How to "Alternatize" a Clustering Algorithm

M. Shahriar Hossain[1], Naren Ramakrishnan[1], Ian Davidson[2], and Layne T. Watson[1]
[1]Dept. of Computer Science, Virginia Tech, Blacksburg, VA 24061
[2]Dept. of Computer Science, UC Davis, CA 95616

Abstract

Given a clustering algorithm, how can we adapt it to find multiple, non-redundant, 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.

1 Introduction

Alternative clustering (e.g., [12]) 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 [29] and in multiple-criteria decision making [21, 22]. Alternative clustering is also typically considered a precursor step to consensus clustering [20, 23].

While it has been long accepted that clustering formulations are generally under-constrained and hence afford multiple solutions, the idea of explicitly mining alternative clusterings has witnessed a recent surge of interest [3, 6–9, 11–13, 17, 25–27, 36]. 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 clustering(s). In the simultaneous learning formulation, also known as disparate clustering [17], 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 [9] 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. [17] 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. [6] 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 [7] 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. [25] describe an approach that is based on learning multiple subspaces in conjunction with learning multiple alternative clustering solutions by optimizing a single objective function.

As the above discussion shows, there are truly '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 framework builds upon the mathematical machinery introduced in [16] but solves fundamentally different problems. At a high level, the work in [16] can be viewed as exploring 'relational space' whereas our work explores 'algorithm space.' The work in [16] can relate clusterings across multiple domains but restrictive in that it can only relate k-means clusters across a relation. In contrast, the present paper is focused on mining alternative clusterings in a given dataset but expressive in the range of algorithms it can alternatize. In particular, we show that 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.

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 [28], co-clustering of bipartite graphs [10], and constraint-based clustering formulations [34].

2. We are able to find an arbitrary number of 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.

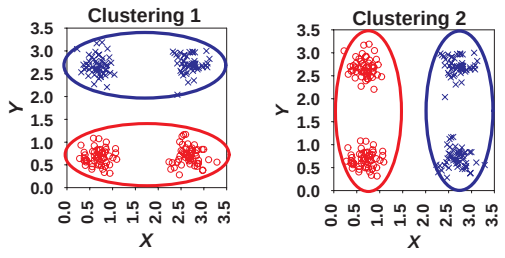
3. Our approach works in both simultaneous and sequential learning formulations. In our experiments here, we demonstrate the use of our simultaneous formulation to first find two alternative clusterings and then use the sequential paradigm to incrementally find more alternative clusterings.

2 Alternatization

Fig. 1 depicts a 2D example involving 200 points where we seek to mine two clusters. In vector quantization algorithms, each cluster is denoted by a prototype and because we desire alternative clusterings, we wish to identify two sets of prototypes—Proto1 and Proto2—each of which has two vectors (one 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. An ideal alternative clustering as shown would result in a uniform (or near uniform) distribution over all contingency table entries. 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



50	50
50	50

Figure 1: Two alternative clusterings compared.

defined by the prototypes. The objective function can hence be formulated as:

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

Here, v computes two sets of clusters given the prototypes Proto1 and Proto2, n compares the results and constructs the contingency table, and $\mathcal{F}$ is the objective function applied over the contingency table that measures dissimilarity over clusterings. The goal is hence to optimize the above function for Proto1 and Proto2.

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

$$\begin{aligned} \text{Obj} \quad &= \quad \mathcal{F}(n(v(\text{Data}, \text{Proto1}), v(\text{Data}, \text{Proto2})) \\ &+ \quad \mathcal{F}(n(v(\text{Data}, \text{Proto1}), v(\text{Data}, \text{Proto3})) \\ &+ \quad \mathcal{F}(n(v(\text{Data}, \text{Proto2}), v(\text{Data}, \text{Proto3})); \end{aligned}$$

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:

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

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 optimality of a sum-of-squared error 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 solve 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.

3 Formalisms

Let $\mathcal{W}$ be a dataset where $\mathcal{W} = \{\mathbf{w}_s\}, s = 1, \ldots, n$ are the real-valued vectors in dataset $\mathcal{W}$. Each vector in $\mathcal{W}$ is of dimension l_w, i.e., $\mathbf{w}_s \in \mathbb{R}^{l_w}$. $g(\mathcal{W})$ is a function that maps vectors from $\mathcal{W}$ into a space over which vector

quantization is conducted. Specifically, $\mathcal{X} = g(\mathcal{W})$ where $\mathcal{X} = \{\mathbf{x}_s\}, s = 1, \ldots, n$ is a set of (real-valued) vectors in the transformed space. Vectors in $\mathcal{X}$ are of dimension l_x, i.e., $\mathbf{x}_s \in \mathbb{R}^{l_x}$. We will occasionally abuse notation and view $\mathcal{W}$ and $\mathcal{X}$ as matrices where the vectors are packaged row-wise.

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 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 $\mathcal{X}'$, an exact replica of $\mathcal{X}$, i.e., $\mathcal{X}' = \{\mathbf{x}_t\}, t = 1, \ldots, n, \mathbf{x}_t \in \mathbb{R}^{l_x}$. Let $C_{(x)}$ and $C_{(x')}$ be the cluster indices, i.e., indicator random variables, corresponding to $\mathcal{X}$ and $\mathcal{X}'$ and let k be the corresponding number of clusters. Thus, both $C_{(x)}$ and $C_{(x')}$ takes values in $\{1, \ldots, k\}$.

3.1 Assigning vectors of $\mathcal{X}$ and $\mathcal{X}'$ to clusters Let $\mathbf{m}_{i,\mathcal{X}}$ be the prototype vector for cluster i in vector-set $\mathcal{X}$ (similarly $\mathbf{m}_{j,\mathcal{X}'}$). (These are precisely the quantities we wish to estimate/optimize for, but in this section, assume they are given). Let $v_i^{(\mathbf{x}_s)}$ (likewise $v_j^{(\mathbf{x}_t)}$) be the cluster membership indicator variables, i.e., the probability that data sample $\mathbf{x}_s$ is assigned to cluster i in vector-set $\mathcal{X}$ (resp). 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:

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

(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. Here, we present a novel smoothing formulation which provides tunable guarantees on its quality of approximation and for which the Gaussian kernel is a special case. First we define

$$\gamma_{(i,i')}(\mathbf{x}_s) = \frac{||\mathbf{x}_s - \mathbf{m}_{i',\mathcal{X}}||^2 - ||\mathbf{x}_s - \mathbf{m}_{i,\mathcal{X}}||^2}{D}, 1 \leq i, i' \leq k,$$

where

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

is the pointset diameter. We now use $\operatorname{argmin}_{i'} \gamma_{(i,i')}(\mathbf{x}_s)$ for cluster assignments so the goal is to track $\min_{i'} \gamma_{(i,i')}(\mathbf{x}_s)$ with high accuracy. The approach we take is to use the Kreisselmeier-Steinhauser (KS) envelope function [19] given by

$$KS_i(\mathbf{x}_s) = \frac{-1}{\rho} \ln \Big[\sum_{i'=1}^k \exp(-\rho \gamma_{(i,i')}(\mathbf{x}_s)) \Big],$$

where $\rho \gg 0$. The KS function is a smooth function that is **infinitely differentiable** (i.e., its first, second, 3rd, .. derivatives exist). Using this the cluster membership indicators are redefined as:

$$\begin{aligned} v_i^{(\mathbf{x}_s)} &= \frac{\exp\Big[\rho KS_i(\mathbf{x}_s)\Big]}{\sum_{i'=1}^k \exp\Big[\rho KS_{i'}(\mathbf{x}_s)\Big]} \\ &= \frac{\exp(-\frac{\rho}{D}||\mathbf{x}_s - \mathbf{m}_{i,\mathcal{X}}||^2)}{\sum_{i'=1}^k \exp(-\frac{\rho}{D}||\mathbf{x}_s - \mathbf{m}_{i',\mathcal{X}}||^2)} \end{aligned} \tag{3.1}$$

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. However, this novel derivation helps tease out how the width must be set in order to achieve a certain quality of approximation. 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:

$$w_{ij} = \sum_{m=1}^n v_i^{(\mathbf{x}_m)} v_j^{(\mathbf{x}'_m)}, \tag{3.2}$$

where $\mathbf{x}_m$ and $\mathbf{x}'_m$ refer to two copies of the same vector in vector-sets $\mathcal{X}$ and $\mathcal{X}'$.

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 row-wise random variables $\alpha_i, i = 1, \ldots, k$ and column-wise random variables $\beta_j, j = 1, \ldots, k$ with probability distributions as follows

$$p(\alpha_i = j) = p(C_{(x')} = j | C_{(x)} = i) = \frac{w_{ij}}{w_{i.}}, \tag{3.3}$$

$$p(\beta_j = i) = p(C_{(x)} = i | C_{(x')} = j) = \frac{w_{ij}}{w_{.j}}. \tag{3.4}$$

The row wise distributions represent the conditional distributions of the clusters in vector-set $\mathcal{X}$ 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 to define the objective function (lower values are better):

(3.5)
$$\mathcal{F} = \sum_{i=1}^{k} D_{KL}\Big(\alpha_i || U(\frac{1}{k})\Big) + \sum_{j=1}^{k} D_{KL}\Big(\beta_j || U(\frac{1}{k})\Big).$$

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.

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 $g(\mathcal{W})$ computes $\mathcal{X}$ 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. 3.1 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 generalized 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 [10, 28, 34] for details of these algorithms.

Table 1: Four different cluster handlers.

k-means:

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

Spectral clustering [28]:

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 [34]:

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 [10]:

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 $\mathcal{X}$.

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 non-linear 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 hypersphere) 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). The LANCELOT software package [5] provides just such an implementation.

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

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

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 hypersphere (i.e., they are of the form $||\mathbf{m}_{1,\mathcal{X}}|| - 1, ||\mathbf{m}_{2,\mathcal{X}}|| - 1, \cdots, ||\mathbf{m}_{1,\mathcal{X}'}|| - 1, ||\mathbf{m}_{2,\mathcal{X}'}|| - 1, \cdots$) The bound constraints are uniformly set to $[-1, 1]$. The augmented Lagrangian Φ is defined by

$$\Phi(\nu, \lambda, \varphi) = \mathcal{F}(\nu) + \sum_{i=1}^{\eta} \big(\lambda_i h_i(\nu) + \varphi h_i(\nu)^2\big), \tag{5.6}$$

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*.

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.

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), package them into the vector ν, and use this vector as starting points 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*.

Algorithm 2 *ProblemSetup*

1. Unpackage ν into values for mean prototype vectors.
2. Use Eq. (3.1) (and its analog) to compute $v_i^{(\mathbf{x}_s)}$ and $v_j^{(\mathbf{x}_t)}$.
3. Use Eq. (3.2) to obtain contingency table counts w_{ij}.
4. Use Eqs. (3.3) and (3.4) to define r.v.s α_i and β_j.
5. Use Eqn. (3.5) to compute $\mathcal{F}$ and $\nabla\mathcal{F}$ (see [33].)
6. Return $\mathcal{F}, \nabla\mathcal{F}$.

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 ρ which controls the accuracy of the KS approximations. It 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 $g(\mathcal{W})$ 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 which 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 such comparisons 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 packaging 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 package 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 Evaluation We present here the evaluation metrics for capturing the locality of clusters in their respective spaces as well as for capturing their 'alternativeness' w.r.t. previously discovered clusterings. The clustering quality is measured using several indicators: Vector Quantization Error (VQE), Dunn Index (DI), and Average Silhouette Coefficient (ASC). 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 (higher values are better). Given two clusterings $C_{(i)}$ and $C_{(j)}$, and for every pair of vectors in the dataset, JI investigates whether the pair are clustered together in both clusterings or separate in both clusterings, or together in one but separate in the other. Specifically, it is defined as:

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

where a is the number of agreements across all pairs, b is the number of disagreements, 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 $J_d(i,j) = 1 - J_{ij}$ between the newly discovered clustering and any previous ones:

$$\min_{i<j} J_d(i,j)$$

A plot of J_d against discovered clusterings is referred to as the minimum dissimilarity plot. The minimum dissimilarity for the first clustering is set to be 1 and the minimum dissimilarity for subsequent clusterings 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?

2. How does the runtime of the framework scale with increasing dimensions, increasing number of clusters, and increasing data points?

3. How do the quality of clusterings computed by our framework compare with the clusterings computed by the original 'un-alternatized' algorithm?

4. What is the potential for knowledge discovery when our framework is applied on real-world datasets?

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

6.1 How many alternatives? We utilize a 2D synthetic dataset having six Gaussians, each with 50 points, arranged around a circle. Fig. 2 depicts the clusterings discovered by our framework for a setting of three clusters. Observe that we

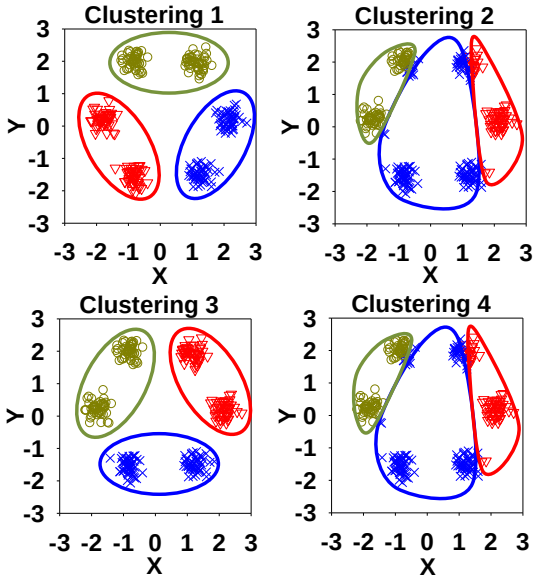


Figure 2: Alternative clusterings (k=3) for a dataset of six Gaussian distributions arranged around a circle. We identify three clusterings before we encounter a repetition.

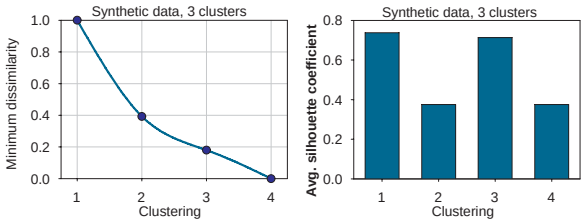


Figure 3: Minimum dissimilarities and average silhouette coefficients of the clusterings with k=3.

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 with any previously discovered clustering. 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.

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 mod-

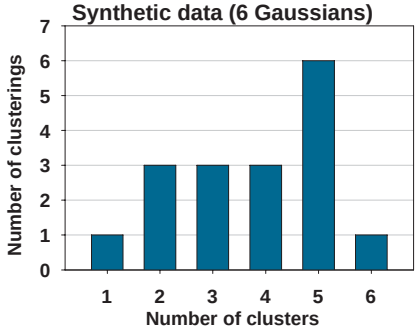


Figure 4: Number of clusterings discovered with different numbers of clusters.

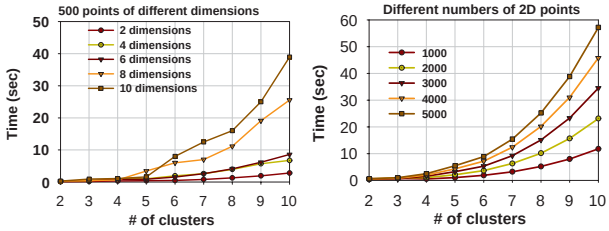


Figure 5: Runtime characteristics.

est. 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.

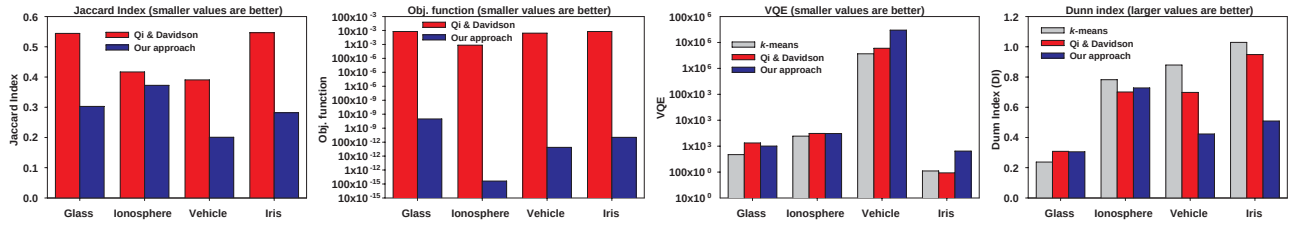
6.3 Comparison with [26] We compare the quality of clusters computed by our alternatization framework with the alternative clustering framework of Qi and Davidson [26]. Since the latter is a sequential approach, for a fair comparison we setup our framework in a sequential fashion as well. We first obtain 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 the frameworks on four UCI repository datasets—Glass, Ionosphere, Vehicle, and Iris—characteristics of whom are shown in Table 2. We find alternative clusterings with k set equal to the number of classes in each dataset. The results in this subsection are averaged over 10 runs of both our approach and the approach of Qi and Davidson.

Fig. 6 (a) depicts the Jaccard 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 [26] 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 [26]. Note 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, Fig. 6 (c)

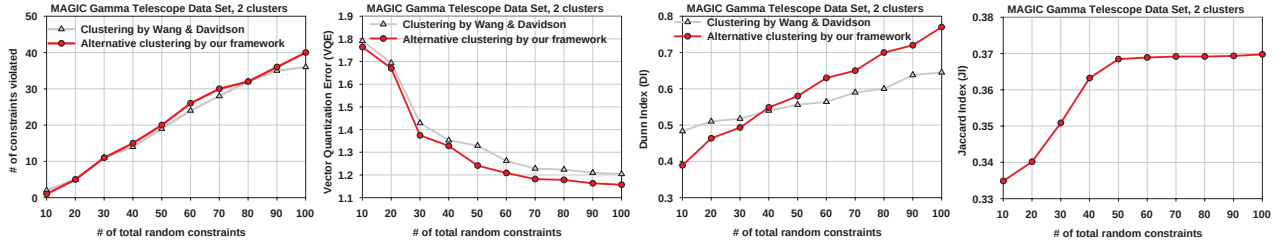
Table 2: Four UCI datasets.

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



(a) (b) (c) (d)

Figure 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 [26] and our method. (a) Jaccard index. (b) Objective function. (c) Vector quantization error. (d) Dunn index.



(a) (b) (c) (d)

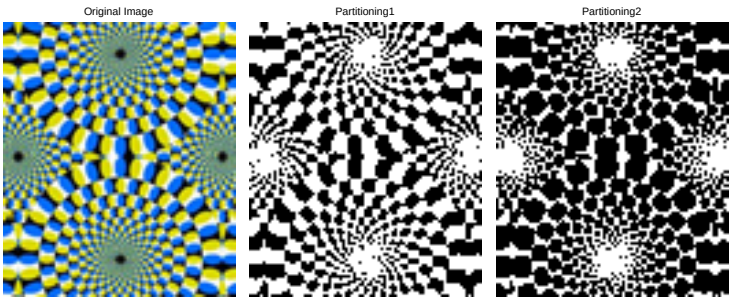
Figure 7: Alternatizing the algorithm in [34] to mine the MAGIC Gamma Telescope dataset. Plots of *(a)*, and *(b) (c)* are respectively numbers of constraint violations, the Vector Quantization Error (VQE), Dunn Index (DI) for the native algorithm and our alternatized version. The plot in *(d)* shows the Jaccard Index (JI) with various numbers of constraints. This shows that there are alternative clusterings even when the given constraints are satisfied.

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 the approach of Qi and Davidson. For the vehicle and iris datasets, the approach of [26] has slightly better locality and DI. This shows that our framework finds high-quality alternatives without compromising the quality of clusters.

6.4 Comparison with [34] We consider the MAGIC Gamma Telescope dataset (UCI ML Repository) which contains 19,020 instances, 11 attributes, and 2 class labels. The objective of this experiment is to alternatize a constrained clustering algorithm ([34]) and to assess if the constraints are satisfied in the alternative clustering while maintaining the clusters' quality. We experimented with different quantities of randomly generated constraints. For each case considered here, half of the constraints are must-link (ML) constraints and the other half are must-not-link (MNL) constraints. Figure 7 (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. 7 (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. 7 (c) shows that the Dunn index is better 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 indices (similarity), Fig. 7 (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.

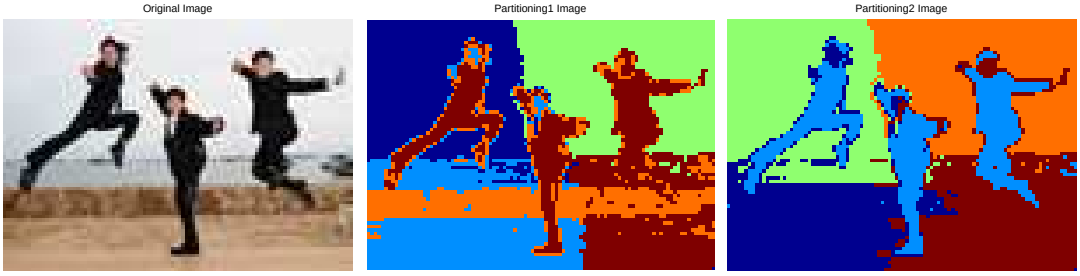
6.5 Image segmentation Image segmentation is a popular application of spectral clustering and in this section we demonstrate the alternatization of Shi and Malik's [28] normalized cut algorithm.

The first image we consider is a 85×100 optical illusion (see Fig. 8 (left)). Fig. 8 (middle) shows a segmentation of this image with the normalized cut algorithm with k=2. Fig. 8 (right) is the alternative segmentation discovered by our framework. The reader might get the illusion from Fig.8 (right) that in the alternative segmentation the cluster labels are merely flipped. However, a closer look reveals key differences. In fact, the Jaccard index between Figs. 8 (middle) and (right) is 0.51 (a very significant result for the case of just two clusters – refer back to the synthetic data example).



Original image Clustering 1 Clustering 2

Figure 8: *(left)* A 85×100 optical illusion. *(middle)* Segmentation ($k = 2$) discovered by [28]. *(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.



Original image Clustering 1 Clustering 2

Figure 9: *(left)* A 100×72 distorted image of actors Liu Fengchao, Jackie Chan, and Wang Wenjie. *(middle)* Clustering 1 ($k = 5$) found by [28]. *(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.

Original image Clustering 1 Clustering 2

Figure 10: *(left)* A 40×50 image of Homer Jay Simpson. *(middle)* Segmentation ($k = 3$) found by [28] *(right)* Segmentation after alternatizing clustering 1 by our framework. Jaccard index between clusterings 1 and 2 is 0.72.

Alternative clusterings can sometimes aid in discerning features of images that were missed in previous clusterings. Fig. 9 shows an example where spectral clustering (with $k = 5$) fails to separate the people from the backgrounds. Note that Wang Wenjie (the right most actor) is conflated with the background (at bottom side). Although the similarity 0.78 between the two segmentations is comparatively high, the alternative clustering found by our framework separates the people from the backgrounds better. This has applications 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. 10 shows a final example of image segmentation with a 40×50 image of the fictional character Homer Jay Simpson. 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.

6.6 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 (normalized to norm 1) 12-length vector of the publications in these conferences.

We first apply Dhillon's framework [10] 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 non-

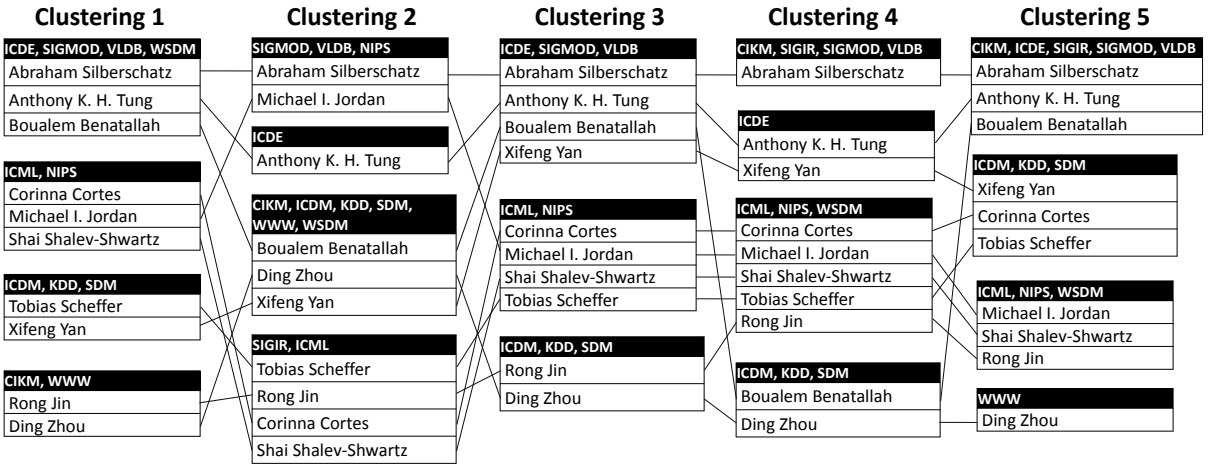


Figure 11: Zig-zag pattern of authors and conferences as discovered through sequential alternative co-clustering.

zero minimum dissimilarity. A partial description of the results is shown in Fig. 11. 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 → SIGIR, ICML → ICML, NIPS → ICML, NIPS, WSDM → ICDM, KDD, SDM. The zig-zag pattern of movements in Fig. 11 reveals the disparateness of consecutive clusterings.

6.7 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 in [10]. As motivated there, we constructed a text dataset by randomly picking 200 documents from Cranfield and 200 documents from Medline abstracts. We evaluated

Confusion matrix of clustering 1

	Medline	Cranfield
Cluster 1	199	9
Cluster 2	1	191

Confusion matrix of clustering 2

	Medline	Cranfield
Cluster 1	123	126
Cluster 2	77	74

(a) Confusion matrices of both the clusterings.

	Clustering 1	Clustering 2
Cluster 1	207	220
Cluster 2	193	180

Term counts in each term cluster of two clusterings

Clustering 2 \ Clustering 1	Cluster 1	Cluster 2
Cluster 1	134	86
Cluster 2	73	107

Term distributions in opposite clusterings

(b) Terms counts in clusterings and clusters.

Clustering 2 \ Clustering 1	Cluster 1	Cluster 2
Cluster 1	126	123
Cluster 2	82	69

Document distributions in opposite clusterings

(c) Distribution of documents in the clusterings.

Figure 12: Document confusion matrices of both the clusterings and distributions of terms and documents in the clusters of the clusterings.

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

Fig. 12 (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. 12 (b) (left) shows that terms are almost uniformly distributed in the clusters of the two clusterings. Fig. 12 (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. 12 (c) catalogs the document distributions in both the clusterings and compares them. It supports the confusion matrix of Fig. 12 (a) (right). To see why, observe that Clustering 1 closely matches with the class labels (see left table of Fig. 12 (a)), whereas the table of Fig. 12 (c) bears resemblance to the confusion matrix of Clustering 2 shown in Fig. 12 (a) (right).

Because this is a co-clustering example, we also consider term distributions across clusters. Fig. 13 depicts the membership probabilities of terms in the two clusters across

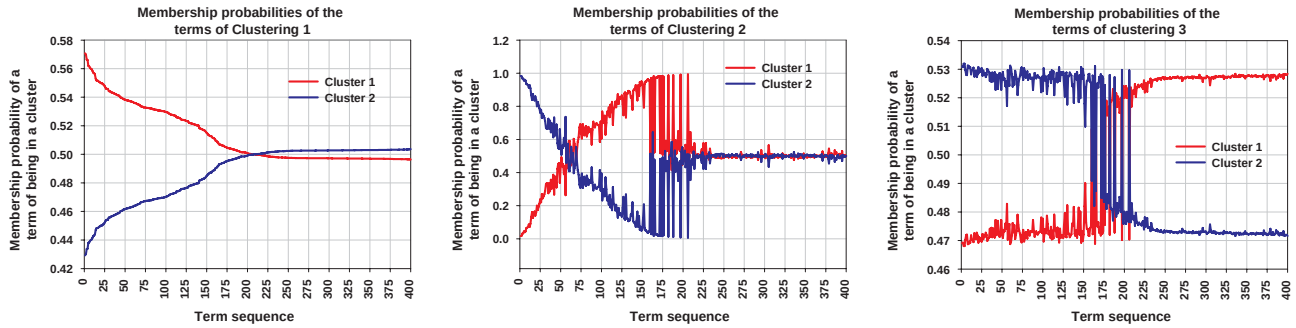


Figure 13: The effect of alternatization on the membership probabilities of terms in clusters across three alternative clusterings. The terms of all three plots are ordered according to the reference of Clustering 1 so these plots can be compared by visual inspection.

three alternative clusterings. We see that these patterns are qualitatively different, again suggesting the ability of our framework to recover alternative clusterings.

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 [1, 4], non-redundant clustering/views [6, 11, 25], associative clustering [18, 31], meta-clustering [3, 35], and consensus clustering [20, 23, 32]. 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 [2, 30], they have been used primarily as criteria to evaluate clusterings. The few works [14, 15, 24] 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 [16] 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.

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 efficient in systematically exploring the space of possible clusterings. We are working on three main directions of future work. First, we would like to increase the expressiveness of our framework to mine alternative sets of clusters with different numbers of clusters. This would result in a 'scatter gather' approach to clustering, where clusters in one alternative clustering can be scattered into multiple clusters in the other alternative clustering (vice versa, for gather). Second, 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. Finally, combined with our earlier work [16], we plan to generalize alternative clustering in the direction of modeling and compressing entire relational schemas of data.

ACKNOWLEDGMENTS

This work is supported in part by NSF FODAVA grant 0937133 "Formal Models, Algorithms, and Visualizations for Storytelling Analytics".

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[19] G. Kreisselmeier and R. Steinhauser. Systematic Control Design by Optimizing a Vector Performance Index. In *IFAC Symp. on CADCS*, pages 113–117, 1979.

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

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

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

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

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

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

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

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

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

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

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

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

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

[33] S. Tadepalli. *Schemas of Clustering*. PhD thesis, Virginia Tech, Feb 2009.

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

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

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