Package 'HMM'

May 16, 2025

Type Package

Version 1.0.2

Title Hidden Markov Models

Date 2025-05-15

Maintainer Lin Himmelmann <hmm@linhi.de>

Depends R (>= 2.0.0)

Description Easy to use library to setup, apply and make inference with discrete time and discrete space Hidden Markov Models.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Lin Himmelmann [aut, cre]

Repository CRAN

Date/Publication 2025-05-16 08:50:32 UTC

Contents

backward *Computes the backward probabilities*

Description

The backward-function computes the backward probabilities. The backward probability for state X and observation at time k is defined as the probability of observing the sequence of observations e_k+1, ... ,e_n under the condition that the state at time k is X. That is:
b[X,k] := Prob(E_k+1 = e_k+1, ... , E_n = e_n | X_k = X).
Where E_1...E_n = e_1...e_n is the sequence of observed emissions and X_k is a random variable that represents the state at time k.

Usage

```
backward(hmm, observation)
```

Arguments

hmm	A Hidden Markov Model.
observation	A sequence of observations.

Format

Dimension and Format of the Arguments.

hmm A valid Hidden Markov Model, for example instantiated by initHMM.

observation A vector of strings with the observations.

Value

Return Value:

backward	A matrix containing the backward probabilities. The probabilities are given on a logarithmic scale (natural logarithm). The first dimension refers to the state and the second dimension to time.

Author(s)

Lin Himmelmann <hmm@linhi.com>, Scientific Software Development

References

Lawrence R. Rabiner: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE 77(2) p.257-286, 1989.

See Also

See forward for computing the forward probabilities.

Examples

```
# Initialise HMM
hmm = initHMM(c("A","B"), c("L","R"), transProbs=matrix(c(.8,.2,.2,.8),2),
emissionProbs=matrix(c(.6,.4,.4,.6),2))
print(hmm)
# Sequence of observations
observations = c("L","L","R","R")
# Calculate backward probablities
logBackwardProbabilities = backward(hmm,observations)
print(exp(logBackwardProbabilities))
```

`baumWelch` *Inferring the parameters of a Hidden Markov Model via the Baum-Welch algorithm*

Description

For an initial Hidden Markov Model (HMM) and a given sequence of observations, the Baum-Welch algorithm infers optimal parameters to the HMM. Since the Baum-Welch algorithm is a variant of the Expectation-Maximisation algorithm, the algorithm converges to a local solution which might not be the global optimum.

Usage

```
baumWelch(hmm, observation, maxIterations=100, delta=1E-9, pseudoCount=0)
```

Arguments

`hmm`	A Hidden Markov Model.
`observation`	A sequence of observations.
`maxIterations`	The maximum number of iterations in the Baum-Welch algorithm.
`delta`	Additional termination condition, if the transition and emission matrices converge, before reaching the maximum number of iterations (`maxIterations`). The difference of transition and emission parameters in consecutive iterations must be smaller than `delta` to terminate the algorithm.
`pseudoCount`	Adding this amount of pseudo counts in the estimation-step of the Baum-Welch algorithm.

Format

Dimension and Format of the Arguments.

hmm A valid Hidden Markov Model, for example instantiated by `initHMM`.

observation A vector of observations.

Value

Return Values:

hmm	The inferred HMM. The representation is equivalent to the representation in initHMM.
difference	Vector of differences calculated from consecutive transition and emission matrices in each iteration of the Baum-Welch procedure. The difference is the sum of the distances between consecutive transition and emission matrices in the L2-Norm.

Author(s)

Lin Himmelmann <hmm@linhi.com>, Scientific Software Development

References

For details see: Lawrence R. Rabiner: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE 77(2) p.257-286, 1989.

See Also

See viterbiTraining.

Examples

```
# Initial HMM
hmm = initHMM(c("A","B"),c("L","R"),
transProbs=matrix(c(.9,.1,.1,.9),2),
emissionProbs=matrix(c(.5,.51,.5,.49),2))
print(hmm)
# Sequence of observation
a = sample(c(rep("L",100),rep("R",300)))
b = sample(c(rep("L",300),rep("R",100)))
observation = c(a,b)
# Baum-Welch
bw = baumWelch(hmm,observation,10)
print(bw$hmm)
```

---

dishonestCasino *Example application for Hidden Markov Models*

Description

The dishonest casino gives an example for the application of Hidden Markov Models. This example is taken from Durbin et. al. 1999: A dishonest casino uses two dice, one of them is fair the other is loaded. The probabilities of the fair die are (1/6,...,1/6) for throwing ("1",...,"6"). The probabilities of the loaded die are (1/10,...,1/10,1/2) for throwing ("1",..."5","6"). The observer doesn't know which die is actually taken (the state is hidden), but the sequence of throws (observations) can be used to infer which die (state) was used.

Usage

```
dishonestCasino()
```

Format

The function dishonestCasino has no arguments.

Value

The function dishonestCasino returns nothing.

Author(s)

Lin Himmelmann <hmm@linhi.com>, Scientific Software Development

References

Richard Durbin, Sean R. Eddy, Anders Krogh, Graeme Mitchison (1999). Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press. ISBN 0-521-62971-3.

Examples

```
# Dishonest casino example
dishonestCasino()
```

---

forward *Computes the forward probabilities*

Description

The forward-function computes the forward probabilities. The forward probability for state X up to observation at time k is defined as the probability of observing the sequence of observations e_1, ... ,e_k and that the state at time k is X. That is:
f[X,k] := Prob(E_1 = e_1, ... , E_k = e_k , X_k = X).
Where E_1...E_n = e_1...e_n is the sequence of observed emissions and X_k is a random variable that represents the state at time k.

Usage

```
forward(hmm, observation)
```

Arguments

hmm	A Hidden Markov Model.
observation	A sequence of observations.

Format

Dimension and Format of the Arguments.

hmm A valid Hidden Markov Model, for example instantiated by `initHMM`.

observation A vector of strings with the observations.

Value

Return Value:

`forward` A matrix containing the forward probabilities. The probabilities are given on a logarithmic scale (natural logarithm). The first dimension refers to the state and the second dimension to time.

Author(s)

Lin Himmelmann <hmm@linhi.com>, Scientific Software Development

References

Lawrence R. Rabiner: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE 77(2) p.257-286, 1989.

See Also

See `backward` for computing the backward probabilities.

Examples

```
# Initialise HMM
hmm = initHMM(c("A","B"), c("L","R"), transProbs=matrix(c(.8,.2,.2,.8),2),
emissionProbs=matrix(c(.6,.4,.4,.6),2))
print(hmm)
# Sequence of observations
observations = c("L","L","R","R")
# Calculate forward probablities
logForwardProbabilities = forward(hmm,observations)
print(exp(logForwardProbabilities))
```

`HMM` *HMM - Hidden Markov Models*

Description

Modelling, analysis and inference with discrete time and discrete space Hidden Markov Models.

Details

Package:	HMM - Rpackage
Type:	Package
Version:	1.0
Date:	2010-01-10
License:	GPL version 2 or later
Maintainer:	Scientific Software Development - Dr. Lin Himmelmann, www.linhi.com
URL:	www.linhi.com

`initHMM` *Initialisation of HMMs*

Description

This function initialises a general discrete time and discrete space Hidden Markov Model (HMM). A HMM consists of an alphabet of states and emission symbols. A HMM assumes that the states are hidden from the observer, while only the emissions of the states are observable. The HMM is designed to make inference on the states through the observation of emissions. The stochastics of the HMM is fully described by the initial starting probabilities of the states, the transition probabilities between states and the emission probabilities of the states.

Usage

```
initHMM(States, Symbols, startProbs=NULL, transProbs=NULL, emissionProbs=NULL)
```

Arguments

`States`	Vector with the names of the states.
`Symbols`	Vector with the names of the symbols.
`startProbs`	Vector with the starting probabilities of the states.
`transProbs`	Stochastic matrix containing the transition probabilities between the states.
`emissionProbs`	Stochastic matrix containing the emission probabilities of the states.

Format

Dimension and Format of the Arguments.

States Vector of strings.

Symbols Vector of strings.

startProbs Vector with the starting probabilities of the states. The entries must sum to 1.

transProbs `transProbs` is a (number of states)x(number of states)-sized matrix, which contains the transition probabilities between states. The entry `transProbs[X,Y]` gives the probability of a transition from state `X` to state `Y`. The rows of the matrix must sum to 1.

emissionProbs `emissionProbs` is a (number of states)x(number of states)-sized matrix, which contains the emission probabilities of the states. The entry `emissionProbs[X,e]` gives the probability of emission `e` from state `X`. The rows of the matrix must sum to 1.

Details

In `transProbs` and `emissionProbs` NA's can be used in order to forbid specific transitions and emissions. This might be useful for Viterbi training or the Baum-Welch algorithm when using pseudocounts.

Value

The function `initHMM` returns a HMM that consists of a list of 5 elements:

`States`	Vector with the names of the states.
`Symbols`	Vector with the names of the symbols.
`startProbs`	Annotated vector with the starting probabilities of the states.
`transProbs`	Annotated matrix containing the transition probabilities between the states.
`emissionProbs`	Annotated matrix containing the emission probabilities of the states.

Author(s)

Lin Himmelmann <hmm@linhi.com>, Scientific Software Development

References

For an introduction in the HMM-literature see for example:

- Lawrence R. Rabiner: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE 77(2) p.257-286, 1989.
- Olivier Cappe, Eric Moulines, Tobias Ryden: Inference in Hidden Markov Models. Springer. ISBN 0-387-40264-0.
- Ephraim Y., Merhav N.: Hidden Markov processes. IEEE Trans. Inform. Theory 48 p.1518-1569, 2002.

See Also

See `simHMM` to simulate a path of states and observations from a Hidden Markov Model.

Examples

```
# Initialise HMM nr.1
initHMM(c("X","Y"), c("a","b","c"))
# Initialise HMM nr.2
initHMM(c("X","Y"), c("a","b"), c(.3,.7), matrix(c(.9,.1,.1,.9),2),
        matrix(c(.3,.7,.7,.3),2))
```

`posterior` *Computes the posterior probabilities for the states*

Description

This function computes the posterior probabilities of being in state X at time k for a given sequence of observations and a given Hidden Markov Model.

Usage

```
posterior(hmm, observation)
```

Arguments

`hmm`	A Hidden Markov Model.
`observation`	A sequence of observations.

Format

Dimension and Format of the Arguments.

hmm A valid Hidden Markov Model, for example instantiated by `initHMM`.

observation A vector of observations.

Details

The posterior probability of being in a state X at time k can be computed from the `forward` and `backward` probabilities:
`Ws(X_k = X | E_1 = e_1, ... , E_n = e_n) = f[X,k] * b[X,k] / Prob(E_1 = e_1, ... , E_n = e_n)`
Where `E_1...E_n = e_1...e_n` is the sequence of observed emissions and `X_k` is a random variable that represents the state at time `k`.

Value

Return Values:

`posterior`	A matrix containing the posterior probabilities. The first dimension refers to the state and the second dimension to time.

Author(s)

Lin Himmelmann <hmm@linhi.com>, Scientific Software Development

References

Lawrence R. Rabiner: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE 77(2) p.257-286, 1989.

See Also

See forward for computing the forward probabilities and backward for computing the backward probabilities.

Examples

```
# Initialise HMM
hmm = initHMM(c("A","B"), c("L","R"), transProbs=matrix(c(.8,.2,.2,.8),2),
emissionProbs=matrix(c(.6,.4,.4,.6),2))
print(hmm)
# Sequence of observations
observations = c("L","L","R","R")
# Calculate posterior probablities of the states
posterior = posterior(hmm,observations)
print(posterior)
```

---

simHMM *Simulate states and observations for a Hidden Markov Model*

Description

Simulates a path of states and observations for a given Hidden Markov Model.

Usage

```
simHMM(hmm, length)
```

Arguments

hmm	A Hidden Markov Model.
length	The length of the simulated sequence of observations and states.

Format

Dimension and Format of the Arguments.

hmm A valid Hidden Markov Model, for example instantiated by initHMM.

Value

The function simHMM returns a path of states and associated observations:

states	The path of states.
observations	The sequence of observations.

Author(s)

Lin Himmelmann <hmm@linhi.com>, Scientific Software Development

See Also

See initHMM for instantiation of Hidden Markov Models.

Examples

```
# Initialise HMM
hmm = initHMM(c("X","Y"),c("a","b","c"))
# Simulate from the HMM
simHMM(hmm, 100)
```

`viterbi` *Computes the Most Probable Path of States*

Description

The Viterbi algorithm computes the most probable path of states for a sequence of observations given a Hidden Markov Model.

Usage

```
viterbi(hmm, observation)
```

Arguments

`hmm`	A valid Hidden Markov Model, for example instantiated by initHMM.
`observation`	A vector of observations.

Value

A vector of strings containing the most probable path of states.

Author(s)

Lin Himmelmann <hmm@linhi.de>

References

Lawrence R. Rabiner: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE 77(2), pp. 257-286, 1989.

Examples

```
# Initialise HMM
hmm = initHMM(c("A","B"), c("L","R"), transProbs=matrix(c(.6,.4,.4,.6),2),
              emissionProbs=matrix(c(.6,.4,.4,.6),2))
print(hmm)
# Sequence of observations
observations = c("L","L","R","R")
# Calculate Viterbi path
viterbi = viterbi(hmm, observations)
print(viterbi)
```

`viterbiTraining` *Inferring the parameters of a Hidden Markov Model via Viterbi-training*

Description

For an initial Hidden Markov Model (HMM) and a given sequence of observations, the Viterbi-training algorithm infers optimal parameters to the HMM. Viterbi-training usually converges much faster than the Baum-Welch algorithm, but the underlying algorithm is theoretically less justified. Be careful: The algorithm converges to a local solution which might not be the optimum.

Usage

```
viterbiTraining(hmm, observation, maxIterations=100, delta=1E-9, pseudoCount=0)
```

Arguments

`hmm`	A Hidden Markov Model.
`observation`	A sequence of observations.
`maxIterations`	The maximum number of iterations in the Viterbi-training algorithm.
`delta`	Additional termination condition, if the transition and emission matrices converge, before reaching the maximum number of iterations (`maxIterations`). The difference of transition and emission parameters in consecutive iterations must be smaller than `delta` to terminate the algorithm.
`pseudoCount`	Adding this amount of pseudo counts in the estimation-step of the Viterbi-training algorithm.

Format

Dimension and Format of the Arguments.

hmm A valid Hidden Markov Model, for example instantiated by `initHMM`.

observation A vector of observations.

Value

Return Values:

hmm	The inferred HMM. The representation is equivalent to the representation in initHMM.
difference	Vector of differences calculated from consecutive transition and emission matrices in each iteration of the Viterbi-training. The difference is the sum of the distances between consecutive transition and emission matrices in the L2-Norm.

Author(s)

Lin Himmelmann <hmm@linhi.com>, Scientific Software Development

References

For details see: Lawrence R. Rabiner: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE 77(2) p.257-286, 1989.

See Also

See baumWelch.

Examples

```
# Initial HMM
hmm = initHMM(c("A","B"),c("L","R"),
transProbs=matrix(c(.9,.1,.1,.9),2),
emissionProbs=matrix(c(.5,.51,.5,.49),2))
print(hmm)
# Sequence of observation
a = sample(c(rep("L",100),rep("R",300)))
b = sample(c(rep("L",300),rep("R",100)))
observation = c(a,b)
# Viterbi-training
vt = viterbiTraining(hmm,observation,10)
print(vt$hmm)
```

Index