Package 'hmma'

June 27, 2020

Type Package

Title Constructs Asymmetric Hidden Markov Models

Version 1.1.0

Description Asymmetric hidden Markov model (HMM-A) learning.
HMM-As are similar to regular HMMs, but use Bayesian Networks (BNs) in their emission distribution. The HMM-As can therefore offer more problem insight [Bueno et al. 2017, <doi:10.1016/j.ijar.2017.05.011>].

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.6.0)

VignetteBuilder knitr

Suggests knitr, testthat, rmarkdown

Imports bnlearn, mhsmm, MCMCpack, Rgraphviz, graph, methods

RoxygenNote 7.1.0

NeedsCompilation yes

Author Marcos L.P Bueno [aut],
Arjen Hommersom [aut, cre],
Joop Thibaudier [aut],
Marco Scutari [ctb],
Robert Ness [ctb],
Robert Gentleman and Ross Ihaka [cph],
The R Development Core Team [cph],
The R Foundation [cph]

Maintainer Arjen Hommersom <arjen.hommersom@ou.nl>

Repository CRAN

Date/Publication 2020-06-26 22:50:03 UTC

R topics documented:

`createHmma` *Create an asymmetric hidden Markov model.*

Description

The `createHmma` method creates a HMM-A from the specifications provided. The amount of states are implicitly derived from the initial distribution. Bayesian networks can be created using the `bnlearn` package.

Usage

```
createHmma(init, trans, bns)
```

Arguments

`init`	This initial distribution as a vector.
`trans`	The transition distribution as a matrix.
`bns`	The Bayesian networks for the states.

See Also

`bnlearn` for more information regarding the creation of Bayesian networks

Examples

```
# Start by creating the initial and transition distribution
init <- c(0.3, 0.7)
trans <- c(0.4, 0.7, 0.6, 0.3)
dim(trans) <- c(2,2)

# Create a Baysian network for each state using 'bnlearn'
library(bnlearn)
struc <- model2network("[X1][X2]")
cptX1 <- matrix(c(0.15, 0.85), ncol = 2, dimnames = list(NULL, c("TRUE", "FALSE")))
cptX2 <- matrix(c(0.7, 0.3), ncol = 2, dimnames = list(NULL, c("TRUE", "FALSE")))

bn1 <- custom.fit(struc, dist = list(X1 = cptX1,
```

```
                                      X2 = cptX2))

struc <- model2network("[X2|X1][X1]")
cptX1 <- matrix(c(0.4, 0.6), ncol = 2, dimnames = list(NULL, c("TRUE", "FALSE")))
cptX2 <- matrix(c(0.9, 0.1, 0.5, 0.5), nrow = 2, ncol = 2)
dimnames(cptX2) <- list("X2" = c("TRUE", "FALSE"),
                        "X1" = c("TRUE", "FALSE"))

bn2 <- custom.fit(struc, dist = list(X1 = cptX1,
                                      X2 = cptX2))

bns <- list()
bns[[1]] <- bn1
bns[[2]] <- bn2

# Create the model
hmma <- createHmma(init = init, trans = trans, bns = bns)
```

---

hmma — hmma*: A package for the contruction of asymmetric hidden Markov models*

Description

The hmma package is able to construct asymmetric hidden Markov models (HMM-As) from data. HMM-As are similar to regular HMMs, but use Bayesian Networks (BNs) in their emission distribution.

Details

HMMs have successfully been used in “speech recognition systems, in numerous applications in-computational molecular biology, in data compression, and in other areas of artificial intelligence-and pattern recognition”.

When limited data is available, HMMs may be unable to correctly capture these distributions.Bueno et al. show that HMMs can be enriched by employing state-specific Bayesian networks(BNs), i.e. BNs that may be different from state to state. This enables the model to better capturecertain in-dependencies, depending on the state. The resulting models are called asymmetrichidden Markov models (HMM-As). (see: http://dx.doi.org/10.1016/j.ijar.2017.05.011 for more information)

`hmmaExampleData` *An example dataset for use on HMM-As*

Description

This dataset has been generated using a HMM-A with two different Bayesian networks. The observations consist of 100 sequences of length 20.

Usage

```
hmmaExampleData
```

Format

A list with the following items

x All observations.

N The length of the observation sequence.

Details

The sum of all observation lengts (hmmaExampleData$N) must be equal to the total number of observations (hmmExampleData$x).

`learnModel` *Learna a HMM-A from data*

Description

The `learnModel` function, learns a HMM-A from the supplied data file. The function first creates a random model: the initial and transition distributions are initialized using a Dirichlet `https://en.wikipedia.org/wiki/Dirichlet_distribution` distribution. Thereafter the model is maximised for the datafile that is supplied.

Usage

```
learnModel(
  data,
  amountOfStates = 2,
  maxit = 50,
  seed,
  iss = 1e-04,
  debug = FALSE
)
```

Arguments

`data`	The datafile. The datafile should be a list containing a dataframe with the data as its $x component and contain the lengths of the observations a the $N component (see details).
`amountOfStates`	The amount of states.
`maxit`	The maximum amount of iterations.
`seed`	Seed (optional).
`iss`	The Imaginary Sample Size (iss), also called priors, to add data.
`debug`	Debugmode.

Details

The `learnModel` makes use of the `mhsmm` `hmmfit` function. An example of the structure of the datafile can be found in `hmmaExampleData`.

Value

The output of the function is an asymmetric hidden Markov model. This model contains the amount of states, the initial distribution, the transition distribution and the emission distribution (Bayesian networks in the different states).

The model can quicly be visualised with the `visualise` method. The `visualise` method does not show the Bayesian networks within the states as this would result in unreadable graphs. Instead, the `bnlearn` `graphviz.plot` method can be used (see the examples below).

Examples

```
fit <- learnModel(data = hmmaExampleData, amountOfStates = 3, seed = 1234)
visualise(fit)

# See bn in first state
library(bnlearn)
graphviz.plot(fit$parms.emission[[1]])
```

---

predict.hmma *Predict sequence of data on HMMA*

Description

`predict.hmma` is simply a wrapper function for the mhsmm `predict.hmmspec` function. It generates the predicted state sequence for dat data, given the model.

Usage

```
## S3 method for class 'hmma'
predict(object, data, method = "viterbi", ...)
```

Arguments

`object`	The HMMA
`data`	The data file
`method`	The prediction method (viterbi or smoothed, see details)
`...`	Futher arguments.

Details

From the mhsmm documentation: If method="viterbi", this technique applies the Viterbi algorithm for HMMs, producing the most likely sequence of states given the observed data. If method="smoothed", then the individually most likely (or smoothed) state sequence is produced, along with a matrix with the respective probabilities for each state. This function differs from predict.hmm in that it takes the output from hmmspec ie. this is useful when users already know their parameters and wish to make predictions.

Value

The return object contains the data, a vector 's' with the reconstructed state sequence, the vector N with the lengths of the sequences, a matrix p with the probabilities of the states (only with smoothed). For more details see `predict.hmmspec`.

The loglikelihood is returned as the `$loglik` component.

See Also

`predict.hmmspec` for more information.

`simulate.hmma` *Simulates a HMM-A.*

Description

`simulate.hmma` simulates a HMM-A. The only difference between `simulate.hmma` and `simulate.hmmspec` is that this function adds the names of the nodes of the Bayesian networks to the results. The `simulate.hmmspec` function only uses numbers, instead of names.

Usage

```
## S3 method for class 'hmma'
simulate(object, nsim, seed = NULL, ...)
```

Arguments

`object`	A hmma object
`nsim`	An integer or vector of integers (for multiple sequences) specifying the length of the sequence(s)
`seed`	seed for the random number generator
`...`	Further arguments passed to or from other methods.

Value

A simulation with the following values

- `s` The states of the model.
- `x` The observations.
- `n` The length of the observation.

Examples

```
# Get a HMM-A (e.g. from learnModel or createHmma)
model <- learnModel(data = hmmaExampleData, amountOfStates = 2, seed = 1234)

# Simulate the data, results in 100 observation sequences of length 20.
data <- simulate(model, nsim = c(rep(20, 100)))
```

---

visualise — *Visualise the HMM-A*

Description

The function *visualise* shows a graphical representation of the found HMM-A. Only the initial distribution and transition distribution are shown. Each state contains a Bayesian network, these would be unreadble when displayed within each state. The `bnlearn` `graphviz.plot` method can be used to display the bayesian networks.

Usage

```
visualise(
  model,
  minProb = 0,
  numDigitsRound = 2,
  relateWidthWithWeight = TRUE,
  widthType = "linear",
  minWidth = 0.05,
  maxWidth = 5
)
```

Arguments

`model`	The model.
`minProb`	The minimum probability of an arc (initial or transition) to be drawn in the visualisation (see details).
`numDigitsRound`	The amount of decimals that should be presented in the graph.
`relateWidthWithWeight`	An edge with a high weight will be drawn with a thicker line compared to an edge with a lower weight.

widthType	The thickness function for the lines. Defaults to linear function. When `sigmoid` is used, a sigmoid function is used to determine the thickness.
minWidth	The minimum thickness for a line. Only used when `relateWidthWithWeight` is `TRUE`
maxWidth	The maximum thickness for a line. Only used when `relateWidthWithWeight` is `TRUE`

Details

Per default, all transitions (initial and transition) are drawn on the canvas. For larger HMM-As, the visualisation can become cluttered with arcs that have a low probability. By setting the `minProb`, an arc is only drawn when it is equal to or greater than the specified `minProb`. The default value for `minProb` is 0.00 (all arcs are drawn).

To further improve the visualisation, edge line width can be related to the respective weight of the edge. When `relateWeightWithThickness` is `TRUE`, an edge with a higher weight will be drawn with a wider line. The minimum width of a line and the maximum width can be supplied via the respective parameters.

The `minWidth` is associated with a weight of 0.00, the maxWidth is related to the weight 1.00. When `widthType` is set to 'linear', the width of a line is linearly related to the weight. When `widthType` is set to 'sigmoid', the width of a line is determined via a sigmoid of the weight.

Examples

```
# First, we need a model that we want to visualise, we create
# one using the learnModel function.
fit <- learnModel(data = hmmaExampleData, amountOfStates = 3)

# To visualise the states and transitions, we use the visualise method.
# Only lines with a weight of 0.10 are drawn.
visualise(fit, minProb = 0.10)

# When it is not desired to relate the width of a line with its weight,
# this can be disabled:
visualise(fit, minProb = 0.10, relateWidthWithWeight = FALSE)

# Finally, it is possible to use a sigmoid instead of a linear relation:
visualise(fit, minProb = 0.10, widthType = 'sigmoid')

# To visualise the BNs within the states, use the code below
library(bnlearn)
graphviz.plot(fit$parms.emission[[1]])
```

Index