The Entropic Geometric Means Algorithm: An Approach to Building Small Clusters for Large Text Datasets

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

September 10, 2003

Abstract

Related documents should contain same, or similar, terms. The number of terms co–occurring in all documents of a document cluster may provide an indication of the cluster's quality. One can expect a "good" document cluster to contain "large" numbers of terms co–occurring in all documents that belong to the cluster. In reality, however, when a document cluster is large, there is no single term that occurs in all the documents of the cluster (stop words may be an exception, but those are usually removed ahead of clustering). In contrast, when a cluster is small, one can expect certain terms to occur in all its documents. In this paper we introduce a new algorithm, called the Entropic Geometric Means (EGM) which emerges by replacing the squared Euclidean distance with an entropy-like distance in a k–means like algorithm. The EGM algorithm attempts to generate small clusters of documents so that a set of terms co–occurring in all documents of a cluster is non–empty for many clusters of the EGM generated partition.

1. Introduction

The classical k-means clustering algorithm [10] is a well known technique for clustering of data. To outline our main goals and contribution, we begin with an informal presentation. More details and precise formulations will be given in the next section. Basically, the clustering problem can be represented as a nonconvex optimization problem with explicit constraints, and the resulting k-mean clustering algorithm is essentially a *gradient based* algorithm, (see e.g., [3]), which leads to a local solution. One of the main computational steps in the k-mean clustering algorithm consists of computing the centroid $c(\pi)$ of a cluster π, and involves the solution of a convex optimization problem

$$\mathbf{c} = \mathbf{c}(\pi) = \arg\min \left\{ \sum_{\mathbf{x}\in\pi} d(\mathbf{c}, \mathbf{x}),\ \mathbf{c} \in \mathbf{S} \right\}, \tag{1.1}$$

where $\mathbf{S}$ is a given convex subset of a finite dimensional vector space, and $d(\cdot, \cdot)$ measures the discrepancy between a cluster centroid and the data points in $\mathbf{X}$. The function d is supposed to be convex in the first argument and should satisfy the property

$$d(\mathbf{x}, \mathbf{y}) \geq 0,\ \forall \mathbf{x}, \mathbf{y} \in \mathbf{S},\ \text{with equality iff}\ \ \mathbf{x} = \mathbf{y},$$

(symmetry and the triangle inequality need not to be satisfied). The classical k-means clustering algorithm corresponds to $\mathbf{S} = \mathbf{R}^n$ and $d(\mathbf{c}, \mathbf{x}) = \|\mathbf{c} - \mathbf{x}\|^2$ as the discrepancy measure. A variety of modifications of the classical k–means clustering algorithm [10] have been introduced recently (see e.g., [8], [2], [9], [15], to name just a few). All these modifications, exactly like the classical algorithm, are also gradient based methods. The main difference between the various modifications is the choice of the "discrepancy measure" $d(\cdot, \cdot)$, which we call here a "distance like" function, and of the set $\mathbf{S}$. We believe that a particular choice of a distance like function d may improve clustering of a specific dataset. Motivated by text mining applications, we shall consider the specific case when $\mathbf{S}$ is the nonnegative orthant $\mathbf{R}^n_+$, which in turn will govern the choice of d in problem (1.1). In this paper we choose a specific entropy-like distance d, and use this choice to derive a new clustering algorithm. The proposed choice for d emerges from earlier works in information theory within the use of the so-called Csiszar or/and Bregman divergences (see e.g., [5], [6]) which are usually defined in that context over the set of probability measures. However here, following ideas developed in several contributions in the area of nonlinear optimization (see e.g., [1], [11], [12], [17], [18] and references therein) it will be used as a distance-like function to measure the discrepancy between two vectors in $\mathbf{R}^n_+$ (see next section for precise definitions and properties). For reasons that will become clear later (see Section 2), we call the proposed algorithm the Entropic Geometric Means (EGM), and we shall show that EGM generates clusters of document that contain the same terms.

The remaining of the paper is as follows. In Section 2 we review k–means like clustering algorithms, the concept of entropy-like distances and introduce the EGM algorithm. The dataset and results of clustering experiments are provided in Section 3. Brief conclusions, further research directions, and possible applications of the proposed algorithm are outlined in Section 4. Arguments leading to centroids computations are collected in the Appendix.

2. The Entropic-Geometric Means algorithm

We shall denote column n dimensional vectors by lowercase boldface Latin letters $\mathbf{x}$, $\mathbf{y}$, ... The entries of the vector $\mathbf{x}$ are denoted by $\mathbf{x}[1], \ldots, \mathbf{x}[n]$. Motivated by Text Mining application we shall consider a set of vectors $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ in the non negative orthant of the n-dimensional Euclidean space $\mathbf{R}^n$, that is we let $\mathbf{S} = \mathbf{R}^n_+$. To develop a k-means type algorithm for clustering, we need to define an appropriate divergence measure d in $\mathbf{S}$, which will replace the usual squared Euclidean distance. In this paper we shall be concerned with a specific "entropy-like" distance function

$$d(\mathbf{c}, \mathbf{x}) = \sum_{j=1}^{n} \mathbf{x}[j] \phi \left(\frac{\mathbf{c}[j]}{\mathbf{x}[j]} \right), \tag{2.1}$$

where ϕ is a strictly convex function defined on the nonnegative real line. The divergence measure defined by (2.1) is known as ϕ-divergence, and was introduced by Csiszar [6] as a generalized measure of information, viewed as a "distance" function between probability measures. Note that divergence measures are not distances in the usual sense, since they need not to be symmetric or satisfy the triangle inequality. The entropy terminology stems from the fact that by choosing the kernel function $\phi(t) = t \log t$, the resulting ϕ-divergence is nothing else but the Kullback-Leibler relative entropy. Here we are not concerned with the statistical information theoretic significance of the ϕ-divergence. Rather, we adopt this concept to define a "distance" between vectors in the non negative orthant in the optimization problem (1.1). Indeed, there are no reasons to require the dataset to belong to the unit simplex (see discussion at the end of this section). One can consider vectors in the nonnegative orthant by *normalizing* the function ϕ so that $\phi(1) = \phi'(1) = 0$. This can be applied to several interesting choices of the kernel functions ϕ to produce adequate entropy-like distances. For more details, properties and applications of entropy-like distances see, for example, [1], [11], [17], and [18].

Here, we concentrate on the entropy kernel, which after normalization (with the convention $0 \log 0 = 0$) is defined by

$$\phi(t) = \begin{cases} t \log t - t + 1 & \text{if} \quad t \geq 0, \\ +\infty & \text{otherwise.} \end{cases} \tag{2.2}$$

With the specific choice of the kernel ϕ given above we obtain

$$d(\mathbf{c},\mathbf{x}) = \sum_{j=1}^{n} \mathbf{c}[j] \log \frac{\mathbf{c}[j]}{\mathbf{x}[j]} + \mathbf{x}[j] - \mathbf{c}[j],$$

which is the (normalized) Kullback-Leibler relative entropy distance. Note that this kernel is also the link between ϕ-divergences and another family of divergence measures, the so-called Bregman distances D_h, see e.g. [5]. For a strictly convex differentiable function satisfying certain technical hypotheses (see e.g., [11], [12]), D_h is defined as $D_h(\mathbf{x},\mathbf{y}) = h(\mathbf{x}) - h(\mathbf{y}) - (\mathbf{x}-\mathbf{y})^T \nabla h(\mathbf{y})$, where ∇h denotes the gradient of h. By taking $h(\mathbf{x}) = \sum_{j=1}^{n} \mathbf{x}[j] \log \mathbf{x}[j]$, we have $D_h(\mathbf{x},\mathbf{y}) = d(\mathbf{x},\mathbf{y})$. Moreover, it can be shown that up to a multiplicative constant, Bregman and Csiszar divergences coincide only for the entropy kernel. For more details see e.g., [11] and references therein.[1]

After this short recall on entropy-like distances, we are ready to present our algorithm. The following basic description of a k–means type clustering algorithm is applicable to a variety of "distance–like" functions such as ϕ-divergences or/and Bregman based divergences, (see [15], and for more general results and applications see [13]).

Consider a partition $\Pi = \{\pi_1, \ldots, \pi_k\}$ of the set $\mathbf{X}$, i.e.,

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

We define the centroid $\mathbf{c} = \mathbf{c}(\pi)$ of a cluster π as a solution of the optimization problem (1.1) with $\mathbf{S} = \mathbf{R}_+^n$. The quality of the cluster π is defined by

$$q(\pi) = \sum_{\mathbf{x} \in \pi_i} d(\mathbf{c},\mathbf{x}). \tag{2.3}$$

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., the one that minimizes $q(\pi_1) + \ldots + q(\pi_k)$.

Note that 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:

$$\pi_i = \{\mathbf{x}_j \ : \ d(\mathbf{c}_i,\mathbf{x}_j) \leq d(\mathbf{c}_l,\mathbf{x}_j) \text{ for each } l \neq i\} \tag{2.4}$$

   (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 solving (1.1).

[1] Note that the ϕ-divergence given in (2.1) is jointly convex in $(\mathbf{c},\mathbf{x})$ for **any** convex kernel ϕ, a property which is not always shared by a Bregman divergence generated by an arbitrary convex function h.

A batch k–means type algorithm is the following procedure alternating between the two steps described above:

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 (1.1) to recompute centroids.*
4. *Apply (2.4) to compute the partition* $\texttt{nextB}\left(\Pi^{(t)}\right) = \Pi^{(t+1)} = \left\{\pi_1^{(t+1)}, \ldots, \pi_k^{(t+1)}\right\}$.
5. *If* $Q\left(\Pi^{(t)}\right) - Q\left(\Pi^{(t+1)}\right) > \texttt{tol}$
   set $t = t + 1$
   goto Step 3
6. *Stop.*

As noted earlier in the introduction, when $d(\mathbf{c}, \mathbf{x}) = \|\mathbf{c} - \mathbf{x}\|^2$, Algorithm 2.1 becomes the classical batch k–means clustering algorithm (see [10]).

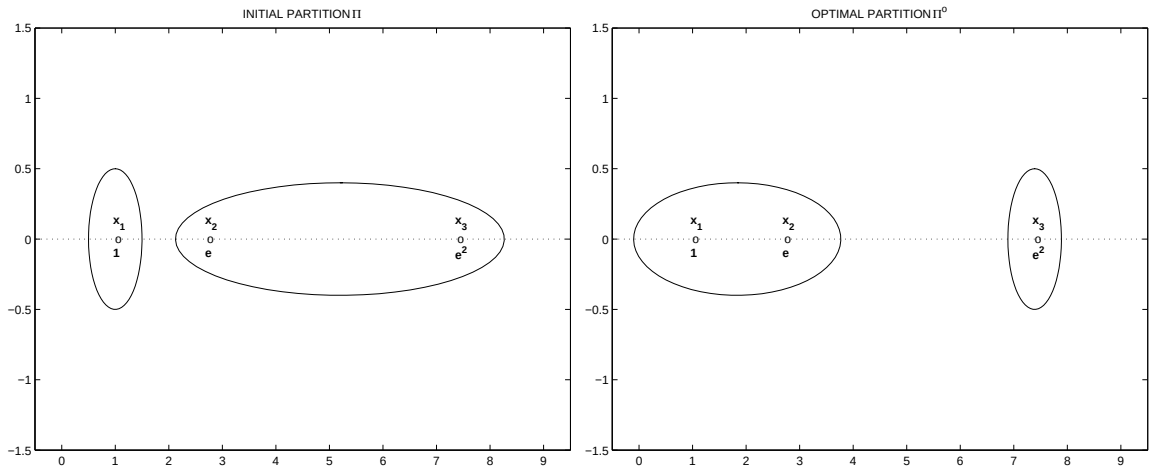
Computation of a centroid $\mathbf{c}$ for a given cluster π as prescribed by equation (1.1) is a convex optimization problem, which can be solved analytically for various choices of d. Indeed, it can be viewed simply as a formal definition of a *mean*. Such an approach to generate *entropic means*, with ϕ-divergence or/and Bregman divergences has been introduced and developed in [1] where more details and examples can be found. The ability to carry out a fast and efficient computation of centroids is key to successful implementation of k–means like algorithms. For the function d given by (2.10) and a non empty cluster π with p vectors the coordinate $\mathbf{c}[j]$ of the centroid $\mathbf{c} = \mathbf{c}(\pi)$ is

$$\mathbf{c}[j] = \begin{cases} \prod\limits_{\mathbf{x} \in \pi} (\mathbf{x}[j])^{\frac{1}{p}} & \text{if} \quad \mathbf{x}[j] > 0 \text{ for each } \mathbf{x} \in \pi, \\ 0 & \text{otherwise.} \end{cases} \tag{2.5}$$

(for details see Section 5).

While fast and computationally efficient, the batch k–means often gets trapped at local minima even for simple one dimensional data sets [7]. Next we show that Algorithm 2.1 shares this deficiency.

Example 2.1 *Consider a three scalar set* $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$ *where* $\mathbf{x}_1 = 1$, $\mathbf{x}_2 = e$, *and* $\mathbf{x}_3 = e^2$. *If the initial partition* $\Pi = \{\pi_1, \pi_2\}$ *where* $\pi_1 = \{\mathbf{x}_1\}$, *and* $\pi_2 = \{\mathbf{x}_2, \mathbf{x}_3\}$,



then $\mathbf{c}_1 = 1$, $\mathbf{c}_2 = e^{\frac{3}{2}}$, *and since*

$$\begin{aligned} d(\mathbf{c}_1, \mathbf{x}_2) = -2 + e &> d(\mathbf{c}_2, \mathbf{x}_2) = -0.5e^{\frac{3}{2}} + e \\ d(\mathbf{c}_1, \mathbf{x}_3) = -3 + e^2 &> d(\mathbf{c}_2, \mathbf{x}_3) = -1.5e^{\frac{3}{2}} + e^2 \end{aligned}$$

a batch iteration of the algorithm does not change the partition. Note that $Q(\Pi) \approx 1.1440$, *and the quality of the partition* $\Pi^o = \{\{\mathbf{x}_1, \mathbf{x}_2\}, \{\mathbf{x}_3\}\}$ *is approximately* 0.4208. *While* $Q(\Pi^o) < Q(\Pi)$ *the batch version of the algorithm misses the better partition.*

An incremental version of k–means helps remedy the problem. To describe the algorithm 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 largest decrease of the objective function.

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

$$Q(\texttt{nextFV}(\Pi)) \leq Q(\Pi'). \tag{2.6}$$

We note that for the partition Π given by Example 2.1 one has $\Pi^o = \texttt{nextFV}(\Pi)$.

To benefit from the speed of batch iterations and the accuracy of the incremental iterations we implement the following "ping–pong" strategy.

Algorithm 2.2 The Entropic Geometric Means (EGM) clustering algorithm.
For user supplied tolerances $\texttt{tol}_\texttt{B} > 0$ *and* $\texttt{tol}_\texttt{I} > 0$ *do the following:*

1. *Set the index of iteration* $t = 0$.

2. *Start with an arbitrary partitioning* $\Pi^{(t)} = \left\{\pi_1^{(t)}, \ldots, \pi_k^{(t)}\right\}$.

3. *Generate the partition* $\Pi^{(t+1)} = \texttt{nextB}\left(\Pi^{(t)}\right)$.
   if $\left[Q\left(\Pi^{(t)}\right) - Q\left(\texttt{nextB}\left(\Pi^{(t)}\right)\right) > \texttt{tol}_\texttt{B}\right]$
   set $t = t + 1$
   goto Step 3

4. *Generate the partition* $\Pi^{(t+1)} = \texttt{nextFV}\left(\Pi^{(t)}\right)$.
   if $\left[Q\left(\Pi^{(t)}\right) - Q\left(\texttt{nextFV}\left(\Pi^{(t)}\right)\right) > \texttt{tol}_\texttt{I}\right]$
   set $t = t + 1$
   goto Step 3

5. *Stop.*

A single iteration of Algorithm 2.2 applied to the partitions Π of Example 2.1 generates the optimal partition Π^o. In the next section we present and discuss partitions generated by the EGM algorithm.

At the end of this section we briefly compare and explain the differences between Algorithm 2.2 with two algorithms introduced recently by [2] and [9]. Both papers consider a set of vectors $\mathbf{X}$ in $\mathbf{R}^n$ so that each $\mathbf{x} \in \mathbf{X}$ has non–negative coordinates and belongs to the unit simplex, i.e.,

$$\mathbf{x}[j] \geq 0, j = 1, \ldots, n, \text{ and } \sum_{j=1}^{n} \mathbf{x}[j] = 1. \tag{2.7}$$

In both papers the "distance" between a vector $\mathbf{x}$ and a centroid $\mathbf{c}$ is given by

$$d_1(\mathbf{c}, \mathbf{x}) = \sum_{j=1}^{n} \mathbf{x}[j] \log \frac{\mathbf{x}[j]}{\mathbf{c}[j]} \tag{2.8}$$

(for definitions of quantities $a \log \frac{a}{0}$ and $0 \log \frac{0}{a}$ see Section 5). The Divisive Information Theoretic clustering algorithm proposed in [9] is the batch version of the classical k–means algorithm with the squared Euclidean distance substituted by d_1, and the algorithm proposed in [2] is its incremental counterpart. With this choice, the resulting centroid $\mathbf{c}(\pi)$ of a cluster π is the arithmetic mean of the vectors in the cluster (see e.g., [9]).

However, we note that (as explained in Section 2), the more stringent unit simplex constraint can be avoided and replaced by the larger non-negative orthant, by simply using the normalized entropy

$\phi(t) = t \log t - t + 1$, and replacing (2.8), with the resulting

$$d_2(\mathbf{c}, \mathbf{x}) = \sum_{j=1}^{n} \mathbf{x}[j] \log \frac{\mathbf{x}[j]}{\mathbf{c}[j]} + \sum_{j=1}^{n} \mathbf{c}[j] - \sum_{j=1}^{n} \mathbf{x}[j] \tag{2.9}$$

It is easy to verify that this leads to the same results, namely the resulting centroid $\mathbf{c}(\pi)$ of a cluster π is the arithmetic mean of the vectors in the cluster (see [15]).

As previously explained the entropy-like distances are not necessarily symmetric, i.e., in general $d(\mathbf{c}, \mathbf{x}) \neq d(\mathbf{x}, \mathbf{c})$. The "entropy-like" distance function d we use

$$d(\mathbf{c}, \mathbf{x}) = \sum_{j=1}^{n} \mathbf{c}[j] \log \frac{\mathbf{c}[j]}{\mathbf{x}[j]} + \sum_{j=1}^{n} \mathbf{x}[j] - \sum_{j=1}^{n} \mathbf{c}[j]. \tag{2.10}$$

can be obtained by switching the roles of $\mathbf{x}$ and $\mathbf{c}$ in d_2. Alternatively, d_2 can also be obtained by using the kernel $\phi(t) = -\log t + t - 1$ in (1.1). Note that both choices of ϕ, lead to an entropy-like distance.[2] This provides the motivation for the entropic part of the name of the EGM clustering algorithm. The key difference with d_2 results from the fact that the corresponding optimization problem (1.1) has a completely different objective function, with resulting minimizer (the centroid of a cluster π) being now the *geometric mean* of the vectors in the cluster, hence the use of the geometric terminology in EGM.

3. Experimental results

We consider the dataset which is a merger of the three document collections available at `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).

We select 600 "best" terms and build vectors of dimension 600 for each document (see [7] for selection procedure 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 in [7]. The Singular Value Decomposition based algorithm is first applied to all unit l_2 document vectors to split the collection into two clusters. The same procedure is then applied recursively to a largest cluster in the partition. Clustering results for three cluster partition are reported in Table 1. When the number of terms is relatively small, some documents may contain

[2] To avoid confusion and controversies (already existing in the related fields of Information Theory, e.g., Shannon versus Burg entropy), we could also call $d(c, x)$ the "forward" entropy-like distance (in the order of the minimization variable $\mathbf{c} \to \mathbf{x}$), and d_2 the "backward" entropy-like distance (in the order $\mathbf{x} \to \mathbf{c}$), where d is defined by (2.1), and ϕ the normalized entropy as defined in (2.2).

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

no selected terms, and their corresponding vectors are zeros. 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 Table 1). The entry ij in Table 1 is the number of documents that belong to cluster i and document collection j. The "confusion" matrix shows that only 68 documents (i.e., less that 2% of the entire collection) have been "misclassified" by the algorithm.

At the second step we would like to use the three cluster partition generated by sPDDP as the initial partition of the EGM means algorithm. First we look for conditions necessary for successful implementation of this two step clustering strategy.

Consider a partition $\Pi = \{\pi_1, \ldots, \pi_k\}$, and assume that the following condition holds for the cluster π_1

$$\forall j = 1, \ldots, n \; \exists \mathbf{x} \in \pi_1 \text{ such that } \mathbf{x}[j] = 0. \tag{3.1}$$

Note, that in terms of "words and documents" condition (3.1) means that there is no word that occurs in *all* the documents of cluster π_1. This is typical for large clusters, and this condition holds for all three clusters of the partition displayed by Table 1. Due to (2.5) one has $\mathbf{c}_1 = \mathbf{c}(\pi_1) = 0$. If (3.1) holds for *all* k clusters of the partition, then $\mathbf{c}_1 = \ldots = \mathbf{c}_k = 0$, and an application of a batch iteration of the EGM clustering algorithm does not change Π. This observation motivates the following definition.

Definition 3.1 *A cluster π is* `full`*, if there exists j such that $\mathbf{x}[j] > 0$ for each $\mathbf{x} \in \pi$.*

In terms of "words and documents" a cluster is full if there is a word that occurs in every document of the cluster. Centroids of full clusters are nonzero vectors, and, as discussed above, an application of a batch iteration of the EGM algorithm may not change a partitions with no full clusters.

We now turn to incremental iterations of the EGM algorithm and consider the following condition for a cluster π.

$$\forall j = 1, \ldots, n \; \exists \mathbf{x}, \; \mathbf{x}' \in \pi \text{ such that } \mathbf{x}[j] = \mathbf{x}'[j] = 0. \tag{3.2}$$

This condition, like condition (3.1), is easily fulfilled by large clusters. Moreover, if one vector is removed from π, then the resulting cluster π' satisfies (3.1). We assume now that condition (3.2) holds for each cluster of Π, and construct a first variation Π' of Π by selecting $\pi_i,\ \pi_j \in \Pi$, removing a vector $\mathbf{x} \in \pi_i$ and assigning $\mathbf{x}$ to π_j. Denoting the resulting clusters by π_i' and π_j' one gets

$$Q(\Pi) - Q(\Pi') = [q(\pi_i) - q(\pi_i')] + \left[q(\pi_j) - q\left(\pi_j'\right)\right] = 0.$$

Next we define a cluster that can be changed by incremental iterations of the EGM.

Definition 3.2 *A cluster π is* `almost full`*, if there exists j such that $\mathbf{x}[j] > 0$ for all but one $\mathbf{x} \in \pi$.*

An application of a first variation iteration of the EGM algorithm may not change a partitions with no almost full clusters. An inspection shows that the partition presented by Table 1 contains no full clusters, or almost full clusters. Hence an application of EGM to the partition will not change it. To generate initial partitions that can be modified by the EGM algorithm one should either (a) increase the number of selected terms, or (b) increase the number of clusters (and thereby reduce cluster size), or (c) both.

First we increase the vector space dimension to 1600, and apply sPDDP until at least one cluster in the sPDDP generated partition becomes almost full. This happens for the first time when the number of clusters is 39. The number of documents "misclassified" by sPDDP is 45, and application of the EGM generates 39 clusters (note that k–means like algorithms may decrease number of clusters, but it does not happen in this experiment) with 49 "misclassifications". EGM performs 2 batch and 2 first variation iterations, and "reshuffles" 60 vectors (see Table 2 for confusion matrices and, for example, compare cluster 37 in both partitions). Note that while 60 vectors change their cluster affiliation the number of "misclassifications" as reported by the confusion matrix grows by 4 only. The EGM clustering algorithm tends not to "mix" documents from different collections, and moves most of the documents between clusters of the same document collection.

We gradually increase the number of clusters and report the results in Table 3. Inspection of the results reveals the following:

1. The number of full and almost full clusters generated by EGM grows monotonically as the number of clusters generated by sPDDP increases.

2. As the number of clusters generated by sPDDP reaches 301, EGM empties some of the clusters.

3. As the number of EGM generated clusters grows, the number of "misclassified" documents grows from 49 to 387, and then falls down to 123.

Cluster/DocCol	DC0	DC1	DC2
cluster 0	116	0	0
cluster 1	0	0	48
cluster 2	1	0	123
cluster 3	1	0	65
cluster 4	0	0	116
cluster 5	0	0	88
cluster 6	1	2	87
cluster 7	0	0	126
cluster 8	2	0	82
cluster 9	0	0	73
cluster 10	0	1	64
cluster 11	0	1	159
cluster 12	0	0	146
cluster 13	0	0	66
cluster 14	1	1	140
cluster 15	8	101	1
cluster 16	0	121	0
cluster 17	2	105	1
cluster 18	0	51	0
cluster 19	0	117	1
cluster 20	1	118	4
cluster 21	0	57	0
cluster 22	0	112	0
cluster 23	0	147	0
cluster 24	0	76	0
cluster 25	1	91	1
cluster 26	1	84	0
cluster 27	1	98	1
cluster 28	1	114	3
cluster 29	5	61	1
cluster 30	108	0	0
cluster 31	64	0	0
cluster 32	141	0	0
cluster 33	107	0	0
cluster 34	79	0	0
cluster 35	81	1	1
cluster 36	52	0	0
cluster 37	109	1	0
cluster 38	150	0	1
"empty" documents			
cluster 39	0	0	0

Cluster/DocCol	DC0	DC1	DC2
cluster 0	115	0	0
cluster 1	0	0	48
cluster 2	1	0	123
cluster 3	1	0	65
cluster 4	0	0	116
cluster 5	0	0	88
cluster 6	1	2	87
cluster 7	0	0	126
cluster 8	2	0	82
cluster 9	0	0	73
cluster 10	0	1	64
cluster 11	0	1	159
cluster 12	0	0	146
cluster 13	0	0	66
cluster 14	1	1	140
cluster 15	8	101	1
cluster 16	0	121	0
cluster 17	2	105	1
cluster 18	0	51	0
cluster 19	0	117	1
cluster 20	1	118	4
cluster 21	0	57	0
cluster 22	0	112	0
cluster 23	0	147	0
cluster 24	0	76	0
cluster 25	1	91	1
cluster 26	1	83	0
cluster 27	1	98	1
cluster 28	1	114	3
cluster 29	5	61	1
cluster 30	107	0	0
cluster 31	63	0	0
cluster 32	141	0	0
cluster 33	107	0	0
cluster 34	77	0	0
cluster 35	62	1	1
cluster 36	30	0	0
cluster 37	150	2	0
cluster 38	155	0	1
"empty" documents			
cluster 39	0	0	0

Table 2: Confusion matrices for partitions generated by sPDDP (left) and sPDDP+EGM (right) algorithms.

dim	sPDDP				sPDDP and EGM				
	# clusts	# full clusts	# almost full clusts	misclass docs	# clusts	# full clusts	# almost full clusts	misclass docs	# vec moved
1600	39	0	1	45	39	1	0	49	60
1600	45	0	2	45	45	4	0	49	550
1600	49	0	3	45	45	5	0	53	694
1600	55	0	4	45	55	11	0	387	2434
1600	57	1	4	45	57	14	0	371	2719
1600	301	162	116	45	271	262	72	197	4483
1600	525	448	369	45	482	480	165	139	3344
1600	712	602	541	45	597	596	246	123	3283

Table 3: Outcome of sPDDP and sPDDP+EGM algorithms.

4. Future research

The paper introduces a k–means like algorithm that combines features of both batch and incremental versions of the classical k–means clustering algorithm. The "entropy-like" distance function proposed in the paper selects the geometric mean of a cluster as the cluster's centroid. The EGM clustering algorithm tends to build small clusters of documents that contain the same words.

We believe the algorithm can contribute significantly to clustering large datasets and building high quality clusters of any size when implemented as a part of a "hybrid scheme" (see [14]), i.e. being an element of a sequence of clustering algorithms so that the output of algorithm i becomes input to algorithm $i+1$. To clarify the potential usefulness of the EGM algorithm consider the following hybrid scheme:

1. Fast SVD type divisive algorithm (like, for example, PDDP [4], or sPDDP [7]) that generates small clusters.

2. EGM clustering algorithm that refines clusters generated by Step 1.

3. Treat small high quality clusters generated by Step 2 as a single point with an appropriate weight (see e.g. [19], [20]). Apply any k–means like algorithm to the "new" dataset.

If Step 1 of the scheme manages to build clusters of average size 10, then the size of the "new" dataset is of an order of magnitude smaller than the size of the original dataset and the proposed scheme is a scalable clustering procedure. Good quality small clusters are crucial for building good final partitions by Step 3.

References

[1] A. Ben-Tal, A. Charnes and M. Teboulle. Entropic means. *Journal of Mathematical Analysis and Applications*, 139:537–551, 1989.

[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] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, New Jersey, 1989.

[4] D. L. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[5] L.M. Bregman. A relaxation method of finding a common point of convex sets and its application to the solution of problems in convex programming. *USSR Comp. Math. and Math Phys.*, 7:200–217, 1967.

[6] I. Csiszar. Information-type measures of difference of probability distributions and indirect observations. *Studia Sci. Mat. Hungar.*, 2:299–318, 1967.

[7] 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.

[8] 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.

[9] 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.

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

[11] B.F. Iusem, A.N. Svaiter and Teboulle M. Entropy-like methods in convex programming. *Mathematics of Operations Research*, 19:790–814, 1994.

[12] K.C. Kiwiel. Proximal minimization methods with generalized bregman functions. *SIAM J. Control and Optimization*, 35:1142–1168, 1997.

[13] J. Kogan and Teboulle M. A unified framework for clustering data with entropy-like means algorithms: theory and applications. *in preparation.*

[14] J. Kogan, C. Nicholas, and V. Volkovich. Text mining with hybrid clustering schemes. In M.W.Berry and W.M. Pottenger, editors, *Proceedings of the Workshop on Text Mining (held in conjunction with the Third SIAM International Conference on Data Mining)*, pages 5–16, 2003.

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

[16] A. L. Peressini, F .E. Sullivan, and J. J. Uhl. *The Mathematics of Nonlinear Programming.* Springer–Verlag, New York, 1988.

[17] 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.

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

[19] G. Zhang, B. Kleyner and M. Hsu. A local search approach to k-clustering. *Tech Report HPL-1999-119*, 1999.

[20] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, Montreal, Canada, 1996.

5. Appendix

In this section we present centroid computations for the EGM clustering algorithm. The general approach to the optimization problem (1.1) is provided in [15]. For the sake of completeness we outline briefly centroid computation for the special case when the "distance–like" function is given by (2.10). Motivated by continuity arguments for each $a \geq 0$ and $b > 0$ we define

$$0 \log \frac{0}{a} = 0 \log \frac{a}{0} = 0 \text{ and } b \log \frac{b}{0} = \infty. \tag{5.1}$$

For a cluster $\pi = \{\mathbf{x}_1, \ldots, \mathbf{x}_p\}$ we have to find $\mathbf{c} \in \mathbf{R}^n_+$ that minimizes the convex function in $\mathbf{c}$,

$$\sum_{\mathbf{x}\in\pi} d(\mathbf{c}, \mathbf{x}) = \sum_{\mathbf{x}\in\pi} \left[\sum_{j=1}^{n} \mathbf{c}[j] \log \frac{\mathbf{c}[j]}{\mathbf{x}[j]} - \mathbf{c}[j] + \mathbf{x}[j] \right].$$

The right hand side of the equation is

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

Keeping in mind that $\sum_{\mathbf{x}\in\pi} \sum_{j=1}^{n} \mathbf{x}[j]$ is independent of $\mathbf{c}$, for a fixed j we consider

$$-p\mathbf{c}[j] + \left(\sum_{\mathbf{x}\in\pi} \mathbf{c}[j] \log \frac{\mathbf{c}[j]}{\mathbf{x}[j]} \right)$$

the contribution of the j^{th} vector component to $\sum_{\mathbf{x}\in\pi} d(\mathbf{c}, \mathbf{x})$. Keeping in mind that $\mathbf{c}[j]$ is a scalar, to simplify the exposition, we substitute $\mathbf{c}[j]$ by s and minimize

$$-ps + \sum_{\mathbf{x}\in\pi} s \log \frac{s}{\mathbf{x}[j]} \tag{5.2}$$

with respect to s. Due to (5.1) the expression is minimized at 0 if $\mathbf{x}[j] = 0$ for at least one $\mathbf{x} \in \pi$. If $\mathbf{x}[j] > 0$ for each $\mathbf{x} \in \pi$, then the derivative of (5.2) should vanish at the minimizer, i.e.,

$$\sum_{\mathbf{x}\in\pi} \log \frac{s}{\mathbf{x}[j]} = 0, \text{ and } s = (\mathbf{x}_1[j])^{\frac{1}{p}} \ldots (\mathbf{x}_p[j])^{\frac{1}{p}}.$$

We remind the reader that s stands for $\mathbf{c}[j]$, and the last expression for s is exactly the formula given by (2.5).

Finally we consider a set of p positive scalars $s_i > 0$, $i = 1, \ldots, p$ and the geometric mean $s = s_1^{\frac{1}{p}} \cdot \ldots \cdot s_p^{\frac{1}{p}}$. The following elementary properties will be useful for evaluation of $Q(\Pi)$:

$$s_1 + \ldots + s_p - p \cdot s = s_1 + \ldots + s_p - p \cdot s_1^{\frac{1}{p}} \cdot \ldots \cdot s_p^{\frac{1}{p}} \geq 0 \tag{5.3}$$

(this is the Arithmetic–Geometric Mean inequality, see e.g. [16]).

$$\sum_{i=1}^{p} s \log \frac{s}{s_i} = s \cdot p \left[\log s - \frac{1}{p} \sum_{i=1}^{p} \log s_i \right] = s \cdot p \left[\log s - \log s \right] = 0. \tag{5.4}$$

The quality of a cluster π with p vectors is given by

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

For a fixed j the nonzero coordinate $\mathbf{c}[j] = \mathbf{x}_1[j]^{\frac{1}{p}} \cdot \ldots \cdot \mathbf{x}_p[j]^{\frac{1}{p}}$, hence, due to (5.4) one has $\sum_{\mathbf{x} \in \pi} \mathbf{c}[j] \log \frac{\mathbf{c}[j]}{\mathbf{x}[j]} = 0$, and $\sum_{j=1}^{n} \left(\sum_{\mathbf{x} \in \pi} \mathbf{c}[j] \log \frac{\mathbf{c}[j]}{\mathbf{x}[j]} \right) = 0$. The formula for $q(\pi)$ simplifies as follows:

$$q(\pi) = -p\mathbf{e}^T\mathbf{c} + \sum_{\mathbf{x} \in \pi} \mathbf{e}^T\mathbf{x}, \tag{5.5}$$

and by definition of q (or by (5.3)), one has $q(\pi) \geq 0$.