Supplementary Material for Von Mises-Fisher Clustering Models

Siddharth Gopal
Carnegie Mellon University

Yiming Yang
Carnegie Mellon University

Contents

1 EM algorithm for VMF mixtures (vMFmix)

Given data , $\mathbf{X} = \{x_i\}_{i=1}^N$ where each $x_i \in \mathcal{R}^D$, the generative process for vMFmix with K classes is defined by,

$$z_i \sim \text{Mult}(.|\pi) \quad i = 1, 2..N$$
$$x_i \sim \text{vMF}(.|\mu_{z_i}, \kappa) \quad i = 1, 2..N$$

To estimate the hidden variables and cluster parameters, we can develop a simple EM algorithm. In the E-step, the expected value of the hidden variables $z_i = (z_{i1}, z_{i2}, ...z_{iK})$ is calculated. In the M-step, the parameter $\{\pi, \kappa, \boldsymbol{\mu} = \{\mu_1, \mu_2, ..\mu_K\}\}$ are optimized to maximize the likelihood.

1.1 E-step

The expected value of z_i is given by,

$$E[z_{ik}] = \frac{\pi_k \text{vMF}(x_i|\mu_k, \kappa)}{\sum_{j=1}^{K} \pi_j \text{vMF}(x_i|\mu_j, \kappa)}$$

1.2 M-step

Maximizing the likelihood w.r.t the cluster parameters leads to the following estimates,

$$\pi_k = \sum_{i=1}^{N} E[z_{ik}]$$
$$R_k = \sum_{i=1}^{N} E[z_{ik}] x_i \qquad , \qquad \mu_k = \frac{R_k}{\|R_k\|}$$

Using the procedure given in [2], the approximate estimate for κ is given by,

$$\bar{r} = \frac{\sum_{k=1}^{K} \|R_k\|}{N} \qquad , \qquad \kappa = \frac{\bar{r}D - \bar{r}^3}{1 - \bar{r}^2}$$

For a more detailed derivation [2]

2 Inference for Bayesian von-Mises Fisher Mixture Model (B-vMFmix)

We outline the procedures for variational inference as well as collapsed gibbs sampling for posterior inference in the bayesian von-Mises Fisher model. Given data , $\mathbf{X} = \{x_i\}_{i=1}^N$ where each $x_i \in \mathcal{R}^D$, the B-vMFmix assumes the following generative model with K classes

$$\begin{aligned}
&\pi \sim \text{Dirichlet}(.|\alpha) \\
&\mu_k \sim \text{vMF}(.|\mu_0, C_0) \quad k = 1, 2..K \\
&\kappa_k \sim \text{logNormal}(.|m, \sigma^2) \quad k = 1, 2..K \\
&z_i \sim \text{Mult}(.|\pi) \quad i = 1, 2..N \\
&x_i \sim \text{vMF}(.|\mu_{z_i}, \kappa_{z_i}) \quad i = 1, 2..N
\end{aligned}$$

In the model, the user specified prior parameters are $\{\alpha, \mu_0, C_0, m, \sigma^2\}$. We learn these prior parameters using empirical bayes and do not rely on the user to set any prior parameters.

2.1 Variational Inference

We assume the following factored form for the approximate posterior,

$$\begin{aligned}
q(\pi) &\equiv \text{Dirichlet}(.|\rho) \\
q(\mu_k) &\equiv \text{vMF}(.|\psi_k, \gamma_k) \\
q(z_i) &\equiv \text{Mult}(.|\lambda_i) \\
q(\kappa_k) &\equiv \text{Distribution discussed later}
\end{aligned}$$

The Likelihood of the data is given by

$$P(\mathcal{D}) = P(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}, \pi | m, \sigma^2, \mu_0, C_0, \alpha) = P(\pi|\alpha) \prod_{i=1}^{N} \text{Mult}(z_i|\pi) \text{vMF}(x_i|\mu_{z_i}, \kappa_{z_i}) \prod_{k=1}^{K} \text{vMF}(\mu_k|\mu_0, C_0) \text{logNormal}(\kappa_k|m, \sigma^2)$$

$$\log P(\mathcal{D}) = \log \text{Dirichlet}(\pi|\alpha) + \sum_{i=1}^{N} \log \text{Mult}(z_i|\pi) + \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik} \log \text{vMF}(x_i|\mu_k, \kappa_k) + \sum_{k=1}^{K} \log \text{vMF}(\mu_k|\mu_0, C_0) + \sum_{k=1}^{K} \log \left(\text{logNormal}(\kappa_k|m, \sigma^2)\right)$$

$$\begin{aligned}
\log P(\mathcal{D}) = &- \log B(\alpha) + \sum_{k=1}^{K} (\alpha - 1) \log(\pi_k) + \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik} \log \pi_k + \\
&\sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik} \left(\log C_D(\kappa_k) + \kappa_k x_i^\top \mu_k\right) + \\
&\sum_{k=1}^{K} \left(\log C_D(C_0) + C_0 \mu_k^\top \mu_0\right) \\
&\sum_{k=1}^{K} \left(-log(\kappa_k) - \frac{1}{2}\log(2\pi\sigma^2) - \frac{(\log(\kappa_k) - m).^2}{2\sigma^2}\right)
\end{aligned}$$

where $C_D(x)$ is the normalization constant of the vMF distribution with concentration parameter x and dimensionality D. Specifically,

$$\log C_D(\kappa_k) = d' \log(\kappa_k) - (d'+1)\log(2\pi) - \log I_{d'}(\kappa_k)$$
$$\text{where } d' = \frac{D}{2} - 1$$

The variational Lower bound is given by

$$VLB = E_q[logP(\mathcal{D})] - H(q)$$

The posterior expectations are given by,

$$E_q[z_{i,k}] = \lambda_{i,k}$$
$$E_q[\log(\pi_k)] = \Psi(\rho_k) - \Psi\left(\sum_j \rho_j\right)$$
$$E_q[\mu_k] = E_q\left[\frac{I_{\frac{D}{2}}(\kappa_k)}{I_{\frac{D}{2}-1}(\kappa_k)}\right]\mu_k$$

Optimizing the VLB w.r.t ρ,

$$\rho_k = \alpha + \sum_{i=1}^{N} \lambda_{ik}$$

Optimizing VLB w.r.t $\mathbf{Z}$,

$$\lambda_{ik} \propto exp(E_q[\log \text{vMF}(x_i|\mu_k, \kappa_k)] + E_q[\log(\pi_k)])$$

where

$$E_q[\log \text{vMF}(x_i|\mu_k, \kappa_k)] = E_q[\log C_D(\kappa_k)] + E_q[\kappa_k] x_i^\top E_q[\mu_k]$$

Optimizing VLB w.r.t cluster means $\boldsymbol{\mu}$

$$\gamma_k = \left\| E_q[\kappa_k]\left(\sum_{i=1}^{N} E_q[z_{ik}]x_i\right) + C_0\mu_0 \right\|$$
$$\psi_k = \frac{E_q[\kappa_k]\left(\sum_{i=1}^{N} E_q[z_{ik}]x_i\right) + C_0\mu_0}{\gamma_k}$$

where

$$E_q[z_{ik}] = \lambda_{ik}$$

Optimizing VLB w.r.t $\boldsymbol{\kappa}$.

1. **Sampling method** In the sampling scheme, we do not assume any specific form of posterior distribution for the cluster concentration parameters (κ_k's).

   $$q(\kappa_k) \equiv \text{No-specific form}$$

   During variational inference we need to calculate the $E_q[\log C_D(\kappa_k)]$. The $C_D(\kappa_k)$ has a first-order modified bessel function term whose expectation is **difficult** to compute for common distributions. We

were not able to find any suitable posterior distribution for κ_k that enables *straight-forward* computation of $\log C_D(\kappa_k)$. Therefore, we rely on a partial MCMC sampling scheme to estimate the distribution for κ_k given the rest of the posterior distributions.

NOTE : In [3] an unnormalized conjugate prior was used - but the problem is it not clear whether it is even a valid distribution !

The true posterior distribution of κ_k is given by,

$$\begin{aligned}
P(\kappa_k|\mathbf{X}, m, \sigma^2, \mu_0, C_0, \alpha) &= \frac{P(\kappa_k, \mathbf{X}|m, \sigma^2, \mu_0, C_0, \alpha)}{P(\mathbf{X}|m, \sigma^2, \mu_0, C_0, \alpha)} \\
&\propto P(\kappa_k, \mathbf{X}|m, \sigma^2, \mu_0, C_0, \alpha) \\
&\propto \int_{\mathbf{Z}} \int_{\pi} \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\kappa}^{-k}} P(\kappa_k, \mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}^{-k}, \pi|m, \sigma^2, \mu_0, C_0, \alpha) \\
&\approx E_q \left[P(\kappa_k, \mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}^{-k}, \pi|m, \sigma^2, \mu_0, C_0, \alpha) \right]
\end{aligned}$$

Using Jensen inequality, we have,

$$E_q \left[P(\kappa_k, \mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}^{-k}|m, \sigma^2, \mu_0, C_0, \pi) \right] \geq \exp\left(E_q \left[\log P(\kappa_k, \mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}^{-k}|m, \sigma^2, \mu_0, C_0, \pi) \right] \right)$$

where, the inner expectation (after discarding terms which do not depend on κ_k) is ,

$$\begin{aligned}
&E_q \left[\log P(\kappa_k, \mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}^{-k}|m, \sigma^2, \mu_0, C_0, \pi) \right] = \\
&\left[\sum_{i=1}^{N} \lambda_{ik} \left(\log C_D(\kappa_k) + \kappa_k x_i^\top E_q[\mu_k] \right) \right] - \log(\kappa_k) - \frac{(\log(\kappa_k) - m)^2}{2\sigma^2} + \text{ constants ..} \\
&= n_k \log C_D(\kappa_k) + \kappa_k \left(\sum_{i=1}^{N} \lambda_{ik} x_i^\top E_q[\mu_k] \right) - \log(\kappa_k) - \frac{(\log(\kappa_k) - m)^2}{2\sigma^2} + \text{ constants ..}
\end{aligned}$$

where

$$n_k = \sum_{i=1}^{N} \lambda_{ik}$$

This implies that,

$$P(\kappa_k|\mathbf{X}, m, \sigma^2, \mu_0, C_0, \pi) \propto \tag{1}$$

$$\exp\left(n_k \log C_D(\kappa_k) + \kappa_k \left(\sum_{i=1}^{N} \lambda_{ik} x_i^\top E_q[\mu_k] \right) \right) \text{logNormal}(\kappa_k|m, \sigma^2) \tag{2}$$

Based on the above, it is easy to see that computation becomes easier if we use the logNormal$(.|m, \sigma^2)$ as the proposal distribution for MCMC sampling i.e. the logNormal likelihood terms cancel.

However, we still recommend using a different proposal distribution such as logNormal or Normal around the current value of the parameter for convergence reasons (note that the κ's are relative large positive values, therefore using a Normal Proposal distribution works fine). This is because the posterior of κ_k's are reasonably concentrated around a small region and since the prior distribution is reasonably vague, simply using logNormal$(.|m, \sigma^2)$ as the proposal distribution increases the number of iterations for convergence.

Note that we can estimate $E_q \log C_D(\kappa_k)$ and $E_q \left[\frac{I_{\frac{D}{2}}(\kappa_k)}{I_{\frac{D}{2}-1}(\kappa_k)} \right]$ by drawing samples after a specific burnin period, and use this for computing $E_q[z_{i,k}]$ and $E_q[\mu_k]$. Because of the sampling, the variational bound

is no longer guaranteed to be a valid lower-bound. However, we observed that the posterior of κ_k's is highly concentrated around the mode and therefore estimating $E_q \log C_D(\kappa_k)$ from samples is accurate enough for good performance.

Another alternative is to use grid-search, where we approximate the calculate the probability density over a finite range of points in an interval. We then use this probability density over the interval to calculate the expectation of various quantities. In practice, this worked equally well (although a little slower).

2. **Bounding method** In this scheme, we assume that posterior distribution for the cluster concentration parameters (κ_k's) is log-normally distributed,

$$q(\kappa_k) \equiv \text{logNormal}(.|a_k, b_k)$$

In order to find the parameters of the posterior of the $k'th$ cluster, i.e. a_k, b_k, we optimize,

$$(\sum_i \lambda_{ik}) E_q[\log C_D(\kappa_k)] + E_q[\kappa_k] \sum_i \lambda_{ik} x_i^\top E_q[\mu_k] - E_q[\log(\kappa_k)] - E_q\left[\frac{(log(\kappa_k) - m)^2}{2\sigma^2}\right] - H(q)$$

where

$$E_q[\kappa_k] = \exp(a_k + .5 * b_k^2)$$
$$E_q[\log(\kappa_k)] = a_k$$
$$E_q\left[\frac{(log(\kappa_k) - m)^2}{2\sigma^2}\right] = \frac{1}{2\sigma^2}\left(E_q[\log(\kappa_k)^2] - 2mE_q[\log(\kappa_k)] + m^2\right)$$
$$E_q[\log(\kappa_k)^2] = a_k^2 + b_k^2$$
$$H(q(\kappa_k)) = .5 + .5\log(2\pi b_k^2) + a_k$$
$$E_q[\log C_D(\kappa_k)] = d' E_q[\log(\kappa_k)] - (d'+1)\log(2\pi) - E_q[\log(I_{d'}(\kappa_k))]$$

The difficult term is the computation of $E_q[\log(I_{d'}(\kappa_k))]$. We upper bound this term using a turner type inequality. From [4], we use the fact that,

$$u\frac{I_v(u)'}{I_v(u)} \leq \sqrt{u^2 + v^2}$$
$$\frac{I_v(u)'}{I_v(u)} \leq \sqrt{1 + \frac{v}{u}^2}$$

Integrating over $u > 0$,

$$log(I_v(u)) \leq \sqrt{u^2+v^2} - v\log\left(v\sqrt{v^2+u^2} + v^2) + v^2\right) - \sqrt{u^2+v^2} - v\log(u)$$

$\Rightarrow$

$$E_q[log(I_v(u))] \leq E_q[\sqrt{u^2+v^2}] - vE_q[\log\left(v\sqrt{v^2+u^2} + v^2) + v^2\right)] - E_q[\sqrt{u^2+v^2}] - vE_q[\log(u)]$$

The individual expectations can be computed using delta method approximation, i.e.,

$$E[f(x)] \approx f(E[x]) + f''(E[x])\frac{Var[x]}{2}$$

Empirical Bayes estimate for prior parameters

The empirical bayes estimate for m,σ, is given by,

$$max_{m,\sigma^2} \quad -\frac{K}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{k=1}^{K}(E_q[\log(\kappa_k)^2] - 2mE_q[\log(\kappa_k)] + m^2)$$

$$\Rightarrow \quad m = \frac{1}{K}\sum E_q[\log(\kappa_k)], \quad \sigma^2 = \frac{1}{K}\sum_{k=1}^{K} E_q[\log(\kappa_k)^2] - m^2$$

The empirical bayes estimate for μ_0, C_0 is given by,

$$\max_{\mu_0, C_0} \quad K\log C_D(C_0) + C_0\mu_0^\top \left(\sum_{k=1}^{K} E_q[\mu_k]\right)$$

Following [7], the updates are given by,

$$\mu_0 = \frac{\sum_{k=1}^{K} E_q[\mu_k]}{\left\|\sum_{k=1}^{K} E_q[\mu_k]\right\|}, \quad C_0 = \frac{\bar{r}D - \bar{r}^3}{1-\bar{r}^2} \quad \text{where} \quad \bar{r} = \frac{\left\|\sum_{k=1}^{K} E_q[\mu_k]\right\|}{K}$$

The empirical bayes estimate for α is given by,

$$\max_{\alpha>0} -\log B(\alpha) + (\alpha-1)s$$

$$\text{where } s = \sum_{k=1}^{K} E_q[\log \pi_k]$$

$$B(\alpha) = \frac{\Gamma(\alpha)^K}{\Gamma(K\alpha)}$$

Since there is no closed-form solution, we need to apply some numerical optimization like gradient descent to find the MLE.

2.2 Collapsed Gibbs sampling

The standard Gibbs sampling of the parameters can be easily derived by calculating their conditional distributions. The problem is that sampling of the high dimensional vMF mean parameter μ_k is hard (computationally expensive). However, because the vMF distributions are conjugate, we can exploit this to completely integrate out the mean parameters and the mixing distribution. This implies that there are only two sets of

parameters $= \{\mathbf{Z}, \boldsymbol{\kappa}\}$. The conditional distributions are given by,

$$P(z_i = t|\mathbf{Z}^{-i}, X, \boldsymbol{\kappa}, m, \sigma^2, \mu_0, C_0) = \tag{3}$$

$$\int_\pi \int_{\boldsymbol{\mu}} P(x_i|z_i, \mu_k, \kappa_k) P(z_i|\pi) P(\pi|\alpha) \prod_{j \neq i} P(z_j|\pi) P(x_j|\mu_{z_j}, \kappa_{z_j}) \prod_{k=1}^{K} P(\mu_k|\mu_0, C_0) \text{logNormal}(\kappa_k|m, \sigma^2) \tag{4}$$

$$\propto \int_\pi P(\pi|\alpha) P(z_i|\pi) \prod_{j \neq i} P(z_j|\pi) \; . \; \prod_{k=1}^{K} \int_{\mu_k} \left(P(x_i|z_i, \mu_k, \kappa_k) \prod_{j \neq i, z_j = k} P(x_j|\mu_k, \kappa_k) \right) P(\mu_k|\mu_0, C_0) \tag{5}$$

$$\propto \frac{\Gamma(K\alpha) \prod_{k=1}^{K} \Gamma(\alpha + n_{k,-i} + I(t=k))}{\Gamma(K\alpha + N) \prod_{k=1}^{K} \Gamma(\alpha)} \prod_{k=1}^{K} \frac{C_D(\kappa_k)^{n_{k,-i} + I(k=t)} C_D(C_0)}{C_D\left(\left\| \kappa_k \left(I(t=k) x_i + \sum_{j \neq i, z_j = k} x_j \right) + C_0 \mu_0 \right\| \right)} \tag{6}$$

$$\propto (\alpha + n_{t,-i}) \, C_D(\kappa_t) \frac{C_D\left(\left\| \kappa_t \sum_{j \neq i, z_j = t} x_j + C_0 \mu_0 \right\| \right)}{C_D\left(\left\| \kappa_t \left(\sum_{j : z_j = t} x_j \right) + C_0 \mu_0 \right\| \right)} \tag{7}$$

Similarly, we derive the conditional distribution for κ_k,

$$P(\kappa_k|..) \propto \int_{\mu_k} \prod_{z_i = k} P(x_i|\mu_k, \kappa_k) P(\mu_k|\mu_0, C_0) \text{logNormal}(\kappa_k|m, \sigma^2) \tag{8}$$

$$\propto \frac{C_D(\kappa_k)^{n_k} C_D(C_0)}{C_D\left(\left\| \kappa_k \sum_{j : z_j = k} x_j + C_0 \mu_0 \right\| \right)} \text{logNormal}(\kappa_k|m, \sigma^2) \tag{9}$$

Since the conditional distribution is not of a standard form, we need to use a step of MCMC sampling (with appropriate proposal distribution) to draw κ_k.
The main advantage of sampling is that no approximation for the posterior is required, it is guaranteed that the samples will eventually be drawn from the true posterior. The downside is that it is not clear how many samples must be drawn.

Empirical Bayes estimate for prior parameters

The empirical bayes estimation in the context of MCMC sampling is done as described in [5]. The prior parameters are estimated by maximizing the sum of the marginal likelihood of all the samples.

3 Inference for Hierarchical von Mises-Fisher Mixture Model (H-vMFmix)

Given data , $\mathbf{X} = \{x_i\}_{i=1}^N$ where each $x_i \in \mathcal{R}^D$, and a hierarchy pa. Assume the following generative model for the data

$$\begin{aligned}
&\pi \sim \text{Dirichlet}(.|\alpha) \\
&\mu_k \sim \text{vMF}(.|\mu_{pa(k)}, \kappa_{pa(k)}) \quad k = 1, 2..K \\
&\kappa_k \sim \text{logNormal}(.|m, \sigma^2) \quad k = 1, 2..K \\
&z_i \sim \text{Mult}(.|\pi), z_i \in \{\text{Leaf nodes}\} \quad i = 1, 2..N \\
&x_i \sim \text{vMF}(.|\mu_{z_i}, \kappa_{z_i}) \quad i = 1, 2..N
\end{aligned}$$

Assume the following factored form for the posterior,

$$\begin{aligned}
&q(\pi) \equiv \text{Dirichlet}(.|\rho) \qquad q(\mu_k) \equiv \text{vMF}(.|\psi_k, \gamma_k) \\
&q(z_i) \equiv \text{Mult}(.|\lambda_i) \qquad q(\kappa_k) \equiv \text{No-specific form}
\end{aligned}$$

The posterior distribution of cluster assignment variables at the leaf-nodes is given by,

$$\begin{aligned}
&\rho_k = \alpha + \sum_{i=1}^{N} \lambda_{ik} \\
&\lambda_{ik} \propto exp(E_q[\log \text{vMF}(x_i|\mu_k, \kappa_k)] + E_q[\log(\pi_k)])
\end{aligned}$$

where

$$\begin{aligned}
&E_q[\log(\pi_k)] = \Psi(\rho_k) - \Psi\left(\sum_j \rho_j\right) \\
&E_q[\log \text{vMF}(x_i|\mu_k, \kappa_k)] = E_q[\log C_D(\kappa_k)] + E_q[\kappa_k] x_i^\top E_q[\mu_k]
\end{aligned}$$

The posterior distribution of the cluster means are updated as follows,

For leaf nodes

$$\begin{aligned}
\gamma_k &= \left\| E_q[\kappa_k] \left(\sum_{i=1}^{N} E_q[z_{ik}] x_i \right) + E_q[\kappa_{pa(k)}] E_q[\mu_{pa(k)}] \right\| \\
\psi_k &= \frac{E_q[\kappa_k] \left(\sum_{i=1}^{N} E_q[z_{ik}] x_i \right) + E_q[\kappa_{pa(k)}] E_q[\mu_{pa(k)}]}{\gamma_k}
\end{aligned}$$

where

$$E_q[z_{ik}] = \lambda_{ik}$$

For non-leaf nodes

$$\begin{aligned}
\gamma_k &= \left\| E_q[\kappa_k] \sum_{c:pa(c)=k}^{N} E_q[\mu_c] + E_q[\kappa_{pa(k)}] E_q[\mu_{pa(k)}] \right\| \\
\psi_k &= \frac{E_q[\kappa_k] \sum_{c:pa(c)=k}^{N} E_q[\mu_c] + E_q[\kappa_{pa(k)}] E_q[\mu_{pa(k)}]}{\gamma_k}
\end{aligned}$$

The posterior computation for κ_k's follows in a manner similar to the previous section.

4 Inference for Temporal von Mises-Fisher Mixture Model (T-vMFmix)

Given data , $\mathbf{X} = \{\{x_{t,i}\}_{i=1}^{N_t}\}_{t=1}^{T}$ where each $x_{t,i} \in \mathcal{R}^D$. Assume the following generative model for the data

$$
\begin{aligned}
&\pi \sim \text{Dirichlet}(.|\alpha) \\
&\mu_{t,k} \sim \text{vMF}(.|\mu_{t-1,k}, C_0) \quad \forall t = 1, 2...T; k = 1, 2, 3..K \\
&\kappa_k \sim \text{logNormal}(.|m, \sigma^2) \quad k = 1, 2..K \\
&z_{t,i} \sim \text{Mult}(.|\pi) \quad \forall i = 1, 2, ..N_t \\
&x_{t,i} \sim \text{vMF}(.|\mu_{z_{t,i}}, \kappa_{z_{t,i}}) \quad \forall t = 1, 2..T; i = 1, 2, ...N_t
\end{aligned}
$$

Note that the data $\mathbf{X}$ is given in the form of T time-steps of data. Each time-step is associated with some N_t instances $x_{t,i}, i = 1, 2, ..N_t$.
The approximate posterior follows the factored form given below,

$$
\begin{aligned}
q(\pi) &\equiv \text{Dirichlet}(.|\rho) \\
q(\kappa_k) &\equiv \text{No-specific form} \\
\mu_{t,k} &\equiv \text{vMF}(.|\psi_{t,k}, \gamma_{t,k}) \\
z_{t,i} &\equiv \text{Mult}(.|\lambda_{t,i})
\end{aligned}
$$

Note that for simplicity we have assumed a fully factorised form for the posterior distribution with no dependencies between the μ's from adjacent time-points in the posterior.
The likelihood of the data is given by,

$$
\begin{aligned}
&P(\mathbf{X}, \mathbf{Z}, \pi, \boldsymbol{\mu}, \boldsymbol{\kappa}|\mu_{-1}, C_0, \alpha, m, \sigma^2) = \\
&P(\pi|\alpha) \prod_{k=1}^{K} P(\kappa_k|m, \sigma^2) \prod_{t=1}^{T} \left[\prod_{k=1}^{K} \left(P(\mu_{t,k}|\mu_{t-1,k}, C_0) \prod_{i=1}^{N_t} \left(P(z_{t,i}|\pi) \prod_{k=1}^{K} \left(P(x_{t,i}|\mu_{t,k}, \kappa_k)^{z_{t,i}} \right) \right) \right) \right]
\end{aligned}
$$

$$
\begin{aligned}
&\log P(\mathbf{X}, \mathbf{Z}, \pi, \boldsymbol{\mu}, \boldsymbol{\kappa}|\mu_{-1}, C_0, \alpha, m, \sigma^2) = \\
&\quad \sum_{k=1}^{K} -\log(\kappa_k) - .5\log(2\pi) - \log(\sigma) - \frac{(\log(\kappa_k) - m)^2}{2\sigma^2} - \log B(\alpha) + \sum_{k=1}^{K} (\alpha - 1) \log \pi_k \\
&+ \sum_{t=1}^{T} \left[\sum_{k=1}^{K} \left(\log C_D(C_0) + C_0 \mu_{t,k}^{\top} \mu_{t-1,k} \right) + \right. \\
&\quad + \sum_{i=1}^{N_t} \sum_{k=1}^{K} \pi_k z_{t,i,k} + \\
&\quad \left. \sum_{i=1}^{N_t} \sum_{k=1}^{K} z_{t,i,k} \left[\log C_D(\kappa_k) + \kappa_k x_i^{\top} \mu_k \right] \right]
\end{aligned}
$$

Following in a similar manner, by optimizing the variational lower-bound, we get the following updates for the posterior parameters,

$$\lambda_{t,i,k} \propto \exp\left(E_q[\log \pi_k] + E_q[\log P(x_{t,i}|\mu_{t,k}, \kappa_k)]\right)$$

$$\rho_k = \alpha + \sum_{i=1}^{N} \lambda_{ik}$$

$$\gamma_{t,k} = \left\| E_q[\kappa_k] \sum_{i=1}^{N_t} E_q[z_{t,i,k}] x_i + C_0 E_q[\mu_{t-1,k}] + I(t < T) C_0 E_q[\mu_{t+1,k}] \right\|$$

$$\psi_{t,k} = \frac{E_q[\kappa_k] \sum_{i=1}^{N_t} E_q[z_{t,i,k}] x_i + C_0 E_q[\mu_{t-1,k}] + I(t < T) C_0 E_q[\mu_{t+1,k}]}{\gamma_{t,k}}$$

For the partial MCMC sampling for κ_k,

$$E_q[\log P(\kappa_k, \mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}^{-k}, \boldsymbol{\eta}|..)] =$$
$$-\log(\kappa_k) - \frac{(\log(\kappa_k) - m)^2}{2\sigma^2} + \left(\sum_{t=1}^{T}\sum_{i=1}^{N_t} E_q[z_{t,i,k}]\right) \log C_d(\kappa_k) + \sum_{t=1}^{T}\sum_{i=1}^{N_t} E_q[z_{t,i,k}] x_i^\top E_q[\mu_k] + constants$$

If instead, one were to perform MLE estimation of the κ_k's without assuming the log-normal prior,

$$\max_{\kappa_k} \left(\sum_{t=1}^{T}\sum_{i=1}^{N_t} E_q[z_{t,i,k}]\right) \log C_d(\kappa_k) + \sum_{t=1}^{T}\sum_{i=1}^{N_t} E_q[z_{t,i,k}] x_i^\top E_q[\mu_k]$$

$$\Rightarrow s = \frac{\sum_{t=1}^{T}\sum_{i=1}^{N_t} E_q[z_{t,i,k}] x_i^\top E_q[\mu_k]}{\sum_{t=1}^{T}\sum_{i=1}^{N_t} E_q[z_{t,i,k}]}$$

$$\bar{r} = ||s||$$

$$\kappa_k = \frac{\bar{r}D - \bar{r}^3}{1 - \bar{r}^2}$$

For the MLE estimations, for C_0

$$\max_{C_0} \quad KT \log C_D(C_0) + C_0 \sum_{t=1}^{T}\sum_{k=1}^{K} E_q[\mu_{t,k}]^\top E_q[\mu_{t-1,k}]$$

$$\Rightarrow s = \frac{\sum_{t=1}^{T}\sum_{k=1}^{K} E_q[\mu_{t,k}]^\top E_q[\mu_{t-1,k}]}{KT}$$

$$\bar{r} = ||s||$$

$$C_0 = \frac{\bar{r}D - \bar{r}^3}{1 - \bar{r}^2}$$

Note that C_0 determines the level of influence of the previous time-step to the next time-step. In some sense it acts as a regularization term forcing the parameters of the next time-step to be similar to the previous one.

We recommend that this be set manually than being learnt from the data. The reason is that, since the estimation is done through maximum likelihood, the likelihood can generally be improved by moving $C_0 \to 0$, i.e. towards no regularization.

5 Datasets

Table 1: Dataset Statistics

Dataset	#Instances	#Training	#Testing	#True Clusters	#Features
TDT4	622	311	311	34	8895
TDT5	6366	3183	3183	126	20733
CNAE	1079	539	540	9	1079
NIPS	2483	1241	1242	-	14036
K9	2340	1170	1170	20	21839
NEWS20	18744	11269	7505	20	53975

6 Clustering performance using true cluster labels

We present the complete results of all the methods on all the datasets using the following evaluation metrics,

1. Mutual Information (MI) [1]
2. Normalized Mutual Information (NM) [1]
3. Rand Index (RI) [1]
4. Adjusted Rand Index (ARI) [2]
5. Purity [1]
6. Macro-F_1 (ma-F1) : Is a measure similar to purity - each cluster is assigned to the class-label which is most frequent in the cluster. Then the standard Macro-F_1 classification measure is used to measure the performance. This measure compliments the purity measure in the sense that we want to detect all clusters; not just the ones which have large number of instances.

Methods compared,

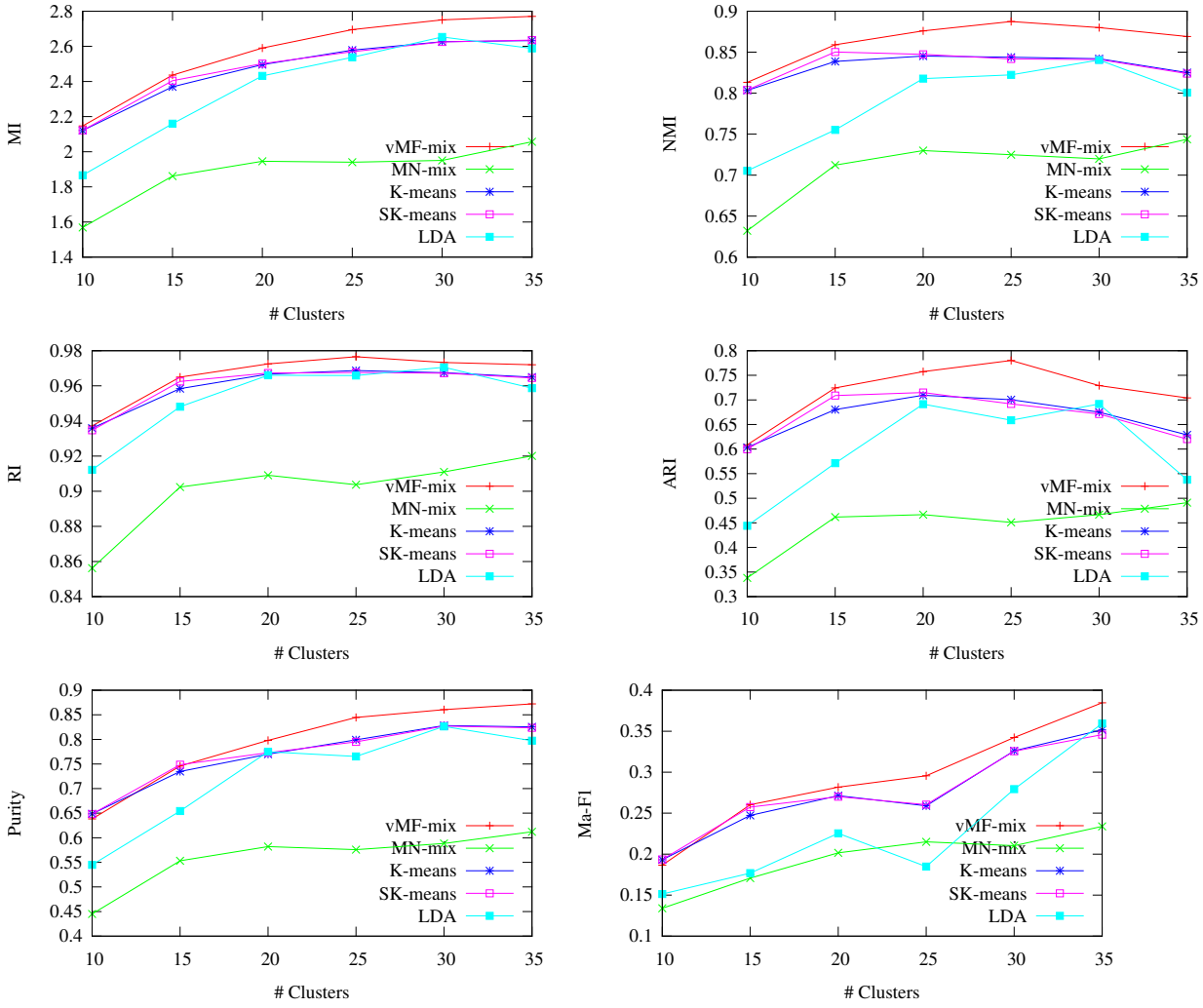
1. von Mises Fisher Mixture Model (vMFmix) trained using the EM-algorithm.
2. Multinomial Mixture Model (MN) with dirichlet prior.
3. K-means (K-means): The standard K-means algorithm with hard assignments after each iteration.
4. Soft K-means (SK-means): The simplest Gaussian mixture model with K uknown means and a single variance parameter learnt from the data. (Presetting the variance to 1 performs so badly that almost all scores are zero). Note that there is no common prior for the means.
5. Latent Dirichlet Allocation: Although not suitable for single-labeled datasets, we still show the results by assigning each document to the class with the largest posterior probability.

The methods are compared on all the datasets on which true-cluster labels are available. Note that there is no splitting of the data into a separate training or testing set. For vMFmix, KM and SKM the Tf-Idf normalized representation is used (KM and SKM perform better with Tf-Idf normalization than without it). For MM and LDA, word-counts based representation is used. To make a fair comparison, the same *random*

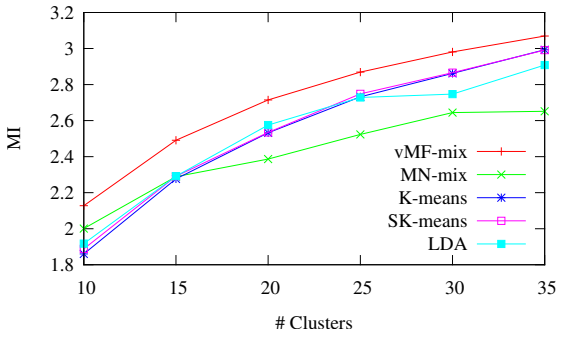
[1] http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html
[2] http://en.wikipedia.org/wiki/Rand_index#Adjusted_Rand_index

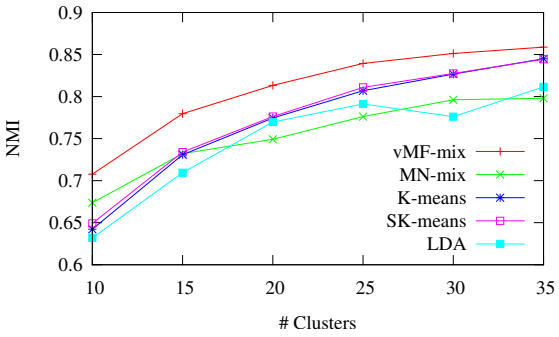
initial values of the cluster assignment variables for all the algorithms is used. For all the experimental settings - refer last section.
After the publication of the paper, we realized that using Kmeans++ initialization instead of random initialization boosts performance on some of the datasets. We leave this comparison for further studies.

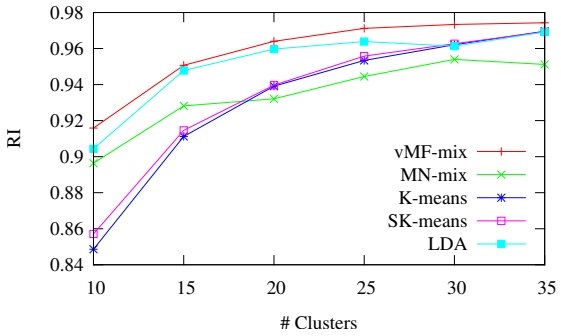
6.1 Results on the TDT4 dataset

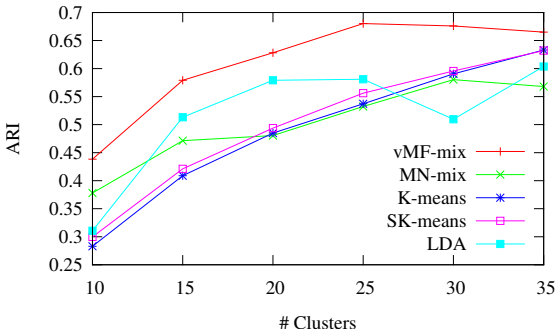


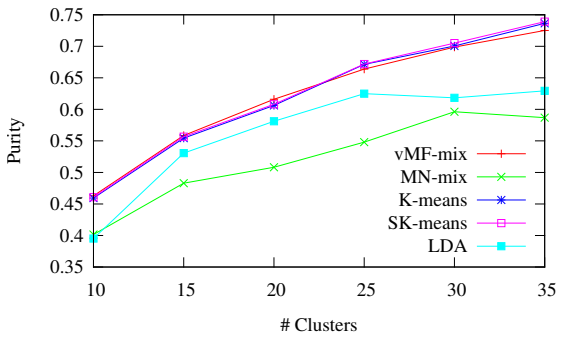
6.2 Results on the TDT5 Dataset

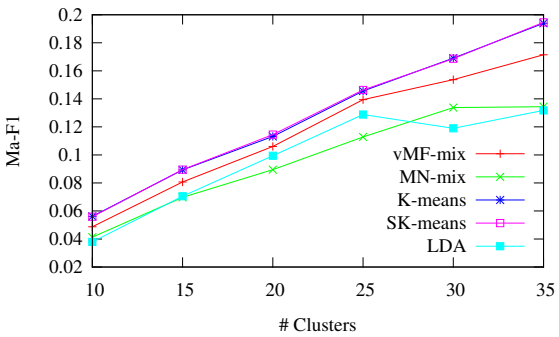




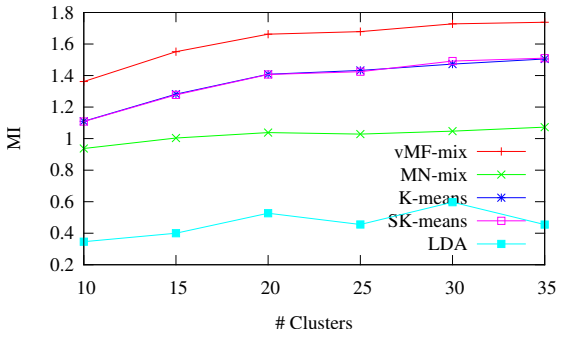


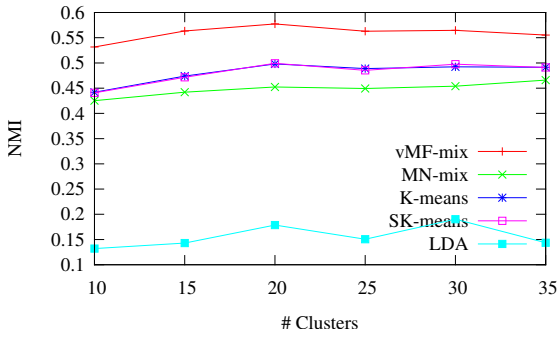


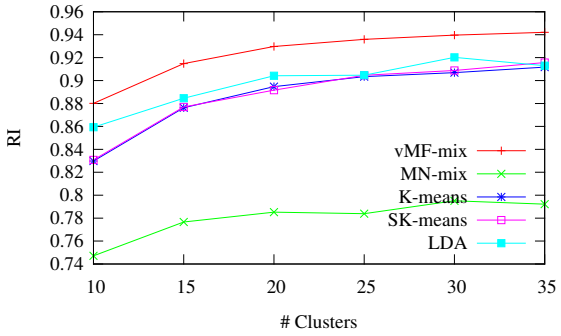


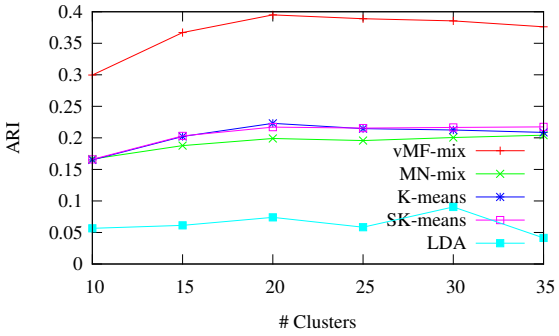


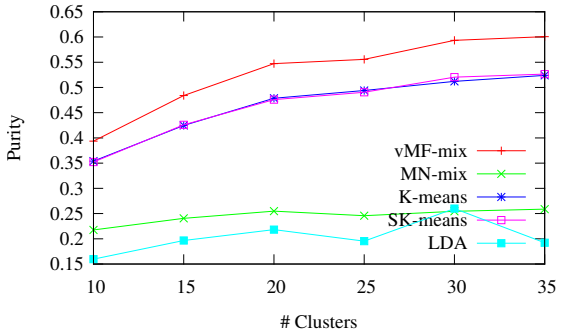
6.3 Results on the News20 Dataset

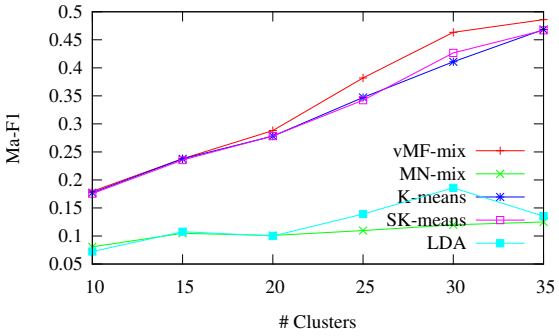




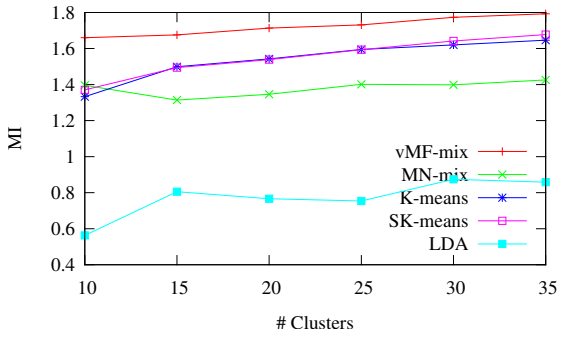


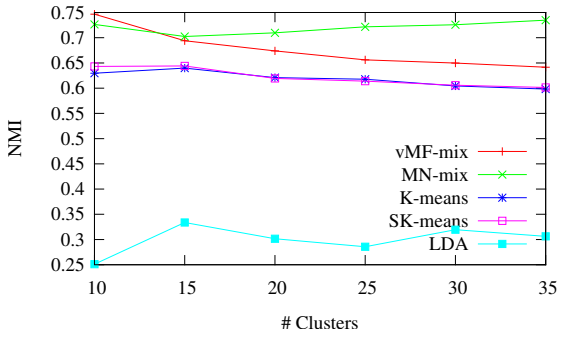


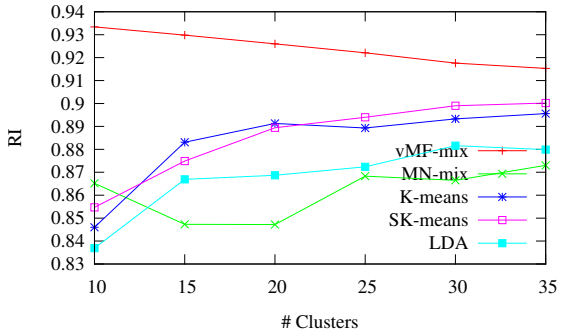


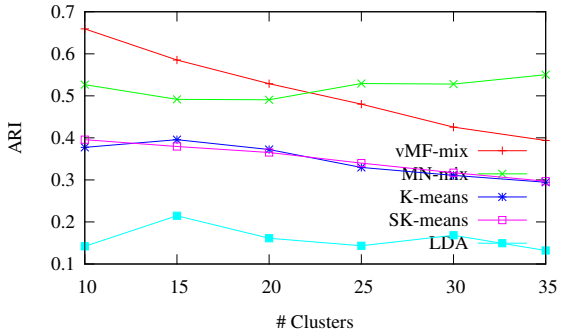


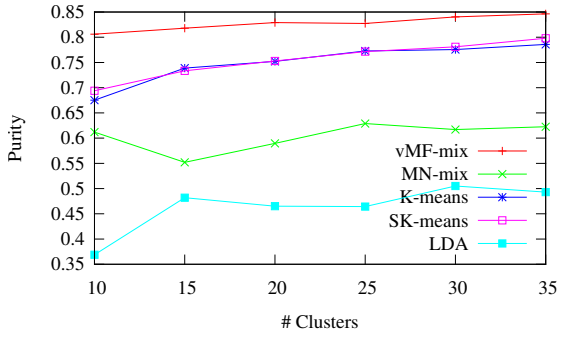
6.4 Results on the CNAE Dataset

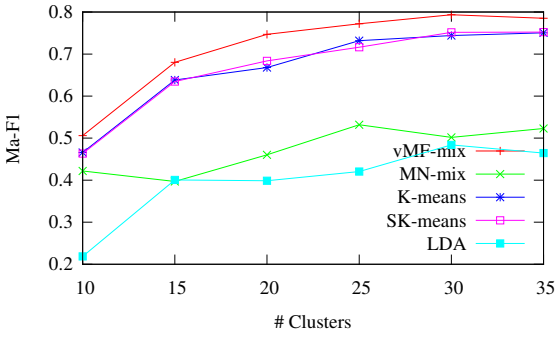




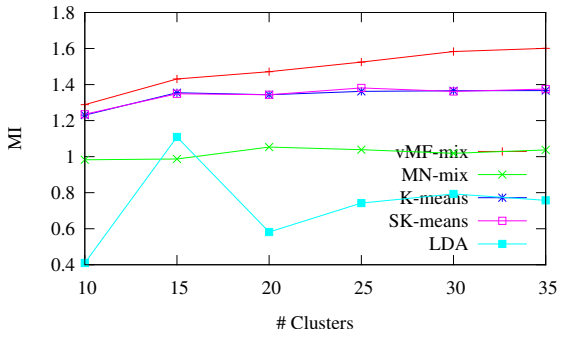


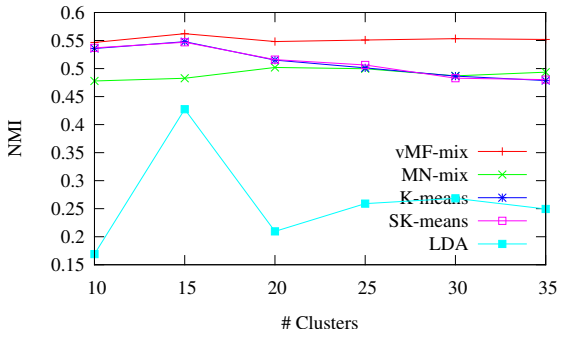


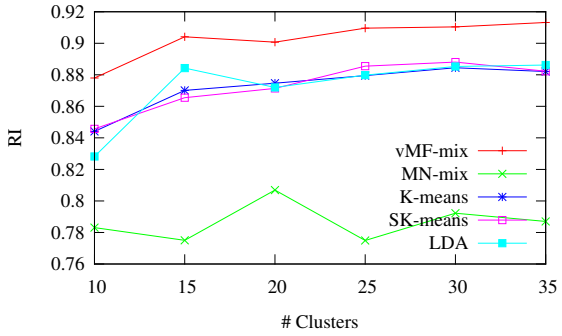


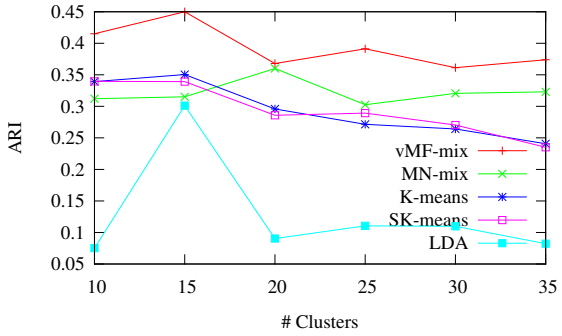


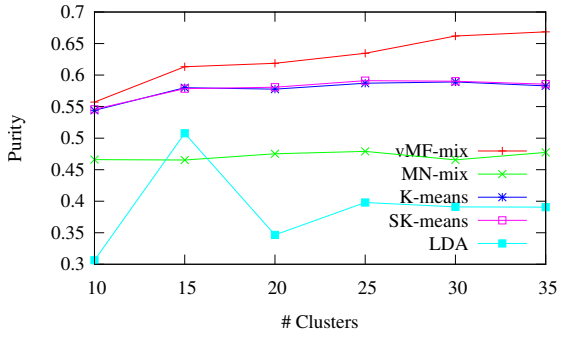
6.5 Results on the K9 Dataset

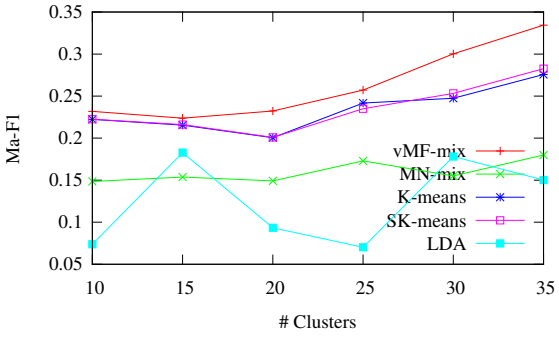






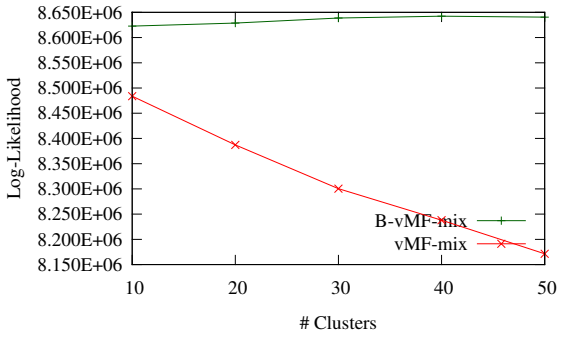




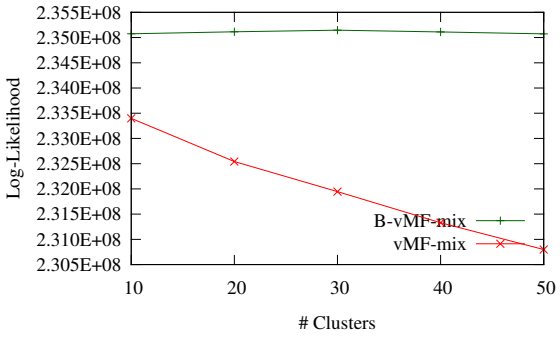


7 Results of Bayesian vs non-Bayesian model

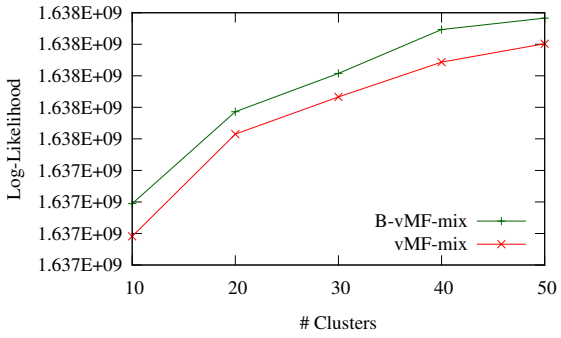
The Bayesian vMF mixture model (B-vMFmix) trained using the variational method with sampling scheme is compared with the simple vMF mixture model (vMFmix) trained through EM algorithm. We compare the methods using the log-likelihood of the data on a held-out test set. Note that true-cluster labels are not required for evaluation.



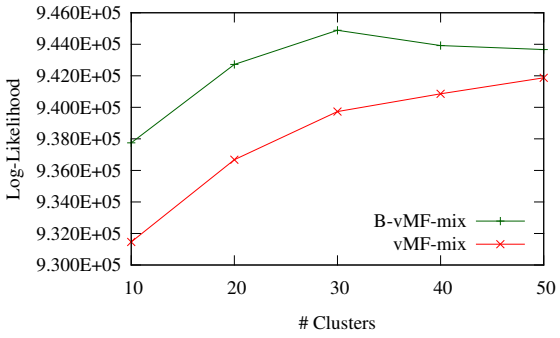
(a) TDT 4 Dataset



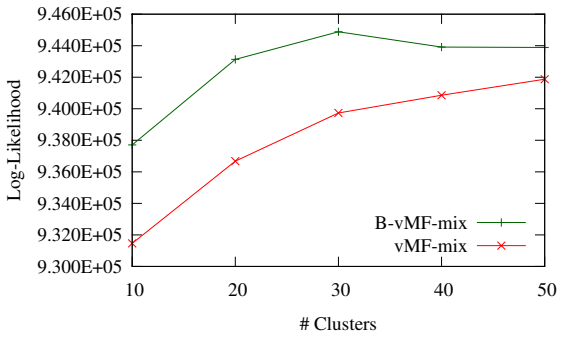
(b) TDT 5 Dataset



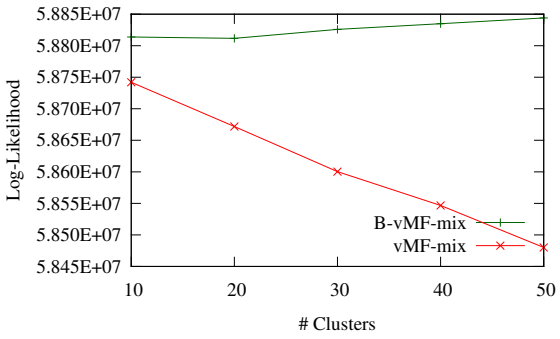
(c) News 20 Dataset



(d) CNAE Dataset



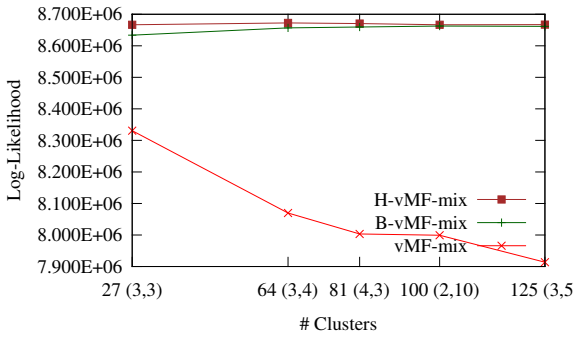
(e) K9 Dataset



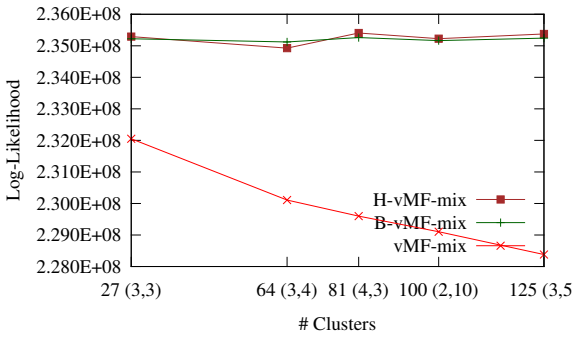
(f) NIPS Dataset

8 Results of Hierarchical Bayesian methods (H-vMFmix) vs other models (vMFmix, B-vMFmix)

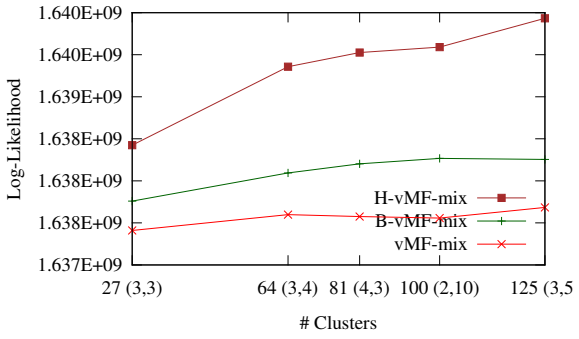
The H-vMFmix and B-vMFmix are trained through variational inference and vMFmix is trained through EM algorithm. We compare the methods using the log-likelihood of the data on a held-out test set.
The results are presented on 5 different hierarchies with varying depth of hierarchy and branch factor (branch factor is the #children under each node). More specifically we tested with (h=height,b=branch). For example (h=3,b=4) is a hierarchy 3 levels deep; the zeroth level is the root-node under which there are 4 children which form the first level, each of these 4 children in turn have 4 children each leading to 16 nodes in the second level, each of these 16 children have 4 children each which forms 64 children at the third level. We present on the results on (h=3,b=3),(h=3,b=4),(h=4,b=3),(h=2,b=10),(h=3,b=5).



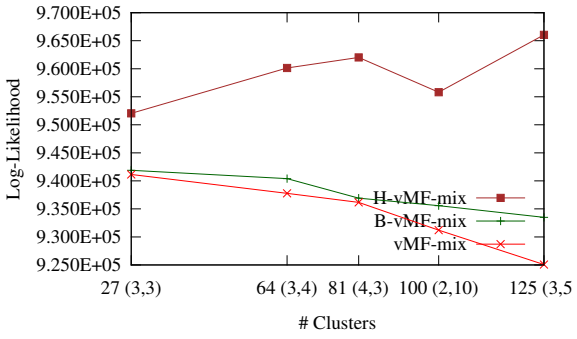
(a) TDT 4 Dataset



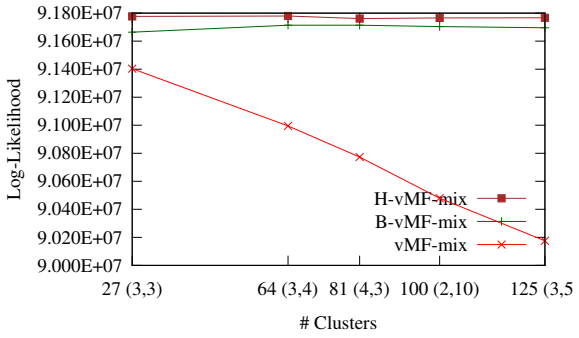
(b) TDT 5 Dataset



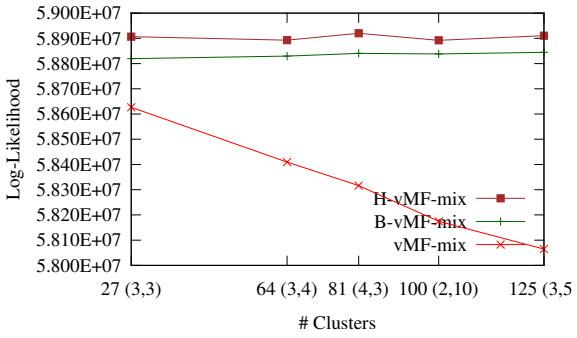
(c) News 20 Dataset



(d) CNAE Dataset



(e) K9 Dataset



(f) NIPS Dataset

9 Results of Temporal Bayesian methods vs other models

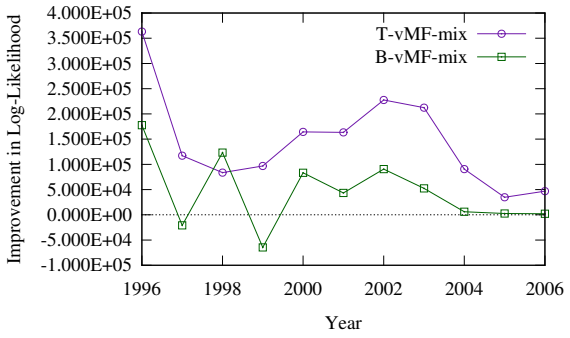
In this section, the results of the temporal Bayesian vMF mixture model (T-vMFmix) is compared with the simple vMF mixture model (vMFmix) as well as the non hierarchical Bayesian vMF mixture model (B-vMFmix). The T-vMFmix and B-vMFmix are trained through variational inference and vMFmix is trained through EM algorithm.
We used the 4 datasets which have associated with time-stamps,

1. NIPS-dataset : This dataset outlined in table 1 has 2483 research papers spread over a span of 17 years. The collection was divided into 17 time-points based on year of publication of each paper.

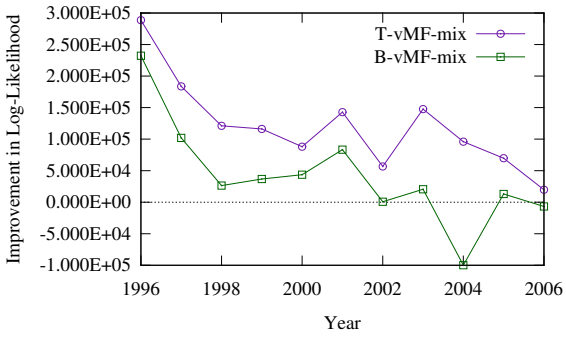
2. TDT-5 dataset: This dataset outlined in table 1 has 6636 news stories over a span of few months (Apr 03 to Sept 03). The collection was divided into 22 time-points where each time-point corresponds to one week of news. Note that we selected only those news stories which have been judged by humans to belong to one or more preselected news 'stories' [1].

3. NYTC-{politics,elections} datasets: This is a collection of New York Times news articles from the period 1996-2007. We formed two different datasets - the first one being more general than the second one. The first subset consist of all news articles which have been tagged as politics and the second subset consists of articles which have been tagged elections. For computational reasons, we removed all words which occurred less than 50 times and news stories which have less than 50 words. This leads to the nytc-politics dataset consisting of 48509 news articles with 16089 words and the nytc-elections dataset consisting of 23665 news articles and 10473 words.

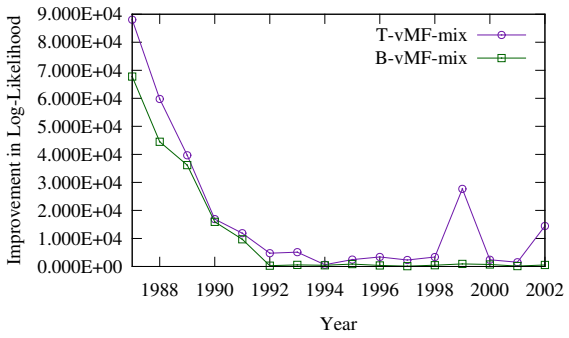
We test the methods by predicting the likelihood of the data of the next time-point given all the articles from the beginning using 30 topics. For ease of visualization, we plot the relative improvement in log-likelihood of the B-vMFmix and T-vMFmix models over the simple vMFmix model (This is because the log-likelihood from adjacent time-points are not comparable and fluctuate wildly).



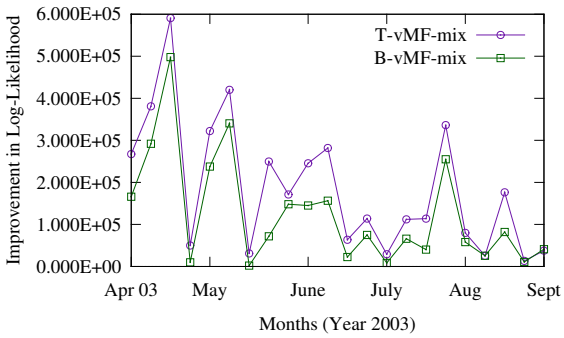
(a) NYTC-politics Dataset



(b) NYTC-elections Dataset



(a) NIPS Dataset



(b) TDT5 Dataset

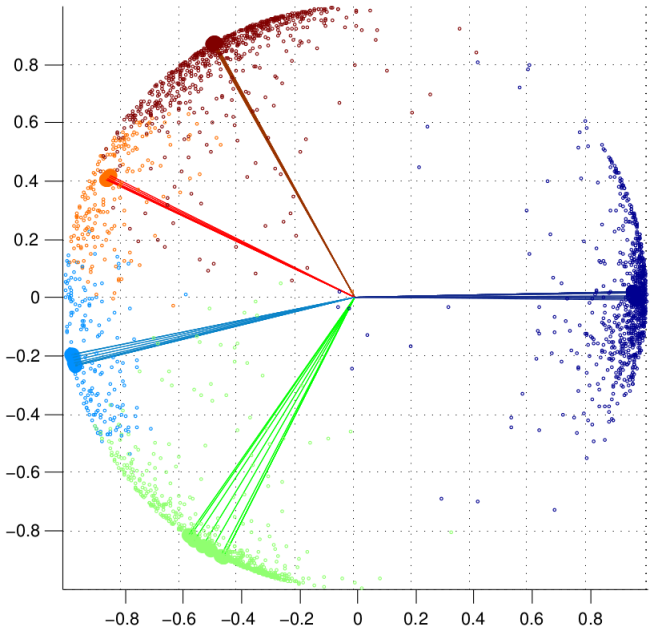
Lastly, we show some qualitative results using D-vMF-mix. Fig 5 shows the changing vocabulary for one of the topics in the NIPS dataset over time. The features with the largest weight are shown for every 3 years.

Figure 5: The change in vocabulary over the years for one of the topics from the NIPS dataset

(1987)	**(1991)**	**(1994)**	**(1997)**	**(2000)**	**(2003)**
excitatory	activity	activity	cortical	scene	scene
inhibitor	synaptic	cortical	visual	spikes	accessible
activity	firing	cortex	neuronal	spike	quantitative
synaptic	inhibitory	stimulus	orientation	quantitative	dissimilarity
firing	excitatory	neuronal	spatial	visual	fold
cells	neuron	response	tuning	stimuli	distinguish

For illustrative purposes, the represention of the clusters generated by D-vMF-mix (using 5 clusters) on a dimension reduced version of the NIPS dataset (Figure 6a) is shown. The data is first projected onto its first 3 principal components followed by a projection to a sphere. The different colors denote cluster membership and the colored lines from the center denotes how the mean parameter of each cluster moves over time.

(a) A 3D representation of the NIPS data



10 Other implementation Details

10.1 Calculating $I_\nu(a)$

We used the approximation given in [6],

$$\log I_\nu(a) = \sqrt{a^2 + (\nu+1)^2} + (\nu + \frac{1}{2}) \log \frac{a}{\nu + \frac{1}{2} + \sqrt{a^2 + (\nu+1)^2}} - \frac{1}{2}\log(\frac{a}{2}) + (\nu + \frac{1}{2}) \log \frac{2\nu + \frac{3}{2}}{2(\nu+1)} - \frac{1}{2}\log(2\pi)$$

$$I_\nu(a) = \exp(\log I_n u(a))$$

10.2 Experimental Settings

All the clustering experiments were run until one of the following condition was met

1. The increase in likelihood was less than 1e-1
2. The convergence in parameters was less than 1e-2
3. Atmost 200 iterations.

For sampling the concentration parameters, the proposal distribution was a log-normal distribution with mean as the current iterate and variance as 1 - The first 300 iterations were used for burn in and the last 200 iterations was used to estimate the samples (500 iterations seemed to be good enough). Since there are only K concentration parameters this does not affect the computational complexity at all.
Since all the algorithms are E-M based algorithms and the objective is not convex, there is certainly a possibility of local maxima. Therefore it is necessary to initialize the EM algorithm appropriately.
For all the vMF mix models we initialized the concentration parameters with relative low values like 5 or 10, which implies that the initial distribution of each cluster is very broad. This performed much better than setting the initial value of the concentration parameters to its MLE (the MLE value of κ was relatively high and makes the cluster distribution immediately very 'pointy' , therefore it is difficult for the instances to get reassigned to different clusters after the first iteration)

10.3 Computational Details

Given a fixed number of clusters K and N instances each with dimensionality D, most of the clustering algorithm iterate between two steps,

1. Assigning instances to clusters (either soft or hard).
2. Calculate cluster parameters.

In most cases, each step take $O(KNP)$ per iteration. If hard-assignments are used, one can '*potentially*' reduce the computation of step-2 by maintaining sufficient statistics which reduces the computation of step-2 to $O(NP)$. In any case the bottleneck is in step-1 i.e. deciding which cluster should the instance be assigned to (either partially or fully). For all the algorithms, this involves calculating a distance matrix. For instance consider the simple K-means algorithm; Given a data matrix $\mathbf{X}$ (dimension NxP) and cluster-means $\mathbf{M}$ (dimension KxP), step-1 involves the computation of,

$$\text{euclidean distance matrix } d(\mathbf{X}, \mathbf{M}) = ((\mathbf{X} \circ \mathbf{X})e_D)e_K^\top - 2\mathbf{X}\mathbf{M} + e_K((\mathbf{M} \circ \mathbf{M})e_D)$$
$$\text{where } e_D = \mathbf{1}^D \quad e_K = \mathbf{1}^K$$

Calculating the first term is $O(NP)$ and calculating the last term is $O(KP)$; the bottleneck is the computation of the middle term i.e. matrix-product $\mathbf{XM}$ which is $O(KNP)$. This is the general case for most of the clustering problems. Sometimes if we have a special structure in the distance metric we might be able to do

better - for e.g. if the distance is euclidean like above and we are doing hard assignments, then we can use triangle inequality to reduce the number of computations.
This matrix product is the bottleneck for 'most' clustering algorithm - for K-means and vMF, the bottleneck is the matrix-product $\mathbf{XM}$, for Multinomial mixture it is the matrix-product $\mathbf{X}log(\mathbf{M})$. For more complicated topic-models like LDA, it is even worse as each word in a document can belong to a different cluster.
Fortunately, people in the systems community have hammered the matrix multiplication over several decades. There are well known standards and widespread implementations that have squeezed even the last bit of performance. Depending on the data (sparse or dense) we have, the bottleneck will be one of the two things,

1. Multiplication of a dense matrix and dense matrix.
2. Multiplication of a sparse matrix and dense matrix.

The former is more widely studied than latter. Both are however popular enough to warrant independent functions in the Blas interface (Basic Linear Algebra Subprograms) - *gemm* and its respective sparse interface. In my limited experience I've found that Intel MKL library is THE fasted matrix multiplication out there - far better than my self-compiled atlas, clamdblas, gotoblas, openblas etc etc. It seamlessly harnessed how much ever cores required (in a multicore machine) without dropping performance - something that other tools struggled to do. For e.g. clamdblas just opened too many threads for their own good.

Using such parallel matrix library blurs the computation time between different algorithms. As long as the distance and parameter calculations can be written in terms of matrix products, all the clustering algorithms can run equally fast. In our experiments, we did not find any noticeable difference in running time **per iteration** between the various clustering algorithms.

Bibliography

[1] James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study final report. 1998.

[2] Arindam Banerjee, Inderjit S Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *JMLR*, 6(2):1345, 2006.

[3] Mark Bangert, Philipp Hennig, and Uwe Oelfke. Using an infinite von mises-fisher mixture model to cluster treatment beam directions in external radiation therapy. In *ICMLA*, 2010.

[4] Árpád Baricz, Saminathan Ponnusamy, and Matti Vuorinen. Functional inequalities for modified bessel functions. *Expositiones Mathematicae*, 29(4):399–414, 2011.

[5] George Casella. Empirical bayes gibbs sampling. *Biostatistics*, 2(4):485–500, 2001.

[6] Kurt Hornik and Bettina Grün. movmf: An r package for fitting mixtures of von mises-fisher distributions.

[7] Suvrit Sra. A short note on parameter approximation for von mises-fisher distributions: and a fast implementation of i s (x). *Computational Statistics*, 27(1):177–190, 2012.