Hypergraph Learning with Hyperedge Expansion

Li Pu and Boi Faltings

Artificial Intelligence Laboratory
École Polytechnique Fédérale de Lausanne
CH-1015, Lausanne, Switzerland
{li.pu,boi.faltings}@epfl.ch

Abstract. We propose a new formulation called *hyperedge expansion* (HE) for hypergraph learning. The HE expansion transforms the hypergraph into a directed graph on the hyperedge level. Compared to the existing works (e.g. star expansion or normalized hypergraph cut), the learning results with HE expansion would be less sensitive to the vertex distribution among clusters, especially in the case that cluster sizes are unbalanced. Because of the special structure of the auxiliary directed graph, the linear eigenvalue problem of the Laplacian can be transformed into a quadratic eigenvalue problem, which has some special properties suitable for semi-supervised learning and clustering problems. We show in the experiments that the new algorithms based on the HE expansion achieves statistically significant gains in classification performance and good scalability for the co-occurrence data.

1 Introduction

Many tasks require clustering in a graph where each edge represents a similarity relation. Often, it is a co-occurrence relation that involves more than two items, such as the co-citation and co-purchase relations. The co-occurrence relation can be represented by a hyperedge that connects two or more vertices in a hypergraph. But most clustering algorithms, such as k-means, or spectral clustering, are defined for graphs but not hypergraphs. Therefore, hyperedge relations are often transformed into another graph that is easier to handle [1,2,3].

For classification and clustering tasks, the hyperedges are usually transformed into cliques of edges. This category of techniques includes *clique expansion*, *star expansion* [4], and *normalized hypergraph cut* (NHC) [5]. In Figure 1 we shown a simple example of such transformation from a hypergraph to a graph (the *induced graph*). Since the transformations are carried out on the vertex level, we call them *vertex expansions*.

With a vertex expansion, evaluating the goodness of clustering is done on the induced graph. For example, in a hyperedge of k vertices, a cut that separates the hyperedge into 1 and $k-1$ vertices would cut $k-1$ pairwise edges, while a cut that splits the vertices in two equal halves would have $k^2/4$ cut edges. Thus the vertex expansion would prefer an unbalanced clustering. To mitigate

P. Flach et al. (Eds.): ECML PKDD 2012, Part I, LNCS 7523, pp. 410–425, 2012.

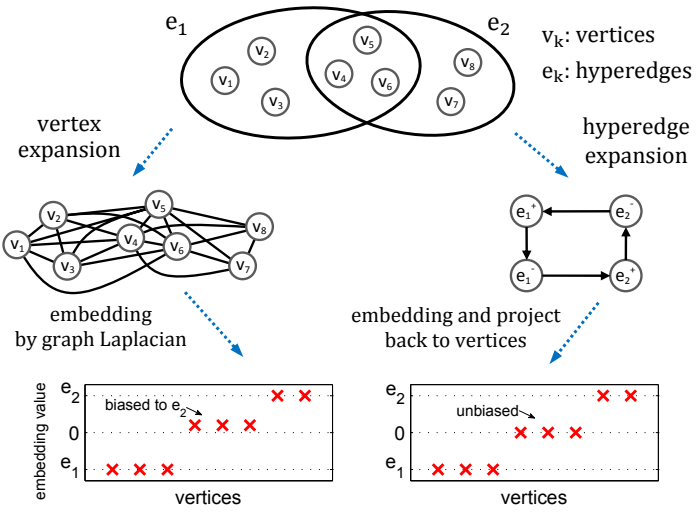

Fig. 1. An example of hypergraph embedding with two hyperedges. The hyperedge expansion embedding is unbiased, while the vertex expansion embedding (not to scale) depends on the hyperedge sizes. (see Section 4 for more details)

the problem of unbalanced clustering, it is proposed in star expansion and NHC to use the cluster volume as a normalizer for balancing the cluster sizes. But such normalization can not completely eliminate the problem. We present the following example of vertex embedding to explain why the problem still exists.

By computing the eigenvectors of the normalized Laplacian $\boldsymbol{L}_{NHC}$ of the induced graph, it is possible to project the vertices into an Euclidian space, which is called *embedding* in spectral graph learning [5]. On the left side of Figure 1, we show the 1-dimensional vertex embedding of NHC by the eigenvector corresponding to the second smallest eigenvalue of $\boldsymbol{L}_{NHC}$. It is worth to focus on the vertices that belong to both hyperedges (the overlapping part). Although the hyperedges have the same weight and the cluster volume normalizer is applied, the overlapping part is still biased to the side with less vertices (in this case e_2 side). This means that the optimal clustering of two clusters should assign the overlapping part and other vertices in e_2 to one cluster. Such bias might be a problem when the hyperedge sizes are unbalanced, e.g. co-citation relations with a lot or a few citations. Moreover, the behavior of the artificial normalization (or "correction") could be undesirable when many hyperedges intersect with each other, because the cost of the clustering would depend on how a hyperedge is split into the clusters. An even split would introduce a different cost compared to an uneven split.

As any hyperedge that is not entirely within the same cluster represents a relation that is violated by the clustering, it would be natural to have the learning result independent of the hyperedge sizes and only depend on the hyperedge connectivity and hyperedge weights. We present a new transformation called

hyperedge expansion (HE) based on a network flow technique so that the learning result is invariant to the distribution of vertices among hyperedges. HE expansion is first carried out on the hyperedge level. Then the learning results on hyperedges are projected back to the vertices through the adjacency information between hyperedges and vertices. In Figure 1, the embedding with HE expansion places the overlapping vertices in the middle without bias.

The main contributions of this paper are as follows. We formulate the HE expansion with the Laplacian of the auxiliary directed graph, which can be transformed into a quadratic eigenvalue problem that has some new properties. To our best knowledge, this is the first work to use such formulation. We also present the embedding and semi-supervised learning algorithms for hypergraphs based on HE expansion. In the experiments the proposed algorithms are compared with state-of-the-art methods and show statistically significant gains in performance.

2 Problem Statement

A hypergraph $\mathcal{H} = \{\mathcal{V}, \mathcal{E}, w\}$ consists of a *vertex set* $\mathcal{V}$, a *hyperedge set* $\mathcal{E}$, and a weighting function $w : \mathcal{E} \to \mathbb{R}^+$. Each hyperedge $e \in \mathcal{E}$ is a subset of $\mathcal{V}$. A hyperedge e is *incident* with a vertex v if $v \in e$. The weighted degree of a vertex is $deg(v) = \sum_{v \in e, e \in \mathcal{E}} w(e)$. The degree of a hyperedge is $deg(e) = |e|$. We say a hypergraph $\mathcal{H}$ is *connected* if for any v_i, v_j there exists a hyperedge path $\{e_1, e_2, ..., e_p\}$ such that $v_i \in e_1$, $v_j \in e_p$ and $e_k \cap e_{k+1} \neq \emptyset$ $(1 \leq k < p)$. Without loss of generality, we assume the hypergraph is always connected in this paper.

The (undirected) *induced graph* $\mathcal{G}_\mathcal{H} = \{\mathcal{V}', \mathcal{E}', w'\}$ derived from the hypergraph $\mathcal{H}$ consists of the same vertex set $\mathcal{V}' = \mathcal{V}$. An edge $e' \in \mathcal{E}'$ is placed between v_i and v_j in $\mathcal{G}_\mathcal{H}$ if there exists a hyperedge e in $\mathcal{H}$ which is incident with both v_i and v_j. The edge weight is defined as $w'(e') = \sum_{e \in \mathcal{E}, e \ni v_i, v_j} w(e)/deg(e)$. One can show that the induced graph is closely related to the star expansion and NHC. In fact there is only a small difference between the Laplacian of the induced graph and the Laplacian of NHC on the main diagonal.

Let $\mathcal{Y}$ denote a set of class labels. A multi-class labeling on a hypergraph $\mathcal{H}$ is a mapping $l : \mathcal{V} \to \mathcal{Y}$ that associates each vertex v with a label $l(v)$. In this paper, only a single label is allowed for one vertex. Since l defines several clusters of vertices by the labels, we interchangeably use "clustering" or "partitioning" in the paper as "labeling". For a hyperedge e, $l(e) = \{l(v) | v \in e\}$ is the set of labels associated to e. When $|l(e)| > 1$, we say that the hyperedge e is broken or violated by l. Let $\mathcal{V}_{l,y}$ denote the set of vertices that carry label y in the labeling l, and $\mathcal{V}^c_{l,y} = \mathcal{V} \setminus \mathcal{V}_{l,y}$ denote the remaining vertices.

In a *hypergraph semi-supervised learning* (HSSL) problem, a partial labeling $\bar{l} : \bar{\mathcal{V}} \to \mathcal{Y}$ is known on a subset of vertices $\bar{\mathcal{V}} \subset \mathcal{V}$. We assume that the vertices in $\bar{\mathcal{V}}$ carry all the labels in $\mathcal{Y}$. The goal of HSSL is to find the full labeling l that coincides with $\bar{l}$ on $\bar{\mathcal{V}}$, and minimizes some objective $\min_l R(\mathcal{H}, l) \in \mathbb{R}$. We list two typical objective functions as following,

$$R_{HE} = \sum_{e \in \mathcal{E}, |l(e)|>1} w(e), \tag{1}$$

$$R_{NHC} = \sum_{y \in \mathcal{Y}} \frac{\sum_{e \in \mathcal{E}} \frac{1}{deg(e)} w(e) |e \cap \mathcal{V}_{l,y}| |e \cap \mathcal{V}_{l,y}^c|}{\sum_{v \in \mathcal{V}_{l,y}} deg(v)}. \tag{2}$$

The R_{HE} is the sum of the weights of hyperedges which are broken [6], and R_{NHC} is defined in [5]. One can show that the (relaxed) optimal solution for R_{NHC} is an eigenvector of the normalized Laplacian of the induced graph.

Some existing works, for example [7] and [8], have already shown that the pairwise affinity relations after the projection to the induced graph would introduce information-loss, and working directly on the hypergraph (like the objective R_{HE}) could produce better performance. Ladicky et al. also experimentally show that the objective which is invariant to the number of objects (vertices) in each co-occurrence relation (hyperedge) [9], namely the *invariance property*, would achieve better performance on classification tasks. We can verify that R_{HE} satisfies the invariance property, while R_{NHC} does not (see Figure 1).

Although the R_{HE} has been proposed for a long time, all the existing works focus on the exact algorithms that directly optimize R_{HE} (e.g. see [10] for an early work and [11] for recent works). These combinatorial algorithms can be efficient when the number of classes is small (e.g. 2 or 3), but the exact algorithm is NP-hard for an arbitrary number of classes. On the other hand, these algorithms produce a combinatorial solution of hard clustering. There is no information about the confidence with which a vertex belongs to a cluster.

We use a different approach, i.e. the spectral technique, to target the same objective R_{HE}. The complexity of our algorithm is linear to the number of classes, and the final result is a soft clustering where the certainty of assigning a vertex to a cluster can be interpreted. To our best knowledge, this is the first work that applies the quadratic eigenvalue analysis to the R_{HE} objective.

In another task called *hypergraph embedding*, we would like to project the vertices into a low dimensional Euclidean space $\mathbb{R}^k$ where the vertices that are close to each other in the hypergraph should also stay close (as shown in Figure 1). R_{HE} and R_{NHC} actually define different notions of "closeness" in the hypergraph.

3 Hyperedge Expansion

In the simple case of two classes, in order to implement R_{HE}, we need to find a set of hyperedges of minimum weight that need to be cut to separate the vertices of the hypergraph into two parts, which is called the *minimum hyperedge cut problem* (MHCP). The optimal MHCP solution for the hypergraph shown in Figure 2 (a) would split hyperedge e_3 into two parts and the optimal R_{HE} cost is $w(e_3)$ (for the moment just consider the example hypergraph without knowing the meaning of the hyperedge subscripts).

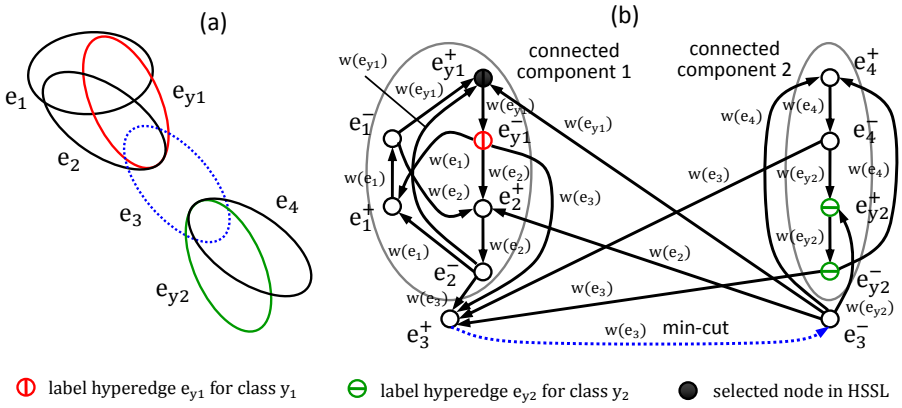

Fig. 2. (a) The original hypergraph where the hyperedge e_3 has the smallest weight. **(b)** The weighted directed graph $\hat{\mathcal{G}}$ constructed from the hypergraph. The directed edge (e_3^+, e_3^-) is the minimum cut of $\hat{\mathcal{G}}$ since its removal separates $\hat{\mathcal{G}}$ into two strongly connected components.

The technique for solving MHCP dates back to [10] where the hypergraph is transformed into a flow network and the minimum hyperedge cut is identified with a max-flow solution. We use a slightly different transformation as in [12].

The hyperedge expansion works as follows. We construct a directed graph $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$ that includes two vertices e^+ and e^- for each hyperedge e in the original hypergraph. Note that the vertices in $\hat{\mathcal{G}}$ correspond to the hyperedges, but not the vertices in the original hypergraph. A directed edge is placed from e^+ to e^- with weight $w(e)$ where w is the weighting function in the hypergraph. For every pair of overlapping hyperedges e_1 and e_2, two directed edges (e_1^-, e_2^+) and (e_2^-, e_1^+) are added to $\hat{\mathcal{G}}$ with weights $w(e_2)$ and $w(e_1)$ (see Figure 2 (b)).

Then we can identify the solution of MHCP by finding the min-cut of $\hat{\mathcal{G}}$ that separates $\hat{\mathcal{G}}$ into at least two strongly-connected components. The correctness follows immediately from the construction of $\hat{\mathcal{G}}$. Note that the edges attached to an e^+ node have the same weights, and an e^+ node has exactly one outgoing edge. For any edge in the min-cut which goes from an e^- node to an e^+ node, we can replace it with the outgoing edge of the e^+ node and construct an equivalent min-cut. Thus the min-cut could contain only the edges from the e^+ nodes to the e^- nodes. As shown in Figure 2 (b), the min-cut solution includes only the directed edge (e_3^+, e_3^-). The cost of the min-cut is $w(e_3)$, which is exactly the same as the cost of the original MHCP.

In matrix form, the adjacency matrix of $\hat{\mathcal{G}}$ can be defined as

$$\boldsymbol{A}_{\hat{\mathcal{G}}} = \begin{bmatrix} \mathbf{0} & \boldsymbol{A}\boldsymbol{W} \\ \boldsymbol{W} & \mathbf{0} \end{bmatrix}, \tag{3}$$

where $\boldsymbol{A}$ is the $|\mathcal{E}| \times |\mathcal{E}|$ adjacency matrix of hyperedges ($\boldsymbol{A}(i,j) = 1$ if $e_i \cap e_j \neq \emptyset$ and $e_i \neq e_j$, otherwise $\boldsymbol{A}(i,j) = 0$), and $\boldsymbol{W} = \text{diag}([w(e_1), w(e_2), ...])$ is the

diagonal matrix of hyperedge weights. The index (i, j) after the matrix indicates the element at ith row and jth column. We sort the rows and columns of $\boldsymbol{A}_{\hat{\mathcal{G}}}$ in the order $[e_1^-, e_2^-, ..., e_1^+, e_2^+, ...]$, so the elements in all other matrices and vectors should follow the same order. The out-degree matrix of $\hat{\mathcal{G}}$ is $\boldsymbol{D}_{\hat{\mathcal{G}},out} = \left[\begin{smallmatrix} \boldsymbol{D} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W} \end{smallmatrix}\right]$, where $\boldsymbol{D} = \text{diag}(\boldsymbol{e}^\top \boldsymbol{W}\boldsymbol{A})$ and $\boldsymbol{e}$ is an all-ones vector. Then the out-degree Laplacian of $\hat{\mathcal{G}}$ can be defined as following

$$\boldsymbol{L} = \boldsymbol{D}_{\hat{\mathcal{G}},out} - \boldsymbol{A}_{\hat{\mathcal{G}}} = \begin{bmatrix} \boldsymbol{D} & -\boldsymbol{A}\boldsymbol{W} \\ -\boldsymbol{W} & \boldsymbol{W} \end{bmatrix}. \tag{4}$$

There are existing theories about the spectral property of the directed graph based on symmetrization of $\boldsymbol{L}$ [13,14], and the corresponding learning problem for directed graph [15]. It is shown that a Cheeger inequality can be established with the first non-trivial eigenvalue of $\tilde{\boldsymbol{L}} = \frac{\boldsymbol{V}\boldsymbol{P}+\boldsymbol{P}^\top \boldsymbol{V}}{2}$, where $\boldsymbol{P}$ is the (non-symmetric) transition probability matrix of the directed graph and $\boldsymbol{V}$ is the diagonal matrix of the first non-trivial eigenvector of $\boldsymbol{P}$. In the transition probability matrix $\boldsymbol{P}$, all out-going edge weights are normalized by the out-degree. For the special structure of $\hat{\mathcal{G}}$ in our case, the out-degree normalization could be problematic since the correct mapping from the min-cut of $\hat{\mathcal{G}}$ to the original MHCP problem relies on the special weighting of the edges. When changing the edge weights, it could be possible that the min-cut of $\hat{\mathcal{G}}$ also contains edges from the e^- nodes to the e^+ nodes, which is undesirable in our case. Instead, we avoid to use the normalized $\boldsymbol{P}$ and show that the unnormalized Laplacian $\boldsymbol{L}$ is connected to a relaxation of the min-cut problem on $\hat{\mathcal{G}}$.

Denote the nodes in the connected component on one side of the min-cut with $\mathcal{S} \subset \hat{\mathcal{V}}$, and the nodes on the other side with $\mathcal{S}^c$. We define a vector $f \in \{1/\sqrt{|\mathcal{S}|}, 0\}^{|\hat{\mathcal{V}}|}$, where $f(e)$ is the entry corresponding to the node e. $f(e) = 1/\sqrt{|\mathcal{S}|}$ if $e \in \mathcal{S}$ and otherwise 0. It can be shown that $f^\top f = 1$ and the cost of the cut can be written as

$$C = \sum_{e_1, e_2 \in \hat{\mathcal{V}}, (e_1, e_2) \in \hat{\mathcal{E}}} |\mathcal{S}| w(e_1, e_2) \left(f(e_1) - f(e_2)\right) f(e_1). \tag{5}$$

The second $f(e_1)$ ensures that only the edges from $\mathcal{S}$ to $\mathcal{S}^c$ are counted when $f(e_1) = 1/\sqrt{|\mathcal{S}|}$ and $f(e_2) = 0$. Then we relax f to take positive continuous values and find the relaxed f that minimizes C by the Lagrange multiplier method with constraint $f^\top f = 1$. When taking the partial derivatives, we drop the contributions from $f(e_2)$ which is close to zero. This implies the following approximation

$$\begin{aligned} \frac{\partial w(e_1, e_2) \left(f(e_1) - f(e_2)\right) f(e_1)}{\partial f(e_1)} &= 2w(e_1, e_2) f(e_1) - w(e_1, e_2) f(e_2) \\ &\approx 2w(e_1, e_2) f(e_1), \end{aligned} \tag{6}$$

$$\frac{\partial w(e_1, e_2) \left(f(e_1) - f(e_2)\right) f(e_1)}{\partial f(e_2)} = -w(e_1, e_2) f(e_1). \tag{7}$$

Setting the partial derivative with respect to $f(e)$ to zero, it results in a matrix form $f^\top \left(2\boldsymbol{D}_{\hat{\mathcal{G}},out} - \boldsymbol{A}_{\hat{\mathcal{G}}}\right) = 2\lambda f^\top$ where λ is the Lagrange multiplier. The matrix on the left side is the same as $\boldsymbol{L}$ except the doubled diagonal. We can also interpret 2λ as an eigenvalue and f as a left eigenvector.

For a non-Hermitian matrix like $\boldsymbol{L}$, the Courant-Fischers min-max theorem does not hold anymore. The field of values of the non-Hermitian matrix is a superset of the convex hull of the eigenvalues [16], and there is no guarantee that all the eigenvalues are real. Although $\boldsymbol{L}$ is a non-Hermitian matrix, we show in the next section that the special structure of $\hat{\mathcal{G}}$ leads to some special properties of $\boldsymbol{L}$ as addition to the properties in the general case. These special properties would allow us to carry out the learning tasks with $\boldsymbol{L}$.

4 Hypergraph Embedding

The embedding of a (hyper)graph projects the vertices into a low dimensional Euclidean space. With the NHC objective one can construct the $|\mathcal{V}| \times |\mathcal{V}|$ normalized hypergraph Laplacian $\boldsymbol{L}_{NHC} = \boldsymbol{I} - \frac{1}{2}\boldsymbol{D}_v^{-\frac{1}{2}}\boldsymbol{H}\,\boldsymbol{W}\,\boldsymbol{H}^\top\boldsymbol{D}_v^{-\frac{1}{2}}$, where $\boldsymbol{H}$ is the $|\mathcal{E}| \times |\mathcal{V}|$ incident matrix ($\boldsymbol{H}(e,v) = 1$ if $v \in e$; otherwise $\boldsymbol{H}(e,v) = 0$) and $\boldsymbol{D}_v = \text{diag}(deg(v))$ is the vertex degree matrix. Let $g_0, ..., g_{k-1}$ be the eigenvectors of $\boldsymbol{L}_{NHC}$ associated with the k smallest eigenvalues. The embedding of vertex v in a k-dimensional space is just the row vector at the v'th row of $[g_0, ..., g_{k-1}]$ [5].

We can also carry out this task with the R_{HE} objective by taking the left eigenvectors of $\boldsymbol{L}$ and mapping the hyperedge embedding back to the vertices. Suppose we have the left (real) eigenvectors $x_0, ..., x_{k-1}$ associated with the k smallest real eigenvalues of $\boldsymbol{L}$, i.e. $x_i^\top \boldsymbol{L} = \lambda_i x_i^\top,\ i \in \{0, ..., k-1\}$. Then the embedding for vertex v can be formulated as

$$\text{embedding}(v) = \left[x_0^-, ..., x_{k-1}^-\right]^\top \boldsymbol{H}(\cdot, v), \tag{8}$$

where x_i^- means the first half ($x_i(e^-)$ part) of x_i. But in the most general case the left eigenvectors could be complex for the non-Hermitian matrix $\boldsymbol{L}$. Fortunately, for most real problems, we show that all eigenvectors of $\boldsymbol{L}$ are real.

Theorem 1. *All eigenvalues of $\boldsymbol{L}$ are non-negative real numbers and the left eigenvectors of $\boldsymbol{L}$ are real if and only if there exists $\gamma \in \mathbb{R}$ such that the matrix $\boldsymbol{Q}(\gamma) = \gamma^2\,\boldsymbol{W}^{-2} + \gamma\,\boldsymbol{W}^{-1}(\boldsymbol{I} + \boldsymbol{W}^{-1}\boldsymbol{D}) + (\boldsymbol{W}^{-1}\boldsymbol{D} - \boldsymbol{A})$ is negative definite.*

Proof. Denote the eigenvalue of $\boldsymbol{L}$ by λ and the left eigenvector by $x = [x^-, x^+]$, where x^- and x^+ are the first and second halves of x. The eigenvalue problem $x^\top \boldsymbol{L} = \lambda x^\top$ can be reformulated as

$$\begin{aligned} \boldsymbol{D}x^- - \boldsymbol{W}x^+ &= \lambda x^-, \\ -\boldsymbol{W}\boldsymbol{A}x^- + \boldsymbol{W}x^+ &= \lambda x^+. \end{aligned}$$

By substituting $x^+ = \boldsymbol{W}^{-1}(\boldsymbol{D} - \lambda \boldsymbol{I})x^-$ in the second equation, we obtain a *quadratic eigenvalue problem* (QEP) $\boldsymbol{Q}(\lambda)x^- = 0$. Note that the coefficient matrices of λ^2 and λ are positive definite. It is known that a QEP is overdamped if and only if there exists $\gamma \in \mathbb{R}$ such that the matrix $\boldsymbol{Q}(\gamma)$ is negative definite and $(\boldsymbol{W}^{-1}\boldsymbol{D} - \boldsymbol{A})$ is positive semi-definite (see Theorem 2 and Definition 4 of [17]). Without loss of generality, the second condition can be always satisfied by scaling the hyperedge weights with the same factor. It is also known that the overdamped QEP $\boldsymbol{Q}(\lambda)x^- = 0$ has $2|\mathcal{E}|$ non-negative real eigenvalues, and thus $2|\mathcal{E}|$ real left eigenvectors. □

The condition stated in Theorem 1 is hard to verify in practice. The state-of-the-art techniques usually require to actually compute all the eigenvalues of the QEP. We give a sufficient condition which is easier to verify.

Corollary 1. *All eigenvalues of* $\boldsymbol{L}$ *are non-negative real numbers and the left eigenvectors of* $\boldsymbol{L}$ *are real if* $d(\boldsymbol{D}(i,i) + \boldsymbol{W}(i,i)) > 8\boldsymbol{D}(i,i)\,\boldsymbol{W}(i,i)$ *for all* $i \in \{1, ..., |\mathcal{E}|\}$, *where* $d = \min_i(\boldsymbol{D}(i,i) + \boldsymbol{W}(i,i))$.

Proof. As shown in Definition 1 of [17], the conclusion of Corollary 1 holds if

$$\left((x^-)^*\,\boldsymbol{W}^{-1}(\boldsymbol{I} + \boldsymbol{W}^{-1}\boldsymbol{D})x^-\right)^2 > 4\left((x^-)^*\,\boldsymbol{W}^{-2}x^-\right)\left((x^-)^*(\,\boldsymbol{W}^{-1}\boldsymbol{D} - \boldsymbol{A})x^-\right)$$

for all non-zero $x^- \in \mathbb{C}^{|\mathcal{E}|}$, where $(x^-)^*$ denotes the conjugate transpose of x^-. Let $z = \boldsymbol{W}^{-1}x^-$ and note that $\boldsymbol{W}^{-1}$ is a diagonal matrix with positive main diagonal. We can transform the above condition into

$$\left(\frac{z^*(\,\boldsymbol{W} + \boldsymbol{D})z}{z^*z}\right)^2 > \frac{z^*\,\boldsymbol{W}(4\,\boldsymbol{W}^{-1}\boldsymbol{D} - 4\boldsymbol{A})\,\boldsymbol{W}z}{z^*z}$$

for all non-zero $z \in \mathbb{C}^{|\mathcal{E}|}$. Both sides of the inequality contain a Rayleigh quotient. It can be shown that $d = \min_z \frac{z^*(\boldsymbol{W}+\boldsymbol{D})z}{z^*z} = \min_i(\boldsymbol{D}(i,i) + \boldsymbol{W}(i,i)) > 0$. Therefore a sufficient condition is

$$\frac{z^*\left(d(\,\boldsymbol{W} + \boldsymbol{D}) - 4\boldsymbol{D}\,\boldsymbol{W} + 4\,\boldsymbol{W}\,\boldsymbol{A}\,\boldsymbol{W}\right)z}{z^*z} > 0$$

for all non-zero $z \in \mathbb{C}^{|\mathcal{E}|}$, which means that the Hermitian matrix $\boldsymbol{R} = (d(\,\boldsymbol{W} + \boldsymbol{D}) - 4\boldsymbol{D}\,\boldsymbol{W} + 4\,\boldsymbol{W}\,\boldsymbol{A}\,\boldsymbol{W})$ must be positive definite. We know that $\boldsymbol{R}$ is positive definite if $\boldsymbol{R}$ is strictly diagonally dominant and has all positive diagonal entries. Noting that each row of $(\,\boldsymbol{W}\,\boldsymbol{A}\,\boldsymbol{W} - \boldsymbol{D}\,\boldsymbol{W})$ sums up to 0, we obtain the sufficient condition in Corollary 1. □

In fact we find that all the hypergraphs tested in the experimental section satisfy this sufficient condition except the dataset *AmazonBook*. But the first 6 eigenvalues (smallest magnitude) of the hypergraph constructed from *AmazonBook* are all real non-negative numbers. Experiments in Section 6 show that the hyperedge expansion embedding works well in general.

5 Hypergraph Semi-supervised Learning

Like the existing works (e.g. NHC and [18]), we convert the multi-class HSSL problem into a set of binary classification problems. We pick up one class y each time, compute a *class score* for each unlabeled vertex v that indicates the possibility of v belonging to class y, and repeat this procedure for all labels. Finally the label with the highest class score is assigned to v.

5.1 Computing Class Scores

The desired procedure should take the hypergraph $\mathcal{H}$, the partial labeling $\bar{l}$, and the chosen class y as input, while output the class scores for all unlabeled vertices. The class score score(v, y) can be defined as the reciprocal of the "distances" from the labeled vertices of label y to an unlabeled vertex v. An intuitive score(v, y) could be the reciprocal of the average commute distance in the induced graph from v to each vertex of label y, which can be computed by the generalized inverse of $\boldsymbol{L}_{NHC}$ [19]. Or score(v, y) could be obtained by simulating a random walk on the induced graph with restart from labeled vertices of label y [20]. Here we compute the scores based on $\boldsymbol{L}$ with hyperedge expansion.

To incorporate the partial labeling, some auxiliary hyperedges have to be added to the hypergraph. For each label $y \in \mathcal{Y}$, we create a *label hyperedge* containing all the vertices in $\bar{\mathcal{V}}$ with label y. In other words, a new hyperedge $e_y = \bar{l}^{-1}(y)$ is added to the original hypergraph, which is illustrated in Figure 2 (a) as e_{y1} and e_{y2}. The weights of all label hyperedges are set to a pre-defined value w_l, i.e. $w(e_{y1}) = w(e_{y2}) = ... = w_l$.

In each step one class is selected (e.g. the chosen class is $y = y_1$ in Figure 2). Then we define the class modified Laplacian

$$\boldsymbol{L}_y = \boldsymbol{L} - \alpha \boldsymbol{B}, \quad \boldsymbol{B} = \begin{bmatrix} \ddots & & \\ & w(e_y) & \\ & & \ddots \end{bmatrix} \tag{9}$$

where α is a parameter and $\boldsymbol{B}$ has only one non-zero entry $w(e_y)$ in the bottom-right half diagonal corresponding to the position of e_y^+. A larger α would have a bigger influence on guiding the direction of the hyperedge partition around e_y^+, while a smaller α would let the partition follow the intrinsic principle direction of $\hat{\mathcal{G}}$. Note that for each class $y \in \mathcal{Y}$ the matrix $\boldsymbol{L}_y$ has to be recomputed. We denote the left eigenvector of $\boldsymbol{L}_y$ by f_y.

With the commonly-used symmetric Laplacian, the eigenvector associated with the second smallest eigenvalue, namely the *Fiedler vector*, is often taken to partition the graph. For $\boldsymbol{L}_y$ with arbitrary $\alpha > 0$, we show that the eigenvector of $\boldsymbol{L}_y$ with the smallest real eigenvalue has the following property.

Theorem 2. *When $\alpha > 0$, there exists one eigenvalue λ_y^0 of $\boldsymbol{L}_y$ which is real and has the smallest real part among all eigenvalues of $\boldsymbol{L}_y$. The left eigenvector f_y^0 corresponding to λ_y^0 has all positive entries. Furthermore we have $\lambda_y^0 \geq -\alpha w(e_y)$.*

Proof. Consider the matrix $\boldsymbol{L}'_y = \mu \boldsymbol{I} - \boldsymbol{L}_y$ where $\boldsymbol{I}$ is the identity matrix and $\mu > 0$. Since the underlying graph is strongly connected, the matrix $\boldsymbol{L}'_y$ is non-negative and irreducible for some μ. By the Perron-Frobenius theorem, there exists an all positive left eigenvector f_y^0 and an eigenvalue $\mu - \lambda_y^0$, which is real and has the biggest magnitude. Thus f_y^0 is a left eigenvector of $\boldsymbol{L}_y$ corresponding to λ_y^0. The bound of λ_y^0 directly follows the spectral radius bound of the Perron-Frobenius theorem. □

Then we can compute the score of each unlabeled vertex v as the sum of all $f_y^0(e^-)$ values where e is a hyperedge and $e \ni v$, i.e. $\text{score}(v, y) = \sum_{e \ni v} f_y^0(e^-)$. This score is repeatedly computed for each class $y \in \mathcal{Y}$ for v, and finally v is assigned to the class with the highest score. We could analogically justify the usage of f_y^0 as in the argument in the end of Section 3. The only difference is that all the vertices labeled with y should be assigned to the same side of the min-cut. This can be modeled as a soft constraint with the term $\alpha \boldsymbol{B}$ in (9).

5.2 The Algorithm and Complexity

Summing up all the procedures above, we obtain the complete HE expansion algorithm for HSSL. Only two parameters are required for the algorithm: the weight for label hyperedges w_l and the parameter α. Empirically it is a good choice to set w_l to the largest weight of all hyperedges.

Algorithm 1. : $l = \textbf{HSSL-HE}(\mathcal{H} = \{\mathcal{V}, \mathcal{E}, w\}, \bar{l} : \bar{\mathcal{V}} \to \mathcal{Y}, w_l, \alpha)$

Let $\mathcal{E}_{ex} = \mathcal{E} \cup \{e_y | y \in \mathcal{Y}\}$, where $e_y = \bar{l}^{-1}(y)$ of weight w_l
Compute $\boldsymbol{L}$ from $\mathcal{H} = \{\mathcal{V}, \mathcal{E}_{ex}, w_{ex}\}$ (see (4))
Initialize the score matrix $\boldsymbol{S}$ of size $|\mathcal{E}_{ex}| \times |\mathcal{Y}|$
for all $y \in \mathcal{Y}$ **do**
    Compute the matrix $\boldsymbol{L}_y = \boldsymbol{L} - \alpha \boldsymbol{B}$ (see (9))
    Compute the left eigenvector f_y^0 of $\boldsymbol{L}_y$ corresponding to the smallest real eigenvalue
    Fill in the y's column of $\boldsymbol{S}$ with the $f_y^0(e^-)$ part, i.e., the first half of f_y^0
end for
for all $v \in \mathcal{V} \setminus \bar{\mathcal{V}}$ **do**
    Let $l(v) = \arg\max_{y \in \mathcal{Y}} \sum_{e \ni v, e \in \mathcal{E}_{ex}} \boldsymbol{S}(e, y)$
end for
return $l = l \cup \bar{l}$

The HSSL-HE algorithm involves the computation of the eigenvector of a matrix of size $2N \times 2N$ ($N = |\mathcal{E}| + |\mathcal{Y}|$), and this procedure has to be repeated $|\mathcal{Y}|$ times. It is known that each eigenvalue problem can be solved in time $O(nN^2)$ by power iteration methods like Lanczos algorithm, where n is the number of iterations. The lower bound in Theorem 2 can be used as a good initial guess of the eigenvalue. If the connectivity between hyperedges is sparse, we can further reduce the time of computing eigenvector to $O(nN)$ and the total time complexity would be $O(nN|\mathcal{Y}|)$. Generally, HSSL-HE would have better scalability when

the number of instances (vertices) is very large and the number of co-occurrence relations (hyperedges) is relatively small. Such scenarios can be found in many real applications like categorical data and census data. On the other hand, the spectral methods that operate on the induced graphs, e.g. star expansion and NHC, need $O(n|\mathcal{E}'|)$ time where $|\mathcal{E}'|$ is the number of edges created in the induced graph.

6 Experimental Results

In this section, we present results on two tasks. First, we test the proposed semi-supervised algorithm on datasets from different domains and compare the performances with the state-of-the-art methods. Second, we present the result of the HE expansion embedding.

6.1 Experiment Settings

All the classification tasks are conducted in a transductive manner: we first create a hypergraph from raw data. Then the hypergraph and some (small amount of) vertex labels are taken as inputs and the algorithm predicts the labels of the unlabeled vertices. When evaluating the algorithms in repeated runs, the labeled vertices are randomly chosen from the vertex set such that every class has at least one labeled vertex, but the same set of labeled vertices is applied to all tested algorithms in each run. For evaluation we mainly use the macro-averaged F-score.

Algorithms for Comparison: since our proposed algorithm belongs to the family that only uses relational information, we choose four state-of-the-art relational-only approaches and one feature-based approach (AnchorGraph) for comparison [1]:

(1) the hMETIS toolkit [6] is a commonly used tool for hypergraph partitioning, which optimizes the R_{HE} objective with a heuristic algorithm. Although hMETIS is mainly designed for VLSI applications, reports show that this toolkit can be applied to general classification/clustering problems [21]. We use the "preassignment of vertices" input file with hMETIS to assign the known labels in the semi-supervised task.

(2) the *normalized hypergraph cut* (NHC) algorithm by Zhou et al. [5] first transforms the hypergraph into an induced graph whose edge weights are normalized by the hyperedge sizes. Then NHC adopts the normalized Laplacian $\boldsymbol{L}_{NHC}$ to the semi-supervised setting. The NHC algorithm is the most representative approach among those based on vertex expansions.

(3) the *rendezvous algorithm* (Rend.) [22] is a semi-supervised learning approach based on a random walk on a graph. The algorithm first constructs a directed graph from the k-nearest neighbors in which all the labeled vertices

[1] The implementation of our proposed algorithm (HSSL-HE) can be found in `http://lia.epfl.ch/index.php/research/relational-learning`

have only incoming edges and thus act as absorbing states of the random walk. Then the algorithm simulates a set of particles that start a random walk from each unlabeled vertex and stop at some labeled vertices. Intuitively, a particle from an unlabeled vertex will stop at a labeled vertex of its true label with higher probability. The algorithm determines the labels based on the outcome of the random walk. We use the distances in the induced graph (the same as NHC) to construct the directed k-NN graph. We also apply a Gaussian kernel function to the distances as instructed by the author.

(4) the *semi-supervised kernel k-means* (SSKKmeans) [23] is an extension of the kernel k-means method where the kernel function is a linear combination of the graph kernel and the label-induced modifier. The label-induced component includes both same-class rewards and different-classes penalties. Again we use the induced graph from the hypergraph (the same as NHC) to compute the graph kernel.

(5) the AnchorGraph algorithm [24] focuses on the scalability of semi-supervised learning. Instead of constructing a k-NN graph from the original data, AnchorGraph chooses a small set of anchors which connect to the s-nearest neighbors in the original data, and represents each data point with a linear combination of the anchors. The semi-supervised algorithm is faster because the values to learn are only the weights of the anchors rather than the labels of the original data.

In the experiments, we use 13 relational-only datasets from three different domains to evaluate the above algorithms. The *AmazonBook co-purchase dataset* contains the books in Amazon.com and the list of books that are co-purchased [25]. We take three subsets of book products to construct the hypergraphs, where a vertex represents a book, and a hyperedge represents a co-purchase list of books. The label of each vertex is simply the category of the corresponding book. The parameter α for algorithm 1 is set to 1 for *AmazonBook*. We also construct co-citation hypergraphs from the commonly-used *Cora*, *citeseer*, and *WebKB* data. For *Cora* and *citeseer*, a vertex represents a paper, and a hyperedge contains all the papers that cite the same paper. For *WebKB* data (*cornell* and *texas*), besides the link information, word-based content information is also available. So we create some additional hyperedges that include all the papers or webpages that contain the same word. In order to show how the link information could help in classification, the hypergraphs using only contents (denoted by C) and contents plus links (denoted by CL) are respectively constructed for each *WebKB* dataset. The parameter α is set to 50 for co-citation datasets. In the last domain, categorical dataset, every instance has a set of nominal attributes which could take values from a finite set. We use 4 labeled categorical datasets *zoo*, *letter*, *20newsgroups*, and *covertype* from the UCI repository. For each dataset, a hypergraph is constructed by taking instances as vertices and creating a hyperedge for each value of the attributes. Then every hyperedge contains the instances that share the same attribute value. We discretize those attributes whose value is an integer with a range larger than 10 into 10 sections of the same size. Some tested algorithms (SSKKmeans, NHC, and Rendezvous) do not scale well on *letter*, *20newsgroups*, and *covertype*, so only a subset is tested for

Table 1. The averaged macro F-scores (and the standard deviation in the parentheses) on 13 datasets. The algorithms are tested on *AmazonBook* (*AB*) and *covertype* with 10 runs, co-citation data with 50 runs, other categorical data with 100 runs. Some information about the dataset is shown below the dataset name (#labeled vertices / #all vertices / #classes). The bold number indicates a algorithm that performs significantly better than others (p-value < 0.05 in paired t-test). The Rendezvous algorithm cannot return a result in a reasonable time period for *AmazonBook*, *Cora* and *citeseer*.

dataset	hMETIS	SSKKmeans	AnchorGraph	Rend.	NHC	HE
AB3 (100/24500/3)	0.565(0.022)	0.446(0.015)	0.519(0.025)	—	0.645(0.019)	**0.657**(0.023)
AB4 (100/18120/4)	0.517(0.107)	0.376(0.016)	0.561(0.058)	—	0.765(0.046)	**0.798**(0.023)
AB5 (80/6965/5)	0.525(0.040)	0.357(0.067)	0.472(0.046)	—	0.724(0.087)	0.716(0.064)
Cora (40/1961/7)	0.477(0.054)	0.449(0.048)	0.500(0.050)	—	0.613(0.046)	**0.637**(0.040)
citeseer (40/1318/6)	0.492(0.046)	0.361(0.030)	0.401(0.038)	—	**0.518**(0.046)	0.509(0.046)
cornell-CL (20/195/5)	0.275(0.055)	0.411(0.091)	0.417(0.058)	0.304(0.068)	0.320(0.091)	**0.497**(0.047)
cornell-C (20/195/5)	0.279(0.057)	0.427(0.092)	0.425(0.059)	0.299(0.056)	0.346(0.069)	**0.480**(0.050)
texas-CL (20/187/5)	0.238(0.028)	0.362(0.050)	0.317(0.066)	0.249(0.042)	0.268(0.089)	**0.425**(0.045)
texas-C (20/187/5)	0.236(0.039)	0.350(0.047)	0.338(0.050)	0.254(0.047)	0.267(0.098)	**0.410**(0.068)
zoo (15/100/7)	0.467(0.066)	0.822(0.058)	0.803(0.075)	0.571(0.088)	0.359(0.147)	**0.832**(0.052)
letterAE (50/1022/5)	0.379(0.049)	0.629(0.023)	**0.664**(0.039)	0.543(0.039)	0.606(0.047)	0.627(0.028)
20newsgroups (50/1067/4)	0.489(0.080)	0.480(0.041)	0.552(0.042)	0.482(0.069)	**0.642**(0.033)	0.628(0.042)
covertype (50/6344/7)	0.164(0.017)	0.285(0.019)	0.238(0.016)	0.268(0.022)	0.254(0.064)	**0.307**(0.028)

each of them. We set $\alpha = 1$ for *20newsgroups* and $\alpha = 100$ for other categorical datasets. Weighting the hyperedges usually depends on the domain knowledge. For simplicity, we assign the same weights to all hyperedges in the experiments.

6.2 Main Results

As shown in Table 1, HE performs significantly better than other methods in most cases. For some datasets, hMETIS does not work very well, partially because it is mainly designed for VLSI applications, but not general classification tasks. For *cornell* and *texas*, we can observe an improvement from C to CL with the HE algorithm, which confirms that the link information does help in classifying webpages. This improvement, however, does not exist with other algorithms.

Nevertheless, the algorithms directly designed for hypergraphs (hMETIS, NHC and HE) generally perform significantly better than those based on graphs

(SSKKmeans) or feature vectors (AnchorGraph). It suggests that hypergraph approaches would be better choices when the data is naturally organized as co-occurrence relations. For *letterAE*, the AnchorGraph actually works best, mainly because the original attributes of *letterAE* are all integer values (such as the mean of x-position of the pixels) rather than nominal variables. When the data naturally follows some pattern in a continuous metric space, methods like AnchorGraph could be better for capturing the underlying regularity.

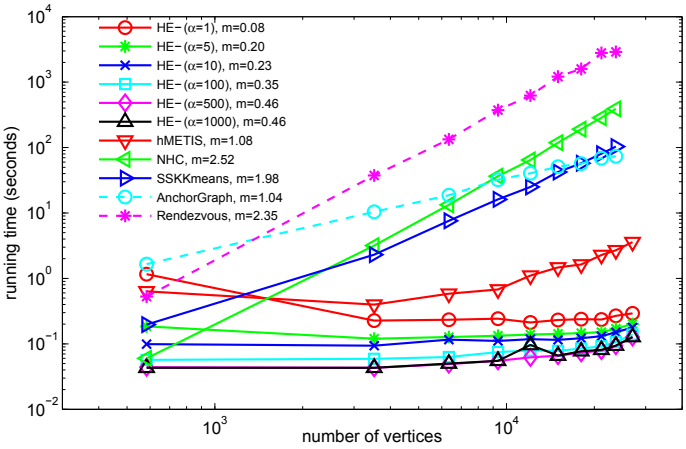

Fig. 3. The measured running times of different algorithms with 100 labeled vertices on subsets of *covertype*. The slope m of each curve is shown in the legend, which is computed by the least square fitting.

The running time of different algorithms is tested with subsets of *covertype* whose vertex set sizes range from 583 to 27056. These subsets are randomly extracted from the original data. Figure 3 shows the measured times in log scale. We have shown that the complexity of the HE algorithm mainly depends on the size of hyperedge set rather than the vertex set. Generally, the HE algorithm always stays in the same running time level regardless of the vertex set size, because the number of hyperedges in each subset does not change too much (from 122 to 143). Therefore the HE algorithm can be orders of magnitude faster than the approaches based on the induced graph when the number of hyperedges is smaller than the number of vertices. By increasing the parameter α, we can observe that HE runs faster due to the higher convergence rate of the eigenvector computation. In practice, the choice of α also depends on the classification performance, but the running time would not change by more than an order of magnitude when tuning α. The running times of hMETIS and AnchorGraph approximately grow linearly with respect to the number of vertices, while for NHC, SSKKmeans and Rendezvous the running time grows quadratically.

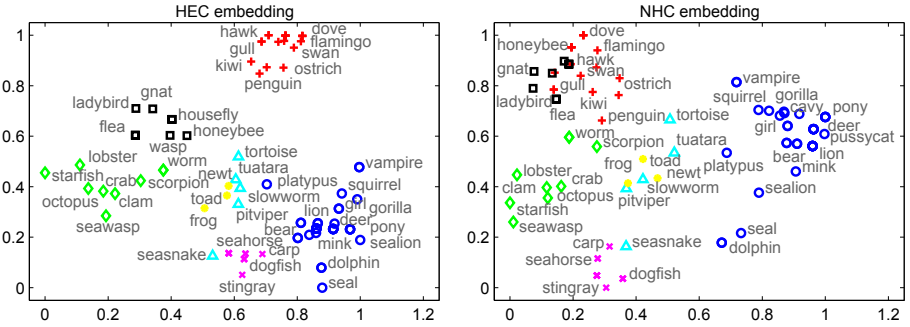

Fig. 4. The vertex embeddings of *zoo* with the eigenvectors (scaled) corresponding to the 2nd and 3rd smallest eigenvalues (for both HE and NHC embedding)

We use both HE embedding and NHC embedding to project the vertices (animals) of *zoo* into a low dimensional space. The results are shown in Figure 4. It can be seen that the HE embedding generates a different picture compared to the NHC embedding. In general the HE embedding shows a clearer separation between different classes in the 2-dimensional space, but for some instances (e.g. `seasnake` and `platypus`) both embeddings fail to give them a clear affiliation, mainly due to their special attributes.

7 Conclusion and Future Work

In this paper we propose a new formulation called hyperedge expansion and new algorithms for the semi-supervised learning and embedding tasks of hypergraph. Compared to the existing methods, the learning results with the hyperedge expansion is less sensitive to the hyperedges sizes when the data is organized with co-occurrence relations.

Our preliminary work has shown that the hyperedge expansion would be generally better than the vertex expansions when the average Jaccard coefficient between the hyperedges is high. Thus it is interesting to theoretically further investigate the applicable scopes of vertex expansions and hyperedge expansion. Moreover, we are interested in applying the hyperedge expansion technique to a broader range of real problems such as social networks and biological networks.

References

1. Chung, F.: The Laplacian of a hypergraph. Expanding graphs (DIMACS series), pp. 21–36 (1993)
2. Storm, C.: The zeta function of a hypergraph. The Electronic Journal of Combinatorics 13(R84) (2006)
3. Balof, B., Storm, C.: Constructing isospectral non-isomorphic digraphs from hypergraphs. Journal of Graph Theory 63(3), 231–242 (2010)

4. Agarwal, S., Branson, K., Belongie, S.: Higher order learning with graphs. In: Proceedings of the 23rd ICML (2006)
5. Zhou, D., Huang, J., Scholkopf, B.: Learning with hypergraphs: Clustering, classification, and embedding. In: Advances in Neural Information Processing Systems (2007)
6. Karypis, G., Aggarwal, R., Kumar, V., Shekhar, S.: Multilevel hypergraph partitioning: application in VLSI domain. In: Proceedings of the 34th Annual Design Automation Conference (1997)
7. Shashua, A., Zass, R., Hazan, T.: Multi-way Clustering Using Super-Symmetric Non-negative Tensor Factorization. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part IV. LNCS, vol. 3954, pp. 595–608. Springer, Heidelberg (2006)
8. Bulò, S., Pelillo, M.: A game-theoretic approach to hypergraph clustering. In: Advances in Neural Information Processing Systems (2009)
9. Ladicky, L., Russell, C., Kohli, P., Torr, P.H.S.: Graph Cut Based Inference with Co-occurrence Statistics. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 239–253. Springer, Heidelberg (2010)
10. Lawler, E.: Cutsets and partitions of hypergraphs. Networks 3(3), 275–285 (1973)
11. Fukunaga, T.: Computing Minimum Multiway Cuts in Hypergraphs from Hypertree Packings. In: Eisenbrand, F., Shepherd, F.B. (eds.) IPCO 2010. LNCS, vol. 6080, pp. 15–28. Springer, Heidelberg (2010)
12. Acid, S., Campos, L.: An algorithm for finding minimum d-separating sets in belief networks. In: Proceedings of the 12th UAI (1996)
13. Wu, C.: On Rayleigh-Ritz ratios of a generalized Laplacian matrix of directed graphs. Linear Algebra and Its Applications 402, 207–227 (2005)
14. Chung, F.: Laplacians and the Cheeger inequality for directed graphs. Annals of Combinatorics 9(1), 1–19 (2005)
15. Zhou, D., Huang, J., Schölkopf, B.: Learning from labeled and unlabeled data on a directed graph. In: Proceedings of the 22nd ICML (2005)
16. Horn, R., Johnson, C.: Topics in Matrix Analysis. Cambridge University Press (1991)
17. Guo, C., Lancaster, P.: Algorithms for hyperbolic quadratic eigenvalue problems. Mathematics of Computation, 1777–1791 (2005)
18. Sun, L., Ji, S., Ye, J.: Hypergraph spectral learning for multi-label classification. In: Proceeding of the 14th ACM SIGKDD (2008)
19. Von Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing 17(4), 395–416 (2007)
20. Lin, F., Cohen, W.: Semi-supervised classification of network data using very few labels. In: International Conference on Advances in Social Networks Analysis and Mining, pp. 192–199 (2010)
21. Strehl, A., Ghosh, J.: Cluster ensembles – a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research 3, 583–617 (2003)
22. Azran, A.: The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks. In: Proceedings of the 24th ICML (2007)
23. Kulis, B., Basu, S., Dhillon, I., Mooney, R.: Semi-supervised graph clustering: a kernel approach. In: Proceedings of the 22nd ICML (2005)
24. Liu, W., He, J., Chang, S.: Large graph construction for scalable semi-supervised learning. In: Proceedings of the 27th ICML (2010)
25. SNAP, `http://snap.stanford.edu/data/amazon-meta.html`