Co-Clustering of High Order Relational Data Using Spectral Hypergraph Partitioning[*]

Jie Chen[†] Yousef Saad[†]

January 14, 2009

Abstract

High order relational data (i.e. multi-type data), such as audience-movies-casts and authors-publications-journals, are ubiquitous in everyday life. Mining the structure of such data is an important tool to help reveal cluster information on each contributing entity of the data. This paper addresses this problem by using hypergraphs to model the data. A hypergraph describes interrelationships among various types of the data entities simultaneously. It is a generalization of graphs that model only pairwise relationships. A spectral theory of hypergraphs is discussed, and a simultaneous clustering technique is advocated based on spectral hypergraph partitioning. This co-clustering technique can be considered a higher order generalization of that obtained from bipartite graphs with the help of the SVD. Experiments show that the proposed co-clustering technique effectively unravels the structure of the data tensor, and can also improve on other data mining techniques, e.g., those in collaborative filtering.

1 Introduction

The co-clustering of heterogeneous data (multi-type data, high order relational data), is an important task in various data mining applications. Traditional clustering methods partition a data set in subsets according to the general principle that data within the same subset should be as similar as possible, while data belonging to different subsets should be as dissimilar as possible. If the data has numerical feature representations, the clustering task is equivalent to partitioning the rows of the data matrix, where each row represents a data record. On the other hand, if the feature representation of the data is unknown, but a relational graph among them can be conveniently constructed, or if it is naturally available, we can partition the graph to achieve a clustering of the data.

Traditional clustering tasks consider only homogeneous data which are of a single type. In practice, it is not uncommon that the data is heterogeneous, meaning that it involves several interrelated types. Real-life examples include documents-words in the text-mining scenario, where a word is related to a document by the number

[*]This work was supported by NSF under grant DMS-0810938 and by the Minnesota Supercomputing Institute.

[†]Department of Computer Science and Engineering, University of Minnesota at Twin Cities, MN 55455. Email: {jchen, saad}@cs.umn.edu.

of its occurrences in the document, and audience-movies-casts in the film rating (collaborative filtering) scenario, where an audience gives a rating to a film that is cast by several actors/actresses. The task of co-clustering is to simultaneously cluster the different types of entities. It is possible to cluster each type separately, but this approach would miss the potential leveraging that could be obtained from the interrelationships among different types.

The co-clustering of bi-type heterogeneous data has been widely investigated in the context of matrix approximation [12, 7, 3, 1], where the rows and the columns represent two types of entities, with matrix entries representing the relationships (proximities) between them. Such methods view the data matrix as a probability distribution, and the objective is to optimally approximate the matrix while preserving the cluster sums and/or other quantities such as marginals. The matrix representation of bi-type data has also been interpreted as a bipartite graph [11, 31, 26], where the rows represent a partite set and the columns the other. Hence, the co-clustering problem becomes a graph cut problem, and spectral graph partitioning techniques can be neatly applied to simultaneously cluster the two partite sets.

In recent years, co-clustering of data with more than two types of entities has attracted increasing attention. A natural generalization of bi-type methods for multi-type data is to consider the pairwise relationships between every two types [4, 22, 23]. The model behind this is a k-partite graph representation [16, 21], and traditional graph partitioning techniques are applied. Meanwhile, ideas that consider the interrelationships among all the entity types have been explored in [2, 6]. In this context, the data is modeled as a tensor which is a high order generalization of matrices. Each mode of the tensor represents a type of entities, and tensor entries encode the relations among entities of all the types. Banerjee *et al.* [2] extended the matrix approximation framework [3] to tensors and proposed a probabilistic approach for tensor approximation by minimizing the Bregman divergence. Chen *et al.* [6] modeled the data tensor as a graph via the clique expansion method [18] and cast the co-clustering problem as a graph cut problem.

In this paper, we interpret the data tensor as a hypergraph and propose a co-clustering method based on spectral hypergraph partitioning. For data analysis, the concept of *hypergraphs* was initially proposed in [32] and found applications in [30, 20]. We discuss a spectral theory of hypergraphs from the angle of hypergraph Laplacians, in parallel the spectral graph theory [8]. We also generalize several widely used spectral graph partitioning techniques (ratio cut [19], normalized cut [29], min-max cut [13]) to hypergraphs under a unified framework. Since a data tensor is associated with a hypergraph, the co-clustering of a tensor can be naturally performed by hypergraph partitioning.

The following contributions are made in this paper: (a) The (normalized) Laplacian matrix is defined, based on which a spectral theory of hypergraphs is developed. (b) Several spectral graph partitioning techniques are generalized for hypergraphs in a unified framework, under which the Rayleigh quotient plays an important role. (c) For applications, a co-clustering method based on the hypergraph model is proposed. This method can be considered a high-order generalization of the co-clustering method [11, 31] of bi-type data, and is much more efficient than high-order tensor approximation methods [10, 9].

The rest of the paper is organized as follows. Section 2 presents the spectral theory of hypergraphs based on the Laplacian matrix, and Section 3 develops the spectral hypergraph partitioning techniques. In Section 4 we propose a co-clustering technique using the hypergraph model. Several experiments are reported in Section 5 to show the effectiveness of the proposed technique, and remarks are given in Section 6.

2 Hypergraph Laplacian

A *hypergraph* $G = (V, E)$ consists of a *vertex* set V and a *hyperedge* set E, where each hyperedge $e \in E$ is a non-empty subset of V. In other words, $E \subset 2^V$. An undirected graph is a special case of hypergraphs, since each edge contains exactly two vertices, i.e., $|e| = 2$. A *weighted hypergraph* $G = (V, E, w)$ is a hypergraph in which a nonnegative weight $w(e)$ is associated with every hyperedge e. Figure 1(a) shows an example of a weighted hypergraph.

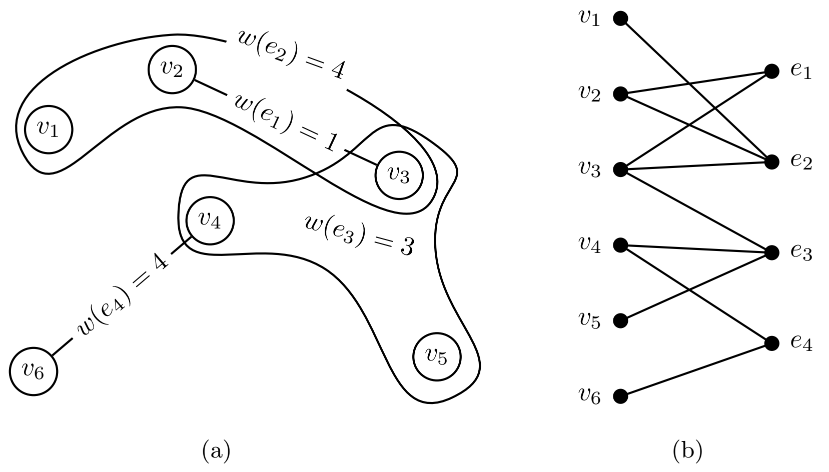


Figure 1: An example weighted hypergraph and its bipartite graph interpretation.

A hyperedge $e \in E$ is *incident* to a vertex $v \in V$ if v is an element of e. The *(weighted) degree of a vertex* v is the sum of the weights of hyperedges incident to v:

$$d(v) := \sum_{v \in e, e \in E} w(e), \tag{1}$$

and the diagonal matrix D where $D(v, v) = d(v)$ for all $v \in V$ is usually called the *degree matrix*. The *degree of a hyperedge* e is the number of vertices it contains:

$$\delta(e) := |e|. \tag{2}$$

A *path* in a hypergraph is a sequence of vertices $v_1, v_2, \ldots, v_n$ such that there exists a hyperedge incident to both v_i and v_{i+1} for $i = 1, 2, \ldots, n-1$. A hypergraph is *connected* if there is a path between every pair of vertices. A hypergraph $G'(V', E')$ is called a *sub-hypergraph* of $G(V, E)$ if $V' \subset V$ and $E' \subset E$. A *connected component* of a hypergraph G is a maximal connected sub-hypergraph of G.

In what follows, the spectral properties of a hypergraph will rely largely on a $|V| \times |V|$ matrix Ψ, whose elements are defined as

$$\psi(u,v) := \begin{cases} 0 & \text{if } \nexists e \in E \text{ s.t. } \{u,v\} \subset e, \\ \sum_{\{u,v\}\subset e, e\in E} w(e)/\delta(e) & \text{otherwise.} \end{cases} \tag{3}$$

Note that this definition includes the case $u = v$. Intuitively, $\psi(u,v)$ is a "weight" between a pair of vertices u and v. Thus, it is the sum of the weights of hyperedges incident to both u and v, with each weight amortized by the degree of the hyperedge. A property is that for Ψ, the column/row sum is equal to the degree of the vertex that column/row corresponds to:

$$\sum_{u\in V} \psi(u,v) \equiv \sum_{u\in V} \psi(v,u) = d(v). \tag{4}$$

In a way Ψ is simlar to the weighted adjacency matrix (affinity matrix) A of a graph. Nevertheless, there are some significant differences. Thus, the diagonal of Ψ is always nonzero (unless the hypergraph contains no hyperedges). Indeed, when a graph is interpreted as a hypergraph,

$$\Psi = \frac{D+A}{2}. \tag{5}$$

The *Laplacian* of the hypergraph is defined as

$$L := D - \Psi, \tag{6}$$

and the *normalized Laplacian* is

$$\Delta := D^{-1/2} L D^{-1/2} = I - D^{-1/2} \Psi D^{-1/2}, \tag{7}$$

where I is the identity matrix. They have properties that reveal the connectivity of the hypergraph, in parallel those of the (normalized) Laplacians of graphs.

Theorem 1. *The Laplacian L and normalized Laplacian Δ of a hypergraph have the following properties:*

1. *Both L and Δ are symmetric positive semi-definite.*

2. *The smallest eigenvalue(s) of L (and Δ) is 0.*

3. *The multiplicity of the eigenvalue 0 of L (and Δ) is one if and only if the hypergraph is connected. In such a case, the corresponding eigenvector of L is $\mathbf{1}$ (the vector of all ones), and that of Δ is the diagonal of $D^{1/2}$.*

4. *The multiplicity of the eigenvalue 0 of L (and Δ) is k if and only if the hypergraph has k connected components.*

5. *Let L_{graph} (resp. Δ_{graph}) denote the Laplacian (resp. normalized Laplacian) of a graph G, and let $L_{hypergraph}$ (resp. $\Delta_{hypergraph}$) denote the hypergraph Laplacian (resp. normalized Laplacian) when G is interpreted as a hypergraph, then*

$$L_{hypergraph} = \frac{1}{2} L_{graph} \qquad \text{and} \qquad \Delta_{hypergraph} = \frac{1}{2}\Delta_{graph}.$$

Proof. 1. By the definition of L, we have for any vector x,

$$x^T Lx = \frac{1}{2} \sum_{u,v \in V} \psi(u,v)(x(u) - x(v))^2 \geq 0. \tag{8}$$

Hence L is symmetric positive semi-definite, and so is Δ.

2. By (4) and (6), we have $L\mathbf{1} = 0$. Since L is positive semi-definite, 0 is the smallest eigenvalue. The argument also holds for Δ since $\Delta(D^{1/2}\mathbf{1}) = 0$. Indeed, for the special eigenvalue 0, the eigenvector(s) z of L and eigenvector(s) z' of Δ are related by $z' = D^{1/2}z$.

3. The equality $x^T Lx = 0$ holds iff for each pair of vertices u and v, either $\psi(u,v) = 0$ or $x(u) = x(v)$. For an initial vertex u, and all the vertices v such that $\psi(u,v) \neq 0$ (i.e., there is a hyperedge incident to both u and v), we have $x(v) = x(u)$. A hypergraph "traversal" procedure similar to the breadth first search of a graph will make $x(v) = x(u)$ for all $v \in V$, if the hypergraph is connected.

   On the other hand, if the hypergraph has $k > 1$ connected components, where the i-th component corresponds to the vertex subset V_i, then there are k mutually orthogonal vectors z_i, where

$$z_i(v) = \begin{cases} 1 & \text{if } v \in V_i \\ 0 & \text{if } v \notin V_i, \end{cases} \tag{9}$$

   satisfying $Lz_i = 0$.

4. Let the vertex set V be partitioned into subsets V_i, $i = 1, \ldots, k$, where a pair of vertices are inside the same partition if and only if there is a path between the two vertices. Then $x^T Lx$ can be written as

$$x^T Lx = \frac{1}{2} \sum_{i=1}^{k} \sum_{u,v \in V_i} \psi(u,v)(x(u) - x(v))^2.$$

   A same previous argument shows that for $x \neq 0$, $x^T Lx = 0$ iff $x(u) = \xi_i$ for some $\xi_i \in \mathbb{R} - \{0\}$ and all $u \in V_i$, $i = 1, \ldots, k$. Such vectors x lie on the space spanned by the vectors $z_1, \ldots, z_k$ in the form (9). This subspace has dimension k.

5. The two equalities immediately follow from (5).

□

A weighted hypergraph $G(V, E, w)$ can be interpreted as a weighted bipartite graph G_b (cf. Figure 1(b)), where V and E are the two partite sets, and there is an edge between $v \in V$ and $e \in E$ if e is incident to v in the hypergraph. The edge weights are defined as

$$h(v,e) := w(e)/\delta(e) \qquad \text{for } v \in e, \tag{10}$$

which are the elements of the matrix $H \in \mathbb{R}^{|V| \times |E|}$ ($h(v,e) = 0$ if $v \notin e$).

Therefore, the weighted adjacency matrix of G_b is

$$A_b = \begin{bmatrix} 0 & H \\ H^T & 0 \end{bmatrix}. \tag{11}$$

Note that the matrix Ψ, which simulates the "adjacency matrix" of the hypergraph, can also be expressed in terms of H:

$$\Psi = H\Omega^{-1/2}\Lambda\Omega^{-1/2}H^T, \tag{12}$$

where both $\Omega \in \mathbb{R}^{|E|\times|E|}$ and $\Lambda \in \mathbb{R}^{|E|\times|E|}$ are diagonal, with diagonal elements equal to $w(e)$ and $\delta(e)$, respectively. The matrix Ψ can be considered the product of H and H^T, up to an internal scaling. From this view point, the eigenvalue decompositions of A_b and Ψ are almost equivalent. A consequence is that the spectral properties of a hypergraph and those of its bipartite graph interpretation are closely related. The following theorem states a precise relationship between the normalized Laplacian Δ of G and that (Δ_b) of G_b for a special case.

Theorem 2. *Let $\hat{H} = \Phi^{-1/2}H\Omega^{-1/2}$, where both $\Phi \in \mathbb{R}^{|V|\times|V|}$ and $\Omega \in \mathbb{R}^{|E|\times|E|}$ are diagonal, with diagonal elements equal to the row sums $(\psi(v,v))$ and the column sums $(w(e))$ of H, respectively. If the hyperedge degrees are the same, then:*

1. *When $\mu \neq 0$, Δ has an eigenvector z corresponding to eigenvalue $1-\mu$ if and only if Δ_b has an eigenvector $\left[\begin{smallmatrix} z \\ \hat{H}^T z/\sqrt{\mu} \end{smallmatrix}\right]$ corresponding to eigenvalue $1-\sqrt{\mu}$ and an eigenvector $\left[\begin{smallmatrix} z \\ -\hat{H}^T z/\sqrt{\mu} \end{smallmatrix}\right]$ corresponding to eigenvalue $1+\sqrt{\mu}$.*
2. *Δ has an eigenvector z corresponding to eigenvalue 1 if and only if Δ_b has an eigenvector $\left[\begin{smallmatrix} y \\ \hat{H}^T z \end{smallmatrix}\right]$ corresponding to eigenvalue 1 for some y satisfying $\hat{H}^T y = 0$.*

Proof. Let the hyperedge degrees $\delta(e) = c$ for all e. Then, $\Delta_b = I - \left[\begin{smallmatrix} 0 & \hat{H} \\ \hat{H}^T & 0 \end{smallmatrix}\right]$ and

$$\begin{aligned} \Delta &= I - D^{-1/2}H\Omega^{-1/2}\Lambda\Omega^{-1/2}H^T D^{-1/2} \\ &= I - (c\Phi)^{-1/2}H\Omega^{-1/2}(cI)\Omega^{-1/2}H^T(c\Phi)^{-1/2} \\ &= I - \hat{H}\hat{H}^T. \end{aligned}$$

Therefore, when $\mu \neq 0$,

$$\hat{H}\hat{H}^T z = \mu z \quad \Longleftrightarrow \quad \begin{bmatrix} 0 & \hat{H} \\ \hat{H}^T & 0 \end{bmatrix} \begin{bmatrix} z \\ \pm\hat{H}^T z/\sqrt{\mu} \end{bmatrix} = \pm\sqrt{\mu} \begin{bmatrix} z \\ \pm\hat{H}^T z/\sqrt{\mu} \end{bmatrix}.$$

Now consider the case $\mu = 0$. If $\hat{H}\hat{H}^T z = 0$ for some $z \neq 0$, $\hat{H}$ does not have full row-rank, hence there exists a $y \neq 0$ such that $\hat{H}^T y = 0$. Thus $\left[\begin{smallmatrix} 0 & \hat{H} \\ \hat{H}^T & 0 \end{smallmatrix}\right] \left[\begin{smallmatrix} y \\ \hat{H}^T z \end{smallmatrix}\right] = 0$. The other direction is trivial. □

The requirement that the hyperedge degrees are the same can be satisfied in a special case where the hypergraph is induced from a data tensor; see Section 4.

3 Spectral Hypergraph Partitioning

We first consider partitioning the vertex set V into two disjoint subsets S and S^c:

$$\begin{cases} S \cap S^c = \emptyset, \\ S \cup S^c = V. \end{cases} \tag{13}$$

The case of partitioning into k disjoint subsets will be discussed in Section 3.2. In different areas, the terminology "cut" is used to mean "partitioning". We will use these two words interchangeably in this paper. Recall that for graphs, some optimal cuts are defined such that the cut size, balanced by the sizes of the two partitions, are minimized. In order to generalize such cuts for hypergraphs, we need to define some related concepts for "sizes".

Given a cut as in (13), the *boundary* of the cut, denoted as ∂S, is the set of hyperedges that cross the two partitions, i.e.,

$$\partial S := \{e \in E \mid e \cap S \neq \emptyset, e \cap S^c \neq \emptyset\}. \tag{14}$$

We define the *volume of the boundary* as

$$\operatorname{vol}(\partial S) := \sum_{u \in S} \sum_{v \in S^c} \psi(u, v). \tag{15}$$

This definition is consistent with that of the cut size of a graph. In the graph case, $\operatorname{vol}(\partial S)$ reduces to $\frac{1}{2}\sum_{e \in \partial S} w(e)$, which is just half of the cut size defined for a graph.

We define a general objective function

$$c(S) := \frac{\operatorname{vol}(\partial S)}{\operatorname{weight}(S)} + \frac{\operatorname{vol}(\partial S)}{\operatorname{weight}(S^c)}, \tag{16}$$

where $\operatorname{weight}(\cdot)$ is a generic weight function[1] defined for a vertex $v \in V$, and thus for vertex subsets such as S:

$$\operatorname{weight}(S) := \sum_{v \in S} \operatorname{weight}(v). \tag{17}$$

The objective function (16) characterizes how an optimal partitioning of a given hypergraph should look like: the volume of the boundary $\operatorname{vol}(\partial S)$ is minimized, while the "sizes" of S and S^c are balanced; otherwise, a small $\operatorname{weight}(S)$ or $\operatorname{weight}(S^c)$ will make $c(S)$ prohibitively large. The generic function $\operatorname{weight}(\cdot)$ can be defined in a number of ways. Remarkably, however it is defined, the objective value $c(S)$ coincides with a Rayleigh quotient.

Theorem 3. *Let a column vector q have elements*

$$q(v) := \begin{cases} +\sqrt{\eta_2/\eta_1} & \text{if } v \in S, \\ -\sqrt{\eta_1/\eta_2} & \text{if } v \in S^c, \end{cases} \tag{18}$$

[1]One shall not confuse the "weight" $\operatorname{weight}(v)$ of a vertex v with the "weighted degree" $d(v)$ of the vertex, or the "weight" $w(e)$ of a hyperedge e.

where $\eta_1 = \text{weight}(S)$ *and* $\eta_2 = \text{weight}(S^c)$, *then*

$$c(S) = \frac{q^T L q}{q^T W q}, \tag{19}$$

where W *is the diagonal matrix with diagonal elements equal to* weight(v)*'s. We call* q *the* partition vector.

Proof. Directly computing the two parts of the quotient, we have

$$\begin{aligned} q^T(D-\Psi)q &= \frac{\eta_2}{\eta_1}\left(\sum_{u\in S} d(u) - \sum_{u,u'\in S}\psi(u,u')\right) + 2\sum_{u\in S, v\in S^c}\psi(u,v) \\ &\quad + \frac{\eta_1}{\eta_2}\left(\sum_{v\in S^c} d(v) - \sum_{v,v'\in S^c}\psi(v,v')\right) \\ &= \frac{\eta_2}{\eta_1}\sum_{u\in S, v\in S^c}\psi(u,v) + 2\sum_{u\in S, v\in S^c}\psi(u,v) + \frac{\eta_1}{\eta_2}\sum_{u\in S, v\in S^c}\psi(u,v) \\ &= \frac{(\eta_1+\eta_2)^2}{\eta_1\eta_2}\sum_{u\in S, v\in S^c}\psi(u,v), \end{aligned}$$

and

$$q^T W q = \frac{\eta_2}{\eta_1}\sum_{u\in S}\text{weight}(u) + \frac{\eta_1}{\eta_2}\sum_{v\in S^c}\text{weight}(v) = \eta_1 + \eta_2.$$

Hence

$$\frac{q^T L q}{q^T W q} = \left(\frac{1}{\eta_1} + \frac{1}{\eta_2}\right)\sum_{u\in S, v\in S^c}\psi(u,v) = c(S).$$

□

With Theorem 3, we now define several optimal cuts for hypergraphs.

Definition 1. The *ratio hypergraph cut* is one that minimizes the objective function $c(S)$ with weight$(v) = 1$, i.e., the objective function is

$$c_{\mathrm{r}}(S) := \frac{\mathrm{vol}(\partial S)}{|S|} + \frac{\mathrm{vol}(\partial S)}{|S^c|}. \tag{20}$$

The above definition implies that the matrix W in Theorem 3 is simply the identity matrix I. Hence,

$$c_{\mathrm{r}}(S) = \frac{q^T L q}{q^T q}.$$

Definition 2. The *normalized hypergraph cut* is one that minimizes the objective function $c(S)$ with weight$(v) = d(v)$, i.e., the objective function is

$$c_{\mathrm{n}}(S) := \frac{\mathrm{vol}(\partial S)}{\sum_{u\in S} d(u)} + \frac{\mathrm{vol}(\partial S)}{\sum_{v\in S^c} d(v)}. \tag{21}$$

The above definition implies that the matrix W in Theorem 3 is the matrix D. Hence,

$$c_{\mathrm{n}}(S) = \frac{q^T Lq}{q^T Dq}.$$

Definition 3. The *min-max hypergraph cut* is one that minimizes the objective function $c(S)$ with $\text{weight}(v) = \sum_{v' \in P} \psi(v', v) = d(v) - \sum_{u \notin P} \psi(u, v)$ for $v \in P$, i.e., the objective function is

$$c_{\mathrm{m}}(S) := \frac{\mathrm{vol}(\partial S)}{[\sum_{u \in S} d(u)] - \mathrm{vol}(\partial S)} + \frac{\mathrm{vol}(\partial S)}{[\sum_{v \in S^c} d(v)] - \mathrm{vol}(\partial S)}. \tag{22}$$

3.1 Spectral Relaxation

Theorem 3 implies that the problem of finding an optimal hypergraph cut can be reduced to computing a vector q in the form (18) which minimizes the quotient (19). However, this is a combinatorial optimization problem that is NP-complete [17]. From standard results in linear algebra, minimizing the quotient $q^T Lq/q^T Wq$ over real vectors q, is equivalent to finding the bottom eigenvector of the matrix pencil (L, W). Hence, we apply this heuristic and relax the optimization problem to the following:

$$\min_{q \in \mathbb{R}^{|V|}} \frac{q^T Lq}{q^T Wq}, \tag{23}$$

whose solution is used to provide an approximate optimal partitioning.

According to the properties of the Laplacian (cf. Theorem 1), under the assumption that the hypergraph is connected, the smallest eigenvalue of L is zero, with corresponding eigenvector $\mathbf{1}$. This vector minimizes (23), but it provides no information on how to perform the partitioning. Hence, we resort to the second smallest eigenvector z of the matrix pencil (L, W), and use z to determine an approximate optimal partitioning. In the graph case, if W is the identity, then z is the second smallest eigenvector of the graph Laplacian, and is called the *Fiedler vector* [14, 15].

The simplest way to partition the hypergraph is according to the signs of the elements of z [32]:

$$S = \{v \in V \mid z(v) \geq 0\} \quad \text{and} \quad S^c = \{v \in V \mid z(v) < 0\}.$$

If this results in very unbalanced subsets, another criterion may be to exploit the median of z to separate the set:

$$S = \{v \in V \mid z(v) \geq \text{median}\} \quad \text{and} \quad S^c = \{v \in V \mid z(v) < \text{median}\}.$$

Perhaps a more sophisticated approach is to replace the median in the above criterion by an optimal "cut-point", which yields

$$S = \{v \in V \mid z(v) \geq \text{cut-point}\} \quad \text{and} \quad S^c = \{v \in V \mid z(v) < \text{cut-point}\}.$$

The cut-point is chosen from the set of candidates $\{z(v) \mid v \in V\}$ that minimizes the original objective function [13]. Finally, taking a dimensionality reduction viewpoint, we can consider the elements of z as a one-dimensional embedding of the vertices, and run a 2-means algorithm on z to obtain a clustering [11].

3.2 Generalization to k-way Partitioning

For a k-way partitioning of the hypergraph:

$$\begin{cases} S_1,\, S_2,\, \ldots,\, S_k \text{ mutually disjoint}, \\ S_1 \cup S_2 \cup \cdots \cup S_k = V, \end{cases} \tag{24}$$

we define the notion of the *boundary between S_i and S_i^c*

$$\partial S_i = \{e \in E \mid e \cap S_i \neq \emptyset, e \cap S_i^c \neq \emptyset\},$$

and the *volume of this boundary*

$$\mathrm{vol}(\partial S_i) = \sum_{u \in S_i} \sum_{v \in S_i^c} \psi(u, v).$$

The above two expressions are essentially generalizations of (14) and (15) to the case of more than two partitions.

Hence, the cost function (16) for a 2-way partitioning is naturally generalized for the k-way case:

$$c(S_1, \ldots, S_k) := \sum_{i=1}^{k} \frac{\mathrm{vol}(\partial S_i)}{\mathrm{weight}(S_i)}, \tag{25}$$

and a similar result to Theorem 3 can be established.

Theorem 4. *Let a matrix $Q \in \mathbb{R}^{|V| \times k}$ be defined as*

$$Q(v, i) := \begin{cases} 1/\sqrt{\eta_i} & \text{if } v \in S_i, \\ 0 & \text{else}, \end{cases} \tag{26}$$

where $\eta_i = \mathrm{weight}(S_i)$, then

$$c(S_1, \ldots, S_k) = \mathrm{tr}(Q^T L Q) \quad \text{and} \quad Q^T W Q = I, \tag{27}$$

where W is the diagonal matrix with diagonal elements equal to $\mathrm{weight}(v)$*'s.*

The proof of Theorem 4 is straightforward, hence omitted here. This theorem indicates that minimizing the objective function $c(S_1, \ldots, S_k)$ is equivalent to finding a matrix Q in the form (26) that minimizes the trace of $Q^T L Q$.

Similar to the 2-way case, this optimization problem can be relaxed to

$$\min_{Q^T W Q = I} \mathrm{tr}(Q^T L Q), \tag{28}$$

which has a unique solution (denoted as Z) up to any orthogonal transformation, where Z is formed by the bottom k eigenvectors of the matrix pencil (L, W). A popular practice, derived from the graph case [24], is to consider each row of Z as a data item, and to run a k-means algorithm on this set of items to obtain a k-clustering (k-way partitioning of the hypergraph).

The above solution Z has a natural interpretation in terms of manifold learning. Imagine that there is a set of high dimensional data points, which are sampled from a manifold and are associated with a relational hypergraph. We perform dimensionality reduction on these data samples using the (hypergraph) Laplacian eigenmaps method [5]. Then Z is the low dimensional embedding of the original manifold.[2] A good k-clustering of Z in some sense implies a reasonable clustering of the original high dimensional data, hence a suitable partitioning of the associated hypergraph.

4 Tensor Co-Clustering

A *tensor* is a multidimensional array of data whose elements are referred by using multiple indices, each of which represents a *mode* of the tensor. The number of indices required is called the *order* of the tensor. Hence, a matrix (order-2 tensor) is a special case of tensors since it uses two indices; the first and second modes correspond to matrix rows and columns, respectively.

Similar to the matrix-bipartite-graph analogy, an order-n tensor can be modeled as a hypergraph where each hyperedge contains exactly n vertices. For now we restrict our discussion to the case $n = 3$ for simplicity. To be specific, let a tensor $w \in \mathbb{R}^{|V_1|\times|V_2|\times|V_3|}$ have elements $w(v_1, v_2, v_3)$, where $v_1 \in V_1$, $v_2 \in V_2$, and $v_3 \in V_3$. The set V, as the union of disjoint sets V_1, V_2 and V_3, is the vertex set, and every hyperedge connects exactly one vertex from each of V_1, V_2, and V_3. If all the $w(v_1, v_2, v_3)$ entries are nonnegative, they can be considered weights of the corresponding hyperedges. Formally, an order-3 nonnegative tensor $w \in \mathbb{R}^{|V_1|\times|V_2|\times|V_3|}$ induces a special weighted hypergraph $G(V, E, w)$ for some disjoint "virtual" sets V_1, V_2 and V_3, where $V = V_1 \cup V_2 \cup V_3$, $E = V_1 \times V_2 \times V_3$, and $w(v_1, v_2, v_3)$ is the weight for a hyperedge $e = \{v_1 \in V_1, v_2 \in V_2, v_3 \in V_3\}$.

This hypergraph model of the data tensor shares similarities with the tri-partite graph model [21], which also uses V as the vertex set, with each V_i as a partite set. However, the hypergraph has hyperedges connecting exactly one vertex from each V_i, with each hyperedge having a single weight, while the tri-partite graph has edges connecting vertices from only two partite sets. In other words, the hypergraph models interrelationships among all the vertex subsets V_i's, while the tri-partite graph models only pairwise relationships between the V_i's. Hence, one can view hypergraphs as natural and convenient models for data tensors.

Extending the idea of co-clustering the rows and columns of a data matrix which is modeled as a bipartite graph [11, 31], we can simultaneously cluster all the modes of a data tensor which is modeled as a hypergraph.

4.1 Structure of the Laplacian

Due to the special structure of the hypergraph (induced from a data tensor), its Laplacian is also structured. By splitting D and Ψ into blocks according to the

[2] A subtle difference is that the method of Laplacian eigenmaps proposed in [5] does not use the eigenvector corresponding to the smallest eigenvalue 0.

vertex subsets V_1, V_2 and V_3, the Laplacian reads

$$L = D - \Psi = \begin{bmatrix} D_1 - \Psi_{11} & -\Psi_{12} & -\Psi_{13} \\ -\Psi_{21} & D_2 - \Psi_{22} & -\Psi_{23} \\ -\Psi_{31} & -\Psi_{32} & D_3 - \Psi_{33} \end{bmatrix}, \tag{29}$$

where the Ψ_{ij} blocks have elements

$$\begin{aligned}
\psi_{11}(v_1, v_1) &= \tfrac{1}{3} \textstyle\sum_{v_2 \in V_2, v_3 \in V_3} w(v_1, v_2, v_3), \\
\psi_{22}(v_2, v_2) &= \tfrac{1}{3} \textstyle\sum_{v_1 \in V_1, v_3 \in V_3} w(v_1, v_2, v_3), \\
\psi_{33}(v_3, v_3) &= \tfrac{1}{3} \textstyle\sum_{v_1 \in V_1, v_2 \in V_2} w(v_1, v_2, v_3), \\
\psi_{12}(v_1, v_2) &= \psi_{21}(v_2, v_1) = \tfrac{1}{3} \textstyle\sum_{v_3 \in V_3} w(v_1, v_2, v_3), \\
\psi_{13}(v_1, v_3) &= \psi_{31}(v_3, v_1) = \tfrac{1}{3} \textstyle\sum_{v_2 \in V_2} w(v_1, v_2, v_3), \\
\psi_{23}(v_2, v_3) &= \psi_{32}(v_3, v_2) = \tfrac{1}{3} \textstyle\sum_{v_1 \in V_1} w(v_1, v_2, v_3),
\end{aligned} \tag{30}$$

and D_1, D_2, D_3 are all diagonal matrices with diagonal elements

$$\begin{aligned}
d_1(v_1) &= \textstyle\sum_{v_2 \in V_2, v_3 \in V_3} w(v_1, v_2, v_3), \\
d_2(v_2) &= \textstyle\sum_{v_1 \in V_1, v_3 \in V_3} w(v_1, v_2, v_3), \\
d_3(v_3) &= \textstyle\sum_{v_1 \in V_1, v_2 \in V_2} w(v_1, v_2, v_3).
\end{aligned} \tag{31}$$

Hence, the diagonal blocks of L are all diagonal:

$$D_1 - \Psi_{11} = \tfrac{2}{3} D_1, \quad D_2 - \Psi_{22} = \tfrac{2}{3} D_2, \quad D_3 - \Psi_{33} = \tfrac{2}{3} D_3, \tag{32}$$

and each (i, j) off-diagonal block encodes the interrelated information between vertex subsets V_i and V_j. A pictorial description of L is shown in Figure 2.

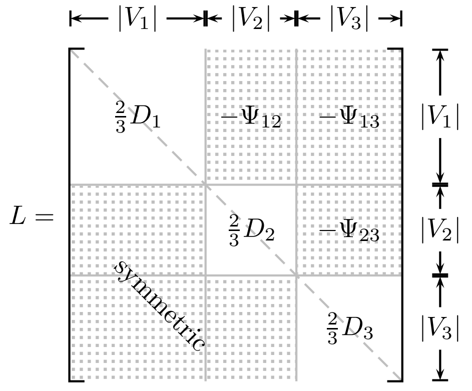


Figure 2: The structure of the Laplacian of a hypergraph induced from an order-3 data tensor.

4.2 The Co-Clustering Algorithm

Our k-way co-clustering algorithm for a data tensor w consists of three steps:

1. Form the Laplacian L and diagonal matrix W.

2. Obtain the k bottom eigenvectors z_1, z_2, …, z_k of the matrix pencil (L, W). Let $Z = [z_1, z_2, \ldots, z_k]$.

3. Use Z to obtain a clustering of each mode of w.

The first step exploits formulas (30) and (31), and the second step needs only an efficient numerical eigen-solver (more details will be discussed in the next subsection). The many varieties come from the last step, where, in the graph case, it has been shown by many practical situations that each approach of defining the final clustering has pros and cons. The following is a discussion of criteria that work well in practice.

For a 2-way co-clustering, the eigenvector z_1 is ignored. For each vertex $u \in V$, we form a clustering

$$S_{(u)} = \{v \in V \mid z_2(v) \geq z_2(u)\} \quad \text{and} \quad S^c_{(u)} = \{v \in V \mid z_2(v) < z_2(u)\}$$

and compute the cost function $c\left(S_{(u)}\right)$. By enumerating all the vertices, we obtain the optimal cut-point $z_2(u)$ such that $c\left(S_{(u)}\right)$ is minimized. Thus, the vector z_2 is partitioned in two subsets, and all the modes of w are simultaneously partitioned.

For a general k-way co-clustering, we normalize each row of Z and treat them as data items. Then we run k-means to obtain a k-clustering of the vertex set V. Hence, each subset V_i is partitioned in k parts.

Recall that there exist many heuristics to recover the partitioning after Z is computed. Different approaches may work well for different data sets, and there are computational cost trade-offs.

4.3 Computing the Eigenvectors

Since the matrix W is diagonal, the most efficient way of computing the eigenvector z of the matrix pencil (L, W) is to solve an equivalent eigenvalue problem

$$(W^{-1/2} L W^{-1/2}) f = \lambda f, \qquad \text{with } f = W^{1/2} z. \tag{33}$$

In the case of ratio hypergraph cut, $W = I$, hence (33) is simply $Lf = \lambda f$, whereas in the case of normalized hypergraph cut, $W = D$, hence (33) becomes $\Delta f = \lambda f$.

The bottom eigenvectors f can be efficiently computed using the Lanczos procedure [25]. The dominant cost of this procedure is $O(k' \cdot \mathrm{nnz}(L))$, where k' is the number of Lanczos steps, and $\mathrm{nnz}(L)$ is the number of nonzeros of the matrix L. The number of steps/iterations k' varies according to the matrix, and is typically a few times of k. The number of nonzeros assumes the form $\mathrm{nnz}(L) = O(|V_1||V_2| + |V_1||V_3| + |V_2||V_3|)$. On one extreme, if $|V_1|$, $|V_2|$ and $|V_3|$ are the same, then $\mathrm{nnz}(L) = \frac{2}{3}|V|^2 + |V|$. On the other extreme, if, say $|V_1|$, is dominant over $|V_2|$ and $|V_3|$ (such that they can be considered just some constants), then $\mathrm{nnz}(L) = O(|V_1|) = O(|V|)$. Hence, the computational cost of computing the eigenvectors lies somewhere between $O(k'|V|)$ and $O(k'|V|^2)$. In particular, if the Laplacian L is sparse, then it has the number of nonzeros proportional to $|V|$, thus the cost is approximately $O(k'|V|)$.

4.4 The Order-n Case

Now that the co-clustering technique for an order-3 tensor has been made clear, we complete this section by briefly summarizing the details for the general order-n case.

An order-n nonnegative tensor $w \in \mathbb{R}^{|V_1|\times|V_2|\times\cdots\times|V_n|}$ induces a weighted hypergraph $G(V, E, w)$ for some disjoint sets V_1, …, V_n, where $V = \bigcup_{i=1}^n V_i$, $E = \prod_{i=1}^n V_i$, and $w(v_1, v_2, \ldots, v_n)$ is the weight for a hyperedge $e = \{v_1 \in V_1, v_2 \in V_2 \ldots, v_n \in V_n\}$.

The Laplacian L of this hypergraph has size $|V| \times |V|$. Split into blocks, it reads

$$L = D - \Psi = \begin{bmatrix} D_1 - \Psi_{11} & -\Psi_{12} & \cdots & -\Psi_{1n} \\ -\Psi_{21} & D_2 - \Psi_{22} & \cdots & -\Psi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -\Psi_{n1} & -\Psi_{n2} & \cdots & D_n - \Psi_{nn} \end{bmatrix}, \tag{34}$$

where each Ψ_{ij} block has elements

$$\psi_{ij}(v_i, v_j) = \psi_{ji}(v_j, v_i) = \tfrac{1}{n} \textstyle\sum_{\{v_1,\ldots,v_n\}-\{v_i,v_j\}} w(v_1, v_2, \ldots, v_n), \tag{35}$$

and each D_i is a diagonal matrix with diagonal elements

$$d_i(v_i) = \textstyle\sum_{\{v_1,\ldots,v_n\}-\{v_i\}} w(v_1, v_2, \ldots, v_n). \tag{36}$$

A pictorial description of L is shown in Figure 3.

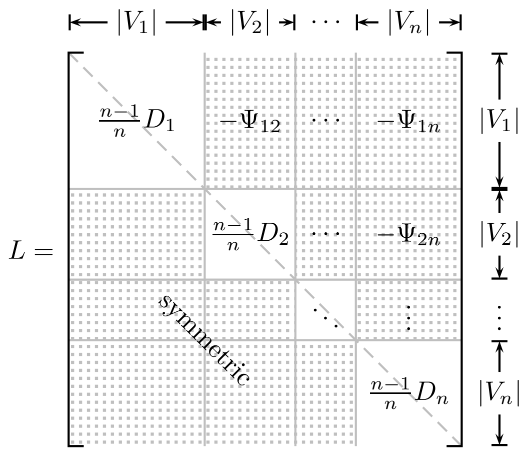


Figure 3: The structure of the Laplacian of a hypergraph induced from an order-n data tensor.

The co-clustering algorithm is exactly the same as that discussed in Section 4.2. Again, the cost of computing the eigenvectors is $O(k' \cdot \text{nnz}(L))$. The term $\text{nnz}(L)$ assumes the form $O(\sum_{i\neq j} |V_i||V_j|)$, which lies somewhere between $O(|V|)$ and $O(|V|^2)$. In particular, when L is sparse, $\text{nnz}(L)$ is roughly $O(|V|)$.

It is easy to see that when $n = 2$, the presented co-clustering algorithm for an order-n tensor (hypergraph) reduces to the co-clustering algorithm for a bipartite graph [11, 31], since when the graph is interpreted as a hypergraph, the Laplacian is equivalent to that defined for the graph, up to a constant factor $\frac{1}{2}$ (cf. Theorem 1). Hence, our algorithm can be considered a higher order generalization of that in [11, 31].

5 Experimental Results

This section gives a few experimental results to show the effectiveness of tensor data co-clustering using the spectral hypergraph partitioning technique. A case study is also provided to demonstrate that co-clustering can be used to enhance collaborative filtering. For all experiments the matrix W is set as D, i.e., the co-clusterings are based on the normalized hypergraph cut.

5.1 Toy Examples

To test the correctness of the tensor co-clustering algorithm, we used two synthetic order-3 tensors that have clear structures. The data were prepared as follows. First, a nonnegative tensor $\hat{w}$ of size $20 \times 20 \times 20$ with random entries (in $[0, 1]$) was generated. Then, a tensor w_1 was constructed by modifying $\hat{w}$ on four "opposite corners":

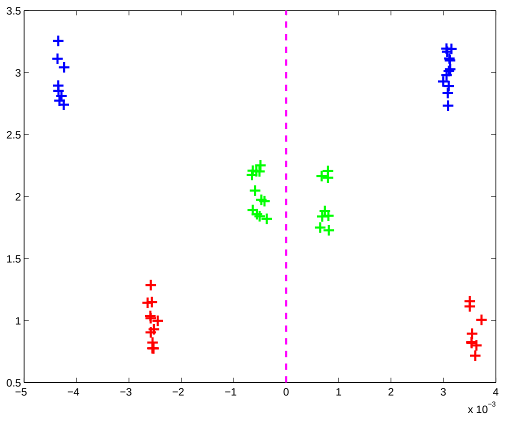
$$\begin{aligned}
w_1 &\leftarrow \hat{w}, \\
w_1(1:8, 1:8, 1:8) &\leftarrow \hat{w}(1:8, 1:8, 1:8) + 10, \\
w_1(9:20, 9:20, 1:8) &\leftarrow \hat{w}(9:20, 9:20, 1:8) + 10, \\
w_1(1:8, 9:20, 9:20) &\leftarrow \hat{w}(1:8, 9:20, 9:20) + 10, \\
w_1(9:20, 1:8, 9:20) &\leftarrow \hat{w}(9:20, 1:8, 9:20) + 10,
\end{aligned}$$

and the other tensor w_2 was constructed by modifying $\hat{w}$ at the same locations in a different way:

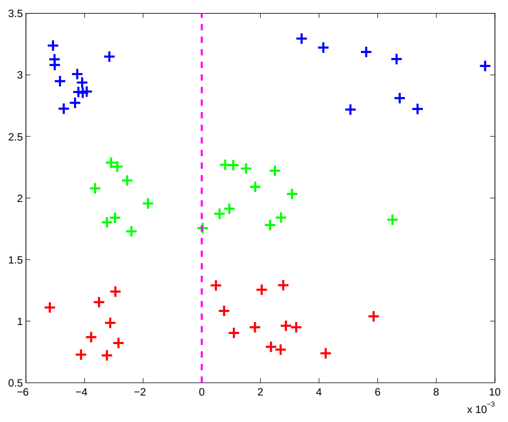
$$\begin{aligned}
w_2 &\leftarrow \hat{w}, \\
w_2(1:8, 1:8, 1:8) &\leftarrow \hat{w}(1:8, 1:8, 1:8) \times 10, \\
w_2(9:20, 9:20, 1:8) &\leftarrow \hat{w}(9:20, 9:20, 1:8) \times 10, \\
w_2(1:8, 9:20, 9:20) &\leftarrow \hat{w}(1:8, 9:20, 9:20) \times 10, \\
w_2(9:20, 1:8, 9:20) &\leftarrow \hat{w}(9:20, 1:8, 9:20) \times 10.
\end{aligned}$$

It is obvious that the data in w_1 has a clear co-clustering—each mode is split between the 8th and the 9th entry—while the data in w_2 may also be split in this way.

Figure 4 shows the second smallest eigenvector z of the Laplacians of the hypergraphs induced from the two tensors. In each of the plot, the horizontal axis corresponds to the entry values, while the entries are plotted at different heights simply for better visualizations. Each color corresponds to a mode of the tensor. For w_1, the entries of z are clearly clustered, and indeed, splitting the vector according to the signs perfectly recovers the co-clustering as intended by the construction of the data. For w_2, the entry values also form separated clusters, and the co-clustering according to the vector z well reveals the structure of the tensor. This experiment shows that our co-clustering algorithm works very well for synthetic examples.



(a) Data tensor w_1.



(b) Data tensor w_2.

Figure 4: Plots of the eigenvectors for two synthetic data tensors (Laplacians). Horizontal locations correspond to entry values, while the entries are plotted at different heights for better visualizations. Colors correspond to different modes of the tensors.

5.2 Knowledge Discovery from Users-Movies-Genres Data

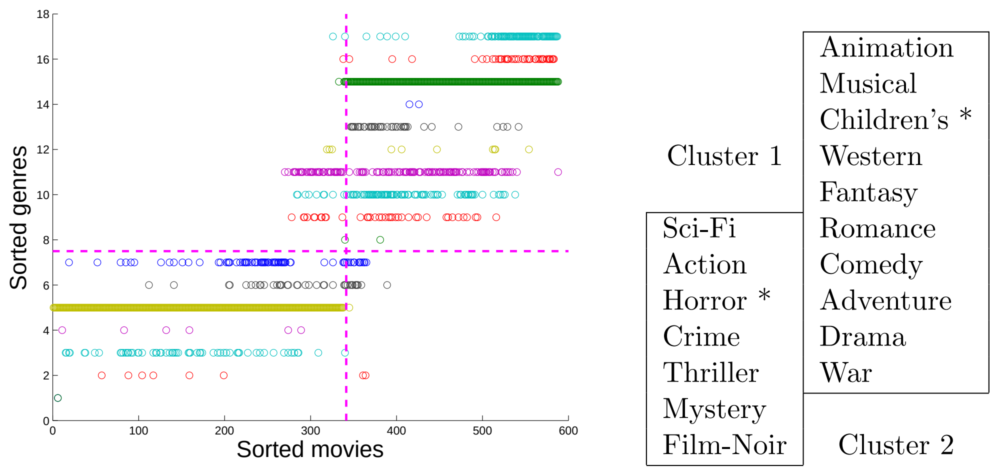
In this subsection, we perform co-clustering on a real-life tensor. The results reveal interesting information on the data, and show advantages over the spectral bipartite graph co-clustering technique.

The tested tensor is constructed from the MovieLens data set (million ratings)[3]. Based on the standard users-by-movies matrix in the collaborative filtering scenario, the data is augmented by introducing a third mode—movie genres. Hence we have a data tensor that represents three types of entities: users, movies, and genres. Each entry signifies the preference/rating of a user to a movie which belongs to some genre. The entries take integer values from 1 to 5. To construct a meaningful set of data, we selected only the movies that belong to either `Horror` or `Children's`. This resulted in a 5844 users $\times$ 588 movies $\times$ 17 genres tensor with $377,480$ ratings (nonzeros), among which $148,359$ are distinct user ratings for certain movies. The rest are duplicates since each movie may belong to various genres.

We performed a 2-way co-clustering on the tensor and show the results in Figures 5 and 6(a). For simplicity, the final co-clustering is defined according to the signs of the eigenvector entries. Indeed, what is important here is their ordering. Figure 5 shows the results along the movies-by-genres side of the tensor. The movies are clearly partitioned in two clusters: one for `Horror` movies and the other for `Children's`. An interesting observation is that the movie genres are also split in two clusters, depending on whether they are more related to `Horror` or to `Children's`. The only exception may be the genre `War`, which lies on the border of the split. It might be preferred to have `War` less related to `Children's` than `Horror`. Nevertheless, in general we see that the co-clustering gives reasonable results on categorizing the movies and the genres.

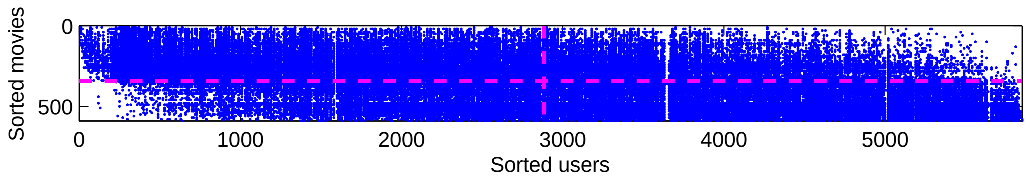
Figure 6(a) gives the co-clustering result along the users-by-movies side of the

[3]`http://www.grouplens.org/`

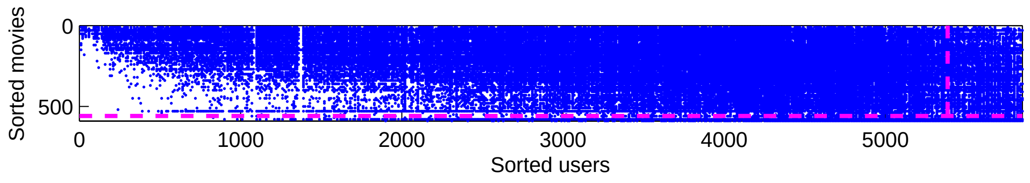


(a) (b)

Figure 5: Co-clustering of the MovieLens tensor: the movies-by-genres side.



(a) Tensor model



(b) Bipartite graph model

Figure 6: Co-clustering of the MovieLens tensor: the users-by-movies side.

tensor. As a comparison, we perform a co-clustering on the users-by-movies matrix based on the bipartite graph model [11, 31] and show the result in 6(b). Intuitively, a good co-clustering should have the nonzero patterns aggregate to the top-left and bottom-right corners of the matrix as much as possible. If the clusters are well defined, the other two corners of the reordered matrix are very sparse. However, for this data it seems that the clusters have significant overlaps. In this case, the information provided by the co-clustering of the tensor is more suggestive than that obtained from the co-clustering on the matrix only.

5.3 A Case Study: Augmented Collaborative Filtering

The role of collaborative filtering is to predict the preference of a customer to a particular product, given his/her preferences to other products and other customers' preferences to the same product. In the context of movie rating, a large user-by-movie matrix is given, where each entry is the rating of a movie given by a specific user. Typically the matrix is rather sparse, since it may not be possible to ask every user to rate all the movies in the database. Given the matrix of existing ratings, the task is to predict some unknown entries.

Co-clustering helps the predictions by identifying strong user-movie connections. Consider that the sets of users V_u and movies V_m are co-clustered into subsets $V_{u,1}$, $\ldots$, $V_{u,k}$ and $V_{m,1}$, $\ldots$, $V_{m,k}$ by k-way co-clustering. Then for each i, the subset $V_{u,i}$ is strongly tied with $V_{m,i}$ since they belong to the same (i-th) partition. Hence, it is very likely that the prediction of a user rating to a movie, based on the training samples in the submatrix $V_{u,i} \times V_{m,i}$, is more accurate than that based on the original whole matrix. On the other hand, if some user belongs to the $V_{u,i}$ group but the movie does not belong to the $V_{m,i}$ group, we may not be able to make improvements other than computing predictions based on all the training data we have.

Our augmented collaborative filtering adds one more dimension to the data before performing co-clustering, which is, the genre dimension. As seen from the previous subsection, augmenting the matrix data to a tensor helps to obtain a more meaningful co-clustering. In short, our scheme is the following: We construct the users-by-movies-by-genres tensor, on which we perform a co-clustering. In particular, the users are clustered into $V_{u,1}$, $\ldots$, $V_{u,k}$, and the movies $V_{m,1}$, $\ldots$, $V_{m,k}$. If a user u and a movie m belongs to $V_{u,i}$ and $V_{m,i}$ respectively for some i, then the prediction of u's rating to m is computed based on the training data in $V_{u,i} \times V_{m,i}$ only. Otherwise, the rating is computed as usual. Here, we haven't specified the "usual" method for predicting. In fact, this scheme works for all the methods we experimented with.

We used the whole MovieLens data set for tests. We did a $4:1$ train/test split and removed empty rows and columns. This resulted in a 6040 users $\times$ 3679 movies matrix. The corresponding tensor in addition has a mode of 18 genres. We performed 3-way co-clusterings, and the collaborative filtering methods were naive average prediction (the baseline), Pearson correlation [27], and SVD [28]. The prediction accuracy is measured by the mean-average-error (MAE).

Table 1 shows the MAE of the predictions for entries that belong to the $V_{u,i} \times V_{m,i}$ blocks. It can be seen that the predictions of these entries in general are improved if a specific method is applied on the $V_{u,i} \times V_{m,i}$ submatrix instead of on the whole

AVE		COR		SVD	
0.7776	0.7600	0.6883	0.6800	0.7063	0.6955
0.7756	0.7724	0.6976	0.7001	0.7273	0.7218
0.6923	0.6926	0.6412	0.6442	0.6670	0.6583

Table 1: Comparisons of MAE for each $V_{u,i} \times V_{m,i}$ submatrices. Three methods were tested: AVE, COR, and SVD. In each sub-table, the left column represents predictions computed based on all the training data, while the right column represents those computed based on only the data in the $V_{u,i} \times V_{m,i}$ submatrix. In general, our scheme (the right column) is better (than the left column).

AVE		COR		SVD	
0.7489	0.7452	0.6834	0.6826	0.7019	0.6976

Table 2: Comparisons of MAE. Three methods were tested: AVE, COR, and SVD. In each sub-table, the value on the left is the error from no co-clustering, while the value on the right is that from co-clustering. Co-clustering slightly improves on these existing methods.

matrix. In particular, for methods that assume an underlying structure of the data (for example, the SVD method assumes that the data matrix consists of a few rank-one structures) the co-clustering results better capture this structure. Thus, clusters are more homogeneous and the methods yield better performance on the small structural clusters. The final prediction accuracies for the test data, as shown in Table 2, further convince that the idea of tensor co-clustering can benefit traditional collaborative filtering.

6 Conclusions

We have presented a co-clustering technique for high order relational data that assume a tensor form. The method is based on the spectral partitioning of hypergraphs, which provide natural models for tensor data. The spectral hypergraph partitioning techniques introduced in this paper are generalizations of those for graphs, by virtue of the consistent definitions of (hyper)graph Laplacians. The rich spectral theories of graphs can thus be seamlessly integrated to hypergraphs, and hypergraph partitionings inherit the advantages of spectral partitionings, such as clear objective functions, easy computations, and a natural spectral embedding interpretation. When applied to structural hypergraphs that are induced from data tensors, the hypergraph partitioning techniques have been shown to be effective in revealing the underlying cluster structures of the relational data. The proposed tensor co-clustering technique is a high order generalization of the bi-partite graph co-clustering technique, and can be applied to enhance other data mining tasks, such as collaborative filtering.

References

[1] A. Anagnostopoulos, A. Dasgupta, and R. Kumar. Approximation algorithms for co-clustering. In *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2008.

[2] Arindam Banerjee, Sugato Basu, and Srujana Merugu. Multi-way clustering on relation graphs. In *Proceedings of the SIAM International Conference on Data Mining*, 2007.

[3] Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S. Modha. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *J. Machine Learning Research*, 8:1919–1986, 2007.

[4] Ron Bekkerman, Ran El-Yaniv, and Andrew McCallum. Multi-way distributional clustering via pairwise interactions. In *Proceedings of the 22nd international conference on Machine learning*, 2005.

[5] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computatioin*, 16(6):1373–1396, 2003.

[6] Shouchun Chen, Fei Wang, and Changshui Zhang. Simultaneous heterogeneous data clustering based on higher order relationships. In *Proceedings of the Workshop on Mining Graphs and Complex Structures (MGCS07), in conjuction with the 7th IEEE International Conference on Data Mining (ICDM)*, 2007.

[7] Hyuk Cho, Inderjit S. Dhillon, Yuqiang Guan, and Suvrit Sra. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, 2004.

[8] Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[9] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.

[10] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000.

[11] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001.

[12] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-theoretic co-clustering. In *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003.

[13] C.H.Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and H.D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings IEEE International Conference on Data Mining (ICDM 2001)*, 2001.

[14] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. J.*, 23:298–305, 1973.

[15] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its applications to graph theory. *Czechoslovak Math. J.*, 25:619–633, 1975.

[16] Bin Gao, Tie-Yan Liu, Xin Zheng, Qian-Sheng Cheng, and Wei-Ying Ma. Consistent bipartite graph co-partitioning for star-structured high-order heterogeneous data co-clustering. In *Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2005.

[17] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[18] Scott W. Hadley. Approximation techniques for hypergraph partitioning problems. *Disc. Appl. Math.*, 59(2):115–127, 1995.

[19] L. Hagen and A.B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 11(9):1074–1085, 1992.

[20] TaeHyun Hwang, Ze Tian, Jean-Pierre Kocher, and Rui Kuang. Learning on weighted hypergraphs to integrate protein interactions and gene expressions for cancer outcome prediction. In *Proceedings of the Eighth IEEE International Conference on Data Mining*, 2008.

[21] Bo Long, Xiaoyun Wu, Zhongfei (Mark) Zhang, and Philip S. Yu. Unsupervised learning on k-partite graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.

[22] Bo Long, Zhongfei (Mark) Zhang, Xiaoyun Wu, and Philip S. Yu. Spectral clustering for multi-type relational data. In *Proceedings of the 23rd international conference on Machine learning*, 2006.

[23] Bo Long, Zhongfei (Mark) Zhang, and Philip S. Yu. A probabilistic framework for relational clustering. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007.

[24] Andrew Y. Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, 2002.

[25] Beresford N. Parlett. *The Symmetric Eigenvalue Problem.* SIAM Press, 1998.

[26] Manjeet Rege, Ming Dong, and Farshad Fotouhi. Co-clustering documents and words using bipartite isoperimetric graph partitioning. In *Proceedings of Sixth IEEE International Conference on Data Mining (ICDM 06)*, 2006.

[27] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, 1994.

[28] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Applications of dimensionality reduction in recommender systems – a case study. In *Proceedings of ACM WebKDD 2000 Web Mining for E-Commerce Workshop*, 2000.

[29] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 22(8):888–905, 2000.

[30] Liang Sun, Shuiwang Ji, and Jieping Ye. Hypergraph spectral learning for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008.

[31] Hongyuan Zha, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. Bipartite graph partitioning and data clustering. In *Proceedings of the tenth international conference on Information and knowledge management*, 2001.

[32] Dengyong Zhou, Jiayuan Huang, and Bernhard Schöelkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in Neural Information Processing Systems 18*, 2006.