Graph based Tensor Recovery For Accurate Internet Anomaly Detection

Kun Xie[1,2], Xiaocan Li[1], Xin Wang[3], Gaogang Xie[4], Jigang Wen[4], Dafang Zhang[1]
[1] College of Computer Science and Electronics Engineering, Hunan University, China
[2] CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, China
[3] Department of Electrical and Computer Engineering, State University of New York at Stony Brook, USA
[4] Institute of Computing Technology, Chinese Academy of Sciences, China

***Abstract*—Detecting anomalous traffic is a crucial task of managing networks. Many anomaly detection algorithms have been proposed recently. However, constrained by their matrix-based traffic data model, existing algorithms often suffer from low detection accuracy. To fully utilize the multi-dimensional information hidden in the traffic data, this paper takes an initiative to investigate the potential and methodologies of performing tensor factorization for more accurate Internet anomaly detection. Only considering the low-rank linearity features hidden in the data, current tensor factorization techniques would result in low anomaly detection accuracy. We propose a novel Graph-based Tensor Recovery model (Graph-TR) to well explore both low rank linearity features as well as the non-linear proximity information hidden in the traffic data for better anomaly detection. We encode the non-linear proximity information of the traffic data by constructing nearest neighbor graphs and incorporate this information into the tensor factorization using the graph Laplacian. Moreover, to facilitate the quick building of neighbor graph, we propose a nearest neighbor searching algorithm with the simple locality-sensitive hashing (LSH). We have conducted extensive experiments using Internet traffic trace data Abilene and GÈANT. Compared with the state of art algorithms on matrix-based anomaly detection and tensor recovery approach, our Graph-TR can achieve significantly lower False Positive Rate and higher True Positive Rate.**

***Index Terms*—Traffic anomaly detection, Tensor Recovery, Graph**

I. Introduction

An anomaly in a data set is defined by Barnett and Lewis as ”an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data” [1]. Anomaly detection aims to identify data that do not conform to the patterns exhibited by the data set . Traffic anomalies, such as flash crowds, denial-of-service attacks, port scans, and the spreading of worms, can have detrimental effects on network services. Detecting and diagnosing these anomalies are critical to both network operators and end users.

Recently, many efforts [2]–[6] have been made to develop various traffic anomaly detection algorithms. They usually model the traffic data as a traffic matrix. As the two-dimensional information is not enough to capture the comprehensive correlations hidden in the traffic data, the accuracy of the anomaly detection is often low.

Instead, we propose to model the traffic monitoring data with a multiway tensor, a higher-order generalization of vectors and matrices. Tensor models have been demonstrated to be able to take full advantage of the multilinear structures to provide better data understanding and information precision. In this work, we will investigate the possibility and methodology of exploiting the correlations in a higher dimensional tensor to more robustly detect the network anomaly.

Our recent experimental study [7], [8] on real traffic traces reveals that normal traffic data can reside in a low-dimensional linear subspace and form a low-rank tensor. The anomalies (outliers) should stay outside this subspace. Therefore, we propose a novel approach to separate the low-rank normal data and outlier data from the noisy traffic data captured, and then detect the anomaly by using the outlier data separated. We exploit tensor factorization to recover the normal data from the traffic data.

Initial efforts on tensor recovery [9]–[11] mainly directly extend either RPCA (Robust Principal Component Analysis) [12] or PCA (Principal Component Analysis) [13] of matrix to the tensor field by unfolding a tensor into matrices, and then utilize the information of different modes in a tensor individually. These approaches are fundamentally still matrix-based and would suffer from the low anomaly detection performance without fully exploiting the tensor pattern and the multilinear information inherent in the data. Only RTD [14] and our TensorDet [15] propose a tensor recovery model to decompose a noisy tensor into low rank and sparse components. Utilizing the multilinear structures hidden in the traffic data, this method is promising to achieve a better anomaly detection performance than current matrix-based detection algorithms.

However, the performance under RTD and TensorDet still suffer for following reason. Normal data recovery directly impacts the outlier data separation thus the anomaly detection performance. In RTD and TensorDet, normal data are recovered from the observed data by only considering the linearity feature (low rank) of the data space while ignoring its possible non-linearity. Low rank implies there exists the linear dependence among data. For the traffic data, besides their linear dependence among the origin, destination, and time domains, there also exist some non-linear proximity

978-1-5386-4128-6/18/$31.00 ©2018 IEEE

information in these three domains. For example, the same network topology should have similar traffic data at some specific time (i.e., the office hours, the break time), thus the traffic data should contain time proximity information.

For more accurate anomaly detection, the normal data should be a low rank tensor that preserves the non-linear proximity information in the original data space. Recent studies on machine learning shows that the manifold learning [16]–[18] is effective in preserving the local geometric (relationships) for the face recognition. With this inspiration, we propose to track the non-linear proximity information of the traffic data with the nearest neighbor graphs, and recover a low-rank tensor that captures the proximities in the origin, destination, and time domains for more accurate anomaly detection. In order to implicitly force these proximities, we propose a novel Graph-based Tensor Recovery model (Graph TR) which formulates the problem by constraining the low rank tensor recovered to be smooth on these neighbor graphs. Moreover, the sparsity of the outlier locations is directly represented by using the L_0-norm constraint instead of the L_1-norm constraint in RTD. Some major designs in Graph TR are:

- Despite the difficulty of handling the rank constraint and the L_0-norm constraint for the tensor rank and set cardinality, we propose a block coordinate descent scheme to solve the tensor recovery problem directly in its original form by iteratively solving two sub problems, a tensor factorization sub-problem and an anomaly detection sub-problem.
- Traffic data contain three types of non-linear proximity information: the time proximity, the origin proximity, and the destination proximity. For more accurate anomaly detection, we construct three nearest neighbor graphs to fully extract these information. Using graph Laplacian, we incorporate the information into low rank tensor factorization. By preserving the graph structure, our algorithm can have more discriminating power on detecting traffic anomaly than current pure low rank tensor factorization solution.
- To facilitate building nearest neighbor graph and alleviate the problem of missing geometric information with current k-nearest neighbors algorithm (KNN) methods, we propose a novel locality sensitive hashing (LSH) based nearest neighbor searching algorithm with a LSH table to reorder and buffer traffic data in a fast and effective way. Thus our searching algorithm can quickly find the nearest neighbors of a traffic data point by placing the data points with closer correlations to close-by positions.
- Using traffic trace data Abilene [19] and GÈANT [20], we compare our Graph TR with the state of art tensor recovery algorithms and matrix-based outlier detection algorithms. Our results demonstrate that Graph TR can achieve significantly higher anomaly detection accuracy with low False Positive Rate and high True Positive Rate.

To the best of our knowledge, this is the first work that demonstrates the capability of applying the tensor factorization and manifold leaning to enable robust tensor data recovery for accurate Internet anomaly detection.

The rest of the paper is organized as follows. We introduce the related work in Section II, and the preliminaries of tensor in Section III. We present the traffic tensor model and the basic tensor recovery problem in Section IV, and describe our graph-based tensor recovery model and the nearest neighbor finding algorithm in Section V and Section VI, respectively. Finally, we evaluate our algorithm performance through extensive experiments in Section VII and conclude the work in Section VIII.

II. Related work

We are not aware of any other work that accurately detects the anomaly based on tensor factorization with the manifold learning to capture both the linear and non-linear features in the traffic data. Following we review some literature work.

The aim of tensor recovery is to recover the low-rank tensor from noisy tensor data with some entries corrupted by outliers. Current literature studies on tensor mainly focus on tensor completion to fill in missing data without considering the data corruption. Recently, some initial efforts are made [9]–[11], [14], [15] to investigate the tensor recovery problem. Among which, RTD [14] and TensorDet [15] exploit the multilinear structure of tensor to decompose a noisy tensor into low rank and sparse components. However, they only consider the linear information while ignoring the non-linear proximity information hidden in the traffic data, which will further reduce the accuracy of traffic anomaly detection.

Recent studies [16]–[18] on manifold learning for face recognition show that when high dimensional data points are random but highly correlated with their close neighbors, data points may lie on (or near) a submanifold and are highly non-linear, and linear methods fail to handle such data. Here, a d-dimensional submanifold of a Euclidean space $\mathbb{R}^M$ is a subset $\mathcal{M}^d \subset \mathbb{R}^M$ which locally looks link a flat d-dimensional Euclidean space. Fig. 1 shows a two-dimensional manifold, embedded in three dimensions in three different ways: a linear embedding (plane) (a), an S-shape (b), and a "Swiss roll" (c).

Fig. 1. Two-dimensional manifolds embedded in three dimensions. (a) Linear embedding, (b) S-shape, (c) Swiss roll

The purpose of (non-linear) manifold learning is to preserve the underling geometric structure and relationship when handling the data. Many manifold learning algorithms have been proposed, such as Locally Linear Embedding (LLE) [21], ISOMAP [16], and Laplacian Eigenmap [22]. All these algorithms use the so-called locally invariant concept [23], i.e., the nearby points remain nearby. It has been shown that learning performance can be significantly enhanced if the

geometrical structure is exploited and the local invariance is considered.

Enlightened by manifold learning, to accurately recover the normal data for more accurate anomaly detection, the recovered normal data should be a low-rank tensor with the non-linear proximities preserved. As graph has been proven to be successful in characterizing pairwise data relationship and manifold exploration, we propose a tensor recovery algorithm with the nearest neighbor graph constructed to encode the non-linear structure hidden in the traffic data. We also propose a novel LSH-based algorithm to quickly find the nearest neighbors to maintain the geometric structure of the traffic data.

III. Preliminaries

The notation used in this paper is described as follows. Scalars are denoted by lowercase letters $(a, b, \cdots)$, vectors are written in boldface lowercase $(\mathbf{a}, \mathbf{b}, \cdots)$, and matrices are represented with boldface capitals $(\mathbf{A}, \mathbf{B}, \cdots)$. Higher-order tensors are written as calligraphic letters $(\mathcal{X}, \mathcal{Y}, \cdots)$. The elements of a tensor are denoted by the symbolic name of the tensor with indexes in subscript. For example, the ith entry of a vector $\mathbf{a}$ is denoted by a_i, element (i, j) of a matrix $\mathbf{A}$ is denoted by a_{ij}, and element (i, j, k) of a third-order tensor $\mathcal{X}$ is denoted by x_{ijk}.

Definition 1. *A* tensor *is a multidimensional array, and is a higher-order generalization of a vector (first-order tensor) and a matrix (second-order tensor). An N-way or Nth-order tensor (denoted as $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$) is an element of the tensor product of N vector spaces, where N is the order of $\mathcal{A}$, also called way or mode.*

The element of $\mathcal{A}$ is denoted by $a_{i_1, i_2, \cdots, i_N}$, $i_n \in \{1, 2, \cdots, I_n\}$ with $1 \leq n \leq N$.

Definition 2. *Slices are two-dimensional sub-arrays, defined by fixing all indexes but two.* Fig.2 shows tensor slices of a 3-way tensor.

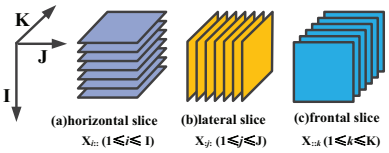


Fig. 2. Tensor slices

Definition 3. *The outer product of two vectors $\mathbf{a} \circ \mathbf{b}$ is the matrix defined by: $(\mathbf{a} \circ \mathbf{b})_{ij} = a_i b_j$.*

Since vectors are first-order tensors, the outer product of three vectors $\mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$ is a tensor given by:

$$(\mathbf{a} \circ \mathbf{b} \circ \mathbf{c})_{ijk} = a_i b_j c_k \tag{1}$$

for all values of the indexes.

Definition 4. *A 3-way tensor $\mathcal{X}$ is a rank one tensor if it can be written as the outer product of three vectors, i.e. $\mathcal{X} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$.*

Definition 5. *The rank of a tensor is the minimal number of rank one tensors, that generate the tensor as their sum, i.e. the smallest R, such that $\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$.*

Definition 6. *The idea of CANDECOMP/PARAFAC (CP) decomposition is to express a tensor as the sum of a finite number of rank one tensors. A 3-way tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ can be expressed as*

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{2}$$

with an entry calculated as

$$x_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr} \tag{3}$$

where $R > 0$, a_{ir}, b_{jr}, c_{kr} are the i-th, j-th, and k-th entry of vectors $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, and $\mathbf{c}_r \in \mathbb{R}^K$, respectively.

Fig.3 illustrates the CP decomposition. By collecting the vectors in the rank one components, we have tensor $\mathcal{X}$'s factor matrices $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_R] \in \mathbb{R}^{I \times R}$, $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_R] \in \mathbb{R}^{J \times R}$, and $\mathbf{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_R] \in \mathbb{R}^{K \times R}$. Using the factor matrices, we can rewrite the CP decomposition as follows.

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!], \tag{4}$$

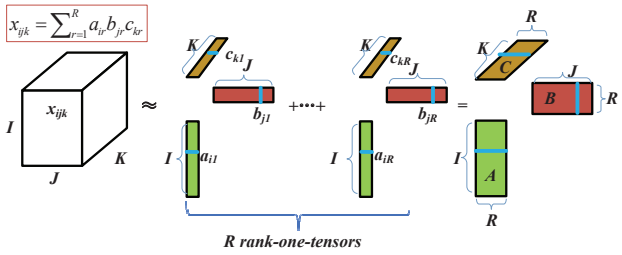


Fig. 3. CP decomposition of 3-way tensor as sum of R outer products (rank one tensors). CP decomposition can be written as a triplet of factor matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, i.e, the r-th column of which contains $\mathbf{a}_r$, $\mathbf{b}_r$, and $\mathbf{c}_r$, respectively. The entry x_{ijk} can be calculated as the sum of the product of the entries of the i-th row of the matrix $\mathbf{A}$, the j-th row of the matrix $\mathbf{B}$, and the k-th row of the matrix $\mathbf{C}$.

IV. Basic Tensor Recovery Problem

For a network consisting of N nodes, the traffic tensor can be formed with a 3-way tensor $\mathcal{X} \in \mathbb{R}^{N \times N \times T}$, corresponding respectively to the origin, destination and the total number of time intervals to consider. The data captured by a traffic tensor tend to be noisy and are subject to outliers and arbitrary corruptions.

For accurate detection of the outliers and corruptions, we propose a tensor recovery model to separate the low-rank normal data and sparse outlier data from the noisy traffic data captured. Our model can be expressed as follows:

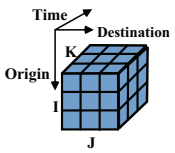


Fig. 4. Traffic tensor

$$\begin{aligned} &\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathcal{E}} \|(\mathcal{X} - \mathcal{E}) - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_F \\ &s.t. \mathcal{X}' = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \\ &rank(\mathcal{X}') \leq R \\ &\|\mathcal{E}\|_0 \leq \varepsilon, \end{aligned} \tag{5}$$

In (5), we decompose a given observation tensor $\mathcal{X}$ into a low-rank tensor $\mathcal{X}'$ (i.e., the normal data) and a sparse tensor $\mathcal{E}$, with a rank constraint ($rank(\mathcal{X}') \leq R$) and a L_0-norm constraint ($\|\mathcal{E}\|_0 \leq \varepsilon$) in their direct form without using some relaxation techniques for more accurate detection of traffic anomaly. With $(\mathcal{X} - \mathcal{E})$, we exclude the outliers $\mathcal{E}$ from the observation tensor. For L_0-norm constraint, we do not need the actual number of outliers, but only use ε to provide an upper limit to prevent too many data items from being classified as outliers. In the experiment part, we will investigate how R impacts the anomaly detection performance.

V. Graph Based Tensor Recovery

It is important to preserve the non-linearity in the low-rank tensor recovered. As recent studies on manifold learning theory demonstrate that the local geometric structure can be effectively modeled through the nearest neighbor graph on a scatter of data points, in this section, we construct nearest neighbor graphs to model the proximity hidden in the traffic data with the local geometric structure. We introduce a novel method called graph Laplacian to regularize the tensor recovery problem by explicitly considering the non-linear proximity structures in traffic data.

A. Nonlinear proximity information

The low rank of the traffic tensor $\mathcal{X}$ [7], [8] implies that there exists linear dependence among the origin, destination, and time. Besides, the network traffic also has some non-linear relationship in these three dimensions. As an example, traffic flows often follow a daily schedule. For a given network topology, traffic data can be similar at some specific time of a day (i.e., the office hours, the break time). Although not having strict periodicity, traffic data do contain the **time proximity information**. In addition, depending on the roles of nodes, their traffic patterns could be different. Nodes in the network can be classified into several groups, such as the normal end users and powerful servers. Nodes from the same class may have a similar Internet access pattern and generate similar traffic load, thus causing the traffic data to have **origin proximity** and **destination proximity**.

To recover the normal data, the problem in (5) only considers the linearity features (i.e., Low rank) hidden in the traffic data while ignoring their possible non-linear relationship. This could compromise the anomaly detection performance. For more accurate detection, we would also like to consider the non-linear proximity in the problem through regularization. As the normal traffic data $\mathcal{X}'$ are determined through $\mathcal{X}' = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$, where variables $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are the factor matrices to look for. To incorporate the non-linear proximity information, the information should be also represented by $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ and embedded into the original problem (5).

B. Relationship between CP decomposition and matrix slice decomposition

To extract the non-linear proximity features, we first investigate the relationship between the tensor CP decomposition and the decomposition of a matrix slice of the tensor.

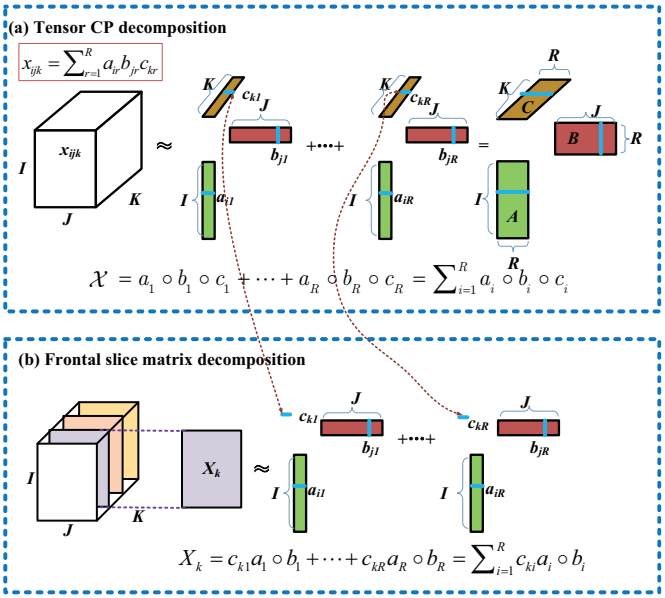


Fig. 5. The relationship between tensor CP decomposition and frontal slice representation.

A frontal slice, a lateral slice, and a horizontal slice of the traffic tensor $\mathcal{X}$ record respectively the data volume of the whole network at a time interval, the data from all nodes to a destination over all time intervals, the data from a origin to all the nodes over all time intervals. Consequently, the time proximity information, the origin proximity information, and the destination proximity information can be extracted using frontal slices, horizontal slices, and lateral slices, respectively.

In Fig.5(a), according to (4), the CP decomposition of a 3-way tensor $\mathcal{X}$ can be written as follows.

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \tag{6}$$

where matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$, $\mathbf{C} \in \mathbb{R}^{K \times R}$ are the factor matrices in the CP decomposition. In Fig.5(b), a frontal slice $\mathbf{X}_{::k}$ can be written as

$$\mathbf{X}_{::k} = c_{k1}\mathbf{a}_1 \circ \mathbf{b}_1 + \cdots + c_{kR}\mathbf{a}_R \circ \mathbf{b}_R = \sum_{i=1}^{R} c_{ki}\mathbf{a}_i \circ \mathbf{b}_i. \tag{7}$$

where $c_{k1}, c_{k2}, \cdots, c_{kR}$ are the entries of the k-th row of the factor matrix $\mathbf{C}$. Similarly, a lateral slice $\mathbf{X}_{:j:}$ and a horizontal slice $\mathbf{X}_{i::}$ can be written in (8) and (9).

$$\mathbf{X}_{:j:} = b_{j1}\mathbf{a}_1 \circ \mathbf{c}_1 + \cdots + b_{kR}\mathbf{a}_R \circ \mathbf{c}_R = \sum_{i=1}^{R} b_{ji}\mathbf{a}_i \circ \mathbf{c}_i. \tag{8}$$

where $b_{j1}, b_{j2}, \cdots, b_{jR}$ are the entries of the j-th row of the factor matrix $\mathbf{B}$.

$$\mathbf{X}_{i::} = a_{i1}\mathbf{b}_1 \circ \mathbf{c}_1 + \cdots + a_{iR}\mathbf{b}_R \circ \mathbf{c}_R = \sum_{j=1}^{R} a_{ij}\mathbf{b}_j \circ \mathbf{c}_j. \tag{9}$$

where $a_{i1}, a_{i2}, \cdots, a_{iR}$ are the entries of the i-th row of the factor matrix $\mathbf{A}$.

Equation (7) shows that each frontal slice $\mathbf{X}_{::k}$ can be expressed as a superposition of R rank-1 matrices $\mathbf{a}_i \circ \mathbf{b}_i$ $(1 \leq i \leq R)$. That is, the traffic data $\mathbf{X}_{::k}$ at a time slot k is

approximated by the linear combination of R rank-1 matrices $\mathbf{a}_i \circ \mathbf{b}_i$, which can thus be called frontal basis matrices. The parameters $c_{k1}, c_{k2}, \cdots, c_{kR}$ are the coordinates of the frontal slice $\mathbf{X}_{::k}$ under this new basis. For abbreviated presentation, given a matrix $\mathbf{Y}$, we denote the i-th row of the matrix as $\mathbf{y_{i:}}$.

Given two frontal slices $\mathbf{X}_{::k_1}$ and $\mathbf{X}_{::k_2}$, if their corresponding traffic data at time slots k_1 and k_2 are close, their new encodings on the frontal basis, $c_{k_1 1}, c_{k_1 2}, \cdots, c_{k_1 R}$ (i.e., $c_{k_1:}$, the k_1-th row of the factor matrix $\mathbf{C}$) and $c_{k_2 1}, c_{k_2 2}, \cdots, c_{k_2 R}$ (i.e., $c_{k_2:}$, the k_2-th row of the factor matrix $\mathbf{C}$) should also be close to each other. Instead of directly using a traffic slice itself, in following sections, we will use this new encoding to facilitate calculating the proximity information hidden in the traffic slices.

C. Incorporating the non-linearity features into the tensor recovery problem

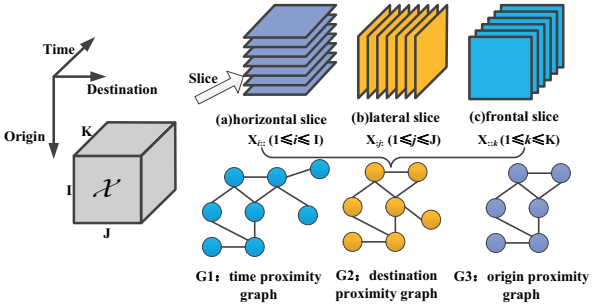


Fig. 6. Neighbor graph to capture the proximity information.

As mentioned in Section V-A, the proximity information on time, origin and destination is hidden in the traffic data, to fully exploit these non-linear proximity features for more accurate tensor data recovery, we build three neighbor graphs corresponding to time, origin and destination respectively, as shown in Fig.6.

Obviously, the frontal slices, horizontal slices, and lateral slices are vertices in the time neighbor graph, the origin neighbor graph, and the destination neighbor graph, respectively. In each graph, an edge is connected between two vertices only if they are nearest neighbors that share some proximity information, thus nearest neighbor finding is an important step to build the neighbor graph. In the next section VI, we will introduce our nearest neighbor finding algorithm based on locality sensitive hashing(LSH).

After finding the nearest neighbors in a graph, edges are connected among vertices, and the weight matrix $\mathbf{F}$ of the graph G can be defined over the edges as

$$f_{ij} = \begin{cases} 1 & if \quad v_i \in N(v_j) \quad and \quad v_j \in N(v_i), \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where v_i, v_j denote the two vertexes in the graph, $N(v_i)$ denotes the set of vertex v_i's nearest neighbors. Only if $v_i \in N(v_j)$ and $v_j \in N(v_i)$ are satisfied, there exists an edge connecting v_i and v_j.

The nearest neighbor graph G and its weight matrix $\mathbf{F}$ characterize the local geometry of proximity information in the traffic data. To retain these information in the tensor factorization, it is reasonable to minimize the following function in the tensor recovery problem

$$\sum_{i,j} \|\mathbf{v_i} - \mathbf{v_j}\|^2 f_{ij} = tr\left(\mathbf{V}^T \mathbf{L} \mathbf{V}\right) \tag{11}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{F}$ is the graph Laplacian matrix and $\mathbf{D}$ is a diagonal matrix whose entries are column sums of the weight matrix $\mathbf{F}$, i.e., $d_{ii} = \sum_j f_{ji}$ and $d_{ij} = 0$ for $i \neq j$. The minimization ensures that the vectors $\mathbf{v_i}$ and $\mathbf{v_j}$ residing on two well connected nodes (i.e., they are nearest neighbors) to have a small distance $\|\mathbf{v_i} - \mathbf{v_j}\|^2$ so that $tr\left(\mathbf{V^T L V}\right)$ is small.

Our goal in this paper is to present a low-rank tensor factorization model that takes into account both linear features and the non-linear proximity features of the traffic data for accurate anomaly detection. To achieve this, we update the tensor recovery problem in (5) as follows:

$$\begin{aligned} &\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathcal{E}} \tfrac{\Upsilon_n}{2}\|(\mathcal{X} - \mathcal{E}) - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_F^2 + \tfrac{\Upsilon_x}{2} O_T + \tfrac{\Upsilon_y}{2} O_O + \tfrac{\Upsilon_z}{2} O_D \\ &s.t. \mathcal{X}' = [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!] \\ &rank(\mathcal{X}') \leq R \\ &\|\mathcal{E}\|_0 \leq \varepsilon, \end{aligned} \tag{12}$$

where O_T, O_O, and O_D represent the constraints corresponding to the time proximity, origin proximity, and destination proximity. They can be expressed as

$$O_T = \sum_{i,j} \|\mathbf{c_{i:}} - \mathbf{c_{j:}}\|^2 f_{ij} = tr\left(\mathbf{C^T L_c C}\right) \tag{13}$$

$$O_O = \sum_{i,j} \|\mathbf{a_{i:}} - \mathbf{a_{j:}}\|^2 f_{ij} = tr\left(\mathbf{A^T L_a A}\right) \tag{14}$$

$$O_D = \sum_{i,j} \|\mathbf{b_{i:}} - \mathbf{b_{j:}}\|^2 f_{ij} = tr\left(\mathbf{B^T L_b B}\right) \tag{15}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are the factor matrices and $\mathbf{L_c}$, $\mathbf{L_a}$, and $\mathbf{L_b}$ are the graph Laplacian matrices of the time graph, the origin graph and the destination graph, respectively.

In (13), $\mathbf{c_{i:}}$ and $\mathbf{c_{j:}}$ denote the encodings of the frontal slices $\mathbf{X}_{::\mathbf{i}}$ and $\mathbf{X}_{::\mathbf{j}}$ under the frontal basis, and $\|\mathbf{c_{i:}} - \mathbf{c_{j:}}\|^2 f_{ij}$ denotes their distance. Similarly, in (14) and (15), $\mathbf{a_{i:}}$ and $\mathbf{a_{j:}}$ denote the encodings of the horizontal slices $\mathbf{X}_{\mathbf{i}::}$ and $\mathbf{X}_{\mathbf{j}::}$ under the horizontal basis, and $\mathbf{b_{i:}}$ and $\mathbf{b_{j:}}$ denote the encodings of the lateral slices $\mathbf{X}_{:\mathbf{i}:}$ and $\mathbf{X}_{:\mathbf{j}:}$ under the lateral basis, respectively. The incorporation of O_T, O_O, and O_D into the objective function ensures that, if data points of two traffic slices are close, their recovered data are also close.

D. Solution to the graph based tensor recovery problem

The problem (12) involves the tensor rank and the L_0-norm (i.e., the cardinality of the outlier set), and is very difficult to solve. As the constraints of $\mathcal{X}'$ and $\mathcal{E}$ are independent, the problem is decomposable. Taking advantage of this property, we propose to adopt the block coordinate descent strategy, and divide the original problem into two sub-problems: a tensor

factorization sub-problem (16) and an anomaly detection sub-problem (17), which can be alternately solved until the solution converges as shown in Algorithm 1.

Algorithm 1 Tensor recovery algorithm

1: **while** not converged **do**
2: solve the tensor factorization sub-problem

$$\begin{aligned} &\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathcal{E}} \tfrac{\Upsilon_n}{2}\|(\mathcal{C}) - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_F^2 + \tfrac{\Upsilon_x}{2}O_T + \tfrac{\Upsilon_y}{2}O_O + \tfrac{\Upsilon_z}{2}O_D \\ &s.t. \mathcal{X}' = [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!] \\ &\mathcal{C} = \mathcal{X} - \mathcal{E} \\ &rank(\mathcal{X}') \leq R \end{aligned} \tag{16}$$

3: solve the anomaly detection sub-problem

$$\begin{aligned} &\mathcal{E} = \arg\min_{\mathcal{E}} \|\mathcal{S} - \mathcal{E}\|_F \\ &s.t. \mathcal{S} = \mathcal{X} - \mathcal{X}' \\ &\|\mathcal{E}\|_0 \leq \varepsilon \end{aligned} \tag{17}$$

4: **end while**

In the tensor factorization subproblem (Eq.(16)), we first fix the current estimate of outliers $\mathcal{E}$ and exclude them from $\mathcal{X}$ to obtain the "clean" data $\mathcal{C}$, and then approximate $\mathcal{C}$ using $\mathcal{X}'$. In the anomaly detection sub-problem (Eq.(17)), we update the outliers $\mathcal{E}$ based on the error $\mathcal{S} = \mathcal{X} - \mathcal{X}'$.

We solve the tensor factorization sub-problem in (16) based on the gradient decent scheme. For $O = \frac{\Upsilon_n}{2}\|(\mathcal{C}) - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_F^2 + \frac{\Upsilon_x}{2}O_T + \frac{\Upsilon_y}{2}O_O + \frac{\Upsilon_z}{2}O_D$, the gradients corresponding to different factor matrices can be denoted by $\frac{\partial O}{\partial \mathbf{A}}$, $\frac{\partial O}{\partial \mathbf{B}}$, $\frac{\partial O}{\partial \mathbf{C}}$. After obtaining the gradient, the factor matrices are updated iteratively by the following rules $\mathbf{A} \leftarrow \mathbf{A} - \eta\frac{\partial O}{\partial \mathbf{A}}$, $\mathbf{B} \leftarrow \mathbf{B} - \eta\frac{\partial O}{\partial \mathbf{B}}$, $\mathbf{C} \leftarrow \mathbf{C} - \eta\frac{\partial O}{\partial \mathbf{C}}$, where η is the learning rate.

The sub-problem described in (17) can easily be solved as:

$$e_{i,j,k} = \begin{cases} s_{i,j,k} & \beta_{i,j,k} > \beta(\varepsilon) \\ 0 & otherwise \end{cases} \tag{18}$$

where $\beta_{i,j,k} = (s_{i,j,k})^2$ and $\beta(\varepsilon)$ is the $\varepsilon - th$ largest value in $\beta_{i,j,k}$. Generally, in each round, largest errors are considered outliers and are put into $\mathcal{E}$ to be excluded from the low-rank fitting in the next round.

VI. Nearest neighbor searching based on Locality-Sensitive Hashing

k-nearest neighbors algorithm (KNN) is usually adopted in manifold learning theory to find the nearest k neighbors for the local geometry preservation. However, this method faces the problem of less geometrically intuitive, as it may return the nearest k points regardless if they are the neighbors of the point of interest. Moreover, for a graph consisting of n vertexes, KNN requires a high time complexity of $O(n^2)$. Instead, we propose a nearest neighbor finding algorithm based on Locality-Sensitive Hashing (LSH). In this section, we first introduce the LSH function, then present our solution to find the nearest neighbors.

Although in our design, there are three nearest neighbor graphs, we only take time proximity graph (i.e., frontal slices are the vertexes in the graph) as an example to illustrate how our algorithm finds the nearest neighbors.

A. Building LSH table to re-order the frontal slices

According to [24], the LSH function family is defined as follows.

Definition 1. ***(LSH Function Family) [24]:*** *$\mathbb{H} = \{g : V \to U\}$ is called (R, cR, P_1, P_2) - sensitive for any $p, q \in V$*

- *If $||p,q||_s \leq R$ then $\Pr_{\mathbb{H}}[g(p) = g(q)] \geq P_1$.*
- *If $||p,q||_s \geq cR$ then $\Pr_{\mathbb{H}}[g(p) = g(q)] \leq P_2$.*

where $\|p,q\|_s$ is the distance of elements p and q, V is the domain of elements. In the LSH, $c > 1$ and $P_1 > P_2$. To find the nearest neighbors of each frontal slice, frontal slices are the elements that need to be reordered according to their distances.

In Section V-B, we have shown that given two frontal slices $\mathbf{X}_{::k_1}$ and $\mathbf{X}_{::k_2}$, if their data points are close, their new encodings on the frontal basis, $c_{k_1:}$ and $c_{k_2:}$ shall also be close to each other. Therefore, to group similar frontal slices and facilitate the nearest neighbor searching, we use the row vector of the factor matric $\mathbf{C}$ as the indication vector and apply the LSH to the vector to reorder the frontal slices. Specially, given a frontal slice $\mathbf{X}_{::k}$ with its indication vector $c_{k:} \in \mathbb{R}^R (1 \leq k \leq K)$, we define the following LSH hash function $h_{\vec{a},b} : \mathbb{R}^R \to \mathbb{R}$ to map the frontal slice $\mathbf{X}_{::k}$ into a single bin with the bin index equal to:

$$h_{\vec{a},b}(c_{k:}) = \left\lfloor \frac{\vec{a}^T c_{k:} + b}{W} \right\rfloor, \tag{19}$$

and the offset in the bin expressed as

$$off_{\vec{a},b}(c_{k:}) = (\vec{a}^T c_{k:} + b) \bmod (W) \tag{20}$$

where $\vec{a}$ is a R-dimensional random vector with each component chosen independently from a Gaussian distribution $\mathcal{N}(0,1)$, W is the width of a bin, and b is a real number randomly selected from the interval $[0, W)$.

When the hashed values of frontal slices are the same, i.e., there is a collision in hashing, the pairs are mapped to the same bin. Given two frontal slice pairs $\mathbf{X}_{::k_1}$ and $\mathbf{X}_{::k_2}$, we analyze the properties of the hash function through calculating the probability of a collision, i.e., $Pr_{\vec{a},b}[h_{\vec{a},b}(c_{k_1:}) = h_{\vec{a},b}(c_{k_2:})]$.

According to [24], Gaussian distribution $\mathcal{N}(0,1)$ is a 2-stable distribution. Because the components in $\vec{a}$ are chosen following the Gaussian distribution, $\vec{a}^T\vec{q} - \vec{a}^T\vec{p}$ is distributed as dZ where $d = \|\vec{p} - \vec{q}\|_2$ is the distance between pairs $\vec{p}$ and $\vec{q}$ and $Z \sim \mathcal{N}(0,1)$. Since b is a real number randomly selected from the interval $[0, W)$, it is easy to obtain that

$$Pr_{\vec{a},b}[h_{\vec{a},b}(\vec{p}) = h_{\vec{a},b}(\vec{q})] = \int_{t=0}^{W} \tfrac{1}{d} f(\tfrac{t}{d}) \left(1 - \tfrac{t}{W}\right) dt \tag{21}$$

where $f(t)$ is the probability density function of 2-stable distribution, that is, $f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$.

The collision probability (21) depends only on the distance d and is monotonically decreasing in d. Therefore, our LSH function in Eq(20) has a good property, that is, it can map

similar frontal slice pairs with short distance into the same bin.

To buffer similar frontal slice pairs into the same bucket while reducing the probability of hashing uncorrelated vertexes (i.e., frontal slices) to the same bin to create the collision, instead of using a single hash function, we compute the hash table index of a frontal slice as the average of n LSH functions. Given a frontal slice with its vector $c_{k:}$, the hash table index of this slice is denoted by $H(c_{k:})$ and calculated as

$$H(c_{k:}) = \left\lfloor \frac{\sum_{i=1}^{n} h_{\vec{a}_i, b_i}(c_{k:})}{n} \right\rfloor \tag{22}$$

where $h_{\vec{a}_i, b_i}(c_{k:})$ is the mapping address of the frontal slice calculated from each individual LSH hash function. The offset of this frontal slice in the bucket is

$$Off(c_{k:}) = \left(\left(\sum_{i}^{n} \vec{a}_i c_{k:} + b_i \right) / n \right) \bmod (W) \tag{23}$$

An index calculated from (22) corresponds a bucket in the LSH hash table. In this paper, we set $n = 10$.

B. Finding the nearest neighbor based on LSH table

Only requiring the hash calculations, our LSH hash table reorders and buffers frontal slices in a fast and effective way. Moreover, it provides a new indexing method for nearest neighbor query by placing frontal slices with closer correlations in close-by positions.

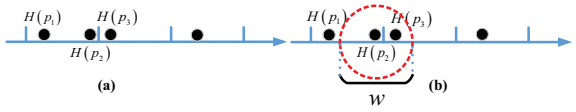


Fig. 7. Illustration of finding the nearest neighbor based on LSH table.

Facilitated by our LSH hash table, a straightforward way to achieve such goal is: given a frontal slice, apply (22) to its indicate vector to locate the corresponding bucket that stores this frontal slice, and return all the frontal slices in the bucket as the required neighbor set.

However, such a straightforward way has an accuracy problem. In Fig.(7), W is the width of hash bucket and the black circles denote the frontal slice data points that are mapped to the table. In the table, p_1 and p_2 are mapped into the same hash bucket while p_3 is mapped into another hash bucket. Under the straightforward way, to query p_2's nearest neighbor, p_1 is returned. However, p_3 is closer to p_2 than p_1.

Instead of this brute force retrieving, to search for the nearest neighbors, we first apply (22) to its indicate vector to identify the target bucket in the table, then among its target bucket and the adjacent buckets (including left adjacent bucket and the right adjacent bucket), return the slices whose distance to the query slice is less than $W/2$. The distance in the bucket can be very easily calculated using the Offset of slices. In Fig.7(b), among the adjacent buckets of p_2, we can easily find that p_3 is the nearest neighbors of p_2.

Compared with KNN, our LSH-based nearest neighbor searching is more geometrically intuitive, with data points selected as the neighbors if their distance is less than $W/2$ in the table. Obviously, the parameter W impacts the number of neighbors for a given slice. As these neighbors are utilized to represent the proximity information around the slice, W further impacts on the accuracy of using neighbor graph to represent local geometric information in the traffic data. In the experiment part, we will vary W and select the appropriate one as the parameter setting.

VII. Performance Evaluations

To evaluate the performance of our scheme, we synthetically generate anomalies by adding data outliers into the public traffic traces Abilene [19] and GÈANT [20] following [13], [25], [26]. As these two traces record the volume of traffic flows between all source and destination pairs, they allow us to form a network-wide traffic tensor.

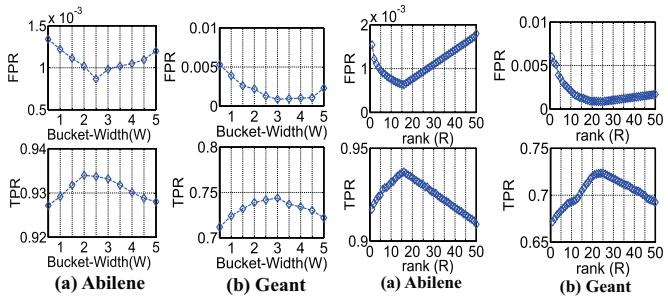
We denote the raw trace data as $\mathcal{X} \in R^{I \times J \times K}$. Given $x_{i,j,k}$, for more efficient data processing, we apply $x_{i,j,k} = \frac{x_{i,j,k} - \min_{u,v,w}\{x_{u,v,w}\}}{\max_{u,v,w}\{x_{u,v,w}\} - \min_{u,v,w}\{x_{u,v,w}\}}$ to normalize the data value within the range [0,1], where $\max_{u,v,w}\{x_{u,v,w}\}$ and $\min_{u,v,w}\{x_{u,v,w}\}$ are the maximum value and minimum value of all the traffic data, respectively.

To simulate anomalies that do not have fixed locations, we randomly select $|\Omega| = \gamma \times (I \times J \times K)$ outlier locations. We set the default value of the outlier ratio γ=0.1. Following [27], we adopt ***Gaussian distribution*** to generate the outlier data values following $\mathcal{N}(\mu, \sigma^2)$, with the defaults values of the mean μ and the variance σ^2 set to 0 and 0.1, respectively. For each experiment setting, we run the experiments ten times with the random seeds and get the average of the results.

We use following two metrics to evaluate the performance of the proposed Graph TR. **False Positive Rate (FPR):** It measures the proportion of non-outliers that are wrongly identified as outliers. **True Positive Rate (TPR):** It measures the proportion of outliers that are correctly identified. Smaller False Positive Rate and higher True Positive Rate mean better detection performance.

We implement six schemes for performance comparison. Based on our traffic tensor model, three tensor-based anomaly detection schemes are implemented: our proposed Graph TR, TensorRPCA proposed in [9], and RTD proposed in [14]. Current traffic data analysis is usually based on a traffic matrix model with its row representing the origin and destination (OD) pair and the column representing the time interval. Accordingly, 3 other matrix-based anomaly detection schemes are implemented: MatrixDR based on the direct robust matrix factorization [28], MatrixPCA using a PCA-based algorithm [13], and MatrixRPCA based on the robust PCA [12].

To fairly compare these algorithms, we adopt the same anomaly detection principle: among all the candidate outlier data, return α data points with the largest absolute values, where α is the number of outliers injected.



(a) Impact of W (b) Impact of R

Fig. 8. Parameter Settings.

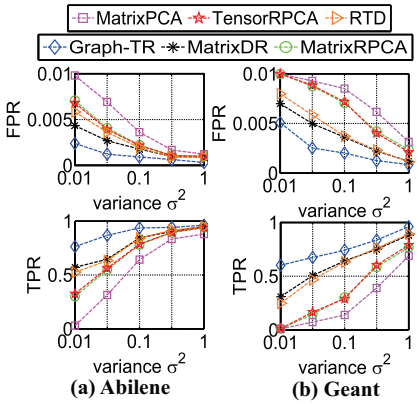


Fig. 9. Variance σ^2

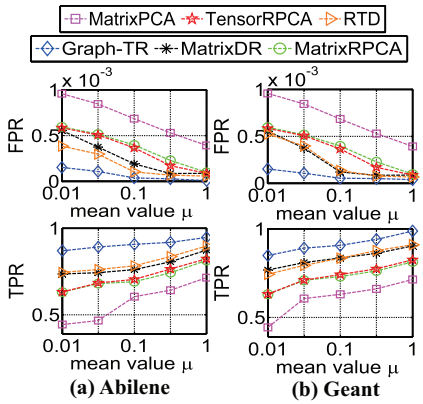


Fig. 10. Mean value μ

A. Impact of the parameters

1) Impact of W

As discussed in Section VI-B, the parameter W impacts the number of neighbors utilized to preserve the proximity information when recovering the normal data. From Fig.8(a), when W increases and more neighbor nodes are utilized to preserve the proximity information, the anomaly detection accuracy increases initially with smaller false positive rate and larger True Positive Rate as more neighbor points can more accurately represent the intrinsic geometry. The detection accuracy reduces after $W = 2.5$ (Abilene) and $W = 3$ (GÈANT) as too large number of neighbor points (with their average distance to the data point of interest increased) reduces the accuracy of representing the local geometry hidden in the data space. According to the results, we set $W = 2.5$ (Abilene) and $W = 3$ (GÈANT) in our rest experiments.

2) Impact of R

In the problem formulation in (12), the rank constraint R can be set to preserve certain amount of the tensor data variability to capture the main features of the normal data. To investigate how the R setting impacts the outlier detection performance, we vary R and draw the anomaly detection performance in Fig.8(b). We can see the initial raising of R increases the anomaly detection accuracy, as CP decomposition cannot capture the full structure of the traffic tensor with an under-estimation of the rank. After R reaches 15(Abilene) and 23(GÈANT), a further increase of R makes the anomaly detection accuracy worse as it causes an overflow problem. Therefore, we set rank $R = 15$ (Abilene) and $R = 23$ (GÈANT) in our rest experiments.

B. Accuracy comparison

To compare the performance of different anomaly detection algorithms, with other parameters fixed, we vary the variance σ^2 and the mean value μ.

From Fig.9, Fig.10, among all the algorithms compared, our Graph TR achieves the best performance. It achieves the lowest False Positive Rate and Highest True Positive Rate, under all the experiment scenarios using different traffic traces (Abilene and GÈANT). These results also demonstrate that Graph TR is a robust anomaly detection technique that can fully utilize the traffic tensor's low-rank linear features as well as non-linear proximity features to more accurately detect the anomaly.

Compared with the tensor-based anomaly detection algorithms TensorRPCA and RTD, our Graph TR achieves much better overall performance. Designed based on unfolding matrices and using the trace norm to relax the low-rank feature of the unfolding matrices, TensorRPCA is fundamentally a matrix-based approach and cannot fully exploit the tensor pattern with the multilinear information to better detect the anomaly. As a result, compared with MatrixDR, the detection performance under TensorRPCA is even worse, as MatrixDR can model and solve the anomaly detection problem directly using low rank feature instead of using trace norm to relax the low rank feature like TensorRPCA.

Besides the low rank linear information adopted in RTD, our Graph TR also utilizes non-linear proximity information to detect the anomaly. Moreover, our Graph TR models the sparsity outlier in its direct form with the L_0 norm instead of its relaxation L_1 norm in RTD. As a result of above techniques, although both RTD and our Graph TR are designed based on CP decomposition, our Graph TR achieves much better performance.

As shown in Fig.9 and Fig.10, with the increase of variance σ^2 and mean μ of the outliers, the True Positive Rate increases

while the False Positive Rate decreases for all algorithms implemented. Obviously, when the variance and mean of outliers are smaller, synthesized outlier data have closer and smaller values, and are more difficult to be differentiated from the normal data.

VIII. Conclusion

Besides the low rank linear feature, the traffic data also have non-linear proximity information. To fully exploit these data features for more accurate anomaly detection, we propose a novel graph-based tensor recovery model. Particularly, to incorporate non-linear proximity information into the tensor factorization, we propose several techniques. Firstly, we propose a method to encode the non-linear proximity information of the traffic data by constructing nearest neighbor graphs and using the graph Laplacian to embed this information into the tensor factorization. Second, to facilitate quick graph neighbor building, we propose a novel locality sensitive hashing (LSH) based algorithm for efficient nearest neighbor searching. We have done extensive experiments using the Internet traffic trace data Abilene and GÈANT. Compared with the state of art algorithms on matrix-based anomaly detection and tensor recovery approach, our Graph TR achieves significantly lower False Positive Rate and higher True Positive Rate in all experiment scenarios. Our scheme is flexible to apply in various systems to detect the false and anomaly data. Not limited to the traffic data, more data sets from different application scenarios [29]–[32] will been used for evaluation in our future work.

Acknowledgment

The work is supported by the National Natural Science Foundation of China under Grant Nos.61572184, 61725206, 61472130, 61472131, and 61772191, Hunan Provincial Natural Science Foundation of China under Grant No.2017JJ1010, Science and Technology Key Projects of Hunan Province under Grant No.2015TP1004, U.S. NSF ECCS 1408247, CNS 1526843, and ECCS 1731238, and the open project funding (CASNDST201704) of CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences.

References

[1] V. Barnett and T. Lewis, "Outliers in statistical data," 1994.
[2] L. Huang, X. Nguyen, M. Garofalakis, J. M. Hellerstein, M. Jordan, A. D. Joseph, N. Taft *et al.*, "Communication-efficient online detection of network-wide anomalies," in *INFOCOM*. IEEE, 2007.
[3] D. Brauckhoff, K. Salamatian, and M. May, "Applying pca for traffic anomaly detection: Problems and solutions," in *INFOCOM 2009, IEEE*. IEEE, 2009.
[4] Y. Liu, L. Zhang, and Y. Guan, "Sketch-based streaming pca algorithm for network-wide traffic anomaly detection," in *ICDCS*. IEEE, 2010.
[5] G. Xie, K. Xie, J. Huang, X. Wang, Y. Chen, and J. Wen, "Fast low-rank matrix approximation with locality sensitive hashing for quick anomaly detection," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017, pp. 1–9.
[6] K. Xie, X. Ning, X. Wang, D. Xie, J. Cao, G. Xie, and J. Wen, "Recover corrupted data in sensor networks: A matrix completion solution," *IEEE Transactions on Mobile Computing*, vol. 16, no. 5, pp. 1434–1448, 2017.
[7] X. Kun, W. Lele, W. Xin, X. Gaogang, W. Jigang, and Z. Guangxing, "Accurate recovery of internet trafficdata: A tensor completion approach," in *IEEE INFOCOM*, 2016.
[8] X. Kun, P. Can, W. Xin, X. Gaogang, and W. Jigang, "Accurate recovery of internet traffic data under dynamic measurements," in *IEEE INFOCOM*, 2017.
[9] D. Goldfarb and Z. Qin, "Robust low-rank tensor recovery: Models and algorithms," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 1, pp. 225–253, 2014.
[10] H. Tan, J. Feng, G. Feng, W. Wang, and Y.-J. Zhang, "Traffic volume data outlier recovery via tensor model," *Mathematical Problems in Engineering*, vol. 2013, 2013.
[11] J. Li, G. Han, J. Wen, and X. Gao, "Robust tensor subspace learning for anomaly detection," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 89–98, 2011.
[12] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the ACM (JACM)*, vol. 58, no. 3, p. 11, 2011.
[13] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4. ACM, 2004, pp. 219–230.
[14] A. Anandkumar, P. Jain, Y. Shi, and U. N. Niranjan, "Tensor vs matrix methods: Robust tensor decomposition under block sparse perturbations," *CoRR*, vol. abs/1510.04747, 2015. [Online]. Available: http://arxiv.org/abs/1510.04747
[15] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, J. Cao, and D. Zhang, "Fast tensor factorization for accurate internet anomaly detection," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3794 – 3807, 2017.
[16] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
[17] D. Lunga, S. Prasad, M. M. Crawford, and O. Ersoy, "Manifold-learning-based feature extraction for classification of hyperspectral data: A review of advances in manifold learning," *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 55–66, 2014.
[18] B. Raducanu and F. Dornaika, "A supervised non-linear dimensionality reduction approach for manifold learning," *Pattern Recognition*, vol. 45, no. 6, pp. 2432–2444, 2012.
[19] "The abilene observatory data collections. http://abilene. internet2.edu /observatory/data-collections.html."
[20] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 83–86, 2006.
[21] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[22] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, 2002, pp. 585–591.
[23] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *CVPR*, vol. 2. IEEE, 2006.
[24] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *SoCG*. ACM, 2004.
[25] R. A. Maxion and K. M. Tan, "Benchmarking anomaly-based detection systems," in *DSN*. IEEE, 2000.
[26] A. Soule, K. Salamatian, and N. Taft, "Combining filtering and statistical methods for anomaly detection," in *IMC*. USENIX Association, 2005.
[27] J. Jiang and S. Papavassiliou, "Detecting network attacks in the internet via statistical network traffic normality prediction," *Journal of Network and Systems Management*, vol. 12, no. 1, pp. 51–72, 2004.
[28] L. Xiong, X. Chen, and J. Schneider, "Direct robust matrix factorizatoin for anomaly detection," in *ICDM*. IEEE, 2011.
[29] K. Xie, L. Wang, X. Wang, G. Xie, J. Wen *et al.*, "Low cost and high accuracy data gathering in wsns with matrix completion," *IEEE Transactions on Mobile Computing*, 2017.
[30] Z. Tang, A. Liu, Z. Li, Y.-j. Choi, H. Sekiya, and J. Li, "A trust-based model for security cooperating in vehicular cloud computing," *Mobile Information Systems*, vol. 2016, 2016.
[31] X. Liu, X. Xie, K. Li, B. Xiao, J. Wu, H. Qi, and D. Lu, "Fast tracking the population of key tags in large-scale anonymous rfid systems," *IEEE/ACM Transactions on Networking (TON)*, vol. 25, no. 1, pp. 278–291, 2017.
[32] X. Liu, K. Li, A. X. Liu, S. Guo, M. Shahzad, A. L. Wang, and J. Wu, "Multi-category rfid estimation," *IEEE/ACM transactions on networking*, vol. 25, no. 1, pp. 264–277, 2017.