Frequency Sensitive Competitive Learning for Clustering on High-dimensional Hyperspheres

Arindam Banerjee and Joydeep Ghosh
Dept. of Electrical and Computer Engineering
University of Texas at Austin, Austin, TX 78712

Abstract—This paper derives three competitive learning mechanisms from first principles to obtain clusters of comparable sizes when both inputs and representatives are normalized. These mechanisms are very effective in achieving balanced grouping of inputs in high dimensional spaces, as illustrated by experimental results on clustering two popular text data sets in 26,099 and 21,839 dimensional spaces respectively.

I. Introduction

Competitive learning mechanisms involve winner-take-all networks to determine the most responsive cell to a given input [9], [16], [10]. If this cell or exemplar then adjusts its afferent weights to respond even stronger to the same input using a Hebbian principle, the resultant system can be shown to perform unsupervised clustering. Well known competitive learning systems include that of Rumelhart and Zipser [16], and the ART network [3] which also involves mechanisms to address the stability-plasticity dilemma.

If the winning cell is selected as the one having the minimum Euclidean distance to the input, an online version of the kmeans clustering algorithm is obtained [11]. However, such mechanisms in general suffer from poor performance when the input dimension is very high ($>$ 1000) because of the well-known "curse of dimensionality" effects [6]. This problem can be mitigated by constraining both input and weight vectors to be of the same length. In this setting, it is often preferable to select the winning cell as the one with the highest inner product of its afferent weight vector with the input vector. In fact, a batch mode version of such a (normalized) competitive learning technique called spherical kmeans has been successfully used to cluster text documents in 2000+ dimensional space [5].

Unfortunately, both normalized and non-normalized competitive learning mechanisms for clustering suffer from another problem which is exacerbated when there are many input cells, i.e., when the input is represented as a vector in some very high-dimensional space. This is the problem of obtaining clusters of widely varying sizes [2]. This problem was addressed for Euclidean space clustering (kmeans and variants) by frequency sensitive competitive learning(FSCL), which was originally formulated to remedy the problem of under-utilization of parts of a codebook in Vector Quantization [1]. Motivated by earlier work of Grossberg [9], FSCL introduced a "conscience mechanism" that multiplicatively scaled the distortion (distance of the exemplar or codebook vector from the input) by the number of times that exemplar was the winner in the past. Thus highly winning exemplars were discouraged from attracting new inputs. However, this mechanism was not derived from first principles or applied to normalized clustering.

Several applications, particularly in the data mining context, involve *clustering very high dimensional data into groups of comparable sizes*. For example, a direct marketing campaign often starts with segmenting customers into groups of roughly equal size or equal estimated revenue generation, (based on market basket analysis, or purchasing behavior at a web site), so that the same number of sales teams (or marketing dollars) can be allocated to each segment. In large retail chains, one often desires product categories/groupings of comparable importance, since subsequent procedures such as shelf/floor space allocation and product placement are influenced by the objective of allocating resources proportional to revenue or gross margins associated with the product groups [19]. Typically, a customer is characterized by a vector denoting, for each product, the quantity bought or some derived number such as amount paid, while products are characterized by which customers purchased them. For real retail data, these vectors have thousands of dimensions [20]. Similarly, in clustering of a large corpus of documents to generate topic hierarchies, balancing greatly facilitates navigation by avoiding the generation of hierarchies that are highly skewed, with uneven depth in different parts of the hierarchy "tree" or having widely varying number of documents at the leaf nodes.

In this paper, we first show that the spherical kmeans algorithm can be derived from a certain maximum likelihood formulation using a mixture of von Mises-Fisher distributions as the generative model. Then, in section III, we propose a variant in which the dispersion of a mixture component decreases if more data points are attributed to it. This results in a batch-mode frequency sensitive competitive algorithm for normalized scenarios. Two online versions of this algorithm are then suggested in section IV. Experimental results on high-dimensional text clustering problems are presented in section V. The paper concludes in section VI with a discussion on the proposed

schemes.

II. Clustering on a Hypersphere

The classical kmeans clustering algorithm gives the maximum likelihood estimates of the means of k Gaussians [15] with identity covariances by using the Expectation Maximization (EM) algorithm [4] (with a delta function prior on the indicator variables). The EM algorithm is guaranteed to give a local optimum for these maximum likelihood estimates. We use the same approach for deriving the spherical kmeans for points on the surface of a hypersphere. Recall that the von Mises-Fisher (vMF) distribution is an analogue of the Gaussian distribution on a hypersphere [18], [13] in that it is the maximum entropy distribution on the hypersphere when the first two moments are fixed [12]. The density of a d-dimensional vMF distribution is given by

$$f(\mathbf{x}; \boldsymbol{\mu}, \kappa) = \frac{1}{Z_d(\kappa)} \exp\left(\kappa \frac{\mathbf{x}^T \boldsymbol{\mu}}{\|\boldsymbol{\mu}\|}\right) \tag{1}$$

where $\boldsymbol{\mu}$ represents the mean direction vector and κ is the dispersion around the mean, analogous to the mean and covariance for the multivariate Gaussian distribution. The normalizing coefficient is

$$Z_d(\kappa) = (2\pi)^{d/2} I_{d/2-1}(\kappa)/\kappa^{d/2-1} \tag{2}$$

where $I_r(y)$ is the modified Bessel function of the first kind and order r [14]. Assume that there are n data-points $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ on the surface of a unit hypersphere and there are k vMF distributions $f_h, h = 1, \cdots, k$ such that each point has been generated from exactly one of these distributions. The clustering problem is to estimate the parameters of the k vMF distributions so that the likelihood of the observed data is maximized. Like the Euclidean kmeans case, we assume κ to be a constant for now. Let z_{ih} be the indicator variable for the event that $\mathbf{x}_i$ was generated by f_h. Then, assuming the data-points have been drawn independently, the likelihood of the observed data is given by

$$\mathcal{L}(\Theta|\mathcal{X}) = \prod_{i=1}^{n} \sum_{h=1}^{k} z_{ih} f(\mathbf{x}_i; \boldsymbol{\mu}_h) \tag{3}$$

where $\boldsymbol{\mu}_h$ is the mean of the h-th vMF distribution. For a given value of the parameters the most likely distribution to have generated $\mathbf{x}_i$ (in terms of log-likelihood) is given by

$$\begin{aligned} h^* &= \arg\max_h \quad \log f(\mathbf{x}_i; \boldsymbol{\mu}_h) \\ &= \arg\max_h \quad \kappa\, \mathbf{x}_i^T \boldsymbol{\mu}_h - \log(Z_d(\kappa)) \quad [\because \|\boldsymbol{\mu}_h\| = 1] \\ &\equiv \arg\max_h \quad \mathbf{x}_i^T \boldsymbol{\mu}_h \qquad [\because \kappa \text{ is constant}] \end{aligned} \tag{4}$$

Using Eqn. 4 to assign each data point to a distribution, we can re-estimate the means of the corresponding distributions as follows:

$$\boldsymbol{m}_h = \sum_{\mathbf{x}_i \in f_h} \mathbf{x}_i, \quad \boldsymbol{\mu}_h = \frac{\boldsymbol{m}_h}{\|\boldsymbol{m}_h\|}, \quad h = 1, \cdots, k \tag{5}$$

Repeating the steps given in Eqns. 4 and 5 results in a gradient ascent scheme that is guaranteed to give a local maxima of the likelihood function in Eqn. 3 after convergence [5]. This scheme is known as the spherical kmeans (`spkmeans`) algorithm since the data-points lie on the surface of the unit sphere. It is a batch mode version of normalized competitive learning [16]. The performance of this algorithm can be evaluated by the normalized spherical kmeans log-likelihood given by

$$\mathcal{J} = \frac{1}{n} \sum_{h=1}^{k} \sum_{\mathbf{x} \in f_h} \mathbf{x}^T \boldsymbol{\mu}_h, \tag{6}$$

interpreted as the average similarity (cosine of the angle) between any vector $\mathbf{x}$ and its cluster representative $\boldsymbol{\mu}_h$.

III. Frequency Sensitive Spherical Kmeans

Like its Euclidean space counterpart, the spherical kmeans quite often gets stuck in poor local solutions resulting in empty clusters or clusters having very few points. Clearly, the formulation does not have any explicit way to ensure against such a scenario. A similar problem had been reported in the signal processing community for the problem of vector quantization where some parts of the codebook were under-utilized as a result of poor local solutions to the optimization problem for codebook generation [16]. The problem was empirically solved by using frequency sensitive competitive learning(FSCL) [1], [7]. FSCL is a conscience type competitive learning approach that overcomes the problems associated with simple competitive learning [1] and Kohonen's self-organizing feature maps in vector quantization applications. In the FSCL, the competitive computing units are penalized in proportion to (some function of) the frequency of their winning, so that eventually all units participate in the quantization of the data space. Convergence properties of the FSCL algorithm to a local minima have been studied by approximating the final phase of the FSCL by a diffusion process described by a Fokker-Plank equation [8].

Using the same basic idea, we propose a change in the formulation of spherical kmeans in order to prevent poor local solutions. Rather than keeping κ constant, we propose to make it inversely proportional to the number of points assigned to the corresponding distribution. Thus, if n_h is the number of points assigned to f_h, then we set $\kappa_h \propto 1/n_h$. Intuitively, this is akin to using shrinking Gaussians in the Euclidean space in the sense that as more

points are assigned to a particular cluster, the "width" of its representative Gaussian reduces. As a result, effective distance of points from this cluster increases, or, in the spherical case, the similarity of points from this cluster decreases. Thus, if a point $\mathbf{x}$ is such that $\mathbf{x}^T\boldsymbol{\mu}_1 = \mathbf{x}^T\boldsymbol{\mu}_2$ but $n_{h_1} < n_{h_2}$, then $\mathbf{x}$ has a higher likelihood of having been generated from f_{h_1} than f_{h_2} in the frequency sensitive setting. Hence, the likelihood of points going to clusters having less number of points is higher and this implicitly prevents poor local solutions having empty clusters or clusters having very small number of points.

Formally, let $\kappa_h \propto 1/n_h \Rightarrow \kappa_h = c/n_h$, where c is a proportionality constant that we will later on choose to our benefit. Then, the log-likelihood of data-point $\mathbf{x}_i$ having been generated from f_h is given by

$$\begin{aligned}
&\log\ f(\mathbf{x}_i;\boldsymbol{\mu}_h,\kappa_h) \\
&\quad= \frac{c}{n_h}\,\mathbf{x}_i^T\boldsymbol{\mu}_h - \log Z_d\left(\frac{c}{n_h}\right) \\
&\quad= \frac{c}{n_h}\,\mathbf{x}_i^T\boldsymbol{\mu}_h - \frac{d}{2}\log(2\pi) - \log(I_{d/2-1}(c/n_h)) \\
&\qquad+ (\frac{d}{2}-1)\log(c/n_h) \\
&\quad\equiv \frac{c}{n_h}\,\mathbf{x}_i^T\boldsymbol{\mu}_h - \log(I_{d/2-1}(c/n_h)) - (\frac{d}{2}-1)\log n_h
\end{aligned}$$

where $\mathcal{F}(h) \equiv \mathcal{G}(h)$ means that $\arg\max_h \mathcal{F}(h) = \arg\max_h \mathcal{G}(h)$. For simplifying the expression further, we choose $c = nd^2/k$. Then, noting that $n_h = O(n/k)$ and d is a large number so that $nd^2/2kn_h \gg d$, we get

$$\begin{aligned}
&\log I_{d/2-1}(c/n_h) \\
&\quad= \log I_{d/2-1}\left(\frac{n/k}{n_h}\cdot\frac{d^2}{2}\right) \\
&\quad\approx \frac{n/k}{n_h}\cdot\frac{d^2}{2} - \frac{1}{2}\log(2\pi\frac{n/k}{n_h}\cdot\frac{d^2}{2}) \\
&\quad\equiv \frac{n/k}{n_h}\cdot\frac{d^2}{2} + \frac{1}{2}\log n_h
\end{aligned}$$

using the fact that $I_n(x) \approx e^x/\sqrt{2\pi x}$ for fixed n and $x \gg n$ [14]. Hence,

$$\begin{aligned}
&\log\ f(\mathbf{x}_i;\boldsymbol{\mu}_h,\kappa_h) \\
&\quad\equiv \frac{n/k}{n_h}\cdot\frac{d^2}{2}\mathbf{x}^T\boldsymbol{\mu}_h - \frac{n/k}{n_h}\cdot\frac{d^2}{2} - \frac{1}{2}\log n_h \\
&\qquad- (\frac{d}{2}-1)\log n_h \\
&\quad= \frac{n/k}{n_h}\cdot\frac{d^2}{2}\left(\mathbf{x}^T\boldsymbol{\mu}_h + 1\right) - \frac{d-1}{2}\log n_h \\
&\quad\approx \frac{d}{2}\left[\frac{(n/k)d}{n_h}\left(\mathbf{x}^T\boldsymbol{\mu}_h + \boldsymbol{\mu}_h^T\boldsymbol{\mu}_h\right) - \log n_h\right] \\
&\quad\equiv \frac{(n/k)d}{n_h}(\mathbf{x}+\boldsymbol{\mu}_h)^T\boldsymbol{\mu}_h - \log n_h
\end{aligned}$$

Hence, the most likely distribution to have generated the point $\mathbf{x}_i$ is given by

$$h^* = \arg\max_h \left\{\frac{(n/k)d}{n_h}(\mathbf{x}+\boldsymbol{\mu}_h)^T\boldsymbol{\mu}_h - \log n_h\right\} \quad (7)$$

$$= \arg\max_h \frac{1}{n_h}\left\{\mathbf{x}^T\boldsymbol{\mu}_h + 1 - \frac{n_h}{(n/k)d}\log n_h\right\} \quad (8)$$

We write Eqn. 7 in a second form in Eqn. 8 in order to get a better understanding of the dependencies on n_h. Note that the spherical kmeans assignment function (Eqn. 4) now gets a multiplicative and an additive term both of which penalize large sized clusters. Note that this particular form of the likelihood function is due to our choice of the proportionality constant and other values of the constant will give slightly different forms for this likelihood function. However, this particular choice helps us use an asymptotic behavior of the modified Bessel function of the first kind thereby making the formulation computationally tractable.

Based on this result, we present the algorithm `fs-spkmeans` (frequency sensitive spherical kmeans) that is a variant of `spkmeans` and prevents it from getting stuck in bad local solutions.

Algorithm `fs-spkmeans`

1.Set iteration count $t \leftarrow 0$. Choose k points (unit vectors) as the cluster means $\boldsymbol{\mu}_h^{(0)}$, set $n_h^{(0)} \leftarrow \frac{n}{k}$, $h = 1, \cdots, k$.

2.Repeat until *convergence*

2a. Assign each data-point $\mathbf{x}$ to the cluster $f_{h^*}^{(t)}$ where $h^* = \arg\max_h \frac{1}{n_h}\left\{\mathbf{x}^T\boldsymbol{\mu}_h + 1 - \frac{n_h}{(n/k)d}\log n_h\right\}$

2b. $n_h^{(t+1)} \leftarrow |\{\mathbf{x} : \mathbf{x} \in f_h^{(t)}\}|$

2c. $\boldsymbol{m}_h^{(t+1)} \leftarrow \frac{1}{n_h^{(t+1)}}\sum_{\mathbf{x}\in f_h^{(t)}}\mathbf{x}, \quad \boldsymbol{\mu}_h^{(t+1)} \leftarrow \frac{\boldsymbol{m}_h^{(t+1)}}{\|\boldsymbol{m}_h^{(t+1)}\|}$

2d. $t \leftarrow (t+1)$

IV. Online fs-spkmeans

Though the algorithm `fs-spkmeans` is motivated by the FSCL, it does not have the online flavor of FSCL. To study the effect of the online behavior of `fs-spkmeans`, we made two online variants of this algorithm. The first, called `ofs-spkmeans` (online `fs-spkmeans`), basically incorporates step 2b of `fs-spkmeans` into step 2a. In other words, in each iteration t, as soon as a point gets assigned to the h-th cluster, the value of $n_h^{(t)}$ is updated. The algorithm is presented below.

Algorithm `ofs-spkmeans`

1.Set iteration count $t \leftarrow 0$. Choose k points (unit vectors) as the cluster means $\boldsymbol{\mu}_h^{(0)}$, set $n_h^{(0)} \leftarrow \frac{n}{k}$,

$h = 1, \cdots, k$.

2.Repeat until *convergence*

2a. For $i = 1$ to n,

(i) Assign the data-point $\mathbf{x}_i$ to the cluster $f^{(t)}_{h^*}$ where $h^* = \arg\max_h \frac{1}{n_h}\left\{\mathbf{x}^T\boldsymbol{\mu}_h + 1 - \frac{n_h}{(n/k)d}\log n_h\right\}$

(ii) $n^{(t)}_{h^*} \leftarrow n^{(t)}_{h^*} + 1\,;\;\; n^{(t)}_h \leftarrow n^{(t)}_h - \frac{1}{k},\ \forall h$

2b. $\boldsymbol{m}^{(t+1)}_h \leftarrow \frac{1}{n^{(t)}_h}\sum_{x \in f^{(t)}_h} x,\;\; \boldsymbol{\mu}^{(t+1)}_h \leftarrow \frac{\boldsymbol{m}^{(t+1)}_h}{\|\boldsymbol{m}^{(t+1)}_h\|}$

2c. $n^{(t+1)}_h \leftarrow n^{(t)}_h, h = 1, \cdots, k, \quad t \leftarrow (t+1)$

A more complete online version is the `fscl-spkmeans` (FSCL `spkmeans`), that is purely in the flavor of the FSCL. There is only one iteration on the entire data and as soon as a point gets assigned to the h-th cluster, both n_h and $\boldsymbol{\mu}_h$ are updated. Thus, in this scheme, we have shrinking as well as moving vMF distributions trying to model the data. The basic algorithm is presented below.

Algorithm `fscl-spkmeans`

1. Choose k points (unit vectors) as the cluster means $\boldsymbol{\mu}_h$, set $n_h \leftarrow \frac{n}{k}, h = 1, \cdots, k$.

2. For $i = 1$ to n

2a. Assign the data-point $\mathbf{x}_i$ to the cluster f_{h^*} where $h^* = \arg\max_h \frac{1}{n_h}\left\{\mathbf{x}^T\boldsymbol{\mu}_h + 1 - \frac{n_h}{(n/k)d}\log n_h\right\}$

2b. $n_{h^*} \leftarrow n_{h^*} + 1\,;\;\; n_h \leftarrow n_h - \frac{1}{k},\ \forall h$

2c. $\boldsymbol{m}_h \leftarrow \boldsymbol{\mu}_h + \frac{1}{n_h}(x_i - \boldsymbol{\mu}_h), \quad \boldsymbol{\mu}_h \leftarrow \frac{\boldsymbol{m}_h}{\|\boldsymbol{m}_h\|}$

Note that in both the online algorithms, after each point is assigned to a cluster and its count incremented, a constant $1/k$ is subtracted from each n_h. This ensures that at any point of time, the total number of points in all the clusters add up to n.

V. Experimental Results

In this section, experimental results on the proposed ideas are presented. We present results on two high-dimensional text datasets. We used the 20-newsgroups dataset [1] and the Yahoo news dataset [2](K1) for the empirical performance analysis. The 20-newsgroup dataset is a collection of 20,000 messages, collected from 20 different usenet newsgroups. One thousand messages from each of the twenty newsgroups were chosen at random and partitioned by newsgroup name. The headers for each of the messages were chopped off so that they do not bias the results. The toolkit MC [5] was used for creating the high-dimensional vector space model for the text documents and a total of 26099 words were used. Thus, each message, after normalization, is represented as a unit vector in a 26099 dimensional space. The Yahoo news K-series dataset is a collection of 2340 Yahoo news articles belonging one of 20 different Yahoo categories. The K1 set actually gives the high-dimensional vector space model having 21839 words. After normalization, the data-points reside on the surface of a 21839 dimensional hypersphere.

First, we take a look at the performance of the frequency sensitive algorithms as compared to the spherical kmeans (`spkmeans`) in terms of the objective function values. For a particular value of number of clusters k, we run all the algorithms on a data-set. The initial k means of the spherical kmeans were generated by computing the mean of the entire data and making k small random perturbations to this mean [5]. For stability and repeatability, the frequency sensitive algorithms were initialized at points of local minima of the spherical kmeans objective. In Fig. 1(a), the normalized spherical kmeans objective function value (Eqn. 6) for all the algorithms (after convergence on their individual objective functions) on the 20-newsgroups data is presented. The results are averaged over 10 runs for each value of k and the standard deviations are shown by error bars. It is interesting to note that all the approaches give approximately the same value for the objective upto around $k \approx 15$ after which they separate out. The frequency sensitive algorithms give a higher value of the objective for $k > 15$. A closer look reveals that `fs-spkmeans` and `fscl-spkmeans` perform almost identically. Also, the objective function for `ofs-spkmeans` is actually slightly less than the rest upto $k \approx 15$ after which it crosses that of `spkmeans` and closely follows that of the other two frequency sensitive algorithms. Similar results are obtained on the Yahoo dataset as shown in Fig. 2(a). Note that in this case `fs-spkmeans` and `fscl-spkmeans` separate out around $k \approx 10$ and the crossover of `ofs-spkmeans` occurs at around $k \approx 30$. Thus, changing the objective function of `spkmeans` once it has got to a local solution to a frequency sensitive version actually improves the `spkmeans` objective function. This is very interesting since this technique has apparently no relation with standard techniques like simulated annealing and genetic algorithms that are normally used to avoid bad local solutions.

In Fig. 1(b), the variance of the number of documents per cluster on the 20-newsgroups data is presented for all the algorithms. The corresponding results for the Yahoo dataset is given in Fig. 2(b). `ofs-spkmeans` gives the minimum variance in cluster sizes in both the cases and is significantly better than the rest of the algorithms. `fs-spkmeans` and `fscl-spkmeans` perform identically and gives a lower variance in cluster sizes as compared to `spkmeans`. As a special and interesting case, we com-

[1] http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.html

[2] ftp://ftp.cs.umn.edu/users/boley/PDDPdata/

pare the algorithms based on the smallest sized cluster they generate in different runs. Fig. 1(c) shows results on the 20-newsgroups data and Fig. 2(c) shows the corresponding results on the Yahoo data. The results are plotted as a ratio of the smallest cluster size to the expected cluster size. Note that the minimum cluster size that is plotted is the mean of the smallest cluster sizes over the 10 runs. For the 20-newsgroups data, `spkmeans` starts misbehaving from around $k \approx 20$ and for $k \geq 22$ it seems to always generate some empty clusters. Similar behavior is observed on the Yahoo data for $k \geq 17$. This is a well known property of the kmeans algorithm, specially in very high dimensional spaces. The frequency sensitive algorithms does not seem to have this problem. `fs-spkmeans` and `fscl-spkmeans` again perform identically. `ofs-spkmeans` gives very balanced clusters and a much higher ratio than the rest of the algorithms. Note that in general the balancing seems to be better for the 20-newsgroups data. The primary reason for this is that the classes in the Yahoo data are not balanced, with sizes ranging from 9 to 494, whereas in the 20-newsgroup data all the classes are of the same size.

VI. Concluding Remarks

Clustering problems in high dimensional spaces tend to be difficult because of the curse of dimensionality. Representing the data on a high-dimensional hypersphere takes care of a part of the problem. However, standard kmeans applied on this high-dimensional data gives bad local solutions having empty clusters or clusters having very few points. In this paper, we proposed applying frequency sensitive competitive learning on top of the spherical kmeans setting to avoid such bad local solutions. A similar study has been made in high dimensional Euclidean spaces [17], this is handicapped since regular kmeans severely under-performs in high dimensional spaces [21].

The problem of clustering data lying on the surface of a hypersphere is posed as a maximum likelihood estimation of k vMF distributions that are assumed to have generated the observed data. It was shown that the spherical kmeans algorithm comes from this formulation under the assumption of constant dispersion of the vMF distributions. Then, by setting the dispersions proportional to the inverse of the number of points in the corresponding cluster, we experimentally show that the results are both superior and significantly better balanced.

The result presented in this paper leads us to a very important question. Though the approach of making the objective function frequency sensitive once we have reached a local optimum of the target objective function does not seem to have any links with standard techniques such as simulated annealing for handling similar problems, the approach seems to work quite well. It will be interesting to study the class of functions and optimization problems for which this approach works, why it works and how this compares to simulated annealing over this class of functions.

References

[1] S. C. Ahalt, A. K. Krishnamurthy, P. Chen, and D. E. Melton. Competitive learning algorithms for vector quantization. *Neural Networks*, 3(3):277–290, 1990.

[2] P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. Technical report, Microsoft Research, May 2000.

[3] G. A. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37:54–115, Jan. 1987.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[5] I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In V. Kumar R. Grossman, C. Kamath and R. Namburu, editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, 2001.

[6] J. H. Friedman. An overview of predictive learning and function approximation. *From Statistics to Neural Networks, Proc. NATO/ASI workshop*, pages 1–61, 1994.

[7] A. S. Galanopoulos and S. C. Ahalt. Codeword distribution for frequency sensitive competitive learning with one-dimensional input data. *IEEE Trans. Neural Networks*, 7(3):752–756, 1996.

[8] A. S. Galanopoulos, R. L. Moses, and S. C. Ahalt. Diffusion approximation of frequency sensitive competitive learning. *IEEE Trans. Neural Networks*, 8(5):1026–1030, Sept 1997.

[9] S. Grossberg. Adaptive pattern classification and universal recoding: 1. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121–134, 1976.

[10] S. Grossberg. Competitive learning: From interactive action to adaptive resonance. *Cognitive Science*, 11:23–63, 1987.

[11] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[12] J. N. Kapur and H. K. Kesavan. *Entropy Optimization Principles with Applications*. Academic Press, 1992.

[13] K. V. Mardia. Statistics of directional data. *J. Royal Statistical Society. Series B (Methodological)*, 37(3):349–393, 1975.

[14] N. W. McLachlan. *Bessel Functions for Engineers*. Oxford University Press, 1955.

[15] T. M. Mitchell. *Machine Learning*. McGraw-Hill Intl, 1997.

[16] D. E. Rumelhart and D. Zipser. Feature discovery by competive learning. *Cognitive Science*, 9:75–112, 1985.

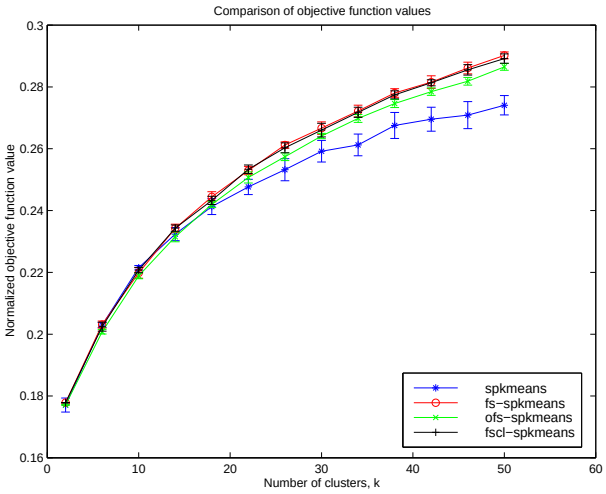
[17] P. Scheunders and S. De Backer. High-dimensional clustering using frequency sensitive competitive learning. *Pattern Recognition*, 32(2):193–202, 1999.

[18] J. Sinkkonen and S. Kaski. Clustering based on conditional distributions in an auxiliary space. *Neural Computation*, 2001. Accepted for publication.

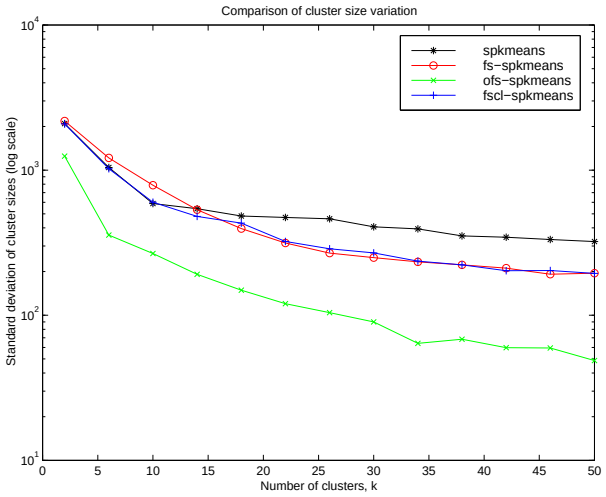
[19] A. Strehl and J. Ghosh. A scalable approach to balanced, high-dimensional clustering of market-baskets. In *Proc 17th Intl Conf on High Performance Computing*, pages 525–536. Springer, Dec 2000.

[20] A. Strehl and J. Ghosh. Value-based customer grouping from large retail data-sets. In *Proc SPIE Conf on Data Mining and Knowledge Discovery : Theory, Tools, and Technology II*, pages 33–42, April 2000.

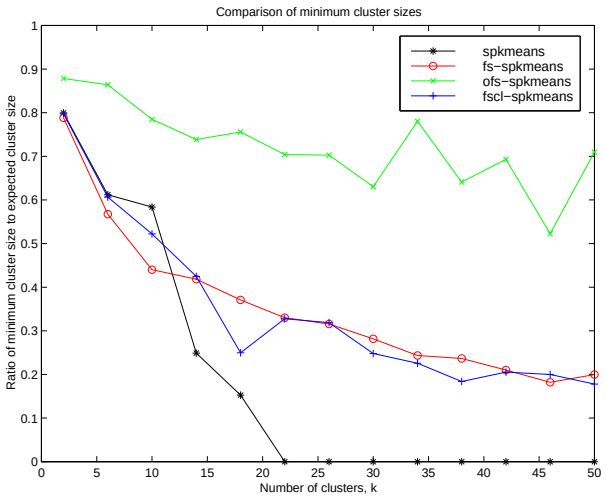
[21] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proc 17th Natl Conf on Artificial Intelligence : Workshop of AI for Web Search*, pages 58–64. AAAI, July 2000.



(a)

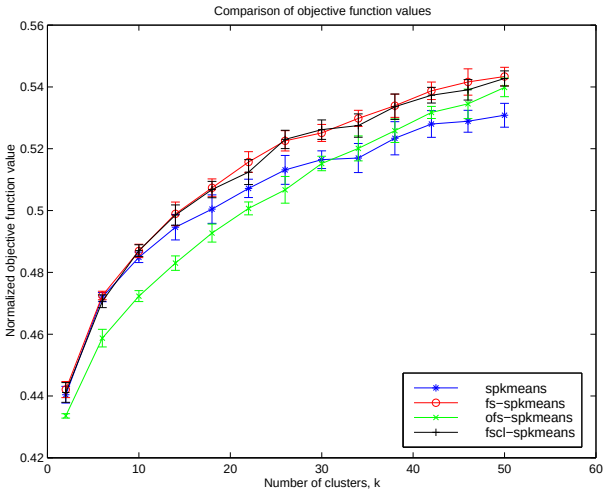


(b)

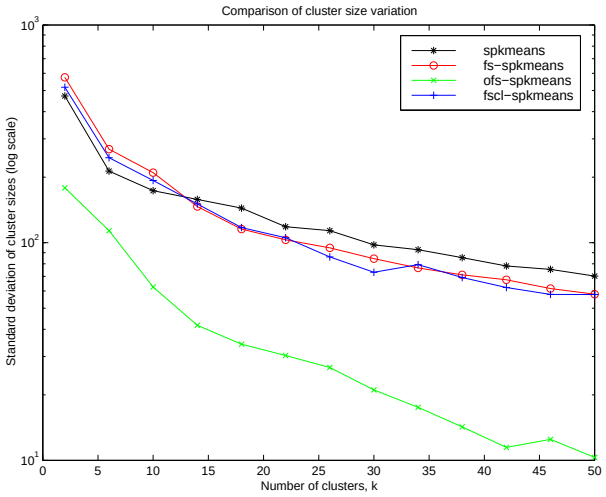


(c)

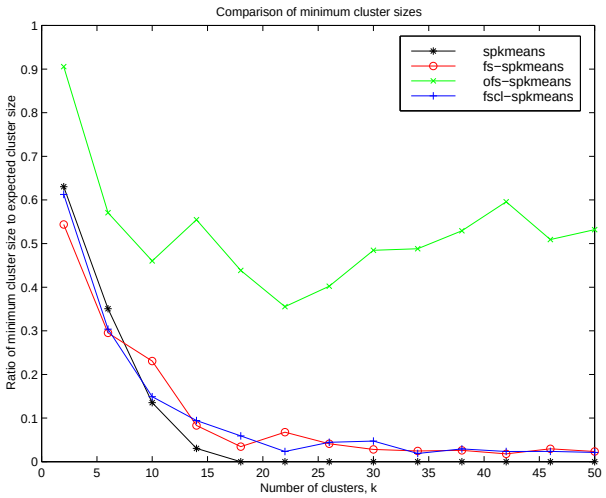
Fig. 1. Comparison on the 20-newsgroups data: (a) the normalized spherical kmeans objective function values, (c) the variance in cluster sizes, and (e) the minimum cluster size, averaged over 10 runs of each algorithm.



(a)



(b)



(c)

Fig. 2. Comparison on the Yahoo data: (a) the normalized spherical kmeans objective function values, (c) the variance in cluster sizes, and (e) the minimum cluster size, averaged over 10 runs of each algorithm.