Refining clusters in high dimensional text data

Inderjit S. Dhillon [*], Yuqiang Guan [†] and J. Kogan [‡]

Mar 1, 2002

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 for small clusters, say 25-30 documents per cluster, where it tends to get stuck at a local maximum far away from the optimal. In this paper, we present the *first-variation* principle that refines a given clustering by incrementally moving data points between clusters, thus achieving a higher objective function value. Combining first-variation with spherical k-means yields a powerful *ping-pong* strategy that often qualitatively improves k-means clustering. We present several experimental results to show that our proposed method works well in clustering high-dimensional and sparse text data.

keywords: clustering, high-dimensional, k-means, refinement algorithm, first variation.

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* [17]. 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, 2, 18]. The classical version of k-means uses Euclidean distance, however this distance measure is often inappropriate for its application to clustering a collection of documents [21]. 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 [17]. The k-means algorithm can be adapted to use the cosine similarity metric, see [16], 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 size of desired clusters is an important requirement for a clustering solution. From a large corpus where the number of documents may range from 100,000 to a few million, to small document collections, such as clustering web search results where the typical number of documents is 100-200, the end user often wants to see small clusters of relevant documents. In addition, hierarchical clustering of large collections often leads to small document clusters deep down in the tree hierarchy.

[*]Department of Computer Sciences, University of Texas, Austin, TX 78712-1188, USA. Email: inderjit@cs.utexas.edu, URL: http://www.cs.utexas.edu/users/inderjit, This research was supported by NSF Grant No. ACI-0093404

[†]Department of Computer Sciences, University of Texas, Austin, TX 78712-1188, USA. Email: yguan@cs.utexas.edu, URL: http://www.cs.utexas.edu/users/yguan, this research was supported by NSF Grant No. ACI-0093404

[‡]Department of Mathematics and Statistics, University of Maryland Baltimore County, Baltimore, MD 21228, USA. Email: kogan@math.umbc.edu, URL: http://www.math.umbc.edu/Faculty/kogan.html

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, 21, 3]. However, as shown 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 for refining the clusters produced by the spherical k-means algorithm. 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. Multiple iterations of first-variation allow an escape from local maximum, so that fresh iterations of spherical k-means can be applied to further increase the objective function. This ping-pong strategy yields a powerful refinement algorithm which often qualitatively improves k-means clustering. Note that our refinement algorithm always improves upon the input clustering in terms of the objective function value. We present several experimental results to validate these claims.

Many variants of the k-means algorithm, such as "batch" and "incremental" versions have been proposed in the literature, see Section 6 for a discussion. The main contribution of our paper is our ping-pong strategy that alternates between "batch" k-means and first-variation iterations, thereby harnessing the power of both in terms of improved 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 in Section 4.1, we present our proposed refinement algorithm that ping-pongs between first variation and spherical k-means. Experimental results in Section 5 show that our refinement 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 $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d\}$ denote the set of non-negative document vectors, where each $\mathbf{x}_i \in R^w$ and $\|\mathbf{x}_i\|_2 = 1$, i.e., each $\mathbf{x}_i$ lies on the unit sphere. A clustering of the document collection is its partitioning into the disjoint subsets $\pi_1, \pi_2, ..., \pi_k$, i.e.,

$$\bigcup_{j=1}^{k} \pi_j = \mathbf{X} \qquad \text{and} \qquad \pi_j \cap \pi_l = \phi, \;\; j \neq l.$$

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 π is the normalized expectation of π. We define the "quality" or "coherence" of a non empty 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

$$\left\{\pi_l^{(0)}\right\}_{l=1}^{k}, \left\{\pi_l^{(1)}\right\}_{l=1}^{k}, \ldots, \left\{\pi_l^{(t)}\right\}_{l=1}^{k}, \ldots \text{ with } \mathcal{Q}\left(\left\{\pi_j^{(t+1)}\right\}_{j=1}^{k}\right) \geq \mathcal{Q}\left(\left\{\pi_j^{(t)}\right\}_{j=1}^{k}\right). \tag{2}$$

To emphasize the relationship between the partitions $\left\{\pi_l^{(t)}\right\}_{l=1}^{k}$ and $\left\{\pi_j^{(t+1)}\right\}_{j=1}^{k}$ we shall denote $\left\{\pi_j^{(t+1)}\right\}_{j=1}^{k}$ by $\texttt{nextKM}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right)$. For a partition $\left\{\pi_l^{(t)}\right\}_{l=1}^{k}$ with concept vectors $\mathbf{c}_l^{(t)} = \mathbf{c}\left(\pi_l^{(t)}\right)$, and a document vector $\mathbf{x} \in \pi_i^{(t)}$ we denote by $\texttt{present}(t, \mathbf{x})$ and $\texttt{max}(t, \mathbf{x})$ two special indices defined as follows:

$$\texttt{present}(t, \mathbf{x}) = i, \quad \texttt{max}(t, \mathbf{x}) = \arg\max_l \mathbf{x}^T \mathbf{c}_l^{(t)}.$$

When there is no ambiguity we shall suppress the iteration parameter t and denote the indices just by $\texttt{present}(\mathbf{x})$ and $\texttt{max}(\mathbf{x})$. With the above notation, we are ready to present the spherical k-means algorithm:

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

1. Start with a partitioning $\left\{\pi_l^{(0)}\right\}_{l=1}^{k}$ and the concept vectors $\mathbf{c}_1^{(0)}, \mathbf{c}_2^{(0)}, \ldots, \mathbf{c}_k^{(0)}$ 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 $\mathbf{c}_{\texttt{max}(\mathbf{x})}$ closest in cosine similarity to $\mathbf{x}$ (unless stated otherwise we break ties arbitrarily). Next, compute the new partitioning $\left\{\pi_l^{(t+1)}\right\}_{l=1}^{k} = \texttt{nextKM}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right)$ induced by the old concept vectors $\left\{\mathbf{c}_l^{(t)}\right\}_{l=1}^{k}$:
$$\pi_l^{(t+1)} = \left\{\mathbf{x} \in \mathbf{X} \ : \ l = \texttt{max}(\mathbf{x})\right\}, \ 1 \leq l \leq k. \tag{3}$$
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 $\left[\mathcal{Q}\left(\texttt{nextKM}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right)\right) - \mathcal{Q}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right) > \texttt{tol}\right]$, increment t by 1, and go to step 2 above.
5. Stop.

As noted in (2), it can be shown that the above algorithm is a gradient-ascent scheme, i.e., the objective function value does not decrease from one iteration to the next. See [4] for details. Like any other gradient-ascent scheme, the spherical k-means algorithm is prone to local maxima.

3 Inadequacy of k-means

In this section, we present some scenarios in which the spherical k-means algorithm 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 $\mathcal{Q}_0 = 2\cos\dfrac{\theta}{2} + 1$. *When* $0 \leq \theta \leq \dfrac{\pi}{3}$ *the concept vector* $\mathbf{c}_1 = \cos\dfrac{\theta}{2}$ *of the cluster* $\pi_1^{(0)}$ *is closest in cosine similarity to vectors* $\mathbf{x}_1$ *and* $\mathbf{x}_2$*. Hence, an application of the spherical k-means does not change the partition.*

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

As we will see later in Section 4, a "spherical first variation" iteration 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* $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{k^2}\}$ *such that*

$$\mathbf{x}_{ik+j} = \frac{1}{k}\mathbf{e}_{i+1} + \mathbf{e}_{(i+1)k+j}, \quad 0 \le i \le k-1,\ 1 \le j \le k,$$

where $\mathbf{e}_l$ *is the vector with* 1 *in its* l^{th}*-location and* 0*'s elsewhere. Note that these vectors have dimensionality* $k^2 + k$*, and form* k *natural clusters with* $\mathbf{x}_{ik+j}$ *being in the* $(i+1)$*-st cluster. The structure of these data vectors can be seen more clearly by arranging them as columns of a matrix:*

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

where $\mathbf{K}_l$ *is the* $k \times k$ *matrix with entries* $\frac{1}{k}$ *in the* l^{th} *row and* 0*'s elsewhere, and* $\mathbf{I}$ *is the* $k \times k$ *identity matrix.*

Note that all vectors have the same norm, $||\mathbf{x}_i||_2 = \sqrt{1 + \frac{1}{k^2}}$, and 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 $(l-1)k + 1 \le i < j \le lk$ and $l = 1, \ldots, k$,
2. $\mathbf{x}_i^T \mathbf{x}_j = 0$, otherwise.

As mentioned above if we want to cluster the k^2 vectors into k clusters, the best clustering would have $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ in the first cluster, the vectors $\mathbf{x}_{k+1}, \mathbf{x}_{k+2}, \ldots, \mathbf{x}_{2k}$ 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 if we happen to 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 $1 \le l \le k$. Then it can be seen that spherical k-means does not move any vectors from one cluster to another. This is because the cosine of any $\mathbf{x}_i$ 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 $\frac{\frac{1}{k^2}}{\sqrt{1+\frac{1}{k^2}}\sqrt{k+\frac{1}{k}}} = \frac{1}{\sqrt{k^2+1}\sqrt{k}}$ with any other concept vector. Therefore spherical k-means gets stuck at this initial clustering. The situation is similar with most other starting partitions. Indeed, in a set of experiments for $k = 5$, we observed that all 100 runs of spherical k-means with different initializations stopped without a change in the initial clustering.

Example 3.3 *The above behavior is often seen in real-life with document collections. 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; 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.*

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, consider $\mathbf{x}$ and its own cluster π_l, $l = \texttt{present}(\mathbf{x})$. 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 we use the fact that $\mathbf{x}^T\mathbf{x} = 1$. Note that the first term above is due to the contribution of $\mathbf{x}$ in $\mathbf{c}_l$. For small high-dimensional data sets and an arbitrary initial partitioning, 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, one could 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 First Variations and Refinement Algorithm

In this section, we describe algorithms that "fix" the above examples.

Definition 4.1 *A first variation of a partition* $\{\pi_l\}_{l=1}^k$ *is a partition* $\{\pi'_l\}_{l=1}^k$ *obtained from* $\{\pi_l\}_{l=1}^k$ *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* $\{\pi_l\}_{l=1}^k$.

Each first variation of a partition $\{\pi_l\}_{l=1}^k$ is associated with a vector $\mathbf{x}$. Note that there are k first variations associated with a vector $\mathbf{x}$ that belongs to a cluster π_i. 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 $\mathcal{FV}\left(\{\pi_l\}_{l=1}^k\right)$ we seek to select the "steepest ascent" first variation 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* $\{\pi'_l\}_{l=1}^k$ *one has*

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

The proposed first variation algorithm starts with an initial partition $\left\{\pi_l^{(0)}\right\}_{l=1}^k$ and generates a sequence of partitions

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

so that

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

We now pause briefly to illustrate differences between first variation iterations and iterations of the spherical k–means algorithm. 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 $\mathbf{Y}^+$, that is

$$\mathbf{Z}^- = \{\mathbf{z}_1, \ldots, \mathbf{z}_{n-1}\}, \text{ and } \mathbf{Y}^+ = \{\mathbf{y}_1, \ldots, \mathbf{y}_m, \mathbf{z}_n\}.$$

Note that the spherical k–means algorithm examines the quantity

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

If $\Delta_k > 0$, 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 the spherical k–means algorithm, the spherical first variation algorithm 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

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

Thus

$$\Delta = \sum_{i=1}^{n-1} \mathbf{z}_i^T \left[\mathbf{c}(\mathbf{Z}^-) - \mathbf{c}(\mathbf{Z})\right] + \left\{\sum_{i=1}^{m} \mathbf{y}_i^T \left[\mathbf{c}(\mathbf{Y}^+) - \mathbf{c}(\mathbf{Y})\right] + \mathbf{z}_n^T \left[\mathbf{c}(\mathbf{Y}^+) - \mathbf{c}(\mathbf{Y})\right]\right\} + \Delta_k. \tag{4}$$

By the Cauchy-Schwarz inequality, we have $\sum_{i=1}^{n-1} \mathbf{z}_i^T \mathbf{c}(\mathbf{Z}^-) \geq \sum_{i=1}^{n-1} \mathbf{z}_i^T \mathbf{c}(\mathbf{Z})$, and, therefore,

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

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 (4) imply that:

$$\Delta \geq \Delta_k.$$

The last inequality shows that even when $\Delta_k \leq 0$ and cluster affiliation of $\mathbf{z}_n$ is not changed by the spherical k–means algorithm, 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 the spherical k–means algorithm (see Example 3.1).
We now turn to the magnitude of $\Delta - \Delta_k$. From (4),

$$0 \leq \Delta - \Delta_k = \mathbf{s}(\mathbf{Z}^-)^T \left[\mathbf{c}(\mathbf{Z}^-) - \mathbf{c}(\mathbf{Z})\right] + \mathbf{s}(\mathbf{Y}^+)^T \left[\mathbf{c}(\mathbf{Y}^+) - \mathbf{c}(\mathbf{Y})\right]. \tag{5}$$

Since the vectors $\mathbf{s}(\mathbf{Z}^-)$ and $\mathbf{c}(\mathbf{Z}^-)$ are proportional, the larger the difference $\mathbf{c}(\mathbf{Z}^-) - \mathbf{c}(\mathbf{Z})$ is the larger the dot product $\mathbf{s}(\mathbf{Z}^-)^T \left[\mathbf{c}(\mathbf{Z}^-) - \mathbf{c}(\mathbf{Z})\right]$ becomes. A similar argument holds for the second term on the right hand side of (5). Hence we can expect a "substantial" difference between Δ and Δ_k 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 the clusters are large. However, first variation iterations become efficient 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 `tol` > 0 do the following:

1. Start with a partitioning $\left\{\pi_l^{(0)}\right\}_{l=1}^{k(0)}$. Set the index of iteration $t = 0$.

2. Generate $\texttt{nextFV}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right)$.

3. If $\left[\mathcal{Q}\left(\texttt{nextFV}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right)\right) - \mathcal{Q}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right) \geq \texttt{tol}\right]$, set $\left\{\pi_l^{(t+1)}\right\}_{l=1}^{k} = \texttt{nextFV}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right)$, increment t by 1, and go to step 2.

4. 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 computational complexity associated with first variation iterations. The time and memory complexity of first variation iterations are basically the same as those of the spherical k–means algorithm. The computational bottleneck of first variation iterations is in the computation of

$$q\left(\pi_i^{(t)} - \{\mathbf{x}\}\right) - q\left(\pi_i^{(t)}\right) \text{ and } q\left(\pi_j^{(t)} \cup \{\mathbf{x}\}\right) - q\left(\pi_j^{(t)}\right)$$

for all the document vectors $\mathbf{x} \in \mathbf{X}$. Note that

$$q\left(\pi_i^{(t)} - \{\mathbf{x}\}\right) - q\left(\pi_i^{(t)}\right) = \sqrt{\left\|\mathbf{s}\left(\pi_i^{(t)}\right)\right\|^2 - 2\left\|\mathbf{s}\left(\pi_i^{(t)}\right)\right\| \mathbf{x}^T\mathbf{c}\left(\pi_i^{(t)}\right) + 1} \; - \; \left\|\mathbf{s}\left(\pi_i^{(t)}\right)\right\| \tag{6}$$

and

$$q\left(\pi_j^{(t)} \cup \{\mathbf{x}\}\right) - q\left(\pi_j^{(t)}\right) = \sqrt{\left\|\mathbf{s}\left(\pi_j^{(t)}\right)\right\|^2 + 2\left\|\mathbf{s}\left(\pi_j^{(t)}\right)\right\| \mathbf{x}^T\mathbf{c}\left(\pi_j^{(t)}\right) + 1} \; - \; \left\|\mathbf{s}\left(\pi_j^{(t)}\right)\right\|. \tag{7}$$

We remark that computation of the quantities $\left\|\mathbf{s}\left(\pi_l^{(t)}\right)\right\|$, and $\mathbf{x}^T\mathbf{c}\left(\pi_l^{(t)}\right)$, $\mathbf{x} \in \mathbf{X}$, $l = 1, \ldots, k$ are needed for iterations of the spherical k–means algorithm as well.

4.1 The Refinement Algorithm

While "spherical first variation" computes an exact change in the values of the objective function, iterations of the algorithm 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 a spherical k–means iteration; if the first step fails the second step runs a spherical first variation iteration. The proposed refinement algorithm is as follows.

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

1. Start with a partitioning $\left\{\pi_l^{(0)}\right\}_{l=1}^{k}$. Set the index of iteration $t = 0$.

2. Generate the partition $\texttt{nextKM}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right)$. If $\left[\mathcal{Q}\left(\texttt{nextKM}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right)\right) - \mathcal{Q}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right) \geq \texttt{tol}\right]$, set $\left\{\pi_l^{(t+1)}\right\}_{l=1}^{k} = \texttt{nextKM}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right)$, increment t by 1 and go to step 2.

3. Generate the partition $\texttt{nextFV}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right)$. If $\left[\mathcal{Q}\left(\texttt{nextFV}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right)\right) - \mathcal{Q}\left(\left\{\pi_j^{(t)}\right\}_{j=1}^{k}\right) \geq \texttt{tol}\right]$, set $\left\{\pi_l^{(t+1)}\right\}_{l=1}^{k} = \texttt{nextFV}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right)$, increment t by 1, and go to step 2.

4. Stop.

Table 1: Confusion matrices for Example 3.2 ($k = 5$). The objective function values for the *initial* partition and *final* partition are **10.8193** and **12.0096** respectively.

cluster 0	2	0	0	0	1
cluster 1	0	2	2	0	1
cluster 2	0	0	1	0	0
cluster 3	0	1	1	4	1
cluster 4	3	2	1	1	2

initial partition

cluster 0	5	0	0	0	0
cluster 1	0	5	0	0	0
cluster 2	0	0	5	0	0
cluster 3	0	0	0	5	0
cluster 4	0	0	0	0	5

final partition

Figure 1: Objective function increase for Example 3.2 ($k = 5$)

We emphasize that most of the computations associated with step 3 above have already been performed in step 2, see (6) and (7). Hence the computational price of running a first variation iteration just after an iteration of the spherical k–means algorithm is negligible. 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 show that our refinement strategy 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.

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, 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 partition and the final partition. 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 as the refinement algorithm progresses.

For real-life experiments, 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. For example the dimension for the 30 document data set is 1073. We then generated initial partitions for each data set. In all cases the spherical k-means algorithm did not change the initial partition. We then applied our refinement algorithm to gen-

Table 2: Confusion matrices for 30 documents with 1073 words. The objective function values for the *initial* partition and *final* partition are **11.0422** and **11.9669** respectively.

cluster 0	5	1	2
cluster 1	2	7	1
cluster 2	3	2	7

initial partition

cluster 0	9	1	0
cluster 1	0	9	0
cluster 2	1	0	10

final partition

Figure 2: Objective function increase for 30 documents

Table 3: Confusion matrices for 150 documents with 3652 words. The objective function values for the *initial* partition and *final* partition are **28.8772** and **35.0355** respectively.

cluster 0	25	10	17
cluster 1	5	18	10
cluster 2	20	22	23

initial partition

cluster 0	49	0	0
cluster 1	1	49	0
cluster 2	0	1	50

final partition

Figure 3: Objective function increase for 150 documents

Table 4: Confusion matrices for 300 documents with 5577 words. The objective function values for the *initial* partition and *final* partition are **52.694** and **58.3033** respectively.

cluster 0	52	10	26
cluster 1	13	55	37
cluster 2	35	35	37

initial partition

cluster 0	99	0	0
cluster 1	0	77	0
cluster 2	1	23	100

final partition

Figure 4: Objective function increase for 300 documents

erate the final partitions. These results are summarized in Tables 2, 3 and 4 by the confusion matrices of the initial and final partitions. In addition, Figures 2, 3 and 4 plot the percentage increase in the objective function values.

All the final partitions generated have almost diagonal confusion matrices and about 10% 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. In Section 7, we address future work to further improve on our algorithm.

6 Related Work

The k-means algorithm has been well-studied and is one of the most widely used clustering methods [5, 10]. Some of the important early work is due to Forgy[6] and MacQueen[14]. In the vector quantization literature, k-means clustering is also referred to as the Lloyd-Max algorithm[7]; see [9] 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, 14]) 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 value of the objective function (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, 8].

Of course, the idea of using a "variation" in order to improve the current value of a function is very old and can be traced to the 1629 work of Fermat[20].

However, as we have found, neither the batch version 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 has a serious computational drawback since it requires an update of cluster centroids with *each* move in the algorithm. 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 [11]). This idea is further pushed forward in [15] and [22].

7 Conclusion and Future Research

The paper introduces two procedures for refining k-means clustering. The first one, the spherical first variation, is a modification of spherical k-means which leads to better clustering results at the expense of execution speed. The improvement in clustering results due to first variations becomes significant in the case of small clusters. This is especially helpful when clustering results are to be presented to a user expecting small clusters containing relevant data. The second algorithm combines spherical k-means and spherical first variation in a "ping-pong" manner. The combination enjoys the speed of spherical k-means and the precision of spherical first variation. The computational complexity of the "ping-pong" combination is practically the same as that of the spherical k-means algorithm.

Most of the clustering methods suffer from the defect that they can never repair what was done in previous steps. Indeed whatever a divisive algorithm has split up can not be reunited. Once an agglomerative algorithm has joined two objects, they can not be separated in the future (see e.g. [11]). Like the k-means clustering algorithm the "ping-pong" algorithm introduced in this paper may decrease the number of clusters, but generates no new clusters.

While computationally efficient the "ping-pong" algorithm suffers from some drawbacks. The algorithm is unable to predict the number of clusters in the "right" partitioning of the data. The initial choice of k centroids, which may be crucial for clustering, remains anybody's guess. The guess does not always reflect the data geometry hidden in a high dimensional vector space.

Our future enhancement of the algorithm focuses on the following main directions:

Direction 1: Variable number of clusters at each iteration.

Following the spirit of [13] we shall allow the number of clusters to change from one iteration to the next. To accomplish this we modify the objective function as follows:

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

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

We plan to push this idea further. Rather that keeping ω constant, we plan to select the value of 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. [19]).

Direction 2: Enhancement of the first variation algorithm.

While the first variation component of the algorithm leads to a small change in the objective function, iterations of first variation push the objective function away from the local maximum. We intend to enhance the first variation algorithm following the ideas of Kernighan and Lin [12] by allowing chains of moves to increase the value of the objective function.

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] L. Bottou and Y. Bengio. Convergence properties of the K-means algorithms. In G. Tesauro and D. Touretzky, editors, *Advances in Neural Information Processing Systems 7*, pages 585–592. The MIT Press, Cambridge, MA, 1995.

[3] I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In R. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar, and R. Namburu, editors, *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, 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. Gnanadesikan. *Methods for Statistical Data Analysis of Multivariate Observations*. John Wiley, NY, second edition, 1997.

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

[10] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

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

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

[13] J. Kogan. Means clustering for text data. In M.W.Berry, editor, *Proceedings of the Workshop on Text Mining at the First SIAM International Conference on Data Mining*, pages 57–54, 2001.

[14] 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.

[15] 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.

[16] E. Rasmussen. Clustering algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 419–442. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

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

[18] S. Z. Selim and M. A. Ismail. K-means-type algorithms: a generalized convergence theorem and characterization of local optimality. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:81–87, 1984.

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

[20] J. Stillwell. *Mathematics and Its History*. Springer-Verlag, 1989.

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

[22] A. K. H. Tung, J. Han, L. V. S. Lakshmanan, and R. T. Ng. Constraint-based clustering in large databases. In *Proc. 2001 Int. Conf. on Database Theory (ICDT'01)*, 2001.