilon = 10^{-5}$ as a tolerance to prune out edges that are not important. In this section we investigate how robust the results are to this value of ϵ, by plotting the same curves as above for these different values of ϵ. Since ϵ is basically a tolerance value, we want to set it to "do no harm," such that we are not pruning edges too soon before they have decayed sufficiently. So, we look for a value of ϵ that will not affect the predictive results.

Figure 6 shows the results for Hellinger distance and weighted Dice. For Hellinger distance, the results barely change with different values of ϵ, suggesting that scores below 0.1 can be ignored safely according to the criterion, which could allow for a significant amount of pruning. However, for weighted Dice, there is a clear improvement as ϵ approaches 0, indicating that this tolerance value needs to be lower.

3.4 SELECTING CRITERION

The results show that for this dataset, using our representation with appropriate settings for the parameters can improve predictive performance. The Hellinger distance and the weighted Dice give slightly different recommendations given the arguments above. From the Hellinger plots (for inbound) above we might select $k = 20$ with its maximized θ of 0.972 and $\epsilon = 0.1$, while weighted Dice suggests $k = 40$, $\theta = 0.952$, and $\epsilon = 0.00001$. The difference in suggested values is due to characteristics of the score functions themselves. weighted Dice is additive in the weights of the overlapping nodes, and as such is maximized whenever all of the edges predicted from the pre-period data appear in the post-period dataset, regardless of their weights. Hellinger score is multiplicative, and so is much more sensitive to the specific edges that overlap; the score gives more "credit" for matching a high weight node than a low weight node. In a particular application, matching the weights

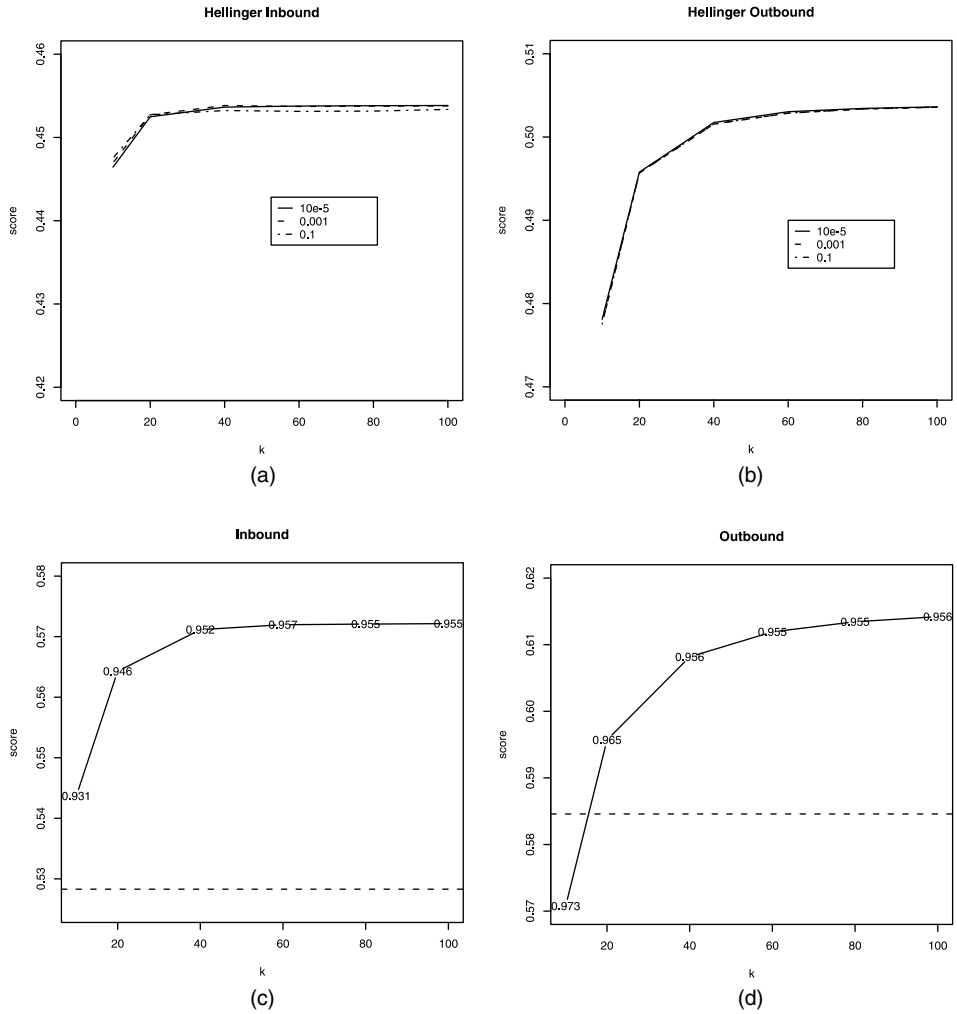

Figure 6. Results for different values of ϵ, shown for Hellinger distance (a) inbound and (b) outbound, and for weighted Dice (c) inbound and (d) outbound.

correctly may or may not be important, and this will play a role in which of these (or other) predictive metrics should be used.

In general, selecting a similarity criterion is similar in spirit to model selection for a given problem and depends very much on the application, nature of data and the purpose of the study. We have suggested and evaluated two such measures, which optimize features important in certain applications. However, other measures might be more useful and appropriate in some other settings. In fact, any metric that defines a distance between networks might be informative, such as those found in Liben-Nowell and Kleinberg (2003). As with model building, constructing appropriate similarity scores is more an art than exact science. However, we have provided a general framework which allows us to plug and play with any user defined match criterion.

In the repetitive fraud example, our goal is to recognize when a known fraudulent case appears as a different ID. In practice, when a fraudulent case is recognized, the entity representation associated with the fraudulent ID is captured and put into a database to compare to future accounts. Future accounts are allowed to establish their entity behavior, and are then compared to the fraud database to look for matches. The biggest evidence for a match is if the new entity has many edges in common with a fraudulent one. This means that they have the same communication profile, in that they communicate with common network IDs. The number of overlapping nodes between the two entities is the most salient factor with the fraud investigators. Therefore, we rely on the weights to rank the important nodes in the top-k. These arguments led to the evaluation of weighted Dice and Hellinger, which both use edge weights.

In order to select the criterion, we evaluate each approximation and similarity criterion in the context of repetitive fraud classification. The classification problem in this example involves distinguishing between *matches*, new accounts that belong to known fraudsters, and *non-matches*, new accounts that do not match to members of our fraud list. For each criterion, we find optimized parameter values for both inbound and outbound behavior. We use the optimized representation to generate similarity scores for each candidate match. We then use the scores as attributes for a classification model. We compare models by area under the ROC curve (AUC), a standard tool used to evaluate classifiers, and select the similarity criterion that maximizes AUC.

3.5 IMPLEMENTATION

We turn now to implementation and evaluation of our fitted parameters in the repetitive fraud application. CPV03 discussed implementation of COI for this problem, using $\phi_c = (0.90, 9, 0.1)$, but did not evaluate the performance of the representation other than to say that it resulted in improved fraud detection. The current process in production uses ϕ_c and identifies 50–100 cases a day to be evaluated by fraud experts. Each case pairs a known fraudulent case with a new account that we believe might belong to the same individual. Each case is then assigned a label by an expert as to whether or not it was truly fraud. This labeling provides us with a post-period dataset to evaluate parameters, independent of the randomly selected set used to optimize the parameters.

In order to compare parameter values, we took a set of 412 actual cases identified from the current process from one week in November, 2004. Of these 412 cases, 217 of them (53%) were ultimately determined to be fraud, that is, the expert concluded via thorough investigation that the new account should be shut down. For each case, we calculated both weighted Dice and Hellinger scores between the old account and the new account for the current $\phi_c = (0.90, 9, 0.1)$ and our optimized fit $\phi_o = (\phi_{oi}, \phi_{oo})$, where ϕ_{oi} and ϕ_{oo} are the optimized parameter sets for inbound and outbound behavior, respectively. (We use the term "optimized" to refer to recommended settings based on our methodology. We realize that we do not technically optimize over three parameters, but believe our solution is optimal for typical user constraints on storage and computation.) We build a classification model using the inbound and outbound similarity scores associated with each pair of candidate

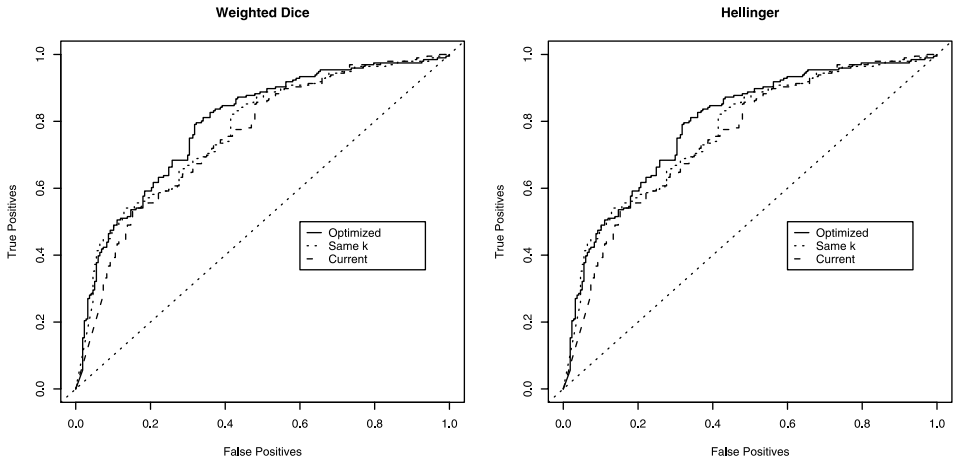

Figure 7. ROC curves for weighted Dice distance (left) and Hellinger distance (right), resulting from application of new parameters to repetitive fraud example. The AUC values for ϕ_o (optimized), ϕ_k (same k), and ϕ_c (current) are (0.831, 0.813, 0.785) and (0.797, 0.771, 0.752) for Hellinger and weighted Dice, respectively. The reference line at the 45-degree line corresponds to the outcome of a random classification model.

matches as attributes. We hope that our new ϕ_o results in increased score values for the cases that eventually were labeled fraud. However, we realize that since both scores are monotonic with increasing k, their resulting scores are almost guaranteed to be higher for all cases (as ϕ_o has a larger k). In order to make clear that our improvement is not simply due to the monotonicity of WD (defined in Section 2) in k, we also investigate ϕ_k which has optimized values for θ and ϵ, but keeps the same k as ϕ_c.

Our goal is to see whether the new values of ϕ_k and ϕ_o allow us to discriminate fraud from non-fraud better than we are able to with ϕ_c. One way to do this is with an ROC curve, which measures the ability of the classifiers built using the different parameters to separate fraud and non-fraud cases. ROC curves plot the false positive rate versus the true positive rate, for different values of a score threshold. A random classifier will fall on the 45-degree line, whereas an ideal classifier tends toward the upper left corner. The performance of each classifier can be evaluated at any particular threshold, or the overall performance can be quantified by the area under the curve (AUC). Figure 7 shows ROC curves for the three classifiers for both weighted Dice and Hellinger. The AUC values for ϕ_o, ϕ_k, and ϕ_c are (0.831, 0.813, 0.785) and (0.797, 0.771, 0.752) for Hellinger and weighted Dice, respectively. We see that in both cases, the ϕ_o is the best, followed by ϕ_k and finally ϕ_c. In this case the best overall AUC results from the model built with Hellinger and the optimized parameter ϕ_o. Based on AUC scores, Hellinger would be our ultimate choice of criterion.

We now show that the improvement in Hellinger scores from our optimized parameters are statistically significant over the already implemented ϕ_c. Table 1 shows results from calculating Hellinger on the three-parameter sets. First, we see that within each of the ϕ values, Hellinger scores separate the fraud and non-fraud cases by giving higher scores to the fraud cases (e.g., 0.37 for fraud vs. 0.12 for non-fraud for ϕ_c). Next we want to see if ϕ_o is an improvement on ϕ_c. Using ϕ_o results in an increase in average Hellinger scores across

Table 1. Hellinger Values from Application of Optimized Parameters to Repetitive Fraud Data for Three-Parameter Values. A star in the "Diff from ϕ_c" column indicates significant difference in a paired *t* test ($\alpha = 0.05$).

Parameter set	*Fraud?*	*Avg Hell.*	*Diff from ϕ_c*	*t stat (p val)*
ϕ_c: current	Y	0.37	0	
	N	0.12	0	
ϕ_o: optimized	Y	0.44	0.06 $\star$	-3.30 (0.001)
	N	0.14	0.02 $\star$	
ϕ_k: same *k*	Y	0.40	0.03 $\star$	-2.22 (0.02)
	N	0.12	0.005	

both fraud (0.44 vs. 0.37) and non-fraud (0.14 vs. 0.12) cases. Both of these increases are significant ($\alpha = 0.05$), as indicated by a star in the "Diff from ϕ_c" column. To show that ϕ_o improves our ability to discriminate fraud from non-fraud, we used a paired t test to test the null hypothesis that the increase in scores for the fraud cases was the same as the increase in scores for the non-fraud cases. The results (as shown in the "t stat" column) indicate that using ϕ_o indeed results in better separation of the classes. Similarly, using ϕ_k results in an overall increase in the scores over ϕ_c (although the increase in non-fraud scores is not significant), and the hypothesis test confirms that the increase in scores for fraud cases is greater than the increase in scores for non-fraud cases.

The implications of a high AUC and statistically significant improvement in prediction are that by setting ϕ wisely we can better rank our cases. This means that our fraud experts, who are only able to work a small number of cases per day, are better used. Practically, we expect a change in the parameters to result in a few extra fraud cases caught per week. In addition, the better we are at separating out fraud, the closer we get to our ultimate goal, where we have enough confidence in our scores that we can do automatic fraud detection, without a fraud expert in the loop.

Below, we summarize the main steps involved in our procedure. We note that although our method is not automatic and involves subjective steps like visual inspection of plots, it is done once in an offline fashion and hence does not compromise the real time nature of our representation at all. In fact, judicious selection of parameters improves our real time analysis. For a given predictive criterion S, the steps are:

1. Select a grid of values for θ and k. A rough guide for θ can be obtained by considering the weights assigned to past observations (as in Figure 1); values of k can be obtained by considering graphical techniques like the 95/95 plot discussed previously.

2. Split the data into multiple pre- and post-sets to evaluate the objective function F in (2.3). In general, we recommend 2-4 splits.

3. Fixing ϵ to a small number, optimize θ for fixed values of k and choose optimal values of θ and k corresponding to the point of diminishing returns which is located graphically.

Conduct a ϵ sensitivity analysis by varying ϵ in a small neighborhood. If results are sensitive, we recommend choosing a small ϵ, if there is little sensitivity then a higher value of ϵ is preferred. Again, graphical plots are helpful in accomplishing this step.

4. If functions derived from entity representations are used in some supervised learning tasks (e.g., repetitive fraud example in our case), it is possible to compare different score functions and even a finite set of competitive parameter settings through ROC curves.

We end this section by commenting on the general utility of our method. We remark that our parameterized representation is useful in applications where the goal is to summarize the global behavior of a dynamic network over time. It may not be useful in applications (e.g., network security, national security) where interactions that are rare are important and should not be pruned.

4. GENERALIZING TO OTHER APPLICATIONS

To demonstrate the generalizability of our framework, we present less detailed analyses for two other dynamic networks. The first experiment is on Internet browsing and the second experiment is on academic e-mail. We know nothing about the content of the transactions involved and we cannot link transactor labels to individuals. The analyses demonstrate that our framework performs substantially better at predicting future entity behavior in dynamic networks than the default case on both datasets.

4.1 WEB LOGS

Generating informative customer signatures from Web usage may enable firms to offer personalized target marketing and services to their consumers and thereby increase profits (Kasanoff 2001). The use of our method to create reliable customer signatures, which can be segmented or evaluated on an individual basis, can be an important step in a creating pre-period data for online marketing prediction tasks. For example, firms may use signatures to select the appropriate bag of goods offered to a new or returning customer.

Our first experiment is a preliminary investigation of how much Web user behavior varies over time. We use the ComScore panelist dataset from Media Metrix on Internet browsing and buying behaviors of 100,000 users across the United States for a period of six months, July 2002–December 2002 (available at *http://wrds.wharton.upenn.edu*). Web data and telecommunications data differ in important ways. On the Web, the dynamic network is defined by a bipartite graph (Gross and Yellen 2004) and user behavior is characterized by much more activity per day in both duration and number of relationships. We use the first five months to predict behavior in the sixth month. We optimize our model parameters to maximize similarity between the time periods based on the Hellinger and weighted Dice criteria (Figure 8). The Hellinger distance may be the more appropriate criterion for Web applications because the relative edge weights may be helpful when distinguishing between the many users that frequent the same subset of popular Web sites.

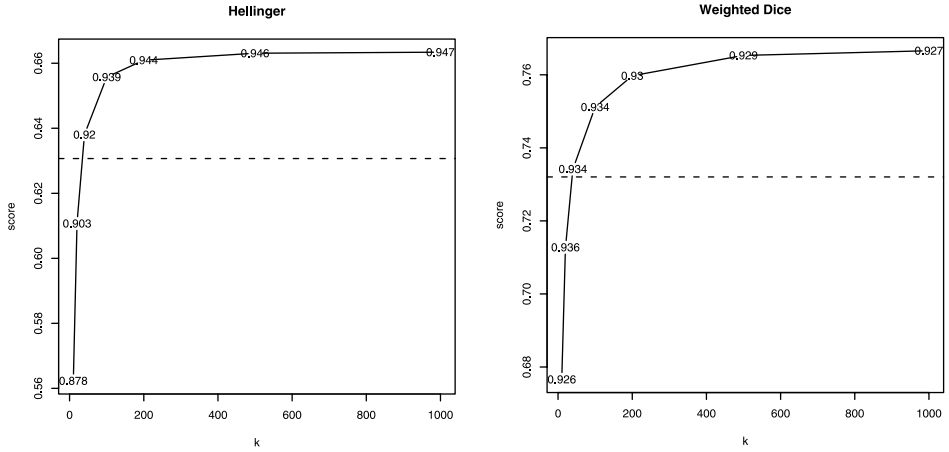

Figure 8. Optimal θ for the Web data, using Hellinger distance (left) and weighted Dice (right). The horizontal line shows the default value of $\phi = (1, \infty, 0)$.

From Figure 8 we can draw a few conclusions. First, using our representation with appropriate settings for the parameters can improve predictive performance over the default case. Second, predictive performance continues to significantly improve at a much higher k than in telephone usage behavior. Based on our datasets, we find the frequency of transactions as well as the number of relational ties between individuals and Web sites is greater than that between individuals on a telephony network. Therefore, in this example, more daily storage may be required to achieve peak performance. Finally, we see that the actual score values in this domain are higher, indicating less within-user variance than in our other examples. Therefore, predictive performance is relatively higher despite having to consider more transactions, and this allows for a more representative signature to be built on each user.

4.2 E-MAIL LOGS

The vast amount of data stored on electronic communication such as the data found in e-mail logs, enables the discovery of communication patterns between individuals, organizations and entire communities. One interesting challenge for organizations is to compare how their formal organization structure compares to their *communities of practice* (Wenger and Snyder 2000), which are their informal collaboration and communication clusters bound by shared expertise and shared objectives. E-mail has been used to identify communities of practice and to identify leadership within organizations (Tyler, Wilkinson, and Huberman 2003). Another interesting problem, where e-mail networks have been used, is in the identification of pockets of expertise (Schwartz and Wood 1993). Our framework may be used not only to identify both individuals and groups from e-mail usage behavior, but also to extend these types of analyses to explore how communities of practice change over time (Gongla and Rizzuto 2001).

For this e-mail experiment, we selected 2,000 e-mail accounts from approximately

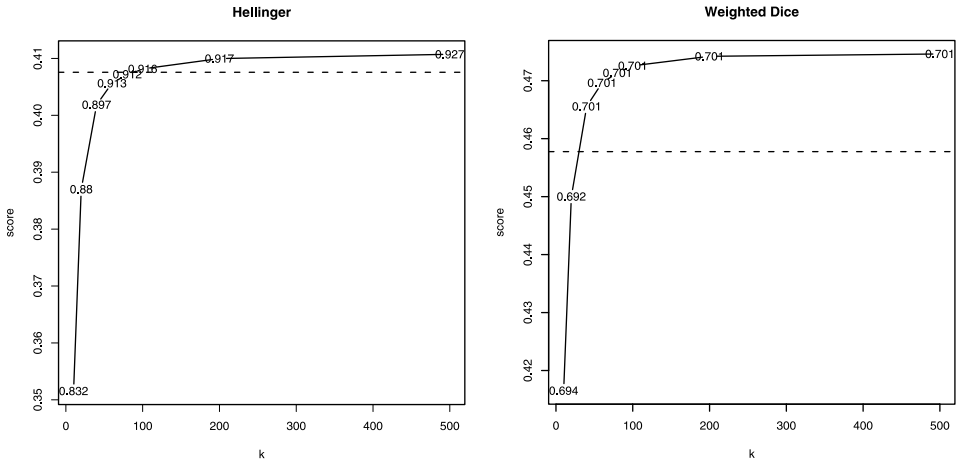

Figure 9. Optimal θ for the e-mail data, using Hellinger distance (left) and weighted Dice (right). The horizontal line shows the default value of $\phi = (1, \infty, 0)$.

16,000 e-mail users associated with a large school at a Northeast university. We were able to collect two months of data from November 18, 2002, to January 19, 2003. We used our framework to select the appropriate model to represent the first month of data for evaluation against the second month of data (Figure 9).

Although we outperform the default case in this example, results in Figure 9 illustrate that our method does not significantly outperform the default setting. We believe the performance is the result of having only one month's worth of pre-period data. Additionally, we find we need a much lower θ for e-mail than when predicting other types of behavior. This result indicates recent behavior is more indicative of future behavior than past behavior. Some intuition about these results may be taken from what we know about e-mail communication. Individual e-mail transactions do not have duration attributes. Instead, we are only able to observe that a message between two people exists. Also, multiple e-mail transactions are often used to communicate and clarify ideas that may have been addressed in one transaction using other methods of communication. Therefore, we may need more data to assess the relative strength of the relationships between nodes to get optimal performance.

5. DISCUSSION

Efficiently and effectively representing large evolving dynamic networks is difficult. Dynamic network evolution has been investigated in domains such as the Web (Brewington and Cybenko 2000) and social networks (Jin, Girvan, and Newman 2001). In addition, dynamic Bayesian networks (Dean and Kanazawa 1989) extend Bayesian networks by representing the state of the world as a set of variables, and modeling the probabilistic dependencies of the variables within and between time steps. Similarly, dynamic probabilistic relational models (Sanghai, Domingos, and Weld 2003) extend probabilistic relational models proposed by (Friedman, Getoor, Koller, and Pfeffer 1999). Although these relational representations are dynamic representations, they are more concerned with the

entire network and the probabilistic relationships between the nodes on the network. When making entity approximations, we are interested in a compact representation that captures the dynamics of an individual entity on the network. To assess the correlation between local node representations of past and future node behavior on dynamic networks, we adopted the Dice criterion from the machine learning literature and the the Hellinger distance from the statistics literature.

We presented a compact dynamic network graph representation for local node analysis. By representing the network as a graph, we aggregate multiple transactions between a pair of nodes to a single edge. In doing so we lose the actual time stamp information in this approximation, and might therefore lose important information. However, our method allows for incremental updates to the representation, which is efficient when many analyses must be made. In our applications, we used one day increments, but the increments could be shortened to the scale of hours or even minutes if needed. This would push the analysis closer to the type of real-time analyses that are needed in security or systems monitoring applications.

Our main contribution is a framework for optimizing the parameter settings in a principled way for our proposed dynamic network representation. The framework can be used to evaluate any local representation that has a goal of predicting future behavior. In addition to optimizing the parameters for predictive performance, our framework visually displays the performance gains from increasing the amount of information kept (increasing k). In addition, we show the improvement over the default setting of using all edges without weight decay.

We evaluated our representation on call detail, e-mail, and Web log network data. In all cases, optimizing the parameters outperforms the default setting. We also find that the optimal parameter settings are different across datasets, in ways that are informative about the data. In addition, our methods applied to a repetitive fraud application are better at catching fraud than those currently employed.

One important benefit of our methodology that we did not focus on is computational storage. A typical phone call data record has hundreds of fields associated with it, but if we just consider the essential data of originating node, terminating node, time stamp, and duration of call, this can be captured in about 10 bytes per record after compression. In a typical day, there are approximately 350 million calls, so storage for one day will be about 5.6 GB, or about 170 GB a month. We store and update our entity representations using a domain-specific C-based programming language called Hancock (Cortes et al. 2000), publicly available for noncommercial use at *http://www.research.att.com/~kfisher/hancock/*. In Hancock, the entire indexed entity representation database takes up 8 GB (compressed), only about 1.5 days worth of raw data.

We plan to extend this work by evaluating other criteria or techniques that can be used for parameter optimization, including methods that are not predictive in nature, such as multinomial likelihood methods. In addition, there has been research in statistical link prediction that can be used in comparing the similarity between two graphs. Liben-Nowell and Kleinberg (2003) compared and contrasted many such measures, including PageRank, latent semantic analysis, and other low rank matrix methods. We will borrow from this

literature to investigate which of these measures might be useful in our framework.

In our work we set parameters jointly for all entities in a given dataset. We could optimize parameters on a per-entity basis, but that would significantly increase the computational complexity of the process. A middle ground could be to define a small number of clusters of entities based on behavior. For instance, Web users could be classified into power users, daily news and weather checkers, and casual, occasional users. Each entity representation might have different parameter values, and as long as we could assign entities to clusters appropriately, this should result in more robust entity representations.

One possible extension will replace exponential smoothing with a Kalman filter (Gray, Smith-Carroll, and Jordan 2004; Meinhold and Singpurwalla 1983) for every pair of nodes on the network. This will require storing the state and variance estimates which doubles the storage but the updating is still linear and can be done efficiently in an online fashion. Indeed, on reaching steady state the Kalman filter converges to the exponential smoothing model we propose (West and Harrison 1997). However, steady state may not always persist and is likely to be accompanied by bursty behavior during which the Kalman filter will adjust θ appropriately for each pair of nodes reflecting recent behavior. Accounting for correlation among state estimates for edges belonging to the same entity may increase efficiency but would require considerable elaboration of techniques to maintain computational tractability for massive networks.

We would like to investigate the potential for more complex entity representations, including those that include nodes that are more than one graph hop away from the seed node. It has been our experience that the information gain that results from expanding the entity has been minimal. However, in some cases it might help, such as for a telecommunications company that has missing data because it only sees calls from its own customers.

When we optimize parameters on a random set of network IDs, we assume that fraudulent users behave the same as others. However, prior work in fraud detection details the need for taking the adversarial behavior of fraudsters into account when building fraud detection models (Fawcett and Provost 1997). We plan to incorporate the method presented in recent work on adversarial classification (Dalvi et al. 2004), which considers the fact that adapting adversaries, may behave differently than the average user, or even worse, they may learn the system and change behavior in response to our methods.

ACKNOWLEDGMENTS

The authors thank Daryl Pregibon and Foster Provost for their very helpful comments on an early draft of this article. We also wish to thank an associate editor and three referees for comments that helped improve the manuscript.

[Received February 2005. Revised November 2005.]

REFERENCES

Albert, R., Jeong, H., and Barabasi, A. L. (2000), "Error and Attack Tolerance of Complex Networks," *Nature*, 406, 378–382.

Anderson, R. M., and May, R. M. (1991), *Infectious Diseases of Humans: Dynamics and Control*, New York: Oxford University Press.

Barabasi, A. L., and Albert, R. (1999), "Emergence of Scaling in Random Networks," *Science*, 286, 509–512.

Beran, R. (1977), "Minimum Hellinger Distance Estimates for Parametric Models," *The Annals of Statistics*, 5, 445–463.

Breuer, M. A. (1966), "Coding Vertexes of a Graph," *IEEE Transactions on Information Theory* 12, 148–153.

Brewington, B. E., and Cybenko, G. (2000), "How Dynamic is the Web?" *Computer Networks-the International Journal of Computer and Telecommunications Networking*, 33, 257–276.

Cortes, C., Fisher, K., Pregibon, D., Rogers, A., and Smith, F. (2000), "Hancock: A Language for Extracting Signatures from Data Streams," in *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining*, pp. 9–17.

Cortes, C., Pregibon, D., and Volinsky, C. (2003), "Computational Methods for Dynamic Graphs," *Journal of Computational and Graphical Statistics*, 12, 950–970.

Dalvi, N. N., Domingos, P., S. Sanghai, Mausam, and Verma, D. (2004), "Adversarial Classification," in *Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining*, pp. 99–108.

Dean, T., and Kanazawa, K. (1989), "A Model for Reasoning About Persistence and Causation," *Computational Intelligence*, 5, 142–150.

Dice, L. R. (1945), "Measures of the Amount of Ecologic Association Between Species," *Ecology*, 26, 297–302.

Eppstein, D., Galil, Z., and Italiano, G. F. (1999), *Dynamic Graph Algorithms*, Boca Raton, FL: CRC Press.

Erdős, P., and Renyi, A. (1960), "On the Evolution of Random Graphs," *Bulletin of the International Statistical Institute*, 38, 343–347.

Fawcett, T., and Provost, F. J. (1997), "Adaptive Fraud Detection," *Data Mining and Knowledge Discovery*, 1, 291–316.

Friedman, N., Getoor, L., Koller, D., and Pfeffer, A. (1999), "Learning Probabilistic Relational Models," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pp. 1300–1309.

Gavoille, C., and Paul, C. (2003), "Distance Labeling Scheme and Split Decomposition," *Discrete Mathematics*, 273, 115–130.

Gongla, P., and Rizzuto, C. R. (2001), "Evolving Communities of Practice: IBM Global Services Experience," *IBM Systems Journal*, 40, 842–862.

Gray, J. E., Smith-Carroll, A. S., and Jordan, L. (2004), "The Solution to the Lyapunov Equation in Constant Gain Filtering and Some of its Applications," in *36th Annual Southeastern Symposium on System Theory*, Atlanta, Georgia, pp. 21–25.

Gross, J. L., and Yellen, J. (2004), *Handbook of Graph Theory*, Boca Raton, FL: CRC Press.

Jin, E. M., Girvan, M., and Newman, M. E. J. (2001), "Structure of Growing Social Networks," *Physical Review E*, 6404, 167–256.

Kasanoff, B. (2001), *Making it Personal: How to Profit from Personalization Without Invading Privacy*, Cambridge, MA: Perseus.

Korman, A., and Peleg, D. (2003), *Labeling Schemes for Weighted Dynamic Trees (Extended Abstract)*, Volume 2719 of *Lecture Notes in Computer Science*, Heidelberg: Springer-Verlag.

Liben-Nowell, D., and Kleinberg, J. (2003), "The Link Prediction Problem for Social Networks," in *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, New York: ACM Press, pp. 556 – 559.

Macy, M. W., Kitts, J., Flache, A., and Benard, S. (2003), "Polarization in Dynamic Networks: A Hopfield Model of Emergent Structure," in *Dynamic Social Network Modeling and Analysis*, Washington, DC: National Academies Press, pp. 162–173.

Meinhold, R. J., and Singpurwalla, N. D. (1983), "Understanding the Kalman Filter," *The American Statistician* 37, 123–127.

Newman, M. E. J. (2003), "The Structure and Function of Complex Networks," *Siam Review*, 45, 167–256.

Sanghai, S., Domingos, P., and Weld, D. S. (2003), "Dynamic Probabilistic Relational Models," in *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9–15, 2003*, eds. G. Gottlob and T. Walsh, San Francisco: Morgan Kaufmann, pp. 992–1002.

Scheinerman, E. R. (1990), "An Evolution of Interval-Graphs," *Discrete Mathematics*, 82, 287–302.

Schwartz, M., and Wood, D. (1993), "Discovering Shared Interests Among People Using Graph Analysis of Global Electronic Mail Traffic," *Communication of the ACM*, 36, 78–89.

Solla, P. D. (1976), "A General Theory of Bibliometric and Other Cumulative Advantage Processes," *Journal of the American Society for Information Science*, 27, 292–306.

Tyler, J. R., Wilkinson, D. M., and Huberman, B. A. (2003), "Email as Spectroscopy: Automated Discovery of Community Structure Within Organizations," in *Proceedings of the First International Conference on Communities and Technologies*, Amsterdam: Kluwer, pp. 81–96.

Van Alstyne, M., and Brynjolfsson, E. (2005), "Global Village or Cyber-Balkans? Modeling and Measuring the Integration of Electronic Communities," *Management Science*, 51, 851–868.

Watts, D. J. (2002), "A Simple Model of Global Cascades on Random Networks," in *Proceedings of the National Academy of Sciences of the United States of America*, 99, pp. 5766–5771.

Wenger, E., and Snyder, W. (2000), "Communities of Practice: The Organizational Frontier," *Harvard Business Review*, (Jan-Feb), 139–145.

West, M., and Harrison, J. (1997), *Bayesian Forecasting and Dynamic Models* (2nd ed.), Springer Series in Statistics. New York: Springer.

Winters, P. R. (1960), "Forecasting Sales by Exponentially Weighted Moving Averages," *Management Science*, 6, 324–342.