Mach Learn (2010) 79: 177–200
DOI 10.1007/s10994-009-5158-y

Ensemble clustering using semidefinite programming with applications

Vikas Singh · Lopamudra Mukherjee · Jiming Peng · Jinhui Xu

Received: 26 February 2009 / Revised: 9 October 2009 / Accepted: 12 October 2009 /
Published online: 22 December 2009

Abstract In this paper, we study the ensemble clustering problem, where the input is in the form of multiple clustering solutions. The goal of ensemble clustering algorithms is to aggregate the solutions into one solution that maximizes the agreement in the input ensemble. We obtain several new results for this problem. Specifically, we show that the notion of agreement under such circumstances can be better captured using a 2D string encoding rather than a voting strategy, which is common among existing approaches. Our optimization proceeds by first constructing a non-linear objective function which is then transformed into a 0-1 Semidefinite program (SDP) using novel convexification techniques. This model can be subsequently relaxed to a polynomial time solvable SDP. In addition to the theoretical contributions, our experimental results on standard machine learning and synthetic datasets show that this approach leads to improvements not only in terms of the proposed agreement measure but also the existing agreement measures based on voting strategies. In addition, we identify several new application scenarios for this problem. These include combining

Editors: Nicolo Cesa-Bianchi, David R. Hardoon, and Gayle Leen.

V. Singh
Department of Biostatistics & Medical Informatics, University of Wisconsin–Madison, Madison, WI, USA
e-mail: vsingh@biostat.wisc.edu

L. Mukherjee (✉)
Department of Mathematics and Computer Science, University of Wisconsin–Whitewater, Whitewater, WI, USA
e-mail: mukherjl@uww.edu

J. Peng
Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL, USA
e-mail: pengj@uiuc.edu

J. Xu
Department of Computer Sci. and Eng., The State University of New York at Buffalo, Buffalo, NY, USA
e-mail: jinhui@cse.buffalo.edu

multiple image segmentations and generating tissue maps from multiple-channel Diffusion Tensor brain images to identify the underlying structure of the brain.

Keywords Ensemble clustering · Semidefinite programming · Cluster ensembles · Segmentation aggregation

1 Introduction

Aggregation (of data) from multiple sources is typically used to reduce noise and obtain a more accurate characterization of information. If data from multiple sources are in the form of clustering schemes (or clusters), we may want to aggregate such knowledge to derive a better clustering with fewer errors. The Ensemble Clustering problem refers to such a scenario where the target is to 'combine' multiple clustering solutions or partitions of a set into a single consolidated clustering that maximizes the information shared (or 'agreement') among all available clustering solutions. The need for this form of clustering arises in many applications, especially real world scenarios with a high degree of uncertainty such as image segmentation with poor contrast, and computer assisted disease diagnosis. It is quite common that a single clustering algorithm may not yield satisfactory results, while multiple algorithms may individually make imperfect choices, assigning some elements to wrong clusters. Usually, by considering the results of several *different* clustering algorithms *together*, one may be able to mitigate degeneracies in individual solutions and consequently obtain better solutions. These issues have been investigated in the literature recently in the context of the stability and accuracy of the solutions (Fred and Jain 2006; Dudoit and Fridlyand 2003; Kuncheva and Vetrov 2006). For instance, Dudoit and Fridlyand (2003) applied bagging based procedures to cluster analysis in an effort to reduce the variability via averaging. In a tumor sample clustering application (Dudoit and Fridlyand 2003), ensemble methods were found to provide robustness to variability in the clustering solution due to starting conditions or convergence to poor local optimum solutions. The experiments in Kuncheva and Vetrov (2006) evaluated the stability of the solution (as a measure of the validity of the clustering). Using the adjusted Rand Index and an entropy based measure of the consensus to characterize stability, the results (Kuncheva and Vetrov 2006) indicated a significant improvement relative to individual solutions (especially for a large number of clusters). A similar behavior was observed in Fred and Jain (2006) on several synthetic and real data-sets, where the stability of the clustering solutions was evaluated using bootstrapping techniques.

However, these are just few among many possible applications. There are numerous other scenarios such as knowledge reuse and applications in distributed computing outlined in Strehl and Ghosh (2003). The idea has also been employed successfully for microarray data clustering analysis (Filkov and Skiena 2003) and document cluster analysis (Bansal et al. 2002). In fact, the problem models a variety of situations where we seek to perform aggregation to obtain a representative clustering with minimal inconsistency and maximum mutual agreement.

Formally, given a data set $D = (d_1, d_2, \dots, d_n)$, a set of clustering solutions $C = (C_1, C_2, \dots, C_m)$ obtained from m different clustering algorithms is called a *cluster ensemble*. Each solution, C_i, is the partition of the data into at most k different clusters. The *Ensemble Clustering* problem requires one to use the individual solutions in C to partition D into k clusters such that information shared (or agreement) among the solutions of C is maximized. We will make the notion of 'agreement' precise shortly.

The remainder of this paper is structured as follows. In Sect. 1.1 we discuss some related work and results on this problem. In Sects. 2, 3, we describe the intuition and main ideas underlying the algorithm. Sections 4–6 include the body of the paper—we start with an Integer Program to formalize the problem and finally obtain a Semidefinite program. Section 7 outlines a rounding mechanism, and we present some promising experimental results in Sect. 8. Relative to the conference version of this work (Singh et al. 2007), this paper includes detailed steps of the modeling and rounding scheme, as well as an extensive set of experiments in the context of image segmentation applications.

1.1 Previous works

The Ensemble Clustering problem was introduced by Strehl and Ghosh (2003), and has since generated a great deal of interest. A number of algorithms have been proposed for Cluster Ensembles or related problems (Strehl and Ghosh 2003; Fern and Brodley 2004; Monti et al. 2003; Gionis et al. 2005; Ailon et al. 2005; Bansal et al. 2002; Charikar et al. 2005; Giotis and Guruswami 2006). A common feature of many approaches is modeling an instance of the ensemble clustering problem as a graph with the edges denoting some measure of example similarity inferred from the ensemble. In the next section, we briefly review some of the popular techniques such as Instance Based Graph and Cluster Based Graph from Strehl and Ghosh (2003) to showcase existing approaches to the problem.

The Instance Based Graph Formulation (IBGF) (Strehl and Ghosh 2003)[1] first constructs a fully connected graph $G = (V, W)$ for the input cluster ensemble $C = (C_1, \ldots, C_m)$ and each node represents an element of D. The weight w_{ij} between the node pair (v_i, v_j) is defined as the number of cluster algorithms in C that assign the nodes d_i and d_j to the same cluster (i.e., w_{ij} measures the *togetherness* frequency of d_i and d_j). Then, standard techniques are used to solve a graph partitioning problem and obtain a final clustering solution.

Meta clustering algorithm (MCLA) (Strehl and Ghosh 2003) follows a Cluster Based graph formulation (CBGF), i.e., it is based on clustering clusters. A given cluster ensemble is represented as $C = \{C_{11}, \ldots, C_{mk}\} = \{\bar{C}_1, \ldots, \bar{C}_{mk}\}$ where C_{ij} denotes the ith cluster of the jth algorithm in C. Like IBGF, this approach also constructs a graph, $G = (V, W)$, to model the correspondence (or 'similarity') relationship among the mk clusters, where the similarity matrix W reflects the Jaccard similarity measure between the clusters $\bar{C}_i$ and $\bar{C}_j$

$$w_{ij} = \frac{|\bar{C}_i \cap \bar{C}_j|}{|\bar{C}_i \cup \bar{C}_j|}.$$

The similarity measure is then used to partition the graph so that the clusters of the same group are similar to one another. Once a partition of the clusters is obtained a final clustering is determined by considering each group of clusters as a metacluster; elements are assigned to the metacluster with which they are most frequently associated. We note that variations of the problem have connections to other well known problems such as Rank Aggregation Clustering for which constant factor approximation algorithms have been proposed (Ailon et al. 2005). In addition to the above mentioned approaches, some simpler learning algorithms have also been proposed. These include Topchy et al. (2003) which proposed generating weak clustering by projection onto 1D plane or bisection by random hyperplanes followed

[1]Note that IBGF was terminology used in Fern and Brodley (2004) to refer to and review the Cluster based similarity partitioning algorithm (CSPA) in Strehl and Ghosh (2003).

by k-means clustering and Topchy et al. (2003) which proposed a mixture modeling based approach for ensemble clustering.

In summary, most existing algorithms rely, at least at the basic level, on a graph construction. Element pairs (cluster pairs or item pairs) are then evaluated and their edges are assigned a weight that reflects their similarity. In the next section, we will discuss these issues in more detail.

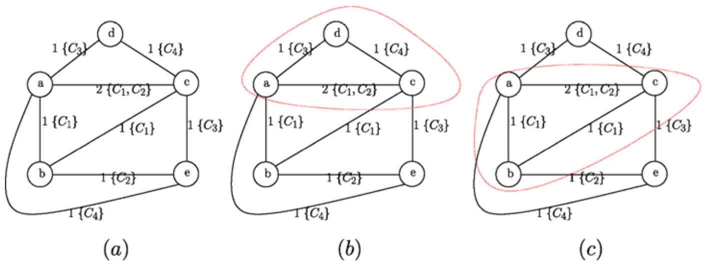
2 Two is company, three is a crowd

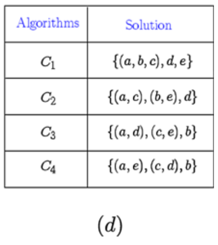
Consider an example where one is 'aggregating' recommendations made by a group of family and friends for dinner table seating assignments at a wedding. The hosts would like each 'table' to be able to find a common topic of dinner conversation. Now, consider three persons, Tom, Dick, and Harry invited to this reception. Tom and Dick share a common interest in Shakespeare, Dick and Harry are both surfboard enthusiasts, and Harry and Tom attended college together. Because they had strong pairwise similarities, they were seated together but had a rather dull evening.

The three guests indeed had strong common interests when considered two at a time, but there was weak communion as a group. All existing algorithms represent the similarity measure between elements in D as a scalar value assigned to the edge joining their corresponding nodes in the graph. This weight is essentially a 'vote' reflecting the number of algorithms in the ensemble that assigned those two elements to the same cluster. The mechanism seems perfect until we ask if strong pairwise coupling necessarily implies coupling for a larger group as well. The weight metric considering two elements does not retain information about which algorithms assigned them together. This information is perhaps not useful directly, but when we seek to characterize higher order interactions (similarity) and expanding the group to include more elements, it is not quite clear if a common feature even exists such that the larger group is similar (if one considers that feature). It seems natural to assign a higher priority to triples or larger groups of people that were recommended to be seated together (must be similar under at least one feature) compared to groups that were never assigned to the same table by any person in the recommendation group (clustering algorithm), notwithstanding pairwise evaluations. While this problem seems to be a distinctive disadvantage for only the IBGF approach; it also affects the meta-clustering approach. Here the analogy of pairwise similarity extends to clusters. While the overlap of pairwise clusters may be high, it is still difficult to precisely infer the degree of consent in a cluster that comprises of all items of these clusters.

We provide a simple toy example (see Fig. 1) to convey this intuition. Consider nodes a, b, c, d, and e in D and four clustering algorithms ($C_1, \ldots, C_4$) dividing D into three clusters. For a voting based algorithm, Δacd (Fig. 1(b)) and Δabc (Fig. 1(c)) are equivalent because they have equal weights (i.e., 4). However, closer inspection shows that these two sets of points are quite different in terms of which clusters put them together. In Δacd, each pair of elements is put in the same cluster by one algorithm, but *no* solution put a, c, and d in the same cluster. However in Δabc, all three nodes were assigned to the same cluster at least by one algorithm (i.e., C_1). Voting strategies are unable to incorporate such information. This problem may naturally amplify in larger datasets.

The two examples above bring out one central idea. It is reasonable to expect that the solution from an 'ensemble' algorithm be such that each cluster has a high vote as a group as a result of a high cohesiveness among members *across* a feature set (the solutions from multiple clustering algorithms). Pairwise (scalar) edge weights do not provide sufficient information for inputs where cluster sizes and the feature set on which clustering is performed





Algorithms	Solution
C_1	$\{(a,b,c),d,e\}$
C_2	$\{(a,c),(b,e),d\}$
C_3	$\{(a,d),(c,e),b\}$
C_4	$\{(a,e),(c,d),b\}$

(d)

Fig. 1 An input ensemble represented as a graph in (**a**) and a table in (**d**). (**b**) and (**c**) evaluate the goodness of two subgraphs with equal weight w.r.t. pairwise evaluations

are larger. Regardless of the sophistication of the graph partitioning algorithm employed on such graphs later on, it is likely to yield sub-optimal results.

3 Main ideas

To model the intuition above, we generalize the similarity metric to maximize similarity or ‘agreement’ by an appropriate encoding of the solutions obtained from individual clustering algorithms. More precisely, in our generalization the similarity is no longer just a scalar value but a multidimensional string. The ensemble clustering problem thus reduces to a form of string clustering problem where our objective is to assign *similar* strings to the same cluster.

The encoding into a string is done as follows. The set of data points is given as D with $|D| = n$, where $|\cdot|$ will denote the number of examples/items in a set. Let m be the number of clustering algorithms with each solution having no more than k clusters. We represent all input information (ensemble) as a single $3D$ matrix, $A \in \Re^{n\times m\times k}$. For every data element $d_l \in D$, $A_l \in \Re^{m\times k}$ is a matrix whose elements are defined by

$$A_{lij} = \begin{cases} 1 & \text{if } d_l \text{ is assigned to cluster } i \text{ by } C_j; \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

It is easy to see that the summation over every row of A_l equals 1. We call each A_l an A-string. Our goal is to cluster the elements $D = (d_1, d_2, \dots, d_n)$ based on similarity of their corresponding binary matrices (or A-strings).

We now consider how to compute the clusters based on the similarity (or dissimilarity) of strings. We note that Gasieniec et al. (2000) discussed the so-called Hamming radius p-clustering (HRC) and Hamming diameter p-clustering (HDC) problems on strings. Though their results shed considerable light on the hardness of string clustering with the selected distance measures, their proposed method is polynomial time solvable only if the string sizes or the number of strings are considered fixed, which is not feasible with our model. Fortunately, our analysis reveals that a simpler objective is sufficient to capture the essence of similarity maximization in clusters using certain special properties of the A-strings.

Our approach is partly inspired by the classical k-means clustering where all data points are assigned to the cluster based on the shortest distance to the cluster center. Imagine an ideal input instance for the ensemble clustering problem (all clustering algorithms behave similarly). Here, we will have only k unique members among n A-strings. The partitioning simply assigns similar strings to the same partition. The representative for each cluster is exactly like the members of the clusters, i.e. a valid A-string, and can be viewed as a *center* in a geometric sense. General input instances will obviously be non-ideal and are likely to contain far more than k unique members. Naturally, the centers of the clusters will vary from its members. This variation is *noise* or *disagreement* within the clusters, our objective is to find a set of clusters (and centers) such that the noise is minimized and we move very close to the ideal case.

Next, we use an example to illustrate how to construct the center A strings and how disagreement in the cluster can be captured in this framework. Let a cluster i in an optimal solution contain items $(d_1, d_2, \dots, d_7)$. A certain algorithm C_j in the input ensemble clusters items (d_1, d_2, d_3, d_4) in cluster s and (d_5, d_6, d_7) in cluster p. If the center of cluster i were an example in the input, we can estimate its assignment by algorithm C_j. The probability it assigns the center to cluster s (and respectively, p) is $4/7$ (and respectively, $3/7$). That is, we pick the choice with the higher probability and assign the center to such a cluster. It can be verified that this choice minimizes the dissent (w.r.t. the center) of all examples in cluster i. The A-string for the center of cluster i will have a "1" at position (j, s). The assignment of A-string (items) to clusters is unknown; however, if it were provided, we could find the centers for all other clusters $i \in \{1, \dots, k\}$ by computing the average value at every cell of the A matrices (i.e., cluster members) and rounding the largest value in every row to 1 (rest to 0) and assigning this as the cluster center. Hence, the dissent within a cluster can be quantified simply by averaging the matrices of elements belonging to the cluster and computing the difference to the center. Our goal is to find such an assignment and group the data items so that the sum of the absolute differences of the averages of clusters to their centers (i.e., dissent) is minimized. Note that when the number of clusters in the input ensemble is different, we can simply "pad" the A strings with zeros appropriately.

In the subsequent sections, we will introduce our optimization framework for ensemble clustering based on the discussed ideas.

4 Integer Program for Model 1

We start with a discussion of an Mixed Integer Program (MIP, for short) formulation for ensemble clustering. For convenience, we denote the final clustering solution by $C^* = \{C_1^*, C_2^*, \dots, C_k^*\}$ and C_{ij} the cluster i by the algorithm j. Each cluster denotes a set of elements. The variables (or unknowns) which the IP on the completion of its optimization will assign values to are $X \in \Re^{n\times k}$ and $s \in \Re^{k\times m\times k}$. Here X denotes the assignment matrix of the data items to the clusters in the final ensemble output and s denotes the A-string of the center of clusters in the output ensemble. To denote individual elements of these matrices, we use the notations X_{lp} and s_{ijp} respectively. These variables are defined as follows.

$$X_{lp} = \begin{cases} 1 & \text{if } d_l \in C_p^*; \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$s_{ijp} = \begin{cases} 1 & \text{if } C_p^* = \arg\max_{i=1,\dots,k}\{|C_p^* \cap C_{ij}|\} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

This indicates X_{lp} is set to 1 if data item d_l is assigned to cluster C_p^* (zero otherwise) and s_{ijp}, is set to 1 only if the overlap of cluster C_p^* with cluster C_{ij} is greater than or equal to its overlap with all other clusters $C_{i'j} : i' \neq i$. We mention that the above definition implies that for a fixed index p, its center, s_{ijp} also provides an indicator to the cluster most similar to

C_p^* in the set of clusters produced by the clustering algorithm C_j. At the cost of repetition, the dimension of the A strings are $n \times m \times k$, as defined in Sect. 3. We can now introduce the following MIP.

$$\min_{s,X} \quad \sum_{p=1}^{k}\sum_{i=1}^{k}\sum_{j=1}^{m}\left| s_{ijp} - \frac{\sum_{l=1}^{n} A_{lij} X_{lp}}{\sum_{l=1}^{n} X_{lp}} \right| \tag{4}$$

$$\text{s.t.} \quad \sum_{p=1}^{k} X_{lp} = 1 \quad \forall l \in \{1, \ldots, n\}, \tag{5}$$

$$\sum_{l=1}^{n} X_{lp} \geq 1 \quad \forall p \in \{1, \ldots, k\}, \tag{6}$$

$$\sum_{i=1}^{k} s_{ijp} = 1 \quad \forall j \in \{1, \ldots, m\},\ \forall p \in \{1, \ldots, k\}, \tag{7}$$

$$X_{lp} \in \{0, 1\}, \qquad s_{ijp} \in \{0, 1\}. \tag{8}$$

The objective function in (4) minimizes the sum of the difference between s_{ijp} (the center for cluster C_p^*) and the average of all A_{lij} bits of the data elements d_l assigned to cluster C_p^*. Recall that s_{ijp} will be 1 if C_{ij} is the most similar cluster to C_p^* among all the clusters produced by algorithm C_j. Hence, if $s_{ijp} = 0$ and $\frac{\sum_{l=1}^{n} A_{lij} X_{lp}}{\sum_{l=1}^{n} X_{lp}} \neq 0$, the value $|s_{ijp} - \frac{\sum_{l=1}^{n} A_{lij} X_{lp}}{\sum_{l=1}^{n} X_{lp}}|$ represents the percentage of data elements in C_p^* that *do not* consent with the majority of the other elements in the group w.r.t. the clustering solution provided by C_j. In other words, we are trying to minimize the dissent and maximize the consent simultaneously. The remaining constraints are relatively simple—(5) enforces the condition that a data element should belong to precisely one cluster in the final solution and that every cluster must have size at least 1; (7) ensures that s is an appropriate A-string for every cluster center.

5 0-1 Semidefinite Program for Model 1

The formulation given by (4)–(8) is a mixed integer program (MIP, for short) with a nonlinear objective function in (4). Solving this model optimally, however, is extremely challenging—(a) the variables as per the constraint (8) are discrete; (b) the objective is nonlinear *and* non-convex. One possible way of attacking the problem is to 'relax' it to some polynomially solvable problems such as SDP.

We now model the integer program of Sect. 4 into a 0-1 SDP. Our primary focus would be to convert the nonlinear form in (4) into a 0-1 SDP form. By introducing additional variable matrices $t \in R^{k\times m\times k}$, and $c \in R^{k\times m\times k}$ whose sizes are same as s, we rewrite (4) as

$$\min_{s,X,t} \quad \sum_{i=1}^{k}\sum_{j=1}^{m}\sum_{p=1}^{k} t_{ijp} \tag{9}$$

$$s_{ijp} - c_{ijp} \leq t_{ijp} \quad \forall i, p, j, \tag{10}$$

$$c_{ijp} - s_{ijp} \leq t_{ijp} \quad \forall i, p, j, \tag{11}$$

where the term c_{ijp} represents the second term in (4) defined by

$$c_{ijp} = \frac{\sum_{l=1}^{n} A_{lij} X_{lp}}{\sum_{l=1}^{n} X_{lp}} \quad \forall i, p, j. \tag{12}$$

Since both A_{lij} and X_{lp} are binary, (12) can be rewritten as

$$c_{ijp} = \frac{\sum_{l=1}^{n} A_{lij}^2 X_{lp}^2}{\sum_{l=1}^{n} X_{lp}^2} \quad \forall i, p, j. \tag{13}$$

Let us introduce a matrix variable $y \in \Re^{n\times n}$ where lth element of column vector y_p is defined as

$$y_{lp} = \frac{X_{lp}}{\sqrt{\sum_{l=1}^{n} X_{lp}^2}} \tag{14}$$

Since $A_{ij} \in \Re^n$ is a vector whose lth element has value A_{lij} which allows us to represent (13) as

$$c_{ijp} = \mathrm{tr}(y_p^T B_{ij} y_p) \quad \forall i, j, p, \tag{15}$$

where $B_{ij} = \mathrm{diag}(A_{ij})$ is a diagonal matrix in $\Re^{n\times n}$ with $(B_{ij})_{ll} = A_{lij}$. Equation (15) can then be written as

$$c_{ijp} = \mathrm{tr}(B_{ij} Z_p), \tag{16}$$

where $Z_p = y_p y_p^T$ is a positive semidefinite matrix in $\Re^{n\times n}$ satisfying the properties

$$Z_p^2 = Z_p, \quad Z_p \succeq 0, \tag{17}$$

the symbol, $\succeq$, denotes positive semidefiniteness. Now, we rewrite the constraints for X in terms of Z. Equation (5) can be transformed to

$$\sum_{p=1}^{k} \sum_{l'=1}^{n} Z_{pll'} = 1 \quad \forall l \in [1, n], \tag{18}$$

where $Z_{pll'}$ refers to the (l, l') entry of matrix Z_p. Notice that (6) is automatically satisfied by the following constraints on the elements of Z_p.

$$\sum_{l=1}^{n} Z_{pll'} = 1 \quad \forall p \in [1, k], \tag{19}$$

$$\sum_{l'=1}^{n} Z_{pll'} \le 1 \quad \forall p \in [1, k], \forall l \in [1, n]. \tag{20}$$

Since Z_p is a symmetric projection matrix by construction, (9)–(19) constitute a precisely defined 0-1 SDP that can be expressed in trace form as follows.

$$\min_{s,Z,t} \quad \sum_{p=1}^{k} \mathrm{tr}(\mathrm{diag}(t_p e_k)) \tag{21}$$

$$\text{s.t.} \quad (s_{ijp} - t_{ijp} - c_{ijp}) \le 0 \quad \forall i, j, p, \tag{22}$$

$$(c_{ijp} - s_{ijp} - t_p) \le 0 \quad \forall i, j, p, \tag{23}$$

$$\left(\sum_{p=1}^{k} Z_p\right) e_n = e_n \quad \forall p \in [1, k], \tag{24}$$

$$\operatorname{tr}(Z_p) = 1 \quad \forall p \in [1, k] \tag{25}$$

$$S_p e_k = e_m \quad \forall p \in [1, k], \tag{26}$$

$$Z \ge 0, \qquad Z_p^2 = Z_p, \qquad Z_p = Z_p^T, \qquad S_p \in \{0, 1\}, \tag{27}$$

where $c_{ijp} = \operatorname{tr}(B_{ij} Z_p)$, t_p refers to the $k \times m$ matrix whose elements are t_{ijp} for all (i, j) and $e_n \in \Re^n$ is a vector of all $1s$.

The experimental results for this model indicate that it performs very well in practice (see Sect. 8). However, because we must solve the model while maintaining the requirement that S_p be binary (otherwise, the problem becomes ill-posed), a branch and bound type method is needed. In the subsequent sections, we will make several changes to this framework based on additional observations in order to obtain a polynomial algorithm for the problem.

6 Integer Program and 0-1 Semidefinite Program for Model 2

Recall the definition of the variables c_{ijp}, which can be interpreted as the size of the overlap between the cluster C_p^* in the final solution and C_{ij}, and is proportional to the cardinality of C_p^*. Let us define

$$c_{i^*jp} = \max_{i=1,\dots,k} c_{ijp}.$$

We also define vector variables q_{jp} whose i^{th} element is $s_{ijp} - c_{ijp}$. In the IP Model 1, we try to minimize the sum of all the L_1-norms of q_{jp}. The main difficulty in the previous formulation stems from the fact that c_{ijp} is a fractional function w.r.t. the assignment matrix X. Fortunately, we note that since entries of c_{ijp} are fractional satisfying $\sum_{i=1}^{k} c_{ijp} = 1$ for any fixed j, p, their sum of squares is maximized when its largest entry is as high as possible. Thus, minimizing the function $1 - \sum_{i=1}^{k} (c_{ijp})^2$ is a reasonable substitute to minimizing the sum of the L_1-norms in the IP model 1. The primary advantage of this observation is that we do not need to know the ‘index’ (i^*) of the maximal element c_{i^*jp}. There is also an intuitive explanation why such an objective can be used in place of the 1-norm objective in Model 1. Consider a variable $t \in [0, 1]$. We observe that $1 - t^2 \ge 1 - t$ for all t in $[0, 1]$. This becomes an equality only at integral values of t (i.e., $t = 0$ or $t = 1$). If we minimize our objective, thus intuitively the use of the squared model (of the form $1 - t^2$) will force us to find a solution closer to the original model (i.e., one with binary constraints on the variables), relative to a linear model based on $(1 - t)$. We now explain our model. As before, X denotes the assignment matrix. We no longer need the variable s, as it can be easily determined from the solution. This yields the following MIP.

$$\min_X \quad \sum_{p=1}^{k} \sum_{j=1}^{m} \left(\sum_{l=1}^{n} X_{lp}\right) \left(1 - \sum_{i=1}^{k} (c_{ijp})^2\right) \tag{28}$$

$$\text{s.t.} \quad \sum_{p=1}^{k} X_{lp} = 1 \quad \forall l \in [1, n], \tag{29}$$

$$\sum_{l=1}^{n} X_{lp} \geq 1 \quad \forall p \in [1, k], \tag{30}$$

$$X_{lp} \in \{0, 1\}. \tag{31}$$

We next discuss how to transform the above problem to a 0-1 SDP. For this, we first note that the objective function (28) can be expressed as follows.

$$\min_X \sum_{p=1}^{k} \sum_{j=1}^{m} \left(\left(\sum_{l=1}^{n} X_{lp} \right) - \sum_{i=1}^{k} \frac{(\sum_{l=1}^{n} A_{lij} X_{lp})^2}{\sum_{l=1}^{n} X_{lp}} \right), \tag{32}$$

which can be equivalently stated as

$$\min_X \left(nm - \sum_{p=1}^{k} \sum_{j=1}^{m} \sum_{i=1}^{k} \underbrace{\frac{(\sum_{l=1}^{n} A_{lij} X_{lp})^2}{\sum_{l=1}^{n} X_{lp}}}_{\text{term 2}} \right). \tag{33}$$

The numerator of term-2 above can be rewritten as

$$\left(\sum_{l=1}^{n} A_{lij} X_{lp} \right)^2 = (A_{1ij} X_{1p} + \cdots + A_{nij} X_{np})^2 = (A_{ij}^T X_p)^2 = X_p^T A_{ij} A_{ij}^T X_p, \tag{34}$$

where X_p is the pth column vector of X. Therefore, the second term of (33) can be written as

$$\begin{aligned}
&= \operatorname{tr}\left(\sum_{p=1}^{k} \sum_{j=1}^{m} \sum_{i=1}^{k} X_p^T A_{ij} A_{ij}^T (X_p^T X_p)^{-1} X_p \right) \\
&= \operatorname{tr}\left(\sum_{p=1}^{k} \sum_{j=1}^{m} \sum_{i=1}^{k} A_{ij} A_{ij}^T Z_p \right) \\
&= \operatorname{tr}\left(\sum_{j=1}^{m} \sum_{i=1}^{k} A_{ij} A_{ij}^T Z \right) \\
&= \operatorname{tr}\left(\sum_{j=1}^{m} B_j Z \right) \\
&= \operatorname{tr}(BZ).
\end{aligned} \tag{35}$$

In (35), $Z_p = X_p (X_p^T X_p)^{-1} X_p^T$ (same as in IP model 1), and $Z = \sum_{p=1}^{k} Z_p$ and $B = \sum_{j=1}^{m} B_j$. Since each matrix Z_p is a symmetric projection matrix and $X_{i'_1}$ and $X_{i'_2}$ are orthogonal to each other when $i'_1 \neq i'_2$, we immediately have the following result.

Lemma 1 *Z is a projection matrix of the form $X(X^T X)^{-1} X$.*

The above property used also in Peng and Wei (2007) is originally attributed to an anonymous referee in Gordon and Henderson (1977). Finally, we derive the 0-1 SDP formulation for the problem (28)–(31) as follows.

$$\min_{Z} \quad (nm - \operatorname{tr}(BZ)) \tag{36}$$

$$\text{s.t.} \quad Ze_n = e_n, \tag{37}$$

$$\operatorname{tr}(Z) = k, \tag{38}$$

$$Z \geq 0, \qquad Z^2 = Z, \qquad Z = Z^T. \tag{39}$$

7 Relaxation and rounding

Overview The relaxation to (36)–(39) exploits the fact that Z is a projection matrix satisfying $Z^2 = Z$. This allows replacing the last three constraints in (39) as $I \succeq Z \succeq 0$. By establishing the result that any feasible solution to the second formulation of 0-1 SDP, $Z^{\mathtt{feas}}$ is a rank k matrix, we first solve the relaxed SDP using SeDuMi (Sturm 1999) and Yalmip (Löfberg 2004), take the rank k projection of Z^* and then adopt a rounding based on a variant of the winner-takes-all approach to obtain a solution in polynomial time. The details follow.

7.1 Relaxation

In this subsection, we describe our relaxation method that allows us to obtain a solution to SDP Model 2. SDP relaxation approaches have a rich history with several such schemes proposed for max-cut (Goemans and Williamson 1995; Anjos and Wolkowicz 1999) and 0-1 non-linear problems (Lasserre 2001). In addition, some of the SDP relaxation models most closely related to this approach are Peng and Wei (2007), Xing and Jordan (2003).

Recall that Z is a projection matrix satisfying $Z^2 = Z$, which implies that Z is also a positive semidefinite matrix. A straightforward relaxation to (35) of 0-1 SDP Model 2 can be done by replacing it with the following relaxed condition,

$$I \succeq Z \succeq 0. \tag{40}$$

Further, the conditions on Z in (35) ensure that all entries of Z are non-negative and the sum of each row is equal to 1. This means that the eigen values of Z are less than or equal to 1. Hence, the constraint $I \succeq Z$ can be waived. Thus, the SDP relaxation can be obtained by converting the constraint as follows.

$$Z \succeq 0, \qquad Z \geq 0. \tag{41}$$

The relaxed program is feasible and bounded and is restated as follows.

$$\min_{Z} \quad (nm - \operatorname{tr}(BZ)) \tag{42}$$

$$\begin{aligned} \text{s.t.} \quad & Ze_n = e_n, \\ & \operatorname{tr}(Z) = k, \\ & I \succeq Z \succeq 0, \quad Z \geq 0. \end{aligned} \tag{43}$$

7.2 Rounding

Due to relaxation, the approximate solution obtained by solving the relaxed problem may not always be a feasible solution for our 0-1 SDP model. Specifically, the approximate Z_p may not be of the form $y_p y_p^T$. So, we need to recover the closest feasible solution to our 0-1 SDP from the obtained solution through a rounding procedure. We will discuss a few technical results before proceeding.

Fact 1 If A has eigen values $s_1, s_2, \dots, s_n$, then $\|A\|_F^2 = \sum s_i^2$.

Fact 2 A projection matrix A has eigen values either 0 or 1.

Lemma 2 *Let $Z_p^{\mathtt{feas}}$ be the feasible solution for cluster p in the final solution based of the definition in Sect. 5. Any feasible solution to the first formulation of* 0-1 *SDP, $Z_p^{\mathtt{feas}}$ is a rank* 1 *matrix.*

Proof Let X_p denote column p in X. X_p has 1's in positions where the corresponding data item belongs to cluster p. Let the cardinality of cluster p be p. Then, X_p will have exactly $p \le n$ 1's in its entries. $\|X_p\|_2$ will be $\frac{1}{\sqrt{p}}$. Recall that

$$y_p^{(l)} = \frac{X_{lp}}{\|X_p\|_2}. \tag{44}$$

A direct implication is that y_p will have $\frac{1}{\sqrt{p}}$ in positions corresponding to non-zero entries of X_p. Since $Z_p = y_p y_p^T$, Z_p will have $(\frac{1}{p})$ in exactly p^2 positions, all other entries being zero. Therefore, the sum of squares of all elements in Z_p is given by $\|Z_p\|_F^2 = p^2 \cdot (\frac{1}{p})^2 + (n^2 - p^2) \cdot 0 = 1$. From Fact (2), we know that the eigen values of Z_p will be 1 or 0 since it is a projection matrix. Using Fact (1) and Fact (2), Z_p must have one non-zero eigen value, all other eigen values must be 0. Since rank of a matrix is simply the number of non-zero eigen values, a feasible solution Z_p to the 0-1 SDP must be a rank one matrix. □

Lemma 3 *Any feasible solution to the second formulation of* 0-1 *SDP, $Z^{\mathtt{feas}}$ is a rank k matrix.*

Proof Lemma 1 states that $Z^{\mathtt{feas}}$ is a projection matrix satisfying $Z^2 = Z$. Hence, the eigen values of $Z^{\mathtt{feas}}$ are either 1 or 0. From Lemma 2, we know that $\|Z_p\|_F^2 = 1, \forall p \in [1, k]$. Also, from the construction of the SDP model, we get $Z^{\mathtt{feas}} = \sum_{p=1}^k Z_p$. Observe that since the clusters are disjoint, the sum of Z_p will result in no two non-zero element being added, i.e., the union of the set of non-zero elements of all Z_p matrices are also the non-zero elements of $Z^{\mathtt{feas}}$. Therefore, $\|Z^{\mathtt{feas}}\|_F^2 = \sum_{p=1}^k \|Z_p\|_F^2 = k$, which is the sum of eigen values of $Z^{\mathtt{feas}}$. The statement of the lemma follows. □

We restate a well known result in the following lemma.

Lemma 4 *Let A be a $m \times n$ matrix with eigen values values $s_i \ge s_2 \ge \dots \ge s_n$ and corresponding eigen vectors $\mathbf{v_1}, \mathbf{v_1}, \dots, \mathbf{v_n}$. If A has rank r, then for any $k < r$, a best rank k approximation of A is $A' = \sum_{i=1}^k \mathbf{v_i} s_i \mathbf{v_i^T}$.*

7.3 Rounding algorithm

Based on the above observations, the rounding scheme we adopt is roughly as follows.

```
Step 1  For the $Z$ obtained by solving the relaxed SDP, we find a rank $k$ approximation $Z^{\mathbf{k}}$ using Lemma (4).
Step 2  If $[\mathbf{v}^{\{1\}}, \mathbf{v}^{\{2\}}, \dots, \mathbf{v}^{\{k\}}]$ are the eigen vectors corresponding to the $k$ largest eigen value of $Z^{\mathbf{k}}$, then $X_p^{\text{frac}} = \mathbf{v}^{\{\mathbf{p}\}}$, $p \in [1, k]$.
Step 3  To obtain $X$ which is a solution to our problem we do the following. For each row of $X^{\text{frac}}$, we select its largest entry and set the corresponding $X$ position to 1 and all other positions in that row to 0.
```

Since the solution obtained by solving the relaxed SDP may not necessarily be a rank k matrix, we first find its closest rank k approximation in Step 1. But it still may not be a projection matrix, where all non zeros entries in any row are equal. Therefore, we perform the Singular Value decomposition to obtain column vectors X_p^{frac}. Finally, to obtain X, we round the largest entry of each row to 1 and all other entries to 0. Our rounding scheme proposed above follows a similar idea as in so-called spectral clustering (Xing and Jordan 2003; Ng et al. 2001). In these papers, PCA has been used to project the data into a lower dimensional space spanned by the eigen vectors of the k largest eigen values. This was followed by k-means clustering. In our model, we found that applying k-means sometimes leads to numerical instability. Rather, adopting the strategy mentioned above leads to more stable solutions.

Theorem 1 *The relaxed SDP model of* (42)–(43) *can be solved and rounded, both in polynomial time.*

Proof Polynomial time algorithms for solving SDP are known (Vandenberghe and Boyd 1996). In addition, we need to compute a rank k approximation of a $n \times n$ matrix in Sect. 7.3. This leads to a $O(kn^2)$ time for the rounding algorithm. □

8 Experiments

Our experiments included evaluations on several publicly available datasets, segmentation databases, face recognition and biomedical imaging data. We discuss these in the following sections.

8.1 UCI datasets

Our first set of experimental evaluation illustrates an application to several datasets from the UCI Machine Learning Repository http://mlearn.ics.uci.edu/MLSummary.html as follows.

1. Lung Cancer dataset: 2 classes, 57 attributes, 32 instances.
2. Diabetes dataset: 2 classes, 8 attributes, 768 instances.
3. Wine dataset: 3 classes, 13 attributes, 178 instances.
4. Iris dataset: 3 classes, 4 attributes, 150 instances.

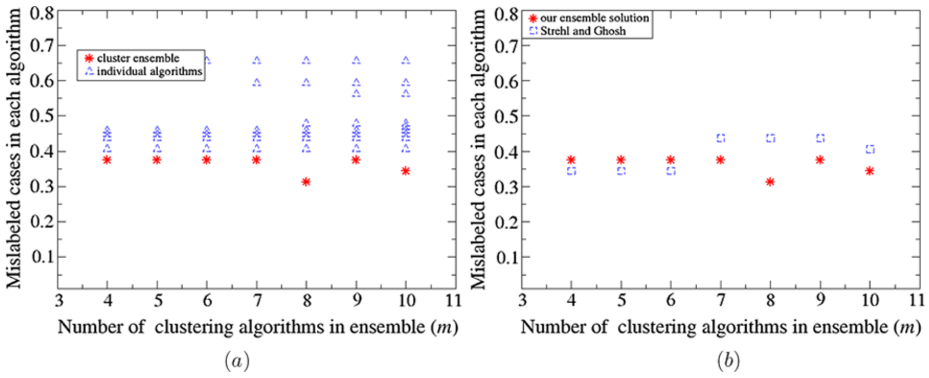


Fig. 2 Lung Cancer dataset. The fraction of mislabeled cases ([0, 1]) in our consensus solution ($*$) is compared to the number of mislabeled cases (Δ) in individual clustering algorithms in (**a**), a comparison of our solution to those obtained by Strehl and Ghosh on the same ensemble in (**b**)

The datasets also include the real class labels, we discard this information in the clustering process and use them only for evaluation. To create the ensemble, we used a set of $4, \ldots, 10$ clustering schemes (by varying the clustering criterion and/or algorithm) from the CLUTO clustering toolkit along with other standard clustering algorithms such as k-means, fuzzy cmeans, k-medoid, and max-margin clustering. The multiple solutions together comprised the cluster ensemble, our model was then used to determine a solution that maximized the agreement between these solutions. The same set of base clusterings were also used to as input to the algorithm of Strehl and Ghosh (2003). Then, the solutions from each scheme and the ensemble solution from our algorithm (Model 2) were compared with the truth. The accuracy results are shown in Figs. 2–5, calculated as the number of examples assigned to an incorrect cluster once cluster-to-cluster correspondence has been established between the given ground truth and the ensemble solution (in other words, the 0-1 loss in the unsupervised setting). In Fig. 2, we show the results on the Lung Cancer dataset, in Fig. 2(a), we see that the ensemble solution is better than the best solution in the ensemble in all cases. In Fig. 2(b), we compare the performance of our ensemble solution with those obtained by running the algorithm by Strehl and Ghosh (2003) (referred to as SG).[2] We see that the algorithms compare favorably for smaller m values, but for $m = \{7, 8, 9, 10\}$ the solution from our model is better.

In Fig. 3, the solution from our model is not only better than all individual clustering algorithms but is also superior to Strehl and Ghosh (2003) in all cases. We see essentially the same behavior for the Wine dataset in Fig. 4. An ensemble solution is useful in such cases because we do not know a priori (in the absence of ground truth) that which algorithm will perform the best. Finally in Fig. 5, we see the performance on the Iris dataset. While the ensemble solution is always better than the input solutions, we notice that Strehl and Ghosh (2003) reports a solution with a slightly lower misclassification error than our solution in some cases. We attribute this to errors that were introduced in the rounding phase.

We also compare our approach with the method outlined by Topchy et al. (2003) for each dataset from the UCI dataset. For fairness of comparison, we use the same set of base clustering as input for both methods. In addition, for Topchy et al. (2003), the 2D A strings created

[2]Note that SG runs the IBGF and MCLA formulations and outputs a *single* solution (determined by a measure of the goodness of the solution); here we evaluate *both* our models individually to the output solution of SG.

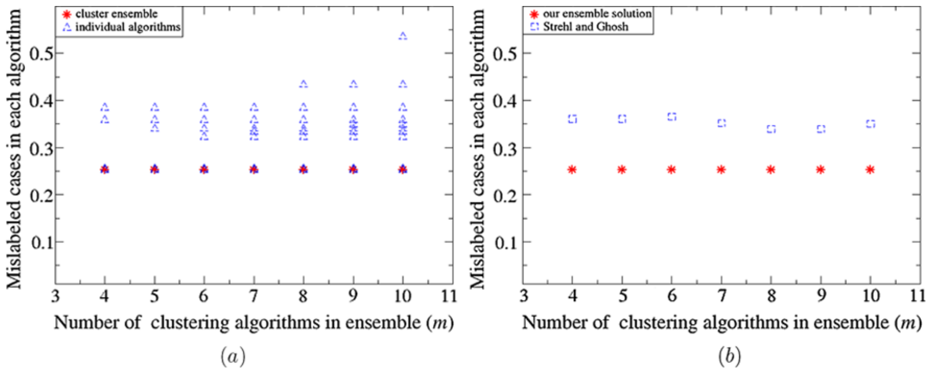


Fig. 3 Diabetes dataset. The fraction of mislabeled cases ([0, 1]) in our consensus solution ($*$) is compared to the number of mislabeled cases (Δ) in individual clustering algorithms in (**a**), a comparison of our solution to those obtained by Strehl and Ghosh on the same ensemble in (**b**)

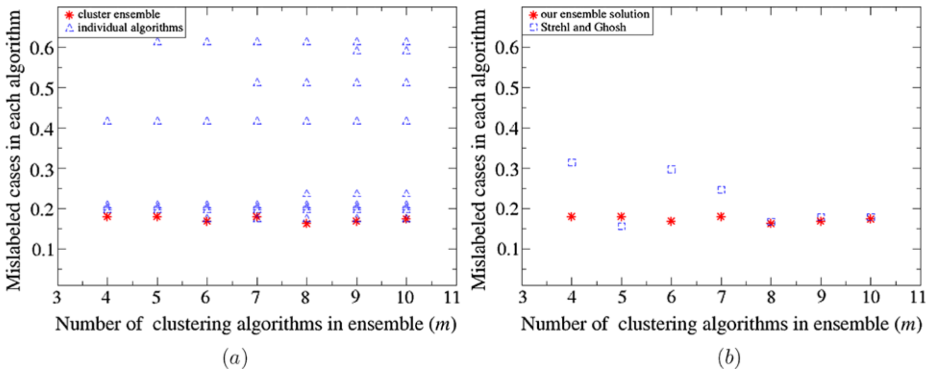


Fig. 4 Wine dataset. The fraction of mislabeled cases ([0, 1]) in our consensus solution ($*$) is compared to the number of mislabeled cases (Δ) in individual clustering algorithms in (**a**), a comparison of our solution to those obtained by Strehl and Ghosh on the same ensemble in (**b**)

Table 1 Normalized mutual information values (and standard deviations) for our approach and that of Strehl and Ghosh (2003) for an input of 10 clusterings on each UCI data

Dataset	Ours	Topchy et al. (2003)	SG (Strehl and Ghosh 2003)
Lung Cancer	0.56 (0.015)	0.53 (0.02)	0.55 (0.018)
Diabetes	0.27 (0.02)	0.25 (0.05)	0.23 (0.01)
Wine	0.59 (0.03)	0.59 (0.02)	0.41 (0.03)
Iris	0.74 (0.004)	0.73 (0.03)	0.74 (0.004)

were converted into a vector representation and then clustered using k-means clustering. The results of these comparison is shown in Table 1. Here, we see the average misclassification errors for each of the 5 datasets mentioned above. Our approach performs better in all cases than Topchy et al. (2003).

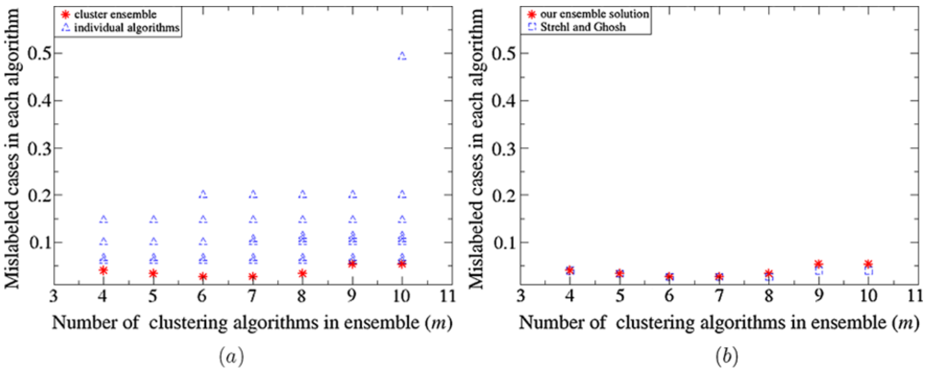


Fig. 5 Iris dataset. The fraction of mislabeled cases ([0, 1]) in our consensus solution ($*$) is compared to the number of mislabeled cases (Δ) in individual clustering algorithms in (**a**), a comparison of our solution to those obtained by Strehl and Ghosh on the same ensemble in (**b**)

Table 2 Mean of misclassification errors in % (with standard deviation) of our approach compared to Topchy et al. (2003) and Strehl and Ghosh (2003)

Dataset	Ours	Topchy et al. (2003)	SG (Strehl and Ghosh 2003)
Lung Cancer	36.16 (2.45)	44.37 (4.92)	39.28 (4.72)
Diabetes	25.26 (0.025)	40.90 (1.46)	35.20 (1.03)
Wine	17.33 (0.06)	19.00 (6.44)	18.20 (6.52)
Iris	3.80 (1.1)	3.90 (1.31)	3.30 (0.66)

8.2 Application to image segmentation

Though the use of ensembling in conventional clustering has been investigated by several researchers in the recent past, methodical ensemble methods in context of improving image segmentation is still a relatively unexplored area; our second set of evaluations focuses on this novel application. The motivation stems from the fact that even sophisticated segmentation algorithms may yield 'different' results on the same image—one may capture some features of the image better than the others. The task of obtaining a good segmentation becomes particularly challenging because we do not know *a priori* (without visual analysis) whether a given method will yield good results. When multiple segmentations are available, it seems reasonable to 'combine' segmentations in a methodical manner in an effort to reduce degeneracies. The concept of cluster ensemble seems appropriate for this purpose. Note that for the small fraction of images where all segmentations are 'good', the ensemble cannot introduce degeneracies and is as good as the input set. The interesting (and majority) case is when most algorithms do not perform very well, when details in the image may be selectively recognized by a subgroup of the algorithms and missed by others. Our experimental results indicate that in almost all cases, we can obtain a better overall segmentation that captures (more) details in the images more accurately with fewer outlying clusters.

We note that the few prior applications of ensemble approaches to the segmentation domain can mostly be found in medical imaging problems; examples include atlas based image segmentation approaches for segmenting microscopic brain images (Rohlfing and Maurer 2005), shape based averaging (Rohlfing and Maurer 2005) and segmentation of mammograms (de Silva et al. 2000). We note that a number of recent medical imaging papers have used the STAPLE algorithm (Warfield et al. 2004) for combining and estimating the performance of segmentations. These results are not intended to show that the proposed algorithm is the best available means for combining segmentations (which will clearly require additional analysis), but to demonstrate the applicability of the algorithm to this important problem.

In this following sections, we provide a framework for applying our approach for generating an ensemble given a few input segmentation results. One may suspect that since images typically have large number of pixels, application of cluster ensembling approach to all pixels in the image may be computationally time consuming. However, we observe that the segmentation ensembles need not be generated for all pixels in image. This does not refer to any sampling, in most cases we generate an equivalent solution as discussed below.

8.2.1 *Preprocessing or sub-sampling methods*

Let segmentation algorithms $\alpha_1, \alpha_2, \ldots, \alpha_m$ yield m different segmentations with $l_1, l_2, \ldots, l_m$ labeled regions (resp.). Let C be the number of classes desired in the ensemble segmentation and each labeled region in the segmentations produced by α_j be identified by $\{\alpha_{1j}, \alpha_{2j}, \ldots, \alpha_{l_j j}\}$. Let $f_j(p)$, $j = 1, \ldots, m$ be the class assigned to pixel p in segmentation α_j. We first divide the pixels of the image into non-overlapping subsets $\{S_1, S_2, \ldots, S_n\}$ such that each pixel in particular subset is assigned the same label by all segmentation algorithms i.e., for any $p, q \in S_i$, $f_j(p) = f_j(q)$, $j = 1, \ldots, m$. Notice that that since there is no *disagreement* among pixels in any S_i, it is unnecessary to ensemble each pixel in this group individually. Instead we can create a representative data item (super-pixel) denoted as p_{S_i}, such that the super-pixel is assigned the label of the group for every segmentation algorithm i.e., $f_j(p_{S_i}) = f_j(q)\ \forall j \in [1, m]$, $\forall q \in S_i$. This creates a new input set for the segmentation ensemble problem which consists of $P_S = \{p_{S_1}, p_{S_2}, \ldots, p_{S_n}\}$. The size of this set is at least $\max(l_1, l_2, \ldots, l_m)$ and at most the number of pixels in the image, though the latter case is very unlikely. Once the ensemble is generated for P_S, the class of each p_{S_i} is also assigned the class for each pixel in the set S_i.

8.2.2 *Experimental results*

We evaluated our method of ensemble segmentation using images from Berkeley Segmentation Database (The Berkeley Segmentation Database and Benchmark 2009). The image segmentations were generated using several powerful and popular algorithms—(1) Normalized Cuts (Shi and Malik 2000), (2) Graph Cuts (Boykov et al. 2001), (3) Curve Evolution (Curve Evolution Segmentation Dataset 2009) and (4) Graph-based segmentation (Felzenszwalb and Huttenlocher 2004). In Fig. 6, we illustrate the results on the images using the above algorithms. Notice that each algorithm performs well but misses out on some details in Fig. 6 (top row). For instance, (a) and (b) divide the stem of the mushroom into two parts, (c) induces an additional cut in the background between the two shrubs, and (d) misses the left boundary of the mushroom stem completely. The ensemble (right-most) on the other hand is able to segment these details nicely by combining (a)–(d).

The example in Fig. 6 (bottom row) shows a case where the segmentation algorithms (Shi and Malik 2000; Boykov et al. 2001; Felzenszwalb and Huttenlocher 2004) generate reasonable segmentations. Nonetheless, we can see that (a) misses the lower contour of the cap and the lip region, (b) misses the inner boundaries of the right hand, (c) induces an additional cut across the face while (d) incorrectly detects the lower boundaries of the cap. The ensemble on the other hand is able to nicely aggregate the details from the individual segmentations.

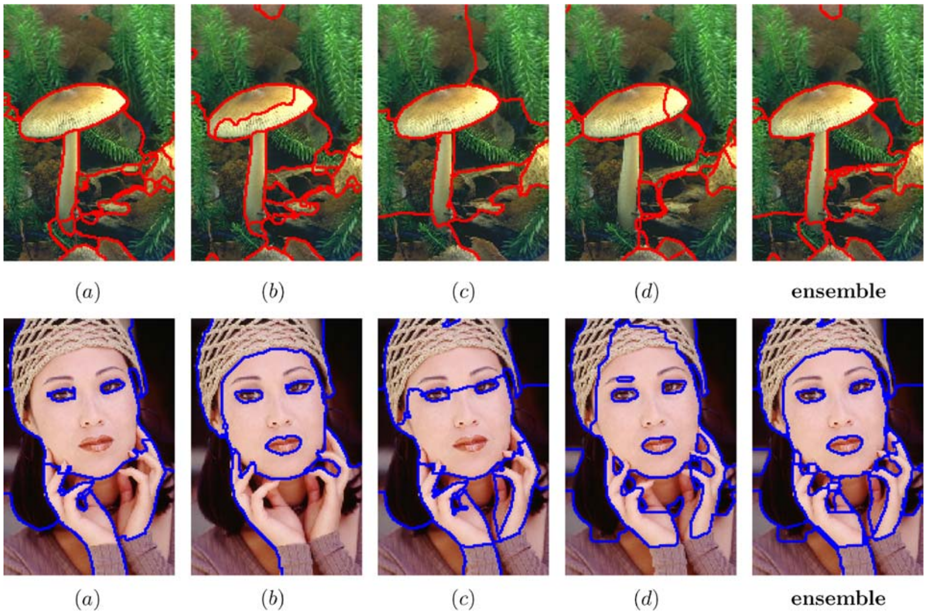


Fig. 6 A segmentation ensemble on images from the Berkeley Segmentation dataset. Segmentation methods used were (3) with different parameter settings, see Sect. 8.2.2 for details

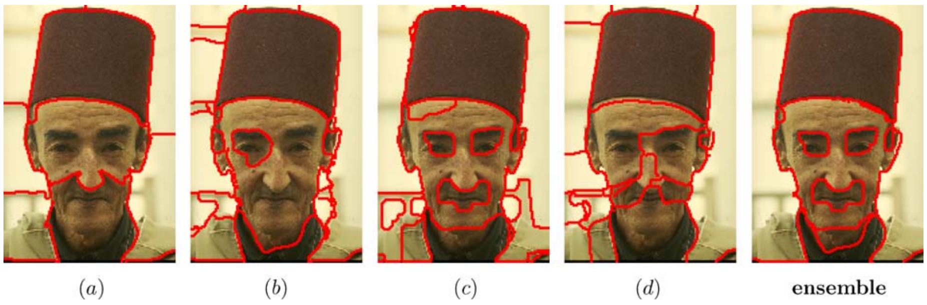


Fig. 7 A segmentation ensemble on an image from the Berkeley Segmentation dataset. (**a**)–(**d**) show the individual segmentations overlaid on the input image, the *right-most* image shows the segmentation generated from ensemble clustering. Segmentation methods used for (**a**)–(**d**) were (1), (3) with two parameter settings, and (4), see Sect. 8.2.2 for details

In Fig. 7, we illustrate the results on another example from Berkeley dataset. Again, each algorithm performs well but misses out on some details. For instance, (a) and (d) do not segment the eyes; (b) does well in segmenting the shirt collar region but can only recognize one of the eyes and (c) distinguishes both eyes but creates an additional cut across the forehead. The ensemble (extreme right) on the other hand is able to segment these details (eyes, shirt collar and cap) nicely by combining (a)–(d).

8.3 Application to face recognition

In addition to combining clustering results of multiple clustering algorithms, ensemble clustering can be also be used to aggregate the solutions of a single clustering algorithm executed

Fig. 8 Sample face images from ORL Face recognition dataset

multiple times using a subset of features (for each run). A nice example of this is in context of face recognition applications. It is well known that images of faces are intrinsically low dimensional data living in very high dimensional space, hence, we may expect that a large number of features can be omitted without a proportional loss of information content. A number of popular face recognition algorithms try to exploit this property by projecting the data into a lower dimensional space using techniques such as PCA (Perlibakas 2004) and ICA (Liu and Wechsler 2003), see Turk and Pentland (1991). A relatively less explored dimensionality reduction approach for face recognition in particular is the Random Projection (RP) method based on the Johnson-Lindenstrauss lemma, see Vempala (2004), Johnson and Lindenstrauss (1984). Recently, some authors have reported on experiments using RP for face detection (Goel et al. 2005). However, as noted by Fern and Brodley (2003), the main disadvantage with Random Projections in practice is that "they are not very stable", different runs on the same data may yield different solutions. To improve the empirical performance, they proposed the idea of ensembling multiple solutions of RP using Strehl and Ghosh (2003) on datasets having dimensions of at most 180. In this section, we discuss how our ensemble algorithm can be used to perform unsupervised clustering on face data sets using Random Projections.

8.3.1 Experimental results

We used the ORL datasets for our experiments. The dataset comprises of 400 images of 40 subjects (10 images per subject). The images reflect variations in pose, expression, illumination and scale, see Fig. 8 for an illustration. The image sizes were 32×32, these were rescaled to yield a $1024D$ feature vector for each image. For our evaluations, we selected a pair of faces for each execution as in Xu et al. (2005). The feature vectors were projected onto $3D$ using RP and then clustering was performed using max-margin clustering algorithm (Peng et al. 2009). This gives m clustering outputs for each pair, and the results were ensembled using our approach. The process was repeated 780 times (all pairs) for $m = \{6, 8, 10, 12, 14, 16\}$.

Our goal in this set of experiments is to propose the usefulness of an ensembling approach for face recognition, we will not comprehensively evaluate the efficacy of RP or max-margin algorithm for face recognition applications. Recall that in the previous set of experiments on several UCI data sets in Sect. 8.1, the ensemble solutions in most cases were at least as good as the best among the input solutions. However, in the present setup (i.e., face recognition), we see that this may not always be true. Part of the reason is the degree of randomness in the projections, which leads to input solutions where one or more clusterings may be arbitrarily worse with as much as 50% misclassification error. Such solutions may influence the ensemble. However, we see from the results in Fig. 9(a), that in more than 90% of the cases, the ensemble performs better than the mean of the input errors. This progressively gets better with an increase in the number of solutions in the input ensemble. In Fig. 9(b), we see the comparison of the ensemble solution to the input solutions ordered w.r.t. misclassification error. The first set of bar plots under the label, $X < A_1$ show that in 60% of cases, the ensemble performs at least as good as the best among the input solutions. This

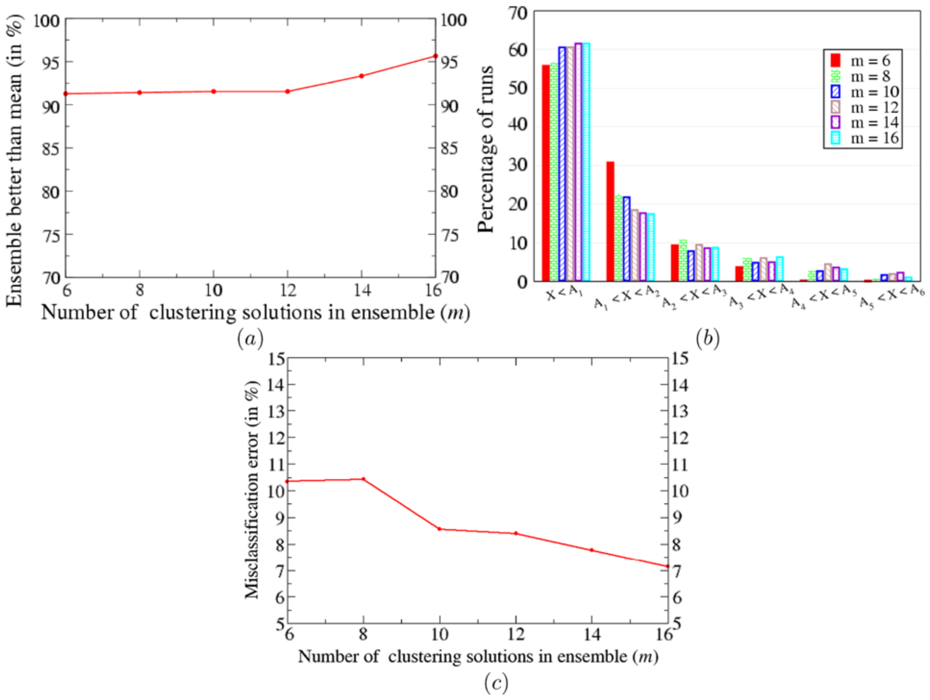


Fig. 9 Performance evaluations on ORL Face Recognition dataset. Percentage of instances where misclassification error of the computed ensemble is better than (or equal to) the mean of misclassification errors of input solutions in (**a**); in (**b**), percentage of instances where the misclassification error of the computed ensemble (X) lies between the errors of A_i and A_{i+1} where A_1 is the lowest error and $A = \{A_1, \ldots, A_m\}$ is the ordering of solutions w.r.t. errors. (**c**) Plot of the average of misclassification errors of the computed ensemble as a function of the number of solutions (m) in the input

behavior also depends on the number of solutions (m) in the input. The performance plots also show a strong concentration in the first three classes, indicating that the ensemble rarely performs worse than the third-best solution. Finally, we show the average misclassification error for the ensemble solution as a function of m for the ORL dataset. We see that the performance improves almost linearly with an increase in the number of input solutions. For $m = 16$, we obtain 7% misclassification error for the ORL dataset suggesting that this may be a viable approach for doing unsupervised face recognition.

8.4 Application to combination of Diffusion Tensor Image segmentations

Diffusion Tensor Imaging (DTI) technique allows the measurement of diffusion of water molecules in human tissues. DTI images are becoming increasingly popular within neuroimaging (and other medical imaging areas) because they may be useful to infer the underlying structure and organizational pattern in the body (e.g., neuronal pathways in the brain). In such images, each pixel (or voxel) is given as a diffusion tensor, D_i located at each voxel i. Here, D_i is a symmetric positive semidefinite matrix of size 3×3, and characterizes the diffusivity in a certain direction. To simplify the processing as well as clinical interpretation of such images, a number of different measures (or channels) are calculated from the diffusion tensor field (image). This typically includes measures such as Apparent Diffusion Coefficient (ADC), Fractional Anisotropy (FA), and Mean Diffusivity (MD).

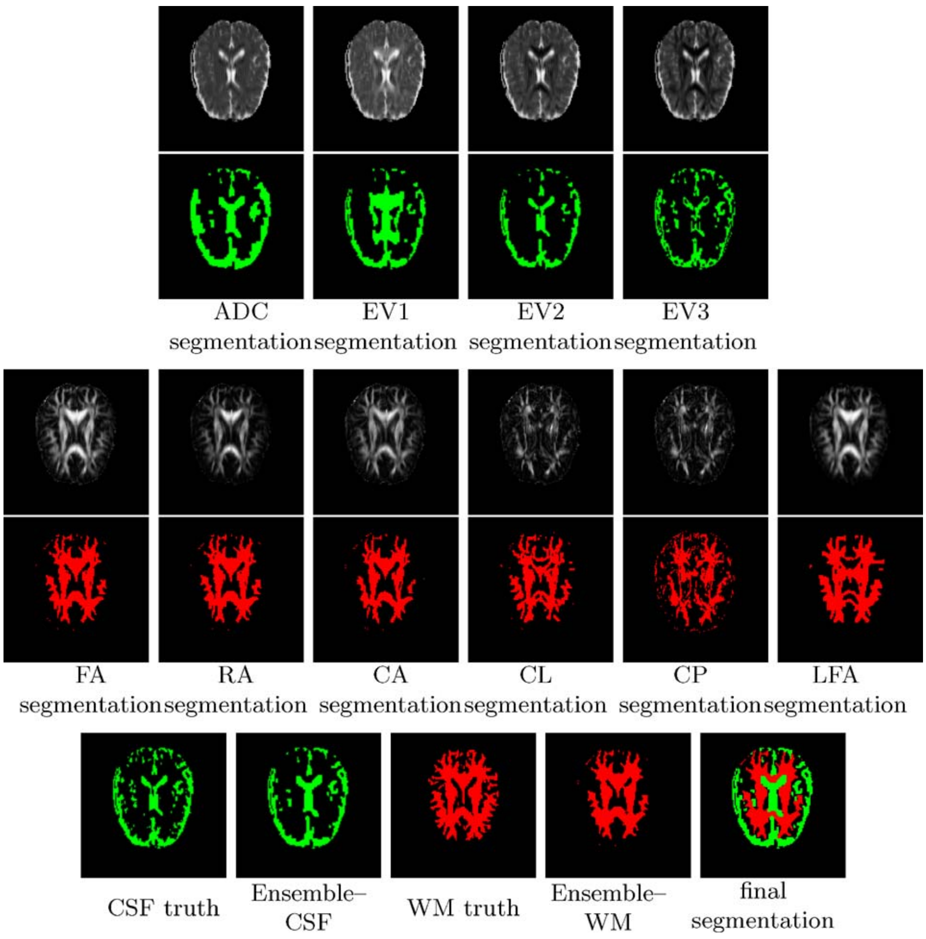


Fig. 10 Ensemble clustering for DTI segmentation. Rows 1–2 show the four channels of the DT image (for WM/non-WM) and the segmentations. Rows 3–4 show six channels of the DT image (for CSF/non-CSF) and the segmentations. Row 5 shows the ground truth and ensemble solution for row 1–2 images, the ground truth and ensemble solution for row 3–4 images, and finally the ensemble segmentation (where the CSF/non-CSF and WM/non-WM segmentation are overlaid)

Since the contrast offered by each channel may vary spatially, a standard strategy is to perform segmentation of each channel separately and then combine the results using a voting based strategy (Liu et al. 2007). In this section, we present our experiments to demonstrate that ensemble clustering is a systematic method for combining such segmentations.

We performed our experiments on four DTI brain images with ten channels per image. Six of these channels are (1) Fractional anisotropy (FA), (2) Relative Anisotropy (RA), (3) Anisotropy index (CA), (4) Linear anisotropy (CL), (5) Planar anisotropy (CP), and (6) Fractional anisotropy (LFA). These are used to distinguish white matter (WM) pixels from non-WM regions. The remaining four channels are (7) Apparent Diffusion coefficient (ADC), and (8–10) the three eigen values of the diffusion tensor (EV1, EV2, EV3). These channels are used to separate CSF and non-CSF regions. The discrimination power offered by each channel to identify the regions of interest varies from one image to the next. Fig-

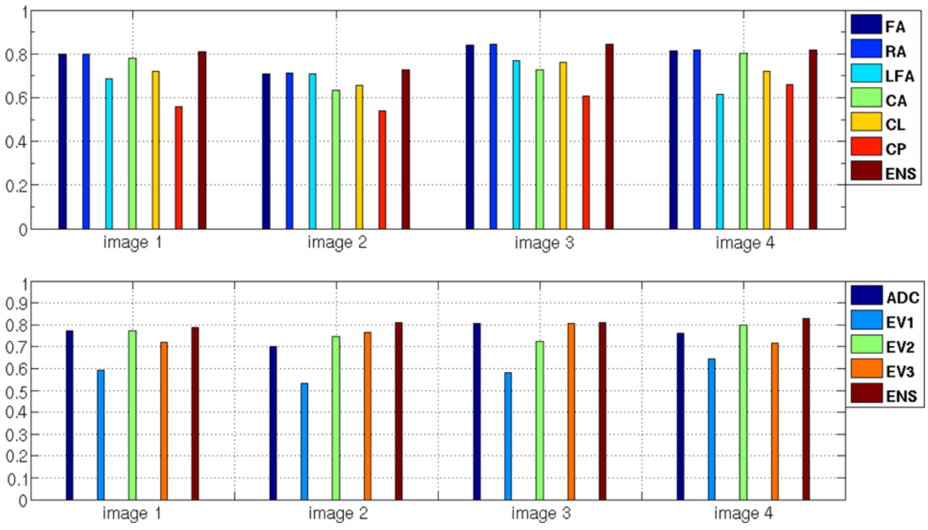


Fig. 11 Performance (in terms of volume overlap) of ensemble clustering on the y-axis for combining segmentations of different DTI channels for WM/non-WM regions (*top*) and CSF/non-CSF regions (*bottom*). The ensemble solution is denoted as ENS

ure 11 shows the foreground volume overlaps of the segmentation results for each such channel with ground truth (manual segmentation by an expert). Since the the proportion of foreground pixels (region of interest) is less than the background, the misclassification errors are typically quite small. Therefore, the volume overlap gives a more accurate description of the performance of the algorithm. In Fig. 10, we show a representative segmentation obtained using ensemble clustering. In general, one channel is never consistently better than the other channels. As a result, by combining the individual segmentations we are able to obtain a consistent and reliable segmentation of the CSF/non-CSF regions (rows 1–2, and last row), and WM and non-WM regions (rows 3–4, and last row). We see that the volume overlap of the ensemble solution is the same as (or better than) the set of segmentations on the channels individually. Our results on all four images are summarized in Fig. 11. We note that the results from Strehl and Ghosh (2003) on this data were nearly identical to those shown in Fig. 10.

Limitations Before concluding the paper, we point out some of the limitations of the algorithm. A practical limitation of the algorithm is that it is tied to the speed of the underlying SDP solver (e.g., Sedumi (Sturm 1999), SDPT3 (Toh et al. 1999)). For example, generating the segmentation ensemble shown in Fig. 6 takes up to a minute on a modern workstation. However, we note that improvements in methods and software for SDP problems will translate into running time improvements for solving the proposed SDP model. For large scale problems, a number of first order methods are also available, see http://plato.asu.edu/dimacs/.

9 Conclusions

In this paper, we have proposed an efficient SDP based model for ensemble clustering. Our contributions include a mechanism to represent dissent among input clusters as a 2D string, which is then formulated as a 0-1 Semidefinite program, and further relaxed to a polynomial time solvable SDP. We show several promising experimental results which highlight different aspects of this approach and also illustrate novel applications such as segmentation ensembles and biomedical image segmentation. The challenges to applying such an approach for images lie in the fact that images are generally large in size and also that the data points or pixels are spatially co-related, a property which cannot be directly incorporated into existing ensemble frameworks. We show that our r-pixels approach mentioned in Sect. 8.2 is useful to reduce the size of the problem without affecting the ensemble quality. In addition, the spatial properties of the pixels can be modified to be applied at the r-pixel level, rather than the pixel level. We believe these problems are of independent research interest and needs to be explored further. The implementation of the algorithm is available for download from http://www.biostat.wisc.edu/~vsingh.

Acknowledgements This work was supported in part by NIH R21AG034315, NIH CTSA award 1UL1RR025011 and NSF grants IIS-0713489, DMS0915240 and CCF-0546509. We thank Andy Alexandar for providing us with the DTI images and useful discussions regarding DTI segmentation and Johan Löfberg for a thorough explanation of the salient features of Yalmip (Löfberg 2004), and the reviewers for suggestions regarding the presentation of the paper.

References

Ailon, N., Charikar, M., & Newman, A. (2005). Aggregating inconsistent information: ranking and clustering. In *Proc. of symposium on theory of computing* (pp. 684–693).

Anjos, M., & Wolkowicz, H. (1999). *A strengthened sdp relaxation via a second lifting for the max-cut problem* (Technical Report).

Bansal, N., Blum, A., & Chawla, S. (2002). Correlation clustering. In *Proc. symposium on foundations of computer science* (p. 238).

Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(11), 1222–1239. http://www.csd.uwo.ca/~olga/code.html.

Charikar, M., Guruswami, V., & Wirth, A. (2005). Clustering with qualitative information. *Journal of Computer and System Science*, *71*(3), 360–383.

Curve Evolution Segmentation Dataset (2009). http://barissumengen.com/seg/bsds_segmentations/html_for_visualization/curveevolution_testset.html.

de Silva, C. J., Masek, M., & Attikiouzel, Y. (2000). Combining data from different algorithms to segment the skin-airinterface in mammograms. In *IEEE engineering in medicine and biology society* (Vol. 2, pp. 1195–1198).

Dudoit, S., & Fridlyand, J. (2003). Bagging to improve the accuracy of a clustering procedure. *Bioinformatics, 19*(9).

Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, *59*(2), 167–181.

Fern, X. Z., & Brodley, C. E. (2003). Random projection for high dimensional data clustering: a cluster ensemble approach. In *Proc. of ieee international conference on machine learning* (pp. 186–193).

Fern, X. Z., & Brodley, C. E. (2004). Solving cluster ensemble problems by bipartite graph partitioning. In *Proc. of international conference on machine learning* (p. 36).

Filkov, V., & Skiena, S. (2003). Integrating microarray data by consensus clustering. In *Proc. of international conference on tools with artificial intelligence* (p. 418).

Fred, A. L. N., & Jain, A. K. (2006). Combining multiple clusterings using evidence accumulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(6).

Gasieniec, L., Jansson, J., & Lingas, A. (2000). Approximation algorithms for hamming clustering problems. In *Proc. of symposium on combinatorial pattern matching* (pp. 108–118).

Gionis, A., Mannila, H., & Tsaparas, P. (2005). Clustering aggregation. In *Proc. of international conference on data engineering* (pp. 341–352).

Giotis, I., & Guruswami, V. (2006). Correlation clustering with a fixed number of clusters. In *Proc. of symposium on discrete algorithms* (pp. 1167–1176).

Goel, N., Bebis, G., & Nefian, A. (2005). Face recognition experiments with random projection. In *Proc. SPIE* (Vol. 5779, pp. 426–437).

Goemans, M. X., & Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, *42*, 1115–1145.

Gordon, A. D., & Henderson, J. T. (1977). An algorithm for euclidean sum of squares classification. *Biometrics*, *33*, 355–362.

Johnson, W. B., & Lindenstrauss, J. (1984). Extensions of lipshitz mapping into hilbert space. *Contemporary Mathematics*, *26*, 189–206.

Kuncheva, L. I., & Vetrov, D. P. (2006). Evaluation of stability of k-means cluster ensembles with respect to random initialization. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 28*(11).

Lasserre, J. B. (2001). An explicit exact sdp relaxation for nonlinear 0-1 programs. In *Proceedings of the 8th international IPCO conference on integer programming and combinatorial optimization* (pp. 293–303).

Liu, C., & Wechsler, H. (2003). Independent component analysis of gabor features for face recognition. *IEEE Transactions on Neural Networks*, *14*(4), 919–928.

Liu, T., Li, H., Wong, K., Tarokh, A., Guo, L., & Wong, S. T. C. (2007). Brain tissue segmentation based on DTI data. *NeuroImage*, *38*(1), 114–123.

Löfberg, J. (2004). YALMIP: A toolbox for modeling and optimization in MATLAB. In *CCA/ISIC/CACSD*, September 2004.

Monti, S., Tamayo, P., Mesirov, J., & Golub, T. (2003). Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, *52*(1–2), 91–118.

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). On spectral clustering: analysis and an algorithm. In *Advances in neural information processing systems* (pp. 849–856).

Peng, J., & Wei, Y. (2007). Approximating k-means-type clustering via semidefinite programming. *SIAM Journal on Optimization*, *18*(1), 186–205.

Peng, J., Mukherjee, L., Singh, V., Schuurmans, D., & Xu, L. (2009). An efficient algorithm for maximal margin clustering. Manuscript (under peer review).

Perlibakas, V. (2004). Distance measures for PCA-based face recognition. *Pattern Recognition Letters*, *25*(6), 711–724.

Rohlfing, T., & Maurer, C. R. Jr. (2005). Shape-based averaging for combination of multiple segmentations. In *Medical image computing and computer-assisted intervention (MICCAI)* (pp. 838–845).

Rohlfing, T., & Maurer, C. R. Jr. (2005). Multi-classifier framework for atlas-based image segmentation. *Pattern Recognition Letters*, *26*(13), 2070–2079.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 888–905.

Singh, V., Mukherjee, L., Peng, J., & Xu, J. (2007). Ensemble clustering using semidefinite programming. In *Advances in neural information processing systems*.

Strehl, A., & Ghosh, J. (2003). Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, *3*, 583–617.

Sturm, J. F. (1999). Using SeDuMi 1.02, A Matlab Toolbox for Optimization over Symmetric Cones. *Optimization Methods and Software*, *11–12*, 625–653.

The Berkeley Segmentation Database and Benchmark. (2009). Computer Science Department, UC-Berkeley.

Toh, K. C., Todd, M. J., & Tutuncu, R. (1999). SDPT3—a Matlab software package for semidefinite programming. *Optimization Methods and Software*, *11*(12), 545–581.

Topchy, A., Jain, A. K., & Punch, W. (2003). A mixture model for clustering ensembles. In *Proc. of SIAM conference on data mining*.

Topchy, A., Jain, A. K., & Punch, W. (2003). Combining multiple weak clusterings. In *Proc. of IEEE international conference on data mining*.

Turk, M., & Pentland, A. (1991). Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, *3*(1), 71–86.

Vandenberghe, L., & Boyd, S. (1996). Semidefinite programming. *SIAM Review*, *38*, 49–95.

Vempala, S. S. (2004). *The random projection method. DIMACS series in discrete mathematics and theoretical computer science* (Vol. 65). Providence: Am. Math. Soc.

Warfield, S. K., Zou, K. H., & Wells, W. M. (2004). Simultaneous truth and performance level estimation (STAPLE): an algorithm for the validation of image segmentation. *IEEE Transactions on Medical Imaging*, *23*(7), 903–921.

Xing, E. P., & Jordan, M. I. (2003). *On semidefinite relaxation for normalized k-cut and connections to spectral clustering* (Technical Report UCB/CSD-03-1265). EECS Department, University of California Berkeley.

Xu, L., Neufeld, J., Larson, B., & Schuurmans, D. (2005). Maximum margin clustering. In *Advances in neural information processing systems* Cambridge: MIT Press.