Iterative Clustering of High Dimensional Text Data Augmented by Local Search

Inderjit S. Dhillon* and Yuqiang Guan*
Department of Computer Sciences
University of Texas
Austin, TX 78712-1188, USA
inderjit, yguan@cs.utexas.edu

J. Kogan
Department of Mathematics and Statistics
University of Maryland Baltimore County
Baltimore, MD 21228, USA
kogan@math.umbc.edu

Abstract

The k-means algorithm with cosine similarity, also known as the spherical k-means algorithm, is a popular method for clustering document collections. However, spherical k-means can often yield qualitatively poor results, especially when cluster sizes are small, say 25-30 documents per cluster, where it tends to get stuck at a local maximum far away from the optimal solution. In this paper, we present a local search procedure, which we call "first-variation" that refines a given clustering by incrementally moving data points between clusters, thus achieving a higher objective function value. An enhancement of first variation allows a chain of such moves in a Kernighan-Lin fashion and leads to a better local maximum. Combining the enhanced first-variation with spherical k-means yields a powerful "ping-pong" strategy that often qualitatively improves k-means clustering and is computationally efficient. We present several experimental results to highlight the improvement achieved by our proposed algorithm in clustering high-dimensional and sparse text data.

1. Introduction

Clustering or grouping document collections into conceptually meaningful clusters is a well-studied problem. A starting point for applying clustering algorithms to unstructured document collections is to create a *vector space model*, alternatively known as a *bag-of-words model* [16]. The basic idea is (a) to extract unique content-bearing words from the set of documents treating these words as *features* and (b) to then represent each document as a vector of certain weighted word frequencies in this feature space. Typically, a large number of words exist in even a moderately sized set of documents where a few thousand words or more are common; hence the document vectors are very high-dimensional. In addition, a single document typically contains only a small fraction of the total number of words in the entire collection; hence, the document vectors are generally very sparse, i.e., contain a lot of zero entries.

The k-means algorithm is a popular method for clustering a set of data vectors [5]. The classical version of k-means uses squared Euclidean distance, however this distance measure is often inappropriate for its application to document clustering [18]. An effective measure of similarity between documents, and one that is often used in information retrieval, is cosine similarity, which uses the cosine of the angle between document vectors [16]. The k-means algorithm can be adapted to use the cosine similarity metric to yield the *spherical* k-means algorithm, so named because the algorithm operates on vectors that lie on the unit sphere [4]. Since it uses cosine similarity, spherical k-means exploits the sparsity of document vectors and is highly efficient [3].

The spherical k-means algorithm, similar to the Euclidean algorithm, is a hill-climbing procedure and is prone to getting stuck at a local optimum (finding the global optimum is NP-complete). For large document clusters, it has been found to yield good results in practice, i.e., the local optimum found yields good conceptual clusters [4, 18, 3]. However, as we show in Section 3, spherical k-means often produces poor results on small and moderately sized clusters where it tends to get stuck in a qualitatively inferior local optimum.

In this paper, we present an algorithm that uses local search to refine the clusters produced by spherical k-means. Our refinement algorithm alternates between two phases: (a) first-variation and (b) spherical k-means itself. A first-variation step moves a single document from one cluster to another, thereby increasing the objective function value. A sequence of first-variation moves allows an escape from a local maximum, so that fresh iterations of spherical k-means can be applied to further increase the objective function value. This ping-pong strategy yields a powerful refinement algorithm which often qualitatively improves k-means clustering and is computationally efficient. Note that our refinement algorithm *always* improves upon the input clustering in terms of the objective function value.

Many variants of k-means, such as "batch" and "incre-

*This research was supported by NSF Grant No. ACI-0093404.

mental" versions have been proposed in the literature, see Section 6 for a discussion. The main contribution of this paper is our use of local search for document clustering. Our strategy leads to a "ping-pong" algorithm which alternates between "batch" k-means and first-variation iterations, thereby harnessing the power of both in terms of improved quality of results and computational speed.

We now give an outline of the paper. In Section 2, we present the spherical k-means algorithm while Section 3 presents scenarios in which this algorithm performs poorly. In Section 4, we introduce the first-variation method and present our proposed refinement algorithm. Experimental results in Section 5 show that our algorithm yields qualitatively better results giving higher objective function values. In Section 6 we discuss related work and finally, in Section 7 we present our conclusions and future work.

2. Spherical k-means algorithm

We start with some necessary notation. Let d be the number of documents, w be the number of words and let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_d\}$ denote the set of non-negative document vectors, where each $\mathbf{x}_i \in R^w$ and $\|\mathbf{x}_i\| (\equiv \|\mathbf{x}_i\|_2) = 1$, i.e., each $\mathbf{x}_i$ lies on the unit sphere. A clustering of the document collection is its partitioning into disjoint subsets $\pi_1, \pi_2, ..., \pi_k$, i.e., $\bigcup_{j=1}^{k} \pi_j = \mathbf{X}$ and $\pi_j \cap \pi_l = \phi$, $j \neq l$. For a cluster π we denote the sum $\sum_{\mathbf{x} \in \pi} \mathbf{x}$ by $\mathbf{s}(\pi)$. The concept vector of the cluster π is defined by

$$\mathbf{c}(\pi) = \frac{\mathbf{s}(\pi)}{\|\mathbf{s}(\pi)\|},$$

i.e., the concept vector of the cluster is its normalized expectation. We define the "quality" or "coherence" of a nonempty cluster π as

$$q\left(\pi\right) = \sum_{\mathbf{x} \in \pi} \mathbf{x}^T \mathbf{c}(\pi) = \|\mathbf{s}(\pi)\|. \tag{1}$$

We set $q(\phi) = 0$ for convenience. Finally, for a partition $\{\pi_j\}_{j=1}^k$ we define the objective function to be the sum of the qualities of the k clusters:

$$\mathcal{Q}\left(\{\pi_j\}_{j=1}^k\right) \;=\; \sum_{j=1}^{k} q\left(\pi_j\right) \;=\; \sum_{j=1}^{k} \sum_{\mathbf{x} \in \pi_j} \mathbf{x}^T \mathbf{c}_j,$$

where we have written $\mathbf{c}_j$ for $\mathbf{c}(\pi_j)$. The goal is to find a clustering that maximizes the value of the above objective function. In what follows we present the spherical k-means algorithm which is an iterative process that generates a sequence of partitions $\{\pi_l^{(t)}\}_{l=1}^k$, $t = 0, 1, ...$, such that

$$\mathcal{Q}\left(\{\pi_j^{(t+1)}\}_{j=1}^k\right) \geq \mathcal{Q}\left(\{\pi_j^{(t)}\}_{j=1}^k\right). \tag{2}$$

To emphasize the relationship between successive partitions we shall denote the "next" partition generated by k-means, $\{\pi_j^{(t+1)}\}_{j=1}^k$, by $\texttt{nextKM}\left(\{\pi_l^{(t)}\}_{l=1}^k\right)$. With the above notation, we now present the spherical k-means algorithm. Given a user supplied tolerance $\texttt{tol} > 0$ do the following:

1. Start with a partitioning $\{\pi_l^{(0)}\}_{l=1}^k$ and the concept vectors $\mathbf{c}_1^{(0)}, \mathbf{c}_2^{(0)}, \dots, \mathbf{c}_k^{(0)}$ associated with the partitioning. Set the index of iteration $t = 0$.
2. For each document vector $\mathbf{x} \in \mathbf{X}$ find the concept vector $\mathbf{c}_{l^*(\mathbf{x})}$ closest in cosine similarity to $\mathbf{x}$ (unless stated otherwise we break ties arbitrarily), i.e.,

   $$l^*(\mathbf{x}) = \arg\max_j \mathbf{x}^T \mathbf{c}_j^{(t)}.$$

   Next, compute the new partitioning $\{\pi_l^{(t+1)}\}_{l=1}^k = \texttt{nextKM}\left(\{\pi_l^{(t)}\}_{l=1}^{k(t)}\right)$ induced by the old concept vectors $\{\mathbf{c}_l^{(t)}\}_{l=1}^k$:

   $$\pi_l^{(t+1)} = \{\mathbf{x} \in \mathbf{X} \;:\; l^*(\mathbf{x}) = l\},\ 1 \leq l \leq k. \tag{3}$$
3. Compute the new concept vectors corresponding to the partitioning computed in (3):

   $$\mathbf{c}_l^{(t+1)} \;=\; \frac{\mathbf{s}(\pi_l^{(t+1)})}{\|\mathbf{s}(\pi_l^{(t+1)})\|}.$$
4. If $\mathcal{Q}\left(\texttt{nextKM}\left(\{\pi_l^{(t)}\}_{l=1}^{k(t)}\right)\right) - \mathcal{Q}\left(\{\pi_l^{(t)}\}_{l=1}^k\right)$ is greater than $\texttt{tol}$, increment t by 1 and go to step 2 above. Otherwise, stop.

As noted in (2), it can be shown that the above algorithm is a gradient-ascent scheme, i.e., the objective function value increases from one iteration to the next. For details, including a proof, see [4]. However, like any gradient-ascent scheme, the spherical k-means algorithm is prone to local maxima.

3 Inadequacy of k-means

We now present some scenarios in which spherical k-means can get stuck in a qualitatively poor local maximum.

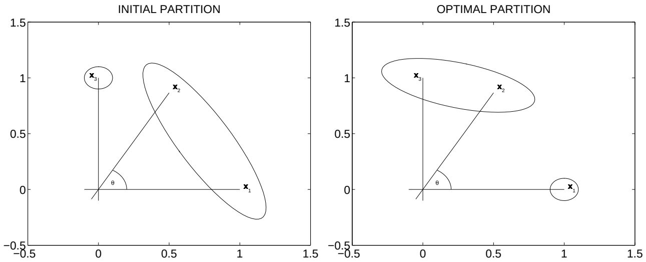
Example 3.1 *Consider the three unit vectors in $\mathbf{R}^2$:*

$$\mathbf{x}_1 = (1, 0)^T,\ \mathbf{x}_2 = (\cos\theta, \sin\theta)^T,\ \mathbf{x}_3 = (0, 1)^T.$$

Let the initial partition be $\pi_1^{(0)} = \{\mathbf{x}_1, \mathbf{x}_2\}$, $\pi_2^{(0)} = \{\mathbf{x}_3\}$. The value of the objective function for this partition is $\mathcal{Q}_0 = 2\cos\left(\theta/2\right) + 1$. When $0 \leq \theta \leq \pi/3$ the concept vector $\mathbf{c}_1 = \cos(\theta/2)$ of the cluster $\pi_1^{(0)}$ is closest in cosine similarity to vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. Hence, an application of spherical k-means does not change the partition(see figure below).

The partition $\pi_1^{(1)} = \{\mathbf{x}_1\}$, $\pi_2^{(1)} = \{\mathbf{x}_2, \mathbf{x}_3\}$, has the objective function value $\mathcal{Q}_1 = 2\cos\left(\pi/4 - \theta/2\right) + 1$. If $\theta = \pi/3$, then $\mathcal{Q}_0 = 2\cos\left(\pi/6\right) + 1 < 2\cos\left(\pi/12\right) + 1 = \mathcal{Q}_1$, and so the optimal partition is missed by k-means.

In Section 4, we show that a "first variation" iteration corrects the above problem and generates the optimal partition $\pi_1^{(1)} = \{\mathbf{x}_1\}$, $\pi_2^{(1)} = \{\mathbf{x}_2, \mathbf{x}_3\}$ starting from $\pi_1^{(0)}$ and $\pi_2^{(0)}$.



Example 3.2 *Consider the set of vectors arranged as columns of the $(k^2 + k) \times k$ matrix:*

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{k^2}] = \begin{bmatrix} \mathbf{K}_1 & \mathbf{K}_2 & \ldots & \mathbf{K}_k \\ \mathbf{I} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \ldots & \mathbf{0} \\ \ldots & \ldots & \ldots & \ldots \\ \mathbf{0} & \ldots & \ldots & \mathbf{I} \end{bmatrix},$$

where $\mathbf{I}$ is the $k \times k$ identity matrix and $\mathbf{K}_l$ is the $k \times k$ matrix with entries $\frac{1}{k}$ in the l^{th} row and 0's elsewhere. Note that all vectors $\mathbf{x}_i \in R^{k^2+k}$ have the same norm, $||\mathbf{x}_i|| = \sqrt{1 + 1/k^2}$, and they form k natural clusters with $\mathbf{x}_{ik+j}$ being placed in cluster $i + 1$.

The intra-cluster and inter-cluster dot products are respectively given by

1. $\mathbf{x}_i^T \mathbf{x}_j = \frac{1}{k^2}$, where $(l - 1)k < i < j \leq lk$ and $l = 1, \ldots, k$,
2. $\mathbf{x}_i^T \mathbf{x}_j = 0$, otherwise.

As mentioned above, the best clustering would have $\mathbf{x}_1, \ldots, \mathbf{x}_k$ in the first cluster, $\mathbf{x}_{k+1}, \ldots, \mathbf{x}_{2k}$ in the second cluster, and so on. For this optimal clustering, the cosine of any $\mathbf{x}_i$ with its own concept vector is $\frac{1+\frac{1}{k}}{\sqrt{1+\frac{1}{k^2}}\sqrt{k+1}} = \sqrt{\frac{k+1}{k^2+1}}$, and 0 with any other concept vector.

But suppose we initialize the k clusters as follows: cluster 1 consists of $\mathbf{x}_1$, $\mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k^2-k+1}$, cluster 2 consists of $\mathbf{x}_2, \mathbf{x}_{k+2}, \ldots, \mathbf{x}_{k^2-k+2}$, and generally cluster l consists of $\mathbf{x}_l, \mathbf{x}_{k+l}, \ldots, \mathbf{x}_{k^2-k+l}$, for $1 \leq l \leq k$. Then it can be seen that spherical k-means does not change this initial clustering at all. This is because the cosine of $\mathbf{x}_i$ with its own concept vector is $\frac{1+\frac{1}{k^2}}{\sqrt{1+\frac{1}{k^2}}\sqrt{k+\frac{1}{k}}} = \frac{1}{\sqrt{k}}$, and $\frac{1}{k^2\sqrt{1+\frac{1}{k^2}}\sqrt{k+\frac{1}{k}}} = \frac{1}{\sqrt{k^2+1}\sqrt{k}}$ with any other concept vector. The situation is similar with most other starting partitions. In a set of experiments for $k = 5$, we observed that all 100 runs of spherical k-means with random initializations stopped without a change in the initial clustering.

Example 3.3 *The above behavior is often seen in real-life applications. As a concrete example, we took a collection of 30 documents consisting of 10 documents each from the three distinct* classes *MEDLINE, CISI and CRAN (see Section 5 for more details). These 30 documents contain a total of 1073 words (after removing stop words). Due to this high-dimensionality, the cosine similarities between the 30 document vectors are quite low. The average cosine similarity between all documents is 0.025; however the average cosine between documents in the same class is 0.294 while the average inter-class cosine is 0.0068. Thus there is a clear separation between intra-class cosines and inter-class cosines and so it seems that a good clustering algorithm should be easily able to recover the original class structure.*

However, when we run spherical k-means on this data set, there is hardly any movement of document vectors between clusters irrespective of the starting partition — indeed we observed that 96% of 1000 different starting partitions resulted in no movement at all, i.e., k-means returned a final partition that was identical to the initial partition. This behavior is unusual; in contrast, for large data sets the first few iterations of spherical k-means typically lead to a lot of movement of data points between clusters [4, 3]. However, a closer look reveals the reasons for this failure. Consider a document vector $\mathbf{x}$, and consider an initial cluster π_i such that $\mathbf{x} \notin \pi_i$. Due to the small number of data points and the small average cosine similarity between documents, $\mathbf{x}^T \mathbf{c}_i$ turns out to be quite small in magnitude for an arbitrary initial partitioning. However, it is instructive to take a closer look at $\mathbf{x}$ and its own cluster. If $\mathbf{x} \in \pi_l$, the cosine similarity $\mathbf{x}^T \mathbf{c}_l$ may be broken down into two parts:

$$\mathbf{x}^T \mathbf{c}_l = \frac{1}{||\mathbf{s}(\pi_l)||} + \sum_{y \in \pi_l - \{\mathbf{x}\}} \frac{\mathbf{x}^T \mathbf{y}}{||\mathbf{s}(\pi_l)||}$$

where the first term has 1 in the numerator since $\mathbf{x}^T \mathbf{x} = 1$ (due to the contribution of $\mathbf{x}$ in $\mathbf{c}_l$). For data sets that are high-dimensional and contain a small number of points, this first term itself is typically much larger in magnitude than $\mathbf{x}^T \mathbf{c}_i$, $\mathbf{x} \notin \pi_i$. As a result, a spherical k-means iteration retains each document vector in its original cluster. Thus, in this case we start with an arbitrarily poor clustering and the k-means algorithm returns this poor clustering as the final output. We now see how to overcome this difficulty.

4 Refinement Algorithm

This section describes algorithms to "fix" the above problems.

4.1 First Variation

Definition 4.1 *A first variation of a partition $\{\pi_l\}_{l=1}^k$ is a partition $\{\pi_l'\}_{l=1}^k$ obtained by removing a single vector $\mathbf{x}$ from a cluster π_i of $\{\pi_l\}_{l=1}^k$ and assigning this vector to an existing cluster π_j of $\{\pi_l\}_{l=1}^k$.*

We denote the set of all first variation partitions of $\{\pi_l\}_{l=1}^k$ by $\mathcal{FV}\left(\{\pi_l\}_{l=1}^k\right)$. Among all the elements of this set, we seek to select the "steepest ascent" partition. The formal definition of this partition is given next.

Definition 4.2 *The partition* $\texttt{nextFV}\left(\{\pi_l\}_{l=1}^k\right)$ *is a first variation of* $\{\pi_l\}_{l=1}^k$ *so that for each first variation* $\{\pi_l'\}_{l=1}^k$,

$$\mathcal{Q}\left(\texttt{nextFV}\left(\{\pi_l\}_{l=1}^k\right)\right) \geq \mathcal{Q}\left(\{\pi_l'\}_{l=1}^k\right).$$

The proposed first variation algorithm generates a sequence of partitions $\{\pi_l^{(t)}\}_{l=1}^k$, $t \geq 0$, so that

$$\{\pi_l^{(t+1)}\}_{l=1}^k = \texttt{nextFV}\left(\{\pi_l^{(t)}\}_{l=1}^k\right),\ t = 0, 1, \ldots.$$

We now pause briefly to illustrate differences between first variation iterations and the spherical k-means iterations. To simplify the presentation we consider a two cluster partition $\{\mathbf{Z}, \mathbf{Y}\}$ where $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$, and $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$. Our goal is to examine whether a single vector, say $\mathbf{z}_n$, should be removed from $\mathbf{Z}$, and assigned to $\mathbf{Y}$. We denote the potential new clusters by $\mathbf{Z}^-$ and $\mathbf{Y}^+$, i.e., $\mathbf{Z}^- = \{\mathbf{z}_1, \ldots, \mathbf{z}_{n-1}\}$ and $\mathbf{Y}^+ = \{\mathbf{y}_1, \ldots, \mathbf{y}_m, \mathbf{z}_n\}$. Note that spherical k-means examines the quantity

$$\Delta_{kmeans} = \mathbf{z}_n^T \left[\mathbf{c}(\mathbf{Y}) - \mathbf{c}(\mathbf{Z})\right]. \tag{4}$$

If $\Delta_{kmeans} > 0$, then the spherical k-means algorithm moves $\mathbf{z}_n$ from $\mathbf{Z}$ to $\mathbf{Y}$. Otherwise $\mathbf{z}_n$ remains in $\mathbf{Z}$.

Unlike k-means, first variation computes

$$\Delta = \left[q\left(\mathbf{Z}^-\right) - q(\mathbf{Z})\right] + \left[q\left(\mathbf{Y}^+\right) - q(\mathbf{Y})\right],$$

where the quality q is as in (1). A straightforward computation shows that

$$\begin{aligned}\Delta = &\left[\sum_{i=1}^{n-1} \mathbf{z}_i^T \mathbf{c}(\mathbf{Z}^-) - \sum_{i=1}^{n-1} \mathbf{z}_i^T \mathbf{c}(\mathbf{Z}) - \mathbf{z}_n^T \mathbf{c}(\mathbf{Z})\right] + \\ &\left[\sum_{i=1}^{m} \mathbf{y}_i^T \mathbf{c}(\mathbf{Y}^+) + \mathbf{z}_n^T \mathbf{c}(\mathbf{Y}^+) - \sum_{i=1}^{m} \mathbf{y}_i^T \mathbf{c}(\mathbf{Y})\right] \\ = &\sum_{i=1}^{m} \mathbf{y}_i^T \left[\mathbf{c}(\mathbf{Y}^+) - \mathbf{c}(\mathbf{Y})\right] + \mathbf{z}_n^T \left[\mathbf{c}(\mathbf{Y}^+) - \mathbf{c}(\mathbf{Y})\right] + \\ &\sum_{i=1}^{n-1} \mathbf{z}_i^T \left[\mathbf{c}(\mathbf{Z}^-) - \mathbf{c}(\mathbf{Z})\right] + \Delta_{kmeans},\end{aligned} \tag{5}$$

where Δ_{kmeans} is defined in (4). By the Cauchy-Schwarz inequality, we have

$$\sum_{i=1}^{n-1} \mathbf{z}_i^T \mathbf{c}(\mathbf{Z}^-) \geq \sum_{i=1}^{n-1} \mathbf{z}_i^T \mathbf{c}(\mathbf{Z}), \qquad \text{and so}$$

$$\sum_{i=1}^{n-1} \mathbf{z}_i^T \left[\mathbf{c}(\mathbf{Z}^-) - \mathbf{c}(\mathbf{Z})\right] \geq 0.$$

For the same reason,

$$\sum_{i=1}^{m} \mathbf{y}_i^T \left[\mathbf{c}(\mathbf{Y}^+) - \mathbf{c}(\mathbf{Y})\right] + \mathbf{z}_n^T \left[\mathbf{c}(\mathbf{Y}^+) - \mathbf{c}(\mathbf{Y})\right] \geq 0.$$

These two inequalities along with (5) imply that:

$$\Delta \geq \Delta_{kmeans}.$$

The last inequality shows that even when $\Delta_{kmeans} \leq 0$ and cluster affiliation of $\mathbf{z}_n$ is not changed by spherical k-means, the quantity Δ may still be positive. Thus, while $\mathcal{Q}\left(\{\mathbf{Z}^-, \mathbf{Y}^+\}\right) > \mathcal{Q}\left(\{\mathbf{Z}, \mathbf{Y}\}\right)$, the partition $\{\mathbf{Z}^-, \mathbf{Y}^+\}$ will be missed by spherical k-means (see Example 3.1). We now turn to the magnitude of $\Delta - \Delta_{kmeans}$. From (5),

$$\begin{aligned}0 \leq \Delta - \Delta_{kmeans} \quad = \quad &\mathbf{s}(\mathbf{Z}^-)^T \left[\mathbf{c}(\mathbf{Z}^-) - \mathbf{c}(\mathbf{Z})\right] + \\ &\mathbf{s}(\mathbf{Y}^+)^T \left[\mathbf{c}(\mathbf{Y}^+) - \mathbf{c}(\mathbf{Y})\right].\end{aligned} \tag{6}$$

Since the vectors $\mathbf{s}(\mathbf{Z}^-)$ and $\mathbf{c}(\mathbf{Z}^-)$ are proportional, a large value of $\mathbf{c}(\mathbf{Z}^-) - \mathbf{c}(\mathbf{Z})$ results in a large dot product $\mathbf{s}(\mathbf{Z}^-)^T \left[\mathbf{c}(\mathbf{Z}^-) - \mathbf{c}(\mathbf{Z})\right]$. A similar argument holds for the second term on the right hand side of (6). Hence we can expect a "substantial" difference between Δ and Δ_{kmeans} when removing a vector from cluster $\mathbf{Z}$ and assigning it to cluster $\mathbf{Y}$ "significantly" changes locations of the corresponding concept vectors. This phenomenon is unlikely to happen when clusters are large. However, first variation iterations become effective in the case of small clusters (clusters of size 100 or less in our experience). The "spherical first variation" algorithm is formally presented below.

Given a user supplied tolerance $\texttt{tol} > 0$ do the following:

1. Start with a partitioning $\{\pi_l^{(0)}\}_{l=1}^k$. Set the index of iteration $t = 0$.
2. Generate $\{\pi_l^{(t+1)}\}_{i=1}^k = \texttt{nextFV}(\{\pi_l^{(t)}\}_{l=1}^k)$.
3. If $\mathcal{Q}\left(\texttt{nextFV}\left(\{\pi_l^{(t)}\}_{l=1}^k\right)\right) - \mathcal{Q}\left(\{\pi_l^{(t)}\}_{l=1}^k\right)$ is greater than $\texttt{tol}$, increment t by 1, and go to step 2. Otherwise, stop.

It is easy to see that a single "spherical first variation" iteration applied to the initial partition of Example 3.1 generates the "optimal partition".

We now address the time and memory complexity of first variation iterations. The computational bottleneck is in computing $q(\pi_i^{(t)} - \{\mathbf{x}\}) - q(\pi_i^{(t)})$ and $q(\pi_j^{(t)} \cup \{\mathbf{x}\}) - q(\pi_j^{(t)})$ for all the document vectors $\mathbf{x} \in \mathbf{X}$. Note that

$$\begin{aligned}&q\left(\pi_i^{(t)} - \{\mathbf{x}\}\right) - q\left(\pi_i^{(t)}\right) = \\ &\left(\|\mathbf{s}(\pi_i^{(t)})\|^2 - 2\|\mathbf{s}(\pi_i^{(t)})\|\mathbf{x}^T\mathbf{c}(\pi_i^{(t)}) + 1\right)^{1/2} - \|\mathbf{s}(\pi_i^{(t)})\|\end{aligned} \tag{7}$$

and

$$q\left(\pi_j^{(t)} \cup \{\mathbf{x}\}\right) - q\left(\pi_j^{(t)}\right) = \left(\|\mathbf{s}(\pi_j^{(t)})\|^2 + 2\|\mathbf{s}(\pi_j^{(t)})\|\mathbf{x}^T\mathbf{c}(\pi_j^{(t)}) + 1\right)^{1/2} - \|\mathbf{s}(\pi_j^{(t)})\|. \quad (8)$$

We remark that computation of the quantities $\|\mathbf{s}(\pi_l^{(t)})\|$, and $\mathbf{x}^T\mathbf{c}(\pi_l^{(t)})$, $\mathbf{x} \in \mathbf{X}$, $l = 1, \dots, k$ are needed for iterations of the spherical k-means algorithm as well. When these quantities are available, a first variation iteration can be completed in $O(d + k)$ amortized time, where d is the number of document vectors.

4.2 Kernighan-Lin Chains

A first variation iteration moves a single vector that maximally increases the objective function value. One way to enhance first variation is by expanding the local search to seek a chain of moves instead of just one move. Note that the first few moves in this chain might lead to a temporary decrease in objective function value, but overall we seek the chain that leads to a maximal increase. We implement this idea by following the well known Kernighan-Lin heuristic for graph partitioning [10].

The search for the optimal chain proceeds as follows. We generate a first variation move, record the improvement in objective function value, and then "mark" the moved vector so that it is not moved again in this chain. We repeat this step, say, f times where f is a user defined parameter. Finally, we search for that prefix of moves that leads to the greatest increase in objective function value. This idea is formally described below (the notation $\texttt{nextFV}\left(\{\pi_l\}_{l=1}^k, \mathbf{U}\right)$ restricts the search for the vector to be moved to the set of unmarked vectors $\mathbf{U}$).
Given a user supplied tolerance $\texttt{tol} > 0$ and a number f, do the following:

1. Start with a partitioning $\{\pi_l^{(0)}\}_{l=1}^k$. Set the index of iteration $t = 0$.
2. Set $j = 0, \mathbf{U} = \mathbf{X}$ and do the following f times:
   (a) Set $\{\pi_l^{(t+j+1)}\}_{l=1}^k = \texttt{nextFV}\left(\{\pi_l^{(t+j)}\}_{l=1}^k, \mathbf{U}\right)$. Mark the vector moved and delete it from $\mathbf{U}$.
   (b) Record $\mathcal{Q}\left(\texttt{nextFV}\left(\{\pi_l^{(t+j)}\}_{l=1}^k, \mathbf{U}\right)\right) - \mathcal{Q}\left(\{\pi_l^{(t+j)}\}_{l=1}^k\right)$ in `ObjChange[j]`. Increment j by 1.
3. Set `MaxChange` $= \max_i \sum_{j=0}^i$ `ObjChange[j]` and `MaxI` $= \arg\max_i \sum_{j=0}^i$ `ObjChange[j]`, $i = 0, 1..., f-1$.
4. If `MaxChange` $\geq$ `tol`, increment t by `MaxI+1`, and go to step 2. Otherwise, output the partition $\{\pi_l^{(t)}\}_{l=1}^k$ and `MaxI`, and then stop.

Given an initial partitioning $\{\pi_l\}_{l=1}^k$, `tol` and f, this KL-chain first variation algorithm outputs the partition $\texttt{nextKLFV}\left(\{\pi_l\}_{l=1}^k\right)$.

4.3 The Refinement Algorithm

The advantage of "spherical first variation" is that it computes an exact change in the value of the objective function; however these iterations lead to small increases in the objective function value. On the other hand, k-means iterations typically lead to larger increases. To achieve best results we combine these iterations. Our "ping-pong" refinement algorithm is a two step procedure — the first step runs spherical k-means; if the first step fails the second step runs the Kernighan-Lin heuristic outlined in the previous section. The proposed refinement algorithm is as follows.
Given a user supplied tolerance $\texttt{tol} > 0$ do the following:

1. Start with a partitioning $\{\pi_l^{(0)}\}_{l=1}^k$. Set the index of iteration $t = 0$.
2. Generate the partition $\texttt{nextKM}\left(\{\pi_l^{(t)}\}_{l=1}^k\right)$. If $\mathcal{Q}\left(\texttt{nextKM}\left(\{\pi_l^{(t)}\}_{l=1}^k\right)\right) - \mathcal{Q}\left(\{\pi_l^{(t)}\}_{l=1}^k\right)$ is greater than `tol`, set $\{\pi_l^{(t+1)}\}_{l=1}^k = \texttt{nextKM}\left(\{\pi_l^{(t)}\}_{l=1}^k\right)$, increment t by 1 and repeat step 2.
3. Generate the partition $\texttt{nextKLFV}\left(\{\pi_l^{(t)}\}_{l=1}^k\right)$. If $\mathcal{Q}\left(\texttt{nextKLFV}\left(\{\pi_l^{(t)}\}_{l=1}^k\right)\right) - \mathcal{Q}\left(\{\pi_j^{(t)}\}_{j=1}^k\right)$ is greater than `tol`, increment t by `MaxI+1`, set $\{\pi_l^{(t)}\}_{l=1}^k = \texttt{nextKLFV}\left(\{\pi_l^{(t)}\}_{l=1}^k\right)$, and go to step 2. Otherwise, stop.

We emphasize that most of the computations associated with step 3 above have already been performed in step 2, see (7) and (8). Hence the computational overhead of running a first variation iteration just after an iteration of spherical k-means is small. Note also that the above algorithm can do no worse than spherical k-means. Indeed, as we show below, in many cases the quality of clusters produced is much superior.

5 Experimental Results

In this section, we present experimental results which demonstrate that our proposed algorithm often qualitatively improves k-means clustering results. In all our experiments with spherical k-means, we tried several initialization schemes varying from random initial partitions to choosing the initial centroids as data vectors that are "maximally" far apart from each other [2].

For our first experiment we created the data set described in Example 3.2 setting $k = 5$. We observed that all 100 runs of spherical k-means with different initializations stopped without changing the initial partition. However, on applying our refinement algorithm to this initial partitioning and

Table 1. Confusion matrices for Example 3.2 ($k = 5$).

π_1	2	0	0	0	1
π_2	0	2	2	0	1
π_3	0	0	1	0	0
π_4	0	1	1	4	1
π_5	3	2	1	1	2

Obj. fun. value for spherical k-means partition:**10.8193**

π_1	5	0	0	0	0
π_2	0	5	0	0	0
π_3	0	0	5	0	0
π_4	0	0	0	5	0
π_5	0	0	0	0	5

Obj. fun. value for final partition:**12.0096**

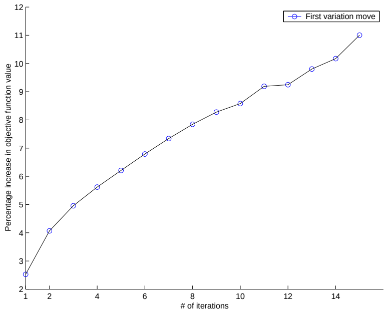


Figure 1. Increase in objective function value for Example 3.2 ($k = 5$, $f = 1$).

setting $f = 1$, i.e. only one first variation move is allowed in the KL chain, we were able to recover the optimal clustering in *all* 100 cases. For a particular run, Table 1 shows the confusion matrices for the initial and final partitions. Note that entry(i, j) in a confusion matrix gives the number of vectors in cluster i that belong to the true class j; thus, a diagonal confusion matrix is desirable. Figure 1 shows the percentage increase in objective function value as our algorithm progresses.

For experiments with real-life text data, we used the MEDLINE, CISI, and CRANFIELD collections (available from ftp://ftp.cs.cornell.edu/pub/smart). MEDLINE consists of 1033 abstracts from medical journals, CISI consists of 1460 abstracts from information retrieval papers, while CRANFIELD consists of 1400 abstracts from aerodynamical systems papers.

For our experiments we created three data sets of 30, 150, and 300 documents respectively (see Example 3.3). Each data set was created by an equal sampling of the three collections. After removing stopwords, the document vectors obtained are very high-dimensional and sparse. The dimensions for the 30, 150 and 300 document data sets are 1073, 3658 and 5577 respectively. In all runs the spherical k-means algorithm *did not change the initial partition*. We then applied our refinement algorithm to generate the final partitions (using $f = 1$). These results are summarized in Tables 2, 3 and 4 by the resulting confusion matrices. In addition, Figures 2, 3 and 4 plot the percentage increase in objective function values.

Table 2. Confusion matrices for 30 documents with 1073 words.

π_1	5	1	2
π_2	2	7	1
π_3	3	2	7

Obj. fun. value for spherical k-means partition:**11.0422**

π_1	9	1	0
π_2	0	9	0
π_3	1	0	10

Obj. fun. value for final partition:**11.9669**

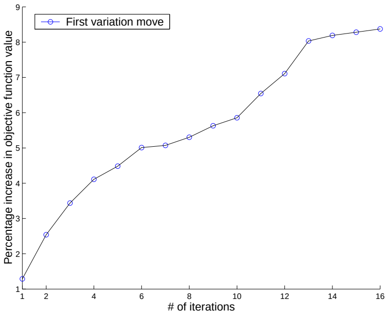


Figure 2. Increase in objective function value for 30 documents ($k = 3$, $f = 1$).

All the final partitions generated have almost diagonal confusion matrices and about $8\% - 25\%$ higher objective function values, which shows that our ping-pong strategy qualitatively improves spherical k-means clustering. The confusion matrices in Table 2 and 3 are almost optimal, while the final partition generated in Table 4 is qualitatively better than the spherical k-means result, but is not optimal. As testified by Figures 2, 3 and 4, the significant fraction of the clustering work is done by first variation iterations.

To improve the final partitioning of the 300 documents and show how KL-chain moves lead to a better local optimum, we applied our refinement algorithm with KL-chains to the partitioning of the 300 documents in Table 4 by setting $f = 30$. Figure 5 shows that during the first 18 moves there is a steady decrease in the objective function value. But from moves 22 to 30 the objective function value increases and becomes greater than the starting value. If we set $f < 22$, then the refinement algorithm will quit without changing the input partition. Figure 5 shows that the first KL-chain of 30 moves triggers a fresh spherical k-means iteration, which substantially increases the objective function value. In the second KL-chain, the maximum increase occurs at the 10th first variation and all the subsequent moves decrease the objective function value. The resulting confusion matrix shown in Table 5 is almost diagonal.

The following experiment shows that our algorithm is more beneficial as the number of clusters k increases. For this experiment we used a much larger collection — the 20-newsgroup data set consists of approximately 20,000 newsgroup postings collected from 20 different usenet newsgroups [12]. Some of the newsgroups are closely related to

Table 3. Confusion matrices for 150 documents with 3652 words.

π_1	25	10	17
π_2	5	18	10
π_3	20	22	23

Obj. fun. value for spherical k-means partition:**28.1934**

π_1	49	0	0
π_2	1	49	0
π_3	0	1	50

Obj. fun. value for final partition:**35.0355**

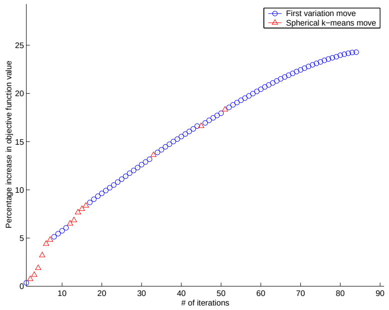


Figure 3. Increase in objective function value for 150 documents ($k = 3$, $f = 1$).

Table 4. Confusion matrices for 300 documents with 5577 words.

π_1	31	28	49
π_2	26	36	34
π_3	43	36	17

Obj. fun. value for spherical k-means partition:**52.7753**

π_1	99	0	0
π_2	1	100	19
π_3	0	0	81

Obj. fun. value for final partition:**58.7598**

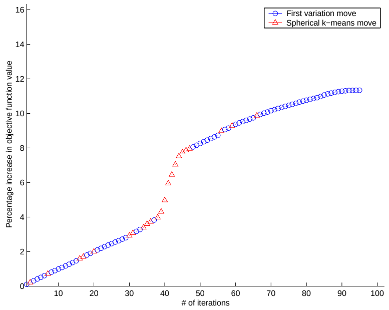


Figure 4. Increase in objective function value for 300 documents ($k = 3$, $f = 1$).

Table 5. Confusion matrices for the 300 documents. using KL-chains with $f = 30$. The objective function values for the *final* partition 59.5886.

π_1	99	0	0
π_2	1	100	19
π_3	0	0	81

Obj. fun. value for *initial* partition:**58.7598**

π_1	99	0	0
π_2	1	100	0
π_3	0	0	100

Obj. fun. value for *final* partition:**59.5886**

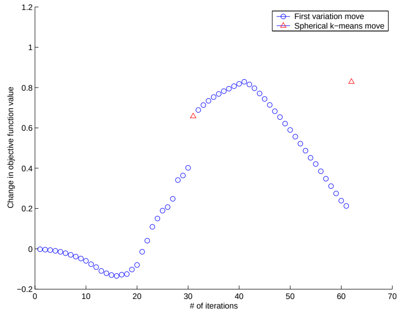


Figure 5. KL-chain moves and change in objective function value for 300 documents ($k = 3$, $f = 30$).

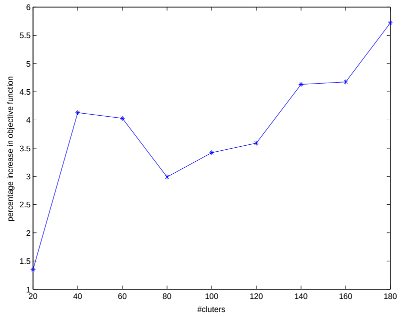


Figure 6. The percentage increase in objective function value of our refinement algorithm over spherical k-means on the 20-newsgroup data set ($f = 20$).

each other (e.g., comp.graphics, comp.os.ms-windows.misc and comp.windows.x), while others are unrelated (e.g., alt.atheism, rec.sport.baseball and sci.space). The headers for each of the messages were removed so that they do not bias clustering results, and 1169 duplicate messages were also taken out of the collection. After removing common stopwords and words that occur in less than 2 documents, the documents are represented by 59534-dimensional vectors, which are more than 99.87% sparse.

Figure 6 shows the percentage increase in objective function due to the refinement algorithm over spherical k-means as k increases. The main trend is that as k increases, i.e., when the average cluster size becomes smaller our refinement algorithm leads to greater improvement.

6 Related Work

The k-means algorithm has been well-studied and is one of the most widely used clustering methods [5]. Some of the important early work is due to Forgy[6] and MacQueen[13]. In the vector quantization literature, k-means clustering is also referred to as the Lloyd-Max algorithm[7]; see [8] for a comprehensive history of quantization and its relations to statistical clustering. Many variants of k-means exist; the version we presented in Section 2 is generally attributed to

Forgy (see [6, 13]) and is similar to the one given in [5, 10.4.3]; we call this “batch” k-means since the centroids are updated after a batch of points has been reassigned. Another version, which we call “incremental” k-means, randomly selects a single vector $\mathbf{x}$ whose re-assignment from a cluster π_i to a cluster π_j leads to a better objective function value (see [5, 10.8] where this incremental algorithm is referred to as “Basic Iterative Minimum-Squared-Error Clustering”). Incremental k-means is similar to our first-variation iterations. The ISODATA algorithm introduces an additional step in each k-means iteration, in which the number of clusters is adjusted[1].

However, as we have found, neither the batch nor the incremental version is satisfactory for our purposes. As shown in Section 3, batch k-means can give poor results in high-dimensional settings. On the other hand, the incremental version can be computationally expensive. Our main contribution in the paper is the “ping-pong” strategy which exploits the strong points of both batch and incremental k-means.

Other clustering algorithms use “medoids” instead of centroids for clustering, for example, the PAM clustering algorithm swaps a single medoid with a non-selected object as long as the swap results in an improvement of the quality of the clustering (see [9, 14]).

7 Conclusions and Future Research

In this paper, we have presented a refinement algorithm that uses local search after completion of spherical k-means iterations. The resulting improvements in the quality of clustering can be substantial, especially when the data is highly dimensional and sparse and when clusters contain a small number of data vectors. Note that while many implementations of k-means can return empty clusters, our refinement strategy guarantees that all k clusters will be non-empty.

Future enhancement of our algorithm will allow variable number of clusters at each iteration [11]. To accomplish this we plan to modify the objective function as follows:

$$\mathcal{Q}_\omega\left(k, \{\pi_j\}_{j=1}^k\right) = \sum_{j=1}^k \left[\sum_{\mathbf{x}\in\pi_j} \mathbf{x}^T\mathbf{c}_j\right] + f(k)\omega$$

where ω is a scalar parameter and $f(k)$ is an increasing function of k. When $\omega = 0$ the trivial partition maximizes the objective function. A negative ω imposes a penalty on the number of clusters, and prevents the trivial partition from becoming the optimal one.

We plan to push this idea further. Instead of keeping ω constant, we plan to select ω at each iteration. The selection will be based on the current partition, and the parameter ω will then become a feedback, and the iterative process can considered to be a discrete control system (see e.g. [17]). We also plan to explore the use of MDL and MML principles to decide on the “right” number of clusters [15].

References

[1] G. Ball and D. Hall. ISODATA: a novel method of data analysis and pattern classification. Technical report, Stanford Research Institute, Menlo Park, CA, 1965.

[2] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *KDD'03*. AAAI Press, 1998.

[3] I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In *Data Mining for Scientific and Engineering Applications*, pages 357–381. Kluwer Academic Publishers, 2001.

[4] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, January 2001.

[5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2000.

[6] E. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21(3):768, 1965.

[7] A. Gersho and R. M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, 1992.

[8] R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Trans. Inform. Theory*, 44(6):1–63, 1998.

[9] L. Kaufman and P. Rousseeuw. *Finding Groups in Data*. Wiley, New York, 1990.

[10] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970.

[11] J. Kogan. Means clustering for text data. In M.W.Berry, editor, *Workshop on Text Mining at the 1st SIAM International Conference on Data Mining*, pages 47–54, 2001.

[12] K. Lang. News Weeder: Learning to filter netnews. In *Proc. 12th Int'l Conf. Machine Learning*, pages 331–339, San Francisco, 1995.

[13] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Math., Stat. and Prob.*, pages 281–296, 1967.

[14] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proc. of the 20th Int'l Conf. on Very Large Data Bases (VLDB)*, pages 144–155, Santiago, Chile, 1994.

[15] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. Series in Computer Science -Vol. 15. World Scientific, Singapore, 1989.

[16] G. Salton and M. J. McGill. *Introduction to Modern Retrieval*. McGraw-Hill Book Company, 1983.

[17] E. D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Springer, New York, second edition, 1998.

[18] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *AAAI 2000 Workshop on AI for Web Search*, pages 58–64, July 2000.