Package 'mixdir'

September 20, 2019

Type Package

Title Cluster High Dimensional Categorical Datasets

Version 0.3.0

Description Scalable Bayesian clustering of categorical datasets. The package implements a hierarchical Dirichlet
(Process) mixture of multinomial distributions. It is thus a probabilistic latent class model (LCM) and can be used
to reduce the dimensionality of hierarchical data and cluster individuals into latent classes. It can automatically
infer an appropriate number of latent classes or find k classes, as defined by the user. The model is based on a
paper by Dunson and Xing (2009) <doi:10.1198/jasa.2009.tm08439>, but implements a scalable variational inference algorithm so that it is
applicable to large datasets. It is described and tested in the accompanying paper by
Ahlmann-Eltze and Yau (2018) <doi:10.1109/DSAA.2018.00068>.

URL https://github.com/const-ae/mixdir

License GPL-3

Encoding UTF-8

LazyData true

Suggests testthat, tibble, purrr, dplyr, rmutil, pheatmap, mcclust,
ggplot2, tidyr, utils

RoxygenNote 6.1.1

Imports extraDistr, Rcpp

Depends R (>= 2.10)

LinkingTo Rcpp

NeedsCompilation yes

Author Constantin Ahlmann-Eltze [aut, cre]
(<https://orcid.org/0000-0002-3762-068X>),
Christopher Yau [ths] (<https://orcid.org/0000-0001-7615-8523>)

Maintainer Constantin Ahlmann-Eltze <artjom31415@googlemail.com>

Repository CRAN

Date/Publication 2019-09-20 15:10:05 UTC

R topics documented:

`find_defining_features` *Find the n defining features*

Description

Reduce the dimensionality of a dataset by calculating how important each feature is for inferring the clustering.

Usage

```
find_defining_features(mixdir_obj, X, n_features = Inf,
  measure = c("JS", "ARI"), subsample_size = Inf, step_size = Inf,
  exponential_decay = TRUE, verbose = FALSE)
```

Arguments

`mixdir_obj`	the result from a call to `mixdir()`. It needs to have the fields category_prob. category_prob a list of a list of a named vector with probabilities for each feature, latent class and possible category.
`X`	the original dataset that was used for clustering.
`n_features`	the number of dimensions that should be selected. If it is `Inf` (the default) all features are returned ordered by importance (most important first).
`measure`	The measure used to assess the loss of clustering quality if a variable is removed. Two measures are implemented: "JS" short for Jensen-Shannon divergence comparing the original class probabilities and the new predicted class probabilities (smaller is better), "ARI" short for adjusted Rand index compares the overlap of the original and the predicted classes (requires the `mcclust` package) (1 is perfect, 0 is as good as random).
`subsample_size`	Running this method on the full dataset can be slow, but one can easily speed up the calculation by randomly selecting a subset of rows from X without usually disproportionately hurting the selection performance.

step_size The method can either remove each feature individually and return the n features that caused the greatest quality loss (step=Inf) or iteratively remove the least important one until the the size of the remaining features equal n_features (step=1). Using a smaller step size increases the sensitivity of the selection process, but takes longer to calculate.

exponential_decay
Boolean or number. Alternative way of calculating how many features to remove each step. The default is to always remove the least important 50% of the features (exponential_decay=2).

verbose Boolean indicating if status messages should be printed.

Details

Iteratively find the variable, whose removal least affects the clustering compared with the original. If n_features is a finite number the quality is a single number and reflects how good those n features maintain the original clustering. If n_features=Inf, the method returns all features ordered by decreasing importance. The accompanying quality vector contains the "cumulative" loss if the corresponding variable would be removed. Note that depending on the step size scheme the quality can differ. For example if all variables are removed in one step (step_size=Inf and exponential_decay=FALSE) the quality is not cumulative, but simply the quality of the clustering excluding the corresponding feature. In that sense the quality vector should not be used as a definitive answer, but should only be used as a guidance to see where there are jumps in the quality.

See Also

find_predictive_features find_typical_features

Examples

```
data("mushroom")
res <- mixdir(mushroom[1:100, ], n_latent=20)
find_defining_features(res, mushroom[1:100, ], n_features=3)
find_defining_features(res, mushroom[1:100, ], n_features=Inf)
```

---

find_predictive_features

Find the top predictive features and values for each latent class

---

Description

Find the top predictive features and values for each latent class

Usage

```
find_predictive_features(mixdir_obj, top_n = 10)
```

Arguments

mixdir_obj	the result from a call to mixdir(). It needs to have the fields lambda and category_prob. lambda a vector of probabilities for each category. category_prob a list of a list of a named vector with probabilities for each feature, latent class and possible category.
top_n	the number of top answers per category that will be returned. Default: 10.

Value

A data frame with four columns: column, answer, class and probability. The probability column contains the chance that an observation belongs to the latent class if all that is known about that observation that `column`=`category`

See Also

find_typical_features find_defining_features

Examples

```
data("mushroom")
res <- mixdir(mushroom[1:30, ], beta=1)
find_predictive_features(res, top_n=3)
```

---

find_typical_features *Find the most typical features and values for each latent class*

Description

Find the most typical features and values for each latent class

Usage

```
find_typical_features(mixdir_obj, top_n = 10)
```

Arguments

mixdir_obj	the result from a call to mixdir(). It needs to have the fields lambda and category_prob. lambda a vector of probabilities for each category. category_prob a list of a list of a named vector with probabilities for each feature, latent class and possible category.
top_n	the number of top answers per category that will be returned. Default: 10.

Value

A data frame with four columns: column, answer, class and probability. The probability column contains the chance to see the answer in that column.

See Also

find_predictive_features find_defining_features

Examples

```
data("mushroom")
res <- mixdir(mushroom[1:30, ], beta=1)
find_typical_features(res, top_n=3)
```

---

mixdir *Cluster high dimensional categorical datasets*

Description

Cluster high dimensional categorical datasets

Usage

```
mixdir(X, n_latent = 3, alpha = NULL, beta = NULL,
  select_latent = FALSE, max_iter = 100, epsilon = 0.001,
  na_handle = c("ignore", "category"), repetitions = 1, ...)
```

Arguments

X	A matrix or data.frame of size (N_ind x N_quest) that contains the categorical responses. The values can be characters, integers or factors. The most flexibility is provided if factors are used.
n_latent	The number of latent factors that are used to approximate the model. Default: 3.
alpha	A single number or a vector of two numbers in case select_latent=TRUE. If it is NULL alpha is initialized to 1. It serves as prior for the Dirichlet distributions over the latent groups. They serve as pseudo counts of individuals per group.
beta	A single number. If it is NULL beta is initialized to 0.1. It serves as a prior for the Dirichlet distributions over the categorical responses. Large numbers favor an equal distribution of responses for a question of the individuals in the same latent group, small numbers indicate that individuals of the same latent group usually answer a question the same way.
select_latent	A boolean that indicates if the exact number n_latent should be used or if a Dirichlet Process prior is used that shrinks the number of used latent variables appropriately (can be controlled with alpha=c(a1, a2) and beta). Default: FALSE.
max_iter	The maximum number of iterations.
epsilon	A number that indicates the numerical precision necessary to consider the algorithm converged.

na_handle	Either "ignore" or "category". If it is "category" all NA's in the dataset are converted to the string "(Missing)" and treated as their own category. If it is "ignore" the NA's are treated as missing completely at random and are ignored during the parameter updates.
repetitions	A number specifying how often to repeat the calculation with different initializations. Automatically selects the best run (i.e. max(ELBO)). Default: 1.
...	Additional parameters passed on to the underlying functions. The parameters are verbose, phi_init, zeta_init and if select_latent=FALSE omega_init or if select_latent=TRUE kappa1_init and kappa2_init.

Details

The function uses a mixture of multinomials to fit the model. The full model specification is

$$\lambda|\alpha \sim DirichletProcess(\alpha)$$
$$z_i|\lambda \sim Multinomial(\lambda)$$
$$U_{j,k}|\beta \sim Dirichlet(\beta)$$
$$X_{i,j}|U_j, z_i = k \sim Multinomial(U_{j,k})$$

In case that `select_latent=FALSE` the first line is replaced with

$$\lambda|\alpha \sim Dirichlet(\alpha)$$

The initial inspiration came from Dunson and Xing (2009) who proposed a Gibbs sampling algorithm to solve this model. To speed up inference a variational inference approach was derived and implemented in this package.

Value

A list that is tagged with the class "mixdir" containing 8 elements:

converged a boolean indicator if the model has converged

convergence a numerical vector with the ELBO of each iteration

ELBO the final ELBO of the converged model

lambda a numerical vector with the `n_latent` class probabilities

pred_class an integer vector with the the most likely class assignment for each individual.

class_prob a matrix of size `n_ind` x `n_latent` which has for each individual the probability to belong to class k.

category_prob a list with one entry for each feature (i.e. column of X). Each entry is again a list with one entry for each class, that contains the probability of individuals of that class to answer with a specific response.

specific_params A list whose content depends on the parameter `select_latent`. If `select_latent=FALSE` it contains the two entries omega and phi which are the Dirichlet hyperparameters that the model has fitted. If `select_latent=TRUE` it contains kappa1, kappa2 and phi, which are the hyperparameters for the Dirichlet Process and the Dirichlet of the answer.

na_handle a string indicating the method used to handle missing values. This is important for subsequent calls to `predict.mixdir`.

References

1. C. Ahlmann-Eltze and C. Yau, "MixDir: Scalable Bayesian Clustering for High-Dimensional Categorical Data", 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), Turin, Italy, 2018, pp. 526-539.

2. Dunson, D. B. and Xing, C. Nonparametric Bayes Modeling of Multivariate Categorical Data. J. Am. Stat. Assoc. 104, 1042–1051 (2009).

3. Blei, D. M., Ng, A. Y. and Jordan, M. I. Latent Dirichlet Allocation. J. Macine Learn. Res. 3, 993–1022 (2003).

4. Blei, D. M. and Jordan, M. I. Variational inference for Dirichlet process mixtures. Bayesian Anal. 1, 121–144 (2006).

Examples

```
data("mushroom")
res <- mixdir(mushroom[1:30, ])
```

`mushroom` *Properties of 8124 mushrooms.*

Description

A dataset containing 23 categorical properties of 23 different species of gilled mushrooms including a categorization if it is edible or not.

Usage

```
mushroom
```

Format

A data frame with 8124 rows and 23 columns:

bruises `bruises` `no`

cap-color `brown` `yellow` `white` `gray` `red` `pink` `buff` `purple` `cinnamon` `green`

cap-shape `convex` `bell` `sunken` `flat` `knobbed` `conical`

cap-surface `smooth` `scaly` `fibrous` `grooves`

edible `poisonous` `edible`

gill-attachment `free` `attached`

gill-color `black` `brown` `gray` `pink` `white` `chocolate` `purple` `red` `buff` `green` `yellow` `orange`

gill-size `narrow` `broad`

gill-spacing `close` `crowded`

habitat `urban` `grasses` `meadows` `woods` `paths` `waste` `leaves`

odor pungent almond anise none foul creosote fishy spicy musty

population scattered numerous abundant several solitary clustered

ring-number one two none

ring-type pendant evanescent large flaring none

spore-print-color black brown purple chocolate white green orange yellow buff

stalk-color-above-ring white gray pink brown buff red orange cinnamon yellow

stalk-color-below-ring white pink gray buff brown red yellow orange cinnamon

stalk-root equal club bulbous rooted NA

stalk-shape enlarging tapering

stalk-surface-above-ring smooth fibrous silky scaly

stalk-surface-below-ring smooth fibrous scaly silky

veil-color white brown orange yellow

veil-type partial

Details

The records are drawn from G. H. Lincoff (1981) (Pres.), *The Audubon Society Field Guide to North American Mushrooms*. New York: Alfred A. Knopf. (See pages 500–525 for the Agaricus and Lepiota Family.)

The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like "leaflets three, let it be" for Poisonous Oak and Ivy.

The actual dataset from the UCI repository has been cleaned up to properly label the missing values and have the full category names instead of their abbreviations.

Source

https://archive.ics.uci.edu/ml/datasets/Mushroom

References

Blake, C.L. & Merz, C.J. (1998). UCI Repository of Machine Learning Databases. Irvine, CA: University of California, Department of Information and Computer Science.

Examples

```
data("mushroom")
summary(mushroom)
```

plot_features *Plot cluster distribution for a subset of features features*

Description

Plot cluster distribution for a subset of features features

Usage

```
plot_features(features, category_prob,
  classes = seq_len(length(category_prob[[1]])))
```

Arguments

features	a character vector with feature names
category_prob	a list over all features containing a list of the probability of each answer for every class. It is usually obtained from the result of a call to `mixdir()`.
classes	numerical vector specifying which latent classes are plotted. By default all.

Examples

```
data("mushroom")
res <- mixdir(mushroom[1:100, ], n_latent=4)
plot_features(c("bruises", "edible"), res$category_prob)

res2 <- mixdir(mushroom[1:100, ], n_latent=20)
def_feats <- find_defining_features(res2, mushroom[1:100, ], n_features=Inf)
plot_features(def_feats$features[1:6], category_prob = res2$category_prob,
              classes=which(res$lambda > 0.01))
```

predict.mixdir *Predict the class of a new observation.*

Description

Predict the class of a new observation.

Usage

```
## S3 method for class 'mixdir'
predict(object, newdata, ...)
```

Arguments

object	the result from a call to `mixdir()`. It needs to have the fields lambda, category_prob and na_handle. lambda is a vector of probabilities for each category. category_prob a list of a list of a named vector with probabilities for each feature, latent class and possible category. na_handle must either be "ignore" or "category" depending how NA's should be handled.
newdata	a named vector with a single new observation or a data.frame with the same structure as the original data used for fitting the model. Missing features or features not encountered during training are replaced by NA.
...	currently unused

Value

A matrix of with the same number of rows as the input and one column for each latent class.

Examples

```
data("mushroom")
X <- as.matrix(mushroom)[1:30, ]

res <- mixdir(X)

# Predict Class
predict(res, mushroom[40:45, ])
predict(res, c(`gill-color`="black"))
```

Index