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

and

Charles Nicholas

nicholas@umbc.edu

Department of Computer Science and Electrical Engineering

UMBC, Baltimore, MD 21250

January 28, 2003

Abstract

A clustering algorithm that exploits special characteristics of a data set may lead to superior results. The spherical k–means algorithm introduced recently by Dhillon and Modha [7] is specifically designed to handle unit length document vectors. A recent contribution by Zhao and Karypis [16] reports results of clustering experiments with 7 clustering algorithms and 12 different text data sets. Zhao and Karypis conclude that the objective function based on cosine similarity (and used in [7]) "leads to the best solutions irrespective of the number of clusters for most of the data sets." We describe a general approach to generate k–means like algorithms, which is based on a family of similarity measures and optimization tools. To demonstrate the viability of our approach we develop and apply a particular family of algorithms to a text data set considered in [7] and report on the derived clustering results.

1. Introduction

The classical k–means algorithm [10] 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., [7], [8], [15], [4] 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 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 Information–Theoretical k–means (see [8], [2]). 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 k–means like clustering algorithms, and provide a number of examples. 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

The following: is a basic description of a 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.,

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

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 $Q(\Pi) = q(\pi_1) + \ldots + q(\pi_k)$. The problem is to identify an optimal partition $\{\pi_1^o, \ldots, \pi_k^o\}$, i.e., one that optimizes $q(\pi_1) + \ldots + q(\pi_k)$. 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:

$$d(\mathbf{x}, \mathbf{y}) \geq 0, \ \forall (\mathbf{x}, \mathbf{y}) \in \mathbf{X} \ \text{ and } \ d(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y} \tag{2.1}$$

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 $d(\mathbf{x}, \mathbf{y}) = 1 \iff \mathbf{x} = \mathbf{y}$.

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

$$\mathbf{c} = \mathbf{c}(\pi) = \arg\operatorname{opt} \left\{ \sum_{\mathbf{x} \in \pi} d(\mathbf{y}, \mathbf{x}), \ \mathbf{y} \in \mathbf{X} \right\}, \tag{2.2}$$

where by $\arg\operatorname{opt}\ f(x)$ 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* (2.2) 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:

$$\pi_i = \{\mathbf{x}_j \ : \ d(\mathbf{c}_i, \mathbf{x}_j) \leq d(\mathbf{c}_l, \mathbf{x}_j) \text{ for each } l \neq i\} \tag{2.3}$$

   (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:

$$\mathbf{c}_i = \arg\min \left\{ \sum_{\mathbf{x} \in \pi_i} d(\mathbf{y}, \mathbf{x}), \ \mathbf{y} \in \mathbf{X} \right\}, \tag{2.4}$$

We now describe a batch k–means type algorithm:

Algorithm 2.1 Batch k–means type algorithm

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

1. *Set* $t = 0$.
2. *Start with an initial partitioning* $\Pi^t = \left\{\pi_1^t, \ldots, \pi_k^t\right\}$
3. *Apply (2.4) to recompute centroids.*
4. *Apply (2.3) to compute the partition* $\Pi^{(t+1)} = \left\{\pi_1^{(t+1)}, \ldots, \pi_k^{(t+1)}\right\}$.
5. *If* $Q\left(\Pi^{(t)}\right) - Q\left(\Pi^{(t+1)}\right) < \texttt{tol}$
   set $t = t + 1$
   goto Step 3
6. *Stop.*

Table 1 below displays a number of 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. 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., $\mathbf{m}(\pi) = \dfrac{\mathbf{y}_1 + \ldots + \mathbf{y}_l}{l}$. We denote by $\mathbf{S}_p^{n-1}$ the l_p unit sphere in $\mathbf{R}^n$, i.e.,

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

(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 equation (2.4) 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 (2.4).

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., coordinates of the document vectors are non-negative).

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

$d(\mathbf{y},\mathbf{x})$	$\mathbf{X}$	constraint	"opt"	Algorithm 2.1 becomes	$\mathbf{c}(\pi)$
$\|\mathbf{x}-\mathbf{y}\|^2$	$\mathbf{R}^n$	none	min	classical batch k–means (see [10])	$\mathbf{m}(\pi)$
$\mathbf{x}^T\mathbf{y}$	$\mathbf{S}_2^{n-1}\bigcap\mathbf{R}_+^n$	$\|\mathbf{x}\|_2=1$ and $\mathbf{x}[i]\geq 0$	max	spherical batch k–means (see [7])	$\frac{\mathbf{m}(\pi)}{\|\mathbf{m}(\pi)\|_2}$
$\sum_{i=1}^{n}\mathbf{x}[i]\log\frac{\mathbf{x}[i]}{\mathbf{y}[i]}$	$\mathbf{S}_1^{n-1}\bigcap\mathbf{R}_+^n$	$\|\mathbf{x}\|_1=1$ and $\mathbf{x}[i]\geq 0$	min	batch IT $-$ means (see [8])	$\mathbf{m}(\pi)$

Table 1: k–means like algorithms

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., [12]):

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

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

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

where $\mathbf{e}=(1,\ldots,1)^T$ is a vector of ones. For a set of centroids $\{\mathbf{c}_1',\ldots,\mathbf{c}_k'\}$ 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 $\inf\limits_x f(x)=-\sup\limits_x\{-f(x)\}$ we get

$$\mathbf{w}_i=\arg\max\sigma_\Delta\left(-\mathbf{d}_i(\mathbf{c}_1',\ldots,\mathbf{c}_k')\right)=\arg\min_{\mathbf{w}\in\Delta}\mathbf{w}^T\mathbf{d}_i(\mathbf{c}_1',\ldots,\mathbf{c}_k'). \tag{3.1}$$

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:

$$(\mathbf{c}_1'',\ldots,\mathbf{c}_k'')=\arg\min_{\mathbf{c}_1,\ldots,\mathbf{c}_k}\sum_{i=1}^{m}\mathbf{w}_i^T\mathbf{d}_i(\mathbf{c}_1,\ldots,\mathbf{c}_k)=\arg\min_{\mathbf{c}_1,\ldots,\mathbf{c}_k}\operatorname{tr}\left(W^TD(\mathbf{c}_1,\ldots,\mathbf{c}_k)\right) \tag{3.2}$$

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 (2.4) equation (3.2) 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 $\psi(\mathbf{c}_1, \ldots, \mathbf{c}_k) = \mathrm{tr}\left(W^T D(\mathbf{c}_1, \ldots, \mathbf{c}_k)\right)$ 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 (3.2) one has

$$\nabla_{\mathbf{c}_j} \psi(\mathbf{c}_1'', \ldots, \mathbf{c}_k'') = 0. \tag{3.3}$$

Furthermore, a straightforward computation shows that

$$\nabla_{\mathbf{c}_j} \psi(\mathbf{c}_1'', \ldots, \mathbf{c}_k'') = \sum_{i=1}^{m} \mathbf{w}_i[j] \nabla_{\mathbf{c}} d(\mathbf{c}_j'', \mathbf{x}_i). \tag{3.4}$$

Next we use (3.3) and (3.4) 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

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

and non-negative scalars ν and μ:

$$d(\mathbf{c}, \mathbf{x}) = \frac{\nu}{2} \|\mathbf{c} - \mathbf{x}\|^2 + \mu \sum_{j=1}^{n} \mathbf{x}[j] \phi\left(\frac{\mathbf{c}[j]}{\mathbf{x}[j]}\right). \tag{3.5}$$

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 (2.1).

Note that when $\nu = 0$, the second term in (3.5) is the Kullback-Leibler relative entropy measure, which is used here to measure the distance between two vectors in $\mathbf{R}_+^n$, see e.g., [13] and references therein. The choice of the above distance 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 [1].

To justify the application of the gradient equation (3.3), we consider the two cases:

1. $\nu > 0, \mu = 0$.

   In this case one can use (3.3) to solve (3.2) over $\mathbf{R}^n$. Due to the convexity of $\mathbf{X}$

the solution of the unconstrained problem $\mathbf{c}_j'' = \mathbf{m}(\pi_j)$ belongs to $\mathbf{X}$. For this reason in Subsection 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 (3.2). The expression to be minimized with respect to $\mathbf{c}_j$ is just

$$\sum_{\mathbf{x} \in \pi_j} \left[\frac{\nu}{2} \|\mathbf{c}_j - \mathbf{x}\|^2 + \mu \sum_{l=1}^{n} \mathbf{x}[l] \phi\left(\frac{\mathbf{c}_j[l]}{\mathbf{x}[l]} \right) \right]. \tag{3.6}$$

We consider the contribution of each coordinate to (3.6). The contribution of coordinate l is

$$\sum_{\mathbf{x} \in \pi_j} \frac{\nu}{2} |\mathbf{c}_j[l] - \mathbf{x}[l]|^2 + \mu \sum_{\mathbf{x} \in \pi_j} \mathbf{x}[l] \phi\left(\frac{\mathbf{c}_j[l]}{\mathbf{x}[l]} \right). \tag{3.7}$$

If there is an index l such that $\mathbf{x}[l] = 0$ for each $\mathbf{x} \in \pi_j$, then (3.7) becomes

$$\sum_{\mathbf{x} \in \pi_j} \frac{\nu}{2} \left(\mathbf{c}_j[l]\right)^2 + \mu \sum_{\mathbf{x} \in \pi_j} \mathbf{c}_j[l]. \tag{3.8}$$

Since $\mathbf{c}_j[l]$ must be non-negative, expression (3.8) 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 l^{th} 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 equation (3.3) 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 (2.4) with d given by (3.5), $\nu > 0$ and $\mu > 0$ is presented in this subsection. We assume that $\mathbf{x}[l] > 0$ for at least one $\mathbf{x} \in \pi_j$ and compute the l^{th} coordinate of the vector $\nabla_{\mathbf{c}_j} \psi(\mathbf{c}_1'', \ldots, \mathbf{c}_k'')$.

A straightforward computation (see (3.4)) leads to the following:

$$\mathbf{c}_j''[l] \cdot \nu \sum_{i=1}^{m} \mathbf{w}_i[j] + \left(\mu \sum_{i=1}^{m} \mathbf{w}_i[j] - \nu \sum_{i=1}^{m} \mathbf{w}_i[j] \mathbf{x}_i[l] \right) - \frac{1}{\mathbf{c}_j''[l]} \cdot \mu \sum_{i=1}^{m} \mathbf{w}_i[j] \mathbf{x}_i[l] = 0. \tag{3.9}$$

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

$$\alpha = \sum_{i=1}^{m} \mathbf{w}_i[j], \ \beta = \sum_{i=1}^{m} \mathbf{w}_i[j] \mathbf{x}_i[l], \tag{3.10}$$

then equation (3.9) becomes a (ν, μ) family of quadratic equations with respect to $\mathbf{c}_j''[l]$

$$\nu\alpha \cdot \left(\mathbf{c}_j''[l]\right)^2 - (\nu\beta - \mu\alpha)\mathbf{c}_j''[l] - \mu\beta = 0. \tag{3.11}$$

We remind the reader (see (3.1)) of the following:

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

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 this 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 (3.10), and since $\mathbf{X} = \mathbf{R}_+^n$ we are interested in the non-negative solution of quadratic equation (3.11)

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

which after substitution of the expressions (α, β) given in (3.10) leads to the following simple formula:

$$\mathbf{c}_j''[l] = \frac{\sum_{i=1}^{m} \mathbf{w}_i[j]\mathbf{x}_i[l]}{\sum_{i=1}^{m} \mathbf{w}_i[j]}. \tag{3.12}$$

Notice that the derived centroid is thus independent of the parameters (ν, μ), and thus in particular, we note that the above derivation recovers also the limit case $\nu = 0, \mu > 0$ and gives through (3.12), the centroid $\mathbf{c}_j'' = \mathbf{m}(\pi_j)$.

4. Experimental results

Dhillon and Modha [7] have recently used the spherical k–means algorithm for clustering text data. In one of the experiments of [7] the algorithm was applied to a data set containing 3893 documents. The 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).

Partitioning the entire collection into 3 clusters generates the "confusion" matrix given by Table 2 and reported in [7] (here the entry ij is the number of documents that belong

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

Table 2: spherical k–means generated "confusion" matrix with **69** "misclassified" documents using 4,099 words

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 [7] 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 [3].

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 [6] 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 [6]. The Singular Value Decomposition based algorithm is applied to unit l_2 document vectors and the clustering results are reported in Table 3. When the number of terms is relatively small, some documents may contain no selected terms, and their corresponding vectors are zeros (see Table 7). 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 3, 4, 5 and 6). Note that the clustering procedures produce "confusion" matrices with a single "dominant" entry in each row.

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 (3.5). Batch $k-$means like algorithms are prone to local minima (see e.g., [9], [5], [6]). An iteration of an incremental version of k–means seeks a single vector $\mathbf{x}$ whose reassignment leads to the sharpest decrease of the objective function. Once the vector is identified, the iteration performs the reassignment. While computationally more accurate, the incremental algorithm is much slower than the batch algorithm. To benefit from the speed of the batch algorithm and the accuracy of the incremental algorithm we augment Algorithm 2.1 by the incremental version of k–means. That is, we run the batch version until it stops, then we run one iteration of the incremental version. The procedure is repeated until consecutive application of batch and incremental iterations fail to change a partition (see [6] for detailed discussion of the procedure and its efficiency in the case of the classical and the spherical k–means algorithms). We address the computational complexity of this addition in the Appendix.

The document vectors are re-normalized in l_1 norm, and the (ν, μ) algorithm is applied to the partition generated by sPDDP. The final partitions for three selected values of the $(\nu,\ \mu)$ pair are reported in Tables 4, 5 and 6 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.

	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 3: sPDDP generated initial "confusion" matrix with **68** "misclassified" documents using best 600 terms

While the number of selected terms is only 600 (i.e., only about 15% of the number of terms reported in [7]) the quality of the sPDDP generated confusion matrix is comparable with that of the confusion matrix generated by the spherical k–means algorithm (see Table 2). A subsequent application of the (ν, μ) algorithms to the partition generated by sPDDP further improves the confusion matrix (see Tables 4, 5 and 6).

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

Table 4: $\nu = 0,\ \mu = 1$ generated final "confusion" matrix with **44** "misclassified" documents.

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

We pause briefly to discuss some properties of the (ν, μ) algorithm.

1. Unlike the Information-Theoretical k–means [8], [2] the data domain $\mathbf{X}$ for the (ν, μ) algorithm is $\mathbf{R}_+^n$ (and the 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 precisely the algorithm reported in [8], [2].

	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

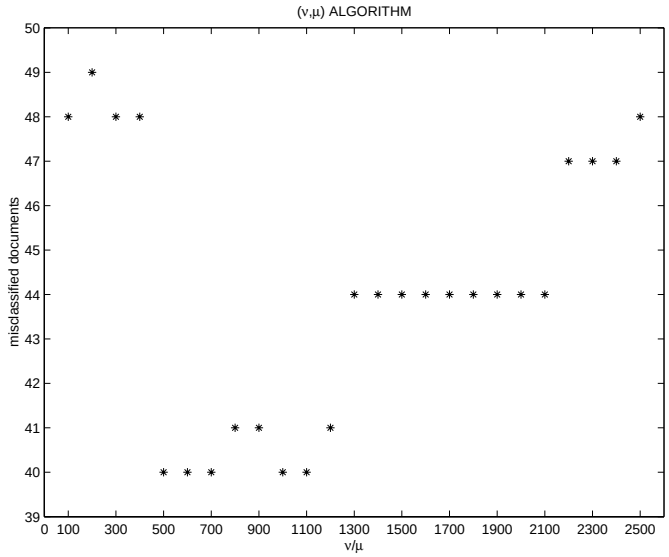
Table 6: $\nu = 1$, $\mu = 0$ generated final "confusion" matrix with **52** "misclassified" documents using best 600 terms

Table 7 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 7 indicates consistent superiority of the

		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 7: Number of documents "misclassified" by sPDDP, and "sPDDP+(ν, μ)" algorithms.

"sPDDP+$(0, 1)$" algorithm. We next show by an example that this indication is not necessarily 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 8 for different choices of index terms.

		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

Table 8: Number of unit l_2 document vectors "misclassified" by "sPDDP+k–means", and "sPDDP+spherical k–means".

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 algorithm, and preliminary

experimental results based on the proposed approach have been described. The "extreme" members of the family recover the classical k–means algorithm [10], and the Information–Theoretical k–means algorithm [8], [2]. The results of numerical experiments indicate, however, that the best clustering results can be obtained for "intermediate" parameter values.

Overall complexity of large data sets motivates application a sequence of algorithms for clustering a single data set (see e.g., [6]). 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). The dependence of the final partition on the parameter values (ν, μ) will be investigated.

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., [14], [1] and references therein).

4. We plan to run the experiments on a variety of large document collections.

References

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

[2] P. Berkhin and Becher J. D. Learning simple relations: Theory and applications. In *Proc. Second SIAM International Conference on Data Mining*, pages 420–436, Arlington, April 2002.

[3] M. Berry and M. Browne. *Understanding Search Engines.* SIAM, 1999.

[4] A. Dempster, N. Laird, and D. Rubin. Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society*, 39, 1977.

[5] I. S. Dhillon, Y. Guan, and J. Kogan. Refining clusters in high-dimensional text data. In I. S. Dhillon and J. Kogan, editors, *Proceedings of the Workshop on Clustering High Dimensional Data and its Applications at the Second SIAM International Conference on Data Mining*, pages 71–82. SIAM, 2002.

[6] I. S. Dhillon, J. Kogan, and C. Nicholas. Feature selection and document clustering. In M.W. Berry, editor, *A Comprehensive Survey of Text Mining.* Springer-Verlag, 2003, to appear.

[7] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, January 2001. Also appears as IBM Research Report RJ 10147, July 1999.

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

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

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

[11] J. Kogan. Clustering large unstructured document sets. In M.W.Berry, editor, *Computational Information Retrieval*, pages 107–117. SIAM, 2000.

[12] R. T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, NJ, 1970.

[13] M. Teboulle. On φ-divergence and its applications. In F.Y. Phillips and J. Rousseau, editors, *Systems and Management Science by Extremal Methods–Research Honoring Abraham Charnes at Age 70*, pages 255–273, Kluwer Academic Publishers. Nowell, MA, 1992.

[14] M. Teboulle. Convergence of proximal-like algorithms. *SIAM J. of Optimization*, 7:1069–1083, 1997.

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

[16] Y. Zhao and G. Karypis. Comparison of agglomerative and partitional document clustering algorithms. In I. S. Dhillon and J. Kogan, editors, *Proceedings of the Workshop on Clustering High Dimensional Data and its Applications at the Second SIAM International Conference on Data Mining*, pages 83–93. SIAM, 2002.

6. 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

$$\mathcal{Q}\left\{\pi_1, \pi_2\right\} - \mathcal{Q}\left\{\pi_1^-, \pi_2^+\right\} = \left[q(\pi_1) - q(\pi_1^-)\right] + \left[q(\pi_2) - q(\pi_2^+)\right] \tag{6.1}$$

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

6.1. Squared Euclidean norm ($\mu = 0$)

The expression for (6.1) is given, for example, in [9] and [11] as follows:

$$\mathcal{Q}\left\{\pi_1, \pi_2\right\} - \mathcal{Q}\left\{\pi_1^-, \pi_2^+\right\} = \frac{|\pi_1|}{|\pi_1| - 1} \left\|\mathbf{x} - \mathbf{c}(\pi_1)\right\|^2 - \frac{|\pi_2|}{|\pi_2| + 1} \left\|\mathbf{x} - \mathbf{c}(\pi_2)\right\|^2. \tag{6.2}$$

Evaluation of $\|\mathbf{x} - \mathbf{c}(\pi)\|$ for each vector $\mathbf{x}_i$ and centroid $\mathbf{c}_j$ has been carried out by step 3 of Algorithm 2.1. Contribution of the Euclidean part of the objective function for incremental iteration of the algorithm comes, therefore, at virtually no additional computational expense.

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

The computational complexity of the incremental Information–Theoretical k–means is discussed in [2]. 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 [2].

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:

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

Finally,

$$q(\pi) - q\left(\pi^-\right) = \sum_{j=1}^{n}(p\mathbf{c}[j] - \mathbf{x}[j])\log\left(\frac{p}{p-1}\mathbf{c}[j] - \frac{1}{p-1}\mathbf{x}[j]\right) + \sum_{j=1}^{n}\mathbf{x}[j]\log\frac{\mathbf{x}[j]}{\mathbf{c}[j]}. \quad (6.3)$$

- $\pi = \{\mathbf{x}_1, \ldots, \mathbf{x}_p\}$ 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:

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

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

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

$$
p\sum_{j=1}^{n}\mathbf{c}[j]\log\left(\frac{p}{p+1}+\frac{1}{p+1}\frac{\mathbf{x}[j]}{\mathbf{c}[j]}\right)+\sum_{j=1}^{n}\mathbf{x}[j]\log\left(\frac{p}{p+1}\frac{\mathbf{c}[j]}{\mathbf{x}[j]}+\frac{1}{p+1}\right). \tag{6.4}
$$

The expression for

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

follows straightforward from (6.3) and (6.4):

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

The computational complexity associated with evaluation of the above four term expression is about the same as that of (6.2). Indeed,

1. Provided $|\pi_1|\mathbf{c}_1-\mathbf{x}$ is a dense vector the number of operation required to compute $\sum_{j=1}^{n}\left(|\pi_1|\mathbf{c}_1[j]-\mathbf{x}[j]\right)\log\left(\frac{|\pi_1|}{|\pi_1|-1}\mathbf{c}_1[j]-\frac{1}{|\pi_1|-1}\mathbf{x}[j]\right)$ 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 2.1, and comes for "free".

3. Provided $\mathbf{c}_2$ is a dense vector 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 computations mentioned in items 1 and 3 above would become much cheaper.