Building an Effective Representation for Dynamic Networks

Shawndra B. Hill, Deepak K. Agarwal, Robert Bell, Chris Volinsky*

February 19, 2005

Abstract

A dynamic network is a special type of network which is comprised 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 paper, we represent dynamic graphs using a data structure introduced by Cortes et. al. [9]. 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 preliminary analysis on web logs and email networks.

Keywords: approximate subgraphs, dynamic graphs, exponential averaging, fraud detection, transactional data streams.

1 Introduction

*First author is at Stern School of Business, New York, New York, all other authors are members of research staff at AT&T Research, Florham Park, New Jersey 07934.

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. A dynamic graph is built from a list of transactions with time stamps and may include other important information such as the duration of the transaction or the physical location of the transactors. Put another way, a dynamic graph is a collection of nodes and edges where the nodes and edges are subject to discrete changes, such as additions or deletions [13].

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 exists 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. While data on dynamic networks are readily available, representing the dynamics in a way that is meaningful for analysis is a challenge.

There has been a vast amount of recent research on real world networks and graphs as representations of those networks [23]. Historically, studies have focused primarily on the global aspects of networks, such as scale-free or small world properties[4]. Global properties allow us to learn about overall characteristics of a graph, such as link structure, average path length, graph diameter, degrees of separation, etc. Global graph topology has informed the design of robust infrastructures [1] and has implications for the rate infectious diseases spread [2] as well as the adoption rate of fads [28]. In addition, global models of network growth such as triad completion[19], polarization and balkanization, and cumulative advantage [26] also commonly referred to as preferential attachment[4] 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 use of the entire network in order 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 the analysis can be distributed and the behavior of individual nodes can be directly represented.

One example of a domain using individual node analysis is fraud detection in telecommunications. One type of fraud is *repetitive fraud*, where we have an individual who 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 methods to obscure its true identity, perhaps through identity theft, in order to obscure the fact that they are a fraudster. Therefore, we cannot use standard record-linkage techniques to link the old fraudulent account and the new account together. However, we assume the new fraudulent ID has communication patterns similar to the old one (for example, the new ID will communicate with the same people that the old fraudulent ID did). In this way, we can use information present in the network of transactions to identify fraudsters.

The challenge is to explore all new IDs on the network to see which exhibit the same network patterns as the known fraudulent ID. This suggests a local analysis, focusing on modelling each transactor. Nonetheless, when analyzing and comparing local relational features of transactors, processing queries against the millions of other network transactors in a short amount of time is still a challenge. To facilitate this, we employ an efficient representation of the dynamic transactional network. Repetitive fraud was the motivating application for our research, and this paper develops our methodology in this context. However, our representation is applicable to any domain with transactional data.

Two main challenges exist for dynamic network representation. The first challenge, is to represent dynamic graphs efficiently so that the massive volumes of data in these domains can be processed in a reasonable timeframe. The second challenge, is to account for the dynamic nature of transactional data by capturing the most relevant information while eliminating spurious information that does not provide important information about the transactors.

Recently, dynamic network representations have been proposed to address these issues using node labeling schemes. Local node labeling schemes allow one to infer the distance between [16] or adjacency of [6] two nodes from their labels. We know of two dynamic local distance labeling schemes that allow for distributed incremental updates, weighted dynamic trees [21] and Communities of Interest (COI) graphs [9]. Weighted

dynamic trees assume a fixed tree graph structure. Therefore, global topological reassignment is needed when nodes and edges are added or deleted. The assignment is costly for dynamic graphs that have sufficiently high node and edge growth or decay rates. In contrast, COI graphs [9] use a more general parametrized vector-based approximation that handles distributed incremental revisions to the labels. The parameter setting, which can be set by employing only a sample of the entire dynamic network, is required only once. The COI representation is stored in an efficient database so that transaction behavior on individual nodes can be queried.

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 we perform 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 for how 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.
- **Application of technique.** We apply our technique to different domains to show its validity for a wide range of data streams. We demonstrate that predictions based on the approximated graph outperform predictions based on non-approximated data on several real world data sets.

The remainder of the paper is organized as follows: Section 2 will define the representation of a dynamic graph using the COI representation proposed in [9], and introduce the three parameters that define the representation; Section 3 will discuss how the parameters should be set for specific applications, using telecommunications data with repetitive fraud as an example; Section 4 will discuss some other applications; and finally we present a discussion in Section 5.

2 Dynamic Graph Approximation

In this paper, we propose a framework for representing large dynamic networks for the class of problems where the level of analysis is at the transactor. We approximate dynamic networks by 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 its transactions with other nodes on the network over time, and is a subgraph of the entire communication network.

We know three things about the frequency of transactions between entities, 1) the frequency of interaction between entities evolves through time, 2) relationships between entities become stale when the frequency of transactions goes to zero, and 3) the frequency of transactions are 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.

Cortes, et al [9] were able capture entity change in a concise representation that changes smoothly through time. The authors use the representation to catch repetitive fraud on telecommunications networks. That paper studied the repetitive fraud problem, and approximated an entity's behavior over time by a parameterizable notion of a dynamic graph called the Communities of Interest (COI) graph. The COI representation proved to be successful in the fraud application. However, parameters were chosen in a heuristic, ad hoc fashion, and there was little discussion about model selection, parameter fitting, properties of the approximation in general, and most importantly, the loss associated with the approximation. The COI representation is the only work we know of which directly modeled the evolution of entity behavior to analyze dynamic graphs. This paper extends and generalizes that paper 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 also be the focus in applications but are beyond the scope of this paper.

2.1 Approximation Objectives

Our approximation of the dynamic network using the COI representation, should contain as much relevant information about entities as possible, 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. 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.
- **Prediction.** Optimize the representation of entity behavior to maximize predictive performance of an entity's behavior in a future time period.

These objectives mesh together very well: Summarization supports efficiency by saving time - by summarizing historical behavior into a single edge in a graph, we do not have to query a massive database every time we want to learn about significant entity relationships. Simplification supports efficiency by saving space and time - by reducing the overall size of the graph, we can update and analyze faster. Simplification also creates better summaries since noisy transactions are not accounted for in the summarizations. Prediction is the evaluation metric, a target by which we can evaluate and compare parameter sets for the representation in the context of an application and across applications.

In our approximation, the edges are represented 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. We simplify by pruning out noise globally by removing edges which have a weight which has decayed below a threshold (ϵ), since we

have high confidence that this is a stale edge. The combined parameter set $\phi = (\theta, k, \epsilon)$ enables tuning of the approximation. We estimate the parameters to maximize our accuracy in predicting future behavior. We achieve this by minimizing a similarity score (which may be application dependent) between representations based on data in some training period and test period. The selected parameters result in a concise signature for each node and captures the most pervasive historical behavior.

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 [9]. 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. Let A and B be two graphs. We first define the weighted sum of these two graphs using the graph operator $\oplus$ such that if $G = \alpha A \oplus \beta B$, with positive scalars α and β, then $N(G) = N(A) \cup N(B)$, $E(G) = E(A) \cup E(B)$, 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 G_t as a graph representing all transactions up to and including g_t, and is defined as a weighted collection of all of the time periods:

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

This definition of G_t includes all historic transactions from the beginning of time. 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 results in an exponentially weighted moving average: $\omega_i = \theta^{t-i}(1-\theta)$, where $0 \leq \theta \leq 1$. Here θ is a (scalar) parameter that allows recent behavior to have more relative weight (θ near 0) or less relative weight (θ near 1) in influencing the current graph. This form of weight function is convenient in the sense that Eq.(1) can be expressed in recurrence form:

$$G_t = \theta G_{t-1} \oplus (1-\theta) g_t. \tag{2}$$

This form is well-known in statistics as exponential smoothing [30]. It provides a smooth dynamic evolution of G_t. The iterative nature of the updating allows us to incorporate the information from all previous time periods without incurring the management and storage of 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 Eq. (2) as the definition of a dynamic graph at time t. The parameter θ is useful in describing both the amount of historical data that should be considered as well as the priority given to recent data. In the limits, $\theta = 1$ is a simple average of all time periods, while $\theta = 0$ only incorporates 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 displays this graphically. The figure shows the decay function of the weight given to a one hour transaction for different values of θ. At time $t = 1$ day, the weight on the edge (in seconds) is $3600(1-\theta)$, and decreases multiplicatively by θ every day thereafter. With $\theta = 0.75$, this transaction's 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 the Figure), it takes almost two years. As we will see later, the decay rate desired will depend on the application, and can be set by using an appropriate θ. [9] sets $\theta = 0.90$ based on heuristic arguments, and later in the paper 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. The weights on the edges will get arbitrarily small over time and at some point should be deleted. We define a

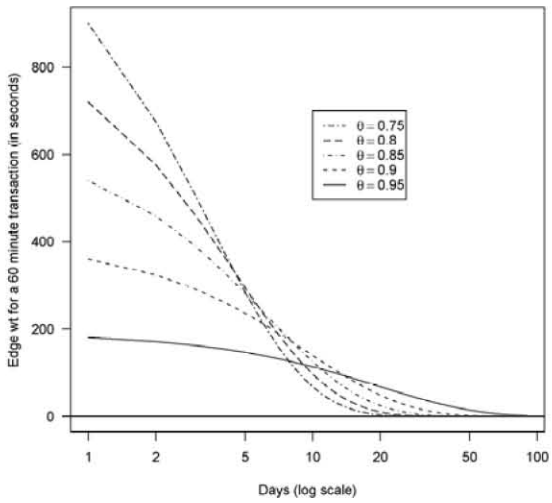


Figure 1: *Contribution of a 60 minute (3600 second) call to edge weight as a function of days. For a smaller θ, this call effectively reaches zero in a few weeks, whereas for a larger θ, it may take one hundred days or more.*

parameter ϵ to be a global threshold such that when an edge weight falls below ϵ, it is removed from the graph. This noise reduction step can have a significant impact on the size of the graph over time. A weight that has decayed to a small value means that it was either a small weight to begin with, or it has not been observed in a long time. Both of these are potential indicators of a transient transaction, meaning that they are connections 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. We should gain in efficiency by pruning these noise edges. There is a subtle trade-off here; we want to delete edges if they are transient or stale, since they do not represent an important relationship for an entity, and they take up 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.

We are also interested in reducing noise with the entity representation. 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 a power law, which states that the vast majority of nodes have small in- and out- degrees. An exponentially smaller

set of nodes have very high in- and out-degrees. Usually these nodes are not interesting entities. They are 'super-nodes' that everyone is connected to, like Google in the web, or the toll-free directory in the phone network. If we are looking to entities to be signatures for nodes' usage behavior, the fact that one is connected to a super-node is not a distinguishing factor, and so we can prune many of these edges without losing vital information.

Typically most entities have a large percentage of total weight accounted for by a small percentage of 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 edges with extremely small weights might be stale or transient as noted before, with high probability we can assert that such an edge would not be observed in future and hence should be dropped. Removing 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). We also add an aggregator edge, called `other`, which collects the weight associated with the edges not in the top k. This new aggregator edge effectively replaces a subset of edges of this subgraph such that it contains the same total weight of the edge subset. By removing a possibly large number of edges that account for a small amount of the total weight, this pruning can have a significant impact on computational efficiency.

2.4 Evaluation Criteria: Predictive Performance

We have now defined an approximation of a dynamic graph in terms of the parameter $\phi = (\theta, k, \epsilon)$. In order to evaluate a given set of parameters, we compare it to a default case where there is no exponential smoothing, and no pruning of edges. For this case $k = \infty$, $\epsilon = 0$, and we use the notation $\theta = 1$, since as θ approaches 1 in Equation (2) G_t approaches the no-smoothing case, a graph where the edge weights are simple averages of transaction weights between pairs of nodes. The parameter θ determines the emphasis placed on recent data. Adjusting k and ϵ determines the amount of pruning. Having defined and interpreted our parameters, we explain how to estimate these 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 [9] used the framework described in the previous sections, and used $\phi = (.90, 9, 0.1)$, each parameter being set via heuristic arguments. We present a more principled way to set the COI parameters based on objective functions that minimize predictive loss between our representation of an entity's historical behavior and the entity's future behavior. By using a predictive criterion to optimize our parameters, we are selecting a model that minimizes variance between the representation and future observed 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 the same in one time period as it is in the next. But in reality there are several factors, which cause behavior to change: evolving relationships, seasonal effects, bursty communications due to life events, or simple statistical variance. Since entity behavior varies across time as well as application, we will need to optimize our model parameter $\phi = (\theta, k, \epsilon)$ for each application to maximize the predictive performance.

We choose two predictive criteria to maximize ϕ. Both criteria are defined for a single node i, the value of a criterion for the entire node set being the average of its value over all nodes. In our framework, we have two sets of transactions involving node i, usually corresponding to two distinct time periods. One is a training set which we use to build a prediction for what will happen during a distinct test period immediately following the training period. To assess the performance of a given parameter value ϕ_A, we apply ϕ_A to the training set to create A_i. The edges contained in A_i include the top k edges plus the aggregator edge, called `other`. We want to see how well this predicts what actually occurs, so we create B_i from the test set using the parameter values associated with no approximation, or $\phi_B = (1, \infty, 0)$. Since $k{=}\infty$, B_i does not have an `other` bin. Now, we have two graphs, A_i and B_i and the predictive criterion is simply a measure of graph distance between the two. Our two criteria are actually score functions that are maximized when the graphs are identical.

Dropping the suffix i from our notation, our first criterion is based on the Dice criterion [12], which is commonly used in information retrieval for measuring similarity between documents and queries. For two

sets A and B, the Dice Criterion is:

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

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 are identical.

We extend the Dice criterion to take account of the weights in the training and test set. First we normalize the weights within training and test set by defining normalized edge weights p for any graph 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:

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

weighted Dice has the same same properties as Dice in that it is maximized at one when all edges in the predicted training set appear in the test set, and it equals zero if there are no overlaps between the two sets. 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 non-zero.

Our second predictive criterion is based on the Hellinger Distance [5], designed to measure distances between statistical distributions. Applied to our problem, Hellinger Distance becomes:

$$\mathrm{HD}(A,B) = \sum_{j \in A \cap B} \sqrt{p_A(j) p_B(j)}.$$

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

These two criteria measure complimentary, but slightly different aspects of the validity of the prediction. Both are penalized prediction criteria, designed to penalize predictions which incorporate noise edges which do not show up in the test set. Weighted Dice depends on the total proportion in the training and test sets which 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 training and test 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. We use these two criteria as our guide to set the parameters in order to maximize the ability of our approximation to predict future transactions, while minimizing noise. For each criteria, we select the parameter set that gives the best average performance on the entire training and test set.

Finally, a choice must be made between criteria. Criteria selection is 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, predictive performance, which in essence is the ability to reliably predict the appearance of future relationships is the criteria 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 criteria 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

There are many different types of telecommunications fraud. In this section, we will apply our methods to the repetitive fraud example described in Section 1, where a perpetrator of fraud is trying to hide his identity 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 buy analyzing their network behavior. The framework we have just described 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.

3.1 Data

We collected data on the usage of 1092 active network IDs over a twelve month period. We see all outbound data from our customers because it is carried on our network, but only the subset of inbound data that originates from another of our customers because these are proprietary data of another firm. Therefore, there is an inherent difference between outbound and inbound data. Because of this we will analyze the inbound and the outbound portions of the entities separately.

Figure 2 shows some descriptive statistics of the 1092 IDs. The plots show cumulative distributions for total calls and edge degree 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. We believe this results is 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

The parameters each control different parts of the optimization. θ represents how steep the decay is, and the closer θ is to one, the longer the weight takes to die out. k represents how many important edges there are per entity; we want to set it low enough that it removes the noise but high enough so that it retains any important links for the entity. Therefore, it is intuitive that for a larger θ, edges will take longer to fall to incremental values, and might necessitate a larger k. The parameter ϵ is a tolerance value which we set at the small value of 10^{-5}, later we will investigate the robustness of the results to this value.

We can show this relationship between θ and k graphically. 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). A 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

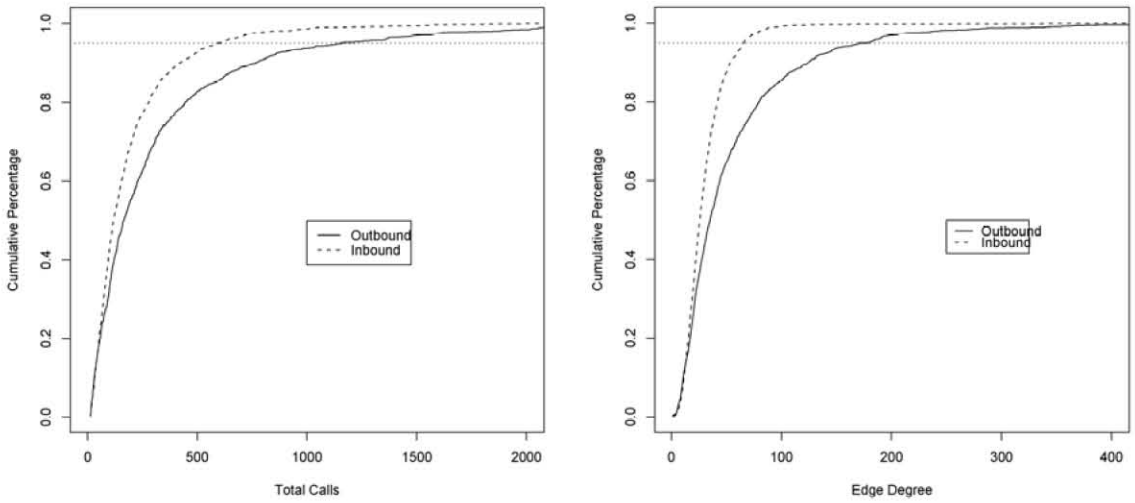


Figure 2: *Descriptive Statistics of the data set of randomly selected IDs. Plots show cumulative distributions for total calls (left) and edge degree (right). The distribution for outbound data is shown as a solid line, inbound data is shown as a dotted line. A horizontal line is plotted at the 95th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

k to see what percentage of the entities have at least 95% of its weight in its 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. The 95/95 point gives guidance into how many edges we might be able to prune while maintaining almost all of the weight of the collection of entities.

For values of θ closer to one, edge weights decay slower, and so more edges will contribute substantially to the overall weight coming out of a node, resulting in a higher 95/95 point. For the outbound data with $\theta = 0.90$, the 95/95 point is at 15 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 percent of the weight for 95 percent 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 most telephone numbers. So we set $\theta = 0.75$ as a lower bound for our investigation. We can also see that if a θ near 0.99 is possible, 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 so low).

To evaluate predictive performance we generated three datasets using a moving window of ten consecutive months of data. The first nine months of each dataset were used as preperiod data and the tenth consecutive month as test data. The selected criteria were computed for each of three datasets in the test period and the results averaged. 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 a range of values of k, we plot the value of θ

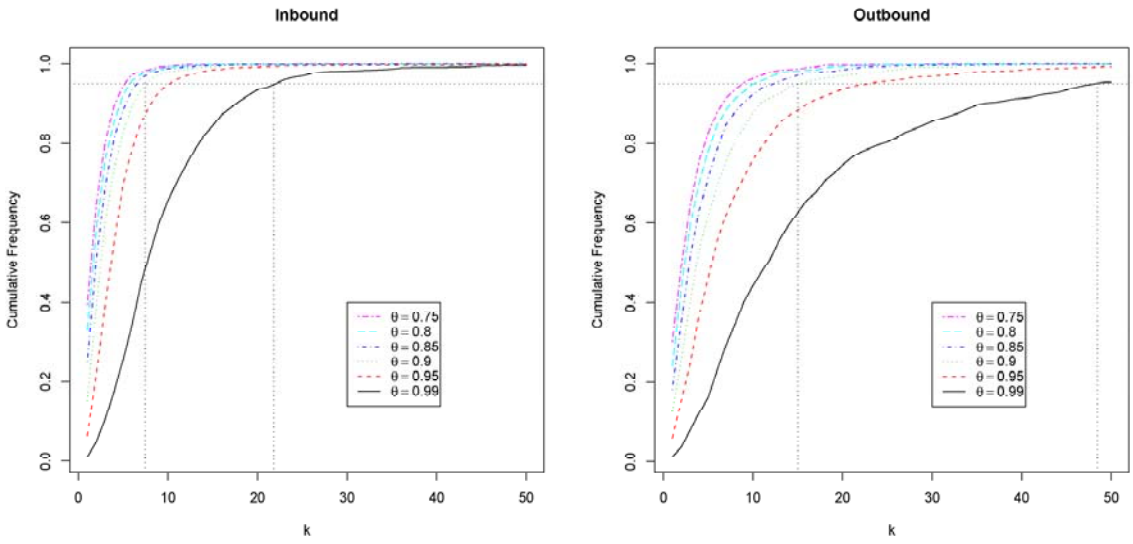


Figure 3: *95/95 plots for inbound (left) and outbound (right). These plots show, for different values of* θ*, the cumulative distribution of edges needed to reach the 95th percentile of overall weight. The horizontal line at 0.95 and vertical guide lines shows the 95/95 point, the number of edges where 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* $k = 22$

which maximized the predictive score, and 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 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.

We also show how our approximation performs compared to a "baseline" prediction. The default case we

considered was one where the prediction made was simply the arithmetic average of transactions over the training set, 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. We can see in all of our plots that the approximation has the ability to outperform the default case. 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. A difference between inbound and outbound is expected due to our inability to observe all inbound calls.

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.

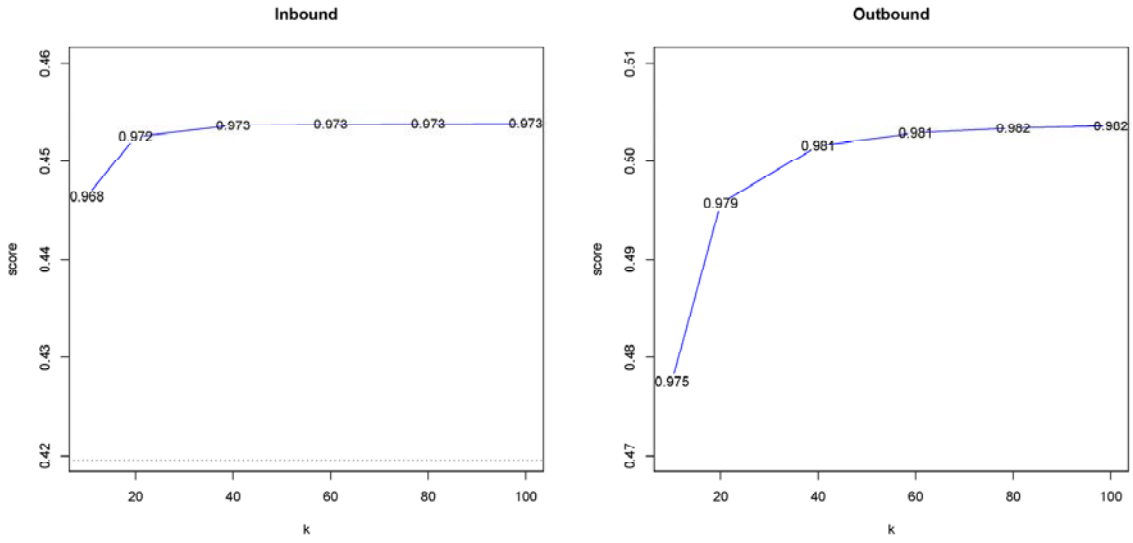


Figure 4: *Optimal θ for the Hellinger Distance, inbound (left) and outbound (right). The plot shows the maximized value of θ plotted for a range of values of k, at the point of the score function for that maximized value. The horizontal dotted line shows the default value of $\phi = (1, \infty, 0)$.*

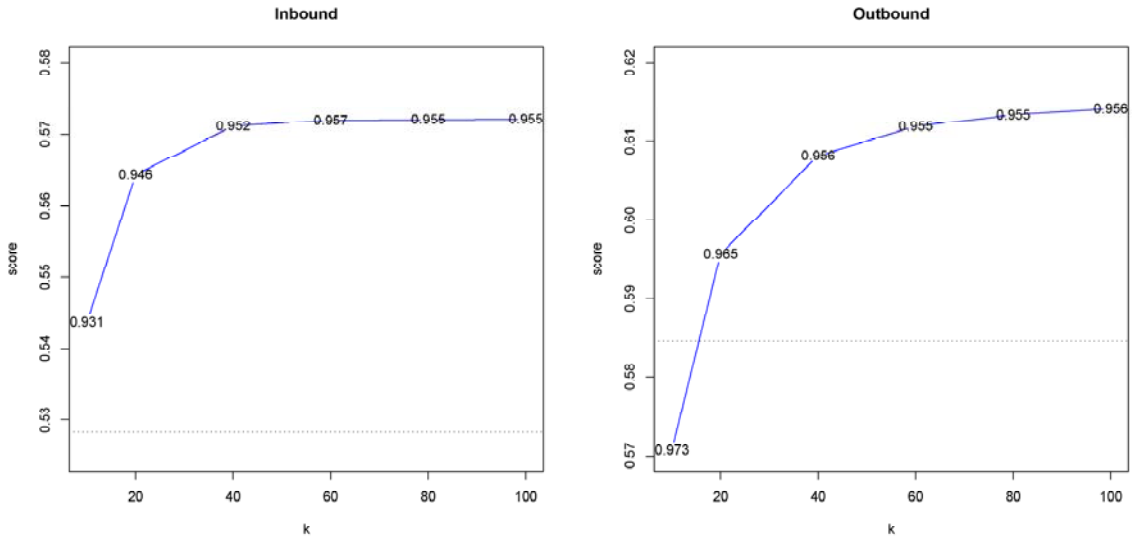
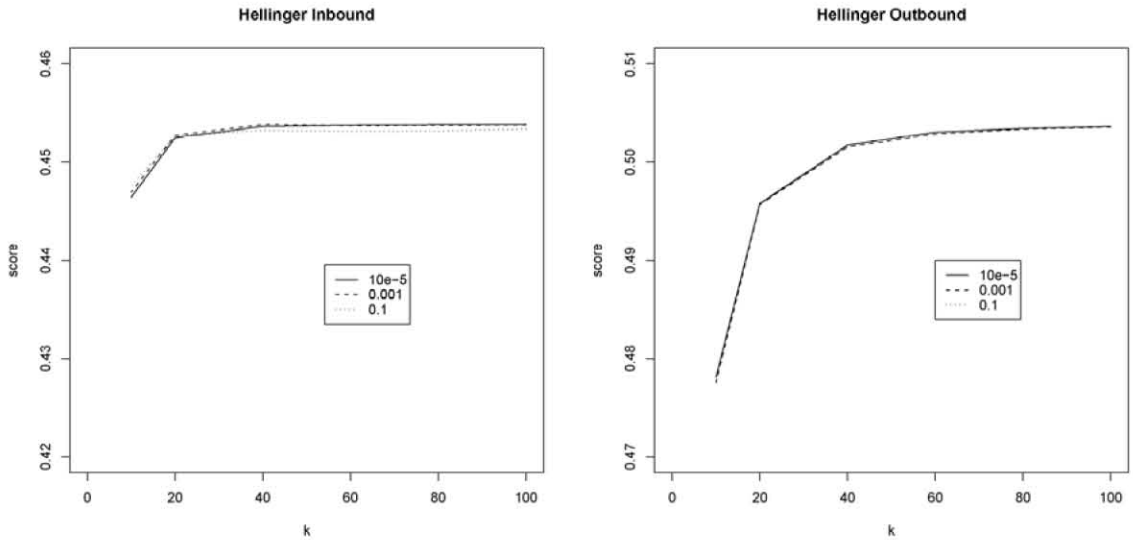


Figure 5: *Optimal θ for the weighted Dice Distance, inbound (left) and outbound (right). The horizontal line shows the default value of $\phi - (1, \infty, 0)$.*

3.3 Tuning ϵ

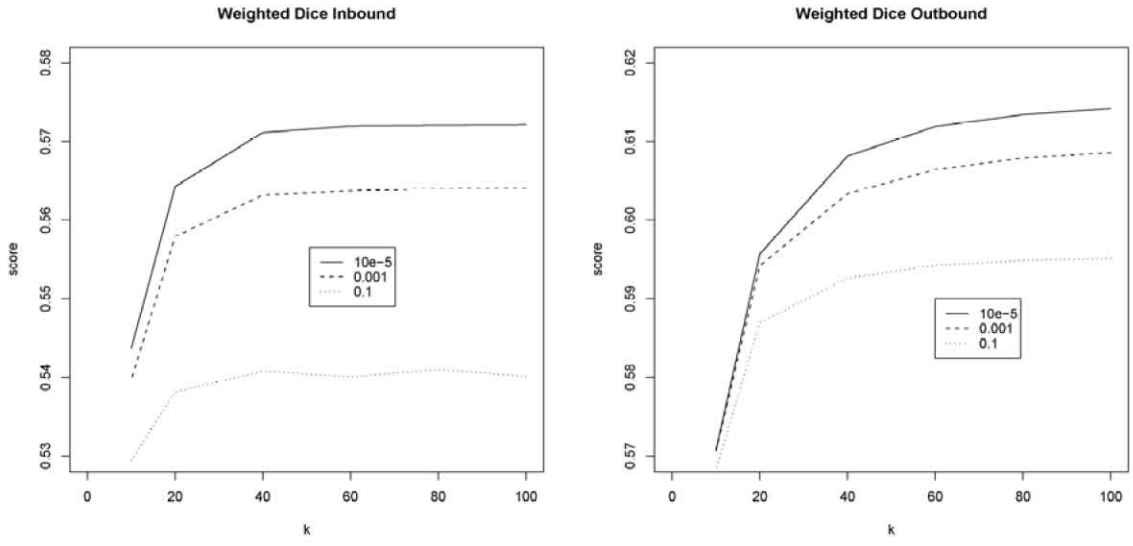
For the above section, we set $\epsilon = 10^{-5}$ as a tolerance to prune out edges that are not important. In this section we investigate how robust the results are to this value of ϵ, by plotting the same curves as above for several different values of ϵ. Since ϵ is basically a tolerance value, we 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.

The results are in Figure 6 for Hellinger Distance and weighted Dice. Here we get the interesting result that for Hellinger Distance, the results barely change with different values of ϵ, suggesting that a value of 0.1 has the same performance as 0.0001. This could allow for a significant amount of pruning while not affecting approximation performance. However, for weighted Dice, there is a clear improvement as ϵ approaches 0, indicating that this tolerance value needs to be lower.



(a)

(b)



(c)

(d)

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.*

3.4 Selecting Criterion

The results show that for this data set, 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 $\epsilon = 0.1$, which 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 training data appear in the test data, regardless of their weights. Hellinger score is multiplicative, and so is much more sensitive to the specific edge that overlaps; the score gives more "credit" for matching a high weight node than a low weight node. In a particular application, matching the weights 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 two different features we have found important in most applications we have worked with but remark other measures might be more useful and appropriate in some other settings. In fact, any metric which defines a distance between networks would suffice, such as those found in [22]. Just as the case is 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 abase to look for matches.

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 fraudulent 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 the same other network IDs. The fact that there is overlap, and the number of overlapping nodes between the two

entities, are the salient facts with our 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 utilize 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 *nonmatches*, new accounts that belong to paying customers. For each criterion, we find optimized parameter sets for both inbound and outbound behavior. We use the optimized representation to generate similarity scores for each candidate match. We then utilize 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. For this application, we select the similarity criterion that maximizes AUC.

3.5 Implementation

We turn now to implementation and evaluation of our fitted parameters in our repetitive fraud application. [9] discussed implementation of COI to this particular repetitive fraud problem, but made strictly heuristic arguments for using $\phi_c = (0.90, 9, 0.1)$, and 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 test set 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[1] parameter sets for inbound and outbound behavior, respectively We build a classification

[1]We use the term "optimized" to refer to recommended settings based on our methodology. We realize that we do not

model using the inbound and outbound similarity scores associated with each pair of candidate matches as attributes. We hope that our new ϕ_o results in increased score values for the cases that eventually were labelled fraud. However, we realize that since both scores are monotonic with increasing k, there 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 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 would also like to see 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 separates 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). So all of the ϕ are able to discriminate fraud from non-fraud. Next we want to see if ϕ_o is an improvement on ϕ_c. For both the fraud and non-fraud cases the scores for ϕ_o (0.44 and 0.14) are greater than for ϕ_c (0.37 and 0.12). These increases are both significant ($\alpha = 0.05$), as indicated by a star in the "Diff from ϕ_c column. However, the increase in score for the fraudulent cases is bigger than the increase in score for the non-fraudulent cases,

technically optimize over three parameters, but believe our solution is optimal for typical user constraints on storage and computation.

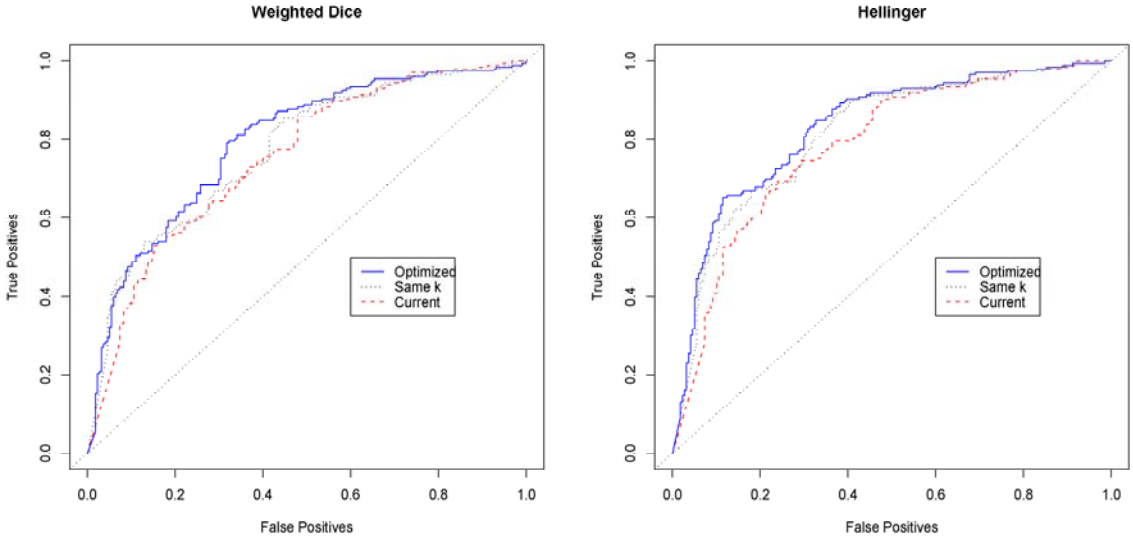


Figure 7: *ROC curves for weighted Dice distance (left) and Hellinger distance (right), resulting from application of new parameters to repetitive fraud example. The AUC values for ϕ_o (Optimized), ϕ_k (Same k), and ϕ_c (Current) are (0.831, 0.813,0.785) and (0.797, 0.771,0.752) for Hellinger and weighted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

and the difference in these increases is significant (as shown in the "t-stat" column) showing that ϕ_o results in better separation of the classes. We also investigate ϕ_k to correct for the overall score inflation, and the table shows that scores for the fraud cases increase significantly, whereas scores for the non-fraud cases do not change, resulting in an overall improvement in discrimination (although not as significant as for ϕ_o).

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 utilized. 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.

ParameterSet	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	

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 ($\alpha = 0.05$).*

4 Generalizing to Other Applications

The purpose of this section is to demonstrate the generalizability of our framework to different domains. We present two experiments in less detail to show proof of concept. The experiments are on dynamic network transactions taken from two real world datasets. The first experiment is on Internet browsing and the second experiment is on academic email. It is important to note that we know nothing about the content of the transactions involved and we cannot link transactor labels to individuals. In this section, we 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

Enabling firms to understand the specific evolving needs of individual customers, allows firms to participate in ongoing personalization and precise target marketing. Generating informative customer signatures from Web usage may enable firms offer personalized target marketing and services to their consumers and thereby increase profits [20]. 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 training data for online marketing prediction tasks. For example, firms may utilize signatures to select the appropriate bag of goods offered to a new or returning customer.

Our first experiment is a preliminary investigation of how similar web user behavior is over time. We use the ComScore panelist dataset from Media Metrix on Internet browsing and buying behaviors of one hundred thousand users across the United States for a period of six months, July 2002-December 2002 (available via Wharton Research Data Services - http://wrds.wharton.upenn.edu). Web data and telecommunications data are somewhat different in nature. On the Web, the dynamic network is defined by a bipartite graph [18] 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 minimize the loss between the time periods based on two loss functions, Hellinger and weighted Dice (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 websites.

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 a higher k than in telephone usage behavior. Based on our data sets, 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. 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 Email Logs

The vast amount of data stored on electronic communication such as the data found in email 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* [29], which are their informal collaboration and communication clusters bound by shared expertise and shared objectives. Email has been used to identify communities of practice [27] and

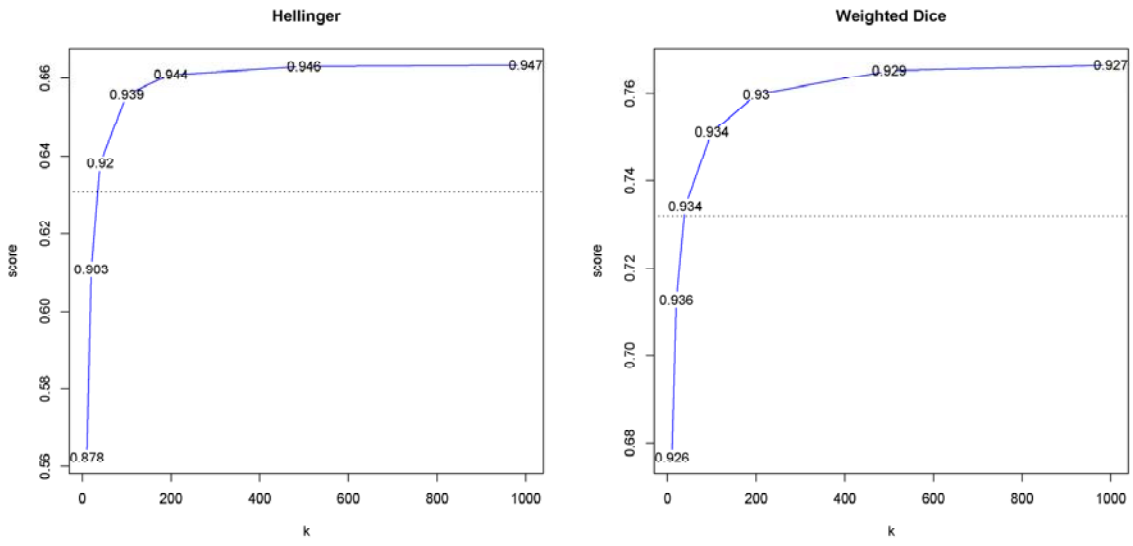


Figure 8: *Optimal θ for the Web data, using Hellinger Distance(left) and weighted Dice (right). The horizontal line shows the default value of $\phi - (1, \infty, 0)$.*

to identify leadership [3] within organizations. Another interesting problem, where email networks have been used, is in the identification of pockets of expertise [25]. Our framework may be used not only to identify both individuals and groups from email usage behavior, but also to extend these types of analyses to explore how communities of practice change over time [17].

For this email experiment, we selected 2,000 email accounts from approximately 16,000 email users with a large school at a Northeast university. We were able to collect two months of data from Nov 18, 2002 to Jan 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 training data. Additionally, we find we need a much lower θ for email 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 email communication. We know individual email transactions do not have duration attributes. Instead, we are only able to observe that a message between two people exists. We also know multiple email 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 asses the relative strength of the relationships between nodes to get optimal performance.

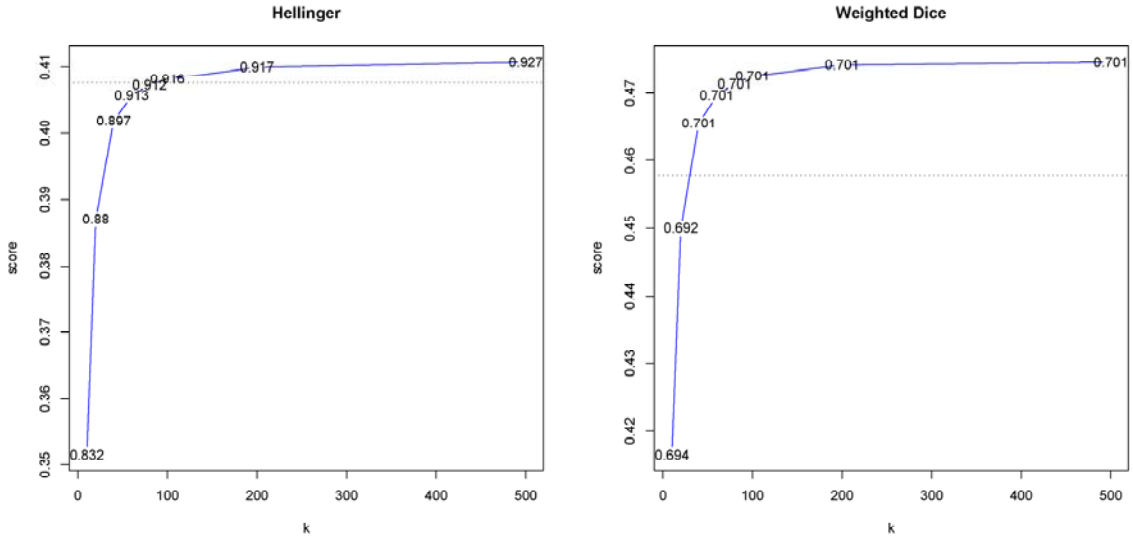


Figure 9: *Optimal θ for the Email data, using Hellinger Distance (left) and weighted Dice (right). The horizontal line shows the default value of $\phi = (1, \infty, 0)$.*

5 Discussion

Efficiently and effectively representing large evolving dynamic networks is difficult. Dynamic network evolution has been investigated in domains such as the Web [7] and social networks such as friend networks [19]. In addition, Dynamic Bayesian networks[11] extend Bayesian networks to consider dynamics by represent-

ing the state of the world as a set of variables, and model the probabilistic dependencies of the variables within and between time steps. Similarly, dynamic probabilistic relational models [24] extend probabilistic relational models proposed by [15]. While 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. In this paper, we employed two similarity measures for assessing the correlation between local node representations of past and future node behavior on dynamic networks. Our choices corresponded to similarity scores that would be familiar to a machine learning audience, who would be familiar with the Dice criterion, or a statistics audience, who might be more familiar with the Hellinger Distance.

We presented a compact dynamic network graph representation for local node analysis. By representing the network as a graph, we are aggregating multiple transactions on the same edge 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.

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 suggests the visualization of the performance gains with increasing the amount of information kept (increasing k). In addition, we visualize the improvement over the default setting of using all edges without weight decay.

We used our framework to evaluate our representation on call detail, email, and web log network data. We found that by optimizing the parameters we outperform the default setting in all cases. We also find that the optimal parameter settings are different across data sets, 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 less than 40 bytes. In a typical day, there are approximately 350 million calls, so storage for one day will be about 14 GB, or about 420 GB a month. Compressed we can get this information down by factor of 2.5. We store and update our entity representations using a domain-specific C-based programming language called Hancock [8], publicly available for non-commercial use at

`http://www.research.att.com/~kfisher/hancock/.`

In Hancock, the entire indexed entity representation database takes up about 8 GB (compressed).

We have several plans to extend this work. We plan to evaluate other criteria or techniques that can be used for parameter optimization, including methods that are not predictive in nature, such as multinomial likelihood methods. Any scoring function that calculates a distance between two simple graphs could work, and it would be interesting to see which of these perform best.

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.

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.

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[14]. We plan to incorporate the method presented in recent work on adversarial classification [10], 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.

6 Acknowledgements

The authors would like to thank Daryl Pregibon and Foster Provost for their very helpful comments on an early draft of this paper.

References

[1] R. Albert, H. Jeong, and A. L. Barabasi. Error and attack tolerance of complex networks. *Nature*, 406(6794):378–382, 2000.

[2] R. M. Anderson and R. M. May. *Infectious diseases of humans : dynamics and control*. Oxford science publications. Oxford University Press, Oxford ; New York, 1991.

[3] P. Ball. E-mail reveals real leaders. *Nature*, 2003.

[4] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[5] R. Beran. Minimum hellinger distance estimates for parametric models. *Annals of Statistics*, 5(3):445–463, 1977.

[6] M. A. Breuer. Coding vertexes of a graph. *IEEE Transactions on Information Theory*, IT12(2):148–153, 1966.

[7] B. E. Brewington and G. Cybenko. How dynamic is the web? *Computer Networks-the International Journal of Computer and Telecommunications Networking*, 33(1-6):257–276, 2000.

[8] C. Cortes, K. Fisher, D. Pregibon, A. Rogers, and F. Smith. Hancock: A language for extracting signatures from data streams. In *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining*, 2000.

[9] C. Cortes, D. Pregibon, and C. Volinsky. Computational methods for dynamic graphs. *Journal of Computational and Graphical Statistics*, 12:950–970, 2003.

[10] N. N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining*, pages 99–108, 2004.

[11] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, Volume 5(3):142–150, 1989.

[12] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[13] D. Eppstein, Z. Galil, and G. F. Italiano. *Dynamic graph algorithms.* CRC Press, 1999.

[14] T. Fawcett and F. J. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.

[15] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 1300–1309, 1999.

[16] C. Gavoille and C. Paul. Distance labeling scheme and split decomposition. *Discrete Mathematics*, 273(1-3):115–130, 2003.

[17] P. Gongla and C. R. Rizzuto. Evolving communities of practice: IBM global services experience. *IBM Systems Journal*, 40(4):842–862, 2001.

[18] J. L. Gross and J. Yellen. *Handbook of graph theory.* Discrete mathematics and its applications. CRC Press, Boca Raton, 2004.

[19] E. M. Jin, M. Girvan, and M. E. J. Newman. Structure of growing social networks. *Physical Review E*, 6404(4):167–256, 2001.

[20] B. Kasanoff. *Making it personal : how to profit from personalization without invading privacy.* Perseus, Cambridge, MA, 2001.

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

[22] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth international conference on Information and knowledge management*, pages 556 – 559. ACM Press, 2003.

[23] M. E. J. Newman. The structure and function of complex networks. *Siam Review*, 45(2):167–256, 2003.

[24] S. Sanghai, P. Domingos, and D. S. Weld. Dynamic probabilistic relational models. In G. Gottlob and T. Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 992–1002. Morgan Kaufmann, 2003.

[25] M. Schwartz and D. Wood. Discovering shared interests among people using graph analysis of global electronic mail traffic. *Communication of the ACM*, 36(8):78–89, 1993.

[26] P. D. d. Solla. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American Society for Information Science*, 27:292–306, 1976.

[27] J. R. Tyler, D. M. Wilkinson, and B. A. Huberman. Email as spectroscopy: automated discovery of community structure within organizations. In *Proceedings of the First International Conference on Communities and Technologies*, pages 81–96. Kluwer, 2003.

[28] D. J. Watts. A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(9):5766–5771, 2002.

[29] E. Wenger and W. Snyder. Communities of practice: The organizational frontier. *Harvard Business Review*, (Jan-Feb):139–145, 2000.

[30] P. R. Winters. Forecasting sales by exponentially weighted moving averages. *Management Science*, 6(3):324–342, 1960.