Constrained Clustering via Spectral Regularization

Zhenguo Li[1,2], Jianzhuang Liu[1,2], and Xiaoou Tang[1,2]
[1]Dept. of Information Engineering, The Chinese University of Hong Kong, Hong Kong
[2]Multimedia Lab, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
{zgli,jzliu,xtang}@ie.cuhk.edu.hk

Abstract

We propose a novel framework for constrained spectral clustering with pairwise constraints which specify whether two objects belong to the same cluster or not. Unlike previous methods that modify the similarity matrix with pairwise constraints, we adapt the spectral embedding towards an ideal embedding as consistent with the pairwise constraints as possible. Our formulation leads to a small semidefinite program whose complexity is independent of the number of objects in the data set and the number of pairwise constraints, making it scalable to large-scale problems. The proposed approach is applicable directly to multi-class problems, handles both must-link and cannot-link constraints, and can effectively propagate pairwise constraints. Extensive experiments on real image data and UCI data have demonstrated the efficacy of our algorithm.

1. Introduction

Clustering plays an important role in data analysis and pattern recognition, with the goal to group similar objects in data. It is conducted typically in an unsupervised manner (unsupervised clustering), and its performance depends critically on the similarity among data. Generally, the similarity is derived based on the features extracted. However, the choice of an appropriate similarity measure for clustering, especially for visual data like images, is a difficult problem that is still at the core of current research in computer vision. The major difficulty lies in that on one hand, we intend to group together objects of the same category, which is a high-level semantic concept, but on the other hand, we only have low-level features that may not well characterize the high-level semantic concepts. Besides, although most clustering algorithms produce a single clustering result for any given data set, a data set may have more than one natural and plausible clustering. For example, consider a set of pictures of different persons in different poses, as shown in Fig. 1. These images can be clustered by person identity, by pose, by with or without glasses, or by gender. These practical concerns naturally lead us to the idea of exploiting high-level information to guide and improve clustering, in addition to low-level data features.

Figure 1. Images of different persons in different poses, with or without glasses. Each row has different persons in the same pose. Each column has the same person in different poses.

Pairwise constraints turn out to be a natural choice, which indicate that some pairs of objects are in the same cluster and some are not, known respectively as the must-link and the cannot-link [21]. By providing pairwise constraints on some objects, the semantic gap between high-level semantic concepts and low-level data features can be reduced, and the clustering ambiguity for the data where multiple clusterings are possible can be removed. Such pairwise relationship may be easily obtained from domain knowledge automatically or using a little human effort. For example, for face categorization, any two faces from a single image must come from two different persons, and thus form a cannot-link constraint, whereas those with spatially overlapping from temporally adjacent frames (e.g., in a video) can be assumed to be must-link. Now we have two sources of similarity information, the feature similarity and the pairwise constraints. The former is dense but less informative, while the latter is sparse but more informative. The task is to combine these two sources of similarity to derive a desirable clustering, known as constrained clustering or semi-supervised clustering.

The seminal work in constrained clustering comes from [21] that uses pairwise constraints to adapt an incremental clustering algorithm. It is demonstrated in [21] that the performance of a clustering algorithm can be significantly improved by respecting the pairwise constraints during clus-

978-1-4244-3991-1/09/$25.00 ©2009 IEEE

tering. Since then, many methods have been proposed to adapt k-means [22], linkage [12], and Gaussian mixtures [19]. Another line of research learns a distance metric [23] or a kernel matrix [9, 14] so that the must-link objects become close and the cannot-link objects become far apart .

Our focus in this paper is to adapt spectral clustering [20, 17], which uses the eigenvectors of the similarity matrix (or its variants) for clustering and attracts considerable attention in recent years. Most constrained spectral clustering algorithms [11, 13, 16] follow the procedure that first modifies the similarity matrix with pairwise constraints and then calls spectral clustering. Kamvar et al. [11] simply modified the similarities to 1's and 0's for must-link and cannot-link objects, respectively. Kulis et al. [13] increased the similarities for must-link objects and decreased those for cannot-link objects even to be negative. These two methods only modify similarities between constrained objects. In contrast, Lu and Carreira-Perpiñán [16] propagated pairwise constraints to other similarities between unconstrained objects using Gaussian process. As noted by the authors, however, this method only makes sense for two-class problems, although heuristics for multi-class problems is also discussed. Yu and Shi [24] and Coleman et al. [6] attempted to formulate constrained normalized cuts, but Yu and Shi's algorithm can handle only must-link constraints, and Coleman et al.'s can deal with only two-class problems.

As noted in [24, 16, 6], the major difficulty in incorporating pairwise constraints into spectral clustering arises from the nontransitive property of cannot-link constraints, in contrast to must-link constraints which are transitive. Consequently, these methods either limit to two-class problems or have to discard the cannot-link constraints even when available. The ways of using cannot-link constraints in [11, 13] are not well justified too, because a similarity of 0 between two cannot-link objects in the input space does not mean that the two objects tend to belong to different categories, and a negative similarity may even cause the resulting kernel matrix to be non-positive-semidefinite, a necessary condition for the (weighted) kernel k-means, used in [13], to converge. This difficulty in handling cannot-link constraints is also noted in [12].

In this paper, we adapt the spectral clustering in a rather different way. Instead of modifying the similarity matrix, we propose to bias the spectral embedding towards an ideal embedding as consistent with the pairwise constraints as possible (see Fig. 2). The idea of adapting the spectral embedding is inspired by the observation that the spectral embedding consists of the smoothest eigenvectors of the normalized Laplacian on the graph, and adapting it to accord with pairwise constraints will in effect propagate the pairwise constraints to unconstrained objects. Our formulation leads to a small semidefinite program whose complexity is independent of the size of the data set and the number of pairwise constraints, making it scale well to large-scale problems. Compared with previous constrained spectral clustering methods, the proposed approach is applicable directly to multi-class problems, handles both must-link and cannot-link constraints, and can effectively propagate pairwise constraints. The rest of the paper is organized as follows. We review necessary preliminaries in Section 2 and propose the overall framework in Section 3. Experimental results are reported in Section 4. Section 5 concludes the paper.

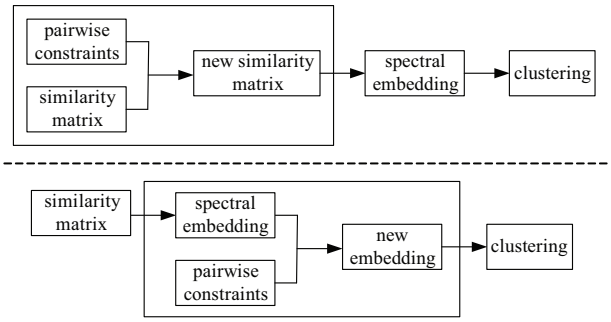

Figure 2. Algorithmic flows for constrained spectral clustering. Top panel: adapt the similarity matrix with pairwise constraints used in previous methods [11, 13, 16]. Bottom panel: adapt the spectral embedding with pairwise constraints proposed in this paper.

2. Spectral Graph Theory

In this section, we review related background on spectral graph theory [5] and introduce the notation that will be used throughout this paper. Given a data set of n object $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$, let $\mathcal{G} = \{V, W\}$ be an undirected, weighted graph with its node set $V = \mathcal{X}$ and weight matrix $W = [w_{ij}]$, where w_{ij} denotes the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. W is assumed to be non-negative and symmetric. Following [25], we measure the smoothness of a function on the graph $f : V \to \mathcal{R}$ by

$$\Omega(f) = \frac{1}{2} \sum_{i,j} w_{ij} \left(\frac{f(\mathbf{x}_i)}{\sqrt{d_i}} - \frac{f(\mathbf{x}_j)}{\sqrt{d_j}} \right)^2, \qquad (1)$$

where $d_i = \sum_j w_{ij}$ is the degree of node $\mathbf{x}_i$. The smaller is $\Omega(f)$, the smoother is f. This measure penalizes large changes over nodes that are strongly connected. An unnormalized smoothness measure also frequently used can be found in [1]. Let $L = D - W$, where $D = \text{diag}(d_1, ..., d_n)$. L is known as the graph Laplacian of $\mathcal{G}$ [5]. The normalized graph Laplacian of $\mathcal{G}$ is defined as $\bar{L} = D^{-1/2} L D^{-1/2}$. We can rewrite (1) as

$$\Omega(f) = \mathbf{f}^T \bar{L} \mathbf{f}, \qquad (2)$$

where $\mathbf{f} = (f(\mathbf{x}_1), ..., f(\mathbf{x}_n))^T$. Note that on the graph a function f is identical to the vector $\mathbf{f}$. So we can also view

a vector of length n (the number of nodes in the graph) as a function on the graph.

It turns out instructive to study the smoothness of the eigenvectors of $\bar{L}$, which can be naturally viewed as functions on the graph. Let $(\lambda_i, \mathbf{v}_i)$, $i = 1, ..., n$, be the eigenvalues and the associated eigenvectors of $\bar{L}$, where $0 \leq \lambda_1 \leq \cdots \leq \lambda_n$ and $\|\mathbf{v}_i\| = 1$. We have

$$\Omega(\mathbf{v}_i) = \mathbf{v}_i^T \bar{L} \mathbf{v}_i = \lambda_i, \tag{3}$$

which implies that the smoothness of eigenvector $\mathbf{v}_i$ is measured by the associated eigenvalue λ_i: the smaller is the eigenvalue λ_i, the smoother is the eigenvector $\mathbf{v}_i$. Let

$$F_i = (\mathbf{v}_1, ..., \mathbf{v}_i), \tag{4}$$

and call F_i the i-th order *spectral embedding* of the graph, with row j of F_i being taken as the new representation for node $\mathbf{x}_j$. Note that spectral clustering [20, 17] uses the k-th order (k is the number of clusters) spectral embedding F_k or its variants to form the new data representation.

$\bar{L}$ is symmetric and positive semidefinite by definition and (1-2). Thus from linear algebra, $\mathbf{v}_i$'s are mutually orthogonal and constitute an orthonormal basis of the functional space on the graph. So any function $\mathbf{f} : V \to \mathcal{R}$ on the graph has the following factorization:

$$\mathbf{f} = \sum_{i=1}^{n} a_i \mathbf{v}_i. \tag{5}$$

As $\mathbf{v}_i$'s are ordered by smoothness, this factorization means that we can decompose any function on the graph into n components ordered by smoothness. This is in essence similar to the Fourier transform [4] in signal processing that decomposes a signal into components from low frequency to high frequency, where the low-frequency components capture most of the energy of the signal while the high-frequency components consist mainly of subtle details and noise that can be removed from the signal. This is also similar to Principal Components Analysis (PCA) [10] where the variance takes the place of the smoothness.

Rewrite (5) as

$$\mathbf{f} = \sum_{i=1}^{m} a_i \mathbf{v}_i + \sum_{i=m+1}^{n} a_i \mathbf{v}_i = T_1 + T_2, \tag{6}$$

where $m \ll n$. We call T_1 and T_2 the *smoothness* component and the *noise* component of $\mathbf{f}$, respectively. Discarding the noise component of a function is to promote the smoothness of the function. Thus, representing a function as T_1 actually imposes smoothness on the function. We call this regularization technique *spectral regularization*.

3. A Spectral Regularization Framework

Given a data set $\mathcal{X}$ and a number of pairwise constraints specifying which two objects are in the same cluster, denoted by $\mathcal{M} = \{(i, j)\}$, and which two are not, denoted by $\mathcal{C} = \{(i, j)\}$, our goal is to partition $\mathcal{X}$ into k clusters, so that $\mathcal{M}$ and $\mathcal{C}$ are well respected. Specifically, we intend to adapt spectral clustering to fulfill this goal. In spectral clustering, one first obtains a spectral embedding of the data, and then applies a baseline clustering algorithm, such as k-means, to the spectral embedding to form clusters. We follow the same procedure but bias the spectral embedding using pairwise constraints before clustering.

Our main idea is to select, from a *smooth* data representation space, the one that is most consistent with the given pairwise constraints. In this way, the pairwise constraints can be propagated to unconstrained objects, which is desirable because of probable sparsity of pairwise constraints in practice [12, 14]. In what follows, we will show how to construct such a representation space and how to select the optimal representation using pairwise constraints.

3.1. A Representation Space

In this subsection, we explicitly construct a smooth data representation space based on the spectral regularization technique described in Section 2. Suppose that we want to find a d-dimensional representation $(\mathbf{f}_1, ..., \mathbf{f}_d)$ for the data. For the purpose of clustering, a good data representation should keep nearby objects in the input space also nearby in the new representation. This requires the representation to be smooth on the graph, meaning that each dimension $\mathbf{f}_i$ of the representation should be smooth on the graph. Based on the analysis in Section 2, we propose to construct a data representation space as

$$\mathcal{H} = \{(\mathbf{f}_1, ..., \mathbf{f}_d) \mid \mathbf{f}_i = \sum_{j=1}^{m} a_{ji} \mathbf{v}_j, 1 \leq i \leq d\}, \tag{7}$$

where m is a free parameter that controls the smoothness of each dimension of the representation, thus controls the smoothness of each representation, and thus the smoothness of the representation space $\mathcal{H}$. In this sense, m is a *regularization* parameter. We next show that $\mathcal{H}$ can be expressed in terms of the m-th order spectral embedding F_m.

Let $\mathbf{a}_i = (a_{1i}, ..., a_{mi})^T$ and $A = (\mathbf{a}_1, ..., \mathbf{a}_d)$. We have

$$\mathbf{f}_i = F_m \mathbf{a}_i, \ (\mathbf{f}_1, ..., \mathbf{f}_d) = F_m A. \tag{8}$$

Thus $\mathcal{H}$ can be succinctly rewritten as

$$\mathcal{H} = \{F_m A \mid A \in \mathcal{R}^{m \times d}\}. \tag{9}$$

Our task is then to choose a representation from $\mathcal{H}$ that is most consistent with the given pairwise constraints, which is equivalent to determining a suitable coefficient matrix A.

3.2. Ideal Representation

To demonstrate our idea of how to use the given pairwise constraints to bias the spectral embedding so that the new representation well respects the pairwise constraints, we show here what an ideal representation should be via an ideal case.

To partition a data set into k clusters, it is desired to learn k cluster indicators $\mathbf{f}_1, ..., \mathbf{f}_k$, each corresponding to a cluster, such that $\mathbf{f}_i(\mathbf{x}) = 1$ if $\mathbf{x}$ belongs to the i-th *ground-truth* cluster and $\mathbf{f}_i(\mathbf{x}) = 0$ otherwise. Let $F = (\mathbf{f}_1, ..., \mathbf{f}_k)$ and denote its i-th row by a column vector $\mathbf{y}_i$, i.e., $F = (\mathbf{y}_1, ..., \mathbf{y}_n)^T$. Then we can see that

$$\mathbf{y}_i^T \mathbf{y}_j = \begin{cases} 1, & l(\mathbf{x}_i) = l(\mathbf{x}_j) \\ 0, & l(\mathbf{x}_i) \neq l(\mathbf{x}_j), \end{cases} \tag{10}$$

where $l(\mathbf{x}_i)$ denotes the cluster label of $\mathbf{x}_i$. The point here is that if we take $\mathbf{y}_i$ as a new representation for object $\mathbf{x}_i$, then: (i) each object is mapped to a unit sphere since $\mathbf{y}_i^T \mathbf{y}_i = 1$ (*data normalization*); (ii) objects within a cluster are mapped to the same point since $\mathbf{y}_i^T \mathbf{y}_j = 1$ if $i \neq j$ and $l(\mathbf{x}_i) = l(\mathbf{x}_j)$ (*cluster collapsing*); and (iii) objects in different clusters are mapped to be orthogonal since $\mathbf{y}_i^T \mathbf{y}_j = 0$ if $i \neq j$ and $l(\mathbf{x}_i) \neq l(\mathbf{x}_j)$ (*cluster maximal separability*). We call a data representation with these three properties an *ideal* representation. Clearly, clustering becomes trivial with an ideal representation.

3.3. A Cost Function

The analysis on the ideal case suggests a representation that is as close to the ideal one as possible. In our case with pairwise constraints, we thus prefer such a representation that in the new presentation, the objects are close to a unit sphere, must-link objects are close to each other, and cannot-link objects are far apart. Formally, let $F = (\mathbf{y}_1, ..., \mathbf{y}_n)^T$ be a data representation. We propose to measure its cost as

$$\begin{aligned} \mathcal{L}(F) = & \sum_{i=1}^{n} (\mathbf{y}_i^T \mathbf{y}_i - 1)^2 + \sum_{(i,j) \in \mathcal{M}} (\mathbf{y}_i^T \mathbf{y}_j - 1)^2 \\ & + \sum_{(i,j) \in \mathcal{C}} (\mathbf{y}_i^T \mathbf{y}_j - 0)^2. \end{aligned} \tag{11}$$

A good representation should result in a small cost.

Let $\mathcal{S} = \{(i, j, t_{ij})\}$ be the set of pairwise constraints, where t_{ij} is a binary variable that takes 1 or 0 to indicate $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same cluster or not. It is reasonable to assume $(i, i, t_{ii}) \in \mathcal{S}$, where $t_{ii} = 1$, $i = 1, ..., n$. Then the cost function (11) can be succinctly rewritten as

$$\mathcal{L}(F) = \sum_{(i,j,t_{ij}) \in \mathcal{S}} (\mathbf{y}_i^T \mathbf{y}_j - t_{ij})^2. \tag{12}$$

3.4. The Optimization Problem

Based on the above analysis, we propose the following optimization problem:

$$\min_{F \in \mathcal{H}} \mathcal{L}(F). \tag{13}$$

Since $F \in \mathcal{H}$, we can write $F = F_m A$. $F_m = (\mathbf{v}_1, ..., \mathbf{v}_m)$ can also be represented as $F_m = (\mathbf{u}_1, ..., \mathbf{u}_n)^T$, where $\mathbf{u}_i^T$ denotes the i-th row of F_m. Then

$$\mathbf{y}_i = (\mathbf{u}_i^T A)^T = A^T \mathbf{u}_i, \tag{14}$$

and

$$\mathbf{y}_i^T \mathbf{y}_j = \mathbf{u}_i^T A A^T \mathbf{u}_j. \tag{15}$$

With (12) and (15), the optimization problem (13) can be formulated as

$$\min_{A \in \mathcal{R}^{m \times d}} \mathcal{L}(A) \triangleq \sum_{(i,j,t_{ij}) \in \mathcal{S}} (\mathbf{u}_i^T A A^T \mathbf{u}_j - t_{ij})^2. \tag{16}$$

This is an unconstrained biquadratic optimization problem, which is not convex with respect to A in general. However, it can be relaxed to a convex one that can be optimally solved efficiently, as shown in the next subsection.

3.5. Semidefinite Programming

Let $M = AA^T$. Then $M \in \mathcal{R}^{m \times m}$ is positive semidefinite, denoted as $M \succeq 0$. The objective function $\mathcal{L}(A)$ in (16) becomes

$$\mathcal{L}(M) \triangleq \sum_{(i,j,t_{ij}) \in \mathcal{S}} (\mathbf{u}_i^T M \mathbf{u}_j - t_{ij})^2. \tag{17}$$

Let $\mathbf{z} = \text{vec}(M)$ be the vector obtained by concatenating all the columns of M. With some mathematical manipulations, we have

$$\mathcal{L}(M) = \mathbf{z}^T B \mathbf{z} + \mathbf{b}^T \mathbf{z} + c, \tag{18}$$

where

$$\begin{aligned} B &= \sum_{(i,j,t_{ij}) \in \mathcal{S}} \mathbf{s}_{ij} \mathbf{s}_{ij}^T, \quad \mathbf{b} = -2 \sum_{(i,j,t_{ij}) \in \mathcal{S}} t_{ij} \mathbf{s}_{ij}, \\ c &= \sum_{(i,j,t_{ij}) \in \mathcal{S}} t_{ij}^2, \quad \mathbf{s}_{ij} = \text{vec}(\mathbf{u}_j \mathbf{u}_i^T). \end{aligned} \tag{19}$$

As B is positive semidefinite, we can write $B = B^{\frac{1}{2}} B^{\frac{1}{2}}$, where $B^{\frac{1}{2}}$ denotes the matrix square root of B. Using Schur Complement [3], the optimization problem (16) is equivalent to the following semidefinite programming (SDP) problem:

$$\begin{aligned} & \min_{\mathbf{z}, \nu} \ \nu + \mathbf{b}^T \mathbf{z} \\ & \text{s.t. } M \succeq 0 \text{ and } \begin{pmatrix} I_{m^2} & B^{\frac{1}{2}} \mathbf{z} \\ (B^{\frac{1}{2}} \mathbf{z})^T & \nu \end{pmatrix} \succeq 0, \end{aligned} \tag{20}$$

where I_{m^2} is the identity matrix of size $m^2 \times m^2$, and ν is a dummy variable serving as an upper bound of $\mathbf{z}^T B\mathbf{z}$. ν is an upper bound of $\mathbf{z}^T B\mathbf{z}$ because according to Schur Complement, the second semidefinite cone constraint is equivalent to $(B^{\frac{1}{2}}\mathbf{z})^T(B^{\frac{1}{2}}\mathbf{z}) \leq \nu$.

It is important to note that the computational complexity of the SDP problem is independent of the number of the objects and the number of the pairwise constraints, but depends mainly on m, the order of the spectral embedding F_m. The SDP problem has $m(m+1)/2+1$ variables, subject to two semidefinite cone constraints of sizes $m \times m$ and $(m^2+1) \times (m^2+1)$, respectively. For a small m, say, 15 as we set in all our experiments, this SDP problem can be solved within one minute, using CSDP 6.0.1[1] [2] in MATLAB 7.6.0 (R2008a) on a PC with 3.4 GHz CPU and 4GB RAM.

3.6. Clustering

After solving for M, we have two ways to perform clustering. The first is to apply kernel k-means to the kernel matrix K of the new representation, which can be obtained by $K = FF^T = F_m AA^T F_m^T = F_m M F_m^T$. Another way is to apply k-means directly on the low-dimensional representation $F_m M^{\frac{1}{2}}$. Since the kernel matrix of the representation $F_m M^{\frac{1}{2}}$ is $F_m M^{\frac{1}{2}}(F_m M^{\frac{1}{2}})^T = K$, the two ways are equivalent in performance because they optimize the same objective function with the same procedure, but the second way is with space complexity only $n \times m$ ($m \ll n$), in contrast to $n \times n$ in the first way. So we adopt the second way in our experiments.

3.7. The CCSR Algorithm

We summarize our algorithm in Algorithm 1. Note that the eigenvectors of the normalized graph Laplacian are used to parameterize the smooth data representation space, and thus play the role of regularizers. In this sense, we call the algorithm *Constrained Clustering with Spectral Regularization* (CCSR). For large-scale problems, the main computational cost of Algorithm 1 comes from computing the m eigenvectors of the sparse matrix $\bar{L}$. It can be efficiently performed using the Lanczos algorithm [8], which is specially designed for the eigenvalue decomposition of large sparse matrices.

Algorithm 1 Constrained Clustering with Spectral Regularization

Input: A data set $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$, pairwise constraint sets $\mathcal{M}$ and $\mathcal{C}$, and the number of clusters k.
Output: Cluster labels for the objects.

1: Form a sparse symmetric similarity matrix $W = [w_{ij}]$.
2: Form the normalized graph Laplacian $\bar{L} = I - D^{-1/2}WD^{-1/2}$, where I is the identity matrix of size $n \times n$ and $D = \text{diag}(d_1, ..., d_n)$ with $d_i = \sum_{j=1}^{n} w_{ij}$.
3: Compute the m eigenvectors $\mathbf{v}_1, ..., \mathbf{v}_m$ of $\bar{L}$ corresponding to the first m smallest eigenvalues. Let $F_m = (\mathbf{v}_1, ..., \mathbf{v}_m)$.
4: Solve the SDP problem (20) for M.
5: Apply k-means to the rows of $F_m M^{\frac{1}{2}}$ to form k clusters.

4. Experimental Results

In this section, we conduct extensive experiments on eight real data sets to evaluate the proposed constrained spectral clustering algorithm CCSR. For comparison, we present the results of two most related constrained spectral clustering algorithms, Spectral Learning (SL) [11] and Semi-Supervised Kernel k-means (SSKK) [13]. The results of Normalized Cuts[2] (NCuts) [20] are also shown for reference. SL modifies the similarities to 1's and 0's for must-link and cannot-link objects, respectively, and SSKK adds a penalty matrix, formed with pairwise constraints, to the original similarity matrix. The resulting similarity matrix of SSKK probably contains negative similarities. CCSR, SL, and SSKK directly address multi-class problems and exploit both must-link and cannot-link constraints. In contrast, the recently proposed constrained spectral clustering algorithms [16] and [6] make sense only for two-class problems, and the method in [24] handles only must-link constraints.

4.1. Clustering Evaluation

We use clustering error to evaluate a clustering result, which is a widely used criterion for the evaluation of clustering algorithms. This measure works by best matching a clustering result to the ground-truth cluster labels. Given a permutation mapping map$(\cdot)$ over the cluster labels, the clustering error with respect to map$(\cdot)$ is

$$1 - \frac{1}{n}\sum_{i=1}^{n} \delta(y_i, \text{map}(y_i')), \tag{21}$$

where y_i and y_i' are the ground-truth label and the obtained cluster label for object $\mathbf{x}_i$, respectively, $\delta(x, y) = 1$ if $x = y$ and $\delta(x, y) = 0$ otherwise, and n is the number of objects. The clustering error is defined as the minimal error over all possible permutation mappings.

4.2. Parameter Selection

In our algorithm CCSR, we set $m = 15$ and use CSDP 6.0.1 [2] as the SDP solver. In SSKK, we use its normalized

[1] https://projects.coin-or.org/Csdp/.

[2] http://www.cis.upenn.edu/∼jshi/software/

cut version since it performs best as reported in [13]. We follow [13] to set the constraint penalty in SSKK as $n/(kc)$ with n, k, and c being the numbers of objects, clusters, and pairwise constraints, respectively.

All the four algorithms are graph-based. To make fair comparisons, we use the same graphs for all the algorithms. Graph construction for clustering is a difficult problem, and so far we are not aware of any method that can build a perfect graph for given data. So as commonly done, we form several candidate graphs and select the one that gives the best result. Specifically, we use the weighted 20-nearest-neighbor graphs: the similarity between objects $\mathbf{x}_i$ and $\mathbf{x}_j$ is set to $w_{ij} = e^{-\|\mathbf{x}_i-\mathbf{x}_j\|^2/(2\sigma^2)}$ if $\mathbf{x}_i$ is among the 20-nearest neighbors of $\mathbf{x}_j$ or vice versa, and $w_{ij} = 0$ otherwise, where σ ranges over the set $\text{linspace}(0.1r, r, 5) \bigcup \text{linspace}(r, 10r, 5)$ with r being the averaged distance from each object to its 20-th nearest neighbor and $\text{linspace}(r_1, r_2, t)$ denoting the set of t linearly equally-spaced numbers between r_1 and r_2. More specifically, we construct 9 sparse graphs for each data set.

For each data set, 10 different numbers of pairwise constraints are randomly generated using the ground-truth cluster labels. For each set of pairwise constraints, the result is averaged over 50 realizations of k-means (for CCSR and SL) or weighted kernel k-means (for SSKK) with different random initializations. For a fixed number of pairwise constraints, the reported result is averaged over 50 realizations of different pairwise constraints. Note that NCuts does not utilize any pairwise constraints, and is engaged as the reference approach. Each reported result by NCuts is the average of 50 realizations of the discretization procedure.

4.3. Image Data

In this experiment, we test the three algorithms on four image databases, USPS[3], MNIST[4], Yale Face Database B (YaleB) [7], and a scene category data set (Scene) [18]. USPS and MNIST contain images of handwritten digits from 0 to 9 of sizes 16×16 and 28×28, respectively. There are 7291 training examples and 2007 test examples in USPS. MNIST has a training set of 60,000 examples and a test set of 10,000 examples. YaleB contains 5760 single light source images of 10 subjects captured under 576 viewing conditions (9 poses $\times$ 64 illumination). We downsample each image in YaleB to 30×40 pixels. The Scene data set was collected by Oliva and Torralba [18], containing 8 categories of natural scenes (see Fig. 3 for some sample images). We use the feature called Spatial Envelope [18] to represent each scene image, although other choices are certainly possible. The feature is a 512-dimensional vector, capturing the dominant spatial structure (naturalness, openness, roughness, expansion and ruggedness) of the scene.

[3]http://www-stat.stanford.edu/ tibs/ElemStatLearn/

[4]http://yann.lecun.com/exdb/mnist/

Figure 3. Example images in the Scene data set from [18].

Table 1. Four image data sets used in the experiment.

	USPS	MNIST	YaleB	Scene
# objects	9298	5139	5760	2688
# dimensions	256	784	1200	512
# clusters	10	5	10	8

For USPS, MNIST, and YaleB, the feature to represent each image is a vector formed by concatenating all the columns of the image intensities. In the experiments, we use all the examples from USPS, YaleB, and Scene, but use only digits 0 to 4 in the test set in MNIST due to its large amount of data. Table 1 summarizes the four data sets used.

The results are shown in Fig. 4. We can see that CCSR consistently performs best on the four data sets under almost all of the settings of pairwise constraints, especially on USPS and Scene. SL also obtains comparable results on MNIST and YaleB. In all the cases, CCSR and SL do improve the performance of NCuts by using pairwise constraints. In contrast, SSKK performs worst on three of the four data sets. This may come from the fact that, instead of the typical procedure that applies k-means to the spectral embedding, SSKK applies a kernel k-means-like algorithm directly to the modified kernel matrix, which is not guaranteed to be positive semidefinite [13].

To see how the proposed CCSR transforms the spectral embedding, Fig. 5 shows, for each data set, the distance matrices of the original data, the k-th (k is the number of clusters) order spectral embedding where NCuts obtains the best result, and the embedding of CCSR where CCSR obtains the best result. We can see that the block structure of the distance matrices of the embeddings output by CCSR is much more significant (Figs. 5(a3), (b3), (c3), and (d3)), compared to those of the k-th spectral embedding and the original data, for all the four data sets, meaning that after being transformed by CCSR, each cluster is more compact and different clusters are more separated. This experiment shows that CCSR can effectively propagate the sparse pairwise constraints to unconstrained objects, and does bias the spectral embedding to a better one for clustering.

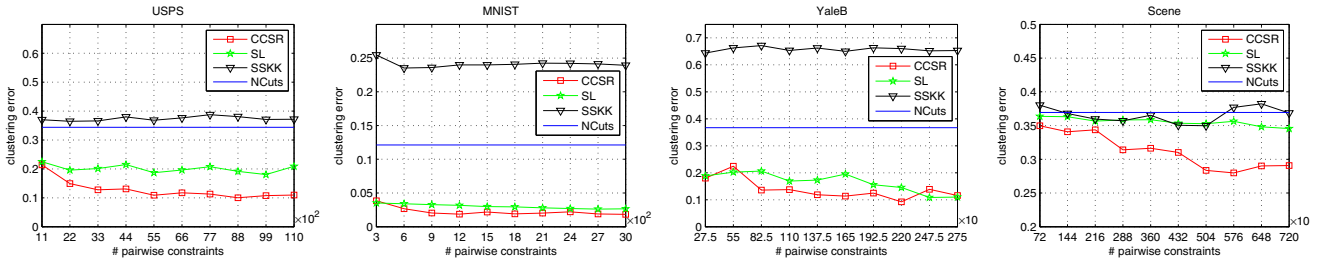

Figure 4. Constrained clustering results on the image data: clustering error vs. the number of pairwise constraints.

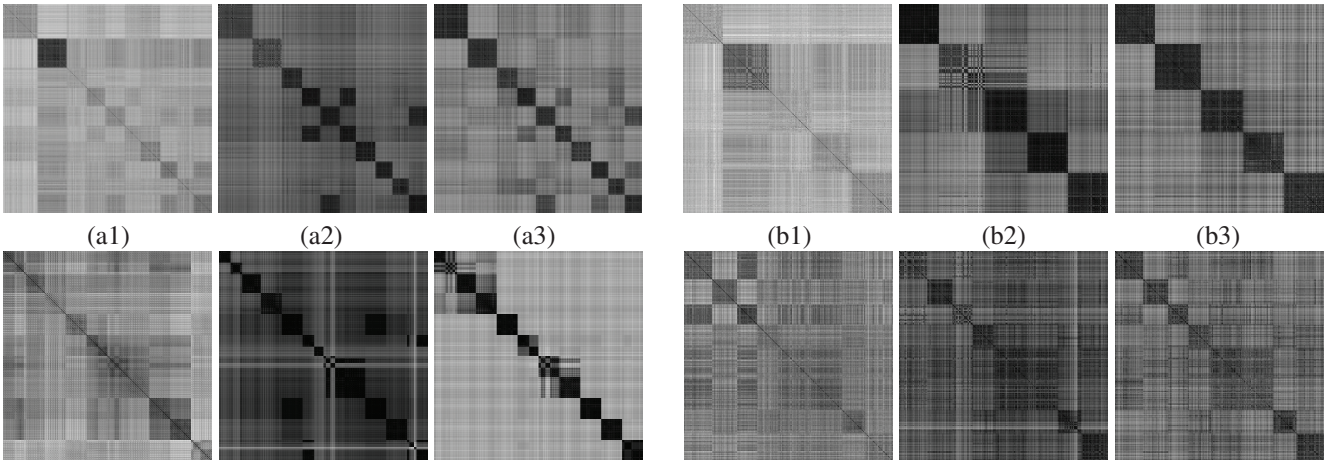

Figure 5. Distance matrices of the original data, the k-th order spectral embedding where NCuts obtains the best result, and the embedding of CCSR where CCSR gives the best result. For illustration purpose, the data are arranged such that objects within a cluster appear consecutively. The darker is a pixel, the smaller is the distance the pixel represents. (a1)–(a3) Results for USPS. (b1)–(b3) Results for MNIST. (c1)–(c3) Results for YaleB. (d1)–(d3) Results for Scene.

Table 2. Four UCI data sets used in the experiment.

	iris	wdbc	sonar	protein
# objects	150	569	208	116
# dimensions	4	30	60	20
# clusters	3	2	2	6

Note that the pairwise constraints used in the experiments are in fact very sparse. To see this, suppose that only 2% of the data from each cluster of USPS are used to form pairwise constraints, where objects from the same cluster are must-link and objects from different clusters are cannot-link. Then we have 14770 pairwise constraints in total, larger than 11000, the largest number of pairwise constraints we used for USPS in the experiment.

The execution times of different algorithms are comparable. For example, the times taken by CCSR, SL, SSKK, and NCuts on USPS are about 120, 68, 69, and 76 seconds, respectively, where CCSR costs 52 seconds to solve the SDP problem. All the algorithms run in MATLAB 7.6.0 (R2008a) on a PC with 3.4 GHz CPU and 4GB RAM.

4.4. UCI Data

We also conduct experiments on four data sets from UCI Machine Learning Repository[5]. UCI data are widely used to evaluate clustering and classification algorithms in machine learning. The four data sets we used in this experiment are described in Table 2. The results are shown in Fig. 6. Again we can see that CCSR performs best in most of the cases.

5. Conclusions

A novel regularization framework has been developed for constrained spectral clustering. Unlike previous methods that simply modify the similarities of constrained objects, we adapt the spectral embedding to accord with pairwise constraints by optimization. Our main idea is to select, from a smooth data representation space parameterized by a spectral embedding, the one that is most consistent with the pairwise constraints. Unlike previous methods that limit to two-class problems or handle only the must-link, the proposed method deals directly with multi-class problems and

[5]http://archive.ics.uci.edu/ml.

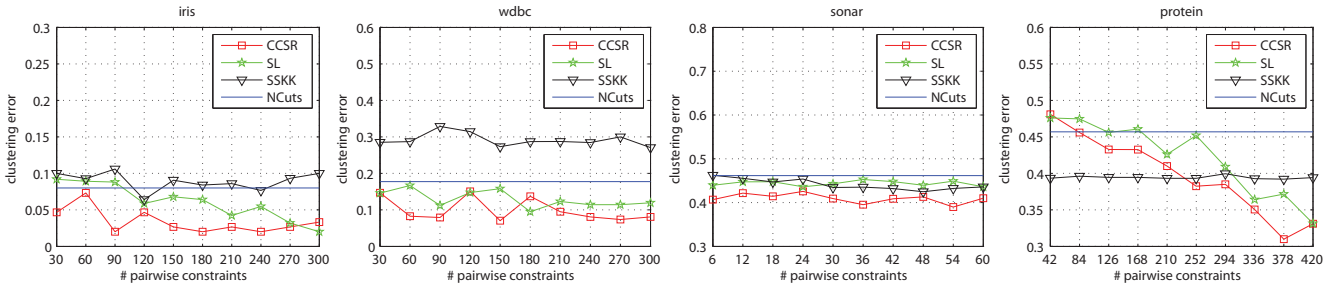

Figure 6. Constrained clustering results on the UCI data: clustering error vs. the number of pairwise constraints.

handle both the must-link and cannot-link, and can propagate the sparse constraints to unconstrained objects. Our framework is formulated as a small SDP problem that is convex and can be optimally solved efficiently, and more importantly, its complexity is independent of the number of the objects in the data set and the number of the given pairwise constraints, making it scale well to large-scale problems. Experimentally, our algorithm CCSR outperforms three most related state-of-the-art methods on eight real image and UCI data sets. Future work should consider automatic selection of the regularization parameter m and apply CCSR to large-scale real-world applications. In addition, our method will benefit from automatic robust graph construction [15].

Acknowledgements

This work was supported by two grants from the Research Grants Council of the Hong Kong SAR, China (Project No. CUHK 414306 and 415408).

References

[1] M. Belkin and P. Niyogi. Semi-Supervised Learning on Riemannian Manifolds. *Machine Learning*, 56(1):209–239, 2004.

[2] B. Borchers. CSDP, a C library for semidefinite programming. *Optimization Methods & Software*, 11-2(1-4):613–623, 1999.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[4] R. Bracewell and P. Kahn. The Fourier Transform and Its Applications. *American Journal of Physics*, 34:712, 1966.

[5] F. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[6] T. Coleman, J. Saunderson, and A. Wirth. Spectral clustering with inconsistent advice. In *ICML*, pages 152–159, 2008.

[7] A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: illumination cone models for face recognitionunder variable lighting and pose. *IEEE Trans. PAMI*, 23(6):643–660, 2001.

[8] G. Golub and C. Van Loan. *Matrix computations*. 1996.

[9] S. Hoi, R. Jin, and M. Lyu. Learning nonparametric kernel matrices from pairwise constraints. In *ICML*, pages 361–368, 2007.

[10] I. Jolliffe. *Principal component analysis*. Springer New York, 2002.

[11] S. Kamvar, D. Klein, and C. Manning. Spectral learning. In *IJCAI*, pages 561–566, 2003.

[12] D. Klein, S. Kamvar, and C. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *ICML*, pages 307–314, 2002.

[13] B. Kulis, S. Basu, I. Dhillon, and R. Mooney. Semi-supervised graph clustering: A kernel approach. In *ICML*, pages 457–464, 2005.

[14] Z. Li, J. Liu, and X. Tang. Pairwise constraint propagation by semidefinite programming for semi-supervised classification. In *ICML*, 2008.

[15] W. Liu and S.-F. Chang. Robust multi-class transductive learning with graphs. In *CVPR*, 2009.

[16] Z. Lu and M. Á. Carreira-Perpiñán. Constrained Spectral Clustering through Affinity Propagation. In *CVPR*, 2008.

[17] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[18] A. Oliva and A. Torralba. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[19] N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall. Computing Gaussian mixture models with EM using equivalence constraints. In *NIPS*, volume 16, pages 465–472, 2004.

[20] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. PAMI*, 22(8):888–905, 2000.

[21] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *ICML*, pages 1103–1110, 2000.

[22] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *ICML*, pages 577–584, 2001.

[23] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *NIPS*, pages 505–512, 2003.

[24] S. Yu and J. Shi. Segmentation Given Partial Grouping Constraints. *IEEE Trans. PAMI*, pages 173–180, 2004.

[25] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, 2004.