Package 'RcppHMM'

November 21, 2017

Type Package

Title Rcpp Hidden Markov Model

Version 1.2.2

Date 2017-11-21

Author Roberto A. Cardenas-Ovando, Julieta Noguez and Claudia Rangel-Escareno

Maintainer Roberto A. Cardenas-Ovando <robalecarova@gmail.com>

Description Collection of functions to evaluate sequences, decode hidden states and estimate parameters from a single or multiple sequences of a discrete time Hidden Markov Model. The observed values can be modeled by a multinomial distribution for categorical/labeled emissions, a mixture of Gaussians for continuous data and also a mixture of Poissons for discrete values. It includes functions for random initialization, simulation, backward or forward sequence evaluation, Viterbi or forward-backward decoding and parameter estimation using an Expectation-Maximization approach.

License GPL (>= 2)

Imports Rcpp (>= 0.12.6)

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

SystemRequirements C++11

Repository CRAN

Date/Publication 2017-11-21 19:27:14 UTC

R topics documented:

RcppHMM-package *Overview of Package RcppHMM*

Description

This package can model observations based on hidden Markov models. The observations can be considered to be emitted by a multinomial distribution, A mixture of Gaussians or a mixture of Poissons. It can be used for inference, parameter estimation and simulation.

Details

The package can be used to represent a discrete-time hidden Markov model. The states can generate categorical (labeled), continuous or discrete observations. The hidden state transition and observations can be randomly generated based on fixed parameters. Also, the inference methods can be used to evaluate sequences or decode the hidden states that generated the observations. Finally, the model parameters can be estimated by a single or multiple observed sequences.

Author(s)

Roberto A. Cardenas-Ovando, Julieta Noguez and Claudia Rangel-Escareno

Maintainer: Roberto A. Cardenas-Ovando <robalecarova@gmail.com>

References

Bilmes, J.E. (1998). A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. *International Computer Science Institute*.

Ibe, O. (2009). Markov processes for stochastic modeling. *Oxford*.

Rabiner, L.R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*.

Rabiner L.; Juang, B.H. (1993) Fundamentals of Speech Recognition. *Prentice Hall Signal Processing Series*.

Examples

```
# Multinomial case
# Set the model parameters to be estimated
n <- c("First","Second")
m <- c("A","T","C","G")
A <- matrix(c(0.8,0.2,
              0.1,0.9),
            nrow = 2,
            byrow = TRUE)

B <- matrix(c(0.2, 0.2, 0.3, 0.3,
              0.4, 0.4, 0.1, 0.1),
            nrow = 2,
            byrow = TRUE)

Pi <- c(0.5, 0.5)


params <- list( "Model" = "HMM",
                "StateNames" = n,
                "ObservationNames" = m,
                "A" = A,
                "B" = B,
                "Pi" = Pi)

# Model parameters validation

HMM <- verifyModel(params)

# Data simulation
# Multiple sequences

set.seed(100)
length <- 100
seqs <- 100
observationSequences<- c()
for(i in 1:seqs){
  Y <- generateObservations(HMM , length)$Y
  observationSequences <- rbind(observationSequences , Y)
}

# New model random initialization

set.seed(1000)
newModel <- initHMM(2,4)
n = c("X1","X2")
m = c("A","T","C","G")

# Change model names

newModel <- setNames(newModel,
                        list( "StateNames" = n,
```

```
                                "ObservationNames" = m) )

# Model parameters estimation

newModel <- learnEM(newModel,
        observationSequences,
        iter=300,
        delta = 1E-8,
        pseudo = 0,
        print = TRUE)

# New sequence simulation to compare the new model
# Data simulation

# Single sequence
Y <- generateObservations(HMM , length)$Y

# Evaluation

evaluation(newModel, Y, "f")
evaluation(newModel, Y, "b")

# Hidden state decoding

hiddenStatesViterbi <- viterbi(newModel, Y)
hiddenStatesFB <- forwardBackward( newModel, Y)
```

Change Log *Changes Made to Package RcppHMM*

Description

This page contains a listing of recent changes made to the package.

Details

1. More examples were added to some functions. (November 2017)
2. Since there are different classes of HMMs and each of them with the same algorithms, a verification step was added to avoid memory leaks and variable compatibility. (May 2017)
3. The class of HMM with observations being modelled by a Gaussian Mixture Model (GHMM) was updated to have also a multivariate version (see initGHMM). (July 2017)
4. The emission matrix of the GHMM model was divided into two parameters: Mu and Sigma. Mu is now a 2D matrix with number of rows equal to the dimensionality of the observation vector and the number of columns equal to the number of hidden states. Sigma is now a 3D matrix with number of rows and columns equal to the the dimensionality of the observation vector and the number of slices equal to the number of hidden states (see initGHMM). (July 2017)

`evaluation` *Observed sequence evaluation given a model*

Description

This function computes the log-likelihood of an observed sequence being generated by a hidden Markov model with fixed parameters.

Usage

```
evaluation(hmm , sequence , method = "f" )
```

Arguments

`hmm`	a list with the necessary variables to define a hidden Markov model.
`sequence`	sequence of observations to be evaluated. HMM and PHMM use a vector. GHMM uses a matrix.
`method`	method specified to perform the evaluation

Details

The methods to be selected can be "f" for the forward algorithm or "b" for the backward algorithm. GHMM uses a matrix with the variables as rows and consecutive observations in the columns.

Value

A value that represents the log-likelihood of the sequence given the hiddden Markov model.

References

Cited references are listed on the RcppHMM manual page.

See Also

`generateObservations` , `verifyModel` , `loglikelihood`

Examples

```
## Values for a hidden Markov model with categorical observations
# Set the model parameters
n <- c("First","Second")
m <- c("A","T","C","G")
A <- matrix(c(0.8,0.2,
              0.1,0.9),
            nrow = 2,
            byrow = TRUE)

B <- matrix(c(0.2, 0.2, 0.3, 0.3,
```

```
              0.4, 0.4, 0.1, 0.1),
            nrow = 2,
            byrow = TRUE)

Pi <- c(0.5, 0.5)

params <- list( "Model" = "HMM",
                "StateNames" = n,
                "ObservationNames" = m,
                "A" = A,
                "B" = B,
                "Pi" = Pi)

HMM <- verifyModel(params)

# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM, length)

#Sequence evaluation

# It assumes that it will be evaluated using the forward algorithm
evaluation(HMM, observationSequence$Y)

# The user sets the backward algorithm to evaluate the algorithm
evaluation(HMM, observationSequence$Y, "b")

## Values for a hidden Markov model with discrete observations

n <- c("Low","Normal","High")

A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol=length(n), byrow=TRUE)

B <- c(2600,  # First distribution with mean 2600
       2700,  # Second distribution with mean 2700
       2800)  # Third distribution with mean 2800

Pi <- rep(1/length(n), length(n))

HMM.discrete <- verifyModel(list("Model"="PHMM", "StateNames" = n, "A" = A, "B" = B, "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM.discrete, length)

#Sequence evaluation

# It assumes that it will be evaluated using the forward algorithm
```

```
evaluation(HMM.discrete, observationSequence$Y)

# The user sets the backward algorithm to evaluate the algorithm
evaluation(HMM.discrete, observationSequence$Y, "b")

## Values for a hidden Markov model with continuous observations
# Number of hidden states = 3
# Univariate gaussian mixture model

N = c("Low","Normal", "High")
A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol= length(N), byrow = TRUE)

Mu <- matrix(c(0, 50, 100), ncol = length(N))
Sigma <- array(c(144, 400, 100), dim = c(1,1,length(N)))
Pi <- rep(1/length(N), length(N))

HMM.cont.univariate <- verifyModel(list( "Model"="GHMM",
                                         "StateNames" = N,
                                         "A" = A,
                                         "Mu" = Mu,
                                         "Sigma" = Sigma,
                                         "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM.cont.univariate, length)

#Sequence evaluation

# It assumes that it will be evaluated using the forward algorithm
evaluation(HMM.cont.univariate, observationSequence$Y)

# The user sets the backward algorithm to evaluate the algorithm
evaluation(HMM.cont.univariate, observationSequence$Y, "b")

## Values for a hidden Markov model with continuous observations
# Number of hidden states = 2
# Multivariate gaussian mixture model
# Observed vector with dimensionality of 3
N = c("X1","X2")
M <- 3

# Same number of dimensions
Sigma <- array(0, dim =c(M,M,length(N)))
Sigma[,,1] <- matrix(c(1.0,0.8,0.8,
                       0.8,1.0,0.8,
                       0.8,0.8,1.0), ncol = M,
                     byrow = TRUE)
Sigma[,,2] <- matrix(c(1.0,0.4,0.6,
```

```
                      0.4,1.0,0.8,
                      0.6,0.8,1.0), ncol = M,
                    byrow = TRUE)
Mu <- matrix(c(0, 5,
               10, 0,
               5, 10),
             nrow = M,
             byrow = TRUE)

A <- matrix(c(0.6, 0.4,
              0.3, 0.7),
            ncol = length(N),
            byrow = TRUE)
Pi <- c(0.5, 0.5)

HMM.cont.multi <- verifyModel(list( "Model" = "GHMM",
                                    "StateNames" = N,
                                    "A" = A,
                                    "Mu" = Mu,
                                    "Sigma" = Sigma,
                                    "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM.cont.multi, length)

#Sequence evaluation

# It assumes that it will be evaluated using the forward algorithm
evaluation(HMM.cont.multi, observationSequence$Y)

# The user sets the backward algorithm to evaluate the algorithm
evaluation(HMM.cont.multi, observationSequence$Y, "b")
```

`forwardBackward` *Forward-backward algortihm for hidden state decoding*

Description

Function used to get the most likely hidden states at each observation in the provided sequence.

Usage

```
forwardBackward(hmm, sequence)
```

Arguments

hmm	a list with the necessary variables to define a hidden Markov model.
sequence	sequence of observations to be decoded. HMM and PHMM use a vector. GHMM uses a matrix.

Details

GHMM uses a matrix with the variables as rows and consecutive observations in the columns.

Value

A vector of hidden states in the traveled path of observations.

References

Cited references are listed on the RcppHMM manual page.

See Also

generateObservations , verifyModel , viterbi

Examples

```
## Values for a hidden Markov model with categorical observations
# Set the model parameters
n <- c("First","Second")
m <- c("A","T","C","G")
A <- matrix(c(0.8,0.2,
              0.1,0.9),
            nrow = 2,
            byrow = TRUE)

B <- matrix(c(0.2, 0.2, 0.3, 0.3,
              0.4, 0.4, 0.1, 0.1),
            nrow = 2,
            byrow = TRUE)

Pi <- c(0.5, 0.5)

params <- list( "Model" = "HMM",
                "StateNames" = n,
                "ObservationNames" = m,
                "A" = A,
                "B" = B,
                "Pi" = Pi)

HMM <- verifyModel(params)

# Data simulation
set.seed(100)
length <- 100
```

```
observationSequence <- generateObservations(HMM, length)

#Sequence decoding
hiddenStates <- forwardBackward(HMM, observationSequence$Y)
print(hiddenStates)

## Values for a hidden Markov model with discrete observations

n <- c("Low","Normal","High")

A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol=length(n), byrow=TRUE)

B <- c(2600,  # First distribution with mean 2600
       2700,  # Second distribution with mean 2700
       2800)  # Third distribution with mean 2800

Pi <- rep(1/length(n), length(n))

HMM.discrete <- verifyModel(list("Model"="PHMM", "StateNames" = n, "A" = A, "B" = B, "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM.discrete, length)

#Sequence decoding
hiddenStates <- forwardBackward(HMM.discrete, observationSequence$Y)
print(hiddenStates)

## Values for a hidden Markov model with continuous observations
# Number of hidden states = 3
# Univariate gaussian mixture model

N = c("Low","Normal", "High")
A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol= length(N), byrow = TRUE)

Mu <- matrix(c(0, 50, 100), ncol = length(N))
Sigma <- array(c(144, 400, 100), dim = c(1,1,length(N)))
Pi <- rep(1/length(N), length(N))

HMM.cont.univariate <- verifyModel(list( "Model"="GHMM",
                                         "StateNames" = N,
                                         "A" = A,
                                         "Mu" = Mu,
                                         "Sigma" = Sigma,
                                         "Pi" = Pi))
```

```
# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM.cont.univariate, length)

#Sequence decoding
hiddenStates <- forwardBackward(HMM.cont.univariate, observationSequence$Y)
print(hiddenStates)

## Values for a hidden Markov model with continuous observations
# Number of hidden states = 2
# Multivariate gaussian mixture model
# Observed vector with dimensionality of 3
N = c("X1","X2")
M <- 3

# Same number of dimensions
Sigma <- array(0, dim =c(M,M,length(N)))
Sigma[,,1] <- matrix(c(1.0,0.8,0.8,
                       0.8,1.0,0.8,
                       0.8,0.8,1.0), ncol = M,
                     byrow = TRUE)
Sigma[,,2] <- matrix(c(1.0,0.4,0.6,
                       0.4,1.0,0.8,
                       0.6,0.8,1.0), ncol = M,
                     byrow = TRUE)
Mu <- matrix(c(0, 5,
               10, 0,
               5, 10),
             nrow = M,
             byrow = TRUE)

A <- matrix(c(0.6, 0.4,
              0.3, 0.7),
            ncol = length(N),
            byrow = TRUE)
Pi <- c(0.5, 0.5)

HMM.cont.multi <- verifyModel(list( "Model" = "GHMM",
                                    "StateNames" = N,
                                    "A" = A,
                                    "Mu" = Mu,
                                    "Sigma" = Sigma,
                                    "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM.cont.multi, length)

#Sequence decoding
hiddenStates <- forwardBackward(HMM.cont.multi, observationSequence$Y)
print(hiddenStates)
```

generateObservations *Generate observations given a model*

Description

Function used to generate simulated observations given a hidden Markov model.

Usage

```
generateObservations(hmm, length)
```

Arguments

hmm	a list with the necessary variables to define a hidden Markov model.
length	the number of observations will be generated.

Value

A "list" that contains the generated observations and the hidden state that generated it.

X	a vector representing the path of hidden states.
Y	generated observations. HMM and PHMM return a vector. GHMM returns a matrix.

Examples

```
## Values for a hidden Markov model with categorical observations
# Set the model parameters
n <- c("First","Second")
m <- c("A","T","C","G")
A <- matrix(c(0.8,0.2,
              0.1,0.9),
            nrow = 2,
            byrow = TRUE)

B <- matrix(c(0.2, 0.2, 0.3, 0.3,
              0.4, 0.4, 0.1, 0.1),
            nrow = 2,
            byrow = TRUE)

Pi <- c(0.5, 0.5)

params <- list( "Model" = "HMM",
                "StateNames" = n,
                "ObservationNames" = m,
                "A" = A,
                "B" = B,
```

```
                "Pi" = Pi)

HMM <- verifyModel(params)

# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM, length)
# Observed data
head(observationSequence$Y)
# Hidden states path
head(observationSequence$X)


## Values for a hidden Markov model with discrete observations

n <- c("Low","Normal","High")

A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol=length(n), byrow=TRUE)

B <- c(2600,  # First distribution with mean 2600
       2700,  # Second distribution with mean 2700
       2800)  # Third distribution with mean 2800

Pi <- rep(1/length(n), length(n))

HMM.discrete <- verifyModel(list("Model"="PHMM", "StateNames" = n, "A" = A, "B" = B, "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM.discrete, length)
# Observed data
head(observationSequence$Y)
# Hidden states path
head(observationSequence$X)


## Values for a hidden Markov model with continuous observations
# Number of hidden states = 3
# Univariate gaussian mixture model

N = c("Low","Normal", "High")
A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol= length(N), byrow = TRUE)

Mu <- matrix(c(0, 50, 100), ncol = length(N))
Sigma <- array(c(144, 400, 100), dim = c(1,1,length(N)))
```

```
Pi <- rep(1/length(N), length(N))

HMM.cont.univariate <- verifyModel(list( "Model"="GHMM",
                                         "StateNames" = N,
                                         "A" = A,
                                         "Mu" = Mu,
                                         "Sigma" = Sigma,
                                         "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM.cont.univariate, length)
# Observed data
observationSequence$Y[,1:6]
# Hidden states path
head(observationSequence$X)


## Values for a hidden Markov model with continuous observations
# Number of hidden states = 2
# Multivariate gaussian mixture model
# Observed vector with dimensionality of 3
N = c("X1","X2")
M <- 3

# Same number of dimensions
Sigma <- array(0, dim =c(M,M,length(N)))
Sigma[,,1] <- matrix(c(1.0,0.8,0.8,
                       0.8,1.0,0.8,
                       0.8,0.8,1.0), ncol = M,
                     byrow = TRUE)
Sigma[,,2] <- matrix(c(1.0,0.4,0.6,
                       0.4,1.0,0.8,
                       0.6,0.8,1.0), ncol = M,
                     byrow = TRUE)
Mu <- matrix(c(0, 5,
               10, 0,
               5, 10),
             nrow = M,
             byrow = TRUE)

A <- matrix(c(0.6, 0.4,
              0.3, 0.7),
            ncol = length(N),
            byrow = TRUE)
Pi <- c(0.5, 0.5)

HMM.cont.multi <- verifyModel(list( "Model" = "GHMM",
                                    "StateNames" = N,
                                    "A" = A,
                                    "Mu" = Mu,
                                    "Sigma" = Sigma,
```

```
                                    "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM.cont.multi, length)
# Observed data
observationSequence$Y[,1:6]
# Hidden states path
head(observationSequence$X)
```

---

initGHMM *Random Initialization for a Hidden Markov Model with emissions modeled as continuous variables*

Description

Function used to generate a hidden Markov model with continuous variables and random parameters. This method allows using the univariate version of a Gaussian Mixture Model when setting m = 1. The code for the methods with categorical values or discrete data can be viewed in "initHMM" and "initPHMM", respectively.

Usage

```
initGHMM(n,m)
```

Arguments

n	the number of hidden states to use.
m	the number of variables generated by the hidden states (Dimensionality of the bbserved vector).

Value

A "list" that contains the required values to specify the model.

Model	it specifies that the observed values are to be modeled as a Gaussian mixture model.
StateNames	the set of hidden state names.
A	the transition probabilities matrix.
Mu	a matrix of means of the observed variables (rows) in each states (columns).
Sigma	a 3D matrix that has the covariance matrix of each state. The number of slices is equal to the maximum number of hidden states.
Pi	the initial probability vector.

References

Cited references are listed on the RcppHMM manual page.

Examples

```
n <- 3
m <- 5
model <- initGHMM(n, m)
print(model)
```

`initHMM` *Random Initialization for a Hidden Markov Model with emissions modeled as categorical variables*

Description

Function used to generate a hidden Markov model with categorical variables and random parameters. The code for the methods with continuous values or discrete data can be viewed in "`initGHMM`" and "`initPHMM`", respectively.

Usage

```
initHMM(n, m)
```

Arguments

`n`	the number of hidden states to use.
`m`	the number of possible categories (labels) generated by the hidden states.

Value

A "`list`" that contains the required values to specify the model.

`Model`	it specifies that the observed values are to be modeled as a multinomial distribution.
`StateNames`	the set of hidden state names.
`ObservationNames`	the set of possible observed values.
`A`	the transition probabilities matrix.
`B`	the emission probabilities matrix.
`Pi`	the initial probability vector.

References

Cited references are listed on the RcppHMM manual page.

Examples

```
n <- 2
m <- 2
model <- initHMM(n,m)
print(model)
```

`initPHMM` *Random Initialization for a Hidden Markov Model with emissions modeled as discrete variables*

Description

Function used to generate a hidden Markov model with discrete observations and random parameters. This model is used when the observed data are counts that can be modelled with a mixture of Poissons. The code for the methods with categorical values or continuous data can be viewed in "initHMM" and "initGHMM", respectively.

Usage

```
initPHMM(n)
```

Arguments

n	the number of hidden states to use.

Value

A "list" that contains all the required values to specify the model.

Model	it specifies that the observed values are to be modeled as a Poisson mixture model.
StateNames	the set of hidden state names.
A	the transition probabilities matrix.
B	a vector with the lambda parameter for each Poisson distribution.
Pi	the initial probability vector.

References

Cited references are listed on the RcppHMM manual page.

Examples

```
n <- 2
model <- initPHMM(n)
print(model)
```

`learnEM` *Expectation-Maximization algorithm to estimate the model parameters*

Description

Expectation-Maximization algorithm to estimate the model parameters based on a single or multiple observed sequences.

Usage

```
learnEM(hmm, sequences, iter = 100, delta = 1e-05, pseudo = 0, print = TRUE )
```

Arguments

`hmm`	a list with the necessary variables to define a hidden Markov model.
`sequences`	sequences of observations to be used as training set. HMM and PHMM use a matrix. GHMM uses a 3D array.
`iter`	a value that sets the maximum number of iterations to run.
`delta`	a value set to be the minimum error considered as a convergence criteria.
`pseudo`	a value set to consider pseudo-counts.
`print`	a logical value to print the error at each iteration.

Details

This function can be used for univariate or multivariate distributions. HMM and PHMM use a matrix with different sequences as rows and consecutive observations in the columns. GHMM uses an array with the variables as rows, consecutive observations in the columns and different sequences as slices.

Value

A `"list"` that contains the estimated hidden Markov model parameters.

References

Cited references are listed on the RcppHMM manual page.

See Also

`generateObservations` , `verifyModel`

Examples

```
## Values for a hidden Markov model with categorical observations
# Set the model parameters
n <- c("First","Second")
m <- c("A","T","C","G")
A <- matrix(c(0.8,0.2,
              0.1,0.9),
            nrow = 2,
            byrow = TRUE)

B <- matrix(c(0.2, 0.2, 0.3, 0.3,
              0.4, 0.4, 0.1, 0.1),
            nrow = 2,
            byrow = TRUE)

Pi <- c(0.5, 0.5)

params <- list( "Model" = "HMM",
                "StateNames" = n,
                "ObservationNames" = m,
                "A" = A,
                "B" = B,
                "Pi" = Pi)

HMM <- verifyModel(params)

# Data simulation
set.seed(100)
length <- 100
seqs <- 10

# Multiple sequences to be used as training set
observationSequences<- c()
for(i in 1:seqs){
  Y <- generateObservations(HMM , length)$Y
  observationSequences <- rbind(observationSequences , Y)
}

# New model random initialization
# Model to be trained
set.seed(1000)
newModel <- initHMM(2,4)
n = c("X1","X2")
m = c("A","T","C","G")
newModel <- setNames(newModel,
                     list( "StateNames" = n,
                           "ObservationNames" = m) )


  newModel <- learnEM(newModel,
                      observationSequences,
                      iter= 50,
```

```
                      delta = 1E-5,
                      pseudo = 3,
                      print = TRUE)


print(newModel)

## Values for a hidden Markov model with discrete observations

n <- c("Low","Normal","High")

A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol=length(n), byrow=TRUE)

B <- c(2600,  # First distribution with mean 2600
       2700,  # Second distribution with mean 2700
       2800)  # Third distribution with mean 2800

Pi <- rep(1/length(n), length(n))

HMM.discrete <- verifyModel(list("Model"="PHMM", "StateNames" = n, "A" = A, "B" = B, "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
seqs <- 50

# Multiple sequences to be evaluated
observationSequences<- c()
for(i in 1:seqs){
  Y <- generateObservations(HMM.discrete , length)$Y
  observationSequences <- rbind(observationSequences , Y)
}

dim(observationSequences)
# New model random initialization
# Model to be trained
set.seed(1000)
newModel <- initPHMM(3)

  newModel <- learnEM(newModel,
                      observationSequences,
                      iter= 50,
                      delta = 1E-5,
                      print = FALSE)


print(newModel)


## Values for a hidden Markov model with continuous observations
```

```
# Number of hidden states = 3
# Univariate gaussian mixture model

N = c("Low","Normal", "High")
A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol= length(N), byrow = TRUE)

Mu <- matrix(c(0, 50, 100), ncol = length(N))
Sigma <- array(c(144, 400, 100), dim = c(1,1,length(N)))
Pi <- rep(1/length(N), length(N))

HMM.cont.univariate <- verifyModel(list( "Model"="GHMM",
                                "StateNames" = N,
                                "A" = A,
                                "Mu" = Mu,
                                "Sigma" = Sigma,
                                "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
seqs <- 50

# Multiple sequences to be evaluated
observationSequences<- array(0, dim = c(1, length, seqs) )
for(i in 1:seqs){
  Y <- generateObservations(HMM.cont.univariate , length)$Y
  observationSequences[,,i] <- Y
}

dim(observationSequences)

# New model random initialization
# Model to be trained
set.seed(1000)
newModel <- initGHMM(3)

  newModel <- learnEM(newModel,
                      observationSequences,
                      iter= 50,
                      delta = 1E-5,
                      print = FALSE)


print(newModel)


## Values for a hidden Markov model with continuous observations
# Number of hidden states = 2
# Multivariate gaussian mixture model
# Observed vector with dimensionality of 3
```

```
N = c("X1","X2")
M <- 3

# Same number of dimensions
Sigma <- array(0, dim =c(M,M,length(N)))
Sigma[,,1] <- matrix(c(1.0,0.8,0.8,
                       0.8,1.0,0.8,
                       0.8,0.8,1.0), ncol = M,
                     byrow = TRUE)
Sigma[,,2] <- matrix(c(1.0,0.4,0.6,
                       0.4,1.0,0.8,
                       0.6,0.8,1.0), ncol = M,
                     byrow = TRUE)
Mu <- matrix(c(0, 5,
               10, 0,
               5, 10),
             nrow = M,
             byrow = TRUE)

A <- matrix(c(0.6,0.4,
              0.3, 0.7),
            ncol = length(N),
            byrow = TRUE)
Pi <- c(0.5, 0.5)

HMM.cont.multi <- verifyModel(list( "Model" = "GHMM",
                                    "StateNames" = N,
                                    "A" = A,
                                    "Mu" = Mu,
                                    "Sigma" = Sigma,
                                    "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
seqs <- 50

# Multiple sequences to be evaluated
observationSequences<- array(0, dim = c(M, length, seqs) )
for(i in 1:seqs){
  Y <- generateObservations(HMM.cont.multi , length)$Y
  observationSequences[,,i] <- Y
}

dim(observationSequences)

# New model random initialization
# Model to be trained
set.seed(1000)
newModel <- initGHMM(2, M)

  newModel <- learnEM(newModel,
                      observationSequences,
```

```
                    iter= 50,
                    delta = 1E-5,
                    print = FALSE)

print(newModel)
```

`loglikelihood` *Evaluation of multiple observed sequences given a model*

Description

This function computes the log-likelihood of multiple observed sequences generated by a hidden Markov model with fixed parameters.

Usage

```
loglikelihood(hmm, sequences)
```

Arguments

`hmm`	a list with the necessary variables to define a hidden Markov model.
`sequences`	sequences of observations to be evaluated. HMM and PHMM use a matrix. GHMM uses a 3D array.

Value

A value that represents the log-likelihood of the multiple observed sequences given the hiddden Markov model. HMM and PHMM use a matrix with different sequences as rows and consecutive observations in the columns. GHMM uses an array with the variables as rows, consecutive observations in the columns and different sequences as slices.

References

Cited references are listed on the RcppHMM manual page.

See Also

`generateObservations` , `verifyModel` , `evaluation`

Examples

```
## Values for a hidden Markov model with categorical observations
# Set the model parameters
n <- c("First","Second")
m <- c("A","T","C","G")
A <- matrix(c(0.8,0.2,
              0.1,0.9),
            nrow = 2,
```

```
            byrow = TRUE)

B <- matrix(c(0.2, 0.2, 0.3, 0.3,
              0.4, 0.4, 0.1, 0.1),
            nrow = 2,
            byrow = TRUE)

Pi <- c(0.5, 0.5)

params <- list( "Model" = "HMM",
                "StateNames" = n,
                "ObservationNames" = m,
                "A" = A,
                "B" = B,
                "Pi" = Pi)

HMM <- verifyModel(params)

# Data simulation
set.seed(100)
length <- 100
seqs <- 10

# Multiple sequences to be evaluated
observationSequences<- c()
for(i in 1:seqs){
  Y <- generateObservations(HMM , length)$Y
  observationSequences <- rbind(observationSequences , Y)
}

dim(observationSequences)

#Sequences evaluation
loglikelihood(HMM, observationSequences)


## Values for a hidden Markov model with discrete observations

n <- c("Low","Normal","High")

A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol=length(n), byrow=TRUE)

B <- c(2600,  # First distribution with mean 2600
       2700,  # Second distribution with mean 2700
       2800)  # Third distribution with mean 2800

Pi <- rep(1/length(n), length(n))

HMM.discrete <- verifyModel(list("Model"="PHMM", "StateNames" = n, "A" = A, "B" = B, "Pi" = Pi))
```

```
# Data simulation
set.seed(100)
length <- 100
seqs <- 10

# Multiple sequences to be evaluated
observationSequences<- c()
for(i in 1:seqs){
  Y <- generateObservations(HMM.discrete , length)$Y
  observationSequences <- rbind(observationSequences , Y)
}

dim(observationSequences)

#Sequences evaluation
loglikelihood(HMM.discrete, observationSequences)

## Values for a hidden Markov model with continuous observations
# Number of hidden states = 3
# Univariate gaussian mixture model

N = c("Low","Normal", "High")
A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol= length(N), byrow = TRUE)

Mu <- matrix(c(0, 50, 100), ncol = length(N))
Sigma <- array(c(144, 400, 100), dim = c(1,1,length(N)))
Pi <- rep(1/length(N), length(N))

HMM.cont.univariate <- verifyModel(list( "Model"="GHMM",
                               "StateNames" = N,
                               "A" = A,
                               "Mu" = Mu,
                               "Sigma" = Sigma,
                               "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
seqs <- 10

# Multiple sequences to be evaluated
observationSequences<- array(0, dim = c(1, length, seqs) )
for(i in 1:seqs){
  Y <- generateObservations(HMM.cont.univariate , length)$Y
  observationSequences[,,i] <- Y
}

dim(observationSequences)

#Sequences evaluation
```

```
loglikelihood(HMM.cont.univariate, observationSequences)


## Values for a hidden Markov model with continuous observations
# Number of hidden states = 2
# Multivariate gaussian mixture model
# Observed vector with dimensionality of 3
N = c("X1","X2")
M <- 3

# Same number of dimensions
Sigma <- array(0, dim =c(M,M,length(N)))
Sigma[,,1] <- matrix(c(1.0,0.8,0.8,
                       0.8,1.0,0.8,
                       0.8,0.8,1.0), ncol = M,
                     byrow = TRUE)
Sigma[,,2] <- matrix(c(1.0,0.4,0.6,
                       0.4,1.0,0.8,
                       0.6,0.8,1.0), ncol = M,
                     byrow = TRUE)
Mu <- matrix(c(0, 5,
               10, 0,
               5, 10),
             nrow = M,
             byrow = TRUE)

A <- matrix(c(0.6,0.4,
              0.3, 0.7),
            ncol = length(N),
            byrow = TRUE)
Pi <- c(0.5, 0.5)

HMM.cont.multi <- verifyModel(list( "Model" = "GHMM",
                                "StateNames" = N,
                                "A" = A,
                                "Mu" = Mu,
                                "Sigma" = Sigma,
                                "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
seqs <- 10

# Multiple sequences to be evaluated
observationSequences<- array(0, dim = c(M, length, seqs) )
for(i in 1:seqs){
  Y <- generateObservations(HMM.cont.multi , length)$Y
  observationSequences[,,i] <- Y
}

dim(observationSequences)
```

```
#Sequences evaluation
loglikelihood(HMM.cont.multi, observationSequences)
```

---

setNames *Set the names of the model*

Description

Function used to set new hidden state names to the model. If it is a categorical model, it also sets the labels for the observations. This function verifies that all the parameters and new names agree in size.

Usage

```
setNames(hmm , names)
```

Arguments

hmm	a list with the necessary variables to define a hidden Markov model.
names	a list with the new names to be set in the model.

Value

A "list" that contains the verified hidden Markov model parameters.

Examples

```
## Values for a hidden Markov model with categorical observations

set.seed(1000)
newModel <- initHMM(2,4)
n <- c("First","Second")
m <- c("A","T","C","G")
newModel <- setNames(newModel,
                     list( "StateNames" = n,
                           "ObservationNames" = m) )

## Values for a hidden Markov model with continuous observations

set.seed(1000)
newModel <- initGHMM(3)
n <- c("Low",  "Normal", "High" )
newModel <- setNames(newModel,
                      list( "StateNames" = n))

## Values for a hidden Markov model with discrete observations

set.seed(1000)
```

```
newModel <- initPHMM(3)
n <- c("Low",  "Normal", "High" )
newModel <- setNames(newModel,
                     list( "StateNames" = n))
```

`setParameters` *Set the model parameters*

Description

Function used to set the model parameters. This function verifies that parameters and names correspond.

Usage

```
setParameters(hmm , params)
```

Arguments

`hmm`	a list with the necessary variables to define a hidden Markov model.
`params`	a list with the new parameters to be set in the model.

Value

A "`list`" that contains the verified hidden Markov model parameters.

Examples

```
## Values for a hidden Markov model with categorical observations

set.seed(1000)
newModel <- initHMM(2,4)

A <- matrix(c(0.378286,0.621714,
              0.830970,0.169030),
            nrow = 2,
            byrow = TRUE)

B <- matrix(c(0.1930795, 0.2753869, 0.3463100, 0.1852237,
              0.2871577, 0.1848870, 0.1614925, 0.3664628),
            nrow = 2,
            byrow = TRUE)

Pi <- c(0.4757797, 0.5242203)

newModel <- setParameters(newModel,
                          list( "A" = A,
                                "B" = B,
```

```
                                "Pi" = Pi) )

## Values for a hidden Markov model with discrete observations

set.seed(1000)
n <- 3
newModel <- initPHMM(n)

A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol=n, byrow=TRUE)

B <- c(2600,  # First distribution with mean 2600
       2700,  # Second distribution with mean 2700
       2800)  # Third distribution with mean 2800

Pi <- rep(1/n , n)


newModel <- setParameters(newModel,
                          list( "A" = A,
                                "B" = B,
                                "Pi" = Pi) )

## Values for a hidden Markov model with continuous observations
# Number of hidden states = 3
# Univariate gaussian mixture model
N <- 3
newModel <- initGHMM(N)

A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol= N, byrow = TRUE)

Mu <- matrix(c(0, 50, 100), ncol = N)
Sigma <- array(c(144, 400, 100), dim = c(1,1,N))
Pi <- rep(1/N, N)

newModel <- setParameters(newModel,
                          list( "A" = A,
                                "Mu" = Mu,
                                "Sigma" = Sigma,
                                "Pi" = Pi))


## Values for a hidden Markov model with continuous observations
# Number of hidden states = 2
# Multivariate gaussian mixture model
# Observed vector with dimensionality of 3
N <- 2
M <- 3
```

```
set.seed(100)
newModel <- initGHMM(N,M)

# Same number of dimensions
Sigma <- array(0, dim =c(M,M,N))
Sigma[,,1] <- matrix(c(1.0,0.8,0.8,
                       0.8,1.0,0.8,
                       0.8,0.8,1.0), ncol = M,
                     byrow = TRUE)
Sigma[,,2] <- matrix(c(1.0,0.4,0.6,
                       0.4,1.0,0.8,
                       0.6,0.8,1.0), ncol = M,
                     byrow = TRUE)
Mu <- matrix(c(0, 5,
               10, 0,
               5, 10),
             nrow = M,
             byrow = TRUE)

A <- matrix(c(0.6,0.4,
              0.3, 0.7),
            ncol = N,
            byrow = TRUE)
Pi <- c(0.5, 0.5)

newModel <- setParameters(newModel,
                          list( "A" = A,
                                "Mu" = Mu,
                                "Sigma" = Sigma,
                                "Pi" = Pi))
```

`verifyModel` *Model parameter verification*

Description

Function used to verify that all the parameters satisfy the model constraints.

Usage

```
verifyModel(model)
```

Arguments

`model`	a list with the necessary parameters to set a hidden Markov model with fixed values.

Details

The model must have a stochastic transition matrix and a stochastic initial probability vector, also the row and column sizes must coincide with the number of provided state names. If the model uses categorical values, the emission matrix also must be stochastic and must have a column for each observation label and a row for each state name. And if the model uses discrete data, all the values must be positive assuming that they are counts.

Value

A "list" that contains the verified hidden Markov model parameters.

Examples

```
## Values for a hidden Markov model with categorical observations

n <- c("First","Second")
m <- c("A","T","C","G")
A <- matrix(c(0.378286,0.621714,
              0.830970,0.169030),
            nrow = 2,
            byrow = TRUE)

B <- matrix(c(0.1930795, 0.2753869, 0.3463100, 0.1852237,
              0.2871577, 0.1848870, 0.1614925, 0.3664628),
            nrow = 2,
            byrow = TRUE)

Pi <- c(0.4757797, 0.5242203)


params <- list( "Model" = "HMM",
                "StateNames" = n,
                "ObservationNames" = m,
                "A" = A,
                "B" = B,
                "Pi" = Pi)

verifiedModel <- verifyModel(params)
print(verifiedModel)

## Values for a hidden Markov model with discrete observations

n <- c("Low","Normal","High")

A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol=length(n), byrow=TRUE)

B <- c(2600,  # First distribution with mean 2600
       2700,  # Second distribution with mean 2700
       2800)  # Third distribution with mean 2800
```

```
Pi <- rep(1/length(n), length(n))

HMM.discrete <- verifyModel(list("Model"="PHMM", "StateNames" = n, "A" = A, "B" = B, "Pi" = Pi))
print(HMM.discrete)

## Values for a hidden Markov model with continuous observations
# Number of hidden states = 3
# Univariate gaussian mixture model

N = c("Low","Normal", "High")
A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol= length(N), byrow = TRUE)

Mu <- matrix(c(0, 50, 100), ncol = length(N))
Sigma <- array(c(144, 400, 100), dim = c(1,1,length(N)))
Pi <- rep(1/length(N), length(N))

newModel <- verifyModel(list( "Model"="GHMM",
                              "StateNames" = N,
                              "A" = A,
                              "Mu" = Mu,
                              "Sigma" = Sigma,
                              "Pi" = Pi))


## Values for a hidden Markov model with continuous observations
# Number of hidden states = 2
# Multivariate gaussian mixture model
# Observed vector with dimensionality of 3
N = c("X1","X2")
M <- 3

# Same number of dimensions
Sigma <- array(0, dim =c(M,M,length(N)))
Sigma[,,1] <- matrix(c(1.0,0.8,0.8,
                       0.8,1.0,0.8,
                       0.8,0.8,1.0), ncol = M)
Sigma[,,2] <- matrix(c(1.0,0.4,0.6,
                       0.4,1.0,0.8,
                       0.6,0.8,1.0), ncol = M)
Mu <- matrix(c(0, 5,
               10, 0,
               5, 10), nrow = M)

A <- matrix(c(0.6,0.3,
              0.4, 0.7), ncol = length(N))
Pi <- c(0.5, 0.5)

newModel <- verifyModel(list( "Model" = "GHMM",
                                "StateNames" = N,
```

```
"A" = A,
"Mu" = Mu,
"Sigma" = Sigma,
"Pi" = Pi))
```

`viterbi` *Viterbi algorithm for hidden state decoding*

Description

Function used to get the most likely path of hidden states generated by the observed sequence.

Usage

```
viterbi(hmm, sequence)
```

Arguments

`hmm`	a list with the necessary variables to define a hidden Markov model.
`sequence`	sequence of observations to be decoded. HMM and PHMM use a vector. GHMM uses a matrix.

Details

The Viterbi algorithm is based in a greedy approach, therefore it would only the give the most probable path. GHMM uses a matrix with the variables as rows and consecutive observations in the columns.

Value

A vector with the path of hidden states that generated the observed sequence.

References

Cited references are listed on the RcppHMM manual page.

See Also

`generateObservations` , `verifyModel` , `forwardBackward`

Examples

```
## Values for a hidden Markov model with categorical observations
# Set the model parameters
n <- c("First","Second")
m <- c("A","T","C","G")
A <- matrix(c(0.8,0.2,
              0.1,0.9),
            nrow = 2,
            byrow = TRUE)

B <- matrix(c(0.2, 0.2, 0.3, 0.3,
              0.4, 0.4, 0.1, 0.1),
            nrow = 2,
            byrow = TRUE)

Pi <- c(0.5, 0.5)

params <- list( "Model" = "HMM",
                "StateNames" = n,
                "ObservationNames" = m,
                "A" = A,
                "B" = B,
                "Pi" = Pi)

HMM <- verifyModel(params)

# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM, length)

#Sequence decoding
hiddenStates <- viterbi(HMM, observationSequence$Y)
print(hiddenStates)

## Values for a hidden Markov model with discrete observations

n <- c("Low","Normal","High")

A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol=length(n), byrow=TRUE)

B <- c(2600,  # First distribution with mean 2600
       2700,  # Second distribution with mean 2700
       2800)  # Third distribution with mean 2800

Pi <- rep(1/length(n), length(n))

HMM.discrete <- verifyModel(list("Model"="PHMM", "StateNames" = n, "A" = A, "B" = B, "Pi" = Pi))
```

```
# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM.discrete, length)

#Sequence decoding
hiddenStates <- viterbi(HMM.discrete, observationSequence$Y)
print(hiddenStates)

## Values for a hidden Markov model with continuous observations
# Number of hidden states = 3
# Univariate gaussian mixture model

N = c("Low","Normal", "High")
A <- matrix(c(0.5, 0.3,0.2,
              0.2, 0.6, 0.2,
              0.1, 0.3, 0.6),
            ncol= length(N), byrow = TRUE)

Mu <- matrix(c(0, 50, 100), ncol = length(N))
Sigma <- array(c(144, 400, 100), dim = c(1,1,length(N)))
Pi <- rep(1/length(N), length(N))

HMM.cont.univariate <- verifyModel(list( "Model"="GHMM",
                              "StateNames" = N,
                              "A" = A,
                              "Mu" = Mu,
                              "Sigma" = Sigma,
                              "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM.cont.univariate, length)

#Sequence decoding
hiddenStates <- viterbi(HMM.cont.univariate, observationSequence$Y)
print(hiddenStates)

## Values for a hidden Markov model with continuous observations
# Number of hidden states = 2
# Multivariate gaussian mixture model
# Observed vector with dimensionality of 3
N = c("X1","X2")
M <- 3

# Same number of dimensions
Sigma <- array(0, dim =c(M,M,length(N)))
Sigma[,,1] <- matrix(c(1.0,0.8,0.8,
                       0.8,1.0,0.8,
                       0.8,0.8,1.0), ncol = M,
                     byrow = TRUE)
Sigma[,,2] <- matrix(c(1.0,0.4,0.6,
```

```
                      0.4,1.0,0.8,
                      0.6,0.8,1.0), ncol = M,
                    byrow = TRUE)
Mu <- matrix(c(0, 5,
               10, 0,
               5, 10),
             nrow = M,
             byrow = TRUE)

A <- matrix(c(0.6, 0.4,
              0.3, 0.7),
            ncol = length(N),
            byrow = TRUE)
Pi <- c(0.5, 0.5)

HMM.cont.multi <- verifyModel(list( "Model" = "GHMM",
                              "StateNames" = N,
                              "A" = A,
                              "Mu" = Mu,
                              "Sigma" = Sigma,
                              "Pi" = Pi))

# Data simulation
set.seed(100)
length <- 100
observationSequence <- generateObservations(HMM.cont.multi, length)

#Sequence decoding
hiddenStates <- viterbi(HMM.cont.multi, observationSequence$Y)
print(hiddenStates)
```

Index