J. Indian Inst. Sci.
A Multidisciplinary Reviews Journal
ISSN: 0970-4140 Coden-JIISAD

REVIEW ARTICLE

Hyperedge Prediction Using Tensor Eigenvalue Decomposition

Deepak Maurya[*] *and Balaraman Ravindran*

Abstract | Link prediction in graphs is studied by modeling the dyadic interactions among two nodes. The relationships can be more complex than simple dyadic interactions and could require the user to model super-dyadic associations among nodes. Such interactions can be modeled using a hypergraph, which is a generalization of a graph where a hyperedge can connect more than two nodes. In this work, we consider the problem of hyperedge prediction in a *k*-uniform hypergraph. We utilize the tensor-based representation of hypergraphs and propose a novel interpretation of the tensor eigenvectors. This is further used to propose a hyperedge prediction algorithm. The proposed algorithm utilizes the *Fiedler* eigenvector computed using tensor eigenvalue decomposition of hypergraph Laplacian. The *Fiedler* eigenvector is used to evaluate the construction cost of new hyperedges, which is further utilized to determine the most probable hyperedges to be constructed. The functioning and efficacy of the proposed method are illustrated using some example hypergraphs and a few real datasets. The code for the proposed method is available on https://github.com/d-maurya/hypred_tensorEVD.

Keywords: *Hypergraphs, Spectral hypergraph theory, Hyperedge prediction, Tensor eigenvalue decomposition*

1 Introduction

Link prediction is the study of predicting the existence of an edge between two nodes in a graph. This problem has found its application in various domains such as bioinformatics[18, 22], social networks[27], and recommender systems[12]. Most of existing heuristic approaches like common neighbours[3], Jaccard index[16], Adamic-Adar[35], PageRank[6] are limited to modelling pairwise interactions only.

Relationships among nodes can be more complex than simple pairwise associations. Hypergraphs relax this assumption of pairwise interaction and provide the freedom to model the interaction among k nodes. Such networks commonly occur in social networks[7,15], metabolic networks[32], recommender systems[17, 28] and multi-actor collaboration[25].

Most of the recent works on hyperedge prediction[13, 30, 32] utilize the approach of clique expansion to reduce the hypergraph to a graph[1], followed by applying standard graph algorithms[34] or recently proposed approach modeling triadic simplicial closure[4]. We believe this step of hypergraph reduction to a graph restricts the user to model some weighted form of dyadic interaction rather than the intended super-dyadic interactions among nodes. This can be argued by the fact that the expression for scalar Laplacian objective function minimized for the estimation of Fiedler vector (eigenvector corresponding to minimum positive eigenvalue) contains only bi-linear terms instead of higher order polynomials. This article emphasis on the use of higher order polynomials to capture the complex interactions among the nodes in a hypergraph.

The availability of vast literature, theoretical guarantees, and scalable algorithms for graphs are the main reasons for adopting hypergraph reduction methods[9]. There is significant loss of information about the hypergraph structure caused by this reduction step. For example, two entirely

Hypergraph: A generalization of a graph in which an edge can join any number of vertices.

Clique: A subset of the vertices in a graph, such that every two distinct vertices are adjacent.

Dyadic: A group of exactly two entities.

Super-dyadic: A group of any number (> 2) of entities.

[1] *Computer Science and Engineering Department, Robert Bosch Centre for Data Science and AI, Indian Institute of Technology Madras, Chennai, India.* **maurya@cse.iitm.ac.in*

different hypergraphs can reduce to same graph after the reduction step. We have given example for such cases later. This observation clearly demonstrates that the unique information about two different hypergraphs is lost after the hypergraph reduction step.

In this work, we approach the problem of hyperlink prediction without performing any reduction. For this purpose, we prefer to utilize the tensor-based representation of hypergraphs[24] rather than the usual matrix based notation widely accepted in machine learning community[33]. The tensor-based representation provides us the freedom to model the super-dyadic interactions among nodes. The proposed algorithm in this work is highly motivated from spectral graph theory with appropriate modifications.

Tensor: A multi-dimensional array.

The widely accepted framework for link prediction using spectral graph theory framework comprises of computing the similarity metric between two nodes[27]. The similarity measure is defined as a function of embeddings of two nodes, which can be derived from the graph Laplacian[29]. We attempt to utilize the same approach for hypergraphs by computing the eigenvectors of hypergraph Laplacian tensor but we encountered several challenges. To name a few, the eigenvectors of a real symmetric tensor are not orthogonal, which is contrary to the case of real symmetric matrices. The number of eigenvectors for a symmetric tensor is not fixed, unlike the simple case of symmetric matrices. The tensor eigenvectors cannot be trivially interpreted.

Despite various existing challenges, we pursue the tensor-based representation of hypergraphs due to the strong motivation developed from some of the recent intriguing results found in spectral hypergraph theory using tensor representation. For example, Hu et al.[10] proved that the algebraic multiplicity of *zero eigenvalue* of a symmetric tensor is equal to the sum of the number of even-bipartite connected components and number of connected components, minus the number of singletons in the corresponding hypergraph. This information couldn't be revealed from the clique reduction methods and its variants[1].

In this work, we present a novel approach to interpret the tensor eigenvectors. This helps us to define the construction cost for new potential hyperedges in a given hypergraph. The next step in the proposed algorithm is to prefer the prediction of hyperedges with minimum construction cost. In the perspective of spectral hypergraph theory, the key idea of the proposed algorithm can also be perceived as the inclusion of new hyperedges such that there is minimal perturbation in the "smoothness" of the hypergraph. In spectral graph theory, the "smoothness" of the graph is characterized by the Fiedler eigenvalue[8]. The same analogy is utilized in this work also. The code for the proposed method can be accessed from[A]

The rest of the paper is organized as follows. We introduce the matrix and tensor-based notation of hypergraphs in Sect. 3. We also discuss the merits of tensor-based notation in this section. The proposed algorithm for hyperedge prediction is described in Sect. 4. The functioning and fruitful merits of the proposed algorithm is demonstrated in Sect. 5 using small synthetic and real hypergraphs. Concluding remarks and future directions of this work are discussed in Sect. 6.

Notations: A scalar is denoted by lowercase alphabet x, a vector by bold face $\mathbf{x}$, a matrix by bold face uppercase alphabet $\mathbf{X}$ and a tensor by italics uppercase alphabet $\mathcal{X}$. The subscript a over a vector such as $\mathbf{x}_a$ indicates the dimension, and for a tensor $\mathcal{X}_a$, it denotes the mode of a tensor, which is defined later.

2 Preliminaries

In this section, we discuss the matrix and tensor-based representation of hypergraphs briefly.

A hypergraph G is formally defined as a pair of $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of entities called vertices or nodes and $E = \{e_1, e_2, \ldots, e_m\}$ is a set of non-empty subsets of V referred as hyperedges. In general, $E \in \mathcal{P}(V)\backslash\{\phi\}$ for a *non-uniform* hypergraph, where $\mathcal{P}(V)$ is the power set of V. In this article, we focus on k-uniform hypergraphs, and hence E is restricted to a subset of all the $\binom{n}{k}$ combinations of elements from V, where $n = |V|$. The strength of interaction among nodes in the same hyperedge is quantified by the positive weight represented by $w_e = \{w_{e_1}, w_{e_2}, \ldots, w_{e_m}\}$. The vertex-edge incidence matrix is denoted by $\mathbf{H}$ and has the dimension $|V| \times |E|$. The entry $h(i, j)$ is defined as:

$$h(i,j) = \begin{cases} 1 & \text{if } v_i \in e_j \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The degree of node v_i is defined by $d(v_i) = \sum_{e_j \in E} w_{e_j} h(i,j)$. $\mathbf{D}$ is a diagonal matrix with $D(i,i) = d(v_i)$.

[A] https://github.com/d-maurya/hypred_tensorEVD.

2.1 Matrix Representation

It should be noted that there is no loss of information about the hypergraph structure in the incidence matrix representation. This could also be perceived as the existence of unique mapping (up to certain reordering) between given hypergraph $G(V, E)$ and its incidence matrix $\mathbf{H}$.

Most of the machine learning algorithms for classification, partitioning of graphs, and link prediction operates on the adjacency matrix rather than incidence matrix. The adjacency matrix for an undirected graph is symmetric and its entries $\mathbf{A}(i,j)$ is defined as:

$$\mathbf{A}(i,j) = \begin{cases} w_{ij} & \text{if } \{v_i, v_j\} \in E \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where w_{ij} indicates the weight of the edge. It can be inferred that the entry (i, j) denotes the existence of an edge.

Agarwal et al.[1] define the adjacency matrix for the reduced hypergraphs as follows:

$$\mathbf{A}_r = \mathbf{H}\mathbf{W}\mathbf{H}^T - \mathbf{D}. \tag{3}$$

It should be noticed that most of the reduction methods are a non-unique mapping from hypergraph to adjacency matrix. It means that there could be distinct hypergraphs which reduce to the same graph. For example, the clique reduction approach reduces the four-uniform hypergraph and the three-uniform hypergraph to the same graph as shown in Fig. 1.

This non-uniqueness property of hypergraph reduction method plays a very crucial role in the task of hyperedge prediction. The reduced hypergraph has lost the information about the original hypergraph structure. So there is no assurance of any analysis on reduced hypergraph to deliver correct results for the original hypergraph. To avoid the loss of information in the reduction step, we utilize the tensor-based representation of hypergraphs discussed in the next section.

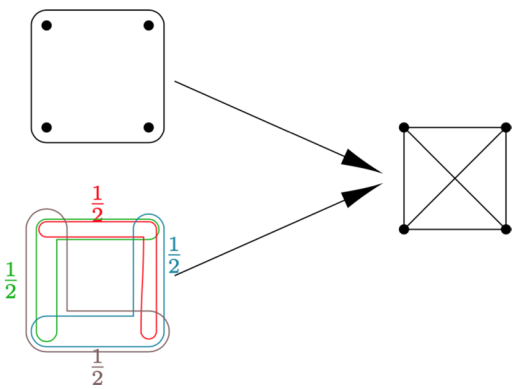

Figure 1: Two hypergraphs reducing to same graph.

2.2 Tensor-Based Representation

The entry $\mathbf{A}(i,j)$ in adjacency matrix for a graph denotes the strength of interaction between the nodes i and j. Similarly, for a hyperedge with 3 nodes, a 3-dimensional tensor is required to represent the strength of interaction among 3 nodes. This idea can be further generalized to represent k-uniform hypergraphs.

Therefore, a natural representation of hypergraphs is a k-order n-dimensional tensor $\mathcal{A}$[24], which consists of n^k entries

$$\mathcal{A} = \left(a_{i_1 i_2 \ldots i_k}\right), \qquad a_{i_1 i_2 \ldots i_k} \in \mathbb{R}, \qquad 1 \le i_1, \ldots, i_k \le n \tag{4}$$

The entries of above tensor is defined as:

$$a_{i_1 i_2 \ldots i_k} = \begin{cases} w_{e_j} \frac{1}{(k-1)!} & \text{if } (i_1, i_2, \ldots, i_k) = \{e_j\} \quad e_j \in E \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

It should be noted that $\mathcal{A}$ is a "*super-symmetric*" tensor i.e., for any permutation of the indices, $\mathcal{A}$ contains the same value:

$$a_{i_1 i_2 \ldots i_k} = a_{\sigma(i_1 i_2 \ldots i_k)},$$

where $\sigma(i_1, i_2, \ldots i_k)$ denotes any permutation of the elements in the set $\{i_1, i_2, \ldots, i_k\}$. The order or mode of tensor refer to hyperedge cardinality, which is k for $\mathcal{A}$. The degree of a vertex v_i is given by

$$d(v_i) = \sum_{i_k=1}^{n} \cdots \sum_{i_3=1}^{n} \sum_{i_2=1}^{n} a_{i i_2 i_3 \ldots i_k}. \tag{6}$$

The particular factor of $1/(k-1)!$ is chosen while defining the adjacency tensor in (5) so that the node degree can be defined appropriately in the graph theoretic sense. One could also think as the generalization for the case of graphs ($k = 2$).

The degree of all the vertices can be represented by k-order n-dimensional diagonal tensor $\mathcal{D}$:

$$d_{i_1 i_2 \ldots i_k} = \begin{cases} d(v_i) & \text{if } i_1 = i_2 \cdots = i_k = i \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The Laplacian tensor $\mathcal{L}$ is defined as follows :

$$\mathcal{L} = \mathcal{D} - \mathcal{A}. \tag{8}$$

The elements of Laplacian tensor $\mathcal{L}$ are described by

$$l_{i_1 i_2 \ldots i_k} = \begin{cases} -w_{e_j} \frac{1}{(k-1)!} & \text{if } (i_1, i_2, \ldots, i_k) \in \{e_j\}, \quad j \in [m] \\ d(v_i) & \text{if } i_1 = i_2 \cdots = i_k = i \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Normalized hypergraph Laplacian tensor[2] denoted by $\mathscr{L}$, is defined:

$$\ell_{i_1 i_2 \ldots i_k} = \begin{cases} -w_{e_j} \frac{1}{(k-1)!} \prod_{i_j=1}^{k} \frac{1}{\sqrt[k]{d_{i_j}}} & \text{if } (i_1, i_2, \ldots, i_k) \in \{e_j\}, \quad j \in [m] \\ 1 & \text{if } i_1 = i_2 \cdots = i_k = i \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

One of the standard approaches in spectral graph theory is to perform spectral decomposition of the graph Laplacian for link prediction. Proceeding along similar lines, we evaluate the eigenvectors of Laplacian tensor[24]:

$$\begin{aligned} \mathcal{L}\mathbf{x}^{k-1} &= \lambda \mathbf{x} \\ \mathbf{x}^T \mathbf{x} &= 1, \end{aligned} \tag{11}$$

where $(\lambda, \mathbf{x}) \in (\mathbb{R}, \mathbb{R}^n \backslash \{0\}^n)$ is called the Z-eigenpair and $\mathcal{L}\mathbf{x}^{k-1} \in \mathbb{R}^n$, whose ith component is defined as

$$\left[\mathcal{L}\mathbf{x}^{k-1}\right]_i = \sum_{i_k=1}^{n} \cdots \sum_{i_3=1}^{n} \sum_{i_2=1}^{n} l_{i i_2 i_3 \ldots i_k} x_{i_2} x_{i_3} \ldots x_{i_k}. \tag{12}$$

The above equations arises from the following optimization problem:

$$\begin{aligned} \min_{\mathbf{x}} \mathcal{L}\mathbf{x}^k &= \sum_{i_k=1}^{n} \cdots \sum_{i_2=1}^{n} \sum_{i_1=1}^{n} l_{i_1 i_2 \ldots i_k} x_{i_1} x_{i_2} \ldots x_{i_k} \\ \text{such that} \quad & \mathbf{x}^T \mathbf{x} = 1. \end{aligned} \tag{13}$$

The eigenvector with minimum positive λ satisfying (11) is termed as Fiedler eigenvector and can be computed using

$$\begin{aligned} \mathbf{v}_\star = \underset{\mathbf{x}}{\text{argmin}} \quad & \mathcal{L}\mathbf{x}^k > 0 \\ \text{s. t} \quad & \mathcal{L}\mathbf{x}^k = \lambda \mathbf{x} \\ & \mathbf{x}^T \mathbf{x} = 1. \end{aligned} \tag{14}$$

The corresponding eigenvalue can be computed as $\lambda_\star = \mathcal{L}\mathbf{v}_\star^k$.

Illustrative Example: A simple example illustrating the procedure to construct the adjacency and Laplacian tensor for a 4-uniform hypergraph has been presented. Consider a 4-uniform hypergraph $G(V, E)$ as shown below:

where the set of vertices and hyperedges are defined by

$$\begin{aligned} V &= \{v_1, v_2, v_3, v_4, v_5\} \\ E &= \{\{v_1, v_2, v_3, v_4\}, \{v_2, v_3, v_4, v_5\}, \{v_1, v_2, v_3, v_5\}\}. \end{aligned}$$

The adjacency tensor for the above hypergraph is denoted by $\mathcal{A}$ and has dimensions $5 \times 5 \times 5 \times 5$. It should be noted that the cardinality (k) of all 3 hyperedges is 4 . The elements of $\mathcal{A}$ are denoted by a_{i_1,i_2,i_3,i_4}, where $1 \le i_k \le 5$. It should be noted that $\mathcal{A}$ contains $n^k = 5^4$ elements but only $m \times k! = 3 \times 4! = 72$ elements have non-zero entries. The elements corresponding to first hyperedge are described by

$$\begin{aligned} a_{1234} =& a_{1243} = a_{1324} = a_{1342} = a_{1423} = a_{1432} \\ =& a_{2134} = a_{2143} = a_{2314} = a_{2341} = a_{2413} = a_{2431} \\ =& a_{3214} = a_{3241} = a_{3124} = a_{3142} = a_{3421} = a_{3412} \\ =& a_{4231} = a_{4213} = a_{4321} = a_{4312} = a_{4123} = a_{4132} = c, \end{aligned}$$

where $c = \frac{1}{(k-1)!} = \frac{1}{6}$. The vertex degrees can be stored in a tensor of dimension $5 \times 5 \times 5 \times 5$ with its diagonal elements being $d(v) = \begin{bmatrix} 2 & 3 & 3 & 2 & 2 \end{bmatrix}$. The tensor Laplacian has dimension of $5 \times 5 \times 5 \times 5$ and its entries can be obtained using (9), which are found to be

$$l_{i_1 i_2 i_3 i_4} = \begin{cases} -\frac{1}{6} & \text{if } (i_1, i_2, i_3, i_4) = \{e_j\}, \quad j = \{1, 2, 3\} \\ d(v_i) & \text{if } i_1 = i_2 = i_3 = i_4 = i \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

This example shows the procedure to construct the adjacency and Laplacian tensor for any k-uniform hypergraph.

3 Proposed Method for Hyperedge Prediction

In this section, we propose the hyperedge prediction algorithm using spectral decomposition for tensors. We have already discussed the tensor-based representation of hypergraphs in Sect. 3.

The proposed framework derives the cost of each potential hyperedge and prefers to choose the hyperedges with minimum cost. The cost of creating a hyperedge is calculated from the Fiedler eigenvector of the Laplacian tensor defined in (14). Hence, we present the following theorem for the expression of hypergraph (tensor) Laplacian objective function used for the computation of tensor eigenvectors.

Theorem 1 *The hypergraph Laplacian cost function for a k-uniform hypergraph can be expressed as*

$$\begin{aligned}\mathcal{L}\mathbf{x}^k &= \sum_{e_j \in E} l_{e_j}(\mathbf{x}) \\ l_{e_j}(\mathbf{x}) &= w_{e_j}\left(\sum_{i_k \in e_j} x_{i_k}^k - k\prod_{i_k \in e_j} x_{i_k}\right) \\ &= w_{e_j} k\left(\underset{i_j \in e_j}{\text{A.M}}\left(x_{i_k}^k\right) - \underset{i_j \in e_j}{\text{G.M}}\left(|x_{i_k}|^k\right)(-1)^{n_s}\right),\end{aligned} \tag{16}$$

where $n_s = |\{i_j : x_{i_j} < 0\}|$, *A.M and G.M stand for the arithmetic and geometric means, respectively. We refer* l_{e_j} *as the cost for hyperedge* e_j *in rest of the paper.*

Proof Please refer Theorem 8 in Maurya et al.[20].

Using Theorem 1, the computation of $\mathcal{L}\mathbf{x}^k$ can be done in $O(|E|)$ steps which would have been $O(|V|^k)$ for any general tensor.

Illustrative Example: Through this example, we demonstrate the use of Theorem 1 in the computation of tensor Laplacian. We also unveil the challenges involved in working with tensor eigenvectors—for example, the non-orthogonality of tensor eigenvectors. Consider the hypergraph shown in Fig. 2. The hypergraph Laplacian cost function for this hypergraph can be derived using (16):

$$\begin{aligned}\mathcal{L}\mathbf{x}^k = {} & x_1^4 + x_2^4 + x_3^4 + x_4^4 - 4x_1x_2x_3x_4 \\ & + x_1^4 + x_2^4 + x_3^4 + x_5^4 - 4x_1x_2x_3x_5 \\ & + x_2^4 + x_3^4 + x_4^4 + x_5^4 - 4x_2x_3x_4x_5.\end{aligned} \tag{17}$$

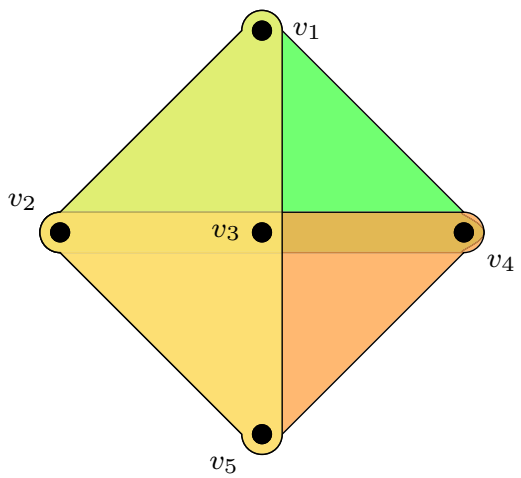

Figure 2: H4: 4-uniform.

The 4th order homogeneous polynomial is the objective function the optimization problem mentioned in (13). This is required for the computation of the eigenvalues and eigenvectors of $\mathcal{L}$. We further discuss the properties of zero eigenvalues and zero eigenvectors of Laplacian tensor.

Lemma 2 *One of the Z-eigenpair of* $\mathcal{L}$ *is* $(0, \mathbf{v})$, *where* $\mathbf{v} = \frac{1}{\sqrt{n}}(1, 1, \ldots, 1) \in \mathbb{R}^n$.

*Proof*Please refer Banerjee et al.[2]: Theorem 3.13 (iv). □

It should be noted that $\sqrt{n}$ is just a scaling factor in $\mathbf{v}$ to ensure $\mathbf{v}^T\mathbf{v} = 1$. One could also consider unity vector as eigenvector with eigenvalue 0.

Lemma 3 *The number of zero eigenvalues of the graph Laplacian indicates the number of connected components*[8].

The above property does not hold for hypergraphs. It means that a fully connected hypergraph can have multiple zero eigenvalues[10]. We also make the same observation in this example as explained below.

It is observed that the tensor Laplacian for hypergraph H4 shown in Fig. 2 has 2 zero eigenvalues and the distinct eigenvectors are as follows:

$$\mathbf{V} = \frac{1}{\sqrt{5}}\begin{bmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 1 \\ 1 & -1 \\ 1 & -1 \end{bmatrix}. \tag{18}$$

This illustrates that the eigenvalues could be zero even if the eigenvector is not unity vector, contrary to graphs. This is surprising because there is only one connected component, but there are two zero eigenvalues.

This observation can be explained by computing the cost of each hyperedge using (16) for the eigenvectors stated above. It is observed that the cost of all the three hyperedges is zero for both the eigenvectors. As a result of which, the eigenvalue is zero. Another distinguishable property is that the eigenvectors are *not* orthogonal unlike the case of graphs (having real symmetric Laplacian matrix).

We have just discussed the use of hyperedge cost from eigenvectors corresponding to zero eigenvalues. We extend the similar discussion on hyperedge score computed from the Fiedler eigenvalue and eigenvector.

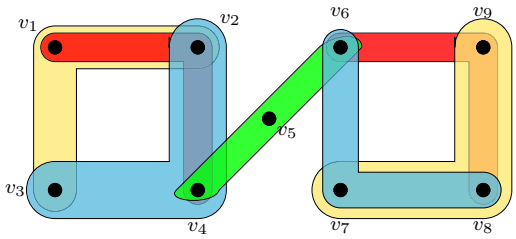

Figure 3: H5: 3-uniform hypergraph.

Illustrative Example: In this example, we demonstrate a novel interpretation of tensor eigenvectors.

Consider the 3-uniform hypergraph shown in Fig. 3. The Laplacian tensor can be easily constructed using (9). The next step is to compute the Fiedler eigenvalues and eigenvectors satisfying (11). It is observed that there are four Fiedler vectors with the eigenvalue of 0.0569 as reported below:

$$\mathbf{V} = \begin{bmatrix} -0.05 & 0.06 & 0.47 & 0.47 \\ 0.03 & 0.03 & 0.46 & 0.46 \\ 0.06 & -0.05 & 0.47 & 0.47 \\ 0.23 & 0.23 & 0.42 & 0.42 \\ 0.34 & 0.34 & 0.34 & 0.34 \\ 0.42 & 0.42 & 0.23 & 0.23 \\ 0.47 & 0.47 & -0.05 & 0.06 \\ 0.46 & 0.46 & 0.03 & 0.03 \\ 0.47 & 0.47 & 0.06 & -0.05 \end{bmatrix}. \quad (19)$$

We compute the cost of each hyperedge denoted by $l_{e_j}(\mathbf{x})$ in (16) using Fiedler vectors and tabulate in Table 1.

The sum of all the hyperedge cost or each column of Table 1 is 0.0569. It can be noticed that the hyperedges among densely connected have less cost as compared to others. For example, hyperedge $\{1, 2, 3\}$ has less cost as compared to the hyperedge $\{4, 5, 6\}$.

Such hyperedges with relatively smaller cost can be termed as "smooth" hyperedge because they contain nodes which are densely connected by other hyperedges. For example, nodes 6 and 8 in Fig. 3 are connected by 3 hyperedges. Ideally, the new hyperedge should be constructed among the nodes which are densely connected by other hyperedges. It is observed that the cost for such hyperedges is smaller compared to other hyperedges. So, we propose a hyperedge prediction algorithm which promotes the construction of hyperedges with minimal cost. The proposed algorithm is summarized in Table 2.

In this section, we proposed a novel hyperedge prediction algorithm using the spectral framework. In the next section, the working and efficacy of the proposed method are demonstrated using simple toy examples and real hypergraphs.

Table 1: Hyperedge cost for Fig. 3.

Hyperedges	v_1	v_2	v_3	v_4
{1, 2, 3}	0.0004	0.0004	0	0
{1, 2, 4}	0.0127	0.0111	0.0025	0.0025
{2, 3, 4}	0.0111	0.0127	0.0025	0.0025
{4, 5, 6}	0.0278	0.0278	0.0278	0.0278
{6, 7, 8}	0.0025	0.0025	0.0127	0.0111
{7, 8, 9}	0	0	0.0004	0.0004
{6, 8, 9}	0.0025	0.0025	0.0111	0.0127

4 Experiments

We consider simple hypergraphs with interesting structural properties to validate the functioning of the proposed algorithm in Sect. 5.1 and real hypergraphs later on.

4.1 Synthetic Hypergraph: Example 1

We consider a 3-uniform symmetrical hypergraph with nine nodes and seven hyperedges shown in Fig. 3. The task is to predict the best new set of hyperedges to be formed from all the potential set of hyperedges.

We construct the tensor Laplacian and compute the eigenvalues and eigenvectors, as stated in the proposed algorithm described in Table 2. We arrive at four Fiedler eigenvectors mentioned in (19) with same eigenvalue of 0.0569.

All the above eigenvectors are then used for computing the cost of each potential hyperedges. To predict the best set of new hyperedges, all the potential hyperedges are considered. As there are 9 nodes, one could have $\binom{9}{3} = 84$. Seven hyperedges are further removed as they already exist in the hypergraph, which leaves us with $84 - 7 = 77$ potential hyperedges. The cost for each of these potential 77 hyperedges is computed using each of the 4 eigenvectors mentioned in (19). The next step is to rank these potential hyperedges based on the increasing order of their formation cost.

It is observed that the cost computed from the first two eigenvectors [first two columns of $\mathbf{V}$ in (19)] are the same. The same holds for the other two eigenvectors. So, the preferential rank of new hyperedge formation from these two sets of eigenvectors is also same. Due to space

Table 2: Hyperedge prediction algorithm.

1.	Construct the unnormalized or normalized tensor Laplacian as shown in (9) or (10) respectively.
2.	Compute the Fiedler eigenpair $(\lambda_\star, \mathbf{v}_\star)$ using (14).
3.	For a given set of potential hyperedges E_p, compute the construction cost using (16) and the Fiedler eigenvector computed in previous step. The same can be stated as $c_l = \{l_{e_j}(\mathbf{v}_\star) \vert e_j \in E_p\}$.
4.	Prefer the construction of hyperedges with minimal construction cost.

Table 3: Cost of new hyperedges using unnormalized Laplacian.

Hyperedges_1	Hyperedges_2	Cost
{6, 7, 9}	{1, 3, 4}	0.0028
{5, 6, 8}	{2, 4, 5}	0.0139
{1, 3, 4}	{6, 7, 9}	0.0142
{5, 6, 7}	{1, 4, 5}	0.0152
{5, 6, 9}	{3, 4, 5}	0.0152
{5, 8, 9}	{1, 2, 5}	0.0195
{5, 7, 8}	{2, 3, 5}	0.0195
{5, 7, 9}	{1, 3, 5}	0.0205
{3, 4, 5}	{5, 6, 9}	0.0365
{1, 4, 5}	{3, 5, 6}	0.0379

Table 4: Cost of new hyperedges using normalized Laplacian.

Hyperedges_1	Hyperedges_2	Cost
{6, 7, 9}	{1, 3, 4}	3.3×10^{-4}
{2, 3, 5}	{5, 8, 9}	0.0142
{1, 2, 5}	{5, 7, 8}	0.0160
{1, 3, 5}	{5, 7, 9}	0.0172
{1, 3, 4}	{6, 7, 9}	0.0173
{3, 4, 5}	{5, 6, 9}	0.0197
{2, 4, 5}	{5, 6, 8}	0.0254
{4, 5, 7}	{1, 5, 6}	0.0375
{4, 5, 9}	{3, 5, 6}	0.0375
{1, 4, 5}	{5, 6, 7}	0.0386

constraints, only 10 hyperedges with minimal formation cost are mentioned in Table 3.

The cost of new hyperedges can also be calculated from the eigenvectors of normalized tensor Laplacian defined in (10). The same analysis can be performed using the normalized tensor Laplacian defined in to favor all nodes equally with respect to their degree distribution. The construction cost of new hyperedges using the eigenvectors of normalized tensor Laplacian is reported in Table 4.

Following observations can be made from Tables 3 and 4:

1. The most obvious hyperedge to be formed for this hypergraph is $\{1, 3, 4\}$ and $\{6, 7, 9\}$. This can also be seen as nodes 1, 2, 3, 4 are densely connected with other hyperedges. So, the only remaining hyperedge among the four possible hyperedges is $\{1, 3, 4\}$. The same study holds for the hyperedge $\{6, 7, 9\}$.

   The most probable hyperedges are predicted by the proposed algorithm as it has minimum construction cost mentioned in first row of Tables 3 and 4.

   This trivial task of predicting the most probable hyperedge helps to validate the functioning of the proposed algorithm.
2. It can be observed that the most probable hyperedge is $\{1, 3, 4\}$ and $\{6, 7, 9\}$ using unnormalized and normalized Laplacian. However, the second best hyperedge is different. The probable hyperedge for unnormalized Laplacian is $\{2, 4, 5\}$ while it is $\{2, 3, 5\}$ for the normalized case. In both cases, nodes 2 and 5 are present. Note that, node three is given more preference in the normalized case as compared to node 4. This behavior is expected because the significance of nodes with a smaller degree will enhance after normalization compared to the unnormalized case. Thus, this observation encourages the use of normalized Laplacian for hypergraphs having high variance in the degree distribution.[29] also establishes a similar preference for using normalized or unnormalized Laplacian in case of graphs.

4.1.1 Eigenvectors of Tensor vs. Matrix Representation

Most of the existing methods using matrix representation model the dyadic interaction among nodes only and further predict the hyperedges of cardinality greater than 2. To manifest the effectiveness of tensor eigenvectors, we propose a slight variation of the proposed algorithm.

One of the crucial steps of the proposed algorithm (in Table 2) is the computation of tensor eigenvectors which captures super-dyadic interactions. To demonstrate the importance of this step, we replace it with the computation of eigenvectors of graph Laplacian (matrix) derived from hypergraph reduction using (3). All the other steps in the algorithm remain the same.

The construction cost of new hyperedges derived from the Fiedler eigenvector of reduced hypergraph is shown in Table 5.

Table 5: Cost of new hyperedges using normalized Laplacian of reduced hypergraph.

Hyperedges_1	Cost
{1, 5, 7}	0
{1, 5, 9}	0
{2, 5, 8}	0
{3, 5, 7}	0
{3, 5, 9}	0
{2, 5, 9}	0.0038
{2, 5, 7}	0.0038
{1, 5, 8}	0.0038
{3, 5, 8}	0.0038
{6, 7, 9}	0.0217

It can be easily stated that the above results are not as expected and do not capture the interaction among three nodes. This can be justified theoretically as the Laplacian cost function is a second order homogeneous polynomial modeling dyadic interaction only whereas the tensor-based Laplacian cost function is a third order homogeneous polynomial capturing the super-dyadic interactions.

In this example, we investigated various features of the proposed algorithm such as

1. Deriving preferential order of new hyperedges.
2. Behaviour of normalized and unnormalized Laplacian.
3. Effectiveness of the tensor eigenvectors in capturing the super-dyadic interactions.

4.2 Real Hypergraphs

In this subsection, we analyze the performance of the proposed algorithm on real hypergraphs. We first describe the datasets, baselines, and then the experimental settings used to evaluate these hyperedge prediction baselines.

4.2.1 Datasets

We consider five datasets with varying number of nodes and hyperedges from different domain. We have mentioned the size of largest of connected component consisting hyperedges of cardinality 3 in Table 6. Please note that we have performed the experiments and shown results for 3-uniform hypergraphs for simplicity but the proposed approach can be applied to any k hypergraph.

These datasets were constructed in following manner:

1. uchoice Bakery: Nodes represent the items in bakery and hyperedges are constructed among the items bought together.
2. uchoice Walmart Dept: Nodes represent the "department" of an item in the shop and a hyperedge is constructed among the department whose items were co-bought.
3. Contact-primary school and contact-high school: Nodes are people in the corresponding school and hyperedges are constructed among the people if they interacted with each other in interval of 20 s. The interaction was recorded by a wearable sensor.
4. NDC-substances: The data is taken from US National Drug Code (NDC), where a hyperedge denotes a drug and the nodes represent the substances used in that drug.

We further briefly discuss the existing hyperedge prediction approaches.

4.2.2 Baselines

We consider some of the most widely used hyperedge prediction baselines in this subsection. Every method tries to construct a "similarity score" of the potential hyperedge by its model. A large similarity score of a potential hyperedge indicates that it is more likely to be formed as compared to potential hyperedges with low similarity score. This similarity score is used to as a

Table 6: Datasets.

Name	$\|V\|$	$\|E_3\|$	References
uchoice Bakery	50	24,674	[5]
uchoice Walmart Dept	66	24,365	[5]
Contact-primary school	242	9262	[26]
Contact-high school	317	7475	[19]
NDC-substances	570	6327	[4]

proxy to predict the new hyperedges. Hence, we discuss the approach in which each of the following baselines construct that similarity score:

1. Common neighbours (CN)[21]: For a potential hyperedge, the similarity score is the sum of number of common neighbours of two nodes taken at a time in the given hypergraph. It should be noted that similarity score is computed using the local information in this approach.
2. Katz[11]: The similarity score is computed based on the global information using the paths connecting the two nodes. For a hyperedge with m nodes, we consider all the possible $k(k-1)/2$ pairs of nodes.
3. HPRA[13]: This is a recently proposed algorithm which computes the similarity score by extending the use of resource allocation approach[18,35] from graphs to hypergraphs. This method also proposes a modified hypergraph reduction method which preserves the node degrees of original hypergraph in the resulting graph[14].

We further describe the experimental settings used in the evaluation of these methods.

4.2.3 Experimental Settings

The first step before applying any of the above methods is to construct a potential set of hyperedges. The naive approach of considering all possible hyperedges can not be used due to the large number of potential hyperedges unlike the case of small synthetic hypergraphs. We used that approach in Sect. 5.1 to show the functioning of the proposed method.

The first step is to remove a few existing hyperedges from the given hypergraph. A hyperedge prediction algorithm is then evaluated on the basis of predicting the removed hyperedges. A good algorithm should also not predict the non-existing hyperedges in original hypergraph. So, we construct our test set (or potential set) of hyperedges by considering both the removed and non-existing hyperedges. The number of removed hyperedges in our experiments is maintained as 10% of the existing number of hyperedges. The rest 90% of the hyperedges are used for training.

As the non-existing hyperedges are considered in test set, the choice of non-existing hyperedges plays a vital role in the evaluation of hyperedge prediction algorithms. We utilize recently proposed negative sampling approach[23] to construct the set non-existing hyperedges. The first step of this negative sampling approach is to reduce the hypergraph to a graph and then connect the neighbors of nodes in a randomly sampled edge to the chosen edge in order to construct the hyperedge. This process is repeated until the desired number of non-existing hyperedges are sampled, which we choose to be 3 times the number of existing hyperedges in the given hypergraph for training. Please note that this approach can finally provide a hyperedge that is already existing in the hypergraph, whereas our motive was to sample non-existing hyperedges. So in this work, we remove those hyperedges from this "non-existing set" of hyperedges which already existed in the original hypergraph in order to have a proper evaluation of hyperedge prediction algorithms.

4.2.4 Results

We run the experiments on each dataset at least 20 times, removing 10% of hyperedges randomly in each run. The quality of predicted hyperedges by any algorithm is compared by using average F1 score[31]. A higher average score indicates that the performance of the corresponding algorithm is better. The mean of average F1 scores computed from the 20 runs on each dataset are presented in Table 7.

In order to compare the performance of all algorithms, we also define the relative performance improvement (PI) as:

$$\text{PI} = \frac{\text{Avg-F1}_{\text{prop}} - \text{Avg-F1}_{\text{base}}}{\text{Avg-F1}_{\text{base}}} \times 100, \quad (20)$$

where $\text{Avg-F1}_{\text{prop}}$ denotes the average F1 score by proposed algorithm and $\text{Avg-F1}_{\text{base}}$ denotes the average F1 scores of the best baseline algorithm for the corresponding dataset. We compute PI score for each of the 20 runs separately and present the mean of those 20 runs in last column of Table 7. A positive PI score indicates that the proposed algorithm has performed better and its magnitude signifies the improvement.

Table 7: Results.

Dataset	CN	Katz	HPRA	Proposed	PI
uchoice Bakery	0.3445	0.3457	0.3459	0.3658	5.53
uchoice Walmart Dept	0.2592	0.2671	0.2623	0.3322	23.4
Contact-primary school	0.2150	0.2164	0.2242	0.2738	21.51
Contact-high school	0.2140	0.2163	0.2226	0.2794	24.19
NDC-substances	0.1743	0.1779	0.1808	0.2343	24.42

It is clearly evident that the proposed algorithm has outperformed the existing baselines by a considerable margin from Table 7. In this section, we discussed the performance of proposed method on real datasets. We make concluding remarks and provide directions for future work in the next section.

5 Conclusion and Future Work

In this article, we proposed a novel framework for hyperedge prediction for k-uniform hypergraphs. The critical challenge for this task was modeling complex interactions among multiple nodes. We utilized the tensor-based representation of hypergraphs, which helps to model the super-dyadic interactions among the nodes. The proposed algorithm prefers to construct the hyperedges with minimal construction cost. In the perspective of spectral hypergraph theory, this can also be perceived as the inclusion of new hyperedges such that there is minimal perturbation in the "smoothness" of the hypergraph. The functioning and fruitful merits of the proposed algorithm were demonstrated using synthetic and real hypergraphs. The future directions of this work are along the lines of performing a similar analysis for non-uniform and directed hypergraphs.

Publisher's Note

Acknowledgements

This work was partially supported by Intel research Grant RB/18-19/CSE/002/INTI/BRAV to BR.

Received: 15 February 2021 Accepted: 18 February 2021
Published online: 21 July 2021

References

1. Agarwal S, Branson K, Belongie S (2006) Higher order learning with graphs. In: Proceedings of the 23rd international conference on machine learning, pp 17–24. ACM
2. Banerjee A, Char A, Mondal B (2017) Spectra of general hypergraphs. Linear Algebra Appl 518:14–30
3. Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512
4. Benson AR, Abebe R, Schaub MT, Jadbabaie A, Kleinberg J (2018) Simplicial closure and higher-order link prediction. Proc Natl Acad Sci 115(48):E11221–E11230
5. Benson AR, Kumar R, Tomkins A (2018) A discrete choice model for subset selection. In: Proceedings of the eleventh ACM international conference on web search and data mining, pp 37–45
6. Brin S, Page L (1998) The anatomy of a large-scale hypertextual web search engine. Comput Netw ISDN Syst 30(1–7):107–117
7. Chodrow PS, Veldt N, Benson AR (2021) Hypergraph clustering: from blockmodels to modularity. arXiv preprint arXiv:2101.09611
8. Chung FR, Graham FC (1997) Spectral graph theory, vol 92. American Mathematical Society, Washington, DC
9. Ghoshdastidar D, Dukkipati A (2017) Uniform hypergraph partitioning: provable tensor methods and sampling techniques. J Mach Learn Res 18(1):1638–1678
10. Hu S, Qi L (2014) The eigenvectors associated with the zero eigenvalues of the Laplacian and signless Laplacian tensors of a uniform hypergraph. Discrete Appl Math 169:140–151
11. Katz L (1953) A new status index derived from sociometric analysis. Psychometrika 18(1):39–43
12. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 8:30–37
13. Kumar T, Darwin K, Parthasarathy S, Ravindran B (2020) Hpra: Hyperedge prediction using resource allocation. In: 12th ACM conference on web science, pp 135–143
14. Kumar T, Vaidyanathan S, Ananthapadmanabhan H, Parthasarathy S, Ravindran B (2020) Hypergraph clustering by iteratively reweighted modularity maximization. Appl Netw Sci 5(1):1–22
15. Li D, Xu Z, Li S, Sun X (2013) Link prediction in social networks based on hypergraph. In: Proceedings of the

22nd international conference on world wide web, pp 41–42. ACM
16. Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inf Sci Technol 58(7):1019–1031
17. Liu M, Veldt N, Song H, Li P, Gleich DF (2020) Strongly local hypergraph diffusions for clustering and semi-supervised learning. arXiv preprint arXiv:2011.07752
18. Lü L, Zhou T (2011) Link prediction in complex networks: a survey. Phys A Stat Mech Appl 390(6):1150–1170
19. Mastrandrea R, Fournet J, Barrat A (2015) Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys. PLoS One 10(9):e0136497
20. Maurya D, Ravindran B (2020) Hypergraph partitioning using tensor eigenvalue decomposition. arXiv preprint arXiv:2011.07683
21. Newman ME (2001) Clustering and preferential attachment in growing networks. Phys Rev E 64(2):025102
22. Oyetunde T, Zhang M, Chen Y, Tang Y, Lo C (2016) Boostgapfill: improving the fidelity of metabolic network reconstructions through integrated constraint and pattern-based methods. Bioinformatics 33(4):608–611
23. Patil P, Sharma G, Murty MN (2020) Negative sampling for hyperlink prediction in networks. In: Pacific-Asia conference on knowledge discovery and data mining, pp 607–619. Springer
24. Qi L, Luo Z (2017) Tensor analysis: spectral theory and special tensors, vol 151. SIAM, Philadelphia
25. Sharma A, Srivastava J, Chandra A (2014) Predicting multi-actor collaborations using hypergraphs. arXiv preprint arXiv:1401.6404
26. Stehlé J, Voirin N, Barrat A, Cattuto C, Isella L, Pinton JF, Quaggiotto M, Van den Broeck W, Régis C, Lina B et al (2011) High-resolution measurements of face-to-face contact patterns in a primary school. PLoS One 6(8):e23176
27. Symeonidis P, Mantas N (2013) Spectral clustering for link prediction in social networks with positive and negative links. Soc Netw Anal Min 3(4):1433–1447
28. Tarakci H, Kesim Cicekli N (2014) Using hypergraph-based user profile in a recommendation system. In: Proceedings of the international joint conference on knowledge discovery, knowledge engineering and knowledge management-vol 2, pp 27–35. SCITEPRESS-Science and Technology Publications, Lda
29. Von Luxburg U (2007) A tutorial on spectral clustering. Stat Comput 17(4):395–416
30. Yadati N, Nitin V, Nimishakavi M, Yadav P, Louis A, Talukdar PP (2018) Link prediction in hypergraphs using graph convolutional networks. https://openreview.net/forum?id=ryeaZhRqFm
31. Yang J, Leskovec J (2013) Overlapping community detection at scale: a nonnegative matrix factorization approach. In: Proceedings of the sixth ACM international conference on Web search and data mining 587–596
32. Zhang M, Cui Z, Jiang S, Chen Y (2018) Beyond link prediction: predicting hyperlinks in adjacency space. In: Thirty-second AAAI conference on artificial intelligence (2018)
33. Zhou D, Bousquet O, Lal TN, Weston J, Schölkopf B (2004) Learning with local and global consistency. In: Advances in neural information processing systems 321–328
34. Zhou D, Huang, J, Schölkopf B (2007) Learning with hypergraphs: clustering, classification, and embedding. In: Advances in neural information processing systems 1601–1608
35. Zhou T, Lü L, Zhang YC (2009) Predicting missing links via local information. Eur Phys J B 71(4):623–630

Deepak Maurya is an MS scholar at the Department of Computer Science and Engineering, Indian Institute of Technology Madras, where he is jointly advised by Prof. Balaraman Ravindran and Prof. Shankar Narasimhan. His Master's work is focused on applications of spectral hypergraph theory. Before starting his Master's, he finished his Dual Degree (B.Tech + M.Tech) in Electrical Engineering, Indian Institute of Technology Madras. He has also worked on system identification.

Professor B. Ravindran heads the Robert Bosch Centre for Data Science & Artificial Intelligence (RBCDSAI) at IIT Madras, the leading interdisciplinary AI research centre in India. He is the Mindtree Faculty Fellow and Professor in the Department of Computer Science and Engineering at IIT Madras. Currently, his research interests are centred on learning from and through interactions and span the areas of geometric deep learning and reinforcement learning. He is currently serving on the editorial boards of Journal of AI Research, PLOS One, and Frontiers in Big Data and AI. He has published nearly 100 papers in premier journals and conferences such as ICML, AAAI, IJCAI, ICDM, ICLR, NeurIPS, UAI, ISMB, and AAMAS. He has also co-authored the chapter on reinforcement learning in the Handbook of Neural Computation published by Oxford University Press. He has been on the program and organizing committees of several premier conferences. His work with students have won multiple best paper awards, the most recent being a best-paper runner-up at AAMAS 2020.He received his PhD from the University of Massachusetts, Amherst and his Master's degree from Indian Institute of Science, Bangalore. He is a senior member of AAAI.