Refining clusters in high dimensional text data

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

Jan 7, 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.

[*]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

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.

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

$$q\left(\pi\right) = \sum_{\mathbf{x}\in\pi}\mathbf{x}^T\mathbf{c}(\pi) = \|\mathbf{s}(\pi)\|. \tag{2.1}$$

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

$$\left\{\pi_l^{(0)}\right\}_{l=1}^k, \left\{\pi_l^{(1)}\right\}_{l=1}^k, \ldots, \left\{\pi_l^{(t)}\right\}_{l=1}^k, \ldots \text{ with } \mathcal{Q}\left(\left\{\pi_j^{(t+1)}\right\}_{j=1}^k\right) \geq \mathcal{Q}\left(\left\{\pi_j^{(t)}\right\}_{j=1}^k\right). \tag{2.2}$$

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 $\texttt{nextKM}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k(t)}\right)$. 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 $\mathbf{c}_{\mathtt{max}(\mathbf{x})}$ closest in cosine similarity to $\mathbf{x}$ (unless stated otherwise we break ties arbitrarily). Next, compute the new partitioning $\left\{\pi_l^{(t+1)}\right\}_{l=1}^{k} = \mathtt{nextKM}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k(t)}\right)$ induced by the old concept vectors $\left\{\mathbf{c}_l^{(t)}\right\}_{l=1}^{k}$:
$$\pi_l^{(t+1)} = \{\mathbf{x} \in \mathbf{X} \ : \ l = \mathtt{max}(\mathbf{x})\}\,, \ 1 \leq l \leq k. \qquad (2.3)$$

3. Compute the new concept vectors corresponding to the partitioning computed in (2.3):
$$\mathbf{c}_l^{(t+1)} \ = \ \frac{\mathbf{s}(\pi_l^{(t+1)})}{\|\mathbf{s}(\pi_l^{(t+1)})\|}.$$

4. If $\left[\mathcal{Q}\left(\mathtt{nextKM}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k(t)}\right)\right) - \mathcal{Q}\left(\left\{\pi_l^{(t)}\right\}_{l=1}^{k}\right) > \mathtt{tol}\right]$

   increment t by 1

   go to step 2 above.

5. Stop.

As noted in (2.2), 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 $\mathbf{R}^2$:*
$$\mathbf{x}_1 = (1,0)^T, \ \mathbf{x}_2 = (\cos\theta, \sin\theta)^T, \ \mathbf{x}_3 = (0,1)^T.$$