
Information Retrieval, 8, 331–349, 2005

Data Driven Similarity Measures for *k*-Means Like Clustering Algorithms*

JACOB KOGAN kogan@umbc.edu
Department of Mathematics and Statistics, UMBC, Baltimore, MD 21250

MARC TEBOULLE teboulle@post.tau.ac.il
School of Mathematical Sciences, Tel-Aviv University, Tel-Aviv, Israel

CHARLES NICHOLAS nicholas@umbc.edu
Department of Computer Science and Electrical Engineering, UMBC, Baltimore, MD 21250

Abstract. We present an optimization approach that generates k-means like clustering algorithms. The batch k-means and the incremental k-means are two well known versions of the classical k-means clustering algorithm (Duda et al. 2000). To benefit from the speed of the batch version and the accuracy of the incremental version we combine the two in a "ping–pong" fashion. We use a distance-like function that combines the squared Euclidean distance with relative entropy. In the extreme cases our algorithm recovers the classical k-means clustering algorithm and generalizes the Divisive Information Theoretic clustering algorithm recently reported independently by Berkhin and Becher (2002) and Dhillon1 et al. (2002). Results of numerical experiments that demonstrate the viability of our approach are reported.

Keywords: clustering algorithms, optimization, entropy

1. Introduction

The classical k-means algorithm (Forgy 1965) is probably the most widely known and used general clustering technique. A number of k-means like algorithms are available in the literature (see e.g., Dhillon and Modha (2001), Berkhin and Becher (2002), Dhillon1 et al. (2002), Xu and Jordan (1995), Kogan1 et al. (2003), Kogan2 et al. (2003), and Dempster et al. (1977) to name just a few). While each version of k-means is a two step "expectation–maximization" procedure, different similarity measures distinguish between various versions of the algorithm. We argue that the choice of a particular similarity measure may improve clustering of a specific dataset. We call this choice the "data driven similarity measure". The main contribution of this paper is the description of an optimization based machinery that generates a variety of k-means like algorithms via an appropriately chosen similarity measure.

For the proposed specific choice of a similarity measure we generate a k-means like algorithm that depends on two non-negative real parameters ν and μ. Accordingly we call

*This research was supported in part by the US Department of Defense, the United States–Israel Binational Science Foundation (BSF), and Northrop Grumman Mission Systems (NG/MS).

this algorithm the (ν, μ) algorithm. When $\nu = 1$ and $\mu = 0$ the (ν, μ) algorithm becomes the classical k-means algorithm, and when $\nu = 0$ and $\mu = 1$ the algorithm generalizes the Divisive Information Theoretic clustering algorithm (see Berkhin and Becher (2002), and Dhillon1 et al. (2002)). Intermediate values of the parameters regularize the logarithmic similarity measure by the squared Euclidean distance.

The outline of the paper is the following. In Section 2 we present general batch and incremental versions of the k-means like clustering algorithms, and provide a number of examples. A clustering scheme that combines the two versions is also presented. The (ν, μ) algorithm introduced in the paper is a specific example of the scheme. Section 3 describes an optimization approach to the centroid update procedure essential to the family of k-means like algorithms. The data set is presented and results of clustering experiments are collected in Section 4. Brief conclusions and further research directions are outlined in Section 5. Technical results concerning computational complexity of the algorithm are provided in an Appendix.

2. *k*-means like algorithms

A variety of modifications of the classical k-means clustering algorithm (Forgy 1965) have been introduced recently (see e.g., Dhillon and Modha (2001), Berkhin and Becher (2002), Dhillon1 et al. (2002) and Kogan1 et al. (2003)). All these modifications, exactly like the classical algorithm, are the gradient based Gauss-Seidel method (see e.g., Bertsekas and Tsitsiklis (1989)). The main difference between the various modifications of the algorithm is the choice of the "distance like" function $d(\cdot, \cdot)$ (for a partial list of specific examples of "distance like" functions see Table 2). The following is a basic description of the batch k-means type clustering algorithm:

Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ be a set of vectors in a subset $\mathbf{X}$ of the n-dimensional Euclidean space $\mathbf{R}^n$. Consider a partition $\Pi = \{\pi_1, \ldots, \pi_k\}$ of the set, i.e.,

$$\pi_1 \cup \ldots \cup \pi_k = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}, \quad \text{and} \quad \pi_i \bigcap \pi_j = \emptyset \text{ if } i \neq j.$$

Given a real valued function q whose domain is the set of subsets of $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ the quality of the partition is given by $Q(\Pi) = q(\pi_1) + \cdots + q(\pi_k)$. The problem is to identify an optimal partition $\{\pi_1^o, \ldots, \pi_k^o\}$, i.e., one that optimizes $q(\pi_1) + \cdots + q(\pi_k)$. Often the function q is associated with a "dissimilarity measure", or a *distance-like* function $d(\mathbf{x}, \mathbf{y})$ that satisfies the following basic properties:

$$d(\mathbf{x}, \mathbf{y}) \geq 0, \ \forall (\mathbf{x}, \mathbf{y}) \in \mathbf{X} \ \text{ and } \ d(\mathbf{x}, \mathbf{y}) = 0 \ \Leftrightarrow \ \mathbf{x} = \mathbf{y} \tag{1}$$

We call $d(\cdot, \cdot)$ a *distance-like* function, since we do not require d to be either symmetric or to satisfy the triangle inequality. Furthermore, when $d(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$ and $\mathbf{x}$ and $\mathbf{y}$ are l_2 unit norm vectors one has $d(\mathbf{x}, \mathbf{y}) = 1 \Leftrightarrow \mathbf{x} = \mathbf{y}$.

To describe the relation between q and d we define a centroid $\mathbf{c}$ of a cluster π by

$$\mathbf{c} = \mathbf{c}(\pi) = \arg\,\mathrm{opt}\left\{ \sum_{\mathbf{x}\in\pi} d(\mathbf{y}, \mathbf{x}),\ \ \mathbf{y} \in \mathbf{X} \right\}, \tag{2}$$

where by arg opt $f(x)$ we denote a point x_0 where the function f is optimized. Depending on the choice of d, "opt" is either "min" or "max." Indeed, to define a centroid one would like to *minimize* (2) with $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2$. On the other hand one would like to *maximize* the same expression when $d(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}$, and all the vectors are normalized. To simplify the exposition in what follows we shall primarily address the case "opt= min" keeping in mind that the inequalities should be reversed in the other case. If $q(\pi)$ is defined as $\sum_{\mathbf{x}\in\pi} d(\mathbf{c}(\pi), \mathbf{x})$, then centroids and partitions are associated as follows:

1. For a set of k centroids $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ one can define a partition $\{\pi_1, \ldots, \pi_k\}$ of the set $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ by:

$$\pi_i = \{\mathbf{x}_j \ :\ d(\mathbf{c}_i, \mathbf{x}_j) \le d(\mathbf{c}_l, \mathbf{x}_j) \text{ for each } l \ne i\} \tag{3}$$

   (we break ties arbitrarily).
2. Given a partition $\{\pi_1, \ldots, \pi_k\}$ of the set $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ one can define the corresponding centroids $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ by:

$$\mathbf{c}_i = \arg\min \left\{ \sum_{\mathbf{x}\in\pi_i} d(\mathbf{y}, \mathbf{x}),\ \ \mathbf{y} \in \mathbf{X} \right\}, \tag{4}$$

Batch k-means type algorithm

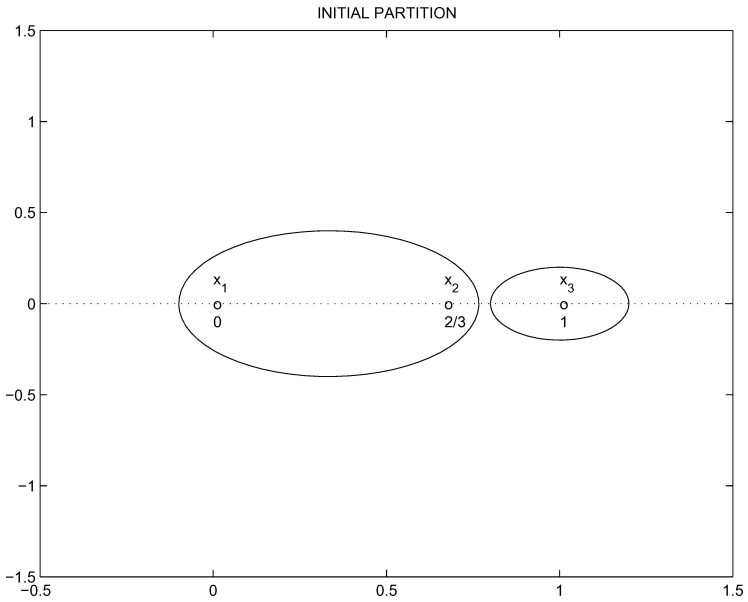
Given a user supplied tolerance `tol` > 0 do the following:

1. Set $t = 0$.
2. Start with an initial partitioning $\Pi^{(t)} = \{\pi_1^{(t)}, \ldots, \pi_k^{(t)}\}$
3. Apply (4) to recompute centroids.
4. Apply (3) to compute the partition $\Pi^{(t+1)} = \{\pi_1^{(t+1)}, \ldots, \pi_k^{(t+1)}\}$.
5. If $Q(\Pi^{(t)}) - Q(\Pi^{(t+1)}) >$ `tol`

   $\begin{cases} \text{set } t = t + 1 \\ \text{go to Step 3} \end{cases}$.
6. Stop.

We now analyze a batch k-means type algorithm: We shall denote by `nextKM`(Π) a partition Π' generated by a single iteration of the batch k-means algorithm from a partition Π.

Table 1. classical batch k-means trapped at a local minimum.



The algorithm may get trapped at a local minimum even for a very simple one dimensional dataset (see e.g., Duda et al. (2000), Dhillon2 et al. (2002), Dhillon et al. (2003), Dhillon1 et al. (2002), Kogan2 et al. (2003)). This remark is illustrated by the following example.

Example 2.1. Consider the distance like function $d(\mathbf{c}, \mathbf{x}) = \|\mathbf{c}-\mathbf{x}\|^2$, the data set $\{\mathbf{x}_1 = 0, \mathbf{x}_2 = \frac{2}{3}, \mathbf{x}_3 = 1\}$ and the initial partition $\Pi^{(0)}$ with $\pi_1^{(0)} = \{\mathbf{x}_1, \mathbf{x}_2\}, \pi_2^{(0)} = \{\mathbf{x}_3\}$ and $Q(\Pi^{(0)}) = \frac{2}{9}$ (see Table 1). Note that an application of the batch k-means algorithm does not change the initial partition $\Pi^{(0)}$. At the same time it is clear that the partition $\Pi' = \{\pi_1', \pi_2'\}$ ($\pi_1' = \{\mathbf{x}_1\}$, $\pi_2' = \{\mathbf{x}_2, \mathbf{x}_3\}$) with $Q(\Pi') = \frac{2}{36}$ is superior to the initial partition.

A different version of the k-means algorithm, incremental k-means clustering, remedies the problem presented by Example 2.1. We now briefly discuss the incremental k-means clustering. The decision whether a vector $\mathbf{x} \in \pi_i$ should be moved from cluster π_i to cluster π_j is made by the batch k-means algorithm based on the sign of

$$\Delta = -\|\mathbf{x} - \mathbf{c}(\pi_i)\|^2 + \|\mathbf{x} - \mathbf{c}(\pi_j)\|^2. \tag{5}$$

The quantity Δ disregards dislocations of the centroids $\mathbf{c}(\pi_i)$ and $\mathbf{c}(\pi_j)$ due to possible reassignment of $\mathbf{x}$. This in turn sometimes causes the batch k-means algorithm to miss a better partition. To avoid this deficiency we seek a single vector $\mathbf{x}$ whose reassignment leads to the sharpest decrease of the objective function. Once the vector is identified, the

reassignment is performed. To formally present the procedure described above we need additional definitions.

Definition 2.1. A first variation of a partition $\Pi = \{\pi_1, \ldots, \pi_k\}$ is a partition $\Pi' = \{\pi'_1, \ldots, \pi'_k\}$ obtained from Π by removing a single vector $\mathbf{x}$ from a cluster π_i of Π and assigning this vector to an existing cluster π_j of Π.

Note that the partition Π is a first variation of itself. Next we look for the "steepest descent" first variation, i.e., a first variation that leads to the maximal decrease of the objective function.

Definition 2.2. The partition $\texttt{nextFV}\ (\Pi)$ is a first variation of Π so that for each first variation Π' one has

$$Q(\texttt{nextFV}(\Pi)) \leq Q(\Pi'). \tag{6}$$

A straightforward computation shows that for the scalars $\mathbf{x}_1,\ \mathbf{x}_2,\ \mathbf{x}_3$ given in Example 2.1 the partition $\Pi' = \texttt{nextFV}(\Pi^0)$, where the initial partition $\Pi^0 = \{\{\mathbf{x}_1, \mathbf{x}_2\}, \{\mathbf{x}_3\}\}$, and $\Pi' = \{\{\mathbf{x}_1\}, \{\mathbf{x}_2, \mathbf{x}_3\}\}$.

To escape the local minimum trap we augment iterations of the batch k-means by an iteration of incremental version of the algorithm. The algoritm described below is a simplified version of the means clustering algorithm suggested in Kogan (2001) for a Euclidean squared distance like function.

k-means type clustering algorithm

For user supplied tolerances $\texttt{tol}_1 > 0$ and $\texttt{tol}_2 > 0$ do the following:

1. Start with an arbitrary partitioning $\Pi^{(0)} = \{\pi_1^{(0)}, \ldots, \pi_k^{(0)}\}$. Set the index of iteration $t = 0$.
2. Generate the partition $\texttt{nextKM}(\Pi^{(t)})$.
   if $[Q(\Pi^{(t)}) - Q(\texttt{nextKM}(\Pi^{(t)})) > \texttt{tol}_1]$
   $\begin{cases} \text{set } \Pi^{(t+1)} = \texttt{nextKM}(\Pi^{(t)}) \\ \text{increment } t \text{ by } 1 \\ \text{go to } 2 \end{cases}$
3. Generate the partition $\texttt{nextFV}(\Pi^{(t)})$.
   if $[Q(\Pi^{(t)}) - Q(\texttt{nextFV}(\Pi^{(t)})) > \texttt{tol}_2]$
   $\begin{cases} \text{set } \Pi^{(t+1)} = \texttt{nextFV}(\Pi^{(t)}) \\ \text{increment } t \text{ by } 1 \\ \text{go to } 2 \end{cases}$
4. Stop.

Due to the additional Step 3 Algorithm 2 always outperforms batch k-means in cluster quality. While execution of Step 3 requires computation of all conceivable first variations of $\Pi^{(t)}$ in some cases it comes at virtually zero additional computational cost (see Appendix).

Table 2. k-means like algorithms.

$d(\mathbf{y}, \mathbf{x})$	$\mathbf{X}$	constraint	"opt"	Algorithm 2 becomes	$\mathbf{c}(\pi)$
$\|\mathbf{x}-\mathbf{y}\|^2$	$\mathbf{R}^n$	none	min	classical batch k-means Forgy, 1965	$\mathbf{m}(\pi)$
$\mathbf{x}^T\mathbf{y}$	$\mathbf{S}_2^{n-1} \bigcap \mathbf{R}_+^n$	$\|\mathbf{x}\|_2 = 1$ and $\mathbf{x}[i] \geq 0$	max	spherical batch k-means Dhillon and Modha, 2001	$\frac{\mathbf{m}(\pi)}{\|\mathbf{m}(\pi)\|_2}$
$\sum_{i=1}^n \mathbf{x}[i] \log \frac{\mathbf{x}[i]}{\mathbf{y}[i]}$	$\mathbf{S}_1^{n-1} \bigcap \mathbf{R}_+^n$	$\|\mathbf{x}\|_1 = 1$ and $\mathbf{x}[i] \geq 0$	min	batch DIT clustering Dhillon et al. 2002	$\mathbf{m}(\pi)$

We conclude the section with examples of *distance-like* functions $d(\mathbf{x}, \mathbf{y})$, data domains $\mathbf{X}$, norm constraints imposed on the data, k-means like algorithms, and centroid formulas (see Table 2). For a set of l vectors $\pi = \{\mathbf{y}_1, \ldots, \mathbf{y}_l\}$ we denote the mean of the set by $\mathbf{m}(\pi)$, i.e., $\mathbf{m}(\pi) = \frac{\mathbf{y}_1+\cdots+\mathbf{y}_l}{l}$. We denote by $\mathbf{S}_p^{n-1}$ the l_p unit sphere in $\mathbf{R}^n$, i.e.,

$$\mathbf{S}_p^{n-1} = \{\mathbf{x} \ : \ \mathbf{x} \in \mathbf{R}^n, \ |\mathbf{x}[1]|^p + \cdots + |\mathbf{x}[n]|^p = 1\},$$

(here $\mathbf{x} = (\mathbf{x}[1], \ldots, \mathbf{x}[n])^T \in \mathbf{R}^n$).

Computation of a centroid $\mathbf{c}$ for a given cluster π as described by Eq. (4) is an optimization problem. The ability to carry out a fast and efficient computation of centroids is key to successful implementation of k-means like algorithms. In the next section we introduce a family of distance-like functions and an optimization technique that solves (4).

3. Optimization approach

In this section we present centroid computations for the (ν, μ) algorithm. Motivated by Text Mining applications we are targeting the data domain $\mathbf{X} = \mathbf{R}_+^n$ (i.e., we are interested in vectors with non-negative coordinates). An optimization formalism introduced next leads to a simple solution of the centroid computation problem.

For a set of vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbf{R}_+^n$ and a set of centroids $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\} \subset \mathbf{R}_+^n$ we define m vectors

$$\mathbf{d}_i(\mathbf{c}_1, \ldots, \mathbf{c}_k) = (d(\mathbf{c}_1, \mathbf{x}_i), \ldots, d(\mathbf{c}_k, \mathbf{x}_i))^T \in \mathbf{R}^k.$$

The support function σ_Δ of a simplex $\Delta \subset \mathbf{R}^k$ provides a convenient way to cast the clustering problem as an optimization problem. The support function $\sigma_\mathbf{S}$ of a closed convex

set $\mathbf{S} \in \mathbf{R}^n$ is defined by (see e.g., (Rockafellar 1970)):

$$\sigma_{\mathbf{S}}(\mathbf{v}) = \sup\{\mathbf{s}^T\mathbf{v} \colon \mathbf{s} \in \mathbf{S}\},$$

and a simplex $\Delta \subset \mathbf{R}^k$ is

$$\Delta = \{\mathbf{w} \ : \ \mathbf{w} \in \mathbf{R}^k, \ \mathbf{w}^T\mathbf{e} = 1, \mathbf{w}[i] \geq 0, \ i = 1, \ldots, k\},$$

where $\mathbf{e} = (1, \ldots, 1)^T$ is a vector of ones. For a set of centroids $\{\mathbf{c}'_l\}_{l=1}^k$ and a vector $\mathbf{x}_i$ we identify a vector $\mathbf{w}_i \in \Delta$ so that $\mathbf{w}_i[l] = 1$ if $\mathbf{c}'_l$ is the centroid nearest $\mathbf{x}_i$, and $\mathbf{w}_i[l] = 0$ otherwise. Keeping in mind $\inf_x f(x) = -\sup_x\{-f(x)\}$ we get

$$\mathbf{w}_i = \arg\max \sigma_\Delta(-\mathbf{d}_i(\mathbf{c}'_1, \ldots, \mathbf{c}'_k)) = \arg\min_{\mathbf{w}\in\Delta} \mathbf{w}^T\mathbf{d}_i(\mathbf{c}'_1, \ldots, \mathbf{c}'_k). \tag{7}$$

For the sets of vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and $\{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$ we define "updated centroids" $\{\mathbf{c}''_1, \ldots, \mathbf{c}''_k\}$ as follows:

$$\begin{aligned}(\mathbf{c}''_1, \ldots, \mathbf{c}''_k) &= \arg\min_{\mathbf{c}_1,\ldots,\mathbf{c}_k} \sum_{i=1}^m \mathbf{w}_i^T\mathbf{d}_i(\mathbf{c}_1, \ldots, \mathbf{c}_k) \\ &= \arg\min_{\mathbf{c}_1,\ldots,\mathbf{c}_k} \operatorname{tr}(W^T D(\mathbf{c}_1, \ldots, \mathbf{c}_k))\end{aligned} \tag{8}$$

where W is the $k \times m$ matrix whose columns are $\mathbf{w}_i$, $i = 1, \ldots, m$, and $D(\mathbf{c}_1, \ldots, \mathbf{c}_k) = D$ is the $k \times m$ matrix with $D_{ij} = d(\mathbf{c}_j, \mathbf{x}_i)$. Unlike (4) Eq. (8) provides a formula for k updated centroids at once.

We shall denote the vector of partial derivatives of the function $d(\mathbf{c}, \mathbf{x})$ with respect to the first n variables by $\nabla_{\mathbf{c}} d(\mathbf{c}, \mathbf{x})$. Analogously, for the function $\psi(\mathbf{c}_1, \ldots, \mathbf{c}_k) = \operatorname{tr}(W^T D(\mathbf{c}_1, \ldots, \mathbf{c}_k))$ the vector of n partial derivatives with respect to the n coordinates $\mathbf{c}_j[1], \ldots, \mathbf{c}_j[n]$ is denoted by $\nabla_{\mathbf{c}_j}\psi(\mathbf{c}_1, \ldots, \mathbf{c}_k)$. If $\mathbf{c}''_j$ belongs to the interior of the domain $\mathbf{X}$, then due to (8) one has

$$\nabla_{\mathbf{c}_j}\psi(\mathbf{c}''_1, \ldots, \mathbf{c}''_k) = 0. \tag{9}$$

Furthermore, a straightforward computation shows that

$$\nabla_{\mathbf{c}_j}\psi(\mathbf{c}''_1, \ldots, \mathbf{c}''_k) = \sum_{i=1}^m \mathbf{w}_i[j]\nabla_{\mathbf{c}} d(\mathbf{c}''_j, \mathbf{x}_i). \tag{10}$$

Next we use (9) and (10) to provide analytic expressions for $\mathbf{c}''_j$, $j = 1, \ldots, k$ through $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and $\{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$. We shall consider a specific "distance–like" function defined through the kernel

$$\phi(t) = \begin{cases} -\ln t + t - 1 & \text{if } t > 0 \\ +\infty & \text{otherwise} \end{cases}$$

and non-negative scalars ν and μ:

$$d(\mathbf{c}, \mathbf{x}) = \frac{\nu}{2}\|\mathbf{c} - \mathbf{x}\|^2 + \mu \sum_{j=1}^{n} \mathbf{x}[j]\phi\left(\frac{\mathbf{c}[j]}{\mathbf{x}[j]}\right). \tag{11}$$

In what follows, motivated by continuity arguments, we define $0 \log \frac{0}{a} = 0 \log \frac{a}{0} = 0$ for each $a \geq 0$. It is easy to verify that $d(\mathbf{c}, \mathbf{x})$ satisfies the required distance-like properties (1).

Note that when $\nu = 0$ and $\mu = 1$, the second term in (11)

$$\sum_{j=1}^{n} \mathbf{x}[j]\phi\left(\frac{\mathbf{c}[j]}{\mathbf{x}[j]}\right) = \sum_{j=1}^{n} \mathbf{x}[j] \ln \frac{\mathbf{x}[j]}{\mathbf{c}[j]} + \sum_{j=1}^{n} \mathbf{c}[j] - \sum_{j=1}^{n} \mathbf{x}[j]. \tag{12}$$

Under the assumption $\|\mathbf{x}\|_1 = \|\mathbf{c}\|_1 = 1$ (i.e., $\sum_{j=1}^{n} \mathbf{x}[j] = \sum_{j=1}^{n} \mathbf{c}[j] = 1$) the right hand side of (12) reduces to

$$d_1(\mathbf{c}, \mathbf{x}) = \sum_{j=1}^{n} \mathbf{x}[j] \ln \frac{\mathbf{x}[j]}{\mathbf{c}[j]}. \tag{13}$$

This is the Kullback-Leibler relative entropy, which is used here to measure the distance between two unit l_1 vectors in $\mathbf{R}^n_+$, see e.g., (Teboulle 1992) and references therein. The Divisive Information Theoretic clustering algorithm proposed in Dhillon et al. (2002) is the batch version of the classical k-means algorithm with the squared Euclidean distance substituted by the distance d_1, and the algorithm proposed in Berkhin and Becher (2002) is its incremental counterpart. With the distance d_1 the resulting centroid $\mathbf{c}(\pi)$ of a cluster π is the arithmetic mean of the vectors in the cluster (see e.g., (Dhillon et al. 2002)), and this result justifies the assumption $\|\mathbf{c}\|_1 = 1$. In this section we show that the stringent unit simplex constraint can be avoided and replaced by the larger non-negative orthant, by simply using the normalized entropy $\phi(t) = -\log t + t - 1$, and replacing (13), with the resulting (12).

The choice of the distance (11) which combines a squared Euclidean distance with the relative entropy is motivated by the following rationale: on one hand to keep the standard features of the k-means algorithm (through the squared distance), while on the other, to handle nonnegative data. This idea has been recently proposed and successfully used in the development of various optimization algorithms, see (Auslender et al. 1999).

To justify the application of the gradient equation (9), we consider the two cases:

1. $\nu > 0$, $\mu = 0$.
   In this case one can use (9) to solve (8) over $\mathbf{R}^n$. It is well known that a centroid $\mathbf{c}''$ of a cluster π in the case of squared Euclidean distance is given by the arithmetic mean $\mathbf{m}(\pi)$. Due to convexity of $\mathbf{X}$ the same result holds when the cluster belongs to $\mathbf{X}$. For this reason in Section 3.1 below we shall compute centroids for the case $\mu > 0$ only.

2. $\mu > 0$.
Consider $\sum_{i=1}^{m} \mathbf{w}_i[j]d(\mathbf{c}_j, \mathbf{x}_i)$, the contribution due to centroid $\mathbf{c}_j$ into the expression (8). The expression to be minimized with respect to $\mathbf{c}_j$ is just

$$\sum_{\mathbf{x}\in\pi_j} \left[\frac{\nu}{2}\|\mathbf{c}_j - \mathbf{x}\|^2 + \mu \sum_{l=1}^{n} \mathbf{x}[l]\phi\left(\frac{\mathbf{c}_j[l]}{\mathbf{x}[l]}\right)\right]. \tag{14}$$

We consider the contribution of each coordinate to (14). The contribution of coordinate l is

$$\sum_{\mathbf{x}\in\pi_j} \frac{\nu}{2}|\mathbf{c}_j[l] - \mathbf{x}[l]|^2 + \mu \sum_{\mathbf{x}\in\pi_j} \mathbf{x}[l]\phi\left(\frac{\mathbf{c}_j[l]}{\mathbf{x}[l]}\right). \tag{15}$$

If there is an index l such that $\mathbf{x}[l] = 0$ for each $\mathbf{x} \in \pi_j$, then (15) becomes

$$\sum_{\mathbf{x}\in\pi_j} \frac{\nu}{2}(\mathbf{c}_j[l])^2 + \mu \sum_{\mathbf{x}\in\pi_j} \mathbf{c}_j[l]. \tag{16}$$

Since $\mathbf{c}_j[l]$ must be non-negative, expression (16) is minimized when $\mathbf{c}_j[l] = 0$.
On the other hand, if there is some $\mathbf{x} \in \pi_j$ so that $\mathbf{x}[l] > 0$, then the lth coordinate of $\mathbf{c}_j''$ should be positive (recall that here $\phi'(t) = -\frac{1}{t} + 1 \to -\infty$ as $t \to 0^+$), and one can apply the gradient Eq. (9) to find $\mathbf{c}_j''[l]$.

Next we provide a formula for $\mathbf{c}_j[l]$ when $\mu > 0$ and $\mathbf{x}[l] > 0$ for at least one $\mathbf{x} \in \pi_j$.

3.1. Centroid computation

A detailed solution to problem (4) with d given by (11), and $\nu > 0$, $\mu > 0$ is presented in this subsection (the solution for the limit case $\nu = 0$, $\mu > 0$ is also obtained as a special case). We assume that $\mathbf{x}[l] > 0$ for at least one $\mathbf{x} \in \pi_j$ and compute the lth coordinate of the vector $\nabla_{\mathbf{c}_j}\psi(\mathbf{c}_1'', \ldots, \mathbf{c}_k'')$.

A straightforward computation (see (10)) leads to the following:

$$\mathbf{c}_j''[l] \cdot \nu \sum_{i=1}^{m} \mathbf{w}_i[j] + \left(\mu \sum_{i=1}^{m} \mathbf{w}_i[j]\nu \sum_{i=1}^{m} \mathbf{w}_i[j]\mathbf{x}_i[l]\right) - \frac{1}{\mathbf{c}_j''[l]} \cdot \mu \sum_{i=1}^{m} \mathbf{w}_i[j]\mathbf{x}_i[l] = 0. \tag{17}$$

To simplify the above equation, define (for convenience we omit the indices):

$$\alpha = \sum_{i=1}^{m} \mathbf{w}_i[j], \;\; \beta = \sum_{i=1}^{m} \mathbf{w}_i[j]\mathbf{x}_i[l], \tag{18}$$

then Eq. (17) becomes a (ν, μ) family of quadratic equations with respect to $\mathbf{c}_j''[l]$

$$\nu\alpha \cdot (\mathbf{c}_j''[l])^2 - (\nu\beta - \mu\alpha)\mathbf{c}_j''[l] - \mu\beta = 0. \tag{19}$$

We remind the reader (see (7)) of the following:

$$\mathbf{w}_i[j] = 1 \text{ if } \mathbf{c}_j \text{ is the centroid nearest } \mathbf{x}_i, \quad \text{and } \mathbf{w}_i[j] = 0 \text{ otherwise.}$$

Hence $\sum_{i=1}^m \mathbf{w}_i[j]$ is the number of vectors in cluster π_j (and $\sum_{i=1}^m \mathbf{w}_i[j]\mathbf{x}_i[l] = \sum_{\mathbf{x}\in\pi_j} \mathbf{x}[l]$). In particular when $\sum_{i=1}^m \mathbf{w}_i[j] = 0$, cluster $\mathbf{c}_j$ is **not** the nearest centroid for **all** the vectors. In the rare (but possible) case cluster π_j should become empty, the number of clusters should be decreased, and $\mathbf{c}_j$ should **not** be recomputed. We now assume that $\sum_{i=1}^m \mathbf{w}_i[j] > 0$. Recalling that $\nu\alpha$ is positive, see (18), and since $\mathbf{X} = \mathbf{R}_+^n$ we are interested in the non-negative solution of quadratic Eq. (19)

$$\begin{aligned}\mathbf{c}_j''[l] &= \frac{(\nu\beta - \mu\alpha) + \sqrt{(\nu\beta - \mu\alpha)^2 + 4\nu\mu\alpha\beta}}{2\nu\alpha} \\ &= \frac{\nu\beta - \mu\alpha + |\nu\beta + \mu\alpha|}{2\nu\alpha} = \frac{\beta}{\alpha},\end{aligned}$$

which after substitution of the expressions (α, β) given in (18) leads to the following simple formula:

$$\mathbf{c}_j''[l] = \frac{\sum_{i=1}^m \mathbf{w}_i[j]\mathbf{x}_i[l]}{\sum_{i=1}^m \mathbf{w}_i[j]} = \frac{\sum_{\mathbf{x}\in\pi_j} \mathbf{x}[l]}{|\pi_j|} \tag{20}$$

where $|\pi_j|$ denotes the number of vectors in cluster π_j. Note that (20) is just the lth coordinate of the cluster's arithmetic mean $\mathbf{m}(\pi_j)$, and the derived centroid is independent of the parameters (ν, μ). In addition we note that the above derivation also recovers the solution for the limit case $\nu = 0$, $\mu > 0$ (when the quadratic Eq. (19) becomes a linear one) and gives through (20), the centroid $\mathbf{c}_j'' = \mathbf{m}(\pi_j)$.

4. Experimental results

In this section we report results of numerical experiments. To compare our results with those already available in the literature we focus on a specific text data set recently examined by Dhillon and Modha (2001). In one of the experiments of Dhillon and Modha (2001) the spherical k-means algorithm was applied to a data set containing 3893 documents. This data set contains the following three document collections (available from `ftp://ftp.cs.cornell.edu/pub/smart`):

- Medlars Collection (1033 medical abstracts),
- CISI Collection (1460 information science abstracts),
- Cranfield Collection (1400 aerodynamics abstracts).

Table 3. spherical k-means generated "confusion" matrix with **69** "misclassified" documents using 4,099 words.

	Medlars	CISI	Cranfield
Cluster 0	1004	5	4
Cluster 1	18	1440	16
Cluster 2	11	15	1380

Partitioning the entire collection into 3 clusters generates the "confusion" matrix given by Table 3 and reported in Dhillon and Modha (2001) (here the entry ij is the number of documents that belong to cluster i and document collection j). The "confusion" matrix shows that only 69 documents (i.e., less that 2% of the entire collection) have been "misclassified" by the algorithm. After removing stopwords (Dhillon and Modha 2001) reported 24,574 unique words, and after eliminating low-frequency and high-frequency words they selected **4,099** words to construct the vector space model (Berry and Browne 1999).

Our data set is a merger of the three document collections (available from `http://www.cs.utk.edu/~lsi/`):

- DC0 (Medlars Collection 1033 medical abstracts)
- DC1 (CISI Collection 1460 information science abstracts)
- DC2 (Cranfield Collection 1398 aerodynamics abstracts)

The Cranfield collection tackled by Dhillon and Modha contained two empty documents. These two documents have been removed from DC2. The other document collections are identical.

We select 600 "best" terms and build vectors of dimension 600 for each document (see (Dhillon et al. 2003) for details). A two step clustering procedure is applied to the document vectors. The first step of the procedure is the Spherical Principal Directions Divisive Partitioning (sPDDP) clustering algorithm recently reported by (Dhillon et al. 2003). The Singular Value Decomposition based algorithm is applied to unit l_2 document vectors and the clustering results are reported in Table 4. When the number of terms is relatively small, some documents may contain no selected terms, and their corresponding vectors are zeros (see Table 8). We always remove these vectors ahead of clustering and assign the "empty" documents into a special cluster. This cluster is the last row in the "confusion" matrix (and is empty in the experiment reported in Tables 4–7). Note that the clustering procedures produce "confusion" matrices with a single "dominant" entry in each row. We regard the document collection corresponding to the "dominant" column as the one represented by the cluster. The other row entries are counted as misclassifications.

The final partition generated by sPDDP is an input for the (ν, μ) clustering algorithm, i.e., the k-means like algorithm with the distance function (11). When the outcome of sPDDP is available the document vectors are re-normalized in l_1 norm, and the (ν, μ) algorithm is

Table 4. sPDDP generated initial "confusion" matrix with **68** "misclassified" documents using best 600 terms.

	DC0	DC1	DC2
Cluster 0	1000	3	1
Cluster 1	8	10	1376
Cluster 2	25	1447	21
"empty" documents			
Cluster 3	0	0	0

Table 5. $\nu = 0,\ \mu = 1$ generated final "confusion" matrix with **44** "misclassified" documents.

	DC0	DC1	DC2
Cluster 0	1010	6	0
Cluster 1	2	4	1387
Cluster 2	21	1450	11
"empty" documents			
Cluster 3	0	0	0

Table 6. $\nu = 100,\ \mu = 1$ generated final "confusion" matrix with **48** "misclassified" documents using best 600 terms.

	DC0	DC1	DC2
Cluster 0	1010	5	1
Cluster 1	3	6	1384
Cluster 2	20	1449	13
"empty" documents			
Cluster 3	0	0	0

Table 7. $\nu = 1,\ \mu = 0$ generated final "confusion" matrix with **52** "misclassified" documents using best 600 terms.

	DC0	DC1	DC2
Cluster 0	1011	6	2
Cluster 1	8	12	1386
Cluster 2	14	1442	10
"empty" documents			
Cluster 3	0	0	0

applied to the partition generated by sPDDP. The final partitions for three selected values of the $(\nu,\ \mu)$ pair are reported in Tables 5–7 respectively. We display results for the "extreme" values (0, 1), (1, 0), and an intermediate value of $(\nu,\ \mu)$. Since (0, 1) objective function value of the initial partition is about 100 times more than the (1, 0) objective function value of the same partition we have decided to choose the intermediate value (100, 1) to balance the "extreme" values of the objective functions.

While the number of selected terms is only 600 (i.e., only about 15% of the number of terms reported in Dhillon and Modha (2001)) the quality of the sPDDP generated confusion matrix is comparable with that of the confusion matrix generated by the spherical k-means

Table 8. Number of documents "misclassified" by sPDDP, and "sPDDP $+ (\nu, \mu)$" algorithms.

		Documents misclassified by sPDDP			
# of terms	Zero vectors	Alone	$+(1, 0)$	$+(100, 1)$	$+(0, 1)$
100	12	383	499	269	166
200	3	277	223	129	112
300	0	228	124	80	68
400	0	88	68	58	56
500	0	76	63	40	41
600	0	68	52	48	44

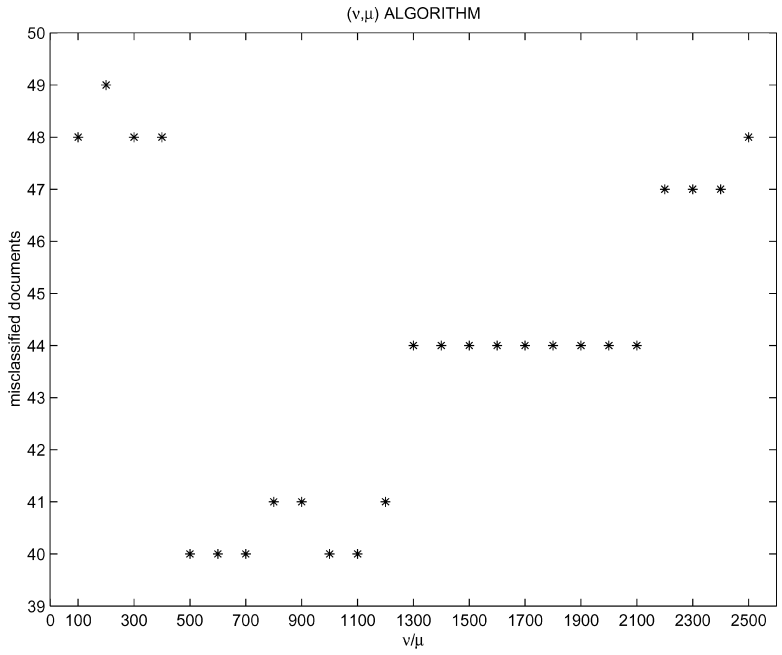


Table 9. Number of unit l_2 document vectors "misclassified" by "sPDDP+k–means", and "sPDDP+spherical k-means".

		Documents misclassified by	
# of terms	Zero vectors	sPDDP +k–means	sPDDP+spherical k-means
100	12	258	229
200	3	133	143
300	0	100	104
400	0	80	78
500	0	62	57
600	0	62	54

algorithm (see Table 3). A subsequent application of the (ν, μ) algorithm to the partition generated by sPDDP further improves the confusion matrix (see Tables 5–7).
We pause briefly to discuss some properties of the (ν, μ) algorithm.

1. Unlike the DIT clustering algorithm (Dhillon et al. 2002), (Berkhin and Becher 2002) the data domain $\mathbf{X}$ for the (ν, μ) algorithm is not restricted to l_1 unit norm vectors in $\mathbf{R}^n_+$ (and the (ν, μ) algorithm does not require l_1 normalization of the data).
2. In the extreme case $\nu = 1$, $\mu = 0$ the classical k-means algorithm is recovered.
3. When $\nu = 0$, $\mu = 1$ and the document vectors are normalized in l_1 norm the (ν, μ) algorithm becomes the combination of the algorithms reported in Dhillon1 et al. (2002) and Berkhin and Becher (2002).

Table 8 summarizes clustering results for the sPDDP algorithm and the combinations of "sPDDP + (ν, μ)" algorithm for the three selected values for (ν, μ) and different choices of index terms. Note that zero document vectors are created when the number of selected terms is less than 300. Table 8 indicates consistent superiority of the "sPDDP + (0, 1)" algorithm. We next show by an example that this indication is not necessarily always correct. The graph below shows the number of misclassified documents for 600 selected terms and 25 values of the (ν, μ) pair. While μ is kept 1, ν varies from 100 to 2500 with step 100.

The graph indicates the best performance for $\frac{\nu}{\mu} = 500$, 600, 700, 1000, and 1100.

We complete the section with clustering results generated by the "sPDDP + k–means" and "sPDDP+spherical k-means". The algorithms are applied to the three document collections DC0, DC1, and DC2. We use the same vector space construction as for the "sPDDP+(ν, μ)" algorithm, but do not change the document vectors unit l_2 norm at the second stage of the clustering scheme. The results of the experiment are summarized in Table 9 for different choices of index terms.

5. Concluding remarks and future work

Optimization tools have been applied to a specific "distance–like" function to generate a two parameter family of k-means like clustering algorithms, and preliminary experimental

results based on the proposed approach have been described. The "extreme" members of the family recover the classical k-means algorithm (Forgy 1965), and the Divisive Information Theoretic clustering algorithm (Dhillon et al. 2002), (Berkhin and Becher 2002). The results of numerical experiments indicate, however, that the best clustering results can be obtained with "intermediate" parameter values.

Overall complexity of large data sets motivates application of a sequence of algorithms for clustering a single data set (see e.g., Dhillon et al. (2003)). The output of algorithm i becomes the input of algorithm $i + 1$, and the final partition is generated by the last algorithm. We call a sequence of two (or more) clustering algorithms applied to a data set a *hybrid scheme*. The results presented in the paper have been generated using a hybrid scheme consisting of two algorithms: the SVD-based sPDDP, and the k-means like (ν, μ) algorithm. We plan to focus on the following problems:

1. While the first step of the sequence is the SVD-based sPDDP algorithm that deals with unit l_2 vectors, the experiments indicate that l_1 unit vectors may better fit text mining applications. The sPDDP algorithm is an optimization procedure that approximates a set of unit l_2 vectors by a circle on the l_2 sphere in $\mathbf{R}^n$. We hope the optimization tools will be useful for solving the corresponding approximation problem when the data set resides on the l_1 sphere in $\mathbf{R}^n$.
2. The experiments presented in Section 4 show that the best clustering results can be achieved at an intermediate value of (ν, μ). We plan to investigate the dependence of the final partition quality on the parameter values (ν, μ).
3. Motivated by success of the (ν, μ) algorithm we plan to investigate additional classes of φ-divergence measures which are a generalization of relative entropy and have been successfully used in various optimization algorithms (see e.g., Teboulle (1999) and Auslender et al. (1999) and references therein).
4. We plan to run the experiments on a variety of large document collections.

Appendix

In this section we briefly analyze the computational complexity of the incremental step of the (ν, μ) algorithm. To simplify the exposition we carry out the computations for the special case $\|\mathbf{x}_i\|_1 = 1, i = 1, \ldots, m$ only.

For two clusters π_i with centroids $\mathbf{c}_i = \mathbf{c}(\pi_i)$ and $|\pi_i|$ vectors each, $i = 1, 2$ and a vector $\mathbf{x} \in \pi_1$ we denote by

1. π_1^- a cluster obtained from the cluster π_1 be removing $\mathbf{x}$ from π_1,
2. π_2^+ a cluster obtained from the cluster π_2 by assigning $\mathbf{x}$ to π_2.

Our goal is to evaluate

$$\mathcal{Q}\{\pi_1, \pi_2\} - \mathcal{Q}\{\pi_1^-, \pi_2^+\} = [q(\pi_1) - q(\pi_1^-)] + [q(\pi_2) - q(\pi_2^+)] \tag{21}$$

for two extreme cases of the (ν, μ) algorithm described below.

Squared euclidean norm ($\mu = 0$)

The expression for (21) is given, for example, in Duda et al. (2000) and Kogan (2000) as follows:

$$\mathcal{Q}\{\pi_1, \pi_2\} - \mathcal{Q}\{\pi_1^-, \pi_2^+\} = \frac{|\pi_1|}{|\pi_1| - 1}\|\mathbf{x} - \mathbf{c}(\pi_1)\|^2 - \frac{|\pi_2|}{|\pi_2| + 1}\|\mathbf{x} - \mathbf{c}(\pi_2)\|^2. \tag{22}$$

Evaluation of $\|\mathbf{x} - \mathbf{c}(\pi)\|$ for each vector $\mathbf{x}_i$ and centroid $\mathbf{c}_j$ is carried out by step 2 of Algorithm 2. Contribution of the Euclidean part of the objective function for incremental iteration of the algorithm comes, therefore, at virtually no additional computational expense.

Information–theoretical distance ($\nu = 0$)

The computational complexity of the incremental version of the Divisive Information Theoretic clustering algorithm is discussed in Berkhin and Becher (2002). Detailed computations provided below lead us to believe that computational cost associated with the incremental step may in fact be much lower than the cost reported in Berkhin and Becher (2002).

First we consider cases of vector "removal" and "addition" separately.

– $\pi = \{\mathbf{x}_1, \ldots, \mathbf{x}_{p-1}, \mathbf{x}\}$ with $\mathbf{c} = \mathbf{c}(\pi)$, and $\pi^- = \{\mathbf{x}_1, \ldots, \mathbf{x}_{p-1}\}$ with $\mathbf{c}^- = \mathbf{c}(\pi^-)$. Since $(p-1)\mathbf{c}^- = p\mathbf{c} - \mathbf{x}$ one gets the following:

$$\begin{aligned} q(\pi) - q(\pi^-) &= \sum_{i=1}^{p-1}\sum_{j=1}^{n}\left(\mathbf{x}_i[j]\log\frac{\mathbf{x}_i[j]}{\mathbf{c}[j]} + \mathbf{c}[j] - \mathbf{x}_i[j]\right) \\ &\quad + \sum_{j=1}^{n}\left(\mathbf{x}[j]\log\frac{\mathbf{x}[j]}{\mathbf{c}[j]} + \mathbf{c}[j] - \mathbf{x}[j]\right) \\ &\quad - \sum_{i=1}^{p-1}\sum_{j=1}^{n}\left(\mathbf{x}_i[j]\log\frac{\mathbf{x}_i[j]}{\mathbf{c}^-[j]} + \mathbf{c}^-[j] - \mathbf{x}_i[j]\right) \\ &= \sum_{i=1}^{p-1}\sum_{j=1}^{n}\left(\mathbf{x}_i[j]\log\frac{\mathbf{c}^-[j]}{\mathbf{c}[j]}\right) + \sum_{j=1}^{n}\mathbf{x}[j]\log\frac{\mathbf{x}[j]}{\mathbf{c}[j]} \\ &= \sum_{j=1}^{n}(p\mathbf{c}[j] - \mathbf{x}[j])\log\left(\frac{p}{p-1}\mathbf{c}[j] - \frac{1}{p-1}\mathbf{x}[j]\right) \\ &\quad + \sum_{j=1}^{n}\mathbf{x}[j]\log\frac{\mathbf{x}[j]}{\mathbf{c}[j]}. \end{aligned}$$

Finally,

$$q(\pi) - q(\pi^-) = \sum_{j=1}^{n} (p\mathbf{c}[j] - \mathbf{x}[j]) \log\left(\frac{p}{p-1}\mathbf{c}[j] - \frac{1}{p-1}\mathbf{x}[j]\right) + \sum_{j=1}^{n} \mathbf{x}[j] \log\frac{\mathbf{x}[j]}{\mathbf{c}[j]}. \tag{23}$$

– $\pi = \{\mathbf{x}_1, \ldots, \mathbf{x}_p\}$ with $\mathbf{c} = \mathbf{c}(\pi)$, and $\pi^+ = \{\mathbf{x}_1, \ldots, \mathbf{x}_p, \mathbf{x}\}$ with $\mathbf{c}^+ = \mathbf{c}(\pi^+)$. We use the identity $(p+1)\mathbf{c}^+ = p\mathbf{c} + \mathbf{x}$ to obtain:

$$\begin{aligned} q(\pi) - q(\pi^+) &= \sum_{i=1}^{p}\sum_{j=1}^{n}\left(\mathbf{x}_i[j]\log\frac{\mathbf{x}_i[j]}{\mathbf{c}[j]} + \mathbf{c}[j] - \mathbf{x}_i[j]\right) \\ &\quad - \sum_{i=1}^{p}\sum_{j=1}^{n}\left(\mathbf{x}_i[j]\log\frac{\mathbf{x}_i[j]}{\mathbf{c}^+[j]} + \mathbf{c}^+[j] - \mathbf{x}_i[j]\right) \\ &\quad - \sum_{j=1}^{n}\left(\mathbf{x}[j]\log\frac{\mathbf{x}[j]}{\mathbf{c}[j]} + \mathbf{c}^+[j] - \mathbf{x}[j]\right) \\ &= \sum_{i=1}^{p}\sum_{j=1}^{n}\left(\mathbf{x}_i[j]\log\frac{\mathbf{c}^+[j]}{\mathbf{c}[j]}\right) + \sum_{j=1}^{n}\mathbf{x}[j]\log\frac{\mathbf{c}^+[j]}{\mathbf{x}[j]} \\ &= p\sum_{j=1}^{n}\mathbf{c}[j]\log\left(\frac{p}{p+1} + \frac{1}{p+1}\frac{\mathbf{x}[j]}{\mathbf{c}[j]}\right) \\ &\quad + \sum_{j=1}^{n}\mathbf{x}[j]\log\left(\frac{p}{p+1}\frac{\mathbf{c}[j]}{\mathbf{x}[j]} + \frac{1}{p+1}\right), \end{aligned}$$

and $q(\pi) - q(\pi^+)$ is given by

$$p\sum_{j=1}^{n}\mathbf{c}[j]\log\left(\frac{p}{p+1} + \frac{1}{p+1}\frac{\mathbf{x}[j]}{\mathbf{c}[j]}\right) + \sum_{j=1}^{n}\mathbf{x}[j]\log\left(\frac{p}{p+1}\frac{\mathbf{c}[j]}{\mathbf{x}[j]} + \frac{1}{p+1}\right). \tag{24}$$

The expression for

$$\mathcal{Q}\{\pi_1, \pi_2\} - \mathcal{Q}\{\pi_1^-, \pi_2^+\} = [q(\pi_1) - q(\pi_1^-)] + [q(\pi_2) - q(\pi_2^+)]$$

follows in a straightforward manner from (23) and (24):

$$\sum_{j=1}^{n}(|\pi_1|\mathbf{c}_1[j]-\mathbf{x}[j])\log\left(\frac{|\pi_1|}{|\pi_1|-1}\mathbf{c}_1[j]-\frac{1}{|\pi_1|-1}\mathbf{x}[j]\right)+\sum_{j=1}^{n}\mathbf{x}[j]\log\frac{\mathbf{x}[j]}{\mathbf{c}_1[j]}$$
$$+|\pi_2|\sum_{j=1}^{n}\mathbf{c}_2[j]\log\left(\frac{|\pi_2|}{|\pi_2|+1}+\frac{1}{|\pi_2|+1}\frac{\mathbf{x}[j]}{\mathbf{c}_2[j]}\right)$$
$$+\sum_{j=1}^{n}\mathbf{x}[j]\log\left(\frac{|\pi_2|}{|\pi_2|+1}\frac{\mathbf{c}_2[j]}{\mathbf{x}[j]}+\frac{1}{|\pi_2|+1}\right).$$

The computational complexity associated with evaluation of the above four term expression is about the same as that of (22). Indeed,

1. Provided $|\pi_1|\mathbf{c}_1-\mathbf{x}$ is a dense vector of dimension n the number of operation required to compute

$$\sum_{j=1}^{n}(|\pi_1|\mathbf{c}_1[j]-\mathbf{x}[j])\log\left(\frac{|\pi_1|}{|\pi_1|-1}\mathbf{c}_1[j]-\frac{1}{|\pi_1|-1}\mathbf{x}[j]\right)$$

   is $O(n)$.
2. The term $\sum_{j=1}^{n}\mathbf{x}[j]\log\frac{\mathbf{x}[j]}{\mathbf{c}[j]}$ has been already computed by step 3 of Algorithm 2, and comes for "free".
3. Provided $\mathbf{c}_2$ is a dense vector of dimension n the number of operation required to compute $|\pi_2|\sum_{j=1}^{n}\mathbf{c}_2[j]\log\left(\frac{|\pi_2|}{|\pi_2|+1}+\frac{1}{|\pi_2|+1}\frac{\mathbf{x}[j]}{\mathbf{c}_2[j]}\right)$ is $O(n)$.
4. The document vector $\mathbf{x}$ is always sparse, hence the number of operation required to compute $\sum_{j=1}^{n}\mathbf{x}[j]\log\left(\frac{|\pi_2|}{|\pi_2|+1}\frac{\mathbf{c}_2[j]}{\mathbf{x}[j]}+\frac{1}{|\pi_2|+1}\right)$ is much smaller than $O(n)$.

Our numerical experiments indicate that (perhaps because of logarithmic function sensitivity to low frequencies) the (ν,μ) algorithm with nontrivial information component (i.e., $\mu>0$) collects documents containing same words together. In such a case it is reasonable to expect to obtain *sparse* centroid vectors, and this would significantly reduce computational effort associated with steps 1 and 3 above.

Acknowledgments

The authors thank anonymous reviewers for valuable suggestions that improved exposition of the results.

References

Auslender A, Teboulle M and Ben–Tiba S (1999) Interior proximal amd multiplier methods based on second order homogeneous kernels. Mathematics of Operations Research, 24:645–668.

Berkhin P and Becher JD (2002) Learning simple relations: Theory and applications. In: Proceedings of the Second SIAM International Conference on Data Mining.

Berry M and Browne M (1999) Understanding Search Engines. SIAM.

Bertsekas DP and Tsitsiklis JN (1989) Parallel and Distributed Computation: Numerical Methods. Prentice-Hall, New Jersey.

Dempster A, Laird N and Rubin D (1977) Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, 39.

Dhillon IS, Guan Y and Kogan J (2002) Refining clusters in high-dimensional text data. In: Proceedings of the Workshop on Clustering High Dimensional Data and its Applications (held in conjunction with the Second SIAM International Conference on Data Mining).

Dhillon, IS, Kogan J and Nicholas C (2003) Feature selection and document clustering, In Berry MW Ed. A Comprehensive Survey of Text Mining, pp. 73–100.

Dhillon IS and Modha DS (2001) Concept decompositions for large sparse text data using clustering. Machine Learning, 42(1):143–175.

Duda RO, Hart PE and Stork DG (2000) Pattern Classification. John Wiley & Sons.

Forgy E (1965) Cluster analysis of multivariate Data: Efficiency vs. interpretability of classifications. Biometrics, 21(3):768.

Inderjit S. Dhillon Subramanyam Mallela and Rahul Kumar (2002) Enhanced word clustering for hierarchical text classification. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2002), pp. 191–200.

Kogan J (2001) Clustering large unstructured document sets. In: Berry MW Ed. Computational Information Retrieval, pp. 107–117.

Kogan J (2001) Means clustering for text data. In: Proceedings of the Workshop on Text Mining at the First SIAM International Conference on Data Mining, pp. 47–54.

Kogan J, Teboulle M and Nicholas C (2003) Optimization approach to generating families of k-means like algorithms. In: Proceedings of the Workshop on Clustering High Dimensional Data and its Applications (held in conjunction with the Third SIAM International Conference on Data Mining).

Kogan J, Teboulle M and Nicholas C (2003) The entropic geometric means algorithm: An approach for building small clusters for large text datasets. In: Proceedings of the Workshop on Clustering Large Data Sets (held in conjunction with the Third IEEE International Conference on Data Mining), pp. 63–71.

Lei Xu and Michael I. Jordan (1995) On convergence properties of the EM Algorithm for Gaussian Mixtures. MIT A.I. Memo No. 1520, C.B.C.L. paper No. 111.

Rockafellar RT (1970) Convex Analysis Princeton University Press, Princeton, NJ.

Teboulle M (1992) On φ-divergence and its applications. In: Phillips FY and Rousseau J Eds. Systems and Management Science by Extremal Methods–Research Honoring Abraham Charnes at Age 70, Kluwer Academic Publishers, pp. 255–273.

Teboulle M (1997) Convergence of proximal-like algorithms. SIAM J. of Optimization, 7:1069-1083.