Efficient Online Spherical K-means Clustering

Shi Zhong

Department of Computer Science and Engineering
Florida Atlantic University, Boca Raton, FL 33431
Email: zhong@cse.fau.edu

Abstract— **The spherical k-means algorithm, i.e., the k-means algorithm with cosine similarity, is a popular method for clustering high-dimensional text data. In this algorithm, each document as well as each cluster mean is represented as a high-dimensional unit-length vector. However, it has been mainly used in batch mode. That is, each cluster mean vector is updated, only after all document vectors being assigned, as the (normalized) average of all the document vectors assigned to that cluster. This paper investigates an online version of the spherical k-means algorithm based on the well-known Winner-Take-All competitive learning. In this online algorithm, each cluster centroid is incrementally updated given a document. We demonstrate that the online spherical k-means algorithm can achieve significantly better clustering results than the batch version, especially when an annealing-type learning rate schedule is used. We also present heuristics to improve the speed, yet almost without loss of clustering quality.**

I. INTRODUCTION

Document clustering has become an increasingly important technique for unsupervised document organization, automatic topic extraction, and fast information retrieval or filtering. For example, a web search engine often returns thousands of pages in response to a broad query, making it difficult for users to browse or to identify relevant information. Clustering methods can be used to automatically group the retrieved documents into a list of meaningful categories. Similarly, a large database of documents can be pre-clustered to facilitate query processing by searching only the cluster that is closest to the query. In these applications, clustering efficiency is desirable due to real-time requirements or high data volumes.

If the popular vector space ("bag-of-words") representation is used, a text document, after suitable pre-processing, gets mapped to a high dimensional vector with one dimension per "term" [1]. Such vectors tend to be very sparse, and they have only non-negative entries. Moreover, it has been widely observed that the vectors have directional properties, i.e., the length of the vector is much less important than their direction. This has led to the widespread practice of normalizing the vectors to unit length before further analysis, as well as to the use of the cosine between two vectors as a popular measure of similarity between them.

Our previous comparative study [2] empirically verified that using unit-length term frequency-inverse document frequency (TF-IDF) vectors leads to better clustering results than simply representing a document by multivariate Bernoulli or multinomial models. It is also shown that the spherical k-means algorithm (SPKM) [3] is one of the most efficient (i.e., fastest) document clustering algorithms, but not of the best quality.

Our main objective in this paper is to significantly improve the clustering quality for k-means, yet to retain its efficiency.

A traditional way of improving the quality of k-means is to employ better optimization (annealing) techniques (e.g., [4], [5]) to avoid (bad) local optima. However, these methods are usually too time-consuming to be valuable in practice. Local search was used in [6] to improve the objective function value for spherical k-means. The algorithm tries to find a data point, and by switching it to a different cluster, to improve the spherical k-means objective function. This approach, however, works favorably only for small text data sets. For large clusters, the "first variation" technique is unlikely to be effective, as admitted in [6]. In addition, the proposed local search algorithm seems to be computationally costly for clustering large text data sets.

Another approach is to replace the batch update of cluster centroids by an online learning strategy. The standard k-means usually update cluster centroids in batch mode, meaning all centroids are updated after going through (partitioning into clusters) all data points. In contrast, competitive learning approaches adjust one or more cluster centroids for each given data input. The "Winner-Take-All (WTA)" strategy updates only one cluster centroid per data input, and is more efficient than soft competitive learning methods such as Self-Organizing Map [7] and Neural-Gas [8], which adjust all centroids accordingly for every given input.

It has been shown that using online updates leads to improved clustering results [8], [9]. For example, Banerjee and Ghosh [9] combined frequency sensitive competitive learning [10], a method to reduce empty clusters (which is a problem plaguing WTA-type learning, especially when the number of clusters is large), with SPKM to achieve more balanced clusters. Using a conscience mechanism, they penalize the probability of a data point going to an already crowded cluster, and reported better clustering quality with this strategy.

Besides the efficiency and quality issues, online algorithms are an important and effective way of analyzing streaming data [9], which is another reason that we investigate online spherical k-means algorithms.

The approach advocated in this paper is most similar to the incremental algorithm proposed in [9]. However, there are significant differences. First, the focus in [9] is different and mainly on achieving non-empty balanced clusters rather than efficiency and/or quality. Second, we use different learning rate schedules. The online update of cluster centroids can be viewed as a gradient ascent approach—the cluster centroids

(parameters) are updated following the gradient direction. The learning rate used in [9] is effectively inversely proportional to the size of a cluster, aiming to balance clusters. We compare two different rate schedules: a flat/static learning rate and a gradually decreasing (annealing type) learning rate, and observe the latter is much superior. Finally, we present implementation tricks to significantly improve the speed of WTA-type spherical k-means, without quality degradation. Third, there are no time results provided and no comparison with other efficient algorithms (e.g., CLUTO [11]) in [9]. Experimental results show that our online SPKM approach is faster than CLUTO and achieves comparable quality.

Borgelt and Nürnberger [12] presented a competitive learning version of the fuzzy c-means clustering algorithm. The difference from our work is that they retained the use of Gaussian models (i.e., Euclidean distances and covariance matrices), which makes the proposed algorithm inefficient for high dimensional text data.

For notation in this paper, we use bold face variables (e.g., $\mathbf{x}$, μ) to represent vectors, upper case alphabets (e.g., $\mathcal{X}$, $\mathcal{Y}$) to represent sets, $\|\cdot\|$ to denote the L_2 norm, and $|\cdot|$ to denote the number of elements in a set.

II. SPHERICAL K-MEANS (SPKM)

A. Standard K-means

The objective of standard k-means clustering [13], [14] is to minimize the mean-squared error

$$E = \frac{1}{N} \sum_{\mathbf{x}} \|\mathbf{x} - \mu_{k(\mathbf{x})}\|^2 \ , \tag{1}$$

where $k(\mathbf{x}) = \arg\min_{k \in \{1,...,K\}} \|\mathbf{x} - \mu_k\|$ is the index of the closest cluster centroid to $\mathbf{x}$, N is the total number of data vectors. It is well-known that the underlying probability distribution for the standard k-means algorithm is Gaussian [15]. A more accurate description of the relationship between k-means and mixture-of-Gaussians is provided in [16], viewing k-means and EM clustering as two stages of an annealing process.

B. K-means on a Hypersphere

For high-dimensional data such as text documents (represented as TF-IDF vectors) and market baskets, cosine similarity has been shown to be a superior measure to Euclidean distance [17]. The implication is that the direction of a document vector is more important than the magnitude. This leads to a unit-vector representation, i.e., each document vector is normalized to be of unit length ($\|\mathbf{x}\| = 1$). Let $\{\mu_1, ..., \mu_K\}$ be a set of unit-length centroid vectors, the spherical k-means algorithm (i.e., k-means on a unit hypersphere) aims to maximize the average cosine similarity objective

$$L = \sum_{\mathbf{x}} \mathbf{x}^T \mu_{k(\mathbf{x})} \ , \tag{2}$$

where $k(\mathbf{x}) = \arg\max_k \mathbf{x}^T \mu_k$. The batch version of SPKM again iterates between a data assignment step and a mean re-estimation step (Fig. 1). It can also be derived from the mixture of von Mises-Fisher distributions [9]. The main difference from standard k-means is that the re-estimated mean vectors $\{\mu\}$ need to be normalized to unit-length and the underlying probabilistic models are not Gaussian.

Algorithm: spherical k-means (SPKM)
Input: A set of N *unit-length* data vectors $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ in $I\!\!R^d$ and the number of clusters K.
Output: A partition of the data vectors given by the cluster identity vector $\mathcal{Y} = \{y_1, ... y_N\}$, $y_n \in \{1, ..., K\}$.
Steps:
1. Initialization: initialize the *unit-length* cluster centroid vectors $\{\mu_1, ..., \mu_K\}$;
2. Data assignment: for each data vector $\mathbf{x}_n$, set $y_n = \arg\max_k \mathbf{x}_n^T \mu_k$;
3. Centroid estimation: for cluster k, let $\mathcal{X}_k = \{\mathbf{x}_n | y_n = k\}$, the centroid is estimated as $\mu_k = \sum_{\mathbf{x} \in \mathcal{X}_k} \mathbf{x} / \| \sum_{\mathbf{x} \in \mathcal{X}_k} \mathbf{x}\|$;
4. Stop if $\mathcal{Y}$ does not change, otherwise go back to Step 2a.

Fig. 1. Spherical k-means algorithm.

An interesting observation is that, when $\mathbf{x}$ and μ are both unit vectors, it is equivalent to use cosine similarity or Euclidean distance for data assignment. The reason is that

$$\|\mathbf{x} - \mu\|^2 = \|\mathbf{x}\|^2 + \|\mu\|^2 - 2\mathbf{x}^T \mu = 2 - 2\mathbf{x}^T \mu \ .$$

That is, on a hypersphere, maximizing (2) is equivalent to minimizing (1). As we will see in the next section, this leads to slightly different update formula for the online spherical k-means algorithm.

III. ONLINE SPHERICAL K-MEANS (OSKM)

A. Competitive Learning

Competitive learning techniques [18] let multiple cells compete for a given input and the winner to adjust itself to respond more strongly to the input. The "Winner-Take-All" mechanism is the simplest strategy and allows only one winner per input. When applied to clustering (with cluster centroids as cells), it leads to an online k-means clustering algorithm. There are other soft competitive learning methods [7], [8] that allow multiple winners to adjust, with the rate of adjustment inversely proportional to the distance (or some function of the distance) between each winner and the given input. For example, the Neural-Gas algorithm [8] update cluster centroids according to the rank of being close to a given data point. It is reportedly one of the best competitive learning techniques [8] in minimizing the MSE objective function (1).

B. Online Spherical K-means

In online spherical k-means clustering, to maximize the objective (2), given a data point $\mathbf{x}$, we incrementally update the closest cluster center $\mu_{k(\mathbf{x})}$ as:

$$\mu_{k(\mathbf{x})}^{(new)} = \frac{\mu_{k(\mathbf{x})} + \eta \frac{\partial L_{\mathbf{x}}}{\partial \mu_{k(\mathbf{x})}}}{\|\mu_{k(\mathbf{x})} + \eta \frac{\partial L_{\mathbf{x}}}{\partial \mu_{k(\mathbf{x})}}\|} = \frac{\mu_{k(\mathbf{x})} + \eta \ \mathbf{x}}{\|\mu_{k(\mathbf{x})} + \eta \ \mathbf{x}\|} \ , \tag{3}$$

where η is the learning rate and $L_{\mathbf{x}} = \mathbf{x}^T \mu_{k(\mathbf{x})}$. Note the normalization is necessary to keep the updated μ on the unit hypersphere. The derivative term on the right hand side is basically the (scaled) gradient of the $\mathbf{x}$-related part of the objective function with respect to $\mu_{k(\mathbf{x})}$. This is essentially an incremental gradient ascent algorithm.

Algorithm: online spherical k-means (OSKM)
Input: A set of N *unit-length* data vectors $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ in $\mathbb{R}^d$ and the number of clusters K.
Output: A partition of the data vectors given by the cluster identity vector $\mathcal{Y} = \{y_1, ... y_N\}$, $y_n \in \{1, ..., K\}$.
Steps:
   1. Initialization: initialize the *unit-length* cluster centroid vectors $\{\mu_1, ..., \mu_K\}$, $\quad t = 0$;
   2. For $m = 1$ to M
      For $n = 1$ to N
      (a) For each data vector $\mathbf{x}_n$, find the closest centroid $y_n = \arg\max_k \mathbf{x}_n^T \mu_k$;
      (b) Estimate each centroid as $\mu_{y_n} \leftarrow \frac{\mu_{y_n} + \eta^{(t)} \mathbf{x}_n}{\|\mu_{y_n} + \eta^{(t)} \mathbf{x}_n\|}$;
      (c) $t \leftarrow t + 1$;

Fig. 2.   Online spherical k-means algorithm.

Before discussing the choice of learning rate η, let us digress for a moment and explain a difference in [9], where the update formula is $\mu_{k(\mathbf{x})}^{(new)} = \frac{\mu_{k(\mathbf{x})} + \eta\ (\mathbf{x} - \mu_{k(\mathbf{x})})}{\|\mu_{k(\mathbf{x})} + \eta\ (\mathbf{x} - \mu_{k(\mathbf{x})})\|}$. The second term in the numerator can be viewed as a derivative of part of the objective function in (1) w.r.t. $\mu_{k(\mathbf{x})}$. Even though these two updates are equivalent after normalization of the centroid vectors, the first one (3) seems to be more interpretable (and save some computation). After all, a cluster is assumed to follow a von Mises-Fisher distribution instead of a Gaussian distribution [9].

The learning rate used in [9] is effectively $\eta = 1/|\mathcal{X}_{k(\mathbf{x})}|$, which is inversely proportional to the cluster size. Since our focus is not to balance clusters, we compare two other rate schedules—a flat rate (e.g., $\eta = 0.05$) and an exponentially decreasing rate $\eta^{(t)} = \eta_0 \left(\frac{\eta_f}{\eta_0} \right)^{\frac{t}{NM}}$, where t is the online iteration index ($0 \leq t \leq NM$) and M is the number of batch iterations. The exponential learning rate schedule has been shown to work better than the flat learning rate [19], due to an annealing effect introduced by the gradual decrease of η. Our results verifies that this annealing-type learning rate schedule leads to significantly better clustering results than the flat rate, as well as than the batch SPKM.

C. Efficient Implementation

This section describes several implementation details of the online spherical k-means algorithm that help improve the efficiency and avoid empty clusters.

Deferred normalization: Readers familiar with text data analysis may already notice that the computational bottleneck for OSKM is with the centroid vector normalization step (Step 2(b) in Fig. 2). Since each document vector is usually very sparse, the complexity of addition $\mu + \eta\ \mathbf{x}$ is

linear in the number of nonzeros in $\mathbf{x}$, much lower than d— the dimensionality of $\mathbf{x}$. The complexity of the normalization, however, is $O(d)$ since the centroid vector μ is usually not sparse. Thus the total complexity for estimating μ's is $O(MNd)$. This is very costly since d is the size of word vocabulary, possibly even larger than N for many text data sets. In contrast, one can easily verify that, with a sparse matrix representation, the total complexity for the data assignment step is just $O(MKN_{nz})$, where N_{nz} is the total number of nonzeros in the whole term-document matrix and usually $N_{nz} \ll Nd$. The number of clusters K is often set *a priori*.

One way of removing this computational bottleneck is to defer the normalization. We keep a record for every μ's L_2 norm (and the square $s(\mu) = \|\mu\|^2$) and don't normalize any μ until its norm gets too large (e.g., close to overflow). As a result, we change the cosine similarity calculation in the assignment step to $\frac{\mathbf{x}^T \mu}{\|\mu\|}$ and the centroid estimation step to the following substeps:

$$s(\mu) \leftarrow s(\mu) - \sum_{j \in J} \mu(j)^2 ,$$
$$\mu(j) \leftarrow \mu(j) + \eta \mathbf{x}_j \|\mu\|, \forall j \in J ,$$
$$s(\mu) \leftarrow s(\mu) + \sum_{j \in J} \mu(j)^2 ,$$
$$\|\mu\| \leftarrow \sqrt{s(\mu)} ,$$

where J is the set of indices for nonzero entries in $\mathbf{x}$, $\mu(j)$ is the j-th dimension of μ, and $s(\mu) = \|\mu\|^2$. Note all these substeps together only takes time linear in the number of nonzeros in $\mathbf{x}$ since only those dimensions in μ get updated. The total complexity for computing μ's now becomes $O(MN_{nz})$, which is a huge improvement over $O(MNd)$. The total time complexity for OSKM is now $O(MKN_{nz})$.

Sampling: Let us consider the online spherical k-mean algorithm with the annealing-type learning rate schedule. Initially, the learning rate is high and the centroids will be moving around a lot and get spreaded out globally in the data space. As the rate decreases, each mean gets refined and starts adapting to the local data structure. We conjecture that many initial moves are redundant and a small random sample would be enough to achieve the same spreading-out effect. Therefore, we propose the following sampling strategy: at the m-th batch iteration (suppose we go through all data points M times), we randomly sample $\frac{mN}{M}$ data points to do the online learning for μ's. Our experiments show that this strategy can double the clustering speed while not damaging the quality of clustering results.

Avoiding Empty Clusters: It is well-known that the standard k-means algorithm, either batch version or online version, tends to generate empty clusters (due to the greed optimization strategy). The situation deteriorates further as number of clusters and data dimensionality grows. There have been two directions to improve this: (a) to use a better initial set of cluster centroids; and (b) to employ a better (non-greedy) optimization process. EM clustering and Neural-Gas (and some other algorithms, e.g., frequency sensitive competitive

learning) are common examples of the latter category. There are only a few proposals for the first direction, yet none seems to work consistently well.

Our objective here is to reduce empty clusters and we use a very simple heuristic to achieve it. The heuristic works as follows: At the end of each batch iteration (i.e., each time after going through all data points), we keep a list of up to K data points that are most distant from their cluster centroids, and a list of empty clusters. The centroid of each empty cluster will then be replaced by a data point (from the list constructed above) that is most remote to its center (and the data point will be removed from the list so that it will not be assigned more than once). We can always achieve zero empty clusters with this strategy.

IV. Experimental Results

A. Text Datasets

We used the 20-newsgroups data[1] and a number of datasets from the CLUTO toolkit[2] [11]. These datasets provide a good representation of different characteristics: number of documents ranges from 30 to 19949, number of words from 1073 to 43586, number of classes from 3 to 20, and balance from 0.043 to 1.0. The balance of a dataset is defined as the ratio of the number of documents in the smallest class to the number of documents in the largest class. So a value close to 1(0) indicates a very (un)balanced dataset. All datasets are very sparse, as shown by the (very) low values for the fraction of nonzero entries in the term-document matrix. A summary of all the datasets used in this paper is shown in Table I.

TABLE I

Summary of text datasets (for each dataset, n_d is the total number of documents, n_w the total number of words, K the number of classes, and $N_{nz}/(Nd)$ the fraction of nonzeros in the term-document matrix)

Data	N	d	K	$\frac{N_{nz}}{Nd}$	Balance
NG20	19949	43586	20	0.0018	0.991
classic	7094	41681	4	0.0008	0.323
classic30	30	1073	3	0.0492	1.0
classic300	300	5577	3	0.0096	1.0
ohscal	11162	11465	10	0.0053	0.437
k1b	2340	21839	6	0.0068	0.043
tr11	414	6429	9	0.0438	0.046

The *NG20* dataset is a collection of 20,000 messages, collected from 20 different usenet newsgroups, 1,000 messages from each. We preprocessed the raw dataset using the Bow toolkit [20], including chopping off headers and removing stop words as well as words that occur in less than three documents. In the resulting dataset, each document is represented by a 43,586-dimensional sparse vector and there are a total of 19,949 documents (after empty documents being removed).

The other datasets are associated with the CLUTO toolkit and have already been preprocessed [21] and we further

removed those words that appear in two or fewer documents. The *classic30* and *classic300* are sampled [6] from the three well-separated categories in the *classic* dataset.

B. Evaluation criteria

Objective clustering evaluation criteria can be based on internal measures or external measures. An internal measure is often the same as the objective function that a clustering algorithm explicitly optimizes, as is the average cosine similarity criteria used for the spherical k-means algorithm.

For document clustering, external measures are more commonly used, since typically the benchmark documents' category labels are actually known (but of course not used in the clustering process) [22]. It has been argued that the mutual information $I(Y; \hat{Y})$ between a *r.v.* Y, governing the cluster labels, and a *r.v.* $\hat{Y}$, governing the class labels, is a superior measure to others [23]. We shall follow the definition of normalized mutual information (NMI) using geometrical mean, $NMI = \frac{I(Y; \hat{Y})}{\sqrt{H(Y) \cdot H(\hat{Y})}}$, as given in [23]. The NMI value is 1 when clustering results perfectly match the external category labels and close to 0 for a random partitioning.

In our experiments, we use both the average cosine similarity (ACS) and the normalized mutual information (NMI) values to gauge the quality of clustering results.

C. Experimental Setting

For SPKM, we used a common initialization strategy—randomly perturbing global mean to get K similar centroid vectors. For OSKM, we use the same initialization though different initial centroids probably do not make a big difference for online learning. For the exponential decreasing learning rate schedule with OSKM, we used an initial rate $\eta_0 = 1.0$ and a final rate $\eta_f = 0.01$.

For k-means algorithms, we use a maximum number of iterations $M = 20$. Our results show that most runs for SPKM converge within 20 iterations even a relative convergence criterion of $1e^{-6}$ is used. No relative convergence criterion was needed for OSKM. Each experiment is run ten times, each time starting from a different random initialization. The averages and standard deviations of the average cosine similarity values, NMI values, and run time results are reported. The run time results were measured on a PC with Intel Pentium 4 3.06GHz CPU and 1GB memory, running Windows XP.

After surveying a range of spectral or graph-based partitioning techniques, we picked the state-of-the-art graph-based clustering algorithm, CLUTO [11], as the leading representative of this class of similarity-based approaches. The CLUTO toolkit is based on the Metis graph partitioning algorithms [24]. We use *vcluster* in the toolkit with the default setting, which is a bisecting approach to nearest-neighbor graph partitioning. The *vcluster* algorithm is greedy and thus dependent on the order of nodes from the input graph. We run each algorithm ten times, each run using a different order of documents. This order effect is small though, as seen from our results in the next section.

In summary, we compare the following algorithms: **SPKM**—batch spherical k-means; **OSKM-1**—online spherical k-means with flat learning rate $\eta = 0.05$; **OSKM-2**—online spherical k-means with the annealing-type learning rate schedule in Section III-B and $\eta_0 = 1.0, \eta_f = 0.01$; **OSKM-fast**—OSKM-2 plus the sampling trick in Section III-C; **CLUTO**—We use the *vcluster* program with default setting in the CLUTO package. The default objective function is equivalent to the average cosine similarity objective in (2). The deferred normalization scheme and the heuristic to avoid empty clusters are used in all OSKM algorithms.

D. Results Analysis

For the two small data sets (*classic30* and *classic300*) that contain well-separated clusters, we simply show the typical confusion matrices obtained by each of the five algorithms. Table II and III show the results for *classic30* and *classic300*, respectively. We put the confusion matrices for OSKM-2 and OSKM-fast together since they deliver the same results for these two small data sets. These data sets have been used in [6] to demonstrate the effectiveness of their local search approach. Here we demonstrate that the OSKM and CLUTO algorithms can achieve the same good results. For the following analysis, we only consider the rest six datasets.

TABLE II
CONFUSION MATRICES FOR *classic30* DATASET (EACH ROW IS A CLUSTER AND EACH COLUMN A CLASS)

SPKM			OSKM-1		
5	3	2	9	0	0
3	5	2	1	9	2
2	2	6	0	1	8
OSKM-2 OSKM-fast			CLUTO		
9	0	0	8	0	0
0	10	1	1	10	2
1	0	9	1	0	8

TABLE III
CONFUSION MATRICES FOR *classic300* DATASET (EACH ROW IS A CLUSTER AND EACH COLUMN A CLASS)

SPKM			OSKM-1		
34	4	28	98	0	21
40	92	2	2	100	0
26	4	70	0	0	79
OSKM-2 OSKM-fast			CLUTO		
98	0	0	99	0	0
2	100	0	1	100	0
0	0	100	0	0	100

Table IV shows the average cosine similarity results. Only the averages (over 10 runs) are presented since the standard deviations are very small. Boldface numbers signify the best algorithm in each column. Table V shows the significance t-test results (p-values) for several hypotheses listed in the first

column. For example, OSKM-2>SPKM means that OSKM-2 is better than SPKM. A p-value of < 0.05 indicates significant evidence for the tested hypothesis to be true, and a value of > 0.95 suggests the opposite of the tested hypothesis be true. These values are highlighted using boldface in the table. We observe that OSKM-2 (the OSKM with exponentially decreasing learning rate schedule) generates significantly better average cosine similarity values than both SPKM and OSKM-1 (except on *classic* vs. SPKM). OSKM-fast is comparable to OSKM-2 for every data set used in our experiments. Compared to CLUTO, OSKM-fast is significantly better for the two largest data sets (*NG20* and *ohscal*), but worse for three other data sets.

TABLE IV
AVERAGE COSINE SIMILARITY (ACS) RESULTS

	NG20	ohscal	classic	k1b	tr11
K	20	10	4	6	9
SPKM	0.1564	0.1880	0.1514	0.1969	0.3541
OSKM-1	0.1500	0.1813	0.1421	0.1957	0.3668
OSKM-2	**0.1596**	0.1905	0.1506	0.2029	0.3723
OSKM-fast	0.1594	**0.1906**	0.1509	0.2028	0.3710
CLUTO	0.1582	0.1881	**0.1663**	**0.2081**	**0.3747**

TABLE V
ACS T-TEST RESULTS (P-VALUES)

	NG20	ohscal	classic	k1b	tr11
OSKM-2 >SPKM	**1.4e-8**	**3.3e-4**	**0.98**	**5.3e-6**	**1.7e-6**
OSKM-2 >OSKM-1	**0**	**0**	**1.1e-16**	**4.3e-14**	**7.6e-7**
OSKM-fast <OSKM-2	0.16	0.8	0.88	0.22	0.14
OSKM-fast <CLUTO	**1.0**	**1.0**	**0**	**0**	**2.5e-3**

When we look at the NMI results, however, the comparison between OSKM-fast and CLUTO is a different story. Table VI shows the NMI results and Table VII the NMI-based t-test results. Now the OSKM-fast approach is significantly better than CLUTO in all date sets. This also suggests that higher objective function values do not necessarily translate into better NMI values (i.e., better categorization of documents). The test results for other hypotheses are similar as above: OSKM-2 is much better than SPKM for all data sets, better than OSKM-1 for all but one data set (*ohscal*), while OSKM-fast is comparable to OSKM-2 for all but one data set (*classic*).

Table VIII shows the average run time results in seconds. OSKM-fast is the fastest for all except one small data set (*tr11*). It reduces the run time of OSKM-2 by half, but performs just as well. Except for *NG20* data set, all the k-means algorithms are faster than CLUTO. Here we fixed the value of K to be the same as the true number of classes in each data set, but will try different K values in the future.

TABLE VI

NORMALIZED MUTUAL INFORMATION (NMI) RESULTS

	NG20	ohscal	classic	k1b	tr11
K	20	10	4	6	9
SPKM	$.54 \pm .02$	$.42 \pm .03$	$.55 \pm .04$	$.56 \pm .05$	$.54 \pm .06$
OSKM-1	$.56 \pm .01$	$.44 \pm .01$	$.56 \pm .05$	$.62 \pm .03$	$.66 \pm .04$
OSKM-2	$\mathbf{.59 \pm .01}$	$.44 \pm .01$	$\mathbf{.63 \pm .03}$	$\mathbf{.66 \pm .03}$	$\mathbf{.71 \pm .02}$
OSKM-fast	$\mathbf{.59 \pm .01}$	$\mathbf{.45 \pm .01}$	$.59 \pm .03$	$.65 \pm .03$	$\mathbf{.71 \pm .02}$
CLUTO	$.58 \pm .01$	$.44 \pm .00$	$.55 \pm .01$	$.62 \pm .03$	$.69 \pm .02$

TABLE VII

NMI T-TEST RESULTS (P-VALUES)

	NG20	ohscal	classic	k1b	tr11
OSKM-2 >SPKM	**9.7e-9**	**0.04**	**1.2e-4**	**1.7e-5**	**1.1e-7**
OSKM-2 >OSKM-1	**1.1e-6**	0.36	**1.2e-3**	**0.003**	**1.1e-3**
OSKM-fast <OSKM-2	0.7	0.86	**0.04**	0.32	0.54
OSKM-fast <CLUTO	**1.0**	**0.99**	**0.996**	**0.99**	**0.98**

TABLE VIII

RUN TIME RESULTS (IN SECONDS)

	NG20	ohscal	classic	k1b	tr11
K	20	10	4	6	9
SPKM	19.87	3.5	0.34	0.96	**0.11**
OSKM-1	20.5	3.16	0.41	0.99	0.13
OSKM-2	21.88	3.45	0.78	1.05	0.14
OSKM-fast	**11.38**	**1.8**	**0.27**	**0.58**	0.14
CLUTO	18.35	5.5	1.16	1.56	0.25

V. CONCLUSION

In this paper, we have investigated an efficient online spherical k-means algorithm, which employs the "Winner-Take-All" competitive learning technique, together with an annealing-type learning rate schedule. We have shown that the online algorithm can achieve significantly better clustering results than batch spherical k-means, yet still run faster, with the help of several heuristics to improve speed. The studied online algorithm also produce comparable clustering results to CLUTO, using less time. Future enhancement of the OSKM-fast algorithm can consider more speedup heuristics such as the pruning strategy introduced in [25].

REFERENCES

[1] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

[2] Shi Zhong and Joydeep Ghosh, "Generative model-based document clustering: A comparative study," *Knowledge and Information Systems: An International Journal*, vol. Online First, February 2005.

[3] I. S. Dhillon and D. S. Modha, "Concept decompositions for large sparse text data using clustering," *Machine Learning*, vol. 42, no. 1, pp. 143–175, 2001.

[4] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[5] Kenneth Rose, "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems," *Proceedings of The IEEE*, vol. 86, no. 11, pp. 2210–2239, 1998.

[6] Inderjit S. Dhillon, Yuqiang Guan, and J. Kogan, "Iterative clustering of high dimensional text data augmented by local search," in *Proc. IEEE Int. Conf. Data Mining*, Maebashi City, Japan, 2002, pp. 131–138.

[7] Teuvo Kohonen, "The Self-Organizing Map," *Proceedings of The IEEE*, vol. 78, no. 9, pp. 1464–1480, September 1990.

[8] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, ""Neural-Gas" network for vector quantization and its application to time-series prediction," *IEEE Trans. Neural Networks*, vol. 4, no. 4, pp. 558–569, July 1993.

[9] Arindam Banerjee and Joydeep Ghosh, "Frequency-sensitive competitive learning for scalable balanced clustering on high-dimensional hyperspheres," *IEEE Trans. Neural Networks*, vol. 15, no. 3, pp. 702–719, May 2004.

[10] A. S. Galanopoulos, R. L. Moses, and S. C. Ahalt, "Diffusion approximation of frequency sensitive competitive learning," *IEEE Trans. Neural Networks*, vol. 8, no. 5, pp. 1026–1030, September 1997.

[11] George Karypis, *CLUTO - A Clustering Toolkit*, Dept. of Computer Science, University of Minnesota, May 2002, http://www-users.cs.umn.edu/~karypis/cluto/.

[12] Christian Borgelt and Andreas Nuernberger, "Fast fuzzy clustering of web page collections," in *Proc. PKDD Workshop on Statistical Approaches to Web Mining*, Pisa, Italy, 2004, pp. 75–86.

[13] E. Forgy, "Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications," *Biometrics*, vol. 21, no. 3, pp. 768, 1965.

[14] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statistics and Probability*, 1967, pp. 281–297.

[15] Tom Mitchell, *Machine Learning*, McGraw Hill, 1997.

[16] Shi Zhong and Joydeep Ghosh, "A unified framework for model-based clustering," *Journal of Machine Learning Research*, vol. 4, pp. 1001–1037, November 2003.

[17] Alexander Strehl, Joydeep Ghosh, and Raymond J. Mooney, "Impact of similarity measures on web-page clustering," in *AAAI Workshop on AI for Web Search*, July 2000, pp. 58–64.

[18] S. C. Ahalt, A. K. Krishnamurthy, P. Chen, and D. E. Melton, "Competitive learning algorithms for vector quantization," *Neural Networks*, vol. 3, no. 3, pp. 277–290, 1990.

[19] B. Fritzke, "Incremental learning of local linear mappings," in *Int. Conf. Artificial Neural Networks*, 1995, pp. 217–222.

[20] Andrew K. McCallum, "Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering," Available at http://www.cs.cmu.edu/~mccallum/bow, 1996.

[21] Ying Zhao and George Karypis, "Criterion functions for document clustering: experiments and analysis," Tech. Rep. #01-40, Department of Computer Science, University of Minnesota, November 2001.

[22] Joydeep Ghosh, "Scalable clustering," in *Handbook of Data Mining*, Nong Ye, Ed., pp. 341–364. Lawrence Erlbaum Assoc., 2003.

[23] Alexander Strehl and Joydeep Ghosh, "Cluster ensembles — a knowledge reuse framework for combining partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2002.

[24] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[25] Inderjit S. Dhillon, James Fan, and Yuqiang Guan, "Efficient clustering of very large document collections," in *Data Mining for Scientific and Engineering Applications*, R. L. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar, and R. R. Namburu, Eds., pp. 357–381. Kluwer Academic publishers, 2001.