JMLR: Workshop and Conference Proceedings 30:1–16, 2015 ACML 2015

Integration of Single-view Graphs with Diffusion of Tensor Product Graphs for Multi-view Spectral Clustering

Le Shu LE.SHU@TEMPLE.EDU
Computer and Information Sciences, Temple University, 1925 N. 12 St. Philadelphia, PA, USA

Longin Jan Latecki LATECKI@TEMPLE.EDU
Computer and Information Sciences, Temple University, 1925 N. 12 St. Philadelphia, PA, USA

Abstract

Multi-view clustering takes diversity of multiple views (representations) into consideration. Multiple views may be obtained from various sources or different feature subsets and often provide complementary information to each other. In this paper, we propose a novel graph-based approach to integrate multiple representations to improve clustering performance. While original graphs have been widely used in many existing multi-view clustering approaches, the key idea of our approach is to integrate multiple views by exploring higher order information. In particular, given graphs constructed separately from single view data, we build cross-view tensor product graphs (TPGs), each of which is a Kronecker product of a pair of single-view graphs. Since each cross-view TPG captures higher order relationships of data under two different views, it is no surprise that we obtain more reliable similarities. We linearly combine multiple cross-view TPGs to integrate higher order information. Efficient graph diffusion process on the fusion TPG helps to reveal the underlying cluster structure and boosts the clustering performance. Empirical study shows that the proposed approach outperforms state-of-the-art methods on benchmark datasets.

Keywords: Multi-view Clustering, Tensor Product Graph, Markov Mixture Models, Graph Diffusion.

1. INTRODUCTION

In many clustering problems, data are collected from various sources or obtained from different feature extractors. For example, web pages can be described by either the content or the number of citation links between other pages; images can be represented by color and shape information of images as well as the textual tags separately; videos can be represented with content of audio and visual information. Although each view or representation may have sufficient information by its own, it often provides complementary information to the other views. Multi-view clustering seeks to integrate the information of multiple views to improve the clustering performance.

One popular way to solve unsupervised clustering is to cluster data based on the similarity between data samples. Spectral clustering von Luxburg (2007) based approaches have gained a lot attention due to their good performance on many benchmark data. It works on a single similarity matrix(graph), the edges of which represents the similarity between data samples. A number of approaches try to extend the single-view spectral clustering algorithms to work on multi-view data Kumar and Daumé III (2011), de Sa (2005), Zhou and Burges (2007), Xia et al. (2014). They first construct a single-view graph for each data

representation separately and then try to integrate the information from multiple single-view graphs. They improve the similarity between data samples either by forcing multiple single-view graphs to be consistent with each other or combining all single-view graphs convexly to form a single fusion graph. Especially, Zhou and Burges (2007) develop multi-view spectral clustering via generalizing the usual single view normalized cut to the multi-view case. This paper shows that the Markov mixture models reduce to the linear combination of normalized graphs for multiple undirected graphs. Kumar et al. (2011) combine multiple graphs from multiple views of data to achieve a better clustering by co-regularizing the clustering hypothesis. Despite their simplicity, approaches which convexly combine single-view graphs have achieved good multi-view clustering performance.

In this paper, we propose a novel multi-view spectral clustering approach based on Tensor Product Graphs (TPG). The key idea is to exploit higher order relations of single-view graphs. Higher order information has been utilized in many work before, e.g., Zhou et al. (2006), Yang et al. (2013). The intuition in their work is that relationships among the objects of our interest are more complex than pairwise in many real-world problems. Therefore higher order graphs can captures more information than graphs which only records pair-wise relationships in single view. Assume there are M views for each data sample, same as other multi-view spectral clustering approaches, we first compute M single-view graphs. A key step in our approach is to map M original single-view graphs to $M \times M$ cross-view TPGs. Each cross-view TPG is a Kronecker product of a pair of single-view graphs. Consequently, instead of computing a linear combination of original single-view graphs, we linearly combine $M \times M$ cross-view TPGs to integrate higher order information. We then perform graph diffusion on the fusion TPG to further discover intrinsic structure of multi-view data. As stated in Yang et al. (2013), graph diffusion on TPG is able to robustly learn similarities between data samples and can achieve state-of-the-art performance in image retrieval, image segmentation, and etc. In this work, we show that TPG diffusion can be generalized to multi-view settings and significantly boost the clustering performance. In fact, we prove that Yang et al. (2013) is a special case (i.e., $M = 1$) of our multi-view TPG diffusion formulation.

In Fig. (1), we use a toy example to better illustrate our motivation. As stated in Roweis and Saul (2000), data represented by data points may lie on data manifolds, e.g. "half-moon" example (Fig. (1a)). Pair-wise relationship between data points is not sufficient to reveal the intrinsic manifold structure. Compared to the original pair-wise similarity graph, a higher order graph, whose edges encode the similarity among a quad of data points, has clear advantage. However, when data has multiple views, every single view alone may not be able to fully preserve the manifold structure, even when considering higher order information. Imagine that half-moon data is provided with two views: one view is the data projected to x-axis, and the other view is the data projected to y-axis. It is easy to see in Fig. (1b) that the manifold structure would not be discovered from any of these two single views, even with graph diffusion, as shown in Fig. (1c and 1d). This motivates us to exploit cross-view TPGs whose edges not only encode high-order similarities but also integrate multiple views. As shown in Fig. (1e), our proposed method can significantly improve the similarity matrix which leads to a better clustering result. We examine multi-view clustering performance of the proposed method on benchmark datasets. Our approach

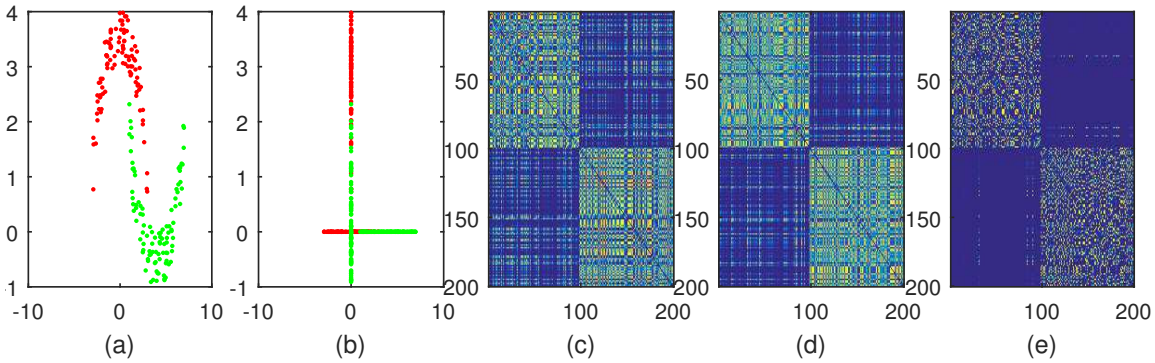

Figure 1: Visualization of half-moon data under two views. (a) Original half-moon data. Data of each class lie on different manifold. (b) Projection of data onto x- and y-axis as two views. (c) Similarity matrix obtained through single graph diffusion using view 1. (d) Similarity matrix obtained through single graph diffusion using view 2. (e) Similarity matrix obtained by our approach which performs cross-view tensor graph diffusion.

outperforms several state-of-the-art approaches. Besides, our method is extremely easy to implement. In summary, our contribution is twofold:

First, to the best of our knowledge, we are the first to utilize higher order graphs (cross-view TPGs) to solve multi-view clustering problems. In particular, higher order relations across every pair of views are captured in cross-view TPGs.

Second, we prove that our approach is a multi-view generalization of a single-view TPG diffusion approach proposed in Yang et al. (2013).

The rest of the paper is organized as follows: In Section 2, we review the existing multi-view clustering methods. Especially, we review graph-based approaches which are most related to the proposed method. In Section 3, we describe the methodology and algorithm. We present empirical studies on benchmark datasets in Section 4. Finally, a conclusion comes in Section 5.

2. RELATED WORK

Multi-view clustering algorithms have been extensively studied in the past, such as Bickel and Scheffer (2004), Chaudhuri et al. (2009), Cui et al. (2007), Gao et al. (2013), Zhou and Burges (2007), Wang et al. (2013), Cai et al. (2013), Davidson et al. (2013), Eaton et al. (2010). Existing approaches can be roughly categorized into three categories: multiple graph fusion algorithms, co-training algorithms, and subspace learning algorithms.

Multiple graph fusion algorithms is closely related to Multiple Kernel Learning(MKL) Lanckriet et al. (2002), Bach et al. (2004). These methods calculate separate single-view graphs based on each data representation, then attempt to convexly combine different graphs to form an ensemble/fusion graph. In Tzortzis and Likas (2012), weights are learnt when combining single-view graphs in parallel to the partitioning. Zhou and Burges (2007) define a random walk associated with each single-view graph and define a Markov Mixture model of those random walk to integrate information from multiple views. Kumar

et al. (2011) propose two spectral clustering algorithms, which combine multiple single-view graphs by co-regularizing the clustering hypotheses across views. de Sa (2005) utilize the 'minimizing-disagreement' idea to create a bipartite graph connecting data samples from one view to those in the other view, then solves standard spectral clustering problem on the bipartite graph. Unlike previous graph-based approaches, our method construct tensor product graphs for each pair of single-view graphs, which encode richer higher order relations.

Co-training style approaches often train separate learners on distinct views, then utilize the information in each learner to constrain other views. Bickel and Scheffer (2004) study to interchange partition information among different views after running clustering approaches separately. Kumar and Daumé III (2011) propose to utilize the spectral embedding from one view to constrain the adjacency matrices used for the other view. By iteratively applying these approach, the clustering of multiple views tends to agree with each other.

Subspace learning approaches are based on the assumption that multiple representations are generated from one common latent space. The goal is to extract shared latent subspace first and then conduct clustering. Subspace learning approaches include canonical correlation analysis (CCA) Johnson and Wichern (1988) based approaches, such as Chaudhuri et al. (2009), kernel variant kernel CCA Hardoon and Shawe-Taylor (2009); factorized orthogonal latent spaces Salzmann et al. (2010); and non-negative matrix factorization (NMF) approach Greene and Cunningham (2009), Gao et al. (2013) and etc.

3. OUR APPROACH

In this section, we present our multi-view clustering approach via diffusion on higher order graphs. Assume we have N examples with M representations (views). The task is to cluster examples into K clusters. In this paper, we assume that the number of clusters K is predefined by users.

The outline of this sections is as follows: In Sec. 3.1, we will introduce how to map single-view graphs to cross-view TPGs. Then a linear cross-view TPG fusion step will be described in Sec. 3.2. In Sec. 3.3 and 3.4, we will show how to perform multi-view TPG diffusion computational and memory efficiently. We will also prove our formulation is a multi-view generalization of single-view TPG diffusion method in Yang et al. (2013). In Sec. 3.5, to reveal the intuitive motivation behind TPG linear fusion step, we will discuss an interesting connection between TPG linear fusion step with Markov mixture models.

3.1. Single-view Graphs to Cross-view Tensor Product Graphs

A popular way to represent N data samples with single view $D = \{(x_i), i = 1, 2, \cdots, N\}$ is the edge-weight graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_N\}$ is the set of vertices representing the data samples and E is a set of edges. We consider the graph G to be a complete graph therefore $E = \{(E_{ij}), i, j = 1, 2, \cdots, N\}$ and $|E| = N^2$. The edge weight between two vertices encodes their pairwise data similarity (or equivalently affinity). We use matrix $A \in \mathbb{R}^{N\times N}$ to represent the edge weights. We let $A(i, j) = \phi(E_{ij})$ where $\phi(.)$ is the function that calculates weight of an edge given the two data samples x_i and x_j. The edge weights matrix A can be also called as similarity matrix or affinity matrix. Assume we are given N data samples with M views $D = \{(x_i^1, x_i^2, \cdots, x_i^M), i = 1, 2, \cdots, N\}$, where x_i^k

corresponds to the ith data sample in the kth view. We construct M edge-weight graphs with the data from each single view. We use $G_k = (V_k, E_k)$ to denote the single-view graph constructed from the kth view, and use A_k to denote its similarity matrix. Accordingly, the entry at ith row and jth column of adjacency matrix $A_k(i, j)$ represents edge weight between x_i^k and x_j^k.

Compared to original single-view graph, higher order graph can better capture intrinsic structure of data. Therefore, a key step in our approach is to map multiple original single-view graphs constructed from multi-view data to higher order cross-view TPGs.

Assume we have two graphs G_i and G_j, tensor product graph(TPG) $\mathbb{G}_{(i,j)} = G_i \bigotimes G_j$ is defined as $\mathbb{G}_{(i,j)} = (V_i \times V_j, \mathbb{E}_{(i,j)})$ whose similarity matrix is denoted by $\mathbb{A}_{(i,j)}$. Each vertex of graph $\mathbb{G}_{(i,j)}$ is a pair of vertices in graph G_i and G_j, and consequently, it is indexed with a pair of indices. $\mathbb{A}_{(i,j)} = A_i \bigotimes A_j$ is the Kronecker product of similarity matrices A_i and A_j. In particular, for $\alpha, \beta, \gamma, \delta = 1, 2, \cdots, N$, we also have

$$\mathbb{A}_{(i,j)}(\alpha\beta, \gamma\delta) = A_i(\alpha, \beta) \cdot A_j(\gamma, \delta) \tag{1}$$

Since the original graph has N vertices, the tensor product graph has $N \times N$ vertices, and $\mathbb{A}_{(i,j)} \in \mathbb{R}^{NN \times NN}$.

We compute the Kronecker product for each pair of single-view graphs to obtain a new set of cross-view TPGs $\{(\mathbb{G}_{(i,j)}), i, j = 1, 2, \cdots, M\}$, each of which encodes richer information than original graphs by capturing higher order similarity information for a pair of views.

To summarize, compared to other graph-based multi-view clustering algorithms, we not only have more graphs, $M \times M$ instead of M, but also have more informative graphs capturing higher order information. In the following section, we will introduce how the tensor product graphs are fused.

3.2. Cross-view Tensor Product Graphs Fusion by Linear Combination

Graph fusion approach has been used in many existing graph-based multi-view clustering methods. Of course, existing works work on the original similarity graph, therefore M single-view graphs need to be integrated. The most straight-forward fusion approach is to take uniform combination of multiple graphs, which will reduce M single-view graphs to a single average graph. Despite the simplicity of this fusion approach, it has shown promising result in most cases in Cortes et al. (2009).

In this work, we take the same graph fusion approach, the only difference is that instead of computing uniform combination of original single-view graphs, we combine $M \times M$ cross-view TPGs.

It seems to be more intuitive to assign different weights to different graphs when performing graph fusion. However, assigning weights is not a trivial task. Especially, considering we have an unsupervised scenario, i.e., solving clustering problem. Lack of supervision makes this task even more difficult.

In this work, our key contribution is to exploit higher order cross-view TPGs, therefore we make the graph fusion step as simple as possible by assigning the same weight to each cross-view TPGs.

In particular, we combine the information of multiple cross-view TPGs as follows:

$$\mathbb{A} = \sum_{i=1}^{M} \sum_{j=1}^{M} \mathbb{A}_{(i,j)} \tag{2}$$

In fact, the linear combination approach for graph fusion has a strong connection to markov mixture model, which has been proven in Zhou and Burges (2007). The fusion of cross-view TPGs also inherit that property, and can also be viewed as Markov mixture model. We will further elaborate that in Sec.3.5.

In the next section, we show how to discover the intrinsic clustering structure by a graph diffusion process on fusion adjacency $\mathbb{A}$. We also prove how to reduce the computation and memory complexity of the graph diffusion process so that it can be calculated efficiently.

3.3. Graph Diffusion Process on Fused Tensor Product Graph

As proved in Szummer and Jaakkola (2001) and Coifman and Lafon (2006), graph diffusion process is able to improve the similarities by exploring the intrinsic relation between data samples. The improved similarities have been shown to boost performance on various tasks, such as shape retrieval, image classification and etc. In this work we follow the definition of diffusion process in Yang et al. (2013). We calculate the tth iteration of **diffusion process on TPG** as follows:

$$\mathbb{A}^{(t)} = \sum_{k=1}^{t} (\sum_{i=1}^{M} \sum_{j=1}^{M} \mathbb{A}_{(i,j)})^k = \sum_{k=1}^{t} \mathbb{A}^k \tag{3}$$

As show in Yang et al. (2013), Eq. (3) is guaranteed to converge to a nontrivial solution given the sum of each row $\mathbb{A}$ is smaller than 1:

$$\mathbb{A}^* = \lim_{t\to\infty} \mathbb{A}^{(t)} = \lim_{t\to\infty} \sum_{k=1}^{t} \mathbb{A}^k = (I - \mathbb{A})^{-1}. \tag{4}$$

Consequently, Eq. (4) provides a closed form solution of the diffusion process on a tensor product graph. Compared to tensor graph $\mathbb{A}$, $\mathbb{A}^*$ has been much improved considering that the updated similarities captures intrinsic relations, such as data manifold.

However, our original goal is to perform clustering of the original data samples. This means we need to re-map the diffused tensor product graph back to original similarity graph. We adopt the same strategy as in Yang et al. (2013) to map tensor product graph to original graph, and derive an improved original adjacency matrix $A^* \in R^{N\times N}$ as follows:

$$A^* = vec^{-1}(\mathbb{A}^* vec(I)), \tag{5}$$

where I is a $N \times N$ identity matrix. The vec operator creates a column vector from a matrix by stacking the column vectors of the matrix below one another. More formally $vec : \mathbb{R}^{N\times N} \to \mathbb{R}^{NN\times 1}$. The inverse operator vec^{-1} maps a column vector into a matrix is often called the reshape operator, that is $vec^{-1} : \mathbb{R}^{NN\times 1} \to \mathbb{R}^{N\times N}$.

As demonstrated in Yang et al. (2013), Eq. (5) makes it possible to develop a computation and memory efficient iterative algorithm for tensor product graph diffusion. In order

to give some intuitive justification for Eq. (5), instead to $\mathbb{A}^*$, we apply this equation to $\mathbb{A}^{(1)} = \mathbb{A}$ (see Eq. (3)). Let $A = vec^{-1}(\mathbb{A}vec(I))$, which can be re-written as:

$$A(\alpha, \beta) = \sum_{\gamma=1}^{N} \mathbb{A}(\alpha\gamma, \beta\gamma) \tag{6}$$

If we substitute $\mathbb{A}$ according to Eq. (2), we finally have:

$$A(\alpha, \beta) = \sum_{\gamma=1}^{N} (\sum_{i=1}^{M} \sum_{j=1}^{M} \mathbb{A}_{(i,j)}(\alpha\gamma, \beta\gamma)) = \sum_{i=1}^{M} \sum_{j=1}^{M} (\sum_{\gamma=1}^{N} A_i(\alpha, \gamma) A_j(\beta, \gamma)) \tag{7}$$

One way to understand Eq. (7) is a kind of soft AND gate, in particular, nodes α and β have high affinity value if both have high affinity values to the same nodes γ under multiple views. In fact, this shows that we can examine the effect of Eq. (7) and diffusion separately. One natural baseline method derived from our proposed method is to compute the multi-view affinity graph using $\mathbb{A}^{(1)}$ and Eq. (7). Since this method does not utilize the diffusion process, we denote it as "Ours w/o diffusion". As shown in Table (2), this baseline method performs on par or slightly worse compared to other approaches. However, significant performance gain can be further achieved through diffusion (See "Ours").

To summarize, we first calculate $\mathbb{A}$ as the average of $M \times M$ tensor product graphs. Then a diffusion process is performed by simply solving Eq. (4). After $\mathbb{A}^*$ is obtained, we use Eq. (5) to map the tensor product graph back to original graph. Finally, we use standard graph embedding and k-means to get the final clustering results.

However, the diffusion process on tensor product graph is both memory and computation demanding, since it has $N^2 \times N^2$ vertices. In particular, the diffusion on tensor product graph fusion requires $O(N^4)$ storage and $O(N^6)$ computation cost. In next section, we show that instead of constructing and storing tensor product graphs with size $N^2 \times N^2$, diffusion process can be done on original graph whose size is $N \times N$. The memory and computation cost is therefore reduced to $O(N^2)$ and $O(N^3)$ respectively.

3.4. Efficient Iterative Algorithm for Graph Diffusion on $\mathbb{A}$

The proof is largely inspired by Yang et al. (2013). However, we want to point out the key difference between our work and Yang et al. (2013). In Yang et al. (2013), they always consider data with a single view. Consequently, the tensor product graph is constructed by simply calculating Kronecker product of exactly the same graph. However, in our work, we construct each cross-view TPG using a pair of original single-view graphs, which captures higher order information across a pair of views. Hence Yang et al. (2013) only derive the iterative algorithm that holds for a single TPG constructed as $\mathbb{A} = A \bigotimes A$. This is actually a special case of Eq. (2) when $M = 1$. We indeed generalize the proof that graph diffusion in a more general form $\mathbb{A} = \sum_{i=1}^{M} \sum_{j=1}^{M} A_i \bigotimes A_j$ can be done on original graph size.

We define $Q^1 = \sum_{i=1}^{M} A_i$ and

$$Q^t = (\sum_{i=1}^{M} A_i) Q^{t-1} (\sum_{j=1}^{M} A_j)^T + I \tag{8}$$

The proof of the convergence of Eq. (8) and a closed form expression for Q^* follow the following key equation:

$$\lim_{t\to\infty} Q^t = Q^* = A^* = vec^{-1}((I-\mathbb{A})^{-1}vec(I)) \tag{9}$$

Since $Q^* = A^*$, we conclude that the iterative algorithm defined by Eq. (8) produce the same results as the diffusion process on $\mathbb{A}$ in Eq. (3). The remainder of this section is to prove this equation.

According to Yang et al. (2013), if we row normalize $\sum_{i=1}^{M} A_i$ to make each row of $(\sum_{i=1}^{M} A_i) < 1$, the following identity holds:

$$\begin{aligned}\lim_{t\to\infty} vec(Q^{t+1}) &= \lim_{t\to\infty} vec(\sum_{k=1}^{t-1}(\sum_{i=1}^{M} A_i)^k I((\sum_{j=1}^{M} A_j)^T)^k) \\ &= \lim_{t\to\infty} \sum_{k=0}^{t-1}((\sum_{i=1}^{M} A_i)\bigotimes(\sum_{j=1}^{M} A_j))^k vec(I)\end{aligned} \tag{10}$$

Proposition 1

$$(\sum_{i=1}^{M} A_i)\bigotimes(\sum_{j=1}^{M} A_j) = \mathbb{A} \tag{11}$$

Proof The proof is trivial according to the bi-linear and associative property of the tensor product. Just for the paper completeness, we detail the proof below:

$$\begin{aligned}&(\sum_{i=1}^{M} A_i)\bigotimes(\sum_{j=1}^{M} A_j) = (A_1 + A_2 + \cdots + A_M)\bigotimes(\sum_{i=1}^{M} A_i) \\ =&A_1\bigotimes(\sum_{i=1}^{M} A_i) + \cdots + A_M\bigotimes(\sum_{i=1}^{M} A_i) \\ =&A_1\bigotimes(A_1 + A_2 + \cdots + A_M) + \cdots + A_M\bigotimes(A_1 + A_2 + \cdots + A_M) \\ =&\sum_{i=1}^{M}\sum_{j=1}^{M} A_i\bigotimes A_j = \mathbb{A}\end{aligned} \tag{12}$$

■

As a consequence of **Proposition 1**, Eq.(10) can be transformed to

$$\lim_{t\to\infty} vec(Q^{t+1}) = \lim_{t\to\infty}\sum_{k=0}^{t-1}(\mathbb{A})^k vec(I) = (I-\mathbb{A})^{-1}vec(I) \tag{13}$$

Finally, we have the optimal solution:

$$A^* = Q^* = \lim_{t\to\infty} vec^{-1}vec(Q^t) = vec^{-1}((I-\mathbb{A})^{-1}vec(I)) \tag{14}$$

A more detailed description of the proposed approach is given in Algorithm 1.

Input : 1. N examples with M representations.
2. The expected number of clusters K.
3. Parameters: $ITER$, p.

Output: Assignments to K clusters

Compute M affinity matrices, $A_1, A_2, \cdots, A_M$.
Let $A = \sum_{i=1}^{M} A_i$. Sparsify A by selecting pth nearest neighbors and row normalize A.
Denote $Q^1 = A$;
for $t = 2 : ITER$ **do**
| $Q^t = AQ^{t-1}A^T + I$;
end
Use $A^* = Q^T$ as the new graph similarities and compute the Laplacian, solve the largest K eigenvectors to obtain U.
Row-normalize U, assign example j to cluster c if the jth row of U is assigned to cluster c by the k-means algorithm.

Algorithm 1: Multi-view Clustering with Tensor Product Graph Diffusion

3.5. Relation to Markov Mixture Models

In this section, we will discuss the interesting connection between our tensor product graphs fusion step and Markov Mixture models. In Zhou and Burges (2007), the connection between the Markov mixture model and a linear combination of adjacency matrices for multiple undirected graphs has been discovered. Here, we extend this connection to tensor product graphs.

For each higher order graph $\mathbb{G}_{(i,j)}$, we associate it with a random walk. We denote the transition probabilities as $p_{(i,j)}$ and the stationary probabilities as $\pi_{(i,j)}$ for graph $\mathbb{G}_{(i,j)}$. As $\mathbb{G}_{(i,j)}$ is a undirected graph, we have $p_{(i,j)}(\alpha,\beta) = \mathbb{A}_{(i,j)}(\alpha,\beta)/d_{(i,j)}(\alpha)$ and $\pi_{(i,j)}(\alpha) = d_{(i,j)}(\alpha)/vol(\mathbb{A}_{(i,j)})$, where $d_{(i,j)}(\alpha) = \sum_{\beta=1}^{N^2} \mathbb{A}_{i,j}(\alpha,\beta)$, $vol(\mathbb{A}_{(i,j)}) = \sum_{\alpha=1}^{N^2} d_{(i,j)}(\alpha)$. All of these definitions are the same as those in Zhou and Burges (2007).

We define a mixture model of those random walks by

$$p(\alpha,\beta) = \sum_{i=1}^{M}\sum_{j=1}^{M} \gamma_{(i,j)}(\alpha) p_{(i,j)}(\alpha,\beta) \tag{15}$$

where

$$\gamma_{(i,j)}(\alpha) = \frac{\lambda_{(i,j)} \cdot \pi_{(i,j)}(\alpha)}{\sum_{i=1}^{M}\sum_{j=1}^{M} \lambda_{(i,j)} \cdot \pi_{(i,j)}(\alpha)} \tag{16}$$

Where $\lambda_{(i,j)}$ is the importance factor for $\mathbb{A}_{(i,j)}$ in Markov mixture models, and $\sum_{i=1}^{M}\sum_{j=1}^{M} \lambda_{(i,j)} = 1$, $\lambda_{(i,j)} \geq 0$.

The stationary distribution of random walk mixture is given by

$$\pi(\alpha) = \sum_{i=1}^{M}\sum_{j=1}^{M} \lambda_{(i,j)} \pi_{(i,j)}(\alpha) \tag{17}$$

Thus

$$\begin{aligned} p(\alpha,\beta) &= \sum_{i=1}^{M}\sum_{j=1}^{M} \gamma_{(i,j)}(\alpha) p_{(i,j)}(\alpha,\beta) \\ &= \sum_{i=1}^{M}\sum_{j=1}^{M} \frac{\lambda_{(i,j)} \cdot \mathbb{A}_{(i,j)}(\alpha,\beta)/vol(\mathbb{A}_{(i,j)})}{\sum_{i=1}^{M}\sum_{j=1}^{M} \lambda_{(i,j)} \cdot d_{(i,j)}(\alpha)/vol(\mathbb{A}_{(i,j)})} \end{aligned} \tag{18}$$

and $\pi(\alpha) = \sum_{i=1}^{M}\sum_{j=1}^{M} \lambda_{(i,j)} \cdot d_{(i,j)}(\alpha)/vol(\mathbb{A}_{(i,j)})$.

If we let

$$\mathbb{A}(\alpha,\beta) = \sum_{i=1}^{M}\sum_{j=1}^{M} \lambda_{(i,j)} \cdot \mathbb{A}_{(i,j)}(\alpha,\beta)/vol(\mathbb{A}_{(i,j)}) \tag{19}$$

and $d(\alpha) = \pi(\alpha)$, then we have $p(\alpha,\beta) = \mathbb{A}(\alpha,\beta)/d(\alpha)$.

In order to reveal the connection between the above Markov Mixture model Eq. (19) and our formulation of $\mathbb{A}$ in Eq. (2), we let

$$\lambda_{(i,j)} = \frac{vol(\mathbb{A}_{(i,j)})}{\sum_{i=1}^{M}\sum_{j=1}^{M} vol(\mathbb{A}_{(i,j)})} \tag{20}$$

Then we can rewrite Eq. (19) as:

$$\mathbb{A}(\alpha,\beta) = \frac{\sum_{i=1}^{M}\sum_{j=1}^{M} \mathbb{A}_{(i,j)}(\alpha,\beta)}{\sum_{i=1}^{M}\sum_{j=1}^{M} vol(\mathbb{A}_{(i,j)})} \tag{21}$$

We can see that Markov mixture model is in fact a linear combination of $\mathbb{A}_{(i,j)}$, same as our definition in Eq. (2) but with a constant scaling factor.

4. EMPIRICAL STUDY

In this section, we evaluate our approach on various benchmark datasets and compare its performance to several multi-view clustering algorithms. In particular, since the proposed method belongs to graph-based category, we focus on comparing with several multi-view spectral clustering algorithms:

- **Single View**: We construct the Laplacian graph for each view separately and compute the top K eigenvectors to obtain U, then we perform k-means on each U. We report the results for both the **best single view** as well as the **worst single view**.
- **Feature Concatenation (Feature Concat)**: concatenating the features of each view followed by spectral clustering using the Laplacian graph derived from the joint view representation.
- **Kernel Addition**: We combine different affinity matrices by adding them together and then running standard spectral clustering on the corresponding Laplacian graph. This simple approach can achieve near optimal results when compared to more sophisticated approaches as stated in Cortes et al. (2009).

Table 1: Statistics of benchmark datasets

dataset	instances	views	clusters
BBC	2225	2	5
BBCSport	737	4	5
UCI Digits	2000	5	10
Flowers	1360	7	17

- **Spectral Co-training (Co-train Spec)**: The approach proposed in Kumar and Daumé III (2011) that utilizes the spectral embedding from one view to constrain the affinity graph used for the other view. By iteratively applying this approach, the clustering of the two views tend to agree with each other. We set the iteration number to 10 and report the best results in these 10 iterations.

- **Co-regularized spectral clustering (Co-Reg)**: The co-regularization methods propose novel spectral clustering objective functions that implicitly combine graphs from multiple kernels (or affinity matrices) for the clustering problem Kumar et al. (2011). We run the centroid-based co-regularization approach in this paper and set parameter λ to 0.3.

4.1. Datasets

We report the experimental results on four real-world datasets: BBC and BBCSport [1] for news article clustering. UCI digits [2], Flower17 [3]for image clustering. For UCI digits, we use the following 5 features: 76 Fourier coefficients of the character shapes, 216 profile correlations, 240 pixel averages in 2 x 3 windows, 47 Zernike moments and 6 morphological features. For Flower17: we directly use the seven pre-computed distance matrices included in the dataset as input for clustering. A more detailed description of these datasets are summarized in Table 1.

We use the absolute value of cosine similarity to construct the affinity matrix A_k for the kth view as follows. The edge weight $A_k(i,j)$ between example i and j is defined by the absolute value of their cosine affinity. If edge weight approaches 0, the amount of affinity is small; if edge weight approaches either $+1$ or -1, there is more of affinity between two examples.

$$A_k(i,j) = |\frac{x_i^k \cdot x_i^k}{\|x_j^k\| \cdot \|x_i^k\|}| \tag{22}$$

It is easy to see that each affinity matrix A_k is a symmetric matrix. In our experiment, to guarantee a fair comparison, the same affinity matrices are used in all graph-based multi-view clustering approaches.

We use six metrics to measure the clustering performances: precision, recall, F-score, normalized mutual information(NMI), average entropy, and adjusted rand index(Adj-RI Manning et al. (2008)). Higher value means better clustering quality with regard to pre-

1. http://mlg.ucd.ie/datasets
2. https://archive.ics.uci.edu/ml/datasets/Multiple Features
3. http://www.robots.ox.ac.uk/ vgg/data/flowers/17

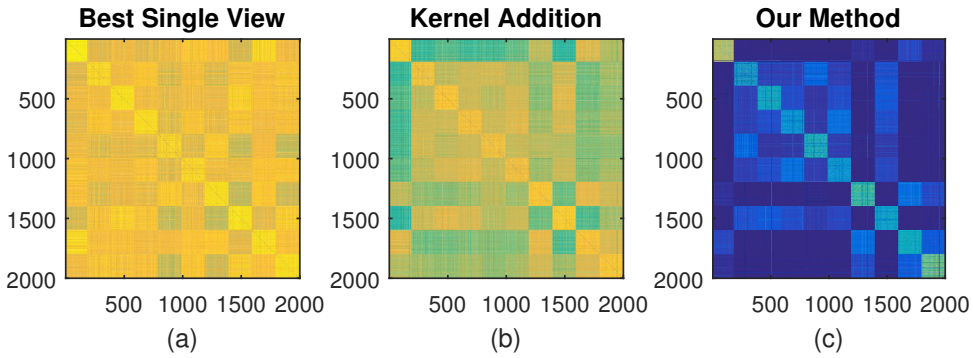

Figure 2: Affinity matrices for different baselines for dataset UCI Digit. (a) The matrix corresponds to the best single view. (b) The matrix corresponds to kernel addition. (c) The matrix after diffusion on higher order graphs, where the structure of the ten clusters is clearly visible.

cision, recall, F-score, NMI, and Adj-RI. Lower value means better clustering quality for average entropy.

4.2. Results Analysis

We first show the difference among affinity matrices in different approaches in Fig 2. It is easy to notice that there are some cluster structures in graphs for the best single view and kernel addition. However, the boundary is very blurry meaning that the similarities are not very distinctive, which greatly degrades the clustering performance. Our approach can reveal the true cluster structures with less noise.

The clustering results are shown in Table 2. It is easy to observe that our approach achieves better performance than the baselines with regard to all the six metrics. The following text gives some statistics:

For UCI digits with five views, our approach has an increase of 0.242 , 0.256 and 0.267 with regard to F-score, NMI and Adj-RI when compared to the best single view; meanwhile, our approach has an improvement of 0.042, 0.092 and 0.044 with respect to F-score, NMI and Adj-RI when compared to the second best baseline.

For Flowers17 with seven views, our approach has an increase of 0.131 , 0.134 and 0.139 with regard to F-score, NMI and Adj-RI when compared to the best single view; meanwhile, our approach has an improvement of 0.0.010, 0.019 and 0.010 with respect to F-score, NMI and Adj-RI when compared to the second best baseline.

For BBCSport with four views, our approach has an increase of 0.177 , 0.199 and 0.226 with regard to F-score, NMI and Adj-RI when compared to the best single view; meanwhile our approach has an improvement of 0.018, 0.016 and 0.024 with respect to F-score, NMI and Adj-RI when compared to the second best baseline.

In summary, our approach shows encouraging performance gains over the state-of-the-art approaches.

Table 2: Comparison of clustering results on benchmark datasets. On each dataset, we run k-means 20 times with different random initializations. The average clustering performance as well as the standard deviation (number in parentheses) are reported. Best scores are in bold font.

dataset	method	F-score	Precision	Recall	Entropy	NMI	Adj-RI
BBC	Worst Single View	0.72(0.05)	0.71(0.06)	0.73(0.04)	0.81(0.10)	0.65(0.04)	0.65(0.07)
	Best Single View	0.79(0.07)	0.78(0.08)	0.80(0.05)	0.63(0.14)	0.73(0.06)	0.74(0.09)
	Feature Concat	0.85(0.04)	0.85(0.05)	0.85(0.01)	0.48(0.08)	0.79(0.03)	0.81(0.05)
	Kernel Addition	0.84(0.05)	0.84(0.06)	0.85(0.03)	0.50(0.09)	0.79(0.03)	0.80(0.06)
	Co-train Spec	0.86(0.02)	0.86(0.01)	0.87(0.03)	0.45(0.10)	0.81(0.05)	0.83(0.04)
	Co-Reg	0.83(0.06)	0.83(0.07)	0.84(0.04)	0.51(0.11)	0.78(0.04)	0.80(0.07)
	Ours w/o diffusion	0.84(0.05)	0.84(0.11)	0.84(0.01)	0.48(0.12)	0.78(0.05)	0.81(0.09)
	Ours	**0.89(0.07)**	**0.89(0.10)**	**0.90(0.02)**	**0.36(0.13)**	**0.84(0.04)**	**0.87(0.09)**
BBCSport	Worst Single View	0.45(0.01)	0.40(0.03)	0.53(0.06)	1.22(0.03)	0.44(0.02)	0.25(0.02)
	Best Single View	0.54(0.02)	0.58(0.03)	0.50(0.02)	0.96(0.07)	0.52(0.03)	0.41(0.03)
	Feature Concat	0.64(0.02)	0.66(0.02)	0.61(0.01)	0.70(0.02)	0.65(0.01)	0.53(0.02)
	Kernel Addition	0.70(0.06)	0.75(0.09)	0.65(0.04)	0.56(0.13)	0.71(0.05)	0.61(0.09)
	Co-train Spec	0.70(0.04)	0.76(0.05)	0.65(0.04)	0.61(0.09)	0.69(0.04)	0.61(0.05)
	Co-Reg	0.62(0.05)	0.64(0.08)	0.60(0.03)	0.76(0.10)	0.63(0.04)	0.50(0.08)
	Ours w/o diffusion	0.65(0.05)	0.73(0.09)	0.59(0.05)	0.54(0.05)	0.66(0.03)	0.56(0.07)
	Ours	**0.72(0.06)**	**0.78(0.10)**	**0.67(0.04)**	**0.53(0.12)**	**0.74(0.05)**	**0.63(0.09)**
Flower17	Worst Single View	0.15(0.01)	0.14(0.01)	0.15(0.01)	3.13(0.03)	0.24(0.01)	0.09(0.01)
	Best Single View	0.30(0.01)	0.30(0.01)	0.31(0.01)	2.30(0.04)	0.44(0.01)	0.26(0.01)
	Kernel Addition	0.37(0.01)	0.36(0.01)	0.37(0.01)	2.03(0.04)	0.51(0.01)	0.33(0.02)
	Co-train Spec	0.38(0.02)	0.37(0.02)	0.39(0.02)	1.97(0.04)	0.52(0.01)	0.34(0.02)
	Co-Reg	0.42(0.01)	**0.42(0.02)**	0.43(0.01)	1.81(0.04)	0.56(0.01)	0.39(0.01)
	Ours w/o diffusion	0.36(0.01)	0.34(0.02)	0.37(0.02)	2.03(0.44)	0.52(0.01)	0.35(0.01)
	Ours	**0.43(0.02)**	0.41(0.02)	**0.46(0.01)**	**1.75(0.05)**	**0.58(0.01)**	**0.40(0.02)**
UCI digits	Worst Single View	0.28(0.02)	0.28(0.02)	0.29(0.02)	2.18(0.04)	0.34(0.01)	0.20(0.02)
	Best Single View	0.61(0.03)	0.59(0.04)	0.62(0.03)	1.22(0.08)	0.64(0.02)	0.56(0.04)
	Feature Concat	0.59(0.03)	0.58(0.03)	0.59(0.03)	1.27(0.08)	0.62(0.02)	0.54(0.04)
	Kernel Addition	0.74(0.04)	0.73(0.04)	0.76(0.03)	0.76(0.08)	0.77(0.02)	0.72(0.04)
	Co-train Spec	0.72(0.05)	0.70(0.05)	0.74(0.04)	0.87(0.10)	0.74(0.03)	0.69(0.05)
	Co-Reg	0.81(0.05)	**0.80(0.06)**	0.83(0.03)	0.62(0.11	0.82(0.03)	0.79(0.06)
	Ours w/o diffusion	0.73(0.03)	0.73(0.05)	0.73(0.02)	0.68(0.15)	0.74(0.03)	0.66(0.08)
	Ours	**0.85(0.07)**	0.79(0.11)	**0.93(0.01)**	**0.43(0.15)**	**0.89(0.03)**	**0.83(0.08)**

4.3. Parameter Sensitivity

In this section, we discuss the parameter choices in our approach. Given similarity matrices, there is only one parameter in our approach: number of diffusion iterations $ITER$. We conduct experiment on UCI digits to see whether the performance of our approach is sensitive to number of diffusion iterations.

We summarize the results in Fig 3. We can see that when $ITER >= 3$, the average NMI for different p can achieve a very stable and good results.

We also look into the graph sparsity parameter p which has to be set in every spectral clustering algorithms. Our approach obtain similar results as the reported one when p varys

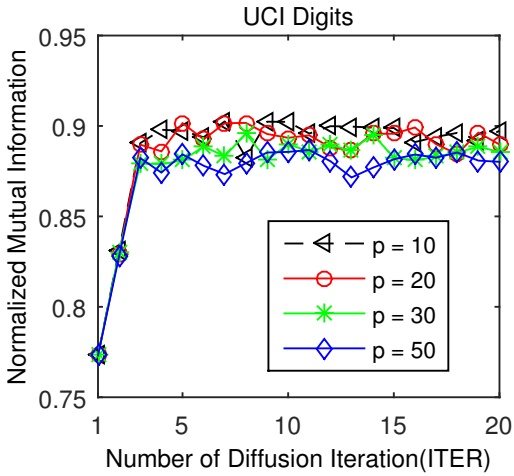

Figure 3: Each line plot the average NMI with varying number of diffusion iterations. For each number of iterations, we run k-means 20 times with different random initialization and report average NMI . Four different lines represent 'Average NMI vs Number of Diffusion Iterations' with different sparsity parameter $p = 10, 20, 30, 50$ respectively.

from 10 to 30. When $p = 50$, the performance degrades a little bit but is still better than that of other baselines. In our experiment, we set $p = 20$ and $ITER = 50$ for all the result we reported in Table(2).

In summary, our approach is not very sensitive to the parameters. Therefore, even without much parameter tuning, our approach can obtain much better results than other multi-view spectral clustering algorithms.

5. CONCLUSIONS

We demonstrate that exploring higher order information of graphs can be a very successful tool for multi-view clustering. Our approach is able to integrate information from multiple representations efficiently. To achieve this goal, we first map original view graphs to a set of cross-view TPGs to capture higher order information; then we combine the set of cross-view TPGs uniformly. We prove that our combination is actually a mixture Markov model on higher order graphs other than random. After that, we run graph diffusion on the sum of cross-view TPG to discover intrinsic clustering structure. The effectiveness of our method is validated on several benchmark datasets for news article clustering and image clustering. Empirical results show that the proposed method achieves robust performance and outperforms state-of-the-art methods.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants No. IIS-1302164 and OIA-1027897.

References

Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, 2004.

Steffen Bickel and Tobias Scheffer. Multi-view clustering. In *ICDM*, pages 19–26, 2004.

Xiao Cai, Feiping Nie, and Heng Huang. Multi-view k-means clustering on big data. In *IJCAI*, 2013.

Kamalika Chaudhuri, Sham M. Kakade, Karen Livescu, and Karthik Sridharan. Multi-view clustering via canonical correlation analysis. In *ICML*, pages 129–136, 2009.

Ronald R. Coifman and Stphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5 – 30, 2006.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Learning non-linear combinations of kernels. In *NIPS*, pages 396–404, 2009.

Ying Cui, Xiaoli Z. Fern, and Jennifer G. Dy. Non-redundant multi-view clustering via orthogonalization. In *ICDM*, pages 133–142, 2007.

Ian Davidson, Buyue Qian, Xiang Wang, and Jieping Ye. Multi-objective multi-view spectral clustering via pareto optimization. In *SDM*, pages 234–242, 2013.

Virginia R. de Sa. Spectral clustering with two views. In *Proceedings of the International Conference on Machine Learning Workshop on Learning with Multiple Views*, 2005.

Eric Eaton, Marie desJardins, and Sara Jacob. Multi-view clustering with constraint propagation for learning with an incomplete mapping between views. In *CIKM*, pages 389–398, 2010.

Jing Gao, Jiawei Han, Jialu Liu, and Chi Wang. Multi-view clustering via joint nonnegative matrix factorization. In *SDM*, pages 252–260, 2013.

Derek Greene and Padraig Cunningham. A matrix factorization approach for integrating multiple data views. In *ECML-PKDD*, pages 423–438, 2009.

David R. Hardoon and John Shawe-Taylor. Convergence analysis of kernel canonical correlation analysis: theory and practice. *Machine Learning*, 74(1):23–38, 2009.

R. A. Johnson and D. W. Wichern, editors. *Applied Multivariate Statistical Analysis*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

Abhishek Kumar and Hal Daumé III. A co-training approach for multi-view spectral clustering. In *ICML*, pages 393–400, 2011.

Abhishek Kumar, Piyush Rai, and Hal Daumé III. Co-regularized multi-view spectral clustering. In *NIPS*, pages 1413–1421, 2011.

Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semi-definite programming. In *ICML*, pages 323–330, 2002.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290:2323–2326, 2000.

Mathieu Salzmann, Carl Henrik Ek, Raquel Urtasun, and Trevor Darrell. Factorized orthogonal latent spaces. In *AISTATIS*, pages 701–708, 2010.

Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *NIPS*, pages 945–952, 2001.

Grigorios Tzortzis and Aristidis Likas. Kernel-based weighted multi-view clustering. In *ICDM*, pages 675–684, 2012.

Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4): 395–416, 2007.

Hua Wang, Feiping Nie, and Heng Huang. Multi-view clustering and feature learning via structured sparsity. In *ICML*, pages 352–360, 2013.

Rongkai Xia, Yan Pan, Lei Du, and Jian Yin. Robust multi-view spectral clustering via low-rank and sparse decomposition. In *AAAI*, pages 2149–2155, 2014.

Xingwei Yang, Lakshman Prasad, and Longin Jan Latecki. Affinity learning with diffusion on tensor product graph. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):28–38, 2013.

Dengyong Zhou and Christopher J. C. Burges. Spectral clustering and transductive learning with multiple views. In *ICML*, pages 1159–1166, 2007.

Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS*, pages 1601–1608, 2006.