Noname manuscript No.
(will be inserted by the editor)

How to "Alternatize" a Clustering Algorithm

M. Shahriar Hossain · Naren Ramakrishnan
· Ian Davidson · Layne T. Watson

the date of receipt and acceptance should be inserted later

Abstract Given a clustering algorithm, how can we adapt it to find multiple, nonredundant, high-quality clusterings? We focus on algorithms based on vector quantization and describe a framework for automatic 'alternatization' of such algorithms. Our framework works in both simultaneous and sequential learning formulations and can mine an arbitrary number of alternative clusterings. We demonstrate its applicability to various clustering algorithms—k-means, spectral clustering, constrained clustering, and co-clustering—and effectiveness in mining a variety of datasets.

Keywords Clustering · Alternative clustering

1 Introduction

Alternative clustering (e.g., Gondek and Hofmann (2007)) is the idea of uncovering multiple clusterings of a dataset so as to suggest varying viewpoints and differing hypotheses. It has been studied in various applications, e.g., to help refine functional classifications of genes (Sinkkonen and Kaski (2002)) and in multicriteria decision making (Malakooti and Yang (2004); Miettinen and Salminen (1999)). Alternative clustering is also typically considered a precursor step to consensus clustering (Li et al. (2007); Monti et al. (2003)).

While it has been long accepted that clustering formulations are generally underconstrained and hence afford multiple solutions, the idea of explicitly mining alternative clusterings has witnessed a recent surge of interest (Bae and Bailey (2006); Caruana et al. (2006); Cui et al. (2007); Dang and Bailey (2010a,b); Davidson and Qi (2008); Gondek and Hofmann (2005, 2007); Gondek et al. (2005); Jain et al. (2008); Niu et al. (2010); Qi and Davidson (2009); Ross and Zemel (2006); Zhang et al. (2009)).

Both sequential and simultaneous learning formulations have been studied. In the sequential formulation, we are given a clustering or set of clusterings, and the goal is to identify a new high-quality clustering that is as different as possible from the supplied cluster-

M. S. Hossain[1], N. Ramakrishnan[1], L. T. Watson[1,2]
[1]Dept. of Computer Science, [2]Dept. of Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061

I. Davidson
Dept. of Computer Science, University of California, Davis, CA 95616

ing(s). In the simultaneous learning formulation, also known as disparate clustering (Jain et al. (2008)), the goal is to simultaneously identify two (or more) different high-quality clusterings.

Algorithms for mining alternative clusterings approach the underlying problem in different ways. Davidson and Qi (2008) propose a constrained optimization formulation to transform the underlying instance space where the results of the previous clustering are used as constraints. Jain et al. (2008) learn two disparate clusterings simultaneously by minimizing a k-means sum-of-squares error objective for the two clustering solutions and at the same time minimizing the correlation between these two clusterings. Cui et al. (2007) find many alternative clusterings using a series of orthogonal projections. Data is repetitively orthogonalized into a space not covered by existing clusterings and a clustering algorithm is applied on the new space. Dang and Bailey (2010a) propose an information-theoretic approach to ensure alternative clustering quality by minimizing the mutual information between the desired clustering and a supplied clustering. Niu et al. (2010) describe an approach that is based on learning multiple subspaces in conjunction with learning multiple alternative clustering solutions by optimizing a single objective function.

There are thus 'alternate' views of alternative clustering. Our goal here is not to present yet another alternative clustering algorithm, but a formulation where we can take an existing algorithm and automatically 'alternatize' it. In other words, given a clustering algorithm, we show how we can automatically adapt it to find alternative clusterings.

Our contributions are:

1. We demonstrate how vector quantization algorithms that optimize for prototypes can be embedded into a larger contingency table framework to identify alternative clusterings. We show how this alternatization approach works for k-means, spectral clustering (Shi and Malik (2000)), co-clustering of bipartite graphs (Dhillon (2001)), and constraint-based clustering formulations (Wang and Davidson (2010)).
2. We are able to find many alternative clusterings, rather than just two alternative clusterings or one clustering alternative to a given clustering. Since there is an intrinsic limitation to mining multiple alternative high-quality clusters, our approach helps explore the space of possible clusterings in a systematic manner. We show how this is a valuable tool in exploratory data analysis.
3. Our approach works in both simultaneous and sequential learning formulations. In our experiments here, we demonstrate the use of our simultaneous formulation to first find two alternative clusterings and then use the sequential paradigm to incrementally find more alternative clusterings.

This paper significantly builds upon a preliminary conference version by Hossain et al. (2010). While the underlying optimization framework is the same, we have generalized both the problem formulation and the domains of applicability. First, the work in Hossain et al. (2010) is not focused on mining alternative clusterings and instead aims to mine clusters with either maximum similarity or maximum dissimilarity when compared through a supplied relation. In contrast, our work here is aimed at mining one or more alternative clusterings. Second, the work in Hossain et al. (2010) is only for k-means algorithms whereas the present paper shows how most algorithms based on vector quantization—i.e., those that choose prototypes/codebook vectors to minimize distortion when the data are replaced by the prototypes—can be alternatized using our approach. As is well known, this covers a broad range of clustering algorithms. Third, the work in Hossain et al. (2010) is focused on a specific data type (attribute vectors over two domains connected by a relation) whereas the

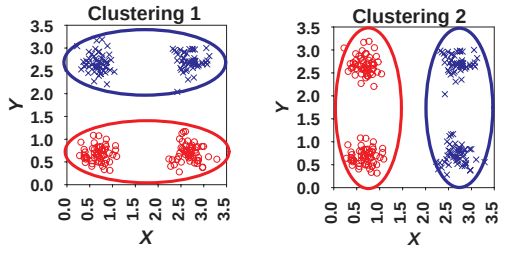
work here is broad and encompasses many data types, viz. attribute vectors in one or multiple domains, one-mode similarity graphs, bipartite, and multipartite graphs. As a result we argue that the presented alternatization framework can form a building block for creating complex data mining algorithms.

2 Alternatization

To introduce the basic ideas behind our alternatization framework, we consider a small synthetic 2D example (Fig. 1) involving 200 points where we seek to mine two clusters. In vector quantization algorithms, each cluster is condensed to a prototype and because we desire alternative clusterings, we wish to identify two sets of prototypes—Proto1 and Proto2—each of which has one vector for each cluster. There are two desired properties for these clusterings: i) when compared across clusterings the clusters must be highly distinct from each other, ii) the individual clusters in each clustering must be local in the respective spaces (i.e., points within a cluster are similar whereas points across clusters are dissimilar).

2.1 Modeling dissimilarity

We model overlap between clusterings by constructing a contingency table, as shown in Fig. 1 (bottom). The table is 2×2, where the rows denote clusters from Fig. 1 (top left) and the columns denote clusters from Fig. 1 (top right). The cells indicate the number of data points that are common among the respective clusters. In an ideal example of alternative clustering, the contingency table would result in a uniform (or near uniform) distribution over all contingency table entries because each cluster from one clustering is uniformly distributed over all the clusters of the second clustering. Each cluster of Clusterings 1 and 2 of Fig. 1 has 100 points. If we take all 100 points of a cluster of Clustering 1, we would find out that 50 of these points belong to one cluster of Clustering 2 and the other 50 points belong to the other cluster of Clustering 2. As a result, each cell of the contingency table of Fig. 1(bottom) has 50. Rows of the contingency table capture the distribution of the clusters



50	50
50	50

Fig. 1 Two alternative clusterings compared. The rows of the 2×2 contingency table denote clusters from top left and columns denote clusters from top right. The cells of the contingency table indicate the number of data points that are common among the respective clusters. The contingency table of this figure is an ideal case and hence possesses a uniform distribution.

of Clustering 1 in Clustering 2, and the columns capture the distribution of the clusters of Clustering 2. The deviation of this distribution from the uniform distribution serves as our objective criterion.

It is important to note, however, that we do not have direct control over the contingency table entries. These entries are computed from the clusters, which in turn are defined by the prototypes. The objective function can hence be formulated as

$$\text{Obj} = \mathcal{F}(\hbar(v(\text{Data}, \text{Proto1}), v(\text{Data}, \text{Proto2}))).$$

Here, the fixed variable is Data. Proto1 and Proto2 are our free variables (these are the cluster centroids for two datasets). $\mathcal{F}$, $\hbar$, and v are functions. $\mathcal{F}$ is the objective function applied over the contingency table that measures dissimilarity over clusterings. $\hbar$ is the function that computes the values in the contingency table. $\hbar$ actually depends on the cluster membership probability function v. Finally, v computes two sets of clusters given the prototypes Proto1 and Proto2 (in other words, v computes vectors' cluster membership probabilities). The goal is hence to optimize Obj for Proto1 and Proto2.

To find three or more alternative clusterings (simultaneously), the above function can be trivially generalized, e.g.,

$$\begin{aligned}\text{Obj} = \mathcal{F}(\hbar(v(\text{Data}, \text{Proto1}), v(\text{Data}, \text{Proto2}))) \\ + \mathcal{F}(\hbar(v(\text{Data}, \text{Proto1}), v(\text{Data}, \text{Proto3}))) \\ + \mathcal{F}(\hbar(v(\text{Data}, \text{Proto2}), v(\text{Data}, \text{Proto3}))).\end{aligned}$$

In other words, all three clusterings must be pairwise different. These notions can also be easily adapted to the sequential mining case where we are given a partition of the data and need to identify a clustering alternative to the given partition:

$$\text{Obj} = \mathcal{F}(\hbar(\text{Clust1}, v(\text{Data}, \text{Proto2}))).$$

Here, Clust1 is the supplied clustering and Proto2 denotes the to-be-found prototype vectors.

2.2 Modeling locality

Now we turn our attention to modeling locality of clusters. It is well known that, for a clustering to satisfy (local) optimality of a sum-of-squared-errors distortion measure, it must satisfy two criteria:

1. Nearest neighbor criterion: A vector (data point) is assigned to the cluster corresponding to the nearest prototype.
2. Centroid criterion: A prototype must be the (possibly weighted) average of the vectors assigned to its cluster.

Classical vector quantization algorithms such as k-means satisfy each of the above criteria alternatively and iteratively. Here, we instead build these criteria into the definition of the cluster assignment function v (see next section for details) rather than as a separate objective measure. In this manner, by optimizing the objective criterion presented above, we achieve the twin goals of dissimilarity across clusterings and locality within clusterings, essentially by solving a bilevel optimization problem. Note that the user of our framework does not need to provide any explicit parameter for the tradeoff between locality and dissimilarity, because there is not such tradeoff between the two levels. As explained in Section 2.1, our

objective function $\mathcal{F}$ evaluates the contingency table. The contingency table captures the dissimilarity and the contingency table computation (the function $\hbar$ in Obj) is dependent on locality (membership probability function v). The details are provided in the following section.

3 Formalisms

Let $\mathcal{W} = \{\mathbf{w}_A\}_{A=1}^n$ be a dataset where $\mathbf{w}_s \in \mathbb{R}^{l_w}$ are the real-valued vectors in dataset $\mathcal{W}$. $g : \mathcal{W} \to \mathbb{R}^{l_x}$ is a function that maps vectors from $\mathcal{W}$ into a space $\mathcal{X} = g(\mathcal{W})$ over which vector quantization is conducted. We will occasionally abuse notation and view $\mathcal{W}$ and $\mathcal{X}$ as matrices where the vectors are the rows.

The function g captures any transformations and pre-processing necessary for the algorithm being alternatized. For the classical k-means algorithm, as we will see, g is simply the identity function (i.e., no special pre-processing is required). For other vector quantization algorithms, its definition is more complicated (see the next section for details). In the remainder of this section, we assume that the transformation through g has been performed and that we work with vectors in the transformed space $\mathcal{X}$.

Because we desire alternative sets of clusters, we create $\mathcal{X}' = \mathcal{X}$, an exact replica of $\mathcal{X}$. Let $C_{(x)}$ and $C_{(x')}$ be the cluster indices, i.e., indicator random variables, corresponding to $\mathcal{X}$ and $\mathcal{X}'$, respectively, taking values in $\{1, \ldots, k\}$.

3.1 Assigning vectors of $\mathcal{X}$ and $\mathcal{X}'$ to clusters

Let $\mathbf{m}_{i,\mathcal{X}}$ ($\mathbf{m}_{j,\mathcal{X}'}$) be the prototype vector for cluster i (j) in $\mathcal{X}$ ($\mathcal{X}'$). (These are precisely the quantities we wish to estimate/optimize, but in this section, assume they are given). Let $v_i^{(\mathbf{x}_s)}$ $\left(v_j^{(\mathbf{x}_t)}\right)$ be the cluster membership indicator variables, i.e., the probability that data sample $\mathbf{x}_s$ ($\mathbf{x}_t$) is assigned to cluster i (j) in $\mathcal{X}$ ($\mathcal{X}'$). Thus, $\sum_{i=1}^k v_i^{(\mathbf{x}_s)} = \sum_{j=1}^k v_j^{(\mathbf{x}_t)} = 1$. The traditional *hard* assignment is given by

$$v_i^{(\mathbf{x}_s)} = \begin{cases} 1, \text{ if } \left|\left|\mathbf{x}_s - \mathbf{m}_{i,\mathcal{X}}\right|\right| \leq \left|\left|\mathbf{x}_s - \mathbf{m}_{i',\mathcal{X}}\right|\right|, \quad i' = 1, \ldots k, \\ 0, \text{ otherwise.} \end{cases}$$

(Likewise for $v_j^{(\mathbf{x}_t)}$.) Ideally, we would like a continuous function that tracks these hard assignments to a high degree of accuracy. A standard approach is to use a Gaussian kernel to smooth out the cluster assignment probabilities:

$$v_i^{(\mathbf{x}_s)} = \frac{\exp(-\frac{\rho}{D}||\mathbf{x}_s - \mathbf{m}_{i,\mathcal{X}}||^2)}{\sum_{i'=1}^k \exp(-\frac{\rho}{D}||\mathbf{x}_s - \mathbf{m}_{i',\mathcal{X}}||^2)}, \tag{1}$$

where

$$D = \max_{s,s'} ||\mathbf{x}_s - \mathbf{x}_{s'}||^2, \quad 1 \leq s, s' \leq n.$$

An analogous equation holds for $v_j^{(\mathbf{x}_t)}$. The astute reader would notice that this is really the Gaussian kernel approximation with ρ/D being the width of the kernel. Notice that D is completely determined by the data but ρ is a user-settable parameter, and precisely what we can tune.

3.2 Preparing contingency tables

Preparing the $k \times k$ contingency table (to capture the relationships between entries in clusters across $\mathcal{X}$ and $\mathcal{X}'$) is now straightforward. We simply iterate over the implicit one-to-one relationships between $\mathcal{X}$ and $\mathcal{X}'$: We suitably increment the appropriate entry in the contingency table in a one-to-one relationship fashion:

$$w_{ij} = \sum_{m=1}^{n} v_i^{(\mathbf{x}_m)} v_j^{(\mathbf{x}_m)}, \tag{2}$$

We also define

$$w_{i.} = \sum_{j=1}^{k} w_{ij}, \qquad w_{.j} = \sum_{i=1}^{k} w_{ij},$$

where $w_{i.}$ and $w_{.j}$ are the row-wise and column-wise counts of the cells of the contingency table, respectively.

We will find it useful to define the probability distribution $\alpha_i(j)$, $i = 1, \ldots, k$ of the row-wise random variables and $\beta_j(i)$, $j = 1, \ldots, k$ of the column-wise random variables as follows

$$\alpha_i(j) = \frac{w_{ij}}{w_{i.}}, \tag{3}$$

$$\beta_j(i) = \frac{w_{ij}}{w_{.j}}. \tag{4}$$

The row-wise distributions represent the conditional distributions of the clusters in $\mathcal{X}'$ given the clusters in $\mathcal{X}$; the column-wise distributions are also interpreted analogously.

3.3 Evaluating contingency tables

Now that we have a contingency table, we must evaluate it to see if it reflects disparateness of the two clusterings. Ideally, we expect that the contingency table would be uniform in a perfect alternative clustering. Therefore for our objective criterion, we compare the row-wise and column-wise distributions from the contingency table entries to the uniform distribution. We use KL-divergences (Kullback and Leibler (1951)) to define the objective function (lower values are better)

$$\mathcal{F} = \sum_{i=1}^{k} D_{KL}\left(\alpha_i \,\|\, U\left(\frac{1}{k}\right)\right) + \sum_{j=1}^{k} D_{KL}\left(\beta_j \,\|\, U\left(\frac{1}{k}\right)\right). \tag{5}$$

Note that the row-wise distributions take values over the columns and the column-wise distributions take values over the rows of the contingency table, in which case if $\mathcal{F}$ is minimized would result in two alternative clusterings in the two vector sets $\mathcal{X}$ and $\mathcal{X}'$ represented respectively by the rows and columns of the contingency table. Finally observe that the KL-divergence of any distribution with respect to the uniform distribution is proportional to the negative *entropy* ($-H$) of the distribution. Thus we are essentially aiming to maximize the entropy of the cluster conditional distributions between a pair of replica of the same vector-set. Appendix A describes a regularization of Equation (5) for degenerate situations.

Table 1 Four different cluster handlers.

k-means:

1. $\mathcal{X}$=$\mathcal{W}$. Return $\mathcal{X}$.

Spectral clustering (Shi and Malik (2000)):

1. Compute the affinity matrix A for the all-pairs similarity graph G of vectors in $\mathcal{W}$ using a Gaussian similarity function.
2. Compute the diagonal degree matrix D, whose (i,i)th element is the sum of all elements of the ith row of A.
3. Compute the unnormalized Laplacian $L = D - A$.
4. Compute the first k generalized eigenvectors $u_1, ..., u_k$ of the generalized eigenproblem $Lu = \lambda Du$. Package the eigenvectors $u_1, ..., u_k$ as columns into a $n \times k$ matrix and return the row vectors of this matrix as $\mathcal{X}$.

Constrained clustering (Wang and Davidson (2010)):

1. Construct the affinity matrix A and degree matrix D of the similarity graph G as above.
2. Construct the constraint matrix Q such that
   - $Q(i,j) = 1$ if vectors i and j in $\mathcal{W}$ have a must-link constraint,
   - $Q(i,j) = -1$ if vectors i and j in $\mathcal{W}$ have a must-not-link constraint,
   - $Q(i,j) = 0$ otherwise.
3. Compute vol $(G) = \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}$, $\overline{L} = I - D^{-1/2}AD^{-1/2}$, and $\overline{Q} = D^{-1/2}QD^{-1/2}$ where I is the identity matrix.
4. Solve the generalized eigenvalue system
$$\overline{L}u = \lambda\left(\overline{Q} - \frac{\beta}{\text{vol}(G)}I\right)u$$
and preserve the top-k eigenvectors $u_1, ..., u_k$ corresponding to positive eigenvalues as columns in $\mathcal{X} \in \mathbb{R}^{n\times k}$.
5. Return $\mathcal{X}$.

Co-clustering (Dhillon (2001)):

1. Construct the affinity matrix A and degree matrix D of the similarity graph G as above. Form $\mathcal{T} = D^{-1/2}\mathcal{W}D^{-1/2}$.
2. Compute the singular value decomposition of $\mathcal{T}$ and form $l = \lceil \log_2 k \rceil$ singular vectors $u_2, ..., u_{l+1}$ from left unitary matrix and similarly l singular vectors $v_2, ..., v_{l+1}$ from the right unitary matrix, and construct $\mathcal{X} = \begin{bmatrix} D_1^{-1/2}U \\ D_2^{-1/2}V \end{bmatrix}$ where D_1 and D_2 are diagonal matrices such that $D_1(i,i) = \sum_j \mathcal{W}_{ij}$, $D_2(j,j) = \sum_i \mathcal{W}_{ij}$, $U = [u_2, ..., u_{l+1}]$, and $V = [v_2, ..., v_{l+1}]$.
3. Return $\mathcal{X}$.

4 Cluster Handlers

The function $g(\mathcal{W})$ handles the necessary computations to construct $\mathcal{X}$ based on specific details of the clustering algorithm. Table 1 shows how $g(\mathbf{w}_s)$ computes $\mathbf{x}_s$ for k-means clustering, spectral clustering, constrained clustering, and co-clustering.

The k-means handler is trivial as it is simply the identity function. In this case, it is easy to verify that Eqn. 1 denotes soft membership probabilities corresponding to soft k-means algorithms. The remaining three algorithms, which are variants or generalizations of spectral clustering, all perform specific transformations before invoking k-means on the transformed space. Consequently, the handlers in Table 1 perform transformations of $\mathcal{W}$ to $\mathcal{X}$ in line with the semantics of the respective algorithms. The spectral clustering handler solves the underlying generalized eigenproblem and prepares the resulting generalized eigenvectors for k-means clustering. The constrained clustering handler encapsulates must-link (ML) and must-not-link (MNL) constraints into the matrix Q and solves the corresponding general-

ized eigenvalue problem. Finally, the co-clustering framework applies to weighted bipartite graphs and finds partitions for both modes of the graph with one-to-one correspondences between the elements of these partitions. See Dhillon (2001); Shi and Malik (2000); Wang and Davidson (2010) for details of these algorithms.

5 Algorithms

Now we are ready to formally present our data mining algorithms as optimization over the space of prototypes.

5.1 Simultaneous alternative clustering

Our goal is to minimize $\mathcal{F}$, a nonlinear function of $\mathbf{m}_{i,\mathcal{X}}$ and $\mathbf{m}_{i,\mathcal{X}'}$. For this purpose, we adopt an augmented Lagrangian formulation with a quasi-Newton trust region algorithm. We require a flexible formulation with equality constraints (i.e., that mean prototypes lie on the unit sphere) and bound constraints (i.e., that the prototypes are bounded by the max and min (componentwise) of the data, otherwise the optimization problem has no solution). These sphere constraints make sense for spectral and graph based clustering, but not for k-means, and vice versa for the bound constraints. Mathematically, the sphere constraints make the bound constraints redundant, but keeping the bound constraints improves computational efficiency. The LANCELOT software package (Conn et al. (1992)) provides just such an implementation.

For simultaneous alternative clustering, we "pack" all the mean prototype vectors for clusters from both $\mathcal{X}$ and $\mathcal{X}'$ (there are $\eta = k + k$ of them) into a single vector ν of length t. The problem to solve is then

$$\min_{\nu} \mathcal{F}(\nu) \quad \text{subject to } h_i(\nu) = 0,\ i = 1, \ldots, \eta,$$
$$L_j \leq \nu_j \leq U_j, j = 1, \ldots, t.$$

where ν is a t-dimensional vector and $\mathcal{F}$, h_i are real-valued functions continuously differentiable in a neighborhood of the box $[L, U]$. Here the h_i ensure that the mean prototypes lie on the unit sphere (i.e., they are of the form $\left|\left|\mathbf{m}_{1,\mathcal{X}}\right|\right| - 1$, $\left|\left|\mathbf{m}_{2,\mathcal{X}}\right|\right| - 1, \cdots$, $\left|\left|\mathbf{m}_{1,\mathcal{X}'}\right|\right| - 1$, $\left|\left|\mathbf{m}_{2,\mathcal{X}'}\right|\right| - 1, \cdots$). The bound constraints are all set to $[-1, 1]$ assuming the data has been normalized. The augmented Lagrangian Φ is defined by

$$\Phi(\nu, \lambda, \varphi) = \mathcal{F}(\nu) + \sum_{i=1}^{\eta} \big(\lambda_i h_i(\nu) + \varphi h_i(\nu)^2\big), \tag{6}$$

where the λ_i are Lagrange multipliers and $\varphi > 0$ is a penalty parameter. The augmented Lagrangian method (implemented in LANCELOT) to solve the constrained optimization problem above is given in *OptPrototypes*.

In Step 1 of *OptPrototypes*, we initialize the prototypes using a k-means algorithm (i.e., one which separately finds clusters in each dataset without coordination), pack them into the vector ν, and use this vector as the starting point for optimization. For each iteration of the augmented Lagrangian method, we require access to $\mathcal{F}$ and $\nabla\mathcal{F}$ which we obtain by invoking Algorithm *ProblemSetup*.

This routine goes step-by-step through the framework developed in earlier sections to link the prototypes to the objective function. There are no parameters in these stages except for

Algorithm 1 *OptPrototypes*

1. Choose initial values $\nu_{(0)}, \lambda_{(0)}$, set $j := 0$, and fix $\varphi > 0$.
2. For fixed $\lambda_{(j)}$, update $\nu_{(j)}$ to $\nu_{(j+1)}$ by using one step of a quasi-Newton trust region algorithm for minimizing $\Phi\big(\nu, \lambda_{(j)}, \varphi\big)$ subject to the constraints on ν. Call *ProblemSetup* with ν as needed to obtain $\mathcal{F}$ and $\nabla\mathcal{F}$.
3. Update λ by $\lambda_{(j+1)_i} = \lambda_{(j)_i} + 2\varphi h_i\big(\nu_{(j)}\big)$ for $i = 1, \ldots, \eta$.
4. If $\big(\nu_{(j)}, \lambda_{(j)}\big)$ has converged, stop; else, set $j := j + 1$ and go to (2).
5. Return ν.

Algorithm 2 *ProblemSetup*

1. Unpack ν into values for mean prototype vectors.
2. Use Eq. (1) (and its analog) to compute $v_i^{(\mathbf{x}_s)}$ and $v_j^{(\mathbf{x}_t)}$.
3. Use Eq. (2) to obtain contingency table counts w_{ij}.
4. Use Eqs. (3) and (4) to define random variables α_i and β_j.
5. Use Eqn. (5) to compute $\mathcal{F}$ and $\nabla\mathcal{F}$ (see (Tadepalli (2009)).)
6. Return $\mathcal{F}, \nabla\mathcal{F}$.

ρ that controls the accuracy of the Kreisselmeier Steinhauser (KS) approximations, used in Tadepalli (2009) to approximate min and max functions for clustering. ρ is chosen so that the KS approximation error is commensurate with the optimization convergence tolerance. Gradients (needed by the trust region algorithm) are mathematically straightforward but tedious, so are not explicitly given here.

The per-iteration complexity of the various stages of our algorithm can be given as follows:

Step	Time Complexity
Assigning vectors to clusters	$\mathcal{O}(nkl_x)$
Preparing contingency tables	$\mathcal{O}(nk^2)$
Evaluating contingency tables	$\mathcal{O}(k^2)$
Optimization	$\mathcal{O}((\eta+1)t^2)$

Observe that this is a continuous, rather than discrete, optimization algorithm, and hence the overall time complexity depends on the number of iterations, which is an unknown function of the requested numerical accuracy. The above complexity figures do not take into account complexity of the handler functions $g\left(\mathbf{w}_s\right)$ and assume the simplest k-means implementation. For each vector, we compare it to each mean prototype, and an inner loop over the dimensionality of the vectors gives $\mathcal{O}(nkl_x)$. The per-cell complexity for preparing the contingency table will simply be a linear function of the length n of the dataset $\mathcal{W}$. Evaluating the contingency tables requires us to calculate KL-divergences that are dependent on the sample space over which the distributions are compared and the number of such comparisons. Either a row-wise or a column-wise comparison has $\mathcal{O}(k^2)$ time complexity. Finally, the time complexity of the optimization is $\mathcal{O}((\eta+1)t^2)$ per iteration, and the space complexity is also $\mathcal{O}((\eta+1)t^2)$, mostly for storage of Hessian matrix approximations of $\mathcal{F}$ and h_i. Note that $t = 2kl_x$ and $\eta = 2k$. In practice, to avoid sensitivity to local minima, we perform several random restarts of our approach, with different initializations of the prototypes.

5.2 Sequential alternative clustering

The sequential alternative clustering in our framework proceeds exactly the same way as the simultaneous approach described above except that the packing style of the mean prototypes in ν is now changed. Since one clustering is already given as an input in the sequential alternative clustering, we prepare the mean prototypes of $\mathcal{X}$ based on those given assignments. We pack only the mean prototypes of $\mathcal{X}'$ in ν where $t = kl_x$. As a result, the cluster membership probabilities of $\mathcal{X}$ remain the same over the iterations of the optimization, but the mean prototypes of $\mathcal{X}'$ vary. At the end of the optimization, we obtain an alternative clustering for $\mathcal{X}'$.

5.3 Finding additional alternative clusterings

Finding more than two alternative clusterings is also straightforward. As described earlier, all known clusterings and their mean prototypes stay fixed during the optimization and only the desired clustering's prototypes vary.

5.4 Applying expressive cluster-level constraints

Our framework is able to take cluster-level constraints in the form of an expected contingency table. Since there are column distributions and row distributions for one contingency table, naturally the expectation is provided from two viewpoints: expected column view $I_{\mathcal{X}}$ and expected row view, $I_{\mathcal{X}'}$. Equation 5 is now modified to

$$\mathcal{F} = \frac{1}{k}\sum_{i=1}^{k} D_{KL}\Big(\alpha_i \parallel I_{\mathcal{X}'}(i,:)\Big) + \frac{1}{k'}\sum_{j=1}^{k'} D_{KL}\Big(\beta_j \parallel I_{\mathcal{X}}(:,j)\Big). \tag{7}$$

Note that the number of clusters in the two clusterings can be different now (denoted by k and k' in Equation (7)). This allows the user to merge multiple clusters as well as split some of them. (Appendix A describes a regularization of Equation (7) for degenerate situations.) Expected row view and column view can be constructed in different ways. In Section 6.9, we provide an illustrative example that shows how user provided simple binary cluster-level constraints can be converted to expected probabilistic views to obtain a desired clustering. Appendix B describes the algorithms to compute expected row and column views from a user provided constraint table.

5.5 Evaluation

We present here the evaluation metrics for capturing the locality of clusters in their respective spaces as well as for capturing their 'alternativeness' with respect to previously discovered clusterings. The clustering quality is measured using several indicators: vector quantization error (VQE) (Davidson and Basu (2007)), Dunn index (DI) (Dunn (1974)), and average silhouette coefficient (ASC) (Tan et al. (2005)). VQE measures the cohesion of a clustering when the data are replaced by prototype vectors. Smaller VQE values are better. DI measures the separation between clusters and larger values are better. ASC is a measure

that takes both cohesion and separation into account (higher values are better). In our experiments, we utilize either ASC, or use both VQE and DI together to evaluate clusterings.

The level to which two clusterings are alternatives of each other is measured using the Jaccard index (JI) (lower values are better). Given two clusterings $C_{(i)}$ and $C_{(j)}$, JI captures whether, for every pair of vectors in the dataset, the pair are clustered together in both clusterings or separate in both clusterings, or together in one but separate in the other. Specifically, for clusterings $C_{(i)}$ and $C_{(j)}$ the Jaccard index (similarity coefficient) is

$$J_{ij} = \frac{a+b}{\binom{n}{2}}$$

and the Jaccard dissimilarity coefficient (distance) is

$$J_d\left(i,j\right) = 1 - \frac{a+b}{\binom{n}{2}},$$

where a is the number of pairs together in both $C_{(i)}$ and $C_{(j)}$, b is the number of pairs separate in both $C_{(i)}$ and $C_{(j)}$, and n is the number of vectors in the dataset.

To assess the quality of clustering alternatives as they are discovered, we track the Jaccard dissimilarity $J_d\left(i,j\right)$ between the newly discovered clustering $C_{(k)}$ and any previous ones $C_{(i)}$, $i < k$, by computing

$$\min_{i<j\leq k} J_d\left(i,j\right).$$

A plot of this min against discovered clusterings is referred to as the minimum dissimilarity plot. The minimum dissimilarity for the first clustering $C_{(1)}$ is set to be 1 and the minimum dissimilarity for subsequent clusterings $C_{(k)}$ will decrease monotonically. How fast it decreases before reaching 0.0 suggests the potential for finding alternatives. Once we reach 0.0, we conclude that there are no further alternatives possible.

6 Experimental Results

In this section we present evaluations of our framework using synthetic and real-world datasets. The questions we seek to answer are:

1. Can the framework help reveal a dataset’s intrinsic potential for alternative clusterings? At what point should we abandon the search for alternatives? (Section 6.1)
2. How does the runtime of the framework scale with increasing dimensions, increasing number of clusters, and increasing number of data points? (Section 6.2)
3. How well does our framework perform when compared with existing alternative clustering algorithms? (Sections 6.3 and 6.4)
4. How do the quality of clusterings computed by our framework compare with the clusterings computed by the original ‘unalternatized’ algorithm? (Sections 6.5, 6.6, 6.7, and 6.8)
5. How can we incorporate user provided cluster-level constraints to steer clustering results using our framework? (Section 6.9)

All the experiments in this paper were conducted on a single machine with an Intel Core2 Quad CPU (Q9459@2.66GHz).

6.1 How many alternatives?

We utilize a 2D synthetic dataset having six Gaussian distributions, each with 50 points, arranged uniformly around a circle. Fig. 2 depicts the clusterings discovered by our framework for a setting of three clusters. Observe that we mine three different clusterings before we encounter a repetition. Fig. 3 (left) tracks the quality of clusterings as they are mined, specifically their minimum dissimilarity

$$\min_{i<j\leq m} J_d\left(i,j\right), \quad 1 \leq m \leq 4.$$

Clustering 1 is the reference and has a score of 1.0. We see a monotonic decrease in the dissimilarity score for the first three clusterings. Clustering 4 has a dissimilarity of 0 and this suggests that we should stop seeking further alternatives. Fig. 3 (right) depicts the average silhouette coefficients (ASC) for each discovered clustering. Note that all the discovered clusterings have positive ASCs indicating both cohesion and separation of the underlying clusters.

The number of clusterings discovered before we run out of alternatives is a complex function of the number of clusters and the nature of the dataset. Using the same dataset as shown in Fig. 2, we varied k, the number of clusters sought. For each setting, we computed alternative clusterings until we experienced no further possibilities. Fig. 4 demonstrates the results. For example, there is only a single clustering with $k = 1$ (likewise with $k = 6$), but three different clusterings with $k = 2$, 3, and 4 (the reader can verify why this is so). The number of clusterings is highest with 6 possible alternatives at $k = 5$.

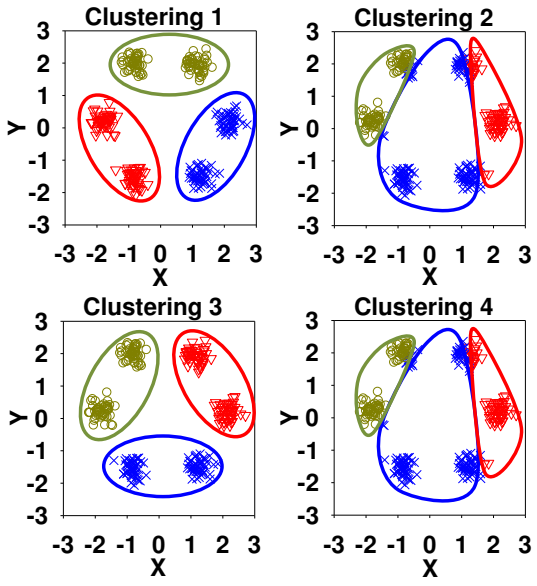


Fig. 2 Alternative clusterings (k=3) for a dataset of six Gaussian distributions arranged around a circle. We identify three clusterings before we encounter a repetition.

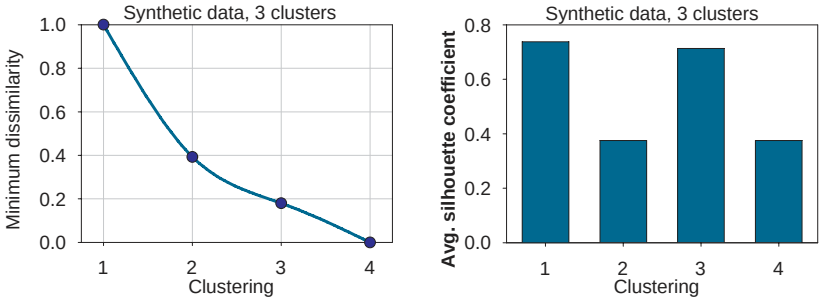


Fig. 3 Minimum dissimilarities and average silhouette coefficients of the clusterings with k=3.

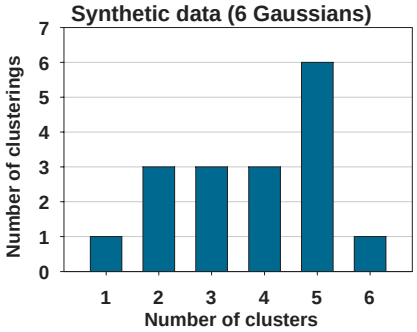


Fig. 4 Number of clusterings discovered with different numbers of clusters.

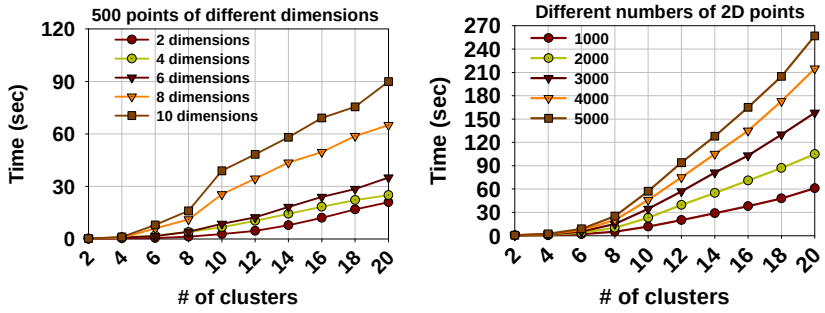


Fig. 5 Runtime characteristics.

6.2 Runtime characteristics

Fig. 5 (left) depicts the runtime behavior of our alternatization framework with the basic k-means algorithm. While the runtime monotonically increases with number of clusters, number of data points, and number of dimensions, we see that the increases are modest. Also note that the runtime includes time for the clustering handler (which is specific to the algorithm being alternatized) and the time for the optimization.

6.3 Comparison with the alternative clustering algorithm of Qi and Davidson (2009)

We compare the quality of clusters computed by our alternatization framework with the alternative clustering framework of Qi and Davidson (2009). Since the latter is a sequential approach, for a fair comparison we set up our framework in a sequential fashion as well. We first obtain a k-means clustering of the dataset, and then alternatize this clustering using Qi and Davidson's framework as well as ours. We applied both frameworks on four UCI repository datasets—Glass, Ionosphere, Vehicle, and Iris—characteristics of which are shown in Table 2. We find alternative clusterings with k equal to the number of classes in each dataset. The results in this subsection are averaged over 10 runs of both our approach and that of Qi and Davidson.

Fig. 6 (a) depicts the Jaccard (similarity) index between the clustering obtained by k-means and the clusterings obtained by either our algorithm or the framework of Qi and Davidson. It shows that our framework provides lower Jaccard index (and hence better alternatives) than the work of Qi and Davidson (2009) with all of the datasets. Another measure of alternativeness is our objective function itself (lower values being better). Fig. 6 (b) shows a comparison of the two approaches in terms of our objective function ($\mathcal{F}$). It shows that $\mathcal{F}$ is much lower with our approach when compared with that of Qi and Davidson (2009). Note

Table 2 Four UCI datasets.

	# Instances	# features	# classes
Glass	214	10	6
Ionosphere	351	34	2
Vehicle	946	18	4
Iris	150	4	3

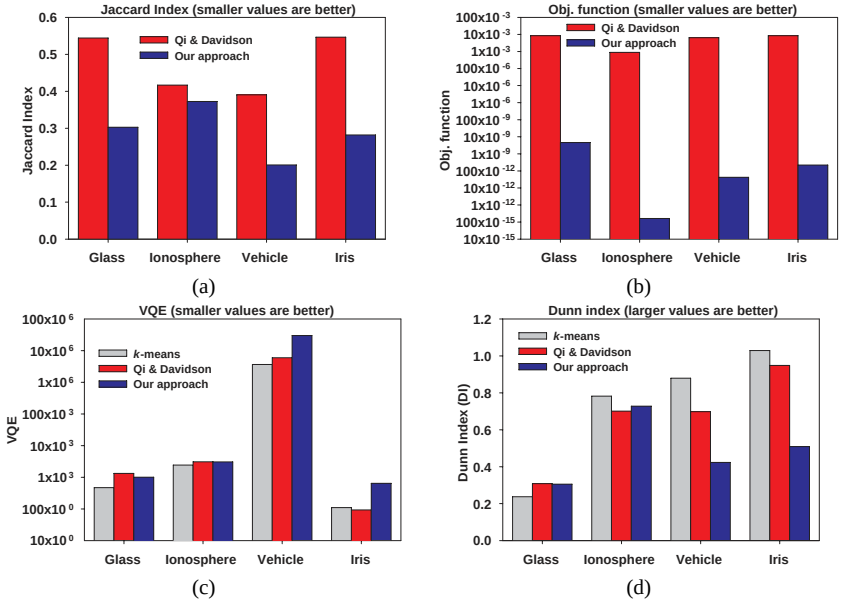


Fig. 6 We used four UCI datasets (Glass, Ionosphere, Vehicle and Iris) to compare the quality of clusterings and alternatives between the approach of Qi and Davidson (2009) and our method. (a) Jaccard index. (b) Objective function. (c) Vector quantization error. (d) Dunn index.

that, for the approach of Qi and Davidson, $\mathcal{F}$ is calculated by comparing the result of k-means (Clustering 1) and the resultant assignments of the discovered clustering (Clustering 2), constructing the contingency table, and evaluating it. Hence, our alternative clustering framework provides better alternatives in terms of both Jaccard index and $\mathcal{F}$.

Considering the measure of cluster quality, Figures 6 (c) and (d) depict the comparison of locality and separation of the clusters in the alternative clusterings discovered by the two approaches using vector quantization error (VQE) and the Dunn index (DI). They depict that, for the glass and ionosphere datasets, VQE and DI of our approach are almost the same as those of Qi and Davidson. For the vehicle and iris datasets, the approach of Qi and Davidson (2009) has better locality and DI. This shows that, while our framework finds high-quality alternatives, for some datasets there can be a compromise in the quality of the clusters.

6.4 Comparison with the state-of-the-art alternative clustering algorithms

We investigate alternative clustering using the Portait dataset as studied in Jain et al. (2008). This dataset comprises 324 images of three people each in three different poses and 36 different illuminations. Preprocessing involves dimensionality reduction to a grid of 64×49 pixels. The goal of finding two alternative clusterings is to assess whether the natural clustering of the images (by person and by pose) can be recovered. We utilize the same 300 features as used in (Jain et al. (2008)) and set up our framework for simultaneous alternative clustering with alternatization of k-means.

Table 3 shows the two contingency tables in the analysis of the Portrait dataset and Table 4 depicts the achieved accuracies using simple k-means, COALA (Bae and Bailey (2006)), convolutional EM (Jain et al. (2008)), decorrelated k-means (Jain et al. (2008)), and our framework for alternative clustering (alternatization of k-means). Our algorithm performs better than all other tested algorithms according to both person and pose clusterings.

Fig. 7 shows how the accuracies of the person and the pose clusterings improve over the iterations, as the objective function is being minimized. The quasi-Newton trust region algorithm guarantees the monotonic improvement of the objective function without directly enforcing error metrics over the feature space. Since the objective function captures the dissimilarity between the two clusterings, indirectly, we notice that the accuracies with respect

Table 3 Contingency tables in analysis of the Portrait dataset. (a) After k-means with random initializations. (b) after using our framework for alternative clustering.

(a)

	C'_1	C'_2	C'_3
C_1	0	0	72
C_2	63	64	0
C_3	3	8	114

(b)

	C'_1	C'_2	C'_3
C_1	36	36	36
C_2	36	36	36
C_3	36	36	36

Table 4 Accuracy on the Portrait dataset.

Method	**Person**	**Pose**
k-means	0.65	0.55
COALA (Bae and Bailey (2006))	0.63	0.72
Conv-EM (Jain et al. (2008))	0.69	0.72
Dec-kmeans (Jain et al. (2008))	0.84	0.78
Our framework (alternatization of k-means)	**0.93**	**0.79**

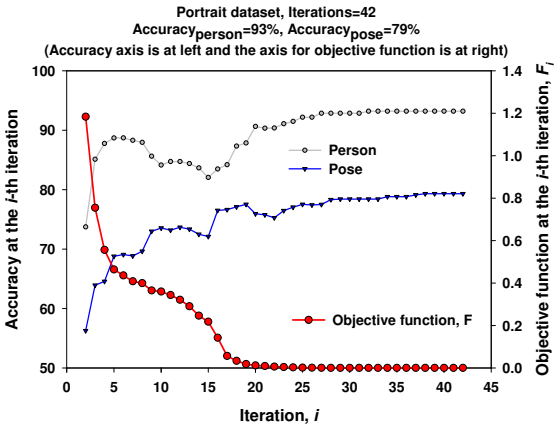


Fig. 7 Monotonic improvement of objective function (finding alternative clusterings for the Portrait dataset).

to the two alternative clusterings improve with the increase in number of iterations (though, not monotically).

6.5 Alternatizing a constrained clustering algorithm

We consider the MAGIC Gamma Telescope dataset (UCI ML Repository), which contains 19,020 instances, 11 attributes, and two class labels. The objective of this experiment is to alternatize a constrained clustering algorithm (Wang and Davidson (2010)) and to assess if the constraints are satisfied in the alternative clustering while maintaining the clusters' quality. We experimented with different numbers of randomly generated constraints. Each must-link (ML) constraint is generated by randomly selecting two datapoints from two different clusters from a k-means clustering outcome. On the other hand, each must-not-link (MNL) constraint is generated by randomly selecting two datapoints from the same cluster. For each case considered here, half of the constraints are ML constraints and the other half are MNL constraints. Figure 8 (a) depicts the number of constraint violations for different constraint sets. It demonstrates that the constrained clustering algorithm of Wang and Davidson and our alternatizatation perform comparably. Fig. 8 (b) shows that the locality of the generated clusters in terms of VQE is slightly better (recall smaller values are better) in the alternative clusterings generated by our framework. Fig. 8 (c) shows that the Dunn index is better (higher) with a smaller number of input constraints in the framework of Wang and Davidson. On the other hand, our framework has better (higher) Dunn index with a larger number of input constraints. Finally, by studying the Jaccard (similarity) indices, Fig. 8 (d) depicts that our framework really generates alternative clusterings with low similarity to the original clustering. However, the similarities tend to become higher with larger number of constraints, indicating a breakdown of alternativeness with higher number of constraints.

Fig. 8 (a, b, and c) plots indicate that the alternative constrained clustering obtained by our framework is quite as good as the constrained clustering results provided by the algorithm of Wang and Davidson (2010). Fig. 8 (d) depicts that the generated alternative clusterings are different than the outcome of the original unalternatized version of the algorithm. This indicates a successful alternatization of a constrained clustering algorithm.

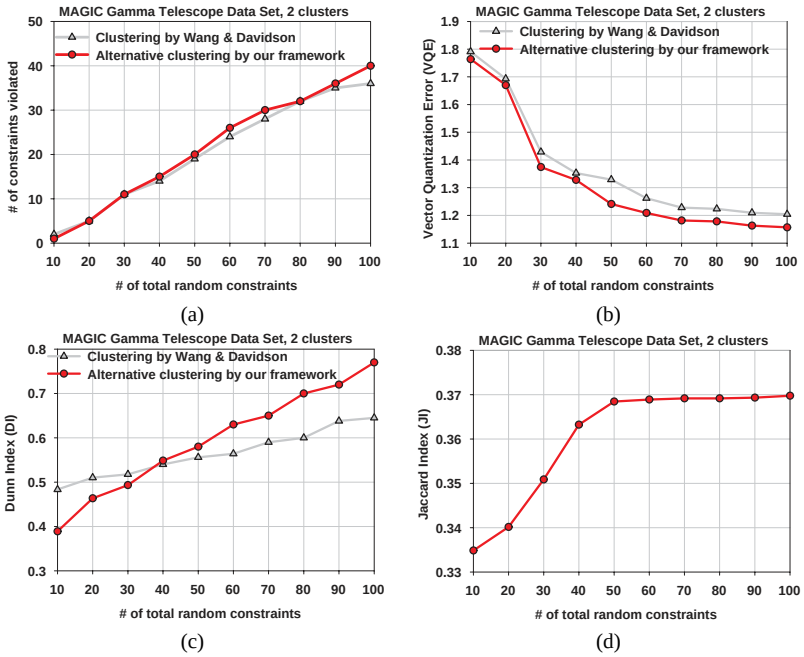


Fig. 8 Alternatizing the algorithm of Wang and Davidson (2010) to mine the MAGIC Gamma Telescope dataset. Plots of *(a)*, *(b)*, and *(c)* are, respectively, numbers of constraint violations, the vector quantization error (VQE), and the Dunn Index (DI) for the native algorithm and our alternatized version. The plot in *(d)* shows the Jaccard index (JI) with various numbers of constraints. This shows that there are alternative clusterings even when the given constraints are satisfied.

6.6 Image segmentation

Image segmentation is a popular application of spectral clustering and in this section we demonstrate the alternatization of Shi and Malik’s 2000 normalized cut algorithm.

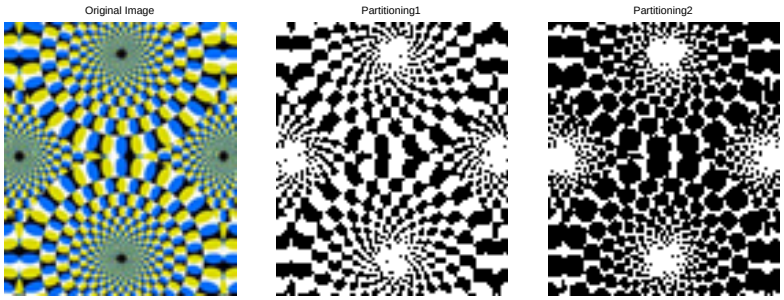


Fig. 9 *(left)* A 85×100 optical illusion. *(middle)* Segmentation ($k = 2$) discovered by Shi and Malik (2000). *(right)* Segmentation after alternatization by our framework. Contrary to appearances, this segmentation is not simply a flipping of colors from the previous segmentation. The Jaccard index between the two segmentations is 0.51.

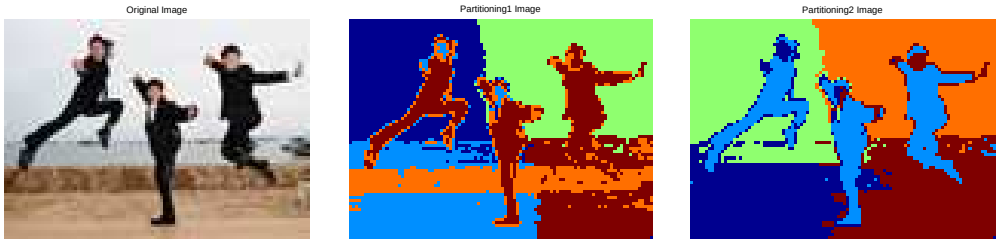


Fig. 10 *(left)* A 100×72 distorted image of actors Liu Fengchao, Jackie Chan, and Wang Wenjie. *(middle)* Clustering 1 ($k = 5$) found by Shi and Malik (2000). *(right)* Segmentation after alternatizing Clustering 1 using our framework. Note that this clustering brings the three people together better. Jaccard index between the two clusterings is 0.78.

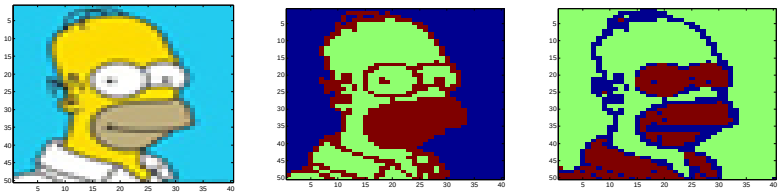


Fig. 11 *(left)* A 40×50 image of Homer Jay Simpson. *(middle)* Segmentation ($k = 3$) found by the algorithm of Shi and Malik (2000) *(right)* Segmentation after alternatizing clustering 1 by our framework. Jaccard index between Clusterings 1 and 2 is 0.72.

The first image we consider is a 85×100 optical illusion (see Fig. 9 (left)). Fig. 9 (middle) shows a segmentation of this image with the normalized cut algorithm with $k = 2$. Fig. 9 (right) is the alternative segmentation discovered by our framework. Two clusters are represented by white and black in both Clustering 1 and 2. The reader might get the illusion from Fig.9 (right) that in the alternative segmentation the cluster labels are merely flipped. However, a closer look reveals key differences. The black parts (or white parts) of Clustering 1 are divided into both black and white in Clustering 2. As a result, the Jaccard index between Figs. 9 (middle) and (right) is 0.51 (a very significant result for the case of just two clusters).

Alternative clusterings can sometimes aid in discerning features of images that were missed in previous clusterings. In Fig. 10, we attempt to discover a segmentation and an alternative segmentation of a 100×72 distorted image of actors Liu Fengchao, Jackie Chan, and WangWenjie. The first segmentation (Fig. 10 (middle)) is obtained by applying Shi and Malik's 2000 segmentation algorithm. An alternative segmentation shown in Fig. 10 (right) is obtained by applying our framework (alternatization of Shi and Malik's segmentation algorithm). Each of these two segmentations has five segments (clusters) denoted by five different colors. All the pixels in a segment (cluster) have the same color. Fig. 10 (middle) shows that the original segmentation algorithm (with $k = 5$) fails to separate the people from the backgrounds. Note that Wang Wenjie (the rightmost actor) is conflated with the background (at bottom side). Although the Jaccard similarity 0.78 between the two segmentations of Fig. 10 (Clustering 1 and 2) is comparatively high, the alternative clustering (right) found by our framework separates the people from the backgrounds better. This has appli-

cations to features like the ‘magic wand’ of photo editing tools like PhotoShop where the user can be presented with a range of alternative possibilities rather than a single selection.

Fig. 11 shows a final example of image segmentation with a 40×50 image of the fictional character Homer Jay Simpson. We attempt to find three segments in one clustering and an alternative segmentation in another clustering. Just like the previous examples of this subsection, three colors are used to show three clusters in both Clustering 1 and 2. In Clustering 1, the shirt, neck, and the head of Simpson are in one cluster. In the alternative clustering (Clustering 2), the lips are better visible. A consensus clustering of these two clusterings can help discern all key characteristics of the image.

6.7 Sequential alternative co-clustering

We consider a subset of DBLP, specifically 12 computer science conferences (ICDM, KDD, SDM, SIGMOD, ICDE, VLDB, CIKM, SIGIR, WSDM, WWW, ICML, and NIPS) and the 500 authors who had top publication counts when aggregated across these conferences. Each author is represented as a norm one 12-length vector of the publications in these conferences.

We first apply the framework of Dhillon (2001) to discover one co-clustering and then repeatedly alternatize it using our framework. Every subsequent co-clustering is alternative to *all* previously discovered ones. Recall that the goal of co-clustering is to discover clusters of authors and concomitant clusters of conferences.

We discovered five different co-clusterings with nonzero minimum Jaccard dissimilarity. That is, Clustering 2 is an alternative of Clustering 1; Clustering 3 is an alternative of Clusterings 1 and 2, and so forth. A partial description of the results is shown in Fig. 12. By tracking a specific author, we can observe how he/she changes clusters and cluster labels in subsequent clusterings. For instance, Corinna Cortes moves through the clusterings: ICML, NIPS $\rightarrow$ SIGIR, ICML $\rightarrow$ ICML, NIPS $\rightarrow$ ICML, NIPS, WSDM $\rightarrow$ ICDM, KDD, SDM. The zig-zag pattern of movements in Fig. 12 reveals the disparateness of consecutive clusterings. Such disparateness indicates that the author has diversity in her publications since a lot of alternative conference groups are obtained. (The term ‘diversity’ might confuse the reader since all the conferences in the current discussion are data mining venues. We assume that each publication venue promotes unique or slightly overlapping areas of data mining.) When we track Abraham Silberschatz in the subsequent alternative clusterings, we find that in each of the clusterings the author is in a cluster that has VLDB and SIGMOD in common. This indicates that the papers of Abraham Silberschatz are published in specific venues and have less diversity.

Here we show how subsequent alternatization of co-clustering might be able to discover insights on different interest groups. An entry in diverse feature clusters in subsequent clusterings indicates diversity and entries staying in similar feature clusters refer to specificity. Quantifications of such diversity and specificity remain as future work.

6.8 Simultaneous alternative co-clustering

In this subsection we alternatize the co-clustering algorithm but in the simultaneous mode. We use a text mining example here as motivated by Dhillon (2001), constructing a text dataset by randomly picking 200 documents from Cranfield and 200 documents from Medline abstracts. We evaluated the contribution of a term by measuring the information gain with respect to the class (Cranfield or Medline), and selected the top 400 terms. So the

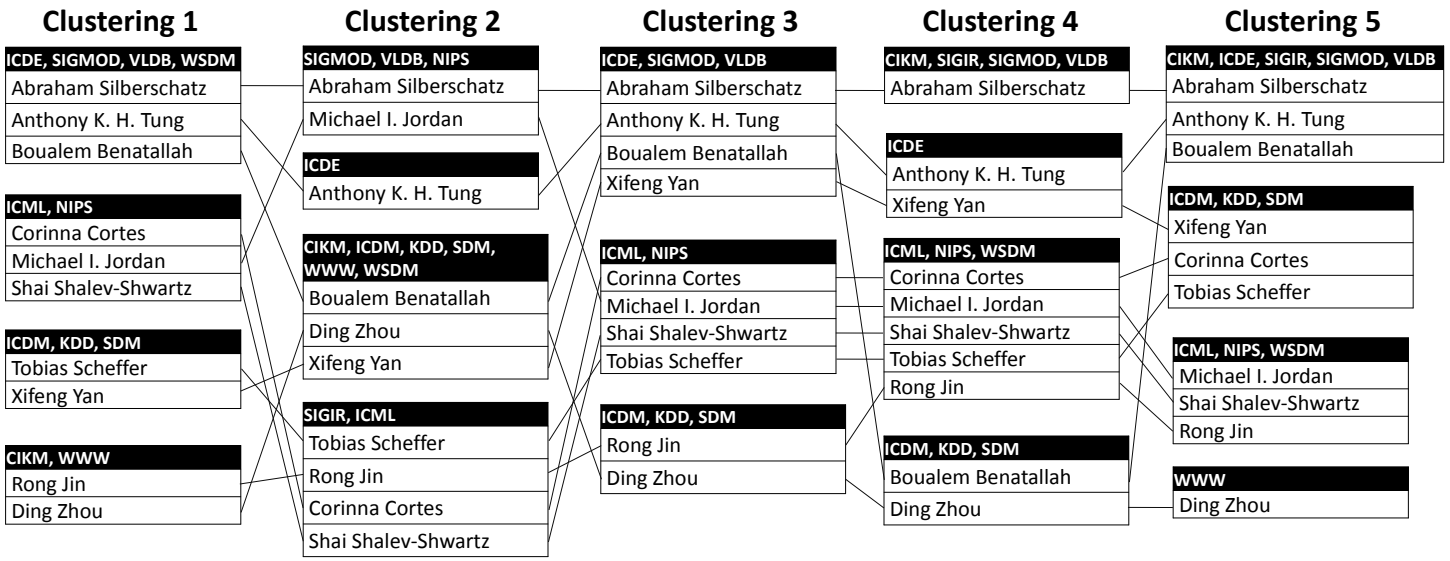
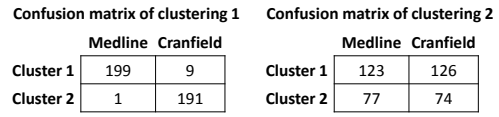
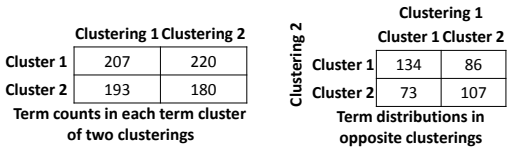


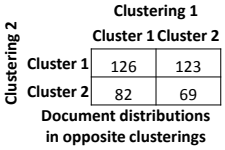
Fig. 12 Zig-zag pattern of authors and conferences as discovered through sequential alternative co-clustering.



(a) Confusion matrices of both the clusterings.



(b) Term counts in clusterings and clusters.



(c) Distribution of documents in the clusterings.

Fig. 13 Document confusion matrices of both the clusterings and distributions of terms and documents in the clusters of the clusterings.

dataset has 400 documents (instances) and 400 terms (attributes). We mined this dataset with $k = 2$ clusters.

Fig. 13 (a) shows the confusion matrices for Clustering 1 and Clustering 2 (recall that *both* are computed using our alternatization framework). The confusion matrix for Clustering 1 indicates that Cluster 1 has 199 documents from the Medline collection whereas Cluster 2 has 191 documents from the Cranfield collection. Numbers in the other cells of the confusion matrix of Clustering 1 are small. On the other hand, Cluster 1 of Clustering 2 contains 249 documents with 123 of them from Medline and the other 126 from Cranfield. 77 documents are from Medline and 74 documents are from Cranfield in Cluster 2 of Clustering 2. This shows that Clustering 2 has no diagonal pattern in its confusion matrix like the confusion matrix of Clustering 1, hence suggesting disparateness (alternativeness).

Fig. 13 (b) (left) shows that terms are almost uniformly distributed in the clusters of the two clusterings. Fig. 13 (b) (right) shows how the terms of one clustering are distributed in the clusters of the other clustering. This shows that there are overlaps of terms between the clusters of the clusterings, indicating some degree of alternativeness in the term clusters.

Fig. 13 (c) catalogs the document distributions in both the clusterings and compares them. It supports the confusion matrix of Fig. 13 (a) (right). To see why, observe that Clustering 1 closely matches with the class labels (see left table of Fig. 13 (a)), whereas the table of Fig. 13 (c) bears resemblance to the confusion matrix of Clustering 2 shown in Fig. 13 (a) (right).

Because this is a co-clustering example, we also consider term distributions across clusters. Fig. 14 depicts the membership probabilities of terms in the two clusters across three alternative clusterings. We see that these patterns are qualitatively different, again suggesting the ability of our framework to recover alternative clusterings.

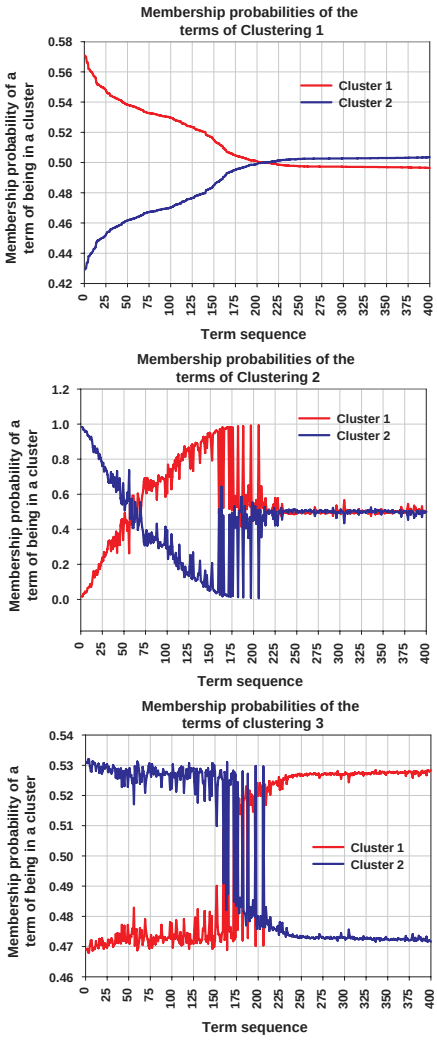


Fig. 14 The effect of alternatization on the membership probabilities of terms in clusters across three alternative clusterings. The terms of all three plots are ordered according to the reference of Clustering 1 so these plots can be compared by visual inspection.

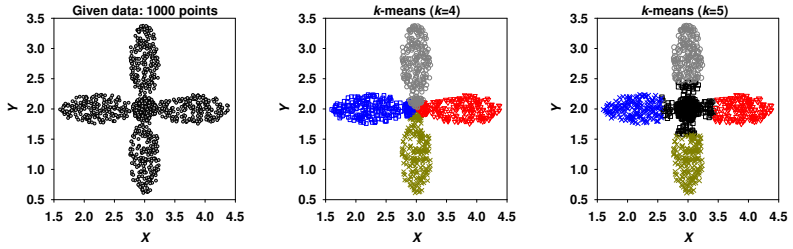
6.9 An example of the use of cluster-level constraints with our framework

To illustrate the idea of cluster-level constraints for alternative clustering, we use a synthetic dataset composed of 1000 two-dimensional points (see Figure 15(a)). The dataset is composed of four petals and a stalk each containing 200 points. When the user applies k-means clustering, with a setting of four clusters (i.e., $k = 4$), the flower is divided into four parts as shown in Figure 15(b) where the petals are indeed in different clusters, but each of the petals also takes up one-fourth of the points from the stalk of the flower. When a setting of five clusters is used, the user obtains the clustering shown in Figure 15(c). It is evident that

the five clusters generated by k-means are not able to cleanly differentiate the stalk from the the petals.

A conventional clustering algorithm like k-means does not take user expectation as an input to produce better clustering results. Even constrained clustering algorithms would require an inordinate number of inputs to clearly separate the stalk from the petals. In our proposed alternative clustering framework (with Equation (7)), the user can provide an input to the algorithm regarding the expected outcome as shown in Figure 16. The constraints shown in the middle of the figure should be read both from *left to right* and from *right to left*. We call these constraints scatter/gather constraints. Reading from left to right, we see that the user expects the four clusters to be broken down (scattered) into five clusters. Reading from right to left, we see that the stalk is expected to gather points from all current clusters, but there is a one-to-one correspondence between the desired petals to the original petals. Figure 16 shows that when cluster-level constraints are added to the k-means clustering (with k=4) results, well-separated petals and stalk are obtained unlike the result provided by simple k-means with k=5 (as shown in Figure 15(c)).

The framework thus requires an existing clustering of the data and a user-expected distribution of the clusters in the new clustering (as shown in Figure 16). We enable the user to supply a binary association matrix between two clusterings in the form of a constraint table (e.g., Fig. 17(a)). The matrix is essentially an encoding of the bipartite graph shown in Fig. 16. For this example, the matrix is of size 4×5 as shown in Fig. 17(a), where each row indicates a petal of the given (k-means) clustering and a column indicates an expected clus-



(a) Original data (b) k-means (k=4) (c) k-means (k=5)

Fig. 15 Clustering the flower dataset. (a) The dataset has 1000 2D points arranged in the form of a flower. (b) Result of k-means clustering with k=4. (c) k-means clustering with k=5. Points from the stalk spill over into the petals.

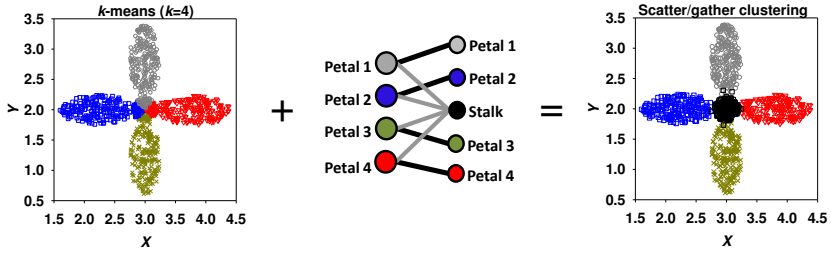


Fig. 16 Clustering the flower dataset with user provided input: Scatter/gather constraints when imposed over a clustering with four clusters yield five clusters with well-separated petals and center with the stalk, unlike Figure 15(c).

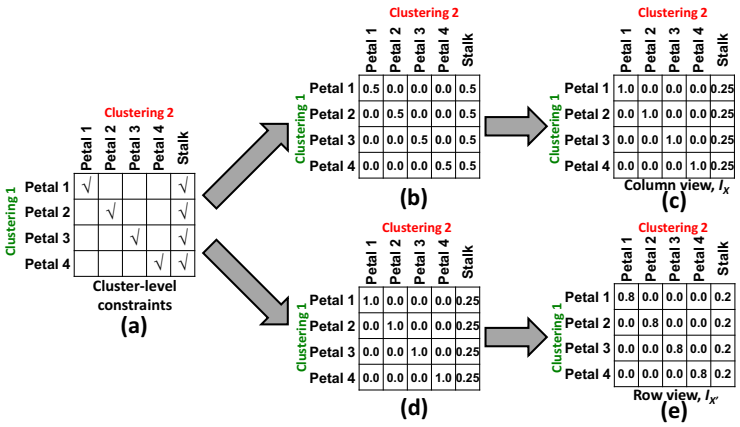


Fig. 17 (a) A tabular representation of the scatter/gather constraints presented in the middle of Figure 16, (b) an intermediate step, (c) Matrix for column-wise conditional distribution, (d) an intermediate step, (e) Matrix for row-wise conditional distribution.

ter of the output of the framework. The tick marks denote the scatter and gather operations desired. Note that each row of Fig. 17(a) has two tick marks and one of these tick marks is in the fifth column, which is the column for the expected stalk of the flower.

A cell of the constraint table is meant to represent an expected (or an ideal case) probability that objects of a cluster in one clustering form part of a cluster in another clustering. We convert the matrix into two probability distributions, one defined along columns and one defined along rows. This results in two matrices, row view $I_{\mathcal{X}'}$ and column view $I_{\mathcal{X}}$ (see Fig. 17(c) and (e)). Although there are many ways to construct such distributions from the binary matrix, we perform simple row-wise and column-wise normalizations here. (Fig. 17 shows the steps involved. Appendix B describes the algorithms to compute expected row and column views from the user-provided constraint table. More complex distributions can, of course, be incorporated based on user input.) Thus, in our example, although not explicitly mentioned by the user, we infer that 25% of the points in the stalk cluster should come from each of the petals of the first clustering. Conversely, these distributions capture the requirement that each of the petals of the first clustering should give up 20% of their points to form the stalk of the second clustering and that the other 80% of the points should go into one cluster of the second clustering.

7 Related Work

As stated earlier, the idea of finding more clusterings than a single one has been studied through various mechanisms and also in various guises, including subspace clustering (Agrawal et al. (1998); Cheng et al. (1999)), nonredundant clustering/views (Cui et al. (2007); Gondek and Hofmann (2005); Niu et al. (2010)), associative clustering (Kaski et al. (2005); Sinkkonen et al. (2004)), metaclustering (Caruana et al. (2006); Zeng et al. (2002)), and consensus clustering (Li et al. (2007); Monti et al. (2003); Strehl and Ghosh (2003)). A key distinguishing feature of our work is the formulation of objective functions for alternatization using a uniform contingency table framework. While contingency tables have been employed elsewhere (Brohee and van Helden (2006); Sinkkonen et al. (2002)), they have been used primarily as criteria to evaluate clusterings. The few works (Govaert and

Nadif (2003); Greenacre (1988); Nadif and Govaert (2005)) that do use contingency tables to formulate objective criteria use them in the context of a specific algorithm such as co-clustering or block clustering, whereas we use them to alternatize a range of algorithms. This work can also be viewed as a form of relational clustering (Hossain et al. (2010)) because we use (two) homogeneous copies of the data to model the ‘alternativeness’ property of two clusterings. However, the locality of clusterings in their respective data spaces is also incorporated into the objective function without any explicit trade-off between locality and the ‘alternativeness’ property. We describe the literature related to our work below.

MDI: The objective functions we have defined have connections to the principle of minimum discrimination information (MDI), introduced by Kullback for the analysis of contingency tables (Kullback and Gokhale (1978)) (the minimum Bregman information (MBI) in (Banerjee et al. (2005)) can be seen as a generalization of this principle). The MDI principle states that if q is the assumed or true distribution, the estimated distribution p must be chosen such that $D_{KL}(p||q)$ is minimized. In our objective functions the estimated distribution p is obtained from the contingency table counts. The true distribution q is assumed to be the uniform or expected distribution. We minimze the KL-divergence from this true distribution as required.

Co-clustering binary matrices, Associative clustering, and Cross-associations: Identifying clusterings over a relation (i.e., a binary matrix) is the topic of many efforts (Chakrabarti et al. (2004); Dhillon et al. (2003)). The former uses information-theoretic criteria to best approximate a joint distribution of two binary variables and the latter uses the MDL (minimum description length) principle to obtain a parameter-free algorithm by automatically determining the number of clusters. Our work is focused on not just binary relations but also attribute-valued vectors. The idea of comparing clustering results using contingency tables was first done in (Kaski et al. (2005)) although our work is the first to attempt “alternatization” of clustering algorithms using the same framework.

Finding disparate clusterings: The idea of finding disparate clusterings has been studied in (Jain et al. (2008)). Here two **dissimilar** clusterings are sought **simultaneously** where the definition of dissimilarity is in terms of orthogonality of the two sets of prototype vectors. This is an indirect way to capture dissimilarity, whereas in our paper we use contingency tables to more directly capture the dissimilarity. Furthermore, our work is able to find two alternative clusterings in a more expressive way. For instance, our framework allows the user to provide cluster-level constraints that are easy to set up. It is diffcult to specify such criteria in terms of instance-level must-link and must-not-link constraints.

Clustering over relation graphs: Clustering over relation graphs is a framework by Banerjee et al. (2007) that uses the notion of Bregman divergences to unify a variety of loss functions and applies the Bregman information principle (from (Banerjee et al. (2005))) to preserve various summary statistics defined over parts of the relational schema. The key difference between this work and ours is that this framework is primarily targeted toward dependent clustering (compression) whereas **our work targets alternative clustering**. We demonstrataded applications of our framework as a dependent clustering tool in (Hossain et al. (2010)). Several key differences beweetn the formawork by Banerjee et al. (2007) and ours are as follows. First, Banerjee et al. aim to **minimize the distortion** as defined through conditional expectations over the original random variables, whereas our work is meant to **both minimize and introduce distortion as needed**, over many subsequent alternatives as applicable. This leads to tradeoffs across alternatives which is unlike the tradeoffs experienced in (Banerjee et al. (2007, 2005)) between compression and accuracy of modeling (see also MIB, discussed below). A second difference is that our work does not directly minimize error metrics over the attribute-value space and uses contingency tables (relationships

between clusters) to exclusively drive the optimization. This leads to the third difference, namely that the distortions we seek to minimize are with respect to idealized contingency tables rather than with respect to the original relational data. The net result of these variations is that relations (idealized as well as real) are given primary importance in influencing the clusterings.

Multivariate information bottleneck: Our work is reminiscent of the multivariate information bottleneck (MIB) (Friedman et al. (2001)), which is a framework for specifying clusterings in terms of two conflicting criteria: locality or compression (of vectors into clusters) and preservation of mutual information (of clusters with auxiliary variables that are related to the original vectors). We share with MIB the formulation of a multicriteria objective function derived from a clustering schema but differ in the specifics of both the intent of the objective function and how the clustering is driven based on the objective function. Furthermore, the MIB framework was originally defined for discrete settings whereas we support soft membership probabilities for multiple alternative clusterings. Some other existing alternative clustering algorithms, e.g., minCEntropy (Vinh and Epps (2010)), CAMI (Dang and Bailey (2010b)), and NACI (Dang and Bailey (2010a)) also use information theoretic measures to compare pairs of clusterings, but our approach explicitly takes advantage of a simple cluster-level contingency table that allows a very flexible setting to encode the user's expectation.

8 Discussion

We have presented a general and expressive framework to alternatize a range of clustering algorithms based on vector quantization. Our results show that the framework is both broadly applicable across algorithms and effective in systematically exploring the space of possible clusterings. We are working on two main directions of future work. First, because alternative clustering is often driven by user input and domain considerations, we would like to develop an interactive tool for guided exploration of complex datasets. Second, combined with our earlier work (Hossain et al. (2010)), we plan to generalize alternative clustering in the direction of modeling and compressing entire relational schemas of data.

Acknowledgements This work is supported in part by the Institute for Critical Technology and Applied Science — Virginia Tech, the US National Science Foundation through grants CCF-0937133, AFRL through grant FA8650-09-2-3938, and AFOSR through grant FA9550-09-1-0153.

References

Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998). Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. *SIGMOD Rec.*, 27(2):94–105.

Bae, E. and Bailey, J. (2006). COALA: A Novel Approach for the Extraction of an Alternate Clustering of High Quality and High Dissimilarity. In *ICDM '06*, pages 53–62.

Banerjee, A., Basu, S., and Merugu, S. (2007). Multi-way Clustering on Relation Graphs. In *SDM '07*, pages 225–334.

Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. (2005). Clustering with Bregman Divergences. *JMLR*, 6:1705–1749.

Brohee, S. and van Helden, J. (2006). Evaluation of Clustering Algorithms for Protein-protein Interaction Networks. *BMC Bioinformatics*, 7:488.

Caruana, R., Elhawary, M., Nguyen, N., and Smith, C. (2006). Meta Clustering. In *ICDM '06*, pages 107–118.

Chakrabarti, D., Papadimitriou, S., Modha, D. S., and Faloutsos, C. (2004). Fully Automatic Cross-associations. In *KDD '04*, pages 79–88.

Cheng, C., Fu, A. W., and Zhang, Y. (1999). Entropy-based Subspace Clustering for Mining Numerical Data. In *KDD '99*, pages 84–93.

Conn, A. R., Gould, N. I. M., and Toint, P. L. (1992). *LANCELOT: A Fortran Package for Large-scale Nonlinear Optimization (Release A)*, volume 17. Springer Verlag.

Cui, Y., Fern, X., and Dy, J. G. (2007). Non-redundant Multi-view Clustering via Orthogonalization. In *ICDM '07*, pages 133–142.

Dang, X. and Bailey, J. (2010a). A Hierarchical Information Theoretic Technique for the Discovery of Non-linear Alternative Clusterings. In *KDD '10*, pages 573–582.

Dang, X. and Bailey, J. (2010b). Generation of Alternative Clusterings Using the CAMI Approach. In *SDM '10*, pages 118–129.

Davidson, I. and Basu, S. (2007). A Survey of Clustering with Instance Level Constraints. *TKDD*, pages 1–41.

Davidson, I. and Qi, Z. (2008). Finding Alternative Clusterings Using Constraints. In *ICDM '08*, pages 773–778.

Dhillon, I. S. (2001). Co-clustering Documents and Words using Bipartite Spectral Graph Partitioning. In *KDD '01*, pages 269–274.

Dhillon, I. S., Mallela, S., and Modha, D. S. (2003). Information Theoretic Co-clustering. In *KDD '03*, pages 89–98.

Dunn, J. C. (1974). Well-Separated Clusters and Optimal Fuzzy Partitions. *Journal of Cybernetics*, 4(1):95–104.

Friedman, N., Mosenzon, O., Slonim, N., and Tishby, N. (2001). Multivariate Information Bottleneck. In *UAI '01*, pages 152–161.

Gondek, D. and Hofmann, T. (2005). Non-redundant Clustering with Conditional Ensembles. In *KDD '05*, pages 70–77.

Gondek, D. and Hofmann, T. (2007). Non-redundant Data Clustering. *Knowl. Inf. Syst.*, 12(1):1–24.

Gondek, D., Vaithyanathan, S., and Garg, A. (2005). Clustering with Model-level Constraints. In *SDM '05*, pages 126–137.

Govaert, G. and Nadif, M. (2003). Clustering with Block Mixture Models. *PR*, 36(2):463–473.

Greenacre, M. (1988). Clustering the Rows and Columns of a Contingency Table. *J. of Classification*, 5(1):39–51.

Hossain, M. S., Tadepalli, S., Watson, L. T., Davidson, I., Helm, R. F., and Ramakrishnan, N. (2010). Unifying Dependent Clustering and Disparate Clustering for Non-homogeneous Data. In *KDD '10*, pages 593–602.

Jain, P., Meka, R., and Dhillon, I. S. (2008). Simultaneous Unsupervised Learning of Disparate Clusterings. In *SDM '08*, pages 858–869.

Kaski, S., Nikkilä, J., Sinkkonen, J., Lahti, L., Knuuttila, J. E. A., and Roos, C. (2005). Associative Clustering for Exploring Dependencies between Functional Genomics Data Sets. *IEEE/ACM TCBB*, 2(3):203–216.

Kullback, S. and Gokhale, D. (1978). *The Information in Contingency Tables*. Marcel Dekker Inc.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86.

Li, T., Ding, C., and Jordan, M. I. (2007). Solving Consensus and Semi-supervised Clustering Problems Using Nonnegative Matrix Factorization. In *ICDM '07*, pages 577–582.

Malakooti, B. and Yang, Z. (2004). Clustering and Group Selection of Multiple Criteria Alternatives with Application to Space-based Networks. *IEEE Trans. on SMC, Part B*, 34(1):40–51.

Miettinen, K. and Salminen, P. (1999). Decision-aid for Discrete Multiple Criteria Decision Making Problems with Imprecise Data. *EJOR*, 119(1):50–60.

Monti, S., Tamayo, P., Mesirov, J., and Golub, T. (2003). Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data. *Machine Learning*, 52:91–118.

Nadif, M. and Govaert, G. (2005). Block Clustering of Contingency Table and Mixture Model. In *IDA '05*, pages 249–259.

Niu, D., Dy, J. G., and Jordan, M. I. (2010). Multiple Non-redundant Spectral Clustering Views. In *ICML '10*, pages 831–838.

Qi, Z. and Davidson, I. (2009). A Principled and Flexible Framework for Finding Alternative Clusterings. In *KDD '09*, pages 717–726.

Ross, D. A. and Zemel, R. S. (2006). Learning Parts-Based Representations of Data. *JMLR*, 7:2369–2397.

Shi, J. and Malik, J. (2000). Normalized Cuts and Image Segmentation. *PAMI*, 22(8):888–905.

Sinkkonen, J. and Kaski, S. (2002). Clustering based on Conditional Distributions in an Auxiliary Space. *Neural Comput.*, 14(1):217–239.

Sinkkonen, J., Kaski, S., and Nikkilä, J. (2002). Discriminative Clustering: Optimal Contingency Tables by Learning Metrics. In *ECML '02*, pages 418–430.

Sinkkonen, J., Nikkilä, J., Lahti, L., and Kaski, S. (2004). Associative Clustering. In *ECML '04*, pages 396–406.

Strehl, A. and Ghosh, J. (2003). Cluster Ensembles — a Knowledge Reuse Framework for Combining Multiple Partitions. *JMLR*, 3:583–617.

Tadepalli, S. (2009). *Schemas of Clustering*. PhD thesis, Virginia Tech.

Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining*. Addison-Wesley.

Vinh, N. X. and Epps, J. (2010). mincentropy: A novel information theoretic approach for the generation of alternative clusterings. In *ICDM '10*, pages 521–530.

Wang, X. and Davidson, I. (2010). Flexible Constrained Spectral Clustering. In *KDD '10*, pages 563–572.

Zeng, Y., Tang, J., Garcia-Frias, J., and Gao, G. R. (2002). An Adaptive Meta-Clustering Approach: Combining the Information from Different Clustering Results. In *CSB '02*, pages 276–287.

Zhang, W., Surve, A., Fern, X., and Dietterich, T. (2009). Learning Non-redundant Codebooks for Classifying Complex Objects. In *ICML '09*, pages 1241–1248.

APPENDIX

A Regularization

Degenerate situations can arise in some cases where the data points can be assigned with equal probability to every cluster, resulting in a trivial solution for ensuring that the contingency table resembles a uniform distribution. To alleviate this issue we add two terms to Equation (5) giving

$$\begin{aligned}\mathcal{F} = &\sum_{i=1}^{k} D_{KL}\left(\alpha_i \parallel U\left(\frac{1}{k}\right)\right) + \sum_{j=1}^{k} D_{KL}\left(\beta_j \parallel U\left(\frac{1}{k}\right)\right) \\ &- \frac{1}{n}\sum_{s=1}^{n} D_{KL}\left(p\left(V^{(\mathbf{x}_s)}\right) \parallel U\left(\frac{1}{k}\right)\right) \\ &- \frac{1}{n}\sum_{t=1}^{n} D_{KL}\left(p\left(V^{(\mathbf{x}_t)}\right) \parallel U\left(\frac{1}{k}\right)\right). \end{aligned} \tag{8}$$

Each of these two terms ensures that the probability distribution of each point being assigned to the clusters is nonuniform. The negative signs of the two additional terms ensure this nonuniformity. $p\left(V^{(\mathbf{x}_s)}\right)$ refers to the vector containing the cluster membership probabilities of the sth datapoint of $\mathcal{X}$ (likewise, $p\left(V^{(\mathbf{x}_t)}\right)$). U is the uniform distribution over k or k' clusters.

Equation (7) with regularization is:

$$\begin{aligned}\mathcal{F} = &\frac{1}{k}\sum_{i=1}^{k} D_{KL}\left(\alpha_i \parallel I_{\mathcal{X}'}(i,:)\right) + \frac{1}{k'}\sum_{j=1}^{k'} D_{KL}\left(\beta_j \parallel I_{\mathcal{X}}(:,j)\right) \\ &- \frac{1}{n}\sum_{s=1}^{n} D_{KL}\left(p\left(V^{(\mathbf{x}_s)}\right) \parallel U\left(\frac{1}{k}\right)\right) \\ &- \frac{1}{n}\sum_{t=1}^{n} D_{KL}\left(p\left(V^{(\mathbf{x}_t)}\right) \parallel U\left(\frac{1}{k'}\right)\right), \end{aligned} \tag{9}$$

where $I_{\mathcal{X}'}(i,:)$ refers to the ith row of the row view and $I_{\mathcal{X}}(:,j)$ represents the jth column of the column view of the expected contingency table generated from the user provided cluster-level constraint table.

B Computing row and column views

Algorithms 3 and 4 outline the pseudocodes to construct the expected column and row views of the probabilistic contingency table. Figure 17 shows the steps involved. The intermediate matrix between the user-provided cluster-level constraint table and the column view $I_{\mathcal{X}}$ of Figure 17 is formed by replacing the tick marks by the receprocal of number of ticks in a row. This intermediate matrix is represented by a temporary variable T in Algorithm 3. The column view $M = I_{\mathcal{X}}$ is produced from the intermediate matrix, by computing the column-wise probability for each cell (and hence each of the columns sums up to 1.0 in $I_{\mathcal{X}}$). Similarily for the row view $I_{\mathcal{X}'}$, we use columns instead of rows while computing the intermediate matrix to count the distribution of the ticks. As shown in Algorithm 4, a row of $M' = I_{\mathcal{X}'}$ is then computed from the distribution of the row values of the intermediate matrix in the columns (thus rows sum up to 1.0).

Algorithm 3 Conversion of user provided constraint table to column view, $I_{\mathcal{X}}$

Output: $M = I_{\mathcal{X}}$
$B \leftarrow$ User provided constraint table. Assume that B is a binary matrix containing 1 in a cell if there is a tick in the corresponding checkbox, otherwise the cell has a 0.
$n_r \leftarrow$ Number of rows in B
$n_c \leftarrow$ Number of columns in B
$M \leftarrow$ Empty matrix of size B
$T \leftarrow$ A Temporary matrix variable of size B
$R \leftarrow$ A Vector of length n_c.
$C \leftarrow$ A Vector of length n_r.
for $i = 1$ to n_r **do**
  $R(i) \leftarrow sum(B(i,:))$ //$B(i,:)$ refers to the i-th row of B.
end for
for $i = 1$ to n_r **do**
  for $j = 1$ to n_c **do**
    $T(i,j) \leftarrow B(i,j)/R(i)$
  end for
end for
for $j = 1$ to n_c **do**
  $C(j) \leftarrow sum(T(:,j))$ //$T(:,j)$ refers to the j-th col. of T.
end for
for $j = 1$ to n_c **do**
  for $i = 1$ to n_r **do**
    $M(i,j) \leftarrow T(i,j)/C(j)$
  end for
end for

Algorithm 4 Conversion of user provided constraint table to row view, $I_{\mathcal{X}'}$

Output: $M' = I_{\mathcal{X}'}$
$B \leftarrow$ User provided constraint table. Assume that B is a binary matrix containing 1 in a cell if there is a tick in the corresponding checkbox, otherwise the cell has a 0.
$n_r \leftarrow$ Number of rows in B
$n_c \leftarrow$ Number of columns in B
$M' \leftarrow$ Empty matrix of size B
$T \leftarrow$ A Temporary matrix variable of size B
$R \leftarrow$ A Vector of length n_c.
$C \leftarrow$ A Vector of length n_r.
for $j = 1$ to n_c **do**
  $C(j) \leftarrow sum(B(:,j))$ //$B(:,j)$ refers to the j-th col. of B.
end for
for $i = 1$ to n_r **do**
  for $j = 1$ to n_c **do**
    $T(i,j) \leftarrow B(i,j)/C(j)$
  end for
end for
for $i = 1$ to n_r **do**
  $R(i) \leftarrow sum(T(i,:))$ //$T(i,:)$ refers to the i-th row of T.
end for
for $j = 1$ to n_c **do**
  for $i = 1$ to n_r **do**
    $M'(i,j) \leftarrow T(i,j)/R(i)$
  end for
end for