The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Learning Multi-Way Relations via Tensor Decomposition with Neural Networks

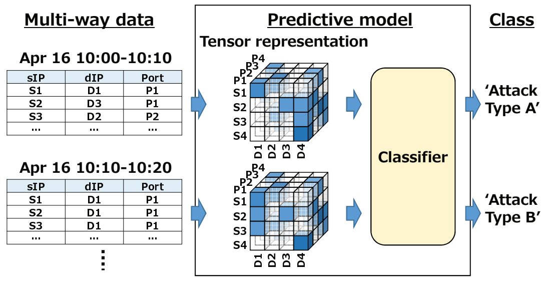
**Koji Maruhashi, Masaru Todoriki, Takuya Ohwa, Keisuke Goto,
Yu Hasegawa, Hiroya Inakoshi, Hirokazu Anai**
Fujitsu Laboratories Ltd.
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki, Kanagawa, Japan
{maruhashi.koji, todoriki.masaru, takuyaohwa, goto.keisuke, latente, inakoshi.hiroya, anai}@jp.fujitsu.com

Abstract

How can we classify multi-way data such as network traffic logs with multi-way relations between source IPs, destination IPs, and ports? Multi-way data can be represented as a tensor, and there have been several studies on classification of tensors to date. One critical issue in the classification of multi-way relations is how to extract important features for classification when objects in different multi-way data, i.e., in different tensors, are not necessarily in correspondence. In such situations, we aim to extract features that do not depend on how we allocate indices to an object such as a specific source IP; we are interested in only the structures of the multi-way relations. However, this issue has not been considered in previous studies on classification of multi-way data. We propose a novel method which can learn and classify multi-way data using neural networks. Our method leverages a novel type of tensor decomposition that utilizes a *target core tensor* expressing the important features whose indices are independent of those of the multi-way data. The target core tensor guides the tensor decomposition into more effective results and is optimized in a supervised manner. Our experiments on three different domains show that our method is highly accurate, especially on higher order data. It also enables us to interpret the classification results along with the matrices calculated with the novel tensor decomposition.

Introduction

Given a set of multi-way data, how can we classify such data based on the structures of the multi-way relations? Some typical examples are network traffic logs with relations between source IPs (sIPs), destination IPs (dIPs), and ports or transactions of online banking with relations between senders, receivers, and branches. The major problem is obtaining a predictive model to classify logs and transactions, i.e., multi-way data, according to whether they have been subject to attacks and/or frauds (Fig. 1). Multi-way data are represented as a tensor, and several studies on the classification of tensors have been conducted such as image classification (Tao et al. 2007; He et al. 2014). They uniquely determined the allocation of indices, i.e., allocation of indices in the tensor to a specific pixel, thanks to an implicit spatial order of the pixels in images, i.e., from top-left to bottom-right.



Apr 16 10:00-10:10

sIP	dIP	Port
S1	D1	P1
S2	D3	P1
S3	D2	P2
...	...	...

Apr 16 10:10-10:20

sIP	dIP	Port
S1	D1	P1
S2	D1	P1
S3	D1	P1
...	...	...

Figure 1: Classification of multi-way data. A tensor representing a multi-way data is classified by a classifier.

However, these methods cannot be applied to situations in which the objects in two tensors, such as sIPs, are not in correspondence; these methods assume each tensor shares the common indices of objects with each other. Our aim is to extract structures of multi-way relations that are both important for classification and that are independent of the indices of objects. Let us illustrate this problem using an example. A typical pattern of network attack is as follows: scan many ports with a fixed sIP and dIP, and when finding an appropriate port, conduct attacks by accessing to many dIPs with the port and a sIP. It does not matter what IPs/Ports are actually used; in fact, different IPs, ports, and accounts are typically used in every attacks and frauds in many cases, so as not to be easily recognized. We want to extract abstract patterns of this kind of multi-relational connectivity. That is, classification should be done based on the structure of multi-way relations, not based on IPs, ports, and accounts themselves. Therefore, we want to extract features that do not depend on how we allocate indices to an object such as a specific sIP. In fact, this is a common issue in the classification of graphs.

In addition, interpretability as well as prediction performance is becoming more important nowadays with the widening of machine-learning applications (Rudin 2014; Lipton 2016). We desire not only to achieve high predictive performance, but also to know the reason for the classification results in order to make decisions in a real-world situation or gain more understandings of the target phenomenon. For example, security engineers are required not only to alarm the occurrence of cyber-attacks based on intrusion detection system (IDS) logs, but also to spot corresponding logs in order to investigate them.

To the best of our knowledge, this is the first study that provides a method of learning and classifying multi-way data by taking into account the structure of multi-way relations independent of the indices of the input tensor while retaining the ability to interpret the prediction results. Our intuition is that we can extract such structure as a *core tensor* calculated by tensor decomposition (Kolda and Bader 2009) which decomposes a tensor into a core tensor and *factor matrices*, in which the indices of the input tensor only affect the factor matrices. However, a core tensor calculated with conventional tensor decompositions only expresses structures learnt in an unsupervised manner, and they are not necessarily the *important* structures for the classification of data. To tackle this problem, we propose *DeepTensor* leveraged by *Structure Restricted Tensor Decomposition (SRTD)*, a novel type of tensor decomposition. SRTD uses a tensor called the *target core tensor*, which contains the aggregated structures that are important for classification and guides the tensor decomposition into more effective results. We also introduce the *extended backpropagation (EBP)* algorithm, which learns the optimal target core tensor along with the parameters of the neural network. Moreover, once we know the indices of the core tensor important for the classification results, we can interpret them in terms of the indices of the input tensor by using the factor matrices, which can connect between the indices of the input tensor and those of the core tensor. We report on the experimental results on three different domains: intrusion detection, peer-to-peer (P2P) lending, and bioinformatics. We empirically show that our method is highly accurate, especially on higher order data, while enabling us to interpret the classification results.

Related Work

Several studies on the classification of tensors have been reported (Tao et al. 2007; He et al. 2014; Zhou, Li, and Zhu 2013; Imaizumi and Hayashi 2016). They assumed that the important features for classification share similar indices of objects among all multi-way data; however, this assumption is not satisfied in the problem we tackle in this paper. Our idea is to use tensor decomposition to extract features that are independent of the indices. We focus on Tucker decomposition (Kolda and Bader 2009; Acar and Yener 2009), a well-known tensor decomposition, which is most often calculated using the alternating least squares (ALS) method (Lathauwer, Moor, and Vandewalle 2000). However, Tucker decomposition is not unique, that is, we can modify the core tensor without affecting the fit so long as we apply the inverse modification to the factor matrices (Kolda and Bader 2009). This is a serious problem when we need to classify tensors by using tensor decomposition. Several solutions using a set of orthogonal rotations (Kiers 1998) or a Jacobi-type algorithm (Martin and Loan 2008) have been proposed. Unlike these unsupervised approaches, our approach constructs core tensors to maximize the classification accuracy in a supervised manner.

One possible way to handle multi-way data is to represent it as a graph, although it often involves loss of original information contained in the multi-way relations. Several graph kernels have been proposed for graph classification. They count the co-occurrence of walks (Kashima, Tsuda, and Inokuchi 2003), paths (Borgwardt and Kriegel 2005), subgraphs called *graphlets* (Shervashidze et al. 2009), or subtrees (Shervashidze et al. 2011) in two graphs. Methods for graph classification based on the idea of deep neural networks have been proposed. Deep Graph Kernels (Yanardag and Vishwanathan 2015) treat the list of sub-structures used in graph kernels as a sentence and learn latent representations of sub-structures. Another direction that has been eagerly explored recently is graph convolutional neural networks. The method proposed by Niepert et al. (Niepert, Ahmed, and Kutzkov 2016) first selects node sequence for which receptive fields are constructed then assembles a neighborhood graph for each node, which is finally normalized to construct a fixed size receptive field. Accordingly, we are likely to face difficulties in determining nodes to be included in the receptive fields when there are hubs connected to a large number of nodes. The method based on spectral graph theory (Defferrard, Bresson, and Vandergheynst 2016) makes considerable assumptions about the target graph and it is not trivial to extend it to such graphs that have labels for edges. In the first place, none of these graph classification methods directly handle the complicated structures of multi-way relations, in contrast to our method that directly handles them. Moreover, it is difficult to infer the reason for their classification results because these methods convert graph data by using nonlinear modification.

Prediction interpretation and justification is one of the most important topics in recent machine learning research (Rudin 2014; Lipton 2016). One effective approach is to produce *interpretable models*. Lake et al. learned a generative model of linguistic character images from sparse data (Lake, Salakhutdinov, and Tenenbaum 2015). Si et al. learned compositional models of objects in images (Si and Zhu 2013). Ribeiro et al. introduced a model-agnostic approach that provides explanation to blackbox models by learning interpretable models based on the *interpretable representations* (Ribeiro, Singh, and Guestrin 2016). We show that the approach of Ribeiro et al. can be applied to our method by regarding the core tensors as the interpretable representations, and we can interpret the prediction results in terms of the input tensor thanks to the factor matrices.

Preliminaries

Notations

We refer to the dimensionality of a tensor and multi-way data as *order*, and to each dimension as a *mode*. Scalars are denoted by lowercase letters (a), vectors by boldface lowercase letters ($\mathbf{a}$), matrices by boldface capital letters ($\mathbf{A}$), and tensors by boldface Euler script letters ($\mathcal{A}$). A transverse matrix of $\mathbf{A}$ is $\mathbf{A}^T$. $\mathbf{I}$ is an identity matrix, and $\mathbf{1}$ is a square matrix in which all values are one. Indices typically range from 1 to their capitalized version, *e.g.*, $i = 1, \ldots, I$. $|\mathcal{A}|$ is the number of elements of $\mathcal{A}$, and $\|\mathcal{A}\|_2$ is the square root of the sum of the squared values in $\mathcal{A}$. The inner product is denoted by $\langle \cdot, \cdot \rangle$. The element-wise multiplication is denoted by $*$, and the element-wise division is denoted by $\oslash$. The mode-k matricization of $\mathcal{A}$ of size $I_1 \times \ldots \times I_K$ creates $\mathbf{A}_{(k)}$ of size $I_1 \ldots I_{k-1} I_{k+1} \ldots I_K \times I_k$. If the $(i_1, \ldots, i_K)$-th element of $\mathcal{A}$ is one otherwise zero, $\mathbf{A}_{(k)}$ is $\mathbf{a}_1 \otimes \cdots \otimes \mathbf{a}_{k-1} \otimes \mathbf{a}_k^T \otimes \mathbf{a}_{k+1} \otimes \cdots \otimes \mathbf{a}_K$ by using a Kronecker product $\otimes$, where the $i_{k'}$-th element of $\mathbf{a}_{k'}$ is one otherwise zero. A k-mode product of $\mathcal{A}$ with $\mathbf{W}$ of size $I_k \times I'_k$ is $\mathcal{A}' = \mathcal{A} \times_k \mathbf{W}$ of size $I_1 \times \ldots \times I_{k-1} \times I'_k \times I_{k+1} \times \ldots \times I_K$, where $\mathbf{A}'_{(k)} = \mathbf{A}_{(k)} \mathbf{W}$. In case $\mathbf{W}$ is a vector $\mathbf{w}$ of length I_k, we remove the mode and resize $\mathcal{A}'$ to $I_1 \times \ldots \times I_{k-1} \times I_{k+1} \times \ldots \times I_K$. We denote $\mathcal{A} \times_1 \mathbf{W}_1 \ldots \times_K \mathbf{W}_K$ as $\mathcal{A} \prod_k \times_k \mathbf{W}_k$. A Khatri-Rao product between $\mathbf{A}$ of size $I_1 \times R$ and $\mathbf{B}$ of size $I_2 \times R$ is $\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1 \cdots \mathbf{a}_R \otimes \mathbf{b}_R]$ of size $I_1 I_2 \times R$, where $\mathbf{a}_r$ and $\mathbf{b}_r$ are the column vectors of $\mathbf{A}$ and $\mathbf{B}$. We define a tensor $\mathcal{A} \odot_k \mathbf{W}$ of size $I_1 \times \ldots \times I_K \times J$ by using $\mathbf{W}$ of size $I_k \times J$, where $(\mathcal{A} \odot_k \mathbf{W})_{(k)} = \mathbf{A}_{(k)} \odot \mathbf{W}^T$. We denote $\mathcal{A} \odot_1 \mathbf{W}_1 \ldots \odot_K \mathbf{W}_K$ as $\mathcal{A} \prod_k \odot_k \mathbf{W}_k$. A vector, matrix, and tensor whose elements are differentials of F over the elements of $\mathbf{a}$, $\mathbf{A}$, and $\mathcal{A}$ are $\frac{\partial F}{\partial \mathbf{a}}$, $\frac{\partial F}{\partial \mathbf{A}}$, and $\frac{\partial F}{\partial \mathcal{A}}$, respectively.

Problem Definition

Problem 1 *Given a set of pairs of multi-way data and classes* $\{(\mathcal{X}^{(1)}, \mathbf{y}^{(1)}), \cdots, (\mathcal{X}^{(N)}, \mathbf{y}^{(N)})\}$*, learn a predictive model* $\mu(\mathcal{X})$ *that can predict classes* $\mathbf{y}$ *accurately.*

The n-th data can be written as $\mathcal{X}^{(n)} = (E_1^{(n)}, \cdots, E_K^{(n)}, R^{(n)}, L_1^{(n)}, \cdots, L_K^{(n)})$, where $E_k^{(n)}$ is a set of objects of k-th mode, $R^{(n)}$ is relation between objects, *i.e.*, $R^{(n)} : E_1^{(n)} \times \cdots \times E_K^{(n)} \to \mathbb{R}$, and $L_k^{(n)}$ is a projection to a set of labels Λ_k, *i.e.*, $L_k^{(n)} : E_k^{(n)} \to \Lambda_k$. For example, a network traffic log is 3rd-order data in which $(E_1^{(n)}, E_2^{(n)}, E_3^{(n)})$ is (*sIP, dIP, ports*), $R^{(n)}$ is communication using them, and Λ_3 is { 'http' , 'ftp' , ...}. We omit the superscripts indicating the indices of data if they are not required.

Tensor Decomposition

Tucker decomposition (Kolda and Bader 2009) approximates a tensor $\mathcal{X}$ of size $I_1 \times \ldots \times I_K$ by a *core tensor* $\overline{\mathcal{X}}$ of size $J_1 \times \cdots \times J_K$ multiplied by *factor matrices*, so as to minimize $\|\mathcal{X} - \overline{\mathcal{X}} \prod_k \times_k \mathbf{C}_k^T\|_2$, where the size of a factor matrix $\mathbf{C}_k$ is size $I_k \times J_k$. With most fitting algorithms (Kolda and Bader 2009), it is assumed that the factor matrices are column-wise orthonormal.

Neural Network

Our method uses a fully connected neural network with D layers, in which the d-th layer contains U_d neurons. This is the same as a multi-layer perceptron with D layers. The input of the n-th sample into the d-th layer is indicated by $\mathbf{r}_d^{(n)}$, and activation is $\mathbf{a}_d^{(n)} = h(\mathbf{r}_d^{(n)})$ where $h()$ is an activation function. In the input layer, $\mathbf{a}_0^{(n)} = \mathbf{r}_0^{(n)}$. The input of the $(d+1)$-th layer $(0 \leq d < D)$ is calculated by forward propagation, *i.e.*, $\mathbf{r}_{d+1}^{(n)} = \mathbf{W}_d \mathbf{a}_d^{(n)} + \mathbf{b}_d$, where $\mathbf{W}_d$ is a matrix of size $U_{d+1} \times U_d$ and $\mathbf{b}_d$ is a bias. The parameters $\mathbf{W}_d$ and $\mathbf{b}_d$ are optimized so as to minimize the loss function $F = \sum_n loss(\mathbf{y}'^{(n)}, \mathbf{y}^{(n)})$, where $\mathbf{y}'^{(n)}$ is a

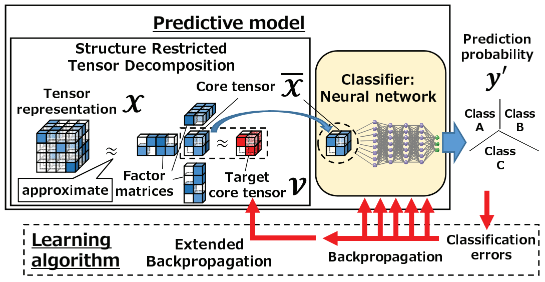

Predictive model
Structure Restricted Tensor Decomposition
Tensor representation $\mathcal{X}$
Core tensor $\overline{\mathcal{X}}$
≈
≈
approximate
Factor matrices
Target core tensor $\mathcal{V}$
Classifier: Neural network
Prediction probability $\mathbf{y}'$
Class A
Class B
Class C
Learning algorithm
Extended Backpropagation
Backpropagation
Classification errors

Figure 2: Overview of DeepTensor. In predictive model, tensor is approximated as *core tensor* multiplied by *factor matrices*, and core tensor is used for classification.

vector indicating the probability of each class calculated by softmax function $y_c'^{(n)} = \mathrm{e}^{r_{Dc}^{(n)}} / \sum_{c'} \mathrm{e}^{r_{Dc'}^{(n)}}$ where $y_c'^{(n)}, r_{Dc}^{(n)}$ are c-th elements of $\mathbf{y}'^{(n)}, \mathbf{r}_D^{(n)}$. In this paper, we use cross entropy $-\sum_n \sum_c y_c^{(n)} \log y_c'^{(n)}$ as the loss function. These parameters are typically optimized using stochastic gradient descent (SGD), which updates parameters into the opposite direction of $\frac{\partial F}{\partial \mathbf{W}_d} = \sum_n \frac{\partial F}{\partial \mathbf{r}_{d+1}^{(n)}} \mathbf{a}_d^{(n)T}$, $\frac{\partial F}{\partial \mathbf{b}_d} = \sum_n \frac{\partial F}{\partial \mathbf{r}_{d+1}^{(n)}}$, which can be calculated using the backpropagation algorithm $\frac{\partial F}{\partial \mathbf{r}_{d+1}^{(n)}} = \left(\mathbf{W}_{d+1}^T \frac{\partial F}{\partial \mathbf{r}_{d+2}^{(n)}} \right) * h'(\mathbf{r}_{d+1}^{(n)})$, where $h'()$ is the first differential of the activation function.

If we use a tensor $\overline{\mathcal{X}}$ of size $J_1 \times \ldots \times J_K$ as the input of a neural network, the parameters between the input layer and 1st layer is a tensor $\mathcal{W}_0$ of size $U_1 \times J_1 \times \ldots \times J_K$.

Proposed Method

DeepTensor Overview

Predictive Model Figure 2 shows an overview DeepTensor, which can learn and classify multi-way data. The input of the predictive model is multi-way data that can be regarded as a tensor $\mathcal{X}$, and the output is a vector $\mathbf{y}'$ indicating the probability of prediction for each class. Our basic idea is to use the core tensor $\overline{\mathcal{X}}$ calculated by Tucker decomposition as input of a neural network. However, the result of Tucker decomposition is not unique (Kolda and Bader 2009), and we do not know which core tensor we should use for classification. To tackle this problem, we propose *Structure Restricted Tensor Decomposition (SRTD)*, explained in more detail in a later section. SRTD not only approximates the input tensor, but also calculates the core tensor as closely as possible to the *target core tensor* $\mathcal{V}$ so that the important structures for classification should be located in similar indices of the core tensor. $\mathcal{V}$ is an "abstract" tensor that expresses important structures for classification, that is, it contains essential structures for each class. By making a core tensor of an instance close to $\mathcal{V}$, the instance is expected to be accurately discriminated; if an instance has an attack-like structure, its core tensor is expected to be similar to the attack-like structure contained in $\mathcal{V}$, and vice versa. The prediction process is shown in Algorithm 1, $Test()$.

Algorithm 1: DeepTensor

```
Function Test ($\mathcal{X}$, $\mathcal{V}$,$\{\mathbf{W}_d\}$,$\{\mathbf{b}_d\}$):
    $\overline{\mathcal{X}}, \{\mathbf{C}_k\} \leftarrow$ SRTD($\mathcal{X}$, $\mathcal{V}$);
    $\mathbf{y}' \leftarrow$ forwardprop($\overline{\mathcal{X}}$, $\{\mathbf{W}_d\}$,$\{\mathbf{b}_d\}$);
    return $\mathbf{y}'$;
Function Train ($\{(\mathcal{X}^{(n)}, \mathbf{y}^{(n)})\}$, size of $\mathcal{V}$, $\{\mathbf{W}_d\}$, $\{\mathbf{b}_d\}$):
    $\mathcal{V}, \{\mathbf{W}_d\}, \{\mathbf{b}_d\} \leftarrow$ init_random(size of $\mathcal{V}$,
      $\{\mathbf{W}_d\}$,$\{\mathbf{b}_d\}$);
    for $epoch \leftarrow 1$ to $maxepoch$ do
        for i $\leftarrow$ mini_batch_indices do
            set $\frac{\partial F}{\partial \mathcal{V}}, \{\frac{\partial F}{\partial \mathbf{W}_d}\}, \{\frac{\partial F}{\partial \mathbf{b}_d}\}$ to zero ;
            for $n \leftarrow$ i do
                $\overline{\mathcal{X}}^{(n)}, \{\mathbf{C}_k^{(n)}\} \leftarrow$ SRTD($\mathcal{X}^{(n)}$, $\mathcal{V}$);
                $\mathbf{y}'^{(n)} \leftarrow$ forwardprop($\overline{\mathcal{X}}^{(n)}$,
                  $\{\mathbf{W}_d\}$,$\{\mathbf{b}_d\}$);
                $\frac{\partial F}{\partial \mathcal{V}}, \{\frac{\partial F}{\partial \mathbf{W}_d}\}, \{\frac{\partial F}{\partial \mathbf{b}_d}\} \leftarrow$
                  $\frac{\partial F}{\partial \mathcal{V}}, \{\frac{\partial F}{\partial \mathbf{W}_d}\}, \{\frac{\partial F}{\partial \mathbf{b}_d}\}$+ backprop($\overline{\mathcal{X}}^{(n)}$,
                  $\mathbf{y}^{(n)}, \mathbf{y}'^{(n)}$, $\mathcal{V}$, $\{\mathbf{W}_d\}$, $\{\mathbf{b}_d\}$);
            $\mathcal{V}, \{\mathbf{W}_d\}, \{\mathbf{b}_d\} \leftarrow$
              SGD($\mathcal{V}$,$\{\mathbf{W}_d\}$,$\{\mathbf{b}_d\}$,$\frac{\partial F}{\partial \mathcal{V}}$,$\{\frac{\partial F}{\partial \mathbf{W}_d}\}$,$\{\frac{\partial F}{\partial \mathbf{b}_d}\}$);
    return $\mathcal{V}$,$\{\mathbf{W}_d\}$,$\{\mathbf{b}_d\}$;
Function SRTD($\mathcal{X}$, $\mathcal{V}$):
    $\{\mathbf{C}_k\} \leftarrow$ init_matrices(size of $\mathcal{X}$,$\mathcal{V}$);
    for $iter \leftarrow 1$ to $maxiter$ do
        for $k \leftarrow 1$ to $K$ do
            $\mathbf{Z}_k \leftarrow$ calcZ($\mathcal{X}$,$\{\mathbf{C}_k\}$,$\mathcal{V}$,k) ;        ▷eq. 7
            $\mathbf{P}_k, \mathbf{S}_k, \mathbf{Q}_k \leftarrow$ SVD($\mathbf{Z}_k$);
            $\mathbf{C}_k \leftarrow \mathbf{P}_k\mathbf{Q}_k^T$;
    $\overline{\mathcal{X}} \leftarrow \mathcal{X}\prod_k \times_k \mathbf{C}_k^T$;
    return $\overline{\mathcal{X}}$,$\{\mathbf{C}_k\}$;
```

Learning Algorithm The training procedure is shown in Algorithm 1, $Train()$. DeepTensor randomly initialize $\mathcal{V}$, $\{\mathbf{W}_d\}$, and $\{\mathbf{b}_d\}$ ($init\_random()$ in $Train()$), and optimize them to minimize the loss function F via SGD. SRTDs are conducted for tensors of all mini-batches and the resulting core tensors are applied to the forward propagation of the neural network. DeepTensor calculates the differentials $\{\frac{\partial F}{\partial \mathbf{W}_d}\}$ and $\{\frac{\partial F}{\partial \mathbf{b}_d}\}$ by using the backpropagation algorithm. The question here is: what is the optimal $\mathcal{V}$ that maximizes the classification accuracy? To answer this question, we introduce the *extended backpropagation (EBP)* algorithm, which calculates the differentials $\frac{\partial F}{\partial \mathcal{V}}$. DeepTensor updates $\mathcal{V}$ along with the parameters $\{\mathbf{W}_d\}$ and $\{\mathbf{b}_d\}$ via SGD. EBP is explained in detail in a later section.

Details of Algorithms

Structure Restricted Tensor Decomposition (SRTD) To get the core tensor as close to $\mathcal{V}$ as possible, SRTD calculates it by using the factor matrices that can approximate the input tensor with $\mathcal{V}$. In case the size of a mode of the input tensor is smaller than that of $\mathcal{V}$, we should multiply the input tensor by the factor matrix to make the core tensor the same size as $\mathcal{V}$. Specifically, the problem of SRTD is:

Problem 2 *Given a tensor $\mathcal{X}$ of size $I_1 \times \ldots \times I_K$ and a target core tensor $\mathcal{V}$ of size $J_1 \times \ldots \times J_K$, calculate the core tensor $\overline{\mathcal{X}}$ that minimizes*

$$\|\mathcal{X} \prod_{k|I_k<J_k} \times_k \mathbf{C}_k - \overline{\mathcal{X}} \prod_{k|I_k \geq J_k} \times_k \mathbf{C}_k^T\|_2, \tag{1}$$

by using the factor matrices $\{\mathbf{C}_k\}$ that minimize

$$\|\mathcal{X} \prod_{k|I_k<J_k} \times_k \mathbf{C}_k - \mathcal{V} \prod_{k|I_k \geq J_k} \times_k \mathbf{C}_k^T\|_2, \tag{2}$$

under the column-wise or row-wise orthonormal condition

$$\begin{cases} \mathbf{C}_k^T\mathbf{C}_k = \mathbf{I} & (I_k \geq J_k), \\ \mathbf{C}_k\mathbf{C}_k^T = \mathbf{I} & (I_k < J_k). \end{cases} \tag{3}$$

We can get the optimal $\overline{\mathcal{X}}$ for Eq. 1 for the given $\{\mathbf{C}_k\}$ as

$$\overline{\mathcal{X}} = \mathcal{X} \prod_k \times_k \mathbf{C}_k, \tag{4}$$

by setting the partial differentials as zero, using Eq. 3. Because calculating the optimal $\{\mathbf{C}_k\}$ is a non-convex optimization problem, we apply the ALS method that alternately optimizes $\mathbf{C}_k$ using the following lemma:

Lemma 1 *By fixing $\mathbf{C}_{k'}(k' \neq k)$, we can calculate a $\mathbf{C}_k$ that minimizes Eq. 2 under the constraints of Eq. 3 as*

$$\mathbf{C}_k = \mathbf{P}_k\mathbf{Q}_k^T, \tag{5}$$

by using singular value decomposition (SVD) as

$$\mathbf{Z}_k = \mathbf{P}_k\mathbf{S}_k\mathbf{Q}_k^T, \tag{6}$$

where $\mathbf{P}_k^T\mathbf{P}_k = \mathbf{I}$, $\mathbf{Q}_k^T\mathbf{Q}_k = \mathbf{I}$, and $\mathbf{S}_k$ is a diagonal matrix with non-negative values, and

$$\mathbf{Z}_k = \left(\mathcal{X} \prod_{k'|k' \neq k} \times_{k'} \mathbf{C}_{k'}\right)_{(k)}^T \mathbf{V}_{(k)}. \tag{7}$$

Proof 1 *To minimize Eq. 2 under the constraints of Eq. 3, we set the partial differentials of the Lagrange function as zero and obtain*

$$\mathbf{Z}_k = \mathbf{C}_k\mathbf{Z}_k^T\mathbf{C}_k, \tag{8}$$

$$\begin{cases} \mathbf{C}_k^T\mathbf{Z}_k = \mathbf{Z}_k^T\mathbf{C}_k & (I_k \geq J_k), \\ \mathbf{C}_k\mathbf{Z}_k^T = \mathbf{Z}_k\mathbf{C}_k^T & (I_k < J_k). \end{cases} \tag{9}$$

We can obtain Eq. 5 by using Eqs. 8 and 9.

Algorithm 1, $SRTD()$ gives the algorithm of SRTD. The value of *maxiter* in SRTD means the maximum number of iteration of the ALS method; if the difference of the value of objective function from that of the previous iteration is below threshold, the iterations stop even before reaching the maxiter. We set the maxiter to 5 on all the experiments.

Extended Backpropagation (EBP) The purpose of EBP is to calculate the differentials of the loss function F over the elements of the target core tensor $\mathcal{V}$. We can calculate $\frac{\partial F}{\partial \mathcal{V}}$ using the following lemma:

Lemma 2 *The differentials $\frac{\partial F}{\partial \mathcal{V}}$ can be calculated as*

$$\frac{\partial F}{\partial \mathcal{V}} = \sum_n \sum_k \mathcal{X}^{(n)} \times_k f^{(n)}(\mathbf{\Delta}_k^{(n)T}) \prod_{k'|k'\neq k} \times_{k'} \mathbf{C}_{k'}^{(n)}, \tag{10}$$

where

$$f(\mathbf{E}) = \mathbf{\Psi}_k \mathbf{E} \mathbf{\Omega}_k + \mathbf{P}_k \left[\left(\mathbf{P}_k^T \mathbf{E} \mathbf{Q}_k - \mathbf{Q}_k^T \mathbf{E}^T \mathbf{P}_k \right) \oslash (\mathbf{S}_k \mathbf{1} + \mathbf{1} \mathbf{S}_k) \right] \mathbf{Q}_k^T,$$

$$\begin{cases} \mathbf{\Psi}_k = (\mathbf{I} - \mathbf{P}_k \mathbf{P}_k^T), & \mathbf{\Omega}_k = \mathbf{Q}_k \mathbf{S}_k^{-1} \mathbf{Q}_k^T \quad (I_k \geq J_k), \\ \mathbf{\Psi}_k = \mathbf{P}_k \mathbf{S}_k^{-1} \mathbf{P}_k^T, & \mathbf{\Omega}_k = (\mathbf{I} - \mathbf{Q}_k \mathbf{Q}_k^T) \quad (I_k < J_k), \end{cases}$$

$$\mathbf{\Delta}_k = \left(\mathcal{W}_0 \times_1 \frac{\partial F}{\partial \mathbf{r}_1} \prod_{k'|k'\neq k} \times_{k'} \mathbf{C}_{k'}^T \right)_{(k)}^T \mathbf{X}_{(k)},$$

under the assumption

$$\begin{cases} \mathbf{Q}_k \mathbf{Q}_k^T = \mathbf{I} & (I_k \geq J_k), \\ \mathbf{P}_k \mathbf{P}_k^T = \mathbf{I} & (I_k < J_k). \end{cases} \tag{11}$$

Proof 2 *The differential of F over an element v of $\mathcal{V}$ is $\frac{\partial F}{\partial v} = \sum_n \langle \frac{\partial F}{\partial \overline{\mathcal{X}}^{(n)}}, \frac{\partial \overline{\mathcal{X}}^{(n)}}{\partial v} \rangle$, which can be written as $\sum_n \sum_k \langle \frac{\partial F}{\partial \overline{\mathcal{X}}^{(n)}}, \mathcal{X}^{(n)} \times_k \frac{\partial \mathbf{C}_k^{(n)}}{\partial v} \prod_{k'|k'\neq k} \times_{k'} \mathbf{C}_{k'}^{(n)} \rangle$ because of Eq. 4. We can obtain $\frac{\partial \mathbf{C}_k}{\partial v} = f(\frac{\partial \mathbf{Z}_k}{\partial v})$ by differentiating both sides of Eqs. 8 and 9 over v, using Eqs. 5 and 6 under the assumption of Eq. 11. We can calculate $\frac{\partial \mathbf{Z}}{\partial v}$ by differentiating both sides of Eq. 7 over v. Equation 10 is obtained using $\frac{\partial F}{\partial v}$, $\frac{\partial \mathbf{C}_k}{\partial v}$, and $\frac{\partial F}{\partial \overline{\mathcal{X}}} = \mathcal{W}_0 \times_1 \frac{\partial F}{\partial \mathbf{r}_1}$, which can be calculated using the backpropagation algorithm.*

Parameter Reduction The labels of objects can be simply regarded as additional modes; *i.e.*, an input tensor is written as $\mathcal{X} \prod_k \odot_k \mathbf{L}_k$, where $\mathbf{L}_k$ are matrices of size $I_k \times I'_k$ in which the (i,j)-th element is one if the i-th object in the k-th mode has a j-th label and is zero otherwise. However, the size of $\mathcal{V}$ becomes multiplicatively too large to avoid overfitting. To remedy this, we propose to write the target core tensor as $\mathcal{V} \prod_k \odot_k \mathbf{M}_k$ by using matrices $\mathbf{M}_k$ of size $J_k \times I'_k$, the size of which increases additively in the size of labels.

The tensor $\mathcal{W}_0$ of size $U_1 \times J_1 \times \ldots \times J_K \times I'_1 \times \ldots \times I'_K$ is still too large to prevent overfitting. To overcome this problem, we set $\mathcal{W}_0$ multi-linear parameters as $\mathcal{W}_0 = \mathbf{G}_1 \odot_1 \ldots \odot_1 \mathbf{G}_K \odot_1 \mathbf{G'}_1 \odot_1 \ldots \odot_1 \mathbf{G'}_K$, where $\mathbf{G}_k$ and $\mathbf{G'}_k$ are matrices of size $U_1 \times J_k$ and $U_1 \times I'_k$ respectively.

Computational Complexity DeepTensor scales linearly to the number of samples; thus we describe the complexity for each sample. The bottleneck part of SRTD is computation of $\mathbf{Z}$ (eq. 7) in $O(|\mathcal{X}||\mathcal{V}|maxiter)$ time and $O(|\mathcal{X}||\mathcal{V}|)$ space. The forward propagation (FP) and backpropagation (BP) in neural networks (NN) between the first layer and the last layer can be done in $O(W)$ time and space for an epoch, where W is the number of edges in NN. The FP from $\mathcal{X}$ to the first layer of NN and the BP from the layer to $\mathcal{V}$, *i.e.* EBP, have the time and space complexity of $O(|\mathcal{X}|((\sum_k J_k + \sum_k I'_k)U_1 + \sum_k J_k^2 + |\mathcal{V}| + \sum_k |\mathbf{M}_k|))$ for an epoch. Because $\mathcal{X}$ is sparse practically, we can reduce $|\mathcal{X}|$ to the number of non-zeros of $\mathcal{X}$.

Interpreting Prediction Results

We now discuss how to determine the reason for the prediction results of DeepTensor, which involves a blackbox model, *i.e.*, neural networks. A core tensor is *interpretable* because we can interpret it in terms of its original tensor by using the factor matrices. Thus, we apply LIME (Ribeiro, Singh, and Guestrin 2016), which tells us the reason for the prediction of blackbox models by using the *interpretable representation* of data. In short, LIME can explain the result of a predictive model f for a sample $\mathbf{x}$ by learning an *interpretable model* g valid locally around $\mathbf{x}$ that minimizes $\mathcal{L}(f, g, \pi_{\mathbf{x}}) = \sum_{\mathbf{z},\mathbf{z}' \in \mathcal{Z}} \pi_{\mathbf{x}}(\mathbf{z}) \left(f(\mathbf{z}) - g(\mathbf{z}') \right)^2$, where $\mathcal{Z}$ is perturbed samples around $\mathbf{x}$ and $\pi_{\mathbf{x}}(\mathbf{z})$ is a proximity measure between $\mathbf{z}$ and $\mathbf{x}$. The input $\mathbf{z}'$ of g is an interpretable representation of $\mathbf{z}$ such as a binary vector.

In our case, f is the predictive model of DeepTensor, and $\mathbf{x}$ is a tensor $\mathcal{X}^{(n)}$. We use training data $\{\mathcal{X}^{(p)}\}$ as the perturbed samples $\mathcal{Z}$ and core tensors $\{\overline{\mathcal{X}}^{(p)}\}$ as the interpretable representations $\{\mathbf{z}'\}$. A proximity measure between $\mathcal{X}^{(p)}$ and $\mathcal{X}^{(n)}$ is calculated by $\pi_n(p) = exp(-\|\overline{\mathcal{X}}^{(n)} - \overline{\mathcal{X}}^{(p)}\|_2^2/\sigma^2)$ where σ is a parameter. We decide the value of σ by observing the interpretation results, however, a certain criteria should be devised in the future. We define g as multiple regression analysis $g_c(\overline{\mathcal{X}}^{(p)}) = \langle \overline{\mathcal{X}}^{(p)}, \mathcal{W}_c^{(n)} \rangle + b_c^{(n)}$ that optimizes $\mathcal{W}_c^{(n)}$ and $b_c^{(n)}$ so as to minimize $\sum_p \pi_n(p) \left\| y_c'^{(p)} - g_c(\overline{\mathcal{X}}^{(p)}) \right\|_2^2$, where $y_c'^{(p)}$ is the probability of predicting class c calculated using DeepTensor. We can set $\mathcal{W}_c^{(n)}$ to multi-linear parameters and optimize it by SGD, the same as $\mathcal{W}_0$ of DeepTensor.

Because the prediction value $\langle \overline{\mathcal{X}}^{(n)}, \mathcal{W}_c^{(n)} \rangle + b_c^{(n)}$ can be written as $\langle \mathcal{X}^{(n)}, \mathcal{W}_c^{(n)} \prod_k \times_k \mathbf{C}_k^T \rangle + b_c^{(n)}$, we can calculate the *contribution scores (CSs)* of the non-zero elements of $\mathcal{X}^{(n)}$ on the prediction of class c as corresponding values of $\mathcal{X}^{(n)} * \mathcal{W}_c^{(n)} \prod_k \times_k \mathbf{C}_k^T$. The higher the CS is, the more likely the non-zero element contributes to the high probability of predicting the class. Note that Ribeiro et al. used sparse linear models, which can highlight elements of the binary vectors. Instead, we use non-sparse linear models because our aim is to calculate the CSs; however, we leave finding better models for future work.

Empirical Results

We evaluate our method from several points of view: classification accuracy, effectivity of SRTD and EBP, scalability, and interpretability.

Datasets

We use multi-way data of various orders from three different domains. A summary of these datasets is shown in Table 1.

Intrusion Detection (IDS) We have downloaded DARPA Intrusion Detection Data Sets[1] and use a 1998 dataset containing logs from seven weeks of training data and two

[1]http://www.ll.mit.edu/ideval/data/

Table 1: Summary of datasets. '# of data': number of data. '# of data in classes': number of data within each class (IDS and LEND have a 'positive' class). '# of avg. relations': average number of relations. 'modes (labels)': name of each mode and its label. '# of avg. objects (# of labels)': average number of objects of each mode and its label.

dataset		# of data	# of data in classes	# of avg. relations	modes (labels)	# of avg. objects (# of labels)
IDS	train(DOS)	112	56(positive); 56	408.4	sIP / dIP / sPort / dPort (service)	16.8 / 35.3 / 304.1 / 52.6 (47)
	train(probing)	1136	568(positive); 568	287.1		12.6 / 25.9 / 208.2 / 59.5 (47)
	test	1224	DOS:506(positive); 718 probing:483(positive); 741	1241.9		15.3 / 36.3 / 819.1 / 165.9 (47)
LEND	train	756	378(positive); 378	4438.8	lender (location) / borrower (location) / lending ID	266.8 (12) / 70.7 (12) / 72.5
	test	1683	9(positive); 1674	4082.5		205.8 (12) / 60.8 (12) / 62.3
BIO	ENZYMES	600	100; 100; 100; 100; 100; 100	62.1	node 1 (label 1) / node 2 (label 2)	32.5 (3) / 32.5 (3)
	NCI1	4110	2057;2053	64.6		29.8 (37) / 29.8 (37)
	NCI109	4127	2079;2048	64.3		29.6 (38) / 29.6 (38)
	CHEM	128404	64202;64202	57.4		26.5 (31) / 26.5 (31)

weeks of test data, which consist of the extensive examples of attacks and background traffic. We use the sIP, dIP, source port (*sPort*), and destination port (*dPort*), and descriptions of *services* such as 'http' or 'ftp' as the label of the dPort. We apply our method to predict whether any attacks occurred in each 10 minute period. The predictive models are learned separately for attacks in two categories that involve the largest number of logs: *DOS* and *probing*. Note that we remove attacks with more than $10,000$ relations that occurred within 10 minutes. We randomly select the same number of periods without attacks as those with attacks to overcome the class-imbalance problem. We repeat these random selections five times and take their average accuracy.

P2P Lending (LEND) P2P lending services enable people to borrow and lend money by matching them directly. We have downloaded transactions from the *Show Me the Money* website[2], which are collected from the three largest P2P lending services in the UK. A lending ID identifies transactions resulting from a lending request from a borrower. We regard lending IDs with interest rates higher than 10.0% as *high-risk lendings* and set our problem to predict them by using the relations between *lender*, *borrower*, and *lending ID* of the *related transactions*. The related transactions are chosen as a combination of transactions of the lending ID and transactions in which the same lenders lent within the past one day. We use *locations* of lenders and borrowers as their labels. If we can identify the high-risk lendings by using very limited information, we can expand the possibilities for new FinTech services. We randomly select the same number of low-risk lendings as high-risk lendings and repeat five times, just as with the IDS.

Bioinformatics (BIO) Note that the primary target of our method is multi-way relations. Yet, we additionally conducted experiments on data representing dyadic relations to gain more understanding of our method. We use ENZYMES, NCI1, and NCI109, downloaded from the ETH zürich website [3]. ENZYMES is a dataset of protein tertiary structures with the 6 EC top-level classes (Borgwardt et al. 2005). NCI1 and NCI109 are datasets of chemical compounds screened for activity against human tumor cell lines (Wale and Karypis 2006). CHEM is provided by the PubChem BioAssay [4] and the task is to predict active compounds which promote or inhibit the activity of a protein, which could lead to drug discovery. We use the AID686978 assay containing the largest number of actives, and randomly selected the same number of inactives as actives.

[2]http://smtm.labs.theodi.org/

[3]https://www.bsse.ethz.ch/mlcb/research/machine-learning/graph-kernels/weisfeiler-lehman-graph-kernels.html

Evaluation Settings

Compared Methods We abbreviate DeepTensor as *DT*. [5] We choose the best parameters by using the validation datasets. The best size of the core tensor is mainly $10 \times 10 \times 10 \times 10$ for IDS, $10 \times 10 \times 10$ for LEND, and 50×50 for BIO, chosen out of $\{10, 25, 50\}$ for each mode. The best number of hidden layers is mainly 4 for IDS, 3 for LEND, 4 for ENZYMES, and 4 for NCI1 and NCI109, chosen out of various numbers of hidden layers; up to 7, with $1,000$ neurons in each layer. The number of epochs for training is 200 and the mini-batch size is 100. We use ReLU (Glorot, Bordes, and Bengio 2011) as the activation function and use it with or without dropout (Srivastava et al. 2014) and batch normalization (Ioffe and Szegedy 2015) techniques.

We compare DT with *TuckerNN*, a method using the same settings as those of DT, except those using Tucker decomposition instead of SRTD. We use the higher-order orthogonal iteration (HOOI) (Lathauwer, Moor, and Vandewalle 2000), and align the elements of the core tensor according to the top-J_k singular values. When $I_{kn} < J_k$, the remaining elements of the core tensor are set to zero. Moreover, we evaluate DeepTensor without EBP (*noEBP*), in which SGD never updates the target core tensor. We also test our method using neural networks without hidden layers (*noHL*), in which a core tensor is directly connected to the softmax layer, which is almost the same as a logistic regression.

We also compare DT with SVMs with graphlet kernel (*GK*) (Shervashidze et al. 2009), shortest path kernel (*SP*) (Borgwardt and Kriegel 2005), and Weisfeiler-Lehman

[4]https://www.ncbi.nlm.nih.gov/pcassay

[5]DeepTensor and TuckerNN are implemented in Python 2.7, using Chainer 1.24 (http://chainer.org/). All the experiments were conducted on an Intel(R) Xeon(R) CPU E5-2630 v3 2.40GHz with 132GB of memory and a Tesla P100 GPU, running Linux.

Table 2: Average precision values (%) (IDS and LEND) and classification accuracies (%) (BIO)

data		GK	SP	WL	TuckerNN	noHL	noEBP	DT
IDS	DOS	N/A	N/A	45.05±7.14	48.75±2.59	43.39±3.03	61.00±5.27	**62.18±3.56**
	probing	N/A	N/A	40.13±2.57	62.10±5.05	36.03±5.72	66.16±10.84	**78.52±3.65**
LEND		N/A	N/A	0.67±0.19	4.99±1.15	10.10±8.51	4.78±3.42	**12.07±1.40**
BIO	ENZYMES	30.60±1.64	39.53±1.34	**50.90±2.19**	21.93±0.44	25.17±1.94	47.07±0.76	47.17±1.46
	NCI1	65.86±0.31	73.32±0.64	**79.98±0.48**	63.25±0.81	70.42±0.62	75.15±0.41	75.22±0.30

Table 3: Accuracy of interpretation (IDS)

	DOS		probing	
	TuckerNN	DT	TuckerNN	DT
sIP	19.96±4.79	**33.49±4.90**	24.96±3.26	**51.54±9.65**
dIP	9.54±0.20	**22.93±3.65**	24.12±1.74	**51.38±9.92**
sPort	13.16±4.66	**28.44±10.68**	26.47±0.00	**49.46±10.03**
dPort	20.60±4.89	**41.92±5.24**	27.43±2.53	**51.89±10.21**

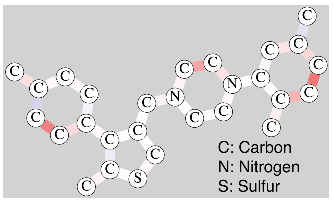


Figure 3: Example interpretation (CHEM). Bonds with high absolute CSs are colored: red (positive) or blue (negative); the stronger color indicates higher absolute value. ($\sigma = 5.0$)

Table 4: Example interpretation (IDS probing). '# of rels.': number of relations corresponding to the objects. ($\sigma = 10.0$)

Jul 29 18:50 - 19:00 (612 relations)							
sIP (total 10)				dIP (total 15)			
objects		# of rels.	CS	objects		# of rels.	CS
⋆	207.253.084.013	6	0.636	⋆	172.016.118.020	6	0.636
	172.016.114.207	66	0.207		207.037.252.205	66	0.207
	207.230.054.203	117	0.001		206.132.025.051	27	0.005
sPort (total 371)				dPort (total 246)			
objects		# of rels.	CS	objects		# of rels.	CS
⋆	49724	6	0.636	⋆	584	3	0.410
	8739	1	0.003		80 (http)	130	0.186
	8328	1	0.003	⋆	410	2	0.181

subtree kernel (*WL*) (Shervashidze et al. 2011) by converting the higher order data into graph data, in which nodes representing relations are connected to the nodes representing objects. The labels are added to the nodes of objects, and we assume an object without a label has a label indicating the mode. BIO is simply evaluated as graph data. We use the Matlab code downloaded from the ETH zürich website [3]. The parameters are selected by grid search. Note that both GK and SP are infeasible on our higher order data because of enormous calculation quantity and memory space.

Evaluation Metrics Since intrusion detection (IDS) and high-risk detection (LEND) are anomaly detection tasks, we use the classification probabilities as anomaly scores. For SVMs, we use the distances from hyperplanes as anomaly scores. We use *average precision* as a measure of accuracy, that is, the average of the precision values (# of true positives divided by # of predicted positives) among all the levels of the recall (# of true positives divided by # of all positives). The average precision is one commonly used evaluation measure for anomaly detection tasks (Akoglu et al. 2012). We plot the precision against the recall by varying the threshold on each anomaly score, and calculate the area under the precision-recall curve as the average precision. On BIO, we evaluate the methods based on 10-fold cross validation repeated five times and take their average accuracy.

Experimental Results

Prediction Accuracy The results are summarized in Table 2. The results of NCI109 are omitted because they are almost the same as those of NCI1. DT is more accurate than other methods on IDS and LEND, which strongly suggests that our method works better than conventional methods on these higher-order data. The accuracies of DT on LEND (about 12%) might seem very low. However, DT successfully detects more than half of the 9 high-risk lendings by picking up only less than 2% of all $1,683$ lendings in the test data. On the other hand, WL exhibits the highest accuracies on BIO, which suggests that this graph kernel is well-designed to classify chemical compounds or proteins. That said, the accuracies of DT on BIO are generally in the middle among all of the results, which is fairly good. Overall, DT performs far better than TuckerNN, which suggests that SRTD works much more effectively than Tucker decomposition in our problems. Moreover, the accuracies of DT are better than those of noEBP, which suggests that EBP can improve the prediction of our method. However, the results of DT are almost the same as those of noEBP on BIO, possibly because the target core tensor is so large that the important structures can be obtained even after random initialization. In addition, we show that DT performs far better than noHL on all datasets, which indicates that the hidden layers of the neural networks are essential for our method.

As a scalability test, we used various numbers of compounds from CHEM and confirmed that a training time scales almost linearly to the number of compounds. The accuracy is about 75% after learning $128,404$ compounds, whereas it is about 66% when learning $1,286$ compounds.

Interpretability We also report the interpretability of DT by showing how accurately the interpretable models can spot the true attacks on the IDS dataset. We select samples of a test dataset with more than 0.9 probability of prediction for attacks and calculate the CSs on the prediction. We sum up the CSs by the objects of each mode and calculate the ratio of samples in which the objects involving attacks are successfully spotted within the objects with top-3 highest CSs (Table 3). We also calculate the CSs of TuckerNN the same as with DT. DT spots attacks much more accurately than TuckerNN on both DOS and probing. Table 4 shows an example of interpretation on probing, which has the highest probability of involving attacks. The objects with the highest CSs successfully spot the true attacks (stars in Table 4). DT can also report the reason for prediction results on graph data such as chemical compounds. Figure 3 shows an example on CHEM with the highest probability of active compounds, summing up two CSs corresponding to each bond (node1 - node2, node2 - node1). Bonds with the high CSs possibly play an important role in protein binding and can be modified to fit better to the target, which should be evaluated based on the biological experiments in the future.

Conclusion

We proposed DeepTensor to tackle the problem of classifying multi-way data by leveraging a novel tensor decomposition called SRTD. SRTD calculates the core tensors, which express the important structures for classification, by making these core tensors close to a target core tensor. We also introduced EBP, which learns the optimal target core tensor along with the parameters of the neural network. We empirically show that DeepTensor is highly accurate on a wide variety of multi-way data, especially on higher order data. In addition, DeepTensor provides us with a way to investigate the reason for the prediction by interpreting the core tensor using the factor matrices, which could enable us to gain more understandings of the target phenomenon.

Acknowledgements

We would like to thank Prof. Yasushi Okuno, Prof. Hisashi Kashima, and Ken Kobayashi for their useful comments.

References

Acar, E., and Yener, B. 2009. Unsupervised multiway data analysis: A literature survey. *IEEE Trans. Knowl. Data Eng.* 21(1):6–20.

Akoglu, L.; Tong, H.; Vreeken, J.; and Faloutsos, C. 2012. Fast and reliable anomaly detection in categorical data. In *CIKM*, 415–424.

Borgwardt, K. M., and Kriegel, H. 2005. Shortest-path kernels on graphs. In *ICDM*, 74–81.

Borgwardt, K. M.; Ong, C. S.; Schönauer, S.; Vishwanathan, S. V. N.; Smola, A. J.; and Kriegel, H. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21(1):47–56.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 3837–3845.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *AISTATS*, 315–323.

He, L.; Kong, X.; Yu, P. S.; Yang, X.; Ragin, A. B.; and Hao, Z. 2014. Dusk: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages. In *SDM*, 127–135.

Imaizumi, M., and Hayashi, K. 2016. Doubly decomposing nonparametric tensor regression. In *ICML*, 727–736.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 448–456.

Kashima, H.; Tsuda, K.; and Inokuchi, A. 2003. Marginalized kernels between labeled graphs. In *ICML*, 321–328.

Kiers, H. A. 1998. Joint orthomax rotation of the core and component matrices resulting from three-mode principal components analysis. *Journal of Classification* 15(2):245–263.

Kolda, T. G., and Bader, B. W. 2009. Tensor decompositions and applications. *SIAM Review* 51(3):455–500.

Lake, B. M.; Salakhutdinov, R.; and Tenenbaum, J. B. 2015. Human-level concept learning through probabilistic program induction. *Science* 350(6266):1332–1338.

Lathauwer, L. D.; Moor, B. D.; and Vandewalle, J. 2000. On the best rank-1 and rank-(R_1 , R_2, ... , R_N) approximation of higher-order tensors. *SIAM J. Matrix Analysis Applications* 21(4):1324–1342.

Lipton, Z. C. 2016. The mythos of model interpretability. *CoRR* abs/1606.03490.

Martin, C. D. M., and Loan, C. F. V. 2008. A jacobi-type method for computing orthogonal tensor decompositions. *SIAM J. Matrix Analysis Applications* 30(3):1219–1232.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *ICML*, 2014–2023.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why should I trust you?": Explaining the predictions of any classifier. In *KDD*, 1135–1144.

Rudin, C. 2014. Algorithms for interpretable machine learning. In *KDD*, 1519.

Shervashidze, N.; Vishwanathan, S. V. N.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. M. 2009. Efficient graphlet kernels for large graph comparison. In *AISTATS*, 488–495.

Shervashidze, N.; Schweitzer, P.; van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12:2539–2561.

Si, Z., and Zhu, S. 2013. Learning AND-OR templates for object recognition and detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 35(9):2189–2205.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Tao, D.; Li, X.; Wu, X.; Hu, W.; and Maybank, S. J. 2007. Supervised tensor learning. *Knowl. Inf. Syst.* 13(1):1–42.

Wale, N., and Karypis, G. 2006. Comparison of descriptor spaces for chemical compound retrieval and classification. In *ICDM*, 678–689.

Yanardag, P., and Vishwanathan, S. V. N. 2015. Deep graph kernels. In *KDD*, 1365–1374.

Zhou, H.; Li, L.; and Zhu, H. 2013. Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association* 108(502):540–552.