Package 'arulesNBMiner'

October 7, 2026

Version 0.1.10

Date 2026-10-06

Title Mining NB-Frequent Itemsets and NB-Precise Rules

Description NBMiner is an implementation of the model-based mining algorithm for mining NB-frequent itemsets and NB-precise rules. Michael Hahsler (2006) <doi:10.1007/s10618-005-0026-2>.

Depends R (>= 3.6), arules (>= 1.6-0), rJava (>= 0.9-0)

URL https://github.com/mhahsler/arulesNBMiner, https://michael.hahsler.net/arulesNBMiner/

BugReports https://github.com/mhahsler/arulesNBMiner/issues

Imports methods, stats, graphics

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

SystemRequirements Java (>= 5.0)

License GPL-3

Encoding UTF-8

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Michael Hahsler [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-2716-1405>)

Maintainer Michael Hahsler <mhahsler@lyle.smu.edu>

Repository CRAN

Date/Publication 2026-10-06 22:40:10 UTC

Contents

Agrawal 2
NBMiner 3
NBMinerParameters 5

Index 7

`Agrawal` *Synthetic Example Dataset Agrawal*

Description

This dataset is generated by the method described by Agrawal and Srikant (1994) using the reimplementation in **arules** which also retains the patterns used in the generation process.

Format

The format is: `transactions Agrawal.db itemsets Agrawal.pat`

Details

`Agrawal.db` contains the dataset (1000 items/20000 transactions) and `Agrawal.pat` contains the patterns that were used to create the dataset.

References

Rakesh Agrawal and Ramakrishnan Srikant (1994). Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, pages 487-499, Santiago, Chile.

See Also

`arules::random.transactions()`

Examples

```
data(Agrawal)

summary(Agrawal.pat)
summary(Agrawal.db)

## the data set was generated with the following code
## Not run:
Agrawal.pat <- random.patterns(1000, nPats = 2000,  method = "agrawal",
    lPats = 2, corr = 0.5, cmean = 0.5, cvar = 0.1, iWeight = NULL,
    verbose = FALSE)
Agrawal.db <- random.transactions(1000, 20000, method="agrawal",
    patterns = Agrawal.pat)

## End(Not run)
```

NBMiner *NBMiner: Mine NB-Frequent Itemsets or NB-Precise Rules*

Description

Mines NB-frequent itemsets or NB-precise rules.

Usage

```
NBMiner(data, parameter, control = NULL, ...)
```

Arguments

data	object of class arules::transactions.
parameter	an NBMinerParameter object or a named list of its parameters. Use NBMinerParameters() to estimate the model parameters.
control	an NBMinerControl object or a named list of control options. verbose and debug are logical options that control progress and diagnostic output.
...	named parameter overrides applied to parameter before mining. For example, use rules = TRUE to mine rules or change pi or theta.

Details

Mines NB-frequent itemsets or NB-precise rules (Hahsler, 2006) are non-spurious patterns that occur significantly more often in the data than one would expect if the items were independent. Under independence, we model each item's frequency as a Poisson count with an item specific rate, and models variation in those rates with a Gamma distribution. Mixing the Poisson counts over the Gamma rates gives a negative binomial distribution. This flexible baseline captures the skewed frequency distributions common in transaction data: a few items occur often, while many occur rarely. The model parameters for the independence model can be estimated from the data using NBMinerParameters().

NBMiner considers for an itemset adding each possible other item. Given the independent baseline it predicts how many extensions would reach each frequency by chance. The pi parameter sets the minimum predicted precision for accepting extensions. Here, precision means the predicted proportion of accepted extensions that are true associations under the model. Only extensions with a precision of at least pi are accepted.

The theta parameter controls pruning during search. For a larger itemset, it sets the required fraction of its immediate subsets that must support the itemset as an NB-frequent pattern. A value of 1 is most restrictive requiring all subsets also to be NB-frequent; 0 relaxes this condition. The intermediate default value 0.5 balances pruning with the chance of retaining associations whose items have different frequencies.

maxlen limits the longest itemset considered, and minlen can set a minimum length for returned patterns.

The mining algorithm uses a depth-first search implemented in Java.

Details can be found in Hahsler (2006).

Value

An object of class arules::itemsets or arules::rules, depending on the rules parameter. Estimated precision is stored in the quality slot.

References

Michael Hahsler. A model-based frequency constraint for mining associations from transaction data. *Data Mining and Knowledge Discovery*, 13(2):137-166, September 2006. doi:10.1007/s1061800500262

Examples

```
data("Agrawal")

# Estimate independence model parameters
param <- NBMinerParameters(Agrawal.db, trim = 0)

# Mine non-spurious patterns
itemsets_NB <- NBMiner(Agrawal.db,
                       parameter = param,
                       pi = 0.99,
                       theta = 0.5,
                       minlen = 2L)

inspect(head(itemsets_NB))

# Compare with the known patterns used to generate the data
num_correct <- function(itemsets, patterns)
    table(factor(rowSums(is.subset(itemsets, patterns)) > 0,
          c(FALSE, TRUE)))

# How many found itemsets are subsets of the patterns used in the db?
num_correct(itemsets_NB, Agrawal.pat)

# Compare with the same number of the most frequent itemsets
itemsets_supp <-  eclat(Agrawal.db, parameter = list(supp = 0.001, minlen = 2))
itemsets_supp <- head(sort(itemsets_supp, by = "support"), length(itemsets_NB))
num_correct(itemsets_supp, Agrawal.pat)

# we see that NBMiner is much more effective to recover the true patterns.


# Mine NB-precise rules
rules_NB <- NBMiner(Agrawal.db,
                    parameter = param,
                    pi = 0.99,
                    theta = 0.5,
                    rules = TRUE)
rules_NB

inspect(head(rules_NB))
```

`NBMinerParameters` *Estimate Global Model Parameters from Data*

Description

Estimate the parameters for the global negative binomial independence model used by NBMiner.

Usage

```
NBMinerParameters(
  data,
  trim = 0.01,
  pi = 0.99,
  theta = 0.5,
  bins = 10,
  minlen = 1,
  maxlen = 5,
  rules = FALSE,
  plot = TRUE,
  verbose = TRUE,
  getdata = FALSE
)
```

Arguments

`data`	the data as an object of class `arules::transactions`.
`trim`	fraction of the most frequent items to exclude when fitting the baseline model.
`pi`	minimum predicted precision required to accept an itemset extension or rule.
`theta`	fraction of an itemset's immediate subsets that must be NB-frequent for the itemset to be considered during search.
`bins`	number of bins used for the chi-squared goodness-of-fit test.
`minlen`	minimum number of items in returned itemsets (default: 1).
`maxlen`	maximum number of items in returned itemsets (default: 5).
`rules`	whether to mine NB-precise rules instead of NB-frequent itemsets.
`plot`	whether to plot the observed and fitted frequency distributions.
`verbose`	whether to print progress and goodness-of-fit results.
`getdata`	whether to return the parameter object together with observed counts, expected counts, and the chi-squared test result.

Details

The model is fit using observed item frequencies in the data. The expectation maximization (EM) algorithm (Dempster et al, 1977) is used to estimate the global NB model because the zero class (missing values representing items that do not occur in the dataset) is not observed. This procedure iteratively estimates missing values using the observed data and the model using intermediate values of the parameters, and then uses the estimated data and the observed data to update the parameters for the next iteration. The procedure stops when the parameters stabilize.

Another common issue is the presence of outliers with unusually high frequencies. These outliers will distort the mean and the variance and thus will lead to a model that grossly overestimates the probability of seeing items with high frequencies. For a more robust estimate, we can trim a suitable percentage of the items with the highest frequencies. A suitable percentage can be found by visual comparison of the empirical data and the estimated model or by minimizing the χ^2-value of the goodness-of-fit test which is reported when run with `verbose = TRUE`. A diagnostic plot comparing the observed data with the model is shown with `plot = TRUE`. The plot shows the number of items with a frequency larger than r.

The result is the two NB parameters k and a, but note that a is rescaled by dividing it by the number of incidences in the data, as required by NBMiner. The estimated total number of items n including the fitted number of unseen items (items with a frequency of 0) is also returned.

Only `data` and `trim` are used for the estimation. `bins` can be used to change the number of bins used in the goodness-of-fit test. The other parameters are stored in the parameter object for use by `NBMiner()`.

Value

An object of class `NBMinerParameter` for use with `NBMiner()`. If `getdata = TRUE`, a list containing the parameter object, observed counts, expected counts, and the result of the chi-squared test is returned.

References

Michael Hahsler. A model-based frequency constraint for mining associations from transaction data. *Data Mining and Knowledge Discovery*,13(2):137-166, September 2006. doi:10.1007/s10618-00500262

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological),* 39:1–38. doi:10.1111/j.25176161.1977.tb01600.x

Examples

```
data("Epub")
Epub

param <- NBMinerParameters(Epub, trim = 0.04)
param
```

Index

∗ **datasets**
Agrawal, 2
∗ **models**
NBMiner, 3
NBMinerParameters, 5

Agrawal, 2
arules::itemsets, *4*
arules::random.transactions(), *2*
arules::rules, *4*
arules::transactions, *3*, *5*

NBMiner, 3
NBMiner(), *6*
NBMinerControl-class *(*NBMiner*)*, 3
NBMinerParameter-class *(*NBMiner*)*, 3
NBMinerParameters, 5
NBMinerParameters(), *3*