Cluster Ensembles via Weighted Graph Regularized Nonnegative Matrix Factorization

Liang Du[1,2], Xuan Li[1,2], and Yi-Dong Shen[1]

[1] State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[2] Graduate University, Chinese Academy of Sciences, Beijing 100049, China
{duliang,lixuan,ydshen}@ios.ac.cn

Abstract. Cluster ensembles aim to generate a stable and robust consensus clustering by combining multiple different clustering results of a dataset. Multiple clusterings can be represented either by multiple co-association pairwise relations or cluster based features. Traditional clustering ensemble algorithms learn the consensus clustering using either of the two representations, but not both. In this paper, we propose to integrate the two representations in a unified framework by means of weighted graph regularized nonnegative matrix factorization. Such integration makes the two representations complementary to each other and thus outperforms both of them in clustering accuracy and stability. Extensive experimental results on a number of datasets further demonstrate this.

Keywords: Cluster Ensembles, Weighted Graph Regularization, NMF.

1 Introduction

It is well recognized that different clustering methods may produce different clustering results on a given data set. The reason for this is that each clustering algorithm has its own bias resulting from different criteria. Therefore, cluster ensembles have emerged in recent years as a technique to overcome such problems [1,2,3,4]. This technique is also known as clustering ensemble [5], clustering aggregation [6] or consensus clustering [7]. It reconciles multiple clustering results of a data set into a single consolidated clustering using a consensus function.

Cluster ensembles contain two key components, i.e. a representation of input clusterings and a consensus function. Two representations are currently widely used; one is *multiple co-association matrices* [2,4] and the other *cluster based features* [1]. To the best of our knowledge, current cluster ensembles approaches learn their consensus functions using either of the two representations, but not both. On the one hand, the multiple co-association matrices contain pairwise relationships of different clusterings and the averaged co-association matrix can be seen as the consensus similarity of input clusterings. On the other hand, a dataset can be represented using cluster based features, which contain connections between data points and clusters. Although both the co-association matrices and

J. Tang et al. (Eds.): ADMA 2011, Part I, LNAI 7120, pp. 215–228, 2011.

cluster based features can be used independently to obtain consensus clustering, algorithms that make use of them simultaneously should be able to generate more meaningful clustering structures; similar idea has also been explored in clustering problem [8].

Besides, each partition may have different clustering performance. It is also required to automatically identify better clusterings from all partitions. One natural idea is to assign different weights to input clusterings based on there contributions for the cluster ensembles [2,4].

Based on the above observations, in this paper we propose a *Weighted Graph regularized Nonnegative Matrix Factorization* (WGNMF) model for cluster ensembles by integrating the above two representations into a unified framework and learn the weights of different clusterings. The basic idea is as follows: We construct two matrices; one is a *consensus affinity matrix* from multiple co-association matrices and the other is a *cluster feature matrix* from cluster based features. These two matrices are used in a regularization process, where we learn an implicit consensus function from the cluster feature matrix by nonnegative matrix factorization (NMF) while the factorization procedure is regularized with the consensus affinity matrix. In our WGNMF model, the weights of input clusterings are learned during the factorization process. We empirically evaluate WGNMF model over benchmark data sets, which demonstrates that it outperforms existing approaches using only one of the above two representations.

The rest of paper is organized as follows. Section 2 reviews related work on cluster ensembles. Section 3 presents our WGNMF model. Section 4 reports the experimental results, and Section 5 concludes the paper with some future work.

2 Related Work

We briefly review some related work on cluster ensembles. Previous work on learning consensus functions are mainly based on the two representations of input clusterings: the multiple co-association matrices and the cluster based features.

One popular strategy of building consensus functions is to utilize multiple co-association matrices [9] (also known as connectivity matrices in [7,2,4]). The averaged co-association matrix coming from input clusterings can be seen as a new data matrix in a new feature space or a similarity matrix, and thus traditional approaches operated on data matrix or similarity can be deployed. Hadjitodorov et al. [10] showed that good clustering results can be obtained by running Kmeans on the averaged co-association matrix. Li et al. used nonnegative matrix factorization (NMF) on the averaged co-association matrix to generate the final consensus clustering. Li and Ding [2] proposed to learn the weighted co-association matrix within an NMF procedure. Wang et al. proposed generalized weighted cluster aggregation (GWCA) [4] to learn the consensus function by minimizing the sum of the Bregman divergence between the consensus function and all of the co-association matrices.

Another different strategy is to construct consensus functions implicitly from the representation of cluster based features. These implicit consensus functions

can be roughly grouped in two categories. The first category is graph based approaches in which a graph is constructed from the cluster based features and some graph partition algorithms are used for the final consensus clustering. Strehl et.al. [1] proposed three graph-based approaches: Cluster-based Similarity Partition Algorithm (CSPA), HyperGraph Partition Algorithm (HGPA), and MetaClustering Algorithm (MCLA). In CSPA, A binary similarity matrix is constructed and specific algorithms like METIS [11] is used to partition the graph. HGPA utilizes the HMETIS [11] algorithm to partition the hypergraph where each hyperedge represents a cluster of an input clusterings. MCLA collapses related hyperedges and assigns each object to the collapsed hyperedge in which it participates most strongly. Fern and Brodley [12] proposed the hybrid bipartite graph partition algorithm, which partitions the bipartite graph using spectral graph partition algorithms. Al-Razgan and Domeniconi [9] proposed an approach to partitioning a weighted similarity graph. The second category makes use of probabilistic graphical models. Topchy et al. [13] proposed a probabilistic model using a finite mixture of multinomial distributions in the space of input clusterings. Wang et al. [3] proposed an generative probabilistic model Bayesian cluster ensembles (BCE) for cluster ensembles, which is derived from Latent Dirichlet Allocation (LDA) [14].

3 WGNMF

We first introduce some notation and briefly review NMF [15] which is used to learn the consensus function implicitly form cluster based representation. We then describe our WGNMF model which integrates multiple co-association matrices and the representation of cluster based features within the process of NMF. We also present specific optimization techniques for WGNMF.

3.1 Notation

Given a data set of n points and a collection of m clustering solutions $\boldsymbol{\mathcal{P}} = \{\mathcal{P}^1, \mathcal{P}^2, \ldots, \mathcal{P}^m\}$. Each clustering solution $\mathcal{P}^c$ for $c = 1, \ldots, m$ is a partition of the data set. A partitioning of these n points into k clusters can be represented as a set of k clusters $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ or a label vector $\mathcal{P}^c \in \mathbb{R}^n$, where k is the number of clusters. Note that the number of clusters k in different $\mathcal{P}^c$ could be different.

The cluster-based representation [1] can be constructed as follows. For each clustering $\mathcal{P}^c$, we construct the binary membership indicator matrix $\boldsymbol{H}^c \in \mathcal{R}^{n \times k}$, where each column corresponds a cluster and each row is a point. $H^c_{ij} = 1$ if the point i is assigned to cluster j in partition c, and $H^c_{ij} = 0$ otherwise. The concatenated matrix $\boldsymbol{X} = (\boldsymbol{H}^1, \boldsymbol{H}^2, \ldots, \boldsymbol{H}^m)$ is used to represent the data matrix in a new feature space.

The co-association matrix [7] of partition $\mathcal{P}^c$ is defined as a $n \times n$ squared matrix $\boldsymbol{W}^c$, where $\boldsymbol{W}^c_{ij} = 1$ if points i and j are assigned to the same cluster, and $\boldsymbol{W}^c_{ij} = 0$ otherwise.

Example 1. Suppose we have 5 samples and 2 clusterings each with 3 clusters. Then we have 2 partitions $\mathcal{P}^1 = [1, 1, 2, 3, 3]$ and $\mathcal{P}^2 = [2, 3, 3, 1, 1]$. The cluster-based representation $\boldsymbol{X}$ and the co-association matrices $\boldsymbol{W}^1$ for $\mathcal{P}^1$ and $\boldsymbol{W}^2$ for $\mathcal{P}^2$ of these samples are given below.

$$\boldsymbol{X} = \begin{bmatrix} 1\,0\,0\,0\,1\,0 \\ 1\,0\,0\,0\,0\,1 \\ 0\,1\,0\,0\,0\,1 \\ 0\,0\,1\,1\,0\,0 \\ 0\,0\,1\,1\,0\,0 \end{bmatrix} \quad \boldsymbol{W}^1 = \begin{bmatrix} 1\,1\,0\,0\,0 \\ 1\,1\,0\,0\,0 \\ 0\,0\,1\,0\,0 \\ 0\,0\,0\,1\,1 \\ 0\,0\,0\,1\,1 \end{bmatrix} \quad \boldsymbol{W}^2 = \begin{bmatrix} 1\,0\,0\,0\,0 \\ 0\,1\,1\,0\,0 \\ 0\,1\,1\,0\,0 \\ 0\,0\,0\,1\,1 \\ 0\,0\,0\,1\,1 \end{bmatrix}$$

3.2 NMF and Extensions

Non-negative Matrix Factorization [15] is a factorization algorithm focused on the analysis of nonnegative matrix. Given a nonnegative matrix $\boldsymbol{X} \in \mathcal{R}_+^{n \times d}$, where each row of $\boldsymbol{X}$ is a data point, NMF approximates $\boldsymbol{X}$ using two low-rank nonnegative matrices $\boldsymbol{U} \in \mathcal{R}_+^{n \times k}$ and $\boldsymbol{V} \in \mathcal{R}_+^{d \times k}$. There are two cost functions commonly used to measure the quality of the approximation. The first one is the square of the Euclidean distance between two matrices

$$O_1 = ||\boldsymbol{X} - \boldsymbol{U}\boldsymbol{V}^T||_F^2 \tag{1}$$

where $||\cdot||_F$ is Frobenius norm. The second one is the divergence between two matrices

$$O_2 = D(\boldsymbol{X}||\boldsymbol{U}\boldsymbol{V}^T) = \sum_{i=1}^{n}\sum_{j=1}^{d}(\boldsymbol{X}_{ij}\log\frac{\boldsymbol{X}_{ij}}{\boldsymbol{Y}_{ij}} - \boldsymbol{X}_{ij} + \boldsymbol{Y}_{ij}) \tag{2}$$

where $\boldsymbol{Y}_{ij} = \boldsymbol{U}_i\boldsymbol{V}_j^T$.

Recently, Cai et al. [16] proposed Graph regularized NMF (GNMF). It aims at learn a compact representation which uncovers the hidden semantics and simultaneously preserves the intrinsic local geometric structure. The basic assumption behind is that if two data points are similar, then their low dimensional representations are also close to each other.

3.3 Weighted Graph Regularized NMF

In subsection 3.1, we obtained a matrix $\boldsymbol{X} = (\boldsymbol{H}^1, \boldsymbol{H}^2, \ldots, \boldsymbol{H}^m)$ from the cluster based representation of input clusterings. Here, we use NMF to find two low-rank matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ to approximate it. Due to the close connection between NMF and clustering, the obtained $\boldsymbol{U}$ can be used to extract the final consensus clustering.

To obtain a better approximation of $\boldsymbol{X}$ by $\boldsymbol{U}$ and $\boldsymbol{V}$, inspired by [16], we incorporate the multiple co-association matrices $\{\boldsymbol{W}^c\}_{c=1}^m$ into NMF. To apply this idea in cluster ensembles, we define a consensus graph, where the affinity matrix $\hat{\boldsymbol{W}}$ is constructed by linearly combining the given multiple co-association matrices

$$\hat{\boldsymbol{W}} = \sum_{c}^{m} \alpha_c \boldsymbol{W}^c \quad \sum_{c=1}^{m} \boldsymbol{\alpha}_c = 1, \boldsymbol{\alpha}_c \geq 0$$

where $\boldsymbol{\alpha}_c, c = 1, 2, \ldots, m$, is the weight associated with partition $\mathcal{P}^c$. Recall that the co-association matrix of $\boldsymbol{W}^c$ for $\mathcal{P}^c$ can be seen as a binary similarity matrix, and it defines a coarse affinity graph. We expect that the weighed combination of multiple co-association matrices can better capture the similarities between data points. The learned weights also provide clues to select individual input cluterings. The reason is that an input clustering with larger weight contributes more to the consensus affinity graph and the final clustering.

Given the above definition of consensus affinity graph, we can use the following two functions to measure the smoothness of the low dimensional representation of data points:

$$\begin{aligned}
\mathcal{R}_1 &= \frac{1}{2} \sum_{c=1}^{m} \sum_{i,j} \boldsymbol{\alpha}_c \boldsymbol{W}_{ij}^c ||\boldsymbol{U}_i - \boldsymbol{U}_j||^2 \\
&= \sum_{c=1}^{m} \boldsymbol{\alpha}_c \left(\sum_i \boldsymbol{D}_{ii}^c \boldsymbol{U}_i \boldsymbol{U}_i^T - \sum_{i,j} \boldsymbol{W}_{ij}^c \boldsymbol{U}_i \boldsymbol{U}_j^T \right) \\
&= \sum_{c=1}^{m} \boldsymbol{\alpha}_c \left(\mathrm{tr}(\boldsymbol{U}^T \boldsymbol{D}^c \boldsymbol{U}) - \mathrm{tr}(\boldsymbol{U}^T \boldsymbol{W}^c \boldsymbol{U}) \right) \\
&= \mathrm{tr}(\boldsymbol{U}^T (\sum_{c=1}^{m} \boldsymbol{\alpha}_c \boldsymbol{L}^c) \boldsymbol{U})
\end{aligned} \tag{3}$$

and

$$\begin{aligned}
\mathcal{R}_2 &= \frac{1}{2} \sum_{c=1}^{m} \sum_{i,j} \boldsymbol{\alpha}_c \boldsymbol{W}_{ij}^c \left(D(\boldsymbol{U}_i || \boldsymbol{U}_j) + D(\boldsymbol{U}_j || \boldsymbol{U}_i) \right) \\
&= \frac{1}{2} \sum_{c=1}^{m} \sum_{i,j} \sum_{l} \boldsymbol{\alpha}_c \boldsymbol{W}_{ij}^c (\boldsymbol{U}_{il} \log \frac{\boldsymbol{U}_{il}}{\boldsymbol{U}_{jl}} + \boldsymbol{U}_{jl} \log \frac{\boldsymbol{U}_{jl}}{\boldsymbol{U}_{il}})
\end{aligned} \tag{4}$$

where $\mathrm{tr}(\cdot)$ denotes the trace of a matrix and $\boldsymbol{D}$ is a diagonal matrix with $\boldsymbol{D}_{ii}^c = \sum_j \boldsymbol{W}_{ij}^c$. $\boldsymbol{L}^c = \boldsymbol{D}^c - \boldsymbol{W}^c$ is called the graph Laplacian.

Combining the weighted affinity graph together with the above smoothness functions with NMF leads to our Weighted Graph regularized Non-negative Factorization (WGNMF) model: If the Euclidean distances is used, WGNMF minimizes the following objective function:

$$\begin{aligned}
\mathcal{O}_1 = \quad & ||\boldsymbol{X} - \boldsymbol{U}\boldsymbol{V}^T||^2 + \lambda \mathrm{tr}(\boldsymbol{U}^T (\sum_{c=1}^{m} \boldsymbol{\alpha}_c \boldsymbol{L}^c) \boldsymbol{U}) \\
\text{s.t.} \quad & \boldsymbol{U} \geq 0, \boldsymbol{V} \geq 0, \\
& \sum_{c=1}^{m} \boldsymbol{\alpha}_c = 1, \boldsymbol{\alpha}_c \geq 0
\end{aligned} \tag{5}$$

If the divergence is used, WGNMF minimizes the following objective function:

$$\begin{aligned}\mathcal{O}_2 = &\sum_{i=1}^{n}\sum_{j=1}^{d}(\boldsymbol{X}_{ij}\log\frac{\boldsymbol{X}_{ij}}{\sum_{l=1}^{k}\boldsymbol{U}_{il}\boldsymbol{V}_{jl}} - \boldsymbol{X}_{ij} + \sum_{l=1}^{k}\boldsymbol{U}_{il}\boldsymbol{V}_{jl}) \\ &+ \frac{\lambda}{2}\sum_{c=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{d}\sum_{l=1}^{k}\boldsymbol{\alpha}_c\boldsymbol{W}^c_{ij}(\boldsymbol{U}_{il}\log\frac{\boldsymbol{U}_{il}}{\boldsymbol{U}_{jl}} + \boldsymbol{U}_{jl}\log\frac{\boldsymbol{U}_{jl}}{\boldsymbol{U}_{il}}) \\ \text{s.t.}\quad &\boldsymbol{U}\geq 0, \boldsymbol{V}\geq 0, \\ &\sum_{c=1}^{m}\boldsymbol{\alpha}_c = 1, \boldsymbol{\alpha}_c \geq 0 \end{aligned} \tag{6}$$

where the regularization parameter $\lambda \geq 0$ controls the smoothness of the low dimensional representations.

3.4 Optimization

In $\mathcal{O}_1$ (Eq. (5)) and $\mathcal{O}_2$ (Eq. (6)), there are 3 unknown variables, $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{\alpha}$. When two of them are fixed, the subproblem of computing the optimal value for the other variable is easy to solve. Hence, $\mathcal{O}_1$ and $\mathcal{O}_2$ can be solved by iteratively updating $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{\alpha}$, so that the value of the objective functions gradually decrease. This process can be viewed as a "block coordinate descent" process [17]. We describe this process for $\mathcal{O}_1$ and $\mathcal{O}_2$ separately.

Minimizing $\mathcal{O}_1$. When $\boldsymbol{\alpha}$ is fixed, the optimization problem of computing $\boldsymbol{U}$, $\boldsymbol{V}$ becomes

$$\begin{aligned}\mathcal{L} = &\text{tr}(\boldsymbol{X}\boldsymbol{X}^T) - 2\text{tr}(\boldsymbol{X}\boldsymbol{V}\boldsymbol{U}^T) + \text{tr}(\boldsymbol{U}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{U}^T) \\ &+ \lambda\sum_{c=1}^{m}\boldsymbol{\alpha}_c\text{tr}(\boldsymbol{U}^T\boldsymbol{L}^c\boldsymbol{U}) + \text{tr}(\boldsymbol{\Phi}\boldsymbol{U}^T) + \text{tr}(\boldsymbol{\Psi}\boldsymbol{V}^T)\end{aligned} \tag{7}$$

where $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are the lagrange multiplier for the nonnegative constraints.

Notice that the above optimization problem is equivalent to GNMF [16] with a combined Laplacian matrix. Thus, the estimation for $\boldsymbol{U}$ and $\boldsymbol{V}$ will be exactly same as that in GNMF with a combined Laplacian matrix. It is [16]:

$$\boldsymbol{U}_{il} = \boldsymbol{U}_{il}\frac{(\boldsymbol{X}\boldsymbol{V} + \lambda(\sum_{c=1}^{m}\boldsymbol{\alpha}_c\boldsymbol{W}^c)\boldsymbol{U})_{il}}{(\boldsymbol{U}\boldsymbol{V}^T\boldsymbol{V} + \lambda(\sum_{c=1}^{m}\boldsymbol{\alpha}_c\boldsymbol{D}^c)\boldsymbol{U})_{il}} \tag{8}$$

$$\boldsymbol{V}_{il} = \boldsymbol{V}_{il}\frac{(\boldsymbol{X}^T\boldsymbol{U})_{il}}{(\boldsymbol{V}\boldsymbol{U}^T\boldsymbol{U})_{il}} \tag{9}$$

When $\boldsymbol{U}$ and $\boldsymbol{V}$ are fixed, the optimization problem for $\boldsymbol{\alpha}$ is equivalent to solving the following problem

$$\min_{\boldsymbol{\alpha}} \quad \sum_{c=1}^{m}\boldsymbol{\alpha}_c\text{tr}(\boldsymbol{U}^T\boldsymbol{L}^c\boldsymbol{U}), \text{ s.t.} \quad \sum_{c=1}^{m}\boldsymbol{\alpha}_c = 1, \boldsymbol{\alpha}_c \geq 0 \tag{10}$$

which is a *linear programming* problem and can be efficiently solved. However, the solution will always be

$$\boldsymbol{\alpha}_c = \begin{cases} 1, & \text{if } c = \arg\min_c \operatorname{tr}(\boldsymbol{U}^T \boldsymbol{L}^c \boldsymbol{U}) \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

To avoid this trivial solution, we propose to add one regularization term to (10) and solve the following problem

$$\min_{\boldsymbol{\alpha}} \quad \sum_{c=1}^{m} \boldsymbol{\alpha}_c \operatorname{tr}(\boldsymbol{U}^T \boldsymbol{L}^c \boldsymbol{U}) + \lambda_2 ||\boldsymbol{\alpha}||^2, \text{ s.t.} \quad \sum_{c=1}^{m} \boldsymbol{\alpha}_c = 1, \boldsymbol{\alpha}_c \geq 0 \tag{12}$$

where $\lambda_2 \geq 0$ is a tradeoff parameter. In this way, we will solve a *quadratic programming* (QP) problem with respect to $\boldsymbol{\alpha}$ when $\boldsymbol{U}$ and $\boldsymbol{V}$ are fixed.

Minimizing $\mathcal{O}_2$. When $\boldsymbol{\alpha}$ is fixed, the optimization problem of computing $\boldsymbol{U}$, $\boldsymbol{V}$ becomes

$$\begin{aligned} \mathcal{L} = & \sum_{i=1}^{n} \sum_{j=1}^{d} (\boldsymbol{X}_{ij} \log \frac{\boldsymbol{X}_{ij}}{\sum_{l=1}^{k} \boldsymbol{U}_{il} \boldsymbol{V}_{jl}} - \boldsymbol{X}_{ij} + \sum_{l=1}^{k} \boldsymbol{U}_{il} \boldsymbol{V}_{jl}) \\ & + \frac{\lambda}{2} \sum_{c=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{d} \sum_{l=1}^{k} \boldsymbol{\alpha}_c \boldsymbol{W}_{ij}^c (\boldsymbol{U}_{il} \log \frac{\boldsymbol{U}_{il}}{\boldsymbol{U}_{jl}} + \boldsymbol{U}_{jl} \log \frac{\boldsymbol{U}_{jl}}{\boldsymbol{U}_{il}}) \\ & + \operatorname{tr}(\boldsymbol{\Phi} \boldsymbol{U}^T) + \operatorname{tr}(\boldsymbol{\Psi} \boldsymbol{V}^T) \end{aligned} \tag{13}$$

Similarly, the above optimization is equivalent to GNMF-KL [16] with a combined Laplacian matrix. We give the linear system for update $\boldsymbol{U}$

$$\left(\sum_j v_{jl} I + \lambda (\sum_c L^c) \right) \boldsymbol{u}_l = \begin{bmatrix} u_{1l} \sum_j (x_{1j} v_{jl} / \sum_l u_{1l} v_{jl}) \\ u_{2l} \sum_j (x_{2j} v_{jl} / \sum_l u_{2l} v_{jl}) \\ \cdots \\ u_{nl} \sum_j (x_{nj} v_{jl} / \sum_l u_{nl} v_{jl}) \end{bmatrix} \tag{14}$$

The above linear system can be solved either by the matrix inverse or some efficient iterative algorithms. The correspond update rule for $\boldsymbol{V}$ is given below.

$$\boldsymbol{V}_{jl} = \boldsymbol{V}_{jl} \frac{\sum_i (\boldsymbol{X}_{ij} \boldsymbol{U}_{il} / \sum_l \boldsymbol{U}_{il} \boldsymbol{V}_{jl})}{\sum_i \boldsymbol{U}_{il}} \tag{15}$$

When $\boldsymbol{U}$ and $\boldsymbol{V}$ are fixed, a similar QP problem with respect to the weights $\boldsymbol{\alpha}$ becomes

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \sum_{c=1}^{m} \boldsymbol{\alpha}_c \left[\sum_{i=1}^{n} \sum_{j=1}^{d} \sum_{l=1}^{k} \boldsymbol{W}_{ij}^c (\boldsymbol{U}_{il} \log \frac{\boldsymbol{U}_{il}}{\boldsymbol{U}_{jl}} + \boldsymbol{U}_{jl} \log \frac{\boldsymbol{U}_{jl}}{\boldsymbol{U}_{il}}) \right] + \lambda_2 ||\boldsymbol{\alpha}||^2 \\ \text{s.t.} \quad & \sum_{c=1}^{m} \boldsymbol{\alpha}_c = 1, \boldsymbol{\alpha}_c \geq 0 \end{aligned} \tag{16}$$

Algorithm 1. Minimizing $\mathcal{O}_1$ and $\mathcal{O}_2$

Input: Partitions $\{\mathcal{P}^i\}_{i=1}^m$, the number of clusters k and regularization parameter λ
Output: $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{\alpha}$

```
Construct m co-association matrix {W^c}_{c=1}^m
Initialize U, V and set α = [1/m, 1/m, ..., 1/m]^T
repeat
    Compute the weighted graph Laplacian based on α
    Update U_il according to Eq. (8) for O_1 or (14) for O_2
    Update V_jl according to Eq. (9) for O_1 or (15) for O_2
    Update α by solving the QP programming in Eq. (12) for O_1 or (16) for O_2
until convergence
return U, V and α
```

The whole process of estimating the low-rank matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and the combination weights $\boldsymbol{\alpha}$ is described in Algorithm 1.

The above learning algorithm will converge. In each iteration the objective function value always decreases. On the one hand, when $\boldsymbol{U}$ and $\boldsymbol{V}$ are fixed, WGNMF solves a quadratic programming problem. On the other hand, when $\boldsymbol{\alpha}$ is fixed WGNMF boils down to GNMF, the update rules of $\boldsymbol{U}$ and $\boldsymbol{V}$ are similar to the update rules in GNMF and therefore Cai's [16] proof can also be applied.

4 Experiments

In this section, we conduct experiments on a number of real-world datasets to evaluate of the effectiveness of the proposed method.

4.1 Datasets

We use a variety of datasets (see Table 1) to evaluate the accuracy of the proposed method. The number of classes is ranged from 3 to 40, the number of samples ranged from 47 to 2340, and the number of dimension ranged from 4 to 21839. Details are as follows:

- Five datasets (Iris, Glass, Ecoli, Soybean, Zoo) are from UCI data repository [18].
- The COIL dataset is an image library from Columbia with 20 objects. The images of each objects were taken 5 degrees apart as the object is rotated on a turntable and each object has 72 images.
- The ORL dataset is a face database from Olivetti Research Laboratory. It consists of 400 face images with 40 people (10 samples per person). The images were captured at different times and have different variations including expressions (open or closed eyes, smiling or non-smiling) and facial details (glasses or no glasses).
- The WebACE dataset is from WebACE project and has been used for document clustering. It contains 2340 documents consisting of news article from Reuters new service of 20 different categories collected in October 1997.

Table 1. Descriptions of datasets.

Data sets	Samples	Dimensions	Classes
Iris	150	4	3
Glass	214	9	6
Ecoli	336	7	8
Soybean	47	35	4
Zoo	101	18	7
COIL	1440	1024	20
ORL	400	1024	40
WebACE	2340	21839	20
Tr11	414	6429	9
Tr12	313	5804	8

– Tr11, Tr12, these datasets are derived from the TREC [1] collection , which are often used in document clustering.

The size of each image in COIL and ORL is 32×32 pixels, with 256 grey levels per pixel. Thus, each image is represented by a 1024-dimensional vector.

4.2 Evaluation Criteria

In the experiments, we set the number of clusters equal to the number of classes k for all the cluster ensembles algorithms. To evaluate their performance, we compare the clustering results generated by these algorithms with the true classes by computing two performance measures, Clustering Accuracy (Acc) and Normalized Mutual Information (NMI) [1].

For a sample x_i, the cluster label assigned by the algorithm is denoted as r_i, and ground true label is y_i. The accuracy is defined as follows:

$$\text{Acc} = \frac{\sum_{i=1}^{n} \delta(y_i, map(r_i))}{n}$$

where n is the total number of samples and $\delta(x, y)$ is the indicator function that equals 1 if $x = y$ and equals 0 otherwise, and $\text{map}(r_i)$ is the permutation mapping function that maps the obtained labels r_i to the equivalent label from the data set. The best mapping function can be found by using the Kuhn-Munkres algorithm [19]. The value of Acc equals 1 if and only if the clustering result and the true label are identical. Larger values of Acc indicate better clustering performance.

Given a clustering $\mathcal{P}$ and the true partitioning $\mathcal{Y}$ (class labels). The number of clusters in $\mathcal{P}$ and classes in $\mathcal{Y}$ are both k. Suppose n_i is the number of objects in the i-th cluster, n_j is the number of objects in the j-th class and n_{ij} is the number of objects which belongs to the i-th cluster and j-th class. NMI between $\mathcal{P}$ and $\mathcal{Y}$ is calculated as follows [1]:

[1] `http://trec.nist.gov`

$$\mathrm{NMI}(\mathcal{P},\mathcal{Y}) = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} n_{ij}\log\frac{n\cdot n_{ij}}{n_i\cdot n_j}}{\sqrt{\sum_{i=1}^{k} n_i \log\frac{n_i}{n}\sum_{j=1}^{k} n_j\log\frac{n_j}{n}}}$$

The value of NMI equals 1 if and only if $\mathcal{P}$ and $\mathcal{Y}$ are identical and is close to 0 if $\mathcal{P}$ is a random partitioning. Larger values of NMI indicate better clustering performance.

Table 2. Experimental Results in Clustering Accuracy

	Kmeans	KC	CSPA	HGPA	MCLA	BCE	GWCA	WGNMF-Euc	WGNMF-KL
Iris	0.81	0.85	0.87	0.62	0.89	0.89	0.89	0.89	0.89
Glass	0.42	0.49	0.43	0.40	0.46	0.49	0.53	0.54	0.51
Ecoli	0.65	0.64	0.56	0.51	0.61	0.66	0.64	0.67	0.65
Soybean	0.72	0.65	0.69	0.72	0.73	0.70	0.73	0.75	0.73
Zoo	0.69	0.67	0.58	0.55	0.74	0.67	0.74	0.77	0.73
COIL	0.59	0.62	0.69	0.55	0.69	0.67	0.58	0.69	0.71
ORL	0.50	0.53	0.58	0.60	0.60	0.51	0.52	0.56	0.60
WebACE	0.43	0.46	0.40	0.35	0.47	0.48	0.47	0.46	0.48
Tr11	0.52	0.57	0.49	0.47	0.52	0.58	0.58	0.60	0.60
Tr12	0.47	0.56	0.54	0.52	0.57	0.58	0.58	0.57	0.60

4.3 Comparison Settings

To generate the input for cluster ensembles algorithms, we adopt a common strategy [3] by running Kmeans 200 times with random initiation and then splitting each 20 clustering results as input. In this way, we repeat the cluster ensembles algorithms 10 times. We report the averaged performance over 10 rounds.

To demonstrate how the clustering performance can be improved by our method, we compare the following clustering ensemble algorithms.

- Standard Kmeans clustering algorithm (Kmeans).
- Kmeans clustering on Consensus matrix (KC). The consensus function is defined as the averaged co-association matrix.
- The Cluster-based Similarity Partitioning Algorithm (CSPA), Hyper Graph Partitioning Algorithm (HGPA) and MetaClustering Algorithm (MCLA) are three algorithms described in [1]. We use the author's matlab implementation ClusterPack[2].
- Bayesian Cluster Ensembles [3] (BCE) is a generative probabilistic model which learns the implicit consensus function from the cluster-based representation.
- Generalized Weighted Cluster Aggregation [4] (GWCA) define the consensus function by the weighted averaged co-association matrix. We use the Euclidean distance to learn the weighted consensus matrix and the spectral clustering algorithm[3] is further used to derive the final clustering.

[2] `www.lans.ece.utexas.edu/~strehl`

[3] `http://www.cis.upenn.edu/~jshi/software/`

Table 3. Experimental Results in Normalized Mutual Information

	Kmeans	KC	CSPA	HGPA	MCLA	BCE	GWCA	WGNMF-Euc	WGNMF-KL
Iris	0.69	0.72	0.71	0.39	0.74	0.74	0.75	0.74	0.74
Glass	0.31	0.33	0.29	0.26	0.32	0.35	0.37	0.38	0.37
Ecoli	0.58	0.59	0.51	0.40	0.56	0.59	0.57	0.59	0.58
Soybean	0.72	0.67	0.63	0.69	0.71	0.69	0.71	0.73	0.71
Zoo	0.69	0.70	0.59	0.60	0.74	0.70	0.73	0.74	0.73
COIL	0.73	0.75	0.76	0.69	0.78	0.77	0.73	0.79	0.80
ORL	0.71	0.74	0.76	0.77	0.77	0.69	0.73	0.75	78
WebACE	0.53	0.55	0.51	0.45	0.54	0.57	0.56	0.58	0.57
Tr11	0.48	0.58	0.52	0.48	0.52	0.60	0.56	0.59	0.60
Tr12	0.38	0.54	0.49	0.43	0.50	0.57	0.50	0.57	0.59

We present the results of our WGNMF algorithm under the Euclidean distance and KL divergence (denoted as WGNMF-Euc and WGNMF-KL). The regularization paramter λ is selected via a coarse grid search process.

4.4 Experimental Results

The experimental results are summarized in Tables 2 and 3. From these two tables we observe that our WGNMF improves Kmeans clustering on all datasets. Moreover, WGNMF (WGNMF-Euc or WNGMF-KL) achieves the best performance on 9 out of 10 datasets and its performance on the remaining datasets is close to the best results. In summary, the experiments clear show the effectiveness of weighted graph regularized NMF for improving the cluster ensembles algorithms in terms of clustering accuracy and NMI.

4.5 Individual Clustering Selection

A useful byproduct of WGNMF is that the learned weights can be used to select individual input clusterings, i.e. asses how important of each input clusterings. The reason is that an input clustering with larger weight contributes more to the consensus affinity graph and the final clustering. We compare the top-5 selected clustering based on the weights learned from WGNMF and GWCA [4], the latter is a weighted consensus clustering technique and also learn the combination weights, with the results of all clusterings. The results are given in Figure 1. We observe that the input clusterings which obtain larger weights are generally good clusterings.

4.6 Impact of Parameter λ

In our model λ is used to control the smoothness of the low dimensional representations. We run WGNMF with varying λ from a coarse grid $(0.1, 1, 10, 100)$. Table 4 shows the impact of λ on Acc and NMI with some datasets. We observe that WGNMF achieves good performance in a wide range and it is easy to tune.

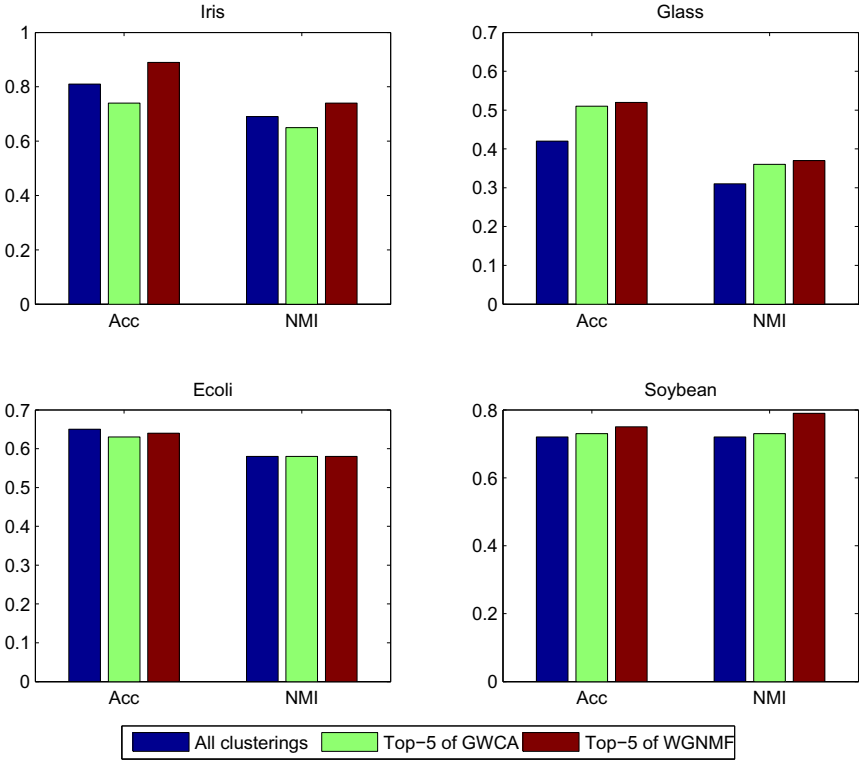

Fig. 1. Performance comparison of All clusterings vs. Top-5 clusterings

Table 4. The performance of WGNMF vs. parameter λ

	Acc				NMI			
	0.1	1	10	100	0.1	1	10	100
Iris	0.89	0.89	0.89	0.89	0.74	0.74	0.74	0.74
Glass	0.53	0.54	0.52	0.52	0.37	0.38	0.37	0.37
Soybean	0.73	0.73	0.73	0.75	0.71	0.71	0.71	0.73
Zoo	0.75	0.75	0.77	0.68	0.73	0.74	0.74	0.70

5 Conclusion and Future Work

In this paper, we propose a weighted graph regularized nonnegative matrix factorization model for cluster ensembles task. We integrate two representations of input clustering, multiple co-association matrices and cluster feature matrix in a unified regularization framework. We learn the implicit consensus function from cluster feature matrix by NMF procedure while the factorization is regularized with consensus matrix. The weights of input clusterings are learned within the factorization process. Extensive experiments on a number of real-world datasets

demonstrate that the proposed method mostly outperforms the other cluster ensemble algorithms.

In future work, we want to extend the idea of this paper to other statistical models, such as topic modeling, to integrate different representations of input clusterings beyond the multiple co-association matrices and the cluster based matrix.

Acknowledgments. We would like to thank all anonymous reviewers for their helpful comments. This work is supported in part by the National Natural Science Foundation of China (NSFC) grants 60970045 and 60833001.

References

1. Strehl, A., Ghosh, J.: Cluster ensembles - a knowledge reuse framework for combining multiple partitions. The Journal of Machine Learning Research 3, 583–617 (2002)
2. Li, T., Ding, C.: Weighted consensus clustering. In: Proceedings of the 8th SIAM International Conference on Data Mining, pp. 798–809 (2008)
3. Wang, H., Shan, H., Banerjee, A.: Bayesian cluster ensembles. In: Proceedings of the 9th SIAM International Conference on Data Mining, pp. 211–222 (2009)
4. Wang, F., Wang, X., Li, T.: Generalized cluster aggregation. In: Proceedings of the 21st International Jont Conference on Artifical Intelligence, pp. 1279–1284 (2009)
5. Topchy, A., Jain, A., Punch, W.: Clustering ensembles: Models of consensus and weak partitions. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1866–1881 (2005)
6. Gionis, A., Mannila, H., Tsaparas, P.: Clustering aggregation. ACM Transactions on Knowledge Discovery from Data 1(1), 4 (2007)
7. Li, T., Ding, C., Jordan, M.: Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In: Proceedings of the 7th IEEE International Conference on Data Mining, pp. 577–582 (2007)
8. Wang, F., Ding, C., Li, T.: Integrated kl (k-means-laplacian) clustering: A new clustering approach by combining attribute data and pairwise relations. In: Proceedings of the 9th SIAM International Conference on Data Mining, pp. 38–48 (2009)
9. Al-Razgan, M., Domeniconi, C.: Weighted clustering ensembles. In: Proceedings of 6th SIAM International Conference on Data Mining, pp. 258–269 (2006)
10. Hadjitodorov, S., Kuncheva, L., Todorova, L.: Moderate diversity for better cluster ensembles. Information Fusion 7(3), 264–275 (2006)
11. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing 20(1), 359 (1999)
12. Fern, X., Brodley, C.: Solving cluster ensemble problems by bipartite graph partitioning. In: Proceedings of the 21th International Conference on Machine Learning, pp. 281–288 (2004)
13. Topchy, A., Jain, A., Punch, W.: A mixture model for clustering ensembles. In: Proceedings of 4th SIAM International Conference on Data Mining, pp. 379–390 (2004)
14. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. The Journal of Machine Learning Research 3, 993–1022 (2003)

15. Lee, D., Seung, H.: Learning the parts of objects by non-negative matrix factorization. Nature 401(6755), 788–791 (1999)
16. Cai, D., He, X., Han, J., Huang, T.S.: Graph regularized non-negative matrix factorization for data representation. IEEE Transactions on Pattern Analysis and Machine Intelligence (to appear, 2011)
17. Bertsekas, D.: Nonlinear programming. Athena Scientific, Belmont (1999)
18. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
19. Lovász, L., Plummer, M.: Matching theory (1986)
20. Fern, X., Brodley, C.: Random projection for high dimensional data clustering: A cluster ensemble approach. In: Proceedings of the 20th International Conference on Machine Learning, pp. 186–193 (2003)
21. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics, 32–38 (1957)
22. Monti, S., Tamayo, P., Mesirov, J., Golub, T.: Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. Machine Learning 52(1), 91–118 (2003)