Package 'movMF'

October 8, 2021

Version 0.2-6

Encoding UTF-8

Title Mixtures of von Mises-Fisher Distributions

Description Fit and simulate mixtures of von Mises-Fisher distributions.

Depends R (>= 3.0.0)

Imports skmeans (>= 0.2-10), clue, slam (>= 0.1-43), stats

Suggests lattice, tm (>= 0.7-5), vcd, SnowballC, HSAUR3, colorspace, corpus.useR.2008.abstracts, flexmix (>= 2.3-17), methods

Additional_repositories https://datacube.wu.ac.at

License GPL-2

NeedsCompilation yes

Author Kurt Hornik [aut, cre] (<https://orcid.org/0000-0003-4198-9911>), Bettina Grün [aut] (<https://orcid.org/0000-0001-7265-4773>)

Maintainer Kurt Hornik <Kurt.Hornik@R-project.org>

Repository CRAN

Date/Publication 2021-10-08 16:20:33 UTC

R topics documented:

FLXMCvMF *Flexmix Driver for Mixtures of von Mises-Fisher Distributions*

Description

This driver for `flexmix` implements estimation of mixtures of von Mises-Fisher distributions where the data can be stored in a dense or a simple triplet matrix (package **slam**) format.

Usage

```
FLXMCvMF(formula = . ~ ., kappa = NULL)
```

Arguments

`formula`	a formula which is interpreted relative to the formula specified in the call to `flexmix` using `update.formula`. Only the left-hand side (response) of the formula is used. Default is to use the original `flexmix` model formula.
`kappa`	see the `control` argument of `movMF`.

Value

An object of class `"FLXMCvMF"`.

Author(s)

Bettina Grün

Examples

```
if (requireNamespace("flexmix", quietly = TRUE)) {
## Generate and fit a "small-mix" data set a la Banerjee et al.
mu <- rbind(c(-0.251, -0.968),
            c(0.399, 0.917))
kappa <- c(4, 4)
theta <- kappa * mu
theta
alpha <- c(0.48, 0.52)
## Generate a sample of size n = 50 from the von Mises-Fisher mixture
## with the above parameters.
set.seed(123)
x <- rmovMF(50, theta, alpha)
## Fit a von Mises-Fisher mixture with the "right" number of components,
## using 10 EM runs.
set.seed(123)
y2 <- flexmix::stepFlexmix(x ~ 1, k = 2, model = FLXMCvMF(), verbose = FALSE)
## Inspect the fitted parameters:
y2
## Compare the fitted classes to the true ones:
table(True = attr(x, "z"), Fitted = flexmix::clusters(y2))
```

```
## To use a common kappa:
y2cv <- flexmix::stepFlexmix(x ~ 1, k = 2,
  model = FLXMCvMF(kappa = list(common = TRUE)), verbose = FALSE)
## To use a common kappa fixed to the true value of 4:
y2cf <- flexmix::stepFlexmix(x ~ 1, k = 2,
  model = FLXMCvMF(kappa = 4), verbose = FALSE)
## Comparing solutions via BIC:
sapply(list(y2, y2cf, y2cv), BIC)
##  Use a different kappa solver:
set.seed(123)
y2a <- flexmix::stepFlexmix(x ~ 1, k = 2,
  model = FLXMCvMF(kappa = "uniroot"), verbose = FALSE)
y2a
## Using a sparse matrix:
x <- slam::as.simple_triplet_matrix(x)
y2 <- flexmix::stepFlexmix(x ~ 1, k = 2,
  model = FLXMCvMF(), verbose = FALSE)
}
```

---

movMF *Fit Mixtures of von Mises-Fisher Distributions*

Description

Fit mixtures of von Mises-Fisher distributions.

Usage

```
movMF(x, k, control = list(), ...)
```

Arguments

x	a numeric data matrix, with rows corresponding to observations. Standardized to unit row lengths if necessary. Can be a dense matrix, a simple triplet matrix (package **slam**), or a dgTMatrix (package **Matrix**).
k	an integer giving the desired number of mixture components (classes).
control	a list of control parameters. See **Details**.
...	a list of control parameters (overriding those specified in control).

Details

movMF returns an object of class "movMF" representing the fitted mixture of von Mises-Fisher distributions model. Available methods for such objects include coef, logLik, print and predict. predict has an extra type argument with possible values "class_ids" (default) and "memberships" for indicating hard or soft prediction, respectively.

The mixture of von Mises-Fisher distributions is fitted using EM variants as specified by control option E (specifying the E-step employed), with possible values "softmax" (default), "hardmax" or

"stochmax" where the first two implement algorithms soft-moVMF and hard-moVMF of Banerjee et al (2005). For "stochmax", class assignments are drawn from the posteriors for each observation in the E-step as outlined as SEM in Celeux and Govaert (1992). The stopping criterion for this algorithm is by default changed to not check for convergence (logical control option converge), but to return the parameters with the maximum likelihood encountered. E may be abbreviated.

In the M-step, the parameters θ of the respective component distributions are estimated via maximum likelihood, which is accomplished by taking θ proportional to suitable weighted sample means $\bar{x}$, with length κ solving the equation $A_d(\kappa) = \|\bar{x}\|$, where $A_d(\kappa) = I_{d/2}(\kappa)/I_{d/2-1}(\kappa)$ with I the modified Bessel function of the first kind. Via control argument kappa, one can specify how to (approximately) solve these equations, and whether a common (possibly given) length κ should be employed. If kappa is a number, it gives a common length to be employed. If it is a character string, it specifies the method to be used for solving the κ equation. The possible methods are:

"Banerjee_et_al_2005" uses the approximation of Banerjee et al (2005).

"Tanabe_et_al_2007" uses the fixed-point iteration of Tanabe et al (2007) with starting point for κ in the interval established by Tanabe et al (2007) implied by a given c with values in [0, 2]. The default is c = 1, the mid-point of the interval.

"Sra_2012" uses two Newton steps as suggested in Sra (2012) starting in the approximation of Banerjee et al (2005).

"Song_et_al_2012" uses two Halley steps as suggested in Song et al (2012) starting in the approximation of Banerjee et al (2005).

"uniroot" uses a straightforward call to uniroot with the bounds established in Hornik and Grün (2014).

"Newton" uses a full Newton algorithm started in the approximation of Hornik and Grün (2014).

"Halley" uses a full Halley algorithm started in the approximation of Hornik and Grün (2014).

"hybrid" implements a combination of a derivative-based step (Newton or Halley) and a bisection step as outlined in Press et al. (2002). The derivative-based step can be specified via the argument step which expects a function performing this step. Currently step_Newton and step_Halley (default) are available.

"Newton_Fourier" **(default)** uses a variant of the Newton-Fourier method for strictly increasing concave functions as for example given in Atkinson (1989, pp. 62–64). Concavity can be established using Hornik and Grün (2013).

The lower-cased version of the given kappa specification is matched against the lower-cased names of the available methods using pmatch. Finally, to indicate using a common (but not given) κ for all component distributions, kappa should be a list with element common = TRUE (and optionally a character string giving the estimation method).

Additional control parameters are as follows.

maxiter an integer giving the maximal number of EM iterations to be performed. Default: 100.

reltol the minimum relative improvement of the objective function per iteration. If improvement is less, the EM algorithm will stop under the assumption that no further significant improvement can be made. Defaults to sqrt(.Machine$double.eps).

ids either a vector of class memberships or TRUE which implies that the class memberships are obtained from the attribute named "z" of the input data; these class memberships are used for initializing the EM algorithm and the algorithm is stopped after the first iteration.

`start` a specification of the starting values to be employed. Can be a list of matrices giving the memberships of objects to components, or of vectors giving component ids (numbers from 1 to the given `k`). Can also be a character vector with elements `"i"` (randomly pick component ids for the observations), or one of `"p"`, `"S"` or `"s"`. The latter first determine component "prototypes", and then determine an optimal "fuzzy" membership matrix from the implied cosine dissimilarities between observations and prototypes. Prototypes are obtained as follows: for `"p"`, observations are randomly picked. For `"S"`, one takes the first prototype to minimize total cosine dissimilarity to the observations, and then successively picks observations farthest away from the already picked prototypes. For `"s"`, one takes a randomly chosen observation as the first prototype, and then proceeds as for `"S"`.

By default, initialization method `"p"` is used.

If several starting values are specified, the EM algorithm is performed individually to each starting value, and the best solution found is returned.

`nruns` an integer giving the number of EM runs to be performed. Default: 1. Only used if `start` is not given.

`minalpha` a numeric indicating the minimum prior probability. Components falling below this threshold are removed during the iteration. If ≥ 1, this is taken as the minimal number of observations in a component. Default: 0.

`converge` a logical, if `TRUE` the EM algorithm is stopped if the `reltol` criterion is met and the current parameter estimate is returned. If `FALSE` the EM algorithm is run for `maxiter` iterations and the parametrizations with the maximum likelihood encountered during the EM algorithm is returned. Default: `TRUE`, changed to `FALSE` if `E="stochmax"`.

`verbose` a logical indicating whether to provide some output on algorithmic progress. Defaults to `getOption("verbose")`.

One popular application context of mixtures of von Mises-Fisher distributions is text mining, where the data matrices are typically very large and sparse. The provided implementation should be able to handle such large corpora with reasonable efficiency by employing suitable sparse matrix representations and computations. In addition, straightforward computations of the normalizing constants in the von Mises-Fisher densities (see movMF_distribution) by directly employing the modified Bessel functions of the first kind are computationally infeasible for large d (dimension of the observations) and/or values of the parameter lengths κ. Instead, we use suitably scaled hypergeometric-type power series for computing (the logarithms of) the normalizing constants.

Value

An object of class `"movMF"` representing the fitted mixture of von Mises-Fisher distributions, which is a list containing at least the following components:

`theta`	a matrix with rows giving the fitted parameters of the mixture components.
`alpha`	a numeric vector with the fitted mixture probabilities.

See vMF for the employed parametrization of the von Mises-Fisher distribution.

References

K. E. Atkinson (1989). *An Introduction to Numerical Analysis.* 2nd edition. John Wiley & Sons.

A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra (2005). Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research*, **6**, 1345–1382. https://jmlr.csail.mit.edu/papers/v6/banerjee05a.html.

G. Celeux, and G. Govaert (1992). A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics & Data Analysis*, **14**, 315–332. doi: 10.1016/01679473(92)90042-E.

K. Hornik, and B. Grün (2013). Amos-type bounds for modified Bessel function ratios. *Journal of Mathematical Analysis and Applications*, **408**(1), 91–101. doi: 10.1016/j.jmaa.2013.05.070.

K. Hornik, and B. Grün (2014). On maximum likelihood estimation of the concentration parameter of von Mises-Fisher distributions. *Computational Statistics*, **29**, 945–957. doi: 10.1007/s00180-01304710.

W. H. Press, S. A. Teukolsky, W. T. Vetterling and Brian P. Flannery (2002). *Numerical Recipes in C: The Art of Scientific Computing.* 2nd edition. Cambridge University Press.

H. Song, J. Liu, and G. Wang. High-order parameter approximation for von Mises-Fisher distributions. *Applied Mathematics and Computation*, **218**, 11880–11890. doi: 10.1016/j.amc.2012.05.050.

S. Sra (2012). A short note on parameter approximation for von Mises-Fisher distributions: and a fast implementation of $I_s(x)$. *Computational Statistics*, **27**, 177–190. doi: 10.1007/s00180011-0232x.

A. Tanabe, K. Fukumizu, S. Oba, T. Takenouchi, and S. Ishii. Parameter estimation for von Mises-Fisher distributions. *Computational Statistics*, **22**, 145–157. doi: 10.1007/s0018000700307.

Examples

```
## Generate and fit a "small-mix" data set a la Banerjee et al.
mu <- rbind(c(-0.251, -0.968),
            c(0.399, 0.917))
kappa <- c(4, 4)
theta <- kappa * mu
theta
alpha <- c(0.48, 0.52)
## Generate a sample of size n = 50 from the von Mises-Fisher mixture
## with the above parameters.
set.seed(123)
x <- rmovMF(50, theta, alpha)
## Fit a von Mises-Fisher mixture with the "right" number of components,
## using 10 EM runs.
set.seed(123)
y2 <- movMF(x, 2, nruns = 10)
## Inspect the fitted parameters:
y2
## Compare the fitted classes to the true ones:
table(True = attr(x, "z"), Fitted = predict(y2))
## To use a common kappa:
y2cv <- movMF(x, 2, nruns = 10, kappa = list(common = TRUE))
## To use a common kappa fixed to the true value of 4:
y2cf <- movMF(x, 2, nruns = 10, kappa = 4)
## Comparing solutions via BIC:
sapply(list(y2, y2cf, y2cv), BIC)
##  Use a different kappa solver:
```

```
set.seed(123)
y2a <- movMF(x, 2, nruns = 10, kappa = "uniroot")
y2a
```

`movMF_distribution` *Mixtures of von Mises-Fisher Distributions*

Description

Density and random number generation for finite mixtures of von Mises-Fisher distributions.

Usage

```
dmovMF(x, theta, alpha = 1, log = FALSE)
rmovMF(n, theta, alpha = 1)
```

Arguments

`x`	a matrix of rows of points on the unit hypersphere. Standardized to unit row length if necessary.
`theta`	a matrix with rows giving the parameters of the mixture components.
`alpha`	a numeric vector with non-negative elements giving the mixture probabilities. Standardized to sum to one if necessary.
`log`	a logical; if `TRUE` log-densities are computed.
`n`	an integer giving the number of samples to draw.

Details

A random d-dimensional unit length vector x has a von Mises-Fisher (or Langevin, short: vMF) distribution with parameter θ if its density with respect to the uniform distribution on the unit hypersphere is given by

$$f(x|\theta) = \exp(\theta' x)/{}_0F_1(;d/2;\|\theta\|^2/4),$$

where ${}_0F_1$ is a generalized hypergeometric function (e.g., https://en.wikipedia.org/wiki/Generalized_hypergeometric_function) and related to the modified Bessel function I_ν of the first kind via

$${}_0F_1(;\nu+1;z^2/4) = I_\nu(z)\Gamma(\nu+1)/(z/2)^\nu.$$

With this parametrization, the von Mises-Fisher family is the natural exponential family through the uniform distribution on the unit sphere, with cumulant transform

$$M(\theta) = \log({}_0F_1(;d/2;\|\theta\|^2/4)).$$

We note that the vMF distribution is commonly parametrized by the *mean direction parameter* $\mu = \theta/\|\theta\|$ (which however is not well-defined if $\theta = 0$) and the *concentration parameter* $\kappa = \|\theta\|$, e.g., https://en.wikipedia.org/wiki/Von_Mises%E2%80%93Fisher_distribution (which also uses the un-normalized Haar measure on the unit sphere as the reference distribution, and hence includes the "area" of the unit sphere as an additional normalizing constant).

dmovMF computes the (log) density of mixtures of vMF distributions.

rmovMF generates samples from finite mixtures of vMF distributions, using Algorithm VM* in Wood (1994) for sampling from the vMF distribution.

Arguments theta and alpha are recycled to a common number of mixture components.

Value

For dmovMF, a numeric vector of (log) density values.

For rmovMF, a matrix with n unit length rows representing the samples from the vMF mixture distribution.

References

A. T. A. Wood (1994). Simulation of the von Mises Fisher distribution. *Communications in Statistics – Simulation and Computation*, **23**(1), 157–164.

Examples

```
## To simulate from the vMF distribution with mean direction
## proportional to c(1, -1) and concentration parameter 3:
rmovMF(10, 3 * c(1, -1) / sqrt(2))
## To simulate from a mixture of vMF distributions with mean direction
## parameters c(1, 0) and c(0, 1), concentration parameters 3 and 4, and
## mixture probabilities 1/3 and 2/3, respectively:
rmovMF(10, c(3, 4) * rbind(c(1, 0), c(0, 1)), c(1, 2))
```

Index