K-means++ for Mixtures of von Mises-Fisher Distributions

Mohamadreza Mash'al
School of ECE, College of Engineering
University of Tehran
Tehran, Iran
Email: mrmashal@ut.ac.ir

Reshad Hosseini
School of ECE, College of Engineering
University of Tehran
Tehran, Iran
Email: reshad.hosseini@ut.ac.ir

***Abstract*—Von Mises-Fisher (vMF) Distribution is one of the most commonly used distributions for fitting directional data. Mixtures of vMF (MovMF) distributions have been used successfully in many applications. One of the important problems in mixture models is the problem of local minima of the objective function. Therefore, approaches to avoid local minima problem is essential in improving the performance. Recently, an algorithm called k-means++ was introduced in the literature and used successfully for finding initial parameters for mixtures of Gaussian (MoG) distributions. In this paper, we adopt this algorithm for finding good initializations for MovMF distributions. We show that MovMF distribution will lead to the same cost function as MoGs and therefore similar guarantee as the case of MoG distributions will also hold here. We also demonstrate the performance of the method on some real datasets.**

I. Introduction

Von Mises Fisher (vMF) Distribution is one of the most commonly used distributions for fitting directional data. This distribution has been used extensively in many fields like biomedical imaging [1], speech processing [2], computer vision [3,4] and text mining [5]. In order to have a more flexible density model, one can use mixtures of von Mises-Fisher (MovMF) distributions. It has been demonstrated that MovMF is indeed improves the modeling power of vMF in application [6].

In addition to offering flexible density modeling capabilities, mixture models are extensively used in clustering [1]–[5,7,8]. Clustering methods segment the data into different components such that the data within each cluster have some similarity so that one would put them into one category. If we model the data within each cluster using a probability distribution, the whole data would have a mixture density form. In order to cluster each observation, we just need to compute the probabilities that this observation goes to different clusters and choose the cluster that maximizes this probability. Therefore, upon having a mixture density model fitted to the data, we have a principled way of clustering. Many clustering algorithm like the famous k-means method can be reformulated as mixture density modeling [9,10].

The performance of data clustering is dependent not only on how we represent the data (i.e. how we extract features from data) but also on which parametric density model we used to model the data within each cluster. One of the commonly used density models is the Gaussian distribution. Therefore, MoG densities have been used extensively in clustering. In this model, the assumption is that the data is scattered around a point (cluster center or equivalently the mean of the Gaussian density) in Euclidean space. However, for the cases where the extracted feature vector is directional [11,12] or when it is projected on the unit hypersphere to make it directional, then Gaussian density is not often the best density for fitting such data. In practice it has been shown that for such data, using a directional density model like vMF leads to improved performance of the clustering procedure [7].

Fitting mixture density to the data or equivalently clustering the data is known to be a hard optimization problem. If we are to minimize the main clustering objective, that is partitioning the data into different components such that the total distance is minimum, it is known that the problem is NP-hard even for special case of k-means objective [13]. Even if we relax the objective and change it to maximizing the log-likelihood of the mixture density, the problem still remains difficult. Since the objective function of mixture density has local minima, it is important to use procedures to avoid this problem. There are many approaches for solving local minima problem, look for example into [14]–[16] and references therein. K-means++ is one of the most effective initialization methods that has an interesting theoretical guarantee [17] and has been proved useful in applications [18,19].

In this paper, we show that the same improvement also holds for k-means++ used for initializing MovMF. This is done by showing that the cost function of MovMF can be reformulated to be expressed as k-means cost function. The difference is that the optimization problem of MovMF has a unit-norm constraint, but k-means++ interestingly satisfies this constraint and therefore the bound of k-means++ algorithm expressed in [17] holds also for MovMF. We examine through experiments on text data that the theoretical guarantee is validated by experimental results.

The rest of the paper is organized as follows. In Section II, we review the material regarding vMF and MovMF distributions. We also examine the hard-clustered version of MovMF in this section. Section III explains the derivation of MovMF subject to the constraint that the concentration parameters of different mixtures are equal. Next, we present the k-means++ algorithm for MovMF and show that the same theoretical guarantee as of normal k-means objective also holds here. In Section V, we demonstrate the performance of k-means++ initialization and compare it to random initialization. We finish our paper by a short conclusion.

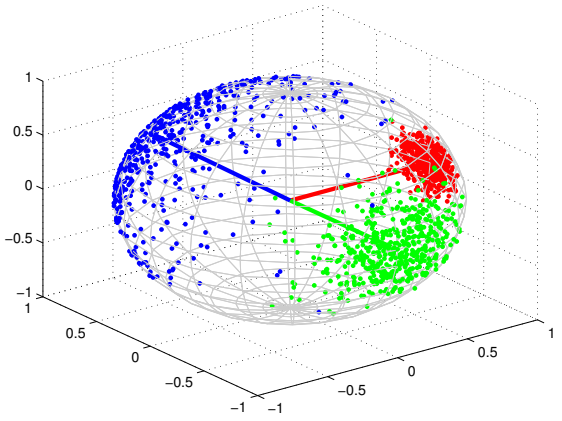

Fig. 1. Scatter plot of data sampled from 3-dimensional vMF distributions with different concentration parameters (blue: 4, green:16 and red: 64)

II. Background

A. MovMF

A d-dimensional random vector $\boldsymbol{x}$ on unit hypersphere is said to be distributed according to vMF distribution, if its density is given by:

$$p(\boldsymbol{x}; \kappa, \boldsymbol{\mu}) = c_d(\kappa) \exp(\kappa \boldsymbol{\mu}^T \boldsymbol{x}),$$

where $\boldsymbol{\mu}$ is a unit norm vector called mean direction and $\kappa \geq 0$ is the concentration parameter. The normalization constant c_d is given by:

$$c_d(\kappa) = \frac{\kappa^{d/2}}{(2\pi)^{d/2} \boldsymbol{I}_{d/2-1}(\kappa)},$$

where $I_{d/2-1}(.)$ is the modified Bessel function of the first kind and order $d/2-1$. Fig. 1 demonstrates the effect of the concentration parameter on the density.

Consider $\{\boldsymbol{x}\}_{i=1}^n$ to be the set of random samples, the log-likelihood of the data is given by:

$$l(\kappa, \boldsymbol{\mu}) = \sum_{i=1}^{n} \log p(\boldsymbol{x}_i; \kappa, \boldsymbol{\mu}).$$

Maximum likelihood solution for parameter $\boldsymbol{\mu}$ is calculated by:

$$\boldsymbol{\mu} = \frac{\sum_{i=1}^n \boldsymbol{x}_i}{\|\sum_{i=1}^n \boldsymbol{x}_i\|}$$

and κ is the solution of the following equation:

$$\frac{\boldsymbol{I}_{d/2}(\kappa)}{\boldsymbol{I}_{d/2-1}(\kappa)} = \underbrace{\frac{\|\sum_{i=1}^n \boldsymbol{x}_i\|}{n}}_{\bar{r}}$$

[8] showed that the following equation gives a pretty good approximation to the true κ solving the above equation:

$$\kappa \approx \frac{\bar{r}d - \bar{r}^3}{1 - \bar{r}^2} \tag{1}$$

Given $\{p_k(\boldsymbol{x}; \kappa, \boldsymbol{\mu})\}_{k=1}^K$ to be K different vMF densities, a mixture of these K densities is given by:

$$p(\boldsymbol{x}; \{\kappa_k, \boldsymbol{\mu}_k, \pi_k\}_{k=1}^K) = \sum_{k=1}^{K} \pi_k p(\boldsymbol{x}; \kappa_k, \boldsymbol{\mu}_k),$$

where $\pi_k \geq 0$ are component weights and sum to one. A common approach for finding ML solution for mixture densities is the *Expectation-Maximization* (EM) algorithm. This algorithm iterates between E-step and M-step till convergence. In E-step, the following weights are calculated:

$$w_{ki} = \frac{\pi_k p(\boldsymbol{x}_i; \kappa_k, \boldsymbol{\mu}_k)}{p(\boldsymbol{x}_i; \{\kappa_k, \boldsymbol{\mu}_k, \pi_k\}_{k=1}^K)}$$

In M-step, the parameters $\{\kappa_k, \boldsymbol{\mu}_k\}$ are updated by maximizing the following weighting log-likelihoods:

$$\{\kappa_k, \boldsymbol{\mu}_k\} \leftarrow \arg\max_{\kappa_k, \boldsymbol{\mu}_k} \sum_{i=1}^{n} w_{ki} \log p(\boldsymbol{x}_i; \kappa_k, \boldsymbol{\mu}_k) \tag{2}$$

and component weights π_k are updated by the following equation:

$$\pi_k \leftarrow \frac{\sum_{i=1}^n w_{ki}}{n}.$$

Maximizing (2) is similar to the ML solution in the case of a single vMF distribution. Its solution is given by [8]:

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^n w_{ki} \boldsymbol{x}_i}{\|\sum_{i=1}^n w_{ki} \boldsymbol{x}_i\|},$$
$$\kappa_k \approx \frac{\bar{r}_k d - \bar{r}_k^3}{1 - \bar{r}_k^2},$$

where

$$\bar{r}_k = \frac{\|\sum_{i=1}^n w_{ki} \boldsymbol{x}_i\|}{\sum_{i=1}^n w_{ki}}.$$

B. Hard-clustered MovMF

When using EM algorithm for training a mixture model, we can interpret the weights computed in the E-step as the probability that an observation goes to a particular cluster. Mathematically speaking, consider $l = 1, \ldots, K$ be the label determining which component to be chosen and $\pi_k = p(l = k)$ as the prior probability of different cluster labels, then the weights are actually equal to the posterior probabilities of data given cluster labels $w_{ki} = p(l = k|\boldsymbol{x}_i)$.

If we assign each observation to the cluster that has the maximum posterior probability and then update the parameters, we obtain a method that maximizes the following hard-clustering objective [10]:

$$\sum_{k=1}^{K} \sum_{i \in \mathcal{I}_k} \log \Big[\pi_k p(\boldsymbol{x}_i; \kappa_k, \boldsymbol{\mu}_k) \Big],$$

where $\mathcal{I}_k = \{i : l_i = K\}$ is a set containing the indices of data that go to cluster k. Maximizing this objective is done by iterating between data assignment and updating the component parameters.

Assignment step:

$$l_i = \arg\max_{k \in \{1, \ldots, K\}} p(l = k|\boldsymbol{x}_i)$$

Update step:

$$\{\kappa_k, \boldsymbol{\mu}_k\} \leftarrow \arg\max_{\kappa_k, \boldsymbol{\mu}_k} \sum_{i \in \mathcal{I}_k} \log p(\boldsymbol{x}_i; \kappa_k, \boldsymbol{\mu}_k)$$

$$\pi_k \leftarrow \frac{|\mathcal{I}_k|}{n},$$

where $|\mathcal{I}_k|$ is the cardinality of the set $\mathcal{I}_k$.

III. MovMF with equal concentration parameters

Similar to [7], we observed that forcing concentration parameters of different components to be equal ($\kappa_k = \kappa$) improves the clustering performance on the dataset we are investigating. This is because of the regularizing effect of this action, especially important for the cases where the size of the problem is large but the number of observations is small. Such is the case in many fields like data-mining where the size of features extracted from a text is very large.

When using the EM algorithm for training an MovMF in this case, the E-step and M-step for updating the π_k and $\boldsymbol{\mu}_k$ parameters stays the same. The only part that needs modification is the M-step for updating κ. To this end, we need to explain briefly how the EM algorithm is derived for fitting mixture models. In the EM algorithm, we compute a lower bound to the log-likelihood objective function and then we maximize this lower bound. E-step computes some weights in the expression for the lower bound [20]:

$$\sum_{k=1}^{K}\sum_{i=1}^{n} w_{ki} \log p(\boldsymbol{x}_i; \kappa_k, \boldsymbol{\mu}_k) \leq \log p(\boldsymbol{x}; \{\kappa_k, \boldsymbol{\mu}_k, \pi_k\}_{k=1}^{K}) + c,$$

where c is a constant. M-step maximizes this lower bound. It is clear that when κ_k are different, we obtain the M-step explained in the previous section. When κ_k are equal, we have the following maximization problem for the M-step:

$$\kappa = \arg\max_{\kappa} \sum_{k=1}^{K}\sum_{i=1}^{n} w_{ki} \log p(\boldsymbol{x}_i; \kappa, \boldsymbol{\mu}_k).$$

Maximizing the objective function in the equation above, we obtain the following update rule similar to (1) for κ, wherein $\bar{r}$ is computed by:

$$\bar{r}_k = \frac{\|\sum_{k=1}^{K}\sum_{i=1}^{n} w_{ki}\boldsymbol{x}_i\|}{n}.$$

IV. K-means++ for MovMF

K-means++ is a probabilistic initialization method that finds the data points that are far apart in the dataset but at the same time it avoids choosing outliers. In the case of MoG distributions, selected data points are used as the initialized mean of the Gaussian components. For the case of MovMF distributions, we would do the same, that is we select some data points that are distant and use them to initialize the parameter $\boldsymbol{\mu}$ for different components.

In order to apply k-mean++ algorithm for the case of MovMF distributions, we need to define a distance function between different points on the unit hypersphere. Consider $\boldsymbol{x}$ and $\boldsymbol{y}$ to be two points on the unit hypersphere, we define their distance to be:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{2 - 2\boldsymbol{x}^T\boldsymbol{y}}.$$

The choice of this distance function will be clear later in this section. K-means++ will randomly choose a point from the data as the first cluster mean. The second point is chosen at random but with a probability proportional to the squared distance from the first point. For the kth point, the probability is proportional to the minimum squared distance from the previously chosen $k-1$ points. K-means++ for MovMF distributions is summarized in Algorithm 1.

Algorithm 1 Pseudo-code for k-means++ algorithm

Input: n data points on the unit hypersphere $\{\boldsymbol{x}_i\}_{i=1}^{n}$; K, the number of components
Set $\boldsymbol{\mu}_1 \leftarrow \boldsymbol{x}_i$, wherein $i \in \{1, ..., n\}$ is chosen randomly
Set $c_i \leftarrow \boldsymbol{\mu}_1^T\boldsymbol{x}_i$, $i = 1, ..., n$
for $k = 2$ to K **do**
  Set $d_i \leftarrow \sum_{j=1}^{k}(2 - 2c_j)$, $i = 1, ..., n$
  Set $\boldsymbol{\mu}_k \leftarrow \boldsymbol{x}_i$, wherein $i \in \{1, ..., n\}$ is chosen by the probability $p_i = d_i / \sum_{i=1}^{n} d_i$
  Set $c_i = \max(c_i, \boldsymbol{\mu}_k^T\boldsymbol{x}_i)$, $i = 1, ..., n$
end for

For the case of MoG distributions with the Euclidean distance function $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$, it was demonstrated in [17] that the k-means++ algorithm mentioned above, gives interesting expected bound on the objective function, while no such bound exists for random initialization. The objective function of k-means is equal to the hard-clustered MoGs, wherein we assume covariances and mixing probabilities to be equal.

In the following, we will show that the objective function of the hard-clustered version of MovMF with equal concentration parameters and equal mixing weight (which is termed spherical MovMF [8]) is equal to that of k-means algorithm. For spherical MovMF, we are maximizing the following objective function:

$$\sum_{k=1}^{K}\sum_{i\in\mathcal{I}_k} \boldsymbol{\mu}_k^T\boldsymbol{x}_i$$

subject to the constraints that all $\boldsymbol{\mu}_k$ would lie on the unit hypersphere. We can replace the operation of maximizing the aforementioned objective function with minimizing the following cost function:

$$\phi = \sum_{k=1}^{K}\sum_{i\in\mathcal{I}_k} \|\boldsymbol{\mu}_k - \boldsymbol{x}_i\|^2 \tag{3}$$

This is true because $\|\boldsymbol{\mu}_k\| = 1$ and $\|\boldsymbol{x}_i\| = 1$ and therefore $\|\boldsymbol{\mu}_k - \boldsymbol{x}_i\|^2 = \|\boldsymbol{\mu}_k\|^2 + \|\boldsymbol{x}_i\|^2 - 2\boldsymbol{\mu}_k^T\boldsymbol{x}_i = 2 - 2\boldsymbol{\mu}_k^T\boldsymbol{x}_i$.

The usual k-means algorithm tries to minimize the cost in (3) but without any constraint on $\boldsymbol{\mu}_k$ and therefore the attainable cost is smaller than that of spherical MovMF. As stated beforehand, k-means++ provides expected bound to the objective function of k-means. Therefore a similar bound also holds in k-means++ for spherical MovMF. We summarize this discussion into the following theorem (the derivation for the case of normal k-means can be seen in [17]):

Theorem 1. *Consider ϕ_{OPT} to be the minimum cost and ϕ to be the cost obtained by the k-means++ initialization, then the following bound exists for the expected cost:*

$$E[\phi] \leq 8(\ln K + 2)\phi_{OPT}$$

TABLE I. DIFFERENT TEXT DATASETS

Dataset	#Instances	#Features	#Categories
NEWS20	18774	61188	20
REUTERS	8293	18933	65
CNAE	1079	857	9
TDT2	10212	36771	96

V. EXPERIMENTAL RESULTS

We evaluate the performance of k-means++ algorithm on several standard text datasets (NEWS20[1], CNAE[2], REUTERS and TDT2[3]). Each data point is a document and there is an assigned label for each document showing the category that it belongs to. The data points are vectors containing word-counts of the underlying document. A summary of these datasets are given in Table I.

Clustering performance observed when using raw-data without any preprocessing is low. A very common and strong representation of the data commonly used in data-mining community is *term frequencyinverse document frequency* (Tf-Idf) [21]. Using this representation makes the data points of different categories more separate and significantly improves the performance. The Tf-Idf of word t in document d from a document corpus D is defined by the following relation [21]:

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D),$$

where term frequency $tf(t,d)$ and inverse document frequency $idf(t,D)$ are computed by:

$$tf(t,d) = 0.5 + \frac{0.5f(t,d)}{\max\{f(w,d) : w \in d\}}$$
$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}| + 1},$$

wherein $f(t,d)$ is the count of word t in document d, $|\{d \in D : t \in d\}|$ is the number of documents in which the word t appears and N is the total number of documents.

We are comparing k-means++ method against a random initialization which is often used in mixture modeling literature [22]. This random initialization assigns each observation randomly to one of K different clusters and then the parameters of the model are estimated using the assigned data. The metric used in the comparisons is the well-known *normalized multi-information* (NMI). Consider $\Omega = \{\omega_1, \ldots, \omega_K\}$ to be the set of found clusters and $\mathcal{C} = \{c_1, \ldots, c_J\}$ be the set of true classes, then the NMI between these two sets is defined by the following equation [21]:

$$NMI(\Omega, \mathcal{C}) = \frac{I(\Omega;\mathcal{C})}{[H(\Omega) + H(\mathcal{C})]/2}, \quad (4)$$

where multi-information I and entropy H is given by:

$$I(\Omega;\mathcal{C}) = \sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} \log \frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|}$$
$$H(\Omega) = -\log_k \frac{|\omega_k|}{N} \log \frac{|\omega_k|}{N},$$

[1]http://people.csail.mit.edu/jrennie/20Newsgroups/
[2]http://archive.ics.uci.edu/ml/datasets/CNAE-9
[3]http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html

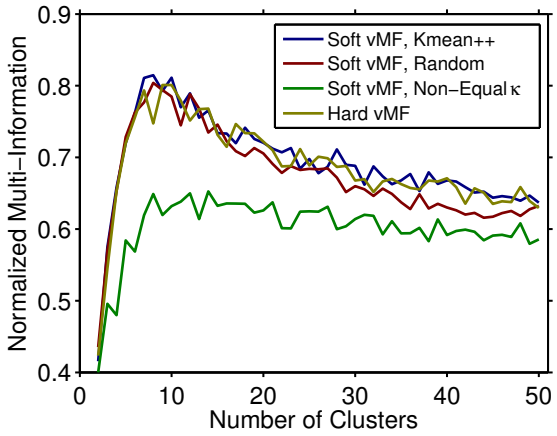

Fig. 2. Performance comparison of different clustering algorithms on CNAE dataset. Blue plot shows the result of soft vMF with kmeans++ initialization, red plot show the same results but for random initialization, green plot shows the result of soft vMF when concentration parameters are not equal and brown plot shows the result of hard vMF.

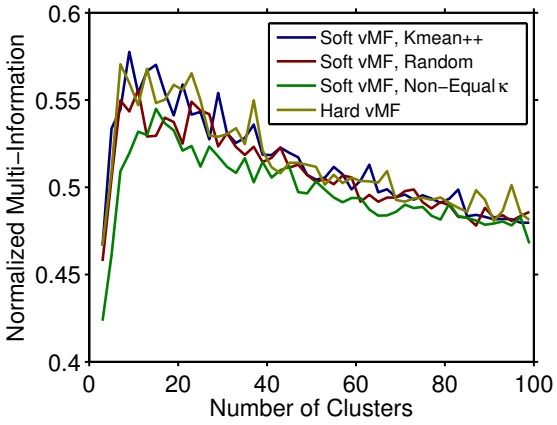

Fig. 3. Performance comparison of different clustering algorithms on REUTERS dataset.

where $|\omega_k \cap c_j|$ is the number of documents that goes to cluster k and their true class is j.

The results of NMI for different datasets are shown in figures 2-5. In these figures, blue and red curves correspond to k-means++ and random initialization, respectively. These two curves are the results for the soft version of MovMF distribution and with equal concentration parameters. We observe that for the investigated datasets, making the concentration parameters equal always helps in improving the performance. However, for the sake of comparison we show the result of fitting MovMFs with non-equal concentration parameters which are shown by green curves in the figures. A different regularization strategy introduced in [8] is to use an upper bound on κ_k. We observed that making κ_k equal leads to significantly better results. Therefore, we are not showing the regularization method of [8]. In addition to these curves, we are showing the result of the hard-clustered version of MovMF plotted as brown curves.

Apparently for two of the four datasets (CNAE and REUTERS), k-means++ initialization improves the performance over random initialization. For NEWS20 dataset, both

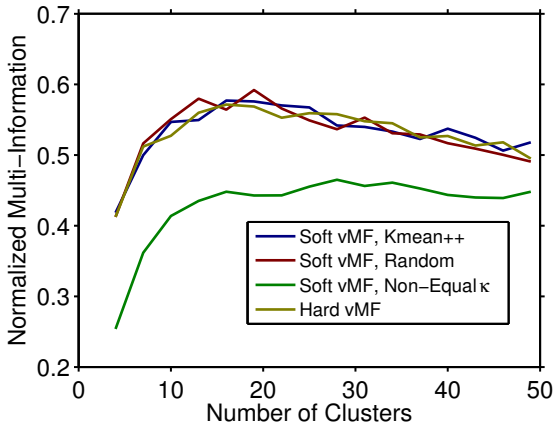

Fig. 4. Performance comparison of different clustering algorithms on NEWS20 dataset.

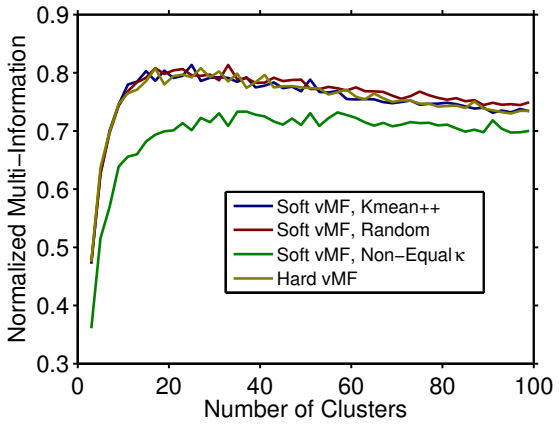

Fig. 5. Performance comparison of different clustering algorithms on TDT2 dataset.

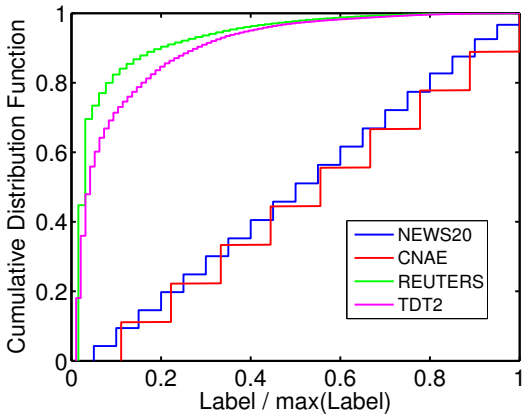

Fig. 6. Cumulative distribution function of percentage of documents for different labels.

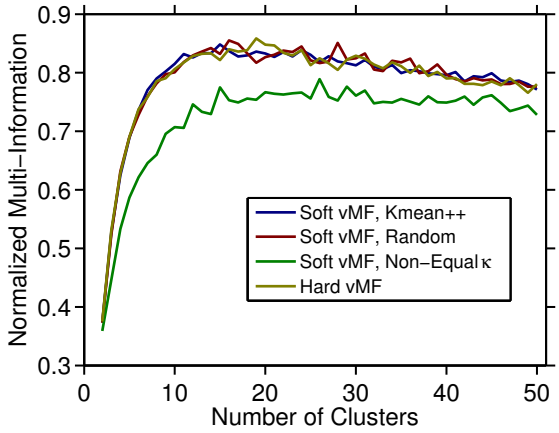

Fig. 7. Performance comparison of different clustering algorithms on modified TDT2 dataset.

initialization lead to similar results. Only for TDT2 dataset, random initialization has slightly better performance than the k-means++ initialization. For each number of clusters, 20 different initialization for both random and k-means++ were tested and the best NMI numbers are plotted in the figures. Although randomness exists in both initialization methods, since the behavior of initialization methods remain unchanged by the number of clusters for each dataset, we can infer that the efficacy of one initialization method over the other one is systematic and not because of chance.

In NEWS20 and CNAE datasets, the peak of NMI curve is about the true number of labels in the dataset corpus. While in REUTERS and TDT2, this behavior is not seen. We attribute this behavior to the unbalanced distribution of the number of datapoints in different classes. Fig. 6 shows this distribution for different datasets. As it is clear from this figure, only few components get the most of datapoints in REUTERS and TDT2 datasets. The objective function that we are optimizing for (i.e. log-likelihood) does not take into account the number of data within each cluster and only overall performance over all clusters is important for the objective. However, NMI measure looks for having good clustering results for each individual cluster. Therefore the observed behavior is predictable.

In order to further investigate the effect of distribution of datapoints, we create a new dataset by choosing only datapoints belonging to first 30 classes with highest number of data in TDT2 dataset. For this new dataset, we see that the performance is improved (see Fig. 7).

In all the previous plots, we observe that hard-clusters MovMF has pretty much similar performance as soft-clustered MovMF. The authors in [8] ran experiments for smaller dataset sizes by reducing the size of original datasets and observed that soft-clustered version works better than hard-clustered version of MovMF. The authors in [8] created a smaller version of NEWS20 dataset by selecting 100 documents randomly for each label. We created small-NEWS20 dataset by similar procedure and run clustering methods on it and the result is shown in Fig. 8. Unlike [8], we do not see performance decrease for hard-clustered MovMF. We attribute the reason of this difference to the kmeans++ initialization method exploited in the current paper.

As it can be seen from Fig. 8, random initialization has significantly lower performance that that of kmeans++ initialization. We observed the same behavior on other datasets. Therefore, we are postulating that the performance of random initialization is more dependent on the number of datapoints.

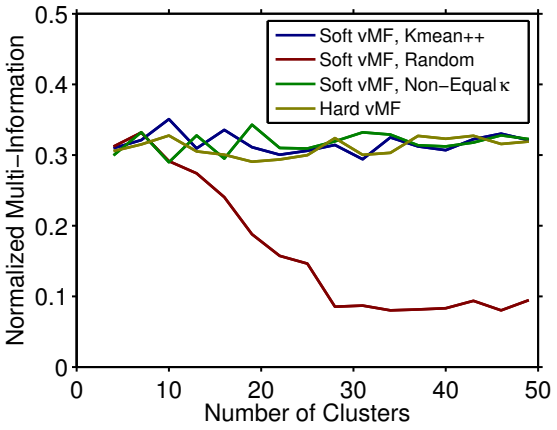

Fig. 8. Performance comparison of different clustering algorithms on Small-NEWS20 dataset.

VI. Conclusion

In this paper, we adopted k-means++ algorithm for finding good initial parameters for MovMF distributions. We showed that this algorithm has the same theoretical bound on the performance as the case of k-means++ objective. Since, we have extra constraint on the cluster centers to be on the unit hyper-sphere, the theoretical bound can be further improved.

We evaluated the performance of k-means++ initialization on four standard text datasets. We observed that only in one of these four datasets, random initialization leads to slightly better result. In one of four datasets, both initialization lead to similar results. For the other two datasets, k-means++ initialization improved the performance. From these observations, we recommend the strategy of trying several initialization using both random and k-means++ and then choosing the best result among them.

References

[1] M. Bangert, P. Hennig, and U. Oelfke, "Using an infinite von mises-fisher mixture model to cluster treatment beam directions in external radiation therapy," in *Ninth International Conference on Machine Learning and Applications (ICMLA)*, 2010, pp. 746–751.

[2] H. Tang, S. M. Chu, and T. S. Huang, "Generative model-based speaker clustering via mixture of von mises-fisher distributions," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2009, pp. 4101–4104.

[3] M. A. Hasnat, O. Alata, and A. Trémeau, "Hierarchical 3-d von mises-fisher mixture model," in *1st Workshop on Divergences and Divergence Learning (WDDL)*, 2013.

[4] A. Panagopoulos, D. Samaras, and N. Paragios, "Robust shadow and illumination estimation using a mixture model," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 651–658.

[5] A. N. Srivastava and M. Sahami, *Text mining: Classification, clustering, and applications*. CRC Press, 2009.

[6] T. McGraw, B. Vemuri, R. Yezierski, and T. Mareci, "Segmentation of high angular resolution diffusion mri modeled as a field of von mises-fisher mixtures," in *European Conference on Computer Vision (ECCV)*, 2006, pp. 463–475.

[7] S. Gopal and Y. Yang, "Von mises-fisher clustering models," in *Proceedings of The 31st International Conference on Machine Learning (ICML)*, 2014, pp. 154–162.

[8] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra, "Clustering on the unit hypersphere using von mises-fisher distributions," *Journal of Machine Learning Research*, vol. 6, pp. 1345–1382, Sep 2005.

[9] J. H. Wolfe, "Pattern clustering by multivariate mixture analysis," *Multivariate Behavioral Research*, vol. 5, no. 3, pp. 329–350, 1970.

[10] S. L. Sclove, "Population mixture models and clustering algorithms," Stanford University, Tech. Rep., 1973.

[11] F. Zhang, E. R. Hancock, C. Goodlett, and G. Gerig, "Probabilistic white matter fiber tracking using particle filtering and von mises–fisher sampling," *Medical image analysis*, vol. 13, no. 1, pp. 5–18, 2009.

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

[13] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "Np-hardness of euclidean sum-of-squares clustering," *Machine Learning*, vol. 75, no. 2, pp. 245–248, 2009.

[14] S. Sra, R. Hosseini, L. Theis, and M. Bethge, "Data modeling with the elliptical gamma distribution," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015, pp. 903–911.

[15] B. Zhang, C. Zhang, and X. Yi, "Competitive em algorithm for finite mixture models," *Pattern recognition*, vol. 37, no. 1, pp. 131–144, 2004.

[16] M. A. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 3, pp. 381–396, 2002.

[17] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, 2007, pp. 1027–1035.

[18] M. Fink, J.-H. Haunert, A. Schulz, J. Spoerhase, and A. Wolff, "Algorithms for labeling focus regions," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 12, pp. 2583–2592, 2012.

[19] L. Xie, L. Zheng, Z. Liu, and Y. Zhang, "Laplacian eigenmaps for automatic story segmentation of broadcast news," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 276–289, 2012.

[20] 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 (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[21] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press, 2008, vol. 1.

[22] E. W. Forgy, "Cluster analysis of multivariate data: efficiency vs interpretability of classifications," *Biometrics*, vol. 21, pp. 768–769, 1965.