Generative Model-based Clustering of Directional Data

Arindam Banerjee
Dept of ECE
University of Texas
Austin, TX, USA
abanerje@ece.utexas.edu

Inderjit Dhillon
Dept of CS
University of Texas
Austin, TX, USA
inderjit@cs.utexas.edu

Joydeep Ghosh
Dept of ECE
University of Texas
Austin, TX, USA
ghosh@ece.utexas.edu

Suvrit Sra
Dept of CS
University of Texas
Austin, TX, USA
suvrit@cs.utexas.edu

ABSTRACT

High dimensional directional data is becoming increasingly important in contemporary applications such as analysis of text and gene-expression data. A natural model for multi-variate directional data is provided by the von Mises-Fisher (vMF) distribution on the unit hypersphere that is analogous to the multi-variate Gaussian distribution in $\mathbb{R}^d$. In this paper, we propose modeling complex directional data as a mixture of vMF distributions. We derive and analyze two variants of the Expectation Maximization (EM) framework for estimating the parameters of this mixture. We also propose two clustering algorithms corresponding to these variants. An interesting aspect of our methodology is that the spherical kmeans algorithm (kmeans with cosine similarity) can be shown to be a special case of both our algorithms. Thus, modeling text data by vMF distributions lends theoretical validity to the use of cosine similarity which has been widely used by the information retrieval community. As part of experimental validation, we present results on modeling high-dimensional text and gene-expression data as a mixture of vMF distributions. The results indicate that our approach yields superior clusterings especially for difficult clustering tasks in high-dimensional spaces.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.5.3 [**Pattern Recognition**]: Clustering

Keywords

Clustering, directional data, mixtures, von Mises-Fisher, EM

1. INTRODUCTION

Clustering or segmentation of data is a fundamental data analysis step that has been widely studied across various disciplines [19]. However, several large datasets that are being acquired from scientific domains, as well as the world wide web, have a variety of complex characteristics that severely challenge traditional methods for clustering [15]. This article is concerned with the clustering of high-dimensional directional data that is becoming increasingly common in several application domains.

One can broadly categorize clustering approaches into generative (parametric) [29, 18] and discriminative (non-parametric) [17, 27] ones. The performance of an approach (and of a specific method within that approach) is quite data dependent; there is no clustering method that works the best across all types of data. Generative models, however, often provide better insight into the nature of the clusters. From an application point of view, a lot of domain knowledge can be incorporated into generative models so that clustering of data uncovers specific desirable patterns that one is looking for. Clustering algorithms using the generative model framework, often involve an appropriate application of the Expectation Maximization (EM) algorithm [8, 5] on a properly chosen statistical generative model for the data under consideration. For vector data, there are well studied clustering algorithms for popular generative models such as a mixture of Gaussians, whose effect is analogous to the use of Euclidean or Mahalanobis type distances from the discriminative perspective.

1.1 Motivation and Related Work

There are several application domains where clustering based on minimizing Euclidean distortions yields poor results [30]. For example, empirical studies in information retrieval applications show that *cosine similarity* is a far more effective measure of similarity for analyzing and clustering text documents. Such domains require the use of *directional data* [22] — data that deals only with the directions of unit vectors. Thus, there is a need for generative models that are more appropriate for the analysis and clustering of directional data. In this article, we propose a generative mixture model for directional data on the unit hypersphere and derive two clustering algorithms using this mixture model. We show the connection between the proposed algorithms and a class of existing algorithms for clustering high-dimensional, directional data and present detailed experimental comparisons among them.

As already mentioed, one important domain where directional data is encountered is text analysis, and text clustering in particular. It has been experimentally shown that in order to remove the bias arising due to the length of a document, it often helps to normalize the data vectors [11]. Further, the `spkmeans` algorithm [11], that performs kmeans using cosine similarity instead of Euclidean distortion, has been found to empirically outperform several other schemes.

There are quite a few other domains such as bioinformatics [13], collaborative filtering [26] etc., in which directional data is encountered. A similarity measure that has been found to be useful in these domains is the Pearson correlation coefficient. Given $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, the Pearson product moment correlation between them is given by $\rho(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^d (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^d (x_i - \bar{x})^2} \times \sqrt{\sum_{i=1}^d (y_i - \bar{y})^2}}$, where $\bar{x} = \frac{1}{d}\sum_{i=1}^d x_i$, $\bar{y} = \frac{1}{d}\sum_{i=1}^d y_i$. Considering the mapping $\mathbf{x} \mapsto \tilde{\mathbf{x}}$ such that $\tilde{x}_i = \frac{x_i - \bar{x}}{\sqrt{\sum_{i=1}^d (x_i - \bar{x})^2}}$, we have $\rho(\mathbf{x}, \mathbf{y}) = \tilde{\mathbf{x}}^T \tilde{\mathbf{y}}$. Further, $\|\tilde{\mathbf{x}}\|_2 = 1$. Thus, the Pearson correlation is exactly the cosine similarity between $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ Hence, analysis and clustering of data using Pearson correlations is essentially a clustering problem for directional data.

The application domains described above share another characteristic, namely that the objects to be clustered reside in a *very high dimensional* space, with d sometimes in the thousands or more. Clustering of such high-dimensional data has been of great interest lately [9], with most of the proposed methods following a density-based heuristic or a discriminatory approach [15, 1, 16]. Perhaps the most noteworthy generative approach for clustering high-dimensional data is to use mixtures of Gaussians [7]. Certain other works such as [24] use a generative model from the exponential family and have been explicitly developed for modeling text. The spkmeans algorithm for clustering normalized data was proposed in [11], while the connection between a generative model involving vMF distributions and the spkmeans algorithm was first observed in [3]. An online competitive learning scheme using vMF distributions for minimizing a KL-divergence based distortion was proposed in [28]. However, these approaches do not cluster directional data using explicit probabilistic generative models. In this work, the directional nature of the data is used explicitly resulting in significantly better clustering performances over certain standard techniques, (such as kmeans with cosine similarity) especially for difficult clustering tasks: when clusters overlap, when cluster sizes are skewed, and when cluster sizes are small relative to the dimensionality of the data. Interestingly, as we shall see later on, one of our proposed approaches has an implicit simulated annealing type behavior that seems to alleviate some of the problems associated with high-dimensionality.

Before proceeding further, we give a brief outline of the paper. We first present the vMF distribution on the sphere in section 2. In section 3, we introduce the generative model for a mixture of vMF distributions and analyze the maximum likelihood parameter estimate of the mixture model from a given dataset using the EM framework. Based on the analysis of section 3, two clustering algorithms using soft and hard-assignments respectively are proposed in section 4. We present detailed experimental results on the proposed algorithms in section 5. Some interesting aspects of the proposed algorithms are discussed in section 6. Section 7 presents concluding remarks and directions for future work.

A word about the notation: bold faced variables, e.g., $\mathbf{x}, \boldsymbol{\mu}$, etc., represent vectors; $\|\cdot\|$ denotes the L_2 norm; sets are represented by calligraphic upper-case letters, e.g., $\mathcal{X}$, $\mathcal{Z}$, etc.; $\mathbb{R}$ denotes the set of reals, $\mathbb{S}^{d-1}$ denotes the $(d-1)$-dimensional sphere embedded in $\mathbb{R}^d$. Probability density functions are denoted by lower case alphabets, e.g., $f(\cdot)$, $p(\cdot)$, etc.; probability of a set of events is denoted by P. If a random variable z is distributed as $p(\cdot)$, expectations of functions of z are denoted by $E_p[\cdot]$.

2. THE VON MISES-FISHER (VMF) DISTRIBUTION

A d-dimensional unit random vector $\mathbf{x}$ (i.e., $\|\mathbf{x}\| = 1$) is said to have d-variate von Mises-Fisher (vMF) distribution if its probability density function is given by:

$$f(\mathbf{x}|\boldsymbol{\mu}, \kappa) = c_d(\kappa) e^{\kappa \boldsymbol{\mu}^T \mathbf{x}}, \tag{1}$$

where $\|\boldsymbol{\mu}\| = 1$, $\kappa \geq 0$. Thus, $\mathbf{x}, \boldsymbol{\mu} \in \mathbb{S}^{d-1}$. The normalizing constant $c_d(\kappa)$ is given by:

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

where $I_r(\cdot)$ represents the modified Bessel function of the first kind of order r, see [22] and [12] for more details on vMF distributions. The density $f(\mathbf{x}|\boldsymbol{\mu}, \kappa)$ is parameterized by the mean direction, $\boldsymbol{\mu}$, and the concentration parameter, κ, so-called because it characterizes how strongly the unit vectors drawn according to $f(\mathbf{x}|\boldsymbol{\mu}, \kappa)$ are concentrated about the mean direction $\boldsymbol{\mu}$. Larger values of κ imply stronger concentration about the mean direction. In particular when $\kappa = 0$, $f(\mathbf{x}|\boldsymbol{\mu}, \kappa)$ reduces to the uniform density on $\mathbb{S}^{d-1}$, and as $\kappa \to \infty$, $f(\mathbf{x}|\boldsymbol{\mu}, \kappa)$ tends to a point density.

The von Mises-Fisher distribution is natural for directional data and has properties analogous to those of the multi-variate Gaussian distribution for multi-variate data in $\mathbb{R}^d$. For example, the maximum entropy density on $\mathbb{S}^{d-1}$ subject to the constraint that $E[\mathbf{x}]$ is fixed is a vMF density (see [25] and [21] for details).

3. EM ON MIXTURE OF VMFS

In this section, we introduce a mixture of k vMF (movMF) distributions as a generative model for directional data, and then derive the mixture-density parameter estimation update equations from a given data set using the expectation maximization (EM) framework. The probability density function of the movMF generative model is given by

$$f(\mathbf{x}|\Theta) = \sum_{h=1}^{k} \alpha_h f_h(\mathbf{x}|\theta_h), \tag{3}$$

where $\Theta = \{\alpha_1, \cdots, \alpha_k, \theta_1, \cdots, \theta_k\}$ with $\sum_{h=1}^k \alpha_h = 1$, $\alpha_h \geq 0$, and $f_h(\mathbf{x}|\theta_h)$ is a single vMF distribution with parameters $\theta_h = (\boldsymbol{\mu}_h, \kappa_h)$. In order to sample a point from the generative model perspective, the h-th vMF is chosen at random with probability α_h, and then a point is sampled from $\mathbb{S}^{d-1}$ following $f_h(\mathbf{x}|\theta_h)$. Let $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ be a data set generated by sampling independently following this generative model. Let $\mathcal{Z} = \{\mathrm{z}_1, \cdots, \mathrm{z}_n\}$ be the corresponding set of the so-called hidden random variables such that $\mathrm{z}_i = h$ when $\mathbf{x}_i$ has been generated following $f_h(\mathbf{x}|\theta_h)$. Then, with the knowledge of the values of the hidden variables, the log-likelihood of the observed data is given by

$$\ln P(\mathcal{X}, \mathcal{Z}|\Theta) = \sum_{i=1}^{n} \ln\left(\alpha_{\mathrm{z}_i} f_{\mathrm{z}_i}(\mathbf{x}_i|\theta_{\mathrm{z}_i})\right), \tag{4}$$

from which maximum likelihood parameter estimates can be obtained. However, the values of the hidden variables are not known and so (4) is really a random variable dependent on the distribution of $\mathcal{Z}$, and will be called the *complete data log-likelihood.* Now, for a given $(\mathcal{X}, \Theta)$, it is possible to estimate the most likely conditional distribution of $\mathcal{Z}|(\mathcal{X}, \Theta)$, and this forms the E-step of the EM framework. The exact details of how this estimation is done will be deferred for the moment. In fact we will discuss two ways of estimating the hidden variable distributions that lead to significantly different algorithms. For now, we will assume that the distribution $p(h|\mathbf{x}_i, \Theta) = p(z_i = h|\mathbf{x} = \mathbf{x}_i, \Theta), \forall h$, is known for all the data points.

3.1 The M-step: Parameter Estimation

Suppose the posterior distribution, $p(h|\mathbf{x}_i, \Theta), \forall h, i$, of the hidden variables $\mathcal{Z}|(\mathcal{X}, \Theta)$ is known. Unless otherwise specified, henceforth all expectations will be taken over the distribution of the (set of) random variable(s) $\mathcal{Z}|(\mathcal{X}, \Theta)$. Now, expectation of the complete data log-likelihood, given by (4), over the given distribution p is

$$\begin{aligned} E_p[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] = \sum_{h=1}^{k}\sum_{i=1}^{n}(\ln \alpha_h)\ p(h|\mathbf{x}_i, \Theta) \\ + \sum_{h=1}^{k}\sum_{i=1}^{n}(\ln f_h(\mathbf{x}_i|\theta_h))\ p(h|\mathbf{x}_i, \Theta). \quad (5) \end{aligned}$$

In the parameter estimation or M-step, Θ is re-estimated such that the above expression is maximized. Note that in order to maximize this expression, we can maximize the term containing α_h and the term containing θ_h separately since they have no functional dependencies (note that $p(h|\mathbf{x}_i, \Theta)$ is fixed).

To find the expression for α_h, we introduce the Lagrangian multiplier λ with the constraint that $\sum_{h=1}^{k} \alpha_h = 1$. Taking partial derivatives of the Lagrangian objective function w.r.t. each α_h and solving, we get

$$\hat{\alpha}_h = \frac{1}{n}\sum_{i=1}^{n} p(h|\mathbf{x}_i, \Theta). \quad (6)$$

Next we concentrate on the terms containing $\theta_h = (\boldsymbol{\mu}_h, \kappa_h)$ under the set of constraints $\boldsymbol{\mu}_h^T\boldsymbol{\mu}_h = 1, \kappa_h \geq 0, \forall h$. Then, using Lagrange multipliers $\lambda_1, \cdots, \lambda_k$ corresponding to these constraints[1], the Lagrangian is given by

$$\begin{aligned} L(\{\boldsymbol{\mu}_h, \lambda_h, \kappa_h\}_{h=1}^{k}) = \sum_{h=1}^{k}\Bigg[\sum_{i=1}^{n}(\ln c_d(\kappa_h))\ p(h|\mathbf{x}_i, \Theta) \\ + \sum_{i=1}^{n}\kappa_h\boldsymbol{\mu}_h^T\mathbf{x}_i\ p(h|\mathbf{x}_i, \Theta) + \lambda_h(1 - \boldsymbol{\mu}_h^T\boldsymbol{\mu}_h)\Bigg]. \quad (7) \end{aligned}$$

Taking partial derivatives of (7) with respect to $\{\boldsymbol{\mu}_h, \lambda_h, \kappa_h\}_{h=1}^{k}$ and setting them to zero, for $h = 1, 2, \ldots, k$ we get:

$$\hat{\boldsymbol{\mu}}_h = \frac{\sum_{i=1}^{n}\mathbf{x}_i p(h|\mathbf{x}_i, \Theta)}{\|\sum_{i=1}^{n}\mathbf{x}_i p(h|\mathbf{x}_i, \Theta)\|}, \quad (8)$$

$$A_d(\hat{\kappa}_h) = \frac{\|\sum_{i=1}^{n}\mathbf{x}_i p(h|\mathbf{x}_i, \Theta)\|}{\sum_{i=1}^{n} p(h|\mathbf{x}_i, \Theta)}. \quad (9)$$

where $A_d(\kappa) = \frac{I_{d/2}(\kappa)}{I_{d/2-1}(\kappa)}$. A closed form solution of (9) is not possible and one can use numerical techniques to solve for $\hat{\kappa}_h$. A reasonable approximation to the solution is obtained by setting

$$\hat{\kappa}_h = \frac{\bar{\mathbf{r}}_h d - \bar{\mathbf{r}}_h^3}{1 - \bar{\mathbf{r}}_h^2}, \quad (10)$$

where $\bar{\mathbf{r}}_h = A_d(\hat{\kappa}_h)$ [2].

3.2 The E-step: Distribution estimation

From the standard setting of the EM algorithm [8, 5], (6), (8), and (9) give the update equations for the parameters involved. Given this set of parameters, in this section we outline two schemes for updating the distributions of $\mathcal{Z}|(\mathcal{X}, \Theta)$ so that the likelihood of the data is maximized.

The first update scheme exactly follows the soft-assignment scheme for learning mixture models using EM. From the standard EM framework, the distribution of the hidden variables [23] is given by:

$$p(h|\mathbf{x}_i, \Theta) = \frac{\alpha_h\ f_h(\mathbf{x}_i|\Theta)}{\sum_{l=1}^{k}\alpha_l\ f_l(\mathbf{x}_i|\Theta)}. \quad (11)$$

Our second update scheme is based on the widely used hard-assignment heuristic for unsupervised learning. In this case, the distribution of the hidden variables is given by

$$q(h|\mathbf{x}_i, \Theta) = \begin{cases} 1, & \text{if } h = \underset{h'}{\text{argmax}}\ \ p(h'|\mathbf{x}_i, \Theta), \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

Though soft-assignments are theoretically very well motivated [5, 23], hard-assignments have not received much theoretical attention (though some discussion can be found in [20]). In the rest of this section, we formally study the connection between soft and hard-assignments in the EM framework. Note that this analysis is applicable to general mixture model learning in the EM framework.

At first, following the arguments in [23], we introduce the function $F(\tilde{p}, \Theta)$ given by

$$F(\tilde{p}, \Theta) = E_{\tilde{p}}[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] + H(\tilde{p}). \quad (13)$$

where $\tilde{p}$ is some distribution of $\mathcal{Z}|(\mathcal{X}, \Theta)$. From a maximum-likelihood perspective, the primary objective function that we want to maximize to get the correct mixture model is the *incomplete data log-likelihood* $\ln P(\mathcal{X}|\Theta)$. Now, one interesting property of the function F is that it lower bounds $\ln P(\mathcal{X}|\Theta)$ over choices of $\tilde{p}$. Hence, it makes sense to try to maximize F with respect to $\tilde{p}$. For a given Θ, the $\tilde{p}$ that maximizes $F(\cdot, \Theta)$ is given by (11) [23]. The corresponding optimal value of the function is given by

$$\begin{aligned} F(p, \Theta) &= E_p[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] + H(p) \\ &= E_p[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] - E_p[\ln P(\mathcal{Z}|(\mathcal{X}, \Theta))] \\ &= E_p\left[\ln\left(\frac{P(\mathcal{X}, \mathcal{Z}|\Theta)}{P(\mathcal{Z}|(\mathcal{X}, \Theta))}\right)\right] = E_p[\ln P(\mathcal{X}|\Theta)] \\ &= \ln P(\mathcal{X}|\Theta), \quad (14) \end{aligned}$$

the incomplete data log-likelihood. Further, one can easily see that for a given $\tilde{p}$, the Θ that optimizes $F(\tilde{p}, \cdot)$ is the the same as the one that optimizes the expectation of the complete data log-likelihood in (5). Hence, the M-step is

[1] analysis using the KKT conditions is skipped for brevity. $\kappa_h \geq 0$ is accounted for by taking positive square roots. The final estimates come out to be the same [2].

equivalent to optimizing F with respect to Θ for a given $\tilde{p}$. It follows [5, 23], that the incomplete data log-likelihood, $\ln p(\mathcal{X}|\Theta)$, is non-decreasing at each iteration of the alternate maximization of the function F in terms of Θ and $\tilde{p}$, or equivalently the parameter and the distribution updates given by (6), (8), (9), and (11). Iteration over this set of updates forms the basis of our algorithm `soft-movMF` to be discussed in section 4.

In the hard-assignment case, effectively each of the hidden random variables has a distribution that has probability 1 for one of the mixture components, and 0 for all the others. We shall denote this class of distributions by $\mathcal{H}$. The important question is: is there a way to optimally pick a distribution from $\mathcal{H}$ and then perform a regular M-step, and guarantee, as before, that the incomplete log-likelihood of the data is non-decreasing at each iteration of the update? Unfortunately, this is not possible in general. However, we show that it is possible to reasonably lower bound the incomplete log-likelihood of the data using expectations over $q \in \mathcal{H}$ as in (12). The distribution $q \in \mathcal{H}$ is optimal in the sense that it gives the tightest lower bound to the incomplete log-likelihood among all distributions in $\mathcal{H}$. The lower bound is reasonable in the sense that the expectation over q is itself lower bounded by (5), the expectation of the complete log-likelihood over the distribution p given by (11). Then, an iterative update scheme analogous to regular EM guarantees that this tight lower bound on the incomplete log-likelihood is non-decreasing at each iteration of the update. The parameter estimation, or, M-step remains practically unchanged, with p replaced by q in the update equations (6), (8), and (9).

Since p given by (11) optimizes F for a given Θ, the functional value $F(\cdot,\Theta)$ is smaller for any other choice of $\tilde{p}$. In particular, if $\tilde{p} = q$ as in (12), we have

$$F(q,\Theta) \leq F(p,\Theta) = \ln P(\mathcal{X}|\Theta). \tag{15}$$

Now, all distributions in $\mathcal{H}$ have the property that their entropy is 0. In particular, $H(q) = 0$. Then, from the definition of the function F, we have

$$E_q[\ln P(\mathcal{X},\mathcal{Z}|\Theta)] \leq \ln P(\mathcal{X}|\Theta). \tag{16}$$

So, the expectation over q actually lower bounds the likelihood of the data. Using the definitions of p and q, one can easily prove [2] that $E_p[\ln P(\mathcal{Z}|(\mathcal{X},\Theta))] \leq E_q[\ln P(\mathcal{Z}|(\mathcal{X},\Theta))]$. Now, adding the incomplete data log-likelihood $\ln P(\mathcal{X}|\Theta)$ to both sides of this inequality, we have

$$E_p[\ln P(\mathcal{X},\mathcal{Z}|\Theta)] \leq E_q[\ln P(\mathcal{X},\mathcal{Z}|\Theta)]. \tag{17}$$

Combining (16) and (17), we get

$$E_p[\ln P(\mathcal{X},\mathcal{Z}|\Theta)] \leq E_q[\ln P(\mathcal{X},\mathcal{Z}|\Theta)] \leq \ln P(\mathcal{X}|\Theta). \tag{18}$$

Thus, the expectation over q lies in between the incomplete data likelihood and the expectation of the complete data likelihood over p and hence is a reasonable lower bound to the incomplete data likelihood value.

Finally, we show that our choice of the distribution q is optimal in the sense that the expectation over q gives the tightest lower bound among all distributions in $\mathcal{H}$. Let $\tilde{q}$ be any other distribution in the subset so that $\tilde{q}(h_i|\mathbf{x}_i,\Theta) = 1$. Let $h_i^* = \underset{h}{\text{argmax}}\ p(h|\mathbf{x}_i,\Theta)$. Hence, $p(h_i^*|\mathbf{x}_i,\Theta) \geq p(h_i|\mathbf{x}_i,\Theta), \forall i$. Then,

$$\begin{aligned}
&E_{\tilde{q}}[\ln P(\mathcal{X},\mathcal{Z}|\Theta)] \\
&\quad = \sum_{i=1}^{N}\sum_{h=1}^{k}\tilde{q}(h|\mathbf{x}_i,\Theta)\ln p(h|\mathbf{x}_i,\Theta) = \sum_{i=1}^{N}\ln p(h_i|\mathbf{x}_i,\Theta) \\
&\quad \leq \sum_{i=1}^{N}\ln p(h_i^*|\mathbf{x}_i,\Theta) = \sum_{i=1}^{N}\sum_{h=1}^{k}q(h|\mathbf{x}_i,\Theta)\ln p(h|\mathbf{x}_i,\Theta) \\
&\quad = E_q[\ln P(\mathcal{X},\mathcal{Z}|\Theta)].
\end{aligned}$$

Hence, the choice of q as in (12) is optimal. This analysis forms the basis of our algorithm `hard-movMF` to be discussed in section 4.

4. ALGORITHMS

In this section, we propose two algorithms for clustering directional data based on the development of the previous section. The two algorithms are based on soft and hard-assignment schemes and are respectively called `soft-movMF` and `hard-movMF`. The `soft-movMF` algorithm, presented in Algorithm 1 estimates the parameters of the mixture model exactly following the derivations in section 3 using EM. Hence, it assigns soft (or probabilistic) labels to each point that are given by the posterior probabilities of the components of the mixture conditioned on the point. On termination, the algorithm gives the parameters $\Theta = \{\alpha_h, \boldsymbol{\mu}_h, \kappa_h\}_{h=1}^{k}$ of the k vMF distributions that model the data set $\mathcal{X}$, as well as the *soft-clustering*, i.e., the posterior probabilities $p(h|\mathbf{x}_i,\Theta), \forall h, i$. Appropriate convergence criteria determine when the algorithm should terminate.

Algorithm 1 `soft-movMF`

```
Input: Set $\mathcal{X}$ of data points on $\mathbb{S}^{d-1}$
Output: A soft clustering of $\mathcal{X}$ over a mixture of $k$ vMF distributions
  Initialize all $\alpha_h, \boldsymbol{\mu}_h, \kappa_h,\ h = 1, \cdots, k$
  repeat
    {The E (Expectation) step of EM}
    for $i = 1$ to $n$ do
      for $h = 1$ to $k$ do
        $f_h(\mathbf{x}_i|\theta_h) \leftarrow c_d(\kappa_h)e^{\kappa_h \boldsymbol{\mu}_h^T \mathbf{x}_i}$
        $p(h|\mathbf{x}_i,\Theta) \leftarrow \dfrac{\alpha_h f_h(\mathbf{x}_i|\theta_h)}{\sum_{l=1}^{k}\alpha_l f_l(\mathbf{x}_i|\theta_l)}$
      end for
    end for
    {The M (Maximization) step of EM}
    for $h = 1$ to $k$ do
      $\alpha_h \leftarrow \dfrac{1}{n}\sum_{i=1}^{n} p(h|\mathbf{x}_i,\Theta)$
      $\boldsymbol{\mu}_h \leftarrow \dfrac{\sum_{i=1}^{n}\mathbf{x}_i p(h|\mathbf{x}_i,\Theta)}{\|\sum_{i=1}^{n}\mathbf{x}_i p(h|\mathbf{x}_i,\Theta)\|}$
      $\kappa_h \leftarrow A_d^{-1}\left(\dfrac{\|\sum_{i=1}^{n}\mathbf{x}_i p(h|\mathbf{x}_i,\Theta)\|}{\sum_{i=1}^{n} p(h|\mathbf{x}_i,\Theta)}\right)$
    end for
  until convergence
```

The `hard-movMF` algorithm estimates the parameters of the mixture model by a hard assignment, based on a derived posterior distribution given by (12). The algorithm is obtained from Algorithm 1 by replacing all the posteriors p by the hardened posteriors q as in (12). Thus, after the hard assignments in every iteration, each point *belongs* to a single cluster. As before, the updates of the component parameters are done using the posteriors of the components given the points. The only difference in this case is that the posterior probabilities can take values only in $\{0,1\}$. On termination, the algorithm gives the parameters $\Theta = \{\alpha_h, \boldsymbol{\mu}_h, \kappa_h\}_{h=1}^k$ of the k vMFs that model the data set $\mathcal{X}$ under the hard assignment setup, and the *hard-clustering*, i.e., a disjoint k-partitioning of $\mathcal{X}$ based on the conditional posteriors q on the component vMF distributions.

4.1 Revisiting Spherical Kmeans

We briefly revisit the `spkmeans` algorithm [11], that has been shown to perform well on real life text clustering tasks [11, 3], in the light of the developments of sections 3 and 4. First, we present the `spkmeans` procedure in Algorithm 2. The main observation is that the `spkmeans` algorithm can be looked upon as a special case of the `soft-movMF` as well as the `hard-movMF` algorithms under certain restrictive assumptions on the generative model. To be more precise, say we assume that the generative model of the mixture of vMFs is such that the priors of all the components are the same, i.e., $\alpha_h = 1/k, \forall h$. In order to get `spkmeans` as a reduction from `soft-movMF`, we further assume that all the components have (equal) infinite concentration parameters, i.e., $\kappa_h = \kappa \to \infty, \forall h$. With these assumptions, the E-step reduces to assigning a point to its *nearest* cluster where nearness is computed as a cosine similarity between the point and the cluster representative. Thus, a point $\mathbf{x}_i$ will be assigned to cluster $h^* = \underset{h}{\operatorname{argmax}}\ \mathbf{x}_i^T \boldsymbol{\mu}_h$, since

$$p(h^*|\mathbf{x}_i, \Theta) = \lim_{\kappa \to \infty} \frac{e^{\kappa\ \mathbf{x}_i^T \boldsymbol{\mu}_{h^*}}}{\sum_{h=1}^k e^{\kappa\ \mathbf{x}_i^T \boldsymbol{\mu}_h}} \to 1 \qquad (19)$$

and $p(h|\mathbf{x}_i, \Theta) \to 0, \forall h \neq h^*$.

To show that `spkmeans` can also be seen as a special case of the `hard-movMF` algorithm, in addition to assuming the priors of the components to be equal, we further assume that the concentration parameters of all the components are equal, i.e., $\kappa_h = \kappa, \forall h$. Then, `hard-movMF` reduces to `spkmeans`. With these assumptions on the model, the estimation of the common concentration parameter becomes unessential since the hard assignment will depend only on the value of the cosine similarity $\mathbf{x}_i^T \boldsymbol{\mu}_h$. Given a set of values for $\{\boldsymbol{\mu}_h\}_{h=1}^k$, define $\mathcal{X}_h = \{\mathbf{x} : \mathbf{x} \in \mathcal{X}, h = \operatorname{argmax}_{h'}\ \mathbf{x}^T \boldsymbol{\mu}_{h'}\}$. It is easy to see that $\{\mathcal{X}_h\}_{h=1}^k$ forms a disjoint k-partitioning of $\mathcal{X}$. For a given set of values for $\{\boldsymbol{\mu}_h, \kappa_h\}_{h=1}^k$, we can rewrite the `hard-movMF` algorithm using a similar notation of set partitions, $\mathcal{X}_h = \{\mathbf{x} : \mathbf{x} \in \mathcal{X}, h = \operatorname{argmax}_{h'}\ \kappa_{h'} \mathbf{x}^T \boldsymbol{\mu}_{h'}\}$.

In addition to the above three algorithms, we report experimental results on another algorithm `fskmeans` [3] that belongs to the same class in the sense that, like `spkmeans`, it can be derived from the mixture of vMF models with some restrictive assumptions. In `fskmeans`, the centroids of the mixture components are estimated as in `hard-movMF`. The concentration of a component is *set* to be inversely proportional to the number of points in the cluster corresponding to that component in order to simulate a frequency sensitive competitive learning that implicitly prevents the formation of null clusters, a well-known problem in regular kmeans [4].

Algorithm 2 `spkmeans`

```
Input: Set $\mathcal{X}$ of data points on $\mathbb{S}^{d-1}$
Output: A disjoint $k$-partitioning $\{\mathcal{X}_h\}_{h=1}^k$ of $\mathcal{X}$
  Initialize $\boldsymbol{\mu}_h,\ h = 1, \cdots, k$
  repeat
    {The E (Expectation) step of EM}
    Set $\mathcal{X}_h \leftarrow \phi,\ h = 1, \cdots, k$
    for $i = 1$ to $n$ do
      $\mathcal{X}_h \leftarrow \mathcal{X}_h \cup \{\mathbf{x}_i\}$ where $h = \underset{h'}{\operatorname{argmax}}\ \mathbf{x}_i^T \boldsymbol{\mu}_{h'}$
    end for
    {The M (Maximization) step of EM}
    for $h = 1$ to $k$ do
      $\boldsymbol{\mu}_h \leftarrow \frac{\sum_{\mathbf{x} \in \mathcal{X}_h} \mathbf{x}}{\|\sum_{\mathbf{x} \in \mathcal{X}_h} \mathbf{x}\|}$
    end for
  until convergence
```

5. EXPERIMENTAL RESULTS

In this section we briefly describe the datasets and experimental methodology used. Then, we discuss the performance of the four algorithms under consideration on the various datasets.

5.1 Datasets

We present results on two standard text datasets: 20 Newsgroups[2], and Yahoo News[3], and a dataset of Yeast gene-expression levels.

The *20 Newsgroups* dataset is a collection of 19997 documents from 20 different Usenet newsgroups with (approximately) equal number of documents from each group. We present results on several subsets of this dataset. The full dataset will be called `news20`. A smaller version `small-news20` was created with 100 randomly chosen documents from each of the 20 newsgroups. Various subsets of these two datasets were studied to understand the impact of dataset properties on the relative performance of the algorithms. `news-diff3` is a subset consisting of 3 very different newsgroups (alt.atheism, rec.sport.baseball, sci.space) with 1000 documents per cluster, and `small-news-diff3` consists of the same 3 newsgroups with 100 documents per cluster. `news-sim3` is a subset consisting of 3 very similar newsgroups (comp.graphics, comp.os.ms-windows, comp.windows.x) with 1000 documents per cluster, and `small-news-sim3` consists of the same 3 newsgroups with 100 documents per cluster.

The *Yahoo News* (K-series) dataset is a collection of 2340 Yahoo news articles from 20 different categories. We used the full `yahoo` dataset for experimentation. The underlying clusters in this dataset are highly skewed in terms of the number of documents per clusters, with sizes ranging from 9 to 494.

The Rosetta Inpharmatics *Yeast gene-expression* dataset [14] is a collection of gene-expression vectors constructed using

[2] http://www.ai.mit.edu/people/jrennie/20_newsgroups

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

DNA microarray experiments on the Yeast genes. The original dataset consists of 300 experiments measuring expression of several yeast genes. We used a subset of 996 genes (with known phylogenetic profiles) for our experiments.

5.2 Methodology

Performance of the four algorithms discussed in section4 on all the text datasets have been analyzed using *mutual information* (MI) between the cluster and class labels. The MI gives the amount of statistical similarity between the cluster and class labels [6]. If X is a random variable for the cluster assignments and Y is a random variable for the pre-existing labels on the same data, then their MI is given by $I(X;Y) = E[\ln \frac{p(X,Y)}{p(X)p(Y)}]$ where the expectation is computed on the joint distribution of (X,Y) estimated from a particular clustering of the dataset under consideration. All results reported are averaged over 10 runs and all algorithms were started with the same random initialization. Since the the standard deviations of MI were reasonably small for all algorithms, to reduce clutter, error bars have not been shown in the plots. Also note that for the gene-expression datasets, since class labels are unavailable, the average cosine similarity between each data-point and its cluster representative is used to evaluate the performance of the algorithms.

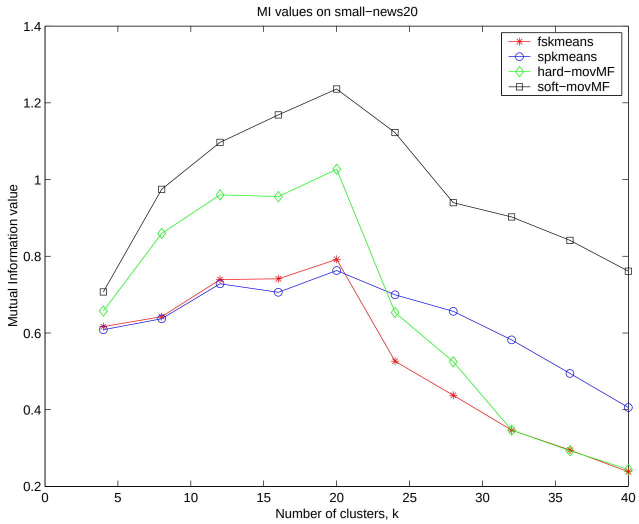

Figure 1: Clustering results on `small-news20`

5.3 Results on 20 Newsgroups

In this section, we briefly discuss the performance of the algorithms on the 20 Newsgroups datasets. All the algorithms performed similarly for the **`news20`** dataset achieving a MI of approximately 1.5 at the correct number of clusters. Experiments with artificial directional data revealed the empirical fact that if the clusters have almost equal priors, have sufficiently large number of data points per cluster, and several of them are reasonably separated the algorithms perform similarly, since the clustering problem is rather simple. On the other hand, if one or more of these conditions are violated, they give quite different clusterings according to their respective biases. Lesion experiments with the various subsets of **`news20`** are performed to study these biases in detail. For example, even though the **`small-news20`** dataset is just a sampled version of the **`news20`** dataset, significant performance differences are observed as shown in Figure 1. The number of documents per cluster being small, **`spkmeans`** and **`fskmeans`** do not perform that well, even for the true number of clusters, since for a small number of points, they become more susceptible to getting trapped in bad local minima of their respective objective functions. On the other hand, **`soft-movMF`** performs significantly better than all the other algorithms over the entire range, while **`hard-movMF`** gives satisfactory MI values till the true number of clusters after which it falls sharply.

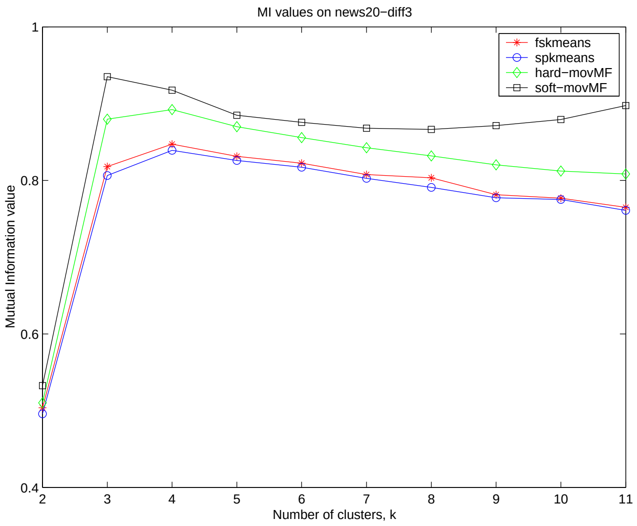

Figure 2: Clustering results on `news20-diff3`

	`spkmeans`			`soft-movMF`		
	I	II	III	I	II	III
Class I	815	1	184	957	5	38
Class II	21	972	7	14	981	5
Class III	58	13	929	27	11	962

Table 1: Confusion matrix for `news20-diff3`

Next, the effect of small number of points per cluster in high-dimensions is studied more closely using the two variants of the **`news20-diff3`** dataset. The basic **`news20-diff3`** dataset is an easy dataset to cluster since it has equal priors, reasonably separate clusters and sufficient number of documents per cluster. However, as we shall shortly see, when the smaller version, **`small-news20-diff`**, of this simple dataset is used, the performance of the algorithms change by significant amounts. The results on **`news20-diff3`** are shown in Figure 2. The performance of all the algorithms are comparable, though the vMF based algorithms give higher values of MI consistently over the entire range of the number of clusters we experimented with. Further, at the correct number of clusters $k = 3$, the vMF-based algorithms perform significantly better. We demonstrate this more clearly by presenting confusion matrices generated by **`spkmeans`** and **`soft-movMF`** for $k = 3$ in Table 1. Among the two vMF based algorithms, **`soft-movMF`** performs consistently better than **`hard-movMF`** over the entire range.

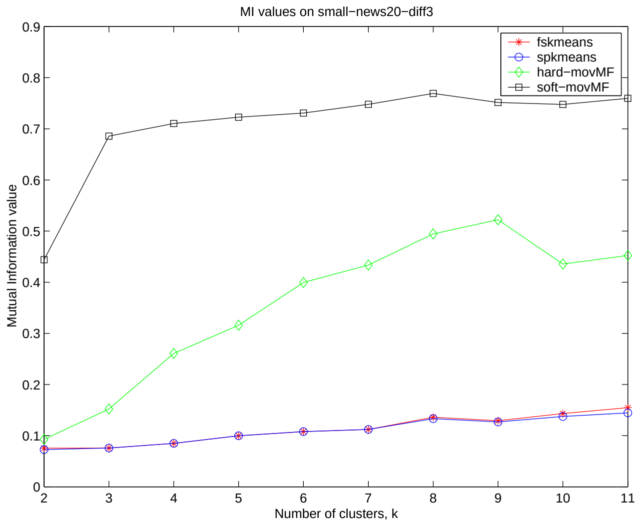

Figure 3: Clustering results on `small-news20-diff3`

	`spkmeans`			**`soft-movMF`**		
	I	II	III	I	II	III
Class I	31	28	41	94	2	4
Class II	26	53	13	1	80	19
Class III	41	21	51	0	18	82

Table 2: Confusion matrix for `small-news20-diff3`

Results on the `small-news20-diff3` dataset are presented in Figure 3. We note that the vMF-based algorithms perform significantly better than **`fskmeans`** and **`spkmeans`**, which show a rather poor performance since the number of data points per cluster is relatively small compared to the dimensionality of the data. In fact, for such high-dimensional data with sparse representation, such as text documents, a few points chosen at random have a high probability of being orthogonal to each other, so that a kmeans-based iterative update scheme may get stuck at a local minimum at the very point it is initialized (see [10]). As a result, kmeans-based algorithms suffer in high-dimensions, especially when the number of data points available is small. This explains the performance of **`fskmeans`** and **`spkmeans`**. Among the vMF-based algorithms, **`soft-movMF`** performs significantly better than **`hard-movMF`** over the entire range. This can be attributed to a very interesting implicit simulated annealing type behavior that **`soft-movMF`** demonstrates. This behavior will be discussed in some detail in section 6. Typical confusion matrices generated by **`spkmeans`** and **`soft-movMF`** for the correct number of clusters are presented in Table 2.

	`spkmeans`			**`soft-movMF`**		
	I	II	III	I	II	III
Class I	618	100	282	644	120	236
Class II	88	629	283	120	766	114
Class III	225	225	550	55	122	823

Table 3: Confusion matrix for `news20-sim3`

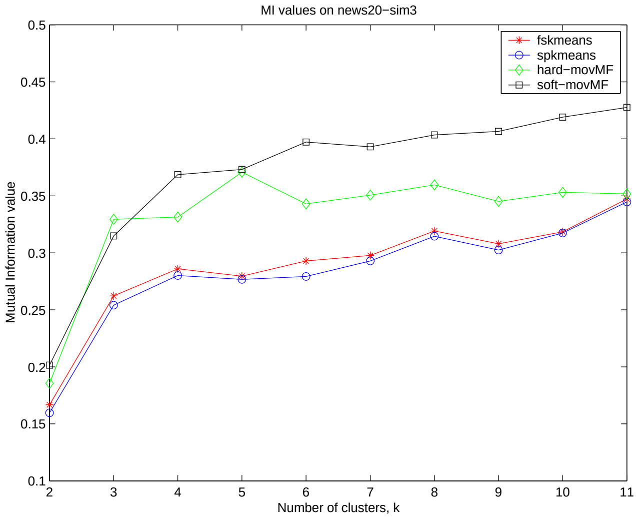

Figure 4: Clustering results on `news20-sim3`

The next two datasets help us study the effect of high-dimensional overlapping clusters on the performance of the algorithms. Results on the **`news20-sim3`** data are presented in Figure 4. Though each of the clusters has sufficient number of points, this is a rather difficult dataset to cluster because the 3 newsgroups are on very similar topics. Since there are enough points to work with, **`fskmeans`** and **`spkmeans`** attain reasonable performance all along, but suffer a little due to the cluster overlaps. Again, the vMF-based algorithms consistently perform better than the kmeans-based algorithms. Among the two vMF-based algorithms, **`soft-movMF`** seems to achieve higher values of MI, though the differences are not always significant. Representative confusion matrices generated by **`spkmeans`** and **`soft-movMF`** are presented in Table 3.

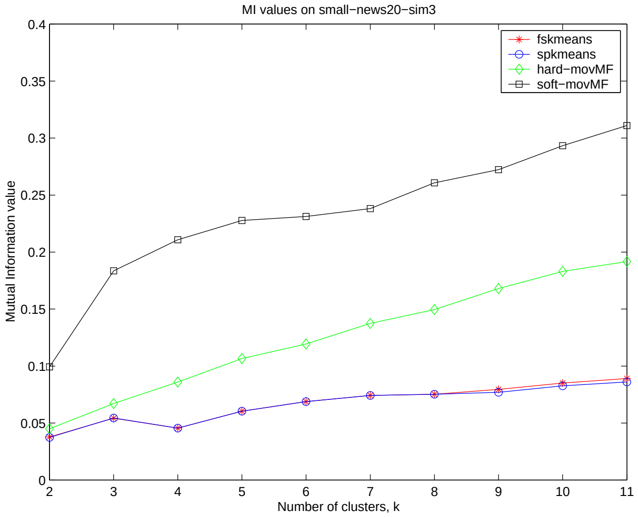

Figure 5: Clustering results on `small-news20-sim3`

The **`small-news20-sim3`** dataset is perhaps the most difficult dataset to cluster among the **`news20`** subsets that we

	spkmeans			soft-movMF		
	I	II	III	I	II	III
Class I	43	28	29	82	47	8
Class II	33	38	29	2	21	27
Class III	34	31	35	16	32	65

Table 4: Confusion matrix for `small-news20-sim3`

consider. It has overlapping clusters with a relatively small number of points per cluster. As expected, **`fskmeans`** and **`spkmeans`** do poorly on this dataset. The vMF-based algorithms perform significantly better as before. Again, **`soft-movMF`** achieves much higher values of MI over the entire range of cluster values over which experiments were performed. Typical confusion matrices are presented in Table 4.

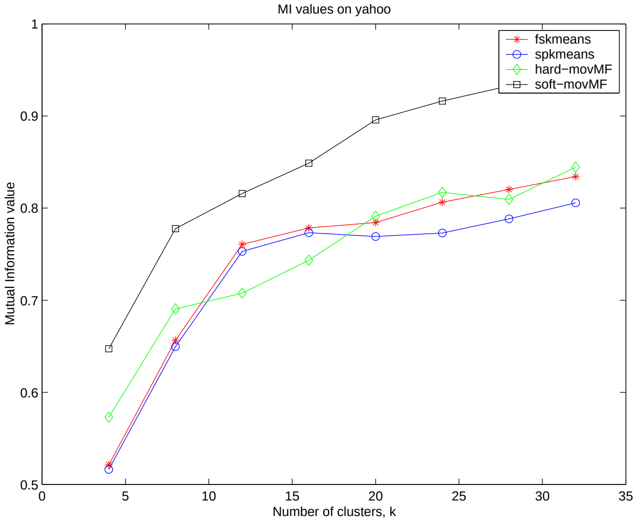

Figure 6: Clustering results on `yahoo`

5.4 Results on Yahoo News

The Yahoo News dataset is a difficult dataset for clustering in the sense that in addition to having some amount of overlap in the clusters and insufficient points per cluster, the clusters are highly skewed in terms of the number of points per cluster. We present results for the different algorithms in Figure 6. Over the entire range, **`soft-movMF`** consistently performs better than the other algorithms. Even at the correct number of clusters $k = 20$, it performs significantly better than the other algorithms. **`fskmeans`** and **`spkmeans`** have a very similar behavior till a moderate number of clusters. For higher numbers of clusters, **`spkmeans`** generates empty clusters because of which its MI does not increase as fast as **`fskmeans`** that has an explicit mechanism for preventing null clusters. The performance of **`hard-movMF`** is interesting; its MI values are comparable and at times worse than those of **`fskmeans`** and **`spkmeans`**. Though it seems to perform marginally better at the correct number of clusters, the improvement is not statistically significant. As seen earlier, **`hard-moVMF`** performed significantly better than the kmeans-based algorithms on the other datasets that we discussed. In fact, similar behavior is observed on all other datasets we experimented with [2]. The skewness of **`yahoo`** in terms of cluster sizes can be a possible reason for its bad performance on this dataset, but experiments on other skewed datasets [2] do not explain this behavior.

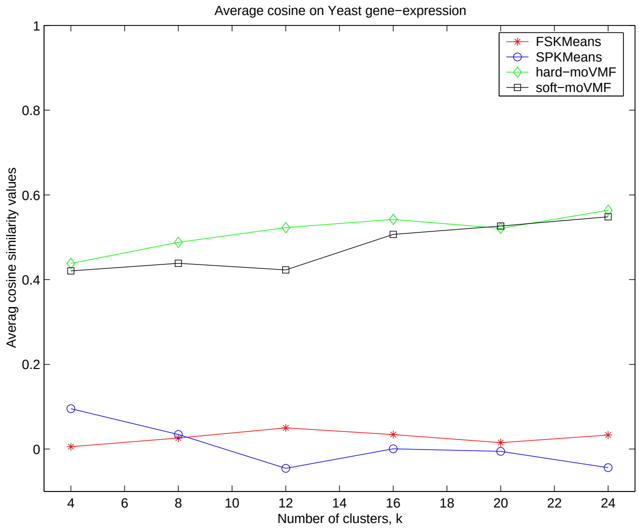

Figure 7: Clustering results on Yeast gene-expression data

5.5 Results on Yeast Gene-expression

Results on the Yeast gene-expression dataset are presented in Figure 7. As can be clearly seen, for a rather different clustering domain with a different performance measure, the vMF-based algorithms perform significantly better than the kmeans-based algorithms. Among the vMF-based algorithms, the performance of **`hard-movMF`** is marginally better than **`soft-movMF`**, but the differences are not significant. It is interesting to note that the vMF-based algorithms, while trying to maximize rather different objective functions, actually perform significantly better than **`spkmeans`** in terms of the average cosine similarity, even though the latter is the objective function that **`spkmeans`** explicitly tries to maximize.

6. DISCUSSION

The mixture of vMF distributions gives a parametric model-based generalization of the widely used cosine similarity measure. As discussed in section 4.1, the spherical kmeans algorithm that uses cosine similarity arises as a special case of the EM on mixture of vMFs when, among other things, the concentration κ of all the distributions is held constant. From a different perspective, we argue that if a particular dataset has been sampled following a vMF distribution with a given κ, say $\kappa = 1$, the *natural* similarity between any pair of data points in that set is given by the cosine similarity. More precisely, if data-points are represented in a feature-space with orthonormal basis, the Fisher-Information matrix is identity, and hence, the Fisher kernel similarity [18] is given by

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= (\nabla_\mu \ln f(\mathbf{x}_i|\boldsymbol{\mu}))^T (\nabla_\mu \ln f(\mathbf{x}_j|\boldsymbol{\mu})) \quad \text{(see (1))} \\ &= (\nabla_\mu (\boldsymbol{\mu}^T \mathbf{x}_i))^T (\nabla_\mu (\boldsymbol{\mu}^T \mathbf{x}_j)) = \mathbf{x}_i^T \mathbf{x}_j , \end{aligned}$$

which is exactly the cosine similarity. In other words, whenever cosine similarity is used in a particular application (not necessarily clustering), the implicit assumption one makes is that the data has been drawn from a vMF distribution.

In terms of performance, the magnitude of improvement shown by the **soft-movMF** algorithm for the difficult clustering tasks was surprising, especially since for low-dimensional non-directional data, the improvements using a soft-assignment based kmeans over the standard hard-assignment based versions are oftentimes quite minimal. In particular, we were curious regarding a couple of issues: (i) why is **soft-movMF** performing substantially better than **hard-movMF**, even though the final probability values obtained by **soft-movMF** are actually very close to 0 and 1; and (ii) why is **soft-movMF**, which needs to estimate more parameters, doing better even when there are insufficient number of points relative to the dimensionality of the space, e.g., all the **small-** datasets.

It turns out that both these issues can be understood by taking a closer look at how the soft-movMF converges. In all our experiments, we initialized κ to 10, and initial centroids to small random perturbations of the global centroid. Hence, for **soft-movMF**, the initial posterior membership distributions of the data points are almost uniform and the entropy $H(p)$ of the hidden random variables is very high. The change of this entropy over iterations for the **news20** subsets is presented in Figure 8. The behavior is similar for all the other datasets. Unlike kmeans-based algorithms where most of the relocation happens in the first two or three iterations with only minor adjustments later on, in **soft-movMF** the data points are noncommittal in the first few iterations, and the entropy remains very high (the maximum possible entropy can be $\log_2 3 = 1.585$). The cluster patterns are discovered only after several iterations, and the entropy drops drastically within a small number of iterations after that. When the algorithm converges, the entropy is practically zero and all points are effectively hard-assigned to their respective clusters. Note that this behavior is strikingly similar to simulated annealing approaches where κ can be considered as the inverse of the temperature parameter. The drastic drop in entropy after a few iterations is the typical critical temperature behavior observed in annealing.

As text data has only non-negative features values, all the data points lie in the first orthant of the d-dimensional hypersphere and hence, is naturally very concentrated. The gene-expression data, though spread all over the hypersphere seemed to have some high concentration regions. In either case, the final κ values on convergence are very high, reflecting the concentration in the data, and implying a low final temperature from the annealing perspective. Then, initializing κ to a low value, or equivalently a high temperature, is a good idea because in that case when **soft-movMF** is running, the κ values will keep on increasing over successive iterations to get to its final large values, giving the effect of a decreasing temperature in the process, without any explicit simulated annealing strategy. This explains why the **soft-movMF** algorithm performs so well for difficult clustering tasks in high-dimensions. The **hard-movMF** algorithm, instead of using the more general vMF model, suffers because of hard-assignments from the very beginning. The fskmeans and **spkmeans** do not do well for difficult datasets due to their hard assignment scheme as well as their significantly less modeling capabilities.

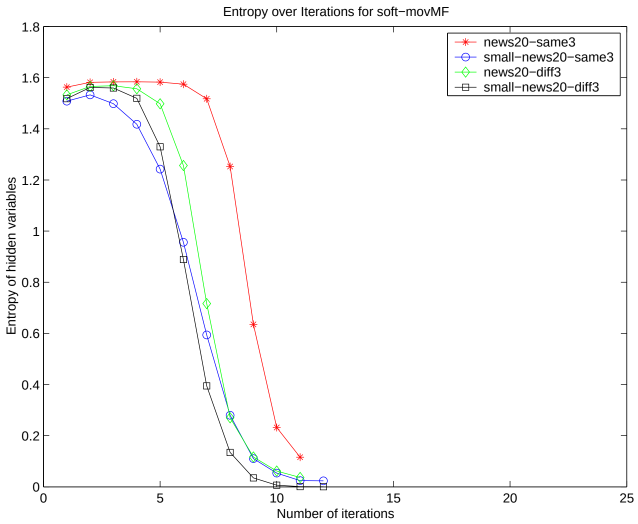

Figure 8: Variation of Entropy of hidden variables with number of Iterations

7. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed two algorithms for clustering directional data. The algorithms are essentially expectation maximization schemes applied to an appropriate generative model, namely a mixture of von Mises-Fisher distributions. We show that the **spkmeans** algorithm [11], that has been shown to work well for text clustering, is a special case of both the proposed schemes. Another high-dimensional clustering algorithm, **fskmeans** [3], is also a special case of one of the proposed algorithms. All the algorithms are comprehensively evaluated using several real-life high dimensional datasets with varying degrees of complexity. From the experimental results, it seems that certain high-dimensional datasets, e.g., text, gene-expression etc., after L_2 normalization have properties that match well with the modeling assumptions of the vMF mixture model. Hence, the proposed algorithms form powerful clustering techniques for such datasets.

The vMF distribution that we considered in the proposed techniques, is one of the simplest parametric distributions for directional data. More general models, e.g., the Fisher-Bingham distribution, have added expressive power and may be useful under certain circumstances. However, the parameter estimation problem is significantly more difficult for such models. Also, one needs substantially more data to get reliable estimates of the parameters. Hence these more complex models may not be viable for most problems. Nevertheless, the tradeoff between model complexity (in terms of the number of parameters), and sample complexity needs to be studied in more detail in the context of directional data. We could also adapt a local search strategy such as the one in [10], for incremental EM to yield better local minima for both hard and soft-assignments.

Acknowledgments: This research was supported in part by NSF grant ECS-9900353 and NSF CAREER Award No. ACI-0093404 and Texas Advanced Research Program grant 003658-0431-2001.

8. REFERENCES

[1] C. Aggarwal. Re-designing distance functions and distance based applications for high dimensional data. *SIGMOD Record*, 30(1), March 2001.

[2] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on hyperspheres using Expectation Maximization. Technical Report TR-03-07, Department of Computer Sciences, University of Texas, February 2003.

[3] A. Banerjee and J. Ghosh. Frequency sensitive competitive learning for clustering on high-dimensional hyperspheres. In *Proceedings International Joint Conference on Neural Networks*, pages 1590–1595, May 2002.

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

[5] M. Collins. The EM algorithm. In fulfillment of Written Preliminary Exam II requirement, September 1997.

[6] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley-Interscience, 1991.

[7] S. Dasgupta. Learning mixtures of Gaussians. In *IEEE Symposium on Foundations of Computer Science*, 1999.

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

[9] I. Dhillon and J. Kogan, editors. *Proc. Workshop on Clustering High Dimensional Data and its Applications.* SIAM, 2002.

[10] I. S. Dhillon, Y. Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In *Proceedings of The 2002 IEEE International Conference on Data Mining*, 2002.

[11] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.

[12] I. S. Dhillon and S. Sra. Modeling data using directional distributions. Technical Report TR-06-03, University of Texas, Dept. of Computer Sciences, February 2003.

[13] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci.*, 95:14863–14868, 1998.

[14] T. R. H. et. al. Functional discovery via a compendium of expression profiles. *Cell*, 102:109–126, 2000.

[15] J. Ghosh. Scalable clustering methods for data mining. In N. Ye, editor, *Handbook of Data Mining*, pages 247–277. Lawrence Erlbaum, 2003.

[16] G. K. Gupta and J. Ghosh. Detecting seasonal and divergent trends and visualization for very high dimensional transactional data. In *Proc. 1st SIAM Intl. Conf. on Data Mining*, April 2001.

[17] P. Indyk. A sublinear-time approximation scheme for clustering in metric spaces. In *40th Symposium on Foundations of Computer Science*, 1999.

[18] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In M. S. Kearns, S. A. Solla, and D. D. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 487–493. MIT Press, 1999.

[19] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, New Jersey, 1988.

[20] M. Kearns, Y. Mansour, and A. Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. In *13th Annual Conf. on Uncertainty in Artificial Intelligence (UAI97)*, 1997.

[21] K. V. Mardia. *Statistical Distributions in Scienctific Work*, volume 3, chapter Characteristics of directional distributions, pages 365–385. Reidel, Dordrecht, 1975.

[22] K. V. Mardia and P. Jupp. *Directional Statistics.* John Wiley and Sons Ltd., 2nd edition, 2000.

[23] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press, 1998.

[24] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[25] C. R. Rao. *Linear Statistical Inference and its Applications.* Wiley, New York, 2nd edition, 1973.

[26] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *World Wide Web*, 10, pages 285–295, 2001.

[27] B. Schölkopf and A. Smola. *Learning with Kernels.* MIT Press, 2001.

[28] J. Sinkkonen and S. Kaski. Clustering based on conditional distributions in an auxiliary space. *Neural Computation*, 14:217–239, 2001.

[29] P. Smyth. Clustering sequences with hidden Markov models. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing*, volume 9, pages 648–654. MIT Press, 1997.

[30] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proc 7th Natl Conf on Artificial Intelligence : Workshop of AI for Web Search (AAAI 2000)*, pages 58–64. AAAI, July 2000.