

Expectation Maximization for Clustering on Hyperspheres

Arindam Banerjee* Inderjit Dhillon† Joydeep Ghosh* Suvrit Sra†

28 February 2003‡

Technical Report # TR-03-07

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 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. We provide several results on modeling high-dimensional text and gene data as experimental validation. The results indicate that our approach yields superior clusterings especially for difficult clustering tasks in high-dimensional space.

1 Introduction

Clustering or segmentation of data is a fundamental data analysis step that has been widely studied across multiple disciplines [JD88, Mac67]. 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. These large datasets also impose demands on the evaluation, scalability and visualization of results [Gho03]. 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* and *discriminative* ones. In a generative approach [Smy97, Bil98, Ros98, JH99], the data is modeled as being generated by an underlying parametric, probabilistic process. Values for the parameters are estimated from the input data, and properties of the clusters are then inferred from these parameters. Discriminative approaches [Vap98, SS01, Ind99], on the other hand, make no assumptions whatsoever about how the data points were generated. Instead, they assume that a well defined distance or similarity measure exists between any pair of objects. The clustering process is then essentially an attempt

*Department of E.C.E., Univ. of Texas at Austin

†Department of Comp. Sci., Univ. of Texas at Austin

‡Revised 8th June 2003.

to partition the objects so that intra-partition (within cluster) dissimilarities are smaller than inter-partition (across clusters) dissimilarities¹.

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 distributions. 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 the generative models so that clustering of data brings out specific desirable patterns that one is looking for. It is for this reason that the generative (parametric) approach is referred to as the method of particular inference in statistical learning theory [Vap98].

Clustering algorithms using the generative model framework, often involve an appropriate application of the Expectation Maximization (EM) algorithm [DLR77, Col97] on a properly chosen statistical generative model for the data under consideration. At present, 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. However, in many cases such distances are not appropriate, e.g., from empirical studies in information retrieval applications, *cosine similarity* has been found to be a good measure of similarity for analyzing and clustering text documents. Thus, some domains require the use of *directional data*[MJ00] — data in which only the direction of the vectors is considered and all the vectors involved have unit Euclidean norm. In fact several large, high-dimensional datasets exhibit directional characteristics. These motivations suggest the need for generative models that are more appropriate for the analysis and clustering of directional data. In this article, we present 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.

In order to motivate our work, we present examples of a few important domains where directional data is becoming increasingly common. One such domain is text analysis, and text clustering in particular. It has been experimentally demonstrated that in order to remove the biases arising from the length of a document, it often helps to normalize the data vectors [DM01]. Note that running a clustering algorithm such as `kmeans` [DHS00, Mac67] on normalized data so that the total *Euclidean distortion* in the data with respect to the cluster representative is minimized, is not a reasonable idea since the Euclidean distortion is not a natural measure of dispersion for normalized data. On the other hand, if the cluster representatives themselves are normalized, then the Euclidean distortion (chordal distance) is negatively proportional to the *cosine similarity* in the data with respect to the cluster representative. Given $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ such that $\|\mathbf{x}\| = \|\mathbf{y}\| = 1$, we have

$$\|\mathbf{x} - \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\mathbf{x}^T\mathbf{y} = 2 - 2\mathbf{x}^T\mathbf{y},$$

where the inner product $\mathbf{x}^T\mathbf{y}$ is the so-called cosine similarity, since it equals the cosine of the angle between \mathbf{x} and \mathbf{y} . Thus, minimizing the Euclidean distortion with respect to normalized cluster representatives is equivalent to maximizing the cosine similarity of the data points with the cluster representatives. However, note that there is a significant conceptual difference compared to the classical approach since, the representatives themselves are normalized in this case. Therefore, maximizing cosine similarity in a particular clustering scheme is tantamount to considering only the directional properties of the data. The use of cosine similarity has been shown to work well for very high-dimensional text collections [DM01, BG02, BBM02] using the standard vector-space model [FBY92, Sal89].

¹Note that the approaches are not mutually exclusive, as `kmeans` can be seen as a member of both the approaches [KMN97].

Another domain in which directional data shows up is bioinformatics. At present, gene expression datasets are a major source of information about genes and their interactions. DNA microarrays measure the mRNA expression of all genes encoded by a genome in a single experiment [LRM⁺97]. From these microarray experiments an expression vector for each gene is constructed. The expression vector describes the expression level of a gene subject to a range of cellular conditions, cell types, genetic backgrounds, etc. Thus analysis of gene expression data can prove to be a valuable source of information for understanding and predicting functions of genes. A fundamental analysis step is the clustering of genes that exhibit similar expression levels. Given enough independent experiments, genes clustered in this fashion tend to be functionally related [ESBB98, MPT⁺99].

A similarity measure that has been found to be useful in the gene clustering domain is the Pearson correlation coefficient of the expression levels of genes. Given $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, the Pearson product moment correlation between them is given by

$$\rho(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^d (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^d (x_i - \bar{x})^2} \times \sqrt{\sum_{i=1}^d (y_i - \bar{y})^2}}, \quad (1.1)$$

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 $\rho(\mathbf{x}, \mathbf{y}) = \tilde{\mathbf{x}}^T \tilde{\mathbf{y}}$. Thus, the Pearson correlation is exactly the cosine similarity between $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ that are obtained by a simple mapping from the given feature vectors \mathbf{x} and \mathbf{y} . Hence, analysis and clustering of gene expression data using Pearson correlations is essentially a clustering problem using directional data. In fact, any algorithm that uses Pearson correlations, e.g., a number of standard techniques in recommender systems based on collaborative filtering [SKKR01], can be considered as a problem involving directional data and hence should be analyzed under the domain of directional statistics [MJ00].

In [DM01], the **spkmeans** algorithm for clustering spherical data was proposed. The connection between a generative model involving von Mises-Fisher distributions and the **spkmeans** algorithm was first observed in [BG02]. In this article, we propose a mixture model on the unit hypersphere for modeling directional data based on the von Mises-Fisher (vMF) distribution and derive two EM-based clustering algorithms based on the same. To this end, we first present the vMF distribution on the sphere and discuss the maximum likelihood parameter estimates for a single vMF in section 2. In section 3, we introduce the generative model for a mixture of vMF distributions and analyze the maximum likelihood parameter estimates of the mixture model from a given dataset using the EM framework. Based on the analysis in section 3, two clustering algorithms, using soft- and hard-assignments respectively, are proposed in section 4. We show the connection of the proposed algorithms to the **spkmeans** algorithm in section 4.1. A brief discussion of the numerical implementation issues is in section 4.2. A summary of artificial and real life datasets used for experimentation is presented in section 5. We discuss the experimental results on various datasets in section 6. Some limitations of our present approach and the clustering process in general are discussed in section 7. Section 8 presents concluding remarks and outlines directions for future work.

A word about the notation: bold faced variables, e.g., \mathbf{x} , $\boldsymbol{\mu}$, represent vectors; $\|\cdot\|$ denotes the L_2 norm; sets are represented by calligraphic upper-case letter, e.g., \mathcal{X} , \mathcal{Z} ; \mathbb{R} denotes the set of reals; \mathbb{S}^{d-1} denotes the $(d-1)$ -dimensional unit hypersphere embedded in \mathbb{R}^d . Probability density functions are denoted by lower case alphabets, e.g., f , p ; probability of sets of events is denoted by capital letter, e.g., P , Q . If a random variable z is distributed as p , we denote this by $z \sim p$. Expectation of functions of a random variable $z \sim p$ are denoted by $E_{z \sim p}[\cdot]$, or, simply $E_p[\cdot]$ if the random variable is distributed as p . Unless otherwise mentioned, we study data that lies on

the unit hypersphere, i.e., is L_2 -normalized — we use the terms “directional data” and “spherical data” interchangeably when talking about such data.

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 $M_d(\boldsymbol{\mu}, \kappa)$, if its probability density function is given by:

$$f(\mathbf{x}|\boldsymbol{\mu}, \kappa) = c_d(\kappa)e^{\kappa\boldsymbol{\mu}^T\mathbf{x}} \quad \mathbf{x}, \boldsymbol{\mu} \in \mathbb{S}^{d-1} \subseteq \mathbb{R}^d, \kappa \in \mathbb{R}_+ \cup \{0\}. \quad (2.1)$$

The normalizing constant $c_d(\kappa)$ is given by (see [DS03] for detailed derivations)

$$c_d(\kappa) = \frac{\kappa^{d/2-1}}{(2\pi)^{d/2}I_{d/2-1}(\kappa)}, \quad (2.2)$$

where $I_r(\kappa)$ represents the modified Bessel function of the first kind of order r and argument κ (see [AS74] for more on Bessel functions). The distribution $M_d(\boldsymbol{\mu}, \kappa)$ is parameterized by the mean direction $\boldsymbol{\mu}$, and the concentration parameter κ , so-called because it characterizes how strongly the unit vectors following $M_d(\boldsymbol{\mu}, \kappa)$ are concentrated about the mean direction $\boldsymbol{\mu}$. Larger values of κ imply stronger concentration about the mean direction. In particular when $\kappa = 0$, $M_d(\boldsymbol{\mu}, \kappa)$ reduces to the uniform distribution on \mathbb{S}^{d-1} , and as $\kappa \rightarrow \infty$, $M_d(\boldsymbol{\mu}, \kappa)$ tends to a point distribution concentrated at $\boldsymbol{\mu}$.

The von Mises-Fisher distribution is natural for directional data and has properties analogous to those of the multi-variate normal distribution in \mathbb{R}^d . Mardia and Jupp [MJ00] mention that the density on \mathbb{S}^{d-1} that maximizes the entropy, while keeping $E[\mathbf{x}]$ fixed, is a vMF density. See Rao [Rao73, pp. 172–174] and Mardia [Mar75] for a proof. A maximum likelihood characterization says: let $f(\mathbf{x}; \boldsymbol{\mu})$ be a probability density function on \mathbb{S}^{d-1} with mean direction $\boldsymbol{\mu}$ and $E[\mathbf{x}] = \zeta\boldsymbol{\mu}$ where $\zeta > 0$. If for all random samples the sample mean direction is a maximum likelihood estimate (m.l.e.) of $\boldsymbol{\mu}$ and $f(\mathbf{x}; \boldsymbol{\mu}) = g(\mathbf{x}^T\boldsymbol{\mu})$ for all $\mathbf{x} \in \mathbb{S}^{d-1}$, where the function g is lower semi-continuous from the left at 1, then $f(\mathbf{x}; \boldsymbol{\mu})$ is a von Mises-Fisher density. Various other results related to the vMF density and its genesis are given in [MJ00].

2.1 Maximum Likelihood Estimates

We now briefly discuss the maximum likelihood estimation of the parameters of a single vMF distribution from a given data set. Let \mathcal{X} be a finite set of sample unit vectors drawn independently following $M_d(\boldsymbol{\mu}, \kappa)$, i.e., we have

$$\mathcal{X} = \{\mathbf{x}_i \mid \mathbf{x}_i \sim M_d(\boldsymbol{\mu}, \kappa) \text{ for } 1 \leq i \leq n\}.$$

We want to find maximum likelihood estimates for the parameters $\boldsymbol{\mu}$ and κ of the distribution $M_d(\boldsymbol{\mu}, \kappa)$. Assuming each $\mathbf{x}_i \in \mathcal{X}$ to be independent we can write the likelihood of \mathcal{X} as:

$$P(\mathcal{X}|\boldsymbol{\mu}, \kappa) = P(\mathbf{x}_1, \dots, \mathbf{x}_n|\boldsymbol{\mu}, \kappa) = \prod_{i=1}^n f(\mathbf{x}_i|\boldsymbol{\mu}, \kappa) = \prod_{i=1}^n c_d(\kappa)e^{\kappa\boldsymbol{\mu}^T\mathbf{x}_i}. \quad (2.3)$$

Taking logarithm of both sides of (2.3), the log-likelihood of the data set \mathcal{X} becomes

$$\mathcal{L}(\mathcal{X}|\boldsymbol{\mu}, \kappa) = \ln P(\mathcal{X}|\boldsymbol{\mu}, \kappa) = n \ln c_d(\kappa) + \kappa\boldsymbol{\mu}^T \mathbf{r}, \quad (2.4)$$

where $\mathbf{r} = \sum_i \mathbf{x}_i$. To obtain the maximum likelihood parameter estimates of $\boldsymbol{\mu}$ and κ , we have to maximize (2.4), subject to the constraints $\boldsymbol{\mu}^T \boldsymbol{\mu} = 1$ and $\kappa \geq 0$. Introducing a Lagrange multiplier λ , the Lagrangian of the objective function is given by²

$$L(\boldsymbol{\mu}, \lambda, \kappa; \mathcal{X}) = n \ln c_d(\kappa) + \kappa \boldsymbol{\mu}^T \mathbf{r} + \lambda(1 - \boldsymbol{\mu}^T \boldsymbol{\mu}). \quad (2.5)$$

Now, taking derivatives of the Lagrangian with respect to $\boldsymbol{\mu}$, λ and κ and setting them to zero, we get the following equations that the parameter estimates $\hat{\boldsymbol{\mu}}$, $\hat{\lambda}$ and $\hat{\kappa}$ must satisfy:

$$\hat{\boldsymbol{\mu}} = \frac{\hat{\kappa}}{2\hat{\lambda}} \mathbf{r}, \quad (2.6a)$$

$$\hat{\boldsymbol{\mu}}^T \hat{\boldsymbol{\mu}} = 1, \quad (2.6b)$$

$$\frac{nc'_d(\hat{\kappa})}{c_d(\hat{\kappa})} = -\hat{\boldsymbol{\mu}}^T \mathbf{r}. \quad (2.6c)$$

Substituting (2.6a) in (2.6b) gives us

$$\hat{\lambda} = \frac{\hat{\kappa}}{2} \|\mathbf{r}\|, \quad (2.7)$$

$$\hat{\boldsymbol{\mu}} = \frac{\mathbf{r}}{\|\mathbf{r}\|} = \frac{\sum_{i=1}^n \mathbf{x}_i}{\|\sum_{i=1}^n \mathbf{x}_i\|}, \quad \text{by (2.6a)}. \quad (2.8)$$

Substituting (2.8) in (2.6c), we get

$$\frac{c'_d(\hat{\kappa})}{c_d(\hat{\kappa})} = -\frac{\|\mathbf{r}\|}{n}. \quad (2.9)$$

For brevity, let us write $s = d/2 - 1$; on differentiating (2.2) w.r.t. κ we obtain:

$$c'_d(\kappa) = \frac{s\kappa^{s-1}}{cI_s(\kappa)} - \frac{\kappa^s I'_s(\kappa)}{cI_s^2(\kappa)}, \quad (2.10)$$

where $c = (2\pi)^{s+1}$ is a constant. The right hand side simplifies to

$$\frac{\kappa^s}{cI_s(\kappa)} \left(\frac{s}{\kappa} - \frac{I'_s(\kappa)}{I_s(\kappa)} \right) = c_d(\kappa) \left(\frac{s}{\kappa} - \frac{I'_s(\kappa)}{I_s(\kappa)} \right). \quad (2.11)$$

Using the following well known recurrence relation ([AS74] 9.6.26)

$$\kappa I_{s+1}(\kappa) = \kappa I'_s(\kappa) - s I_s(\kappa), \quad (2.12)$$

we finally obtain:

$$\frac{-c'_d(\kappa)}{c_d(\kappa)} = \frac{I_{s+1}(\kappa)}{I_s(\kappa)} = \frac{I_{d/2}(\kappa)}{I_{d/2-1}(\kappa)}. \quad (2.13)$$

Thus we can obtain the m.l.e. $\hat{\kappa}$ by solving

$$A_d(\hat{\kappa}) = \frac{\|\mathbf{r}\|}{n}, \quad (2.14)$$

where $A_d(\kappa) = \frac{I_{d/2}(\kappa)}{I_{d/2-1}(\kappa)}$. In Section 4.2 we shall discuss the computation of $\hat{\kappa}$ from (2.14) using various approximations.

²strictly speaking, we should introduce the inequality constraint for κ in the Lagrangian and work with the necessary KKT conditions. If we assume κ cannot be 0, then the multiplier for the inequality constraint has to be zero by the complementary slackness conditions, and the Lagrangian we are working with is adequate.

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. We 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

$$f(\mathbf{x}|\Theta) = \sum_{h=1}^k \alpha_h f_h(\mathbf{x}|\theta_h), \quad (3.1)$$

where $\Theta = \{\alpha_1, \dots, \alpha_k, \theta_1, \dots, \theta_k\}$, $\alpha_h \geq 0$, $\sum_{h=1}^k \alpha_h = 1$ 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 $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ be a data set generated by sampling independently following this generative model. Let $\mathcal{Z} = \{z_1, \dots, z_n\}$ be the corresponding set of the so-called hidden random variables such that $z_i = h$ 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

$$\ln P(\mathcal{X}, \mathcal{Z}|\Theta) = \sum_{i=1}^n \ln(\alpha_{z_i} f_{z_i}(\mathbf{x}_i|\theta_{z_i})), \quad (3.2)$$

from which maximum likelihood parameter estimates can be obtained. However, the values of the hidden variables are not known, whereby (3.2) 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}, Θ) , it is possible to obtain an estimate of 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 conditional distribution, $p(h|\mathbf{x}_i, \Theta), \forall h, i$, of the hidden variables $\mathcal{Z}|\mathcal{X}, \Theta$ is known. Unless otherwise specified, from this point onward, all expectations will 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 (3.2), over the given distribution p , is given by

$$\begin{aligned} E_p[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] &= \sum_{i=1}^n E_{p(z_i|\mathbf{x}_i, \Theta)}[\ln(\alpha_{z_i} f_{z_i}(\mathbf{x}_i|\theta_{z_i}))] \\ &= \sum_{i=1}^n \sum_{h=1}^k \ln(\alpha_h f_h(\mathbf{x}_i|\theta_h)) p(h|\mathbf{x}_i, \Theta) \\ &= \sum_{h=1}^k \sum_{i=1}^n (\ln \alpha_h) p(h|\mathbf{x}_i, \Theta) + \sum_{h=1}^k \sum_{i=1}^n (\ln f_h(\mathbf{x}_i|\theta_h)) p(h|\mathbf{x}_i, \Theta). \end{aligned} \quad (3.3)$$

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 are not related (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 $\sum_{h=1}^k \alpha_h = 1$ and take partial derivatives of the Lagrangian objective function w.r.t. each α_h to get

$$\frac{\partial}{\partial \alpha_h} \left[\sum_{h=1}^k \sum_{i=1}^n (\ln \alpha_h) p(h|\mathbf{x}_i, \Theta) + \lambda \left(\sum_{h=1}^k \alpha_h - 1 \right) \right] = 0 \quad \Rightarrow \quad \sum_{i=1}^n p(h|\mathbf{x}_i, \Theta) = -\lambda \alpha_h, \quad \forall h. \quad (3.4)$$

Summing both sides of (3.4) over all h , we get $\lambda = -n$ and hence from (3.4) we have

$$\alpha_h = \frac{1}{n} \sum_{i=1}^n p(h|\mathbf{x}_i, \Theta). \quad (3.5)$$

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 $\lambda_1, \dots, \lambda_k$ corresponding to these constraints, the Lagrangian is given by

$$\begin{aligned} L(\{\boldsymbol{\mu}_h, \lambda_h, \kappa_h\}_{h=1}^k) &= \sum_{h=1}^k \sum_{i=1}^n (\ln f_h(\mathbf{x}_i|\theta_h)) p(h|\mathbf{x}_i, \Theta) + \sum_{h=1}^k \lambda_h (1 - \boldsymbol{\mu}_h^T \boldsymbol{\mu}_h) \\ &= \sum_{h=1}^k \left[\sum_{i=1}^n (\ln c_d(\kappa_h)) p(h|\mathbf{x}_i, \Theta) + \sum_{i=1}^n \kappa_h \boldsymbol{\mu}_h^T \mathbf{x}_i p(h|\mathbf{x}_i, \Theta) + \lambda_h (1 - \boldsymbol{\mu}_h^T \boldsymbol{\mu}_h) \right]. \end{aligned} \quad (3.6)$$

Note that (3.6) is quite similar to the Lagrangian in (2.5) for a single vMF distribution. Now, taking partial derivatives of (3.6) with respect to $\{\boldsymbol{\mu}_h, \lambda_h, \kappa_h\}_{h=1}^k$ and setting them to zero, we get the following set of equations for $h = 1, 2, \dots, k$:

$$\boldsymbol{\mu}_h = \frac{\kappa_h}{2\lambda_h} \sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta), \quad (3.7a)$$

$$\boldsymbol{\mu}_h^T \boldsymbol{\mu}_h = 1 \quad \forall h, \quad (3.7b)$$

$$\frac{c'_d(\kappa_h)}{c_d(\kappa_h)} \sum_{i=1}^n p(h|\mathbf{x}_i, \Theta) = -\boldsymbol{\mu}_h^T \sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta). \quad (3.7c)$$

As before, substituting (3.7a) in (3.7b), we get

$$\lambda_h = \frac{\kappa_h}{2} \left\| \sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta) \right\|, \quad (3.8)$$

$$\Rightarrow \boldsymbol{\mu}_h = \frac{\sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta)}{\left\| \sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta) \right\|}, \quad \text{by (3.7a)}. \quad (3.9)$$

Finally, substituting (3.9) in (3.7c), we get

$$\frac{c'_d(\kappa_h)}{c_d(\kappa_h)} = -\frac{\left\| \sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta) \right\|}{\sum_{i=1}^n p(h|\mathbf{x}_i, \Theta)}, \quad (3.10)$$

which simplifies to

$$A_d(\kappa_h) = \frac{\left\| \sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta) \right\|}{\sum_{i=1}^n p(h|\mathbf{x}_i, \Theta)}, \quad (3.11)$$

where $A_d(\kappa) = \frac{I_{d/2}(\kappa)}{I_{d/2-1}(\kappa)}$. Note that (3.9) and (3.11) are intuitive generalizations of (2.8) and (2.14) respectively.

3.2 The E-step: Distribution estimation

From the standard setting of the EM algorithm [DLR77, Col97, Bil98], (3.5), (3.9), and (3.11) give the update equations of the parameters involved. Given this set of parameters, the distribution of the hidden variables can be computed [NH98, Bil98] as:

$$p(h|\mathbf{x}_i, \Theta) = \frac{\alpha_h f_h(\mathbf{x}_i|h, \Theta)}{\sum_{l=1}^k \alpha_l f_l(\mathbf{x}_i|l, \Theta)}. \quad (3.12)$$

It can be shown [Col97] that the *incomplete data log-likelihood*, $\ln p(\mathcal{X}|\Theta)$, is non-decreasing at each iteration of the parameter and distribution update. Iteration over this set of updates forms the basis of our algorithm **soft-moVMF** to be discussed in section 4.

Next, we discuss a nonstandard setting of the EM algorithm that is more practical for clustering algorithms. In this setting, points are *hard-assigned* to clusters so that effectively each of the hidden variables has a distribution that has probability 1 for one of the mixture components and zero for all the others. We shall denote this class of distributions as \mathcal{H} . Note that this does not follow naturally from the mixture model setting. However, we investigate this setting because of its computational efficiency and usability in practical clustering problems.

Consider the same mixture model setting, with the added constraint that the distributions of the hidden variables are restricted to \mathcal{H} . Note that \mathcal{H} is a subset of all possible distributions on the events, and for a typical mixture model, the distribution following (3.12) will not belong to this subset. 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 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 an *optimal* distribution $q \in \mathcal{H}$. The distribution $q \in \mathcal{H}$ is called optimal because 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 (3.3), the expectation of the complete log-likelihood over the distribution p given by (3.12). Then, an iterative update scheme analogous to regular EM guarantees that the 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 (3.5), (3.9), and (3.11). In the E-step, the optimal q is estimated. Thus, until a saddle-point is reached, this scheme does a greedy-optimal maximization of a lower bound on the incomplete log-likelihood of the data. This justifies the use of hard-assignment for learning mixture models and forms the basis of our algorithm **hard-moVMF** to be discussed in section 4.

At first, we state our hard assignment rule: if the optimal distribution in \mathcal{H} is denoted by $q(h|\mathbf{x}_i, \Theta)$, then

$$q(h|\mathbf{x}_i, \Theta) = \begin{cases} 1, & \text{if } h = \operatorname{argmax}_{h'} p(h'|\mathbf{x}_i, \Theta), \\ 0, & \text{otherwise.} \end{cases} \quad (3.13)$$

We justify this hard assignment in two steps. First we show that the expectation over q is a reasonable lower bound of the incomplete log-likelihood of the data so that maximizing this expectation makes sense. Then, we show that the choice of this particular hard-assignment is the optimum in the sense that it gives the tightest lower bound among all distributions in \mathcal{H} .

At first, following the arguments in [NH98], we introduce the function $F(\tilde{p}, \Theta)$ given by

$$F(\tilde{p}, \Theta) = E_{\tilde{p}}[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] + H(\tilde{p}). \quad (3.14)$$

This function has the property that the E- and M-steps of the EM algorithm can be shown to *alternately maximize* this function. In the E-step, for a given value of Θ , the distribution \tilde{p} is chosen to maximize $F(\tilde{p}, \Theta)$ for that Θ , and in the M-step, for a given value of \tilde{p} , the parameters Θ are estimated to maximize $F(\tilde{p}, \Theta)$ for the given \tilde{p} . It can be shown [NH98] that for a given Θ , the optimal \tilde{p} is given by (3.12). When $\tilde{p} = p$, the optimal value of the function is given by

$$\begin{aligned}
F(p, \Theta) &= E_p[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] + H(p) \\
&= E_p[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] - E_p[\ln P(\mathcal{Z}|(\mathcal{X}, \Theta))] \\
&= E_p \left[\ln \left(\frac{P(\mathcal{X}, \mathcal{Z}|\Theta)}{P(\mathcal{Z}|(\mathcal{X}, \Theta))} \right) \right] = E_p[\ln P(\mathcal{X}|\Theta)] \\
&= \ln P(\mathcal{X}|\Theta).
\end{aligned} \tag{3.15}$$

Since (3.12) gives the optimal choice of the distribution, the functional value is smaller for any other choice of \tilde{p} . In particular, if $\tilde{p} = q$ as in (3.13), we have

$$F(q, \Theta) \leq F(p, \Theta) = \ln P(\mathcal{X}|\Theta). \tag{3.16}$$

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

$$E_q[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] \leq \ln P(\mathcal{X}|\Theta). \tag{3.17}$$

Thus, the expectation over q actually lower bounds the likelihood of the data. We go one step further to show that this is in fact a reasonably tight lower bound in the sense that the expectation over q is lower bounded by the expectation over p of the complete data log-likelihood. To this end, at first we prove the following result.

Lemma 1 $E_p[\ln P(\mathcal{Z}|(\mathcal{X}, \Theta))] \leq E_q[\ln P(\mathcal{Z}|(\mathcal{X}, \Theta))]$

Proof: Let $h_i^* = \underset{h}{\operatorname{argmax}} p(h|\mathbf{x}_i, \Theta)$. Then, $p(h|\mathbf{x}_i, \Theta) \leq p(h_i^*|\mathbf{x}_i, \Theta), \forall h$. Now, using the definitions of p and q , we have

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

That completes the proof. ■

Now, adding the incomplete data log-likelihood to both sides of the inequality in Lemma 1, we have

$$\begin{aligned}
E_p[\ln P(\mathcal{Z}|(\mathcal{X}, \Theta))] + \ln P(\mathcal{X}|\Theta) &\leq E_q[\ln P(\mathcal{Z}|(\mathcal{X}, \Theta))] + \ln P(\mathcal{X}|\Theta) \\
\Rightarrow E_p[\ln(P(\mathcal{Z}|(\mathcal{X}, \Theta))P(\mathcal{X}|\Theta))] &\leq E_q[\ln(P(\mathcal{Z}|(\mathcal{X}, \Theta))P(\mathcal{X}|\Theta))] \\
\Rightarrow E_p[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] &\leq E_q[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)].
\end{aligned} \tag{3.18}$$

From, (3.17) and (3.18), we have

$$E_{\mathcal{Z}|\mathcal{X},\Theta\sim p}[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] \leq E_{\mathcal{Z}|\mathcal{X},\Theta\sim q}[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] \leq \ln P(\mathcal{X}|\Theta). \quad (3.19)$$

Thus, the expectation over the hard-assignment distribution lies in between the incomplete data likelihood and the expectation over p of complete data likelihood 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. Then,

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

Hence, the choice of q as in (3.13) 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 in the previous section. These two algorithms are based on soft- and hard-assignment schemes [KMN97], 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. 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 vMFs 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.³

The **hard-moVMF** algorithm, presented in Algorithm 2 estimates the parameters of the mixture model by a hard assignment, or, *winner take all* strategy. In other words, we do the assignment of the points based on a derived posterior distribution given by (3.13). 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 only 0/1 values. 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 spkmeans

We briefly revisit the **spkmeans** algorithm [DM01] that has been shown to perform quite well for real life text clustering tasks [DM01, DFG01, BG02, BBM02]. We look at **spkmeans** in the

³Note that we can obtain a hybrid algorithm by combining the soft- and hard- approaches. In such an algorithm one would first perform some iterations that do soft assignments and later perform hard assignments. Further justification of this idea appears in section 8.

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 distributionsInitialize all $\alpha_h, \boldsymbol{\mu}_h, \kappa_h, h = 1, \dots, k$ **repeat**

{The E (Expectation) step of EM}

for $i = 1$ to n **do****for** $h = 1$ to k **do**

$$p(\mathbf{x}_i|\theta_h) \leftarrow c_d(\kappa_h)e^{\kappa_h\boldsymbol{\mu}_h^T\mathbf{x}_i}$$

$$p(h|\mathbf{x}_i, \Theta) \leftarrow \frac{\alpha_h p(\mathbf{x}_i|\theta_h)}{\sum_{h'=1}^k \alpha_{h'} p(\mathbf{x}_i|\theta_{h'})}$$

end for**end for**

{The M (Maximization) step of EM}

for $h = 1$ to k **do**

$$\alpha_h \leftarrow \frac{1}{n} \sum_{i=1}^n p(h|\mathbf{x}_i, \Theta)$$

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

$$\kappa_h \leftarrow A_d^{-1} \left(\frac{\|\sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta)\|}{\sum_{i=1}^n p(h|\mathbf{x}_i, \Theta)} \right)$$

end for**until** *convergence criteria* are met

light of the developments in sections 3 and 4. Algorithm 3 outlines the `spkmeans` procedure. We observe that the `spkmeans` algorithm can be looked upon as a special case of `soft-moVMF` as well as of `hard-moVMF`, under certain restrictive assumptions on the generative model. More precisely, 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 special case of `soft-moVMF`, we further assume that all the components have (equal) infinite concentration parameters, i.e., $\kappa_h = \kappa \rightarrow \infty, \forall h$. 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 $h^* = \operatorname{argmax}_h \mathbf{x}_i^T \boldsymbol{\mu}_h$, since

$$p(h^*|\mathbf{x}_i, \Theta) = \lim_{\kappa \rightarrow \infty} \frac{e^{\kappa \mathbf{x}_i^T \boldsymbol{\mu}_{h^*}}}{\sum_{h'=1}^k e^{\kappa \mathbf{x}_i^T \boldsymbol{\mu}_{h'}}} \rightarrow 1, \quad (4.1)$$

and $p(h|\mathbf{x}_i, \Theta) \rightarrow 0, \forall h \neq h^*$ (assuming no ties).

In order 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 is not needed 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, $\mathcal{X}_h = \{\mathbf{x} : \mathbf{x} \in \mathcal{X}, h = \operatorname{argmax}_{h'} \kappa_{h'} \mathbf{x}^T \boldsymbol{\mu}_{h'}\}$.

Algorithm 2 hard-moVMF

Input: Set \mathcal{X} of data points on \mathbb{S}^{d-1}

Output: A disjoint k -partitioning of \mathcal{X}

Initialize all $\alpha_h, \boldsymbol{\mu}_h, \kappa_h, h = 1, \dots, k$

repeat

{The E (Expectation) step of EM}

for $i = 1$ to n **do**

for $h = 1$ to k **do**

$$p(\mathbf{x}_i|\theta_h) \leftarrow c_d(\kappa_h)e^{\kappa_h\boldsymbol{\mu}_h^T\mathbf{x}_i}$$

$$q(h|\mathbf{x}_i, \Theta) \leftarrow \begin{cases} 1, & \text{if } h = \arg \max_{h'} \alpha_{h'} p(\mathbf{x}_i|\theta_{h'}) \\ 0, & \text{otherwise.} \end{cases}$$

end for

end for

{The M (Maximization) step of EM}

for $h = 1$ to k **do**

$$\alpha_h \leftarrow \frac{1}{n} \sum_{i=1}^n q(h|\mathbf{x}_i, \Theta)$$

$$\boldsymbol{\mu}_h \leftarrow \frac{\sum_{i=1}^n \mathbf{x}_i q(h|\mathbf{x}_i, \Theta)}{\|\sum_{i=1}^n \mathbf{x}_i q(h|\mathbf{x}_i, \Theta)\|}$$

$$\kappa_h \leftarrow A_d^{-1} \left(\frac{\|\sum_{i=1}^n \mathbf{x}_i q(h|\mathbf{x}_i, \Theta)\|}{\sum_{i=1}^n q(h|\mathbf{x}_i, \Theta)} \right)$$

end for

until *convergence criteria* are met

Algorithm 3 spkmeans [DM01]

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 $\boldsymbol{\mu}_h, h = 1, \dots, k$

repeat

{The E (Expectation) step of EM}

Set $\mathcal{X}_h \leftarrow \phi, h = 1, \dots, k$

for $i = 1$ to N **do**

$$\mathcal{X}_h \leftarrow \mathcal{X}_h \cup \{\mathbf{x}_i\} \text{ where } h = \arg \max_{h'} \mathbf{x}_i^T \boldsymbol{\mu}_{h'}$$

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 criteria* are met

4.2 Some Implementation issues

We infer from (2.14) that to calculate the m.l.e. for κ , we have to compute $A_d^{-1}(\bar{R})$. Since $A_d(\kappa)$ is a function that involves the ratio of Bessel Functions it is not possible to obtain a closed form functional inverse. Instead we resort to numerical or asymptotic methods. Using Appendix B we can show that for large κ , the m.l.e. $\hat{\kappa}$ is approximately $(d-1)/(2(1-\bar{R}))$. Similarly for small κ we obtain the approximation, $\tilde{\kappa} \approx d\bar{R}$. These estimates for $\tilde{\kappa}$ are not that good in practice, especially for high dimensional data. A better estimate (see Appendix B) is given by

$$\tilde{\kappa} = \frac{\bar{R}d - \bar{R}^3}{1 - \bar{R}^2}, \quad (4.2)$$

and this is the estimate that we use in our implementation.

5 Datasets

In this section we describe the datasets that we used for our experiments. We used data drawn from five sources: Simulation, Classic3, Yahoo News, CMU 20 Newsgroup and Yeast Gene Expressions.

- **Simulated Datasets:** We simulated mixtures of vMFs for testing the “quality” and correctness of our algorithms that were designed to work for moVMFs. We used a slight modification of the algorithm given in [Woo94] to generate a set of data points following a given vMF distribution. The algorithm is shown in Table 9 and was implemented in MATLAB. This algorithm allows us to sample data from a specified vMF distribution. We use this algorithm as a subroutine when simulating a Mixture of vMFs (see Table 8). Let the tuple (N, d, k) denote the number of sample points, the dimensionality of a sample point and the number of clusters respectively. We present two main simulated datasets in this article. The first dataset **small-mix** is just 2-dimensional and is used to illustrate soft-clustering. The second dataset **big-mix** is a high-dimensional dataset that could serve as a model for real world text datasets.
 1. **small-mix:** This dataset had $(N, d, k) = (50, 2, 2)$. The mean direction of each component was set to some random vector. For each component κ was set to 4. We present detailed results for this particular dataset later in this section.
 2. **big-mix:** This dataset had $(N, d, k) = (5000, 1000, 4)$. The mean direction of each component was set to some random vector, and κ for each component was also set to a random number of $O(d)$. The mixing weights for each component were: (.251, .238, .252, .259).
- **Classic3 Datasets:** Classic3 is a well known collection of documents. Classic3 contains documents from three well-separated sources, so the clusters that are formed should be quite disjoint.
 1. **Classic3:** This dataset contains 3893 files, among which 1400 Cranfield documents are from aeronautical system papers, 1033 Medline documents are from medical journals, and 1460 Cisi documents are from information retrieval papers. The toolkit MC [DFG01] was used for creating the high-dimensional vector space model for the text documents and a total of 4666 words were used. Thus, each document, after normalization, is represented as a unit vector in a 4666 dimensional space. This is a relatively simple dataset in the sense that the documents in the 3 clusters are on completely different topics.

2. **Classic300:** Classic300 is a subset of 300 documents that we created from the original Classic3 dataset. This dataset has 100 documents from each of the three categories in Classic3. The dimensionality of the data was 5471.
 3. **Classic400:** Classic400 is a subset of 400 documents that we created from the original Classic3 dataset. This dataset has 100 documents each from Medline and Cisi categories. The remaining 200 documents are from the Cran category. This dataset was specifically designed to create unbalanced clusters in an otherwise easily separable and balanced dataset. The dimensionality of the data was 6205.
- **Yahoo News dataset** K-series dataset is a collection of 2340 Yahoo news articles from 20 different categories. The underlying clusters in this dataset are highly skewed in terms of the number of documents per cluster, with sizes ranging from 9 to 494. The skewness presents some more challenges for clustering algorithms.
 - **CMU Newsgroup datasets** The CMU Newsgroup dataset is a well known collection of documents. The original distribution is one that is mentioned in item 1 below. We considered not only the original dataset but some of its subsets.
 1. **The News20 dataset** is a collection of 19,997 messages, collected from 20 different USENET newsgroups. A 1000 messages from 19 newsgroups, and 997 from another were chosen at random and partitioned by newsgroup name. The headers for each of the messages were removed so that they do not bias the results. Using the toolkit MC, the high-dimensional model had a total of 25924 words. This is a typical text dataset that one may encounter in real life - it very high-dimensional, sparse and there is significant overlap between its clusters. In fact, some cross-posted articles appear multiple times in the dataset – once under every group in which they were posted. Another feature of this dataset is that the natural classes are perfectly balanced, i.e., the number of points in every class is the same.
 2. **Small CMU 20** is a collection of 2000 messages selected from the original News20 dataset. We selected 100 messages from each category in the original dataset. Hence this dataset has balanced classes (though there may be overlap). The dimensionality of the data was 13406.
 3. **Same 100/1000** is a collection of 100/1000 messages from 3 very similar newsgroups: comp.graphics, comp.os.ms-windows, comp.windows.x.
 4. **Similar 100/1000** is a collection of 100/1000 messages from 3 somewhat similar newsgroups: talk.politics.guns, talk.politics.mideast, talk.politics.misc.
 5. **Different 100/1000** is a collection of 100/1000 messages from 3 very different newsgroups: alt.atheism, rec.sport.baseball, sci.space.
 - **Yeast Gene Expression dataset:** Gene expression data was selected to offer a different clustering domain. This domain brings with it the associated troubles of cluster validation because of the unavailability of true labels. Thus this dataset is a truer representative of a typical clustering scenario.

Gene expression data is presented as a matrix of genes (rows) by expression values (columns). The expression vectors are constructed using DNA microarray experiments. We used a subset of the Rosetta Inpharmatics yeast gene expression set [HMJ⁺00]. The original dataset consists of 300 experiments measuring expression of 6,048 yeast genes. Out of these we selected a

subset of 996 genes for clustering. For each of the 996 genes the 300-element expression vector was normalized to have unit Euclidean (L_2) norm.

6 Experimental Results

In this section we compare the following four clustering algorithms on the datasets described in Section 5:

1. Spherical K-Means [DM01]—`spkmeans`,
2. Frequency Sensitive K-Means [BG02]—`fskmeans`,
3. moVMF based clustering using hard assignments (Section 3)—`hard-moVMF`,
4. moVMF based clustering using soft assignments (Section 3)—`soft-moVMF`.

Except the gene expression dataset, performance of the algorithms on all the datasets has been analyzed using *mutual information* between the cluster and class labels. Our algorithms were implemented in both C++ and MATLAB. Most of the results presented in this section are based on our C++ implementation. In all the experiments, all the algorithms were initialized with identical starting partitions to ensure a fair comparison. For all the artificial and text datasets, the correct underlying class labels of the data points are known. For a given clustering of the data, we construct the class-by-cluster confusion matrix and normalize it by the total number of documents under consideration. Now, a data point having class label c and cluster label h is an indicator of the joint occurrence of (c, h) . For a large number of documents, the normalized confusion matrix approaches the expectation of this indicator by the law of large numbers. But since this expectation is exactly the probability of the joint event, in the limit of large data, the normalized confusion matrix gives the joint distribution of the class and cluster labels. The first measure we use for performance evaluation is the mutual information (MI) of this joint distribution. The MI gives the amount of statistical similarity between the class and cluster labels [CT91]. 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 the mutual information is given by $I(X; Y) = E_{X, Y} [\ln \frac{p(X, Y)}{p(X)p(Y)}]$. Note that we have experimented with various normalizations of the MI (see, e.g., [SGM00, SG02]), but these normalizations hardly make any difference in terms of analysis and interpretability of the results. The Appendix shows all the plots with normalized values of mutual information.

6.1 Simulated data sets

This section presents the results obtained by running the experiments on data that was simulated to have been drawn from a mixture of vMF distributions. For a description of the data see Section 5.

6.1.1 Dataset small-mix

The **small-mix** dataset is a collection of fifty, two-dimensional points, drawn from a mixture of two vMF distributions. Figure 1 (a) shows a plot of the points. From the plot we observe that there are two clusters of points. Most points belong to either one cluster or the other. Some of the points seem to have mixed membership to each cluster. As we shall soon see, the soft-clustering algorithm identifies these points and assigns them fractionally to either cluster.

The clustering produced by our soft cluster assignment algorithm is shown in Figure 1 (b).