Package: mixturemodelsr (via r-universe)

May 31, 2026

Type Package

Title R Wrapper for the Mixture-Models Python Library

Version 0.1.20

Description First R package enabling mixture models for high-dimensional data through gradient-based optimization with Automatic Differentiation (AD). Provides an R interface to the 'Mixture-Models' Python package (Kasa et al., 2024) via reticulate. Unlike traditional EM-based approaches (e.g., mclust, flexmix), this package uses AD and gradient-based optimization (including second-order Newton-CG) to fit Gaussian Mixture Models (GMM), Mixture of Factor Analyzers (MFA), Parsimonious GMM (PGMM), MCLUST family, and t-mixture models without requiring stringent modeling constraints, making it suitable for high-dimensional settings where the number of parameters exceeds the sample size. Reference: Kasa, S. R., Yijie, H., Kasa, S. K., & Rajan, V. (2024). Mixture-Models: a one-stop Python Library for Model-based Clustering using various Mixture Models. arXiv preprint arXiv:2402.10229.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.6.0)

Imports reticulate (>= 1.14)

Suggests testthat (>= 3.0.0), knitr, rmarkdown

SystemRequirements Python (>= 3.6), Python package 'mixture-models' (available on PyPI)

URL https://github.com/kasakh/mixturemodelsr, https://github.com/kasakh/Mixture-Models

BugReports https://github.com/kasakh/mixturemodelsr/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs libpng-dev python3

Repository https://kasakh.r-universe.dev

Date/Publication 2026-01-01 00:33:16 UTC

RemoteUrl https://github.com/kasakh/mixturemodelsr

RemoteRef HEAD

RemoteSha bf21e3d062d01e7f854d7f9197eec0d6e559147d

Contents

`mm_aic` *Compute AIC for fitted model*

Description

Compute AIC for fitted model

Usage

```
mm_aic(fit)
```

Arguments

`fit`	An mm_fit object

Value

Numeric AIC value

Examples

```
## Not run:
fit <- mm_gmm_fit(iris[,1:4], k = 3)
mm_aic(fit)

## End(Not run)
```

`mm_bic` *Compute BIC for fitted model*

Description

Compute BIC for fitted model

Usage

```
mm_bic(fit)
```

Arguments

`fit`	An mm_fit object

Value

Numeric BIC value

Examples

```
## Not run:
fit <- mm_gmm_fit(iris[,1:4], k = 3)
mm_bic(fit)

## End(Not run)
```

`mm_gmm_constrained_fit` *Fit a Constrained Gaussian Mixture Model*

Description

Fits a GMM with a common covariance matrix across all components.

Usage

```
mm_gmm_constrained_fit(
  x,
  k,
  optimizer = "Newton-CG",
  scale = 1,
  use_kmeans = TRUE,
  ...
)
```

Arguments

`x`	Numeric matrix or data.frame (rows = observations, columns = features)
`k`	Number of mixture components
`optimizer`	Optimizer name: "Newton-CG" (default), "grad_descent", "rms_prop", or "adam"
`scale`	Initialization scale parameter (default 1.0)
`use_kmeans`	Logical, whether to use k-means for initializing component means (default TRUE)
`...`	Additional arguments passed to Python init_params() or fit() methods

Value

An mm_fit object

Examples

```
## Not run:
fit <- mm_gmm_constrained_fit(iris[, 1:4], k = 3)
mm_bic(fit)

## End(Not run)
```

mm_gmm_fit *Fit a Gaussian Mixture Model (GMM)*

Description

Fits a Gaussian mixture model to data using gradient-based optimization. This is a wrapper around the Python Mixture-Models GMM implementation.

Usage

```
mm_gmm_fit(x, k, optimizer = "Newton-CG", scale = 1, use_kmeans = TRUE, ...)
```

Arguments

x	Numeric matrix or data.frame (rows = observations, columns = features)
k	Number of mixture components
optimizer	Optimizer name: "Newton-CG" (default), "grad_descent", "rms_prop", or "adam"
scale	Initialization scale parameter (default 1.0)
use_kmeans	Logical, whether to use k-means for initializing component means (default TRUE)
...	Additional arguments passed to Python init_params() or fit() methods

Value

An mm_fit object containing:

py_model	Python model object
params_store	Full optimization path
params	Final fitted parameters
model_name	Model family name
k	Number of components
call	Original function call
n	Sample size
d	Number of dimensions
optimizer	Optimizer used

Examples

```
## Not run:
# Setup (first time only)
mm_setup()

# Fit a 3-component GMM on iris data
fit <- mm_gmm_fit(iris[, 1:4], k = 3)
print(fit)
```

```
# Get cluster labels
labels <- mm_predict(fit)

# Model selection
mm_bic(fit)
mm_aic(fit)

## End(Not run)
```

`mm_likelihood` *Compute log-likelihood for fitted model*

Description

Compute log-likelihood for fitted model

Usage

```
mm_likelihood(fit)
```

Arguments

`fit`	An mm_fit object

Value

Numeric log-likelihood value

Examples

```
## Not run:
fit <- mm_gmm_fit(iris[,1:4], k = 3)
mm_likelihood(fit)

## End(Not run)
```

`mm_mclust_fit` *Fit an MCLUST Family Model*

Description

Fits a model from the MCLUST family of constrained Gaussian mixture models. MCLUST models specify different parameterizations of the covariance structure.

Usage

```
mm_mclust_fit(
  x,
  k,
  model_type = NULL,
  optimizer = "Newton-CG",
  scale = 1,
  use_kmeans = TRUE,
  ...
)
```

Arguments

x	Numeric matrix or data.frame (rows = observations, columns = features)
k	Number of mixture components
model_type	Character string specifying the MCLUST model type. Common types include: "EII", "VII", "EEI", "VEI", "EVI", "VVI", "EEE", "VEE", "EVE", "VVE", "EEV", "VEV", "EVV", "VVV" If NULL, uses default from Python implementation.
optimizer	Optimizer name: "Newton-CG" (default), "grad_descent", "rms_prop", or "adam"
scale	Initialization scale parameter (default 1.0)
use_kmeans	Logical, whether to use k-means for initializing component means (default TRUE)
...	Additional arguments passed to Python init_params() or fit() methods

Details

MCLUST model types follow a three-letter naming convention:

- First letter: Volume (E=equal across components, V=variable across components)
- Second letter: Shape (E=equal, V=variable, I=spherical/identity)
- Third letter: Orientation (E=equal, V=variable, I=axis-aligned/identity)

Common model types:

- "EII": Spherical, equal volume
- "VII": Spherical, variable volume
- "EEE": Ellipsoidal, equal volume, shape, and orientation
- "VVV": Ellipsoidal, variable volume, shape, and orientation (most flexible)

Value

An mm_fit object

Examples

```
## Not run:
# Fit MCLUST with VVV (most flexible) model
fit <- mm_mclust_fit(iris[, 1:4], k = 3, model_type = "VVV")
mm_bic(fit)

# Fit MCLUST with spherical, equal volume model
fit_eii <- mm_mclust_fit(iris[, 1:4], k = 3, model_type = "EII")
mm_bic(fit_eii)

# Compare models
labels <- mm_predict(fit)
table(labels, iris$Species)

## End(Not run)
```

`mm_mfa_fit` *Fit a Mixture of Factor Analyzers (MFA)*

Description

Fits a mixture of factor analyzers model to data using gradient-based optimization.

Usage

```
mm_mfa_fit(
  x,
  k,
  q = NULL,
  optimizer = "Newton-CG",
  scale = 1,
  use_kmeans = TRUE,
  ...
)
```

Arguments

`x`	Numeric matrix or data.frame (rows = observations, columns = features)
`k`	Number of mixture components
`q`	Number of latent factors (NULL for automatic selection)
`optimizer`	Optimizer name: "Newton-CG" (default), "grad_descent", "rms_prop", or "adam"
`scale`	Initialization scale parameter (default 1.0)
`use_kmeans`	Logical, whether to use k-means for initializing component means (default TRUE)
`...`	Additional arguments passed to Python init_params() or fit() methods

Value

An mm_fit object

Examples

```
## Not run:
# Fit MFA with 3 components and 2 latent factors
fit <- mm_mfa_fit(iris[, 1:4], k = 3, q = 2)
mm_bic(fit)

# Get cluster labels
labels <- mm_predict(fit)

## End(Not run)
```

---

mm_params *Get parameter values from fit object*

Description

Extracts the parameter values from a fitted mixture model.

Usage

```
mm_params(fit, convert = TRUE)
```

Arguments

fit	An mm_fit object
convert	Logical, whether to convert Python objects to R (default TRUE)

Value

Parameter object (structure depends on model family)

Examples

```
## Not run:
fit <- mm_gmm_fit(iris[,1:4], k = 3)
params <- mm_params(fit)

## End(Not run)
```

`mm_pgmm_fit` *Fit a Parsimonious Gaussian Mixture Model (PGMM)*

Description

Fits a parsimonious GMM with constraints on the covariance structure.

Usage

```
mm_pgmm_fit(
  x,
  k,
  model_type = NULL,
  q = NULL,
  optimizer = "Newton-CG",
  scale = 1,
  use_kmeans = TRUE,
  ...
)
```

Arguments

`x`	Numeric matrix or data.frame (rows = observations, columns = features)
`k`	Number of mixture components
`model_type`	Character string specifying the PGMM model type (e.g., "VVV", "EEE", "VEV"). If NULL, uses default from Python implementation.
`q`	Number of latent factors (NULL for automatic selection)
`optimizer`	Optimizer name: "Newton-CG" (default), "grad_descent", "rms_prop", or "adam"
`scale`	Initialization scale parameter (default 1.0)
`use_kmeans`	Logical, whether to use k-means for initializing component means (default TRUE)
`...`	Additional arguments passed to Python init_params() or fit() methods

Details

PGMM model types follow the mclust naming convention:

- First letter: Volume (E=equal, V=variable)
- Second letter: Shape (E=equal, V=variable)
- Third letter: Orientation (E=equal, V=variable, I=identity)

Value

An mm_fit object

Examples

```
## Not run:
# Fit PGMM with variable volume, shape, and orientation
fit <- mm_pgmm_fit(iris[, 1:4], k = 3, model_type = "VVV")
mm_bic(fit)

## End(Not run)
```

`mm_predict` *Predict cluster labels*

Description

Predicts cluster membership labels for observations using a fitted mixture model.

Usage

```
mm_predict(fit, newx = NULL)
```

Arguments

`fit`	An mm_fit object returned by mm_*_fit functions
`newx`	Optional matrix or data.frame of new observations. If NULL, predictions are made on the training data.

Value

Integer vector of cluster labels (0-indexed from Python, converted to 1-indexed for R)

Examples

```
## Not run:
fit <- mm_gmm_fit(iris[,1:4], k = 3)
labels <- mm_predict(fit)
table(labels, iris$Species)

## End(Not run)
```

mm_py_info *Python Configuration Diagnostics*

Description

Prints detailed information about the Python environment configuration for mixturemodelsr. Useful for troubleshooting installation issues.

Usage

```
mm_py_info()
```

Value

Invisibly returns a list with diagnostic information

Examples

```
## Not run:
mm_py_info()

## End(Not run)
```

mm_python_available *Check if Python module is available*

Description

Check if Python module is available

Usage

```
mm_python_available()
```

Value

Logical indicating if either mixture_models or Mixture_Models is importable

mm_setup *Setup mixturemodelsr (user-friendly wrapper)*

Description

One-time setup for R users. Provisions a dedicated conda environment with Python 3.10 + NumPy 1.23.5 and installs Mixture-Models==0.0.8.

Usage

```
mm_setup(force = FALSE)
```

Arguments

force	Logical, force reinstallation by deleting and recreating the env

Value

TRUE invisibly on success

mm_tmm_fit *Fit a t-Mixture Model (TMM)*

Description

Fits a mixture of multivariate t-distributions to data using gradient-based optimization. T-distributions are more robust to outliers than Gaussian distributions.

Usage

```
mm_tmm_fit(x, k, optimizer = "Newton-CG", scale = 1, use_kmeans = TRUE, ...)
```

Arguments

x	Numeric matrix or data.frame (rows = observations, columns = features)
k	Number of mixture components
optimizer	Optimizer name: "Newton-CG" (default), "grad_descent", "rms_prop", or "adam"
scale	Initialization scale parameter (default 1.0)
use_kmeans	Logical, whether to use k-means for initializing component means (default TRUE)
...	Additional arguments passed to Python init_params() or fit() methods

Details

T-mixture models use multivariate t-distributions instead of Gaussians, making them more robust to outliers. Each component has its own degrees of freedom parameter, allowing different tail behaviors.

Value

An mm_fit object

Examples

```
## Not run:
# Fit TMM with 3 components (robust to outliers)
fit <- mm_tmm_fit(iris[, 1:4], k = 3)
mm_bic(fit)

# Get cluster labels
labels <- mm_predict(fit)
table(labels, iris$Species)

## End(Not run)
```

---

print.mm_fit *Print method for mm_fit objects*

Description

Print method for mm_fit objects

Usage

```
## S3 method for class 'mm_fit'
print(x, ...)
```

Arguments

x	An mm_fit object
...	Additional arguments (ignored)

Index