Package ‘SERB’

June 15, 2023

Type Package

Title Síntesis de Explicaciones en Redes Bayesianas

Version 1.0.0

Author Bartomeu Ramis

Maintainer Bartomeu Ramis <bartomeuramis99@gmail.com>

Description A simple package that allows for the creation of Knowledge Bases from Bayesian Networks, their spacial optimisation and quering.

License /

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports dplyr,
bnlearn

Depends R (>= 2.10)

R topics documented:

`count_rules` *Count Rules of Knowledge Base*

Description

Count the number of independent "Rules" that define the Knowledge Base.

Usage

```
count_rules(KB, verbose)
```

Arguments

`KB`	Table structure of a Knowledge Base. The two last columns must be Decision and Prob.
`verbose`	Boolean value. If TRUE prints out the ruleset in indices notation, if FALSE does not print any information.

Details

A Rule of a KB is defined as the last item of a adjacent set of evidences with the same Decision. The fewer rules a KB has, the more optimal its space optimization is.

Value

Returns an the number of rules that define the KB.

References

Fernández del Pozo, J.A., Bielza, C. and Gómez, M. (2003) *A list-based compact representation for large decision tables management*, European Journal of Operational Research.

Examples

```
#Create the associated KB
KB <- createKB(bn, "tub")
count_rules(KB, FALSE)
```

`createKB` *Create Knowledge Base*

Description

Create a Knowledge Base from a given Bayesian Network and a Target node from that same network. The table structure created contains two new columns: Decision and Prob.

Usage

```
createKB(BN, node)
```

Arguments

`BN`	Must be a Discrete Bayesian Network created with the “bnlearn” package.
`node`	String name of the Target Node we want to study.

Details

This function creates a table with all possible combinations of input values for the target node. From this table we query the model repeatedly, saving the Decision given and the Probability of said Decision. Each Decision and Probability is saved alongside the evidence that generated it, and it is return in the structure of a Table, called Knowledge Base.

Value

A Table with all possible evidence combinations and the value and probability of the Target node for each instance.

References

Fernández del Pozo, J.A., Bielza, C. and Gómez, M. (2003) *A list-based compact representation for large decision tables management*, European Journal of Operational Research.

Examples

```
#Create the KB from a preloaded Bayesian Network (bn)
KB <- createKB(bn, "tub")
```

`createKBM2L` *Creation of Knowledge Base Matrix To List structure*

Description

Creates a KBM2L list from a KB structure. A KBM2L is a list that reduces the space needed to store a KB structure with, almost, no information loss.

Usage

```
createKBM2L(KB)
```

Arguments

`KB`	Knowledge Base table from which to create the KBM2L.

Value

A list that contains the rules that form the KB entered as input.

Examples

```
#Create the associated KB
KB <- createKB(bn, "tub")
#Optimize KB
optimalKB <- genetic(KB, 20, 0.05, 0.9, 100)

KBM2L <- createKBM2L(optimalKB)
```

`fixed` *Identify the Fixed of a Rule*

Description

Identifies the Fixed part of a rule. The Fixed part explains the parameters that are most relevant to the decision chose. So this function returns the Most Relevant attributes of a set of evidence.

Usage

```
fixed(kbm2l, evidence)
```

Arguments

`kbm2l`	KBM2L structure to query.
`evidence`	Must be a vector containing evidence for all the atributes of a KB, including the Decision and Prob columns.

Value

Returns the list of the s that comprise the Fixed Part of a rule.

Examples

```
#Create the associated KB
KB <- createKB(bn, "tub")
#Optimize KB
optimalKB <- genetic(KB, 20, 0.05, 0.9, 100)
#Create the KBM2L
KBM2L <- createKBM2L(optimalKB)

#Identify the fixed part of the first item of a KB
fix1 <- fixed(KBM2L, optimalKB[1,])

#Identify the fixed part of the second rule (this time using the KBM2L structure)
fix2 <- fixed(KBM2L, KBM2L[2,])
```

`genetic` *Genetic Algorithm*

Description

Genetic algorithm that searches for the Base that optimizes the number of rules of a KB.

Usage

```
genetic(KBo, np, m, q, t)
```

Arguments

`KBo`	Table structure of the KB to optimize.
`np`	(Integer) Number of individuals in the population.
`m`	(Double) Mutation chance for each new individual. Should be between 0 and 1.
`q`	(Double) Selection chance for a individual. Should be between 0 and 1.
`t`	(Integer) Number of generations to iterate through.

Value

Returns the KB that has been found that has the less number of rules, and therefore is the more space optimal.

Examples

```
#Create the associated KB
KB <- createKB(bn, "tub")

optimalKB <- genetic(KB, 20, 0.05, 0.9, 100)
```

`query.list` *Query the KBM2L with partial evidence*

Description

Make a query to the KBM2L structure in order to obtain all possible instances of a partial (or complete) evidence. Given some partial evidence, this function return all possible alternatives with their associated decision and probability. Using complete evidence is a particular case where the there is only one possible decision.

Usage

```
query.list(KBM2L, evidence)
```

Arguments

`KBM2L`	KBM2L structure to query.
`evidence`	Vector of partial evidence to search in the KB structure.

Value

Returns a list of the possible instances of the evidence given with their associated Decision and Probability.

Examples

```
#Create the associated KB
KB <- createKB(bn, "tub")
#Optimize KB
optimalKB <- genetic(KB, 20, 0.05, 0.9, 100)
#Create the KBM2L
KBM2L <- createKBM2L(optimalKB)

#Create some partial evidence
evidence <- setNames(c("yes", "yes", "no"), c("Tub", "Asia", "Smoke"))

query.list(KBM2L, evidence)
```

`readKB` *Read Knowledge Base*

Description

Load the Knowledge Base structure from a text file.

Usage

```
readKB(file)
```

Arguments

`file`	String name of the file that contains the KB.

Value

Returns the Table structure saved in the file.

Examples

```
  #File should be a KB previously saved
  #KB <- readKB("output.txt")
```

`sima` *Simulated Annealing algorithm*

Description

Simulated Annealing search algorithm that searches for the Base that optimizes the number of rules of a KB.

Usage

```
sima(KB, T0, k, c)
```

Arguments

`KB`	Table structure of the KB to optimize.
`T0`	(Integer) Initial temperature of the system.
`k`	(double) Random chance of choosing a non-optimal partial solution. Should be between 0 and 1.
`c`	(double) Rate of cooling of the system. Should be between 0 and 1.

Value

Returns the KB that has been found that has the less number of rules, and therefore is the more space optimal.

Examples

```
#Create the associated KB
KB <- createKB(bn, "tub")

optimalKB <- sima(KB, 1000, 0.1, 0.9)
```

`tabu` *Taboo Algorithm*

Description

Taboo search algorithm that searches for the Base that optimizes the number of rules of a KB.

Usage

```
tabu(KB, m, maxtabu)
```

Arguments

`KB`	Table structure of the KB to optimize.
`m`	(Integer) Number of iterations to search for.
`maxtabu`	(Integer) Max length of the Taboo list. The longer the list is made, the more memory it has to avoid previous solutions. This parameter has a maximum length defined by the number of attributes of a KB. If the taboo list is too long, at some point all neighbors of a base will be considered taboo and the program will throw out an error.

Value

Returns the KB that has been found that has the less number of rules, and therefore is the more space optimal.

Examples

```
#Create the associated KB
KB <- createKB(bn, "tub")

optimalKB <- tabu(KB, 100, 10)
```

`writeKB` *Write Knowledge Base*

Description

Write a Knowledge Base structure to an external file.

Usage

```
writeKB(file, KB)
```

Arguments

`KB`	Knowledge Base to write to file.
`file`	String name of the file name where you want to store the KB. If the file does not exist it creates it.

Value

Does not return anything.

Author(s)

Bartomeu Ramis Tarragó

Examples

```
#Create a example KB
KB <- data.frame(var1=c(1, 3, 3, 4, 5),
                 var2=c(7, 7, 8, 3, 2),
                 var3=c(3, 3, 6, 6, 8),
                 var4=c(1, 1, 2, 8, 9))
#Save it to file_output.txt
writeKB("file_output.txt", KB)
```

Index