PathSim and Laplacian Weights Preliminary Results

Willliam Boshell
Technology, Cybersecurity, and Policy Program
University of Colorado Boulder
Boulder, CO, USA
william.boshell@colorado.edu

***Abstract*—This paper covers theoretical and mathematical concepts behind a HinDom classification system for network traffic. It first describes a heterogeneous information network (HIN), then describes the process and equations that it uses to classify domains the network encounters. This is part of a series of multiple projects that build the HinDom system and assumes matrices derived from the network traffic have already been computed and pruned. It takes a (Laplacian) weighted average of matrix products (metapaths) to produce a single similarity matrix through the PathSim algorithm. This is then run through a transductive classifier to label domains as benign or malicious.**

I. Introduction

This project seeks to implement the PathSim algorithm with Laplacian weights on network traffic data from the University of Colorado Boulder's Office of Information Technology (OIT) and run it through a transductive classifier. This is part of a larger series of projects to approach this data as a HinDom system. Notably, a key assumption is that six matrices have already been computed from the network traffic.

II. Definitions

A. Terms

- Heterogeneous Information Network (HIN): Given a graph $G =< V, E >$, where V is the set of nodes and E is the set of edges, let m be the number of types of objects represented by V and p be the number of types of relationships represented by E. G is considered a HIN if $m \geq 2$ or $p \geq 2$ [1].
- Network Schema: $T_G =< A, R >$ is the network schema of the HIN $G =< V, E >$, where A is the set of object types and R is the set of relationship types [1].
- Metapath: Given a network schema $T_G =< A, R >$, a metapath P of length L defines a composite relation $R = R_1 \circ R_2 \circ \ldots \circ R_L$, where $\circ$ is the relation composition operator. P is denoted as $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \ldots \xrightarrow{R_l} A_{L+1}$ [1].
- Transductive Classification: Given a HIN $G =< V, E >$ and a subset of labeled nodes $\tilde{V} \subseteq V$, transductive classification is to predict labels for nodes in $V - \tilde{V}$ [1].
- HinDom System: A HinDom system consists of the following five parts:
  1) Data Collector. This is essentially a log of DNS data.
  2) HIN Constructor. This defines six matrices from the DNS data that will be detailed in the HIN Matrices subsection.
  3) Graph Pruner. This removes unusual domains, popular domains, large clients, inactive clients, and rare IPs [1].
  4) Metapath Combiner. This creates six different metapaths from the HIN matrices and computes similarity matrices using an algorithm called PathSim. This is the main focus of the project and will be discussed in depth in the next section.
  5) Transductive Classifier. This defines a transductive classifier to give the probability that a domain is malicious. This will also be a part of the project and will be discussed in the next section.

B. HIN Matrices

- Client-query-Domain (Q): This denotes when domain i is queried by client j.
- Client-segment-Client (N): This denotes whether clients i and j belong to the same network segment.
- Domain-resolve-IP (R): This denotes when domain i is resolved to IP address j.
- Domain-similar-Domain (S): This denotes the character level similarity between domains i and j,
- Domain-cname-Domain (C): This denotes whether domains i and j are in a CNAME record.
- IP-domain-IP (D): This denotes whether IP addresses i and j are mapped to the same domain.

III. Algorithms

From this point forward, it will be assumed that the HIN Matrices have already been computed and pruned.

A. Metapaths

The first step is to compute six different metapaths defined as follows [1]:

1) $d \xrightarrow{s} d$ denotes domains similar to each other on the character level and is represented by the matrix S.
2) $d \xrightarrow{C} d$ denotes the cname relationship among domains and is represented by the matrix C.
3) $d \xrightarrow{Q} c \xrightarrow{Q^T} d$ denotes domains queried by the same clients and is represented by the matrix product QQ^T.
4) $d \xrightarrow{R} ip \xrightarrow{R^T} d$ denotes domains resolved to the same IP address and is represented by the matrix product RR^T.

5) $d \xrightarrow{Q} c \xrightarrow{N} c \xrightarrow{Q^T} d$ denotes domains queried by clients that belong to the same subset and is represented by the matrix product QNQ^T.
6) $d \xrightarrow{R} ip \xrightarrow{D} ip \xrightarrow{R^T} d$ denotes domains resolved to IPs that belong to the same attacker and is represented by the matrix product RDR^T.

B. PathSim

After the metapaths are computed, they are then run through the PathSim algorithm [1]:

$$M'_{i,j} = \sum_{k=1}^{6} \omega_k \cdot \frac{2M_{k(i,j)}}{M_{k(i,i)} + M_{k(j,j)}} \tag{1}$$

The inputs M_k are the matrix product representing each metapath and the output M' is a matrix that denotes the similarity between the ith and jth nodes. ω_k represent the weight of the kth metapath.

C. Laplacian Weights

While setting ω_k to $\frac{1}{n}$ is an option, some metapaths are more important than others for detecting malicious domains [1]. One method to account for this is called the Laplacian score. The Laplacian Score is especially powerful because it can be implemented in an unsupervised setting (without labeled data). The algorithm [2] is:

1) Construct a nearest neighbor graph G with with n nodes, where the i-th node corresponds to $\mathbf{x}_i$. An edge is created between nodes i and j if they have the same label. In the unsupervised case, the edge is created if $\mathbf{x}_i$ and $\mathbf{x}_j$ are 'close'; when one is one of a set number of nearest neighbors of another.
2) If nodes i and j are connected, let $S_{ij} = e^{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{t}}$, where t is a tuneable constant. Otherwise, $S_{ij} = 0$. This defines the weight matrix S used to model the local structure of the data space.
3) Let $\mathbf{f}_k$ be a column vector of the n samples of the k-th feature, D be a matrix with only the diagonal elements of S, and $\mathbf{1}$ be a column vector of n ones. Then, define $\tilde{\mathbf{f}}_k = \mathbf{f}_k - \frac{\mathbf{f}_k^T D\mathbf{1}}{\mathbf{f}_k^T D\mathbf{1}}\mathbf{1}$.
4) Let $L = D - S$, where L is called the graph Laplacian. Define the Laplacian score of the k-th feature as:

$$\omega_k = \frac{\tilde{\mathbf{f}}_k^T L \tilde{\mathbf{f}}_k}{\tilde{\mathbf{f}}_k^T D \tilde{\mathbf{f}}_k} \tag{2}$$

This is combined with the PathSim algorithm to compute M'. Note that the matrices D and S in this algorithm are not the same as the HIN matrices denoted by those letters.

D. Transductive Classifier

While the similarity matrix M' is useful, it cannot predict whether a domain is benign or malicious. To do this, a cost function must be minimized. This cost function is based on two basic assumptions: smoothness and fitting. Smoothness means that nearby objects should usually belong to the same class and fitting means that the classification should match the known labels [1]. This leads to the following cost function:

$$Q(F) = \frac{1}{2}\left(\sum_{i,j=0}^{n-1} M'_{i,j} \left|\left| \frac{F_i}{\sqrt{D_i i}} - \frac{F_j}{\sqrt{D_j j}} \right|\right|^2 + \mu \sum_{i=0}^{n-1} ||F_i - Y_i||^2\right) \tag{3}$$

$M' \in R^{n \times n}$ is the similarity matrix computed with PathSim, $D \in R^{n \times n}$ is a diagonal matrix where D_{ii} is the sum of the ith row of M', $F \in R^{n \times 2}$ contains each domain's probability of being benign or malicious, $Y \in R^{n \times 2}$ is the labeling information, n is the number of domains in the HIN, and $\mu \in R$ is a tuneable parameter that denotes the relative importance of fitting compared to smoothness [1]. Again, these matrices are not to be confused with HIN matrices that share the same label; at this point all of the HIN matrices are accounted for in M'. F is unknown and must be solved for by finding the F^* that minimizes $Q(F)$. This is done by computing $\frac{dQ}{dF}$, setting it to 0, and solving for F^*. The result is

$$F^* = \beta(I - \alpha S)^{-1} Y \tag{4}$$

where I is the identity matrix, $\alpha = \frac{1}{1+\mu}$, $\beta = \frac{\mu}{1+\mu}$, and $S = D^{-1/2} M' D^{1/2}$ [1]. However, inverting a matrix is very computationally expensive, so an iterative approach is taken. This is done by setting $F(0) = Y$ and iterating $F(t+1) = \alpha S F(T) = \beta Y$ until the change is within a some tolerance [1]. Domain i is labeled malicious if and only if $F_{i,1} - F_{i,0} > \theta$, where $\theta \in [0, 1]$ is a tuneable parameter [1].

IV. Implementation

The metapath calculation, PathSim, Laplacian weights, and Transducitve Classifier will be all be fairly straightforward to implement. The largest challenge will be implementing the algorithms efficiently since n is very large in practice. However, using sparse methods such as the Scipy Python library's compressed sparse row (CSR) format [3] allow for much faster matrix computations by taking advantage of the fact that the matrices are mostly zeroes. Another possible area of investigation is reworking the Transductive Classifier to have D be an $n \times 1$ vector rather than a diagonal matrix and have F and Y be column vectors instead of $n \times 2$ matrices since their rows sum to 1. However, this would have limited effect on the runtime since it would only remove O(n) operations. Additionally, only having a partially labelled dataset may make it complicated to replace the Y matrix.

References

[1] X. Sun, M. Tong, J. Yang, X. Liu, and H. Liu. "HinDom: A Robust Malicious Domain Detection System based on Heterogeneous Information Network with Transductive Classification" RAID 2019, pp. 399–412, Sept. 2019.

[2] Xiaofei He, Deng Cai, and Partha Niyogi. "Laplacian Score for Feature Selection," In Advances in neural information processing systems, pages 507–514, 2006.

[3] The SciPy community. "scipy.sparse.csr_matrix - SciPy v1.6.1 Reference Guide." Feb. 2020. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html. [Accessed: 25-Mar-2021].