Package: hmmTensor (via r-universe)

May 28, 2026

Type Package

Title Hidden Markov Model by Matrix and Tensor Decomposition

Version 0.1.0

Description Solves Hidden Markov Models (HMMs) via matrix and tensor decomposition. Converts observation sequences to co-occurrence matrices/tensors and applies Symmetric Non-negative Matrix Factorization (symNMF), Singular Value Decomposition (SVD), CANDECOMP/PARAFAC (CP) decomposition, or Tensor-Train (TT) decomposition to recover HMM parameters. Also provides standard HMM algorithms (Forward, Backward, Viterbi, Baum-Welch) for comparison. The spectral learning approach for HMMs is based on Hsu, Kakade, and Zhang (2012) <doi:10.1016/j.jcss.2011.12.025>. The symNMF method is described in Kuang, Yun, and Park (2015) <doi:10.1007/s10898-014-0247-2>. The Tensor-Train decomposition is described in Oseledets (2011) <doi:10.1137/090752286>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.5.0)

Imports rTensor, symTensor, methods, stats

Suggests testthat

RoxygenNote 7.3.2

NeedsCompilation no

Author Koki Tsuyuzaki [aut, cre]

Maintainer Koki Tsuyuzaki <k.t.the-answer@hotmail.co.jp>

Repository https://kokitsuyuzaki.r-universe.dev

Date/Publication 2026-05-27 09:10:02 UTC

RemoteUrl https://github.com/cran/hmmTensor

RemoteRef HEAD

RemoteSha be8657d954e97303151fce55019125bf37df7449

Contents

`Backward` *Backward Algorithm for HMM*

Description

Computes backward probabilities $\beta_t(i) = P(Y_{t+1}, \ldots, Y_T | X_t = i)$ using the same scaling factors from the Forward algorithm.

Usage

```
Backward(Y, T_mat, O, scale)
```

Arguments

`Y`	Integer vector of observations (values in 1:N)
`T_mat`	Transition matrix (K x K), `T_mat[i,j] = P(X_t=j | X_{t-1}=i)`
`O`	Emission matrix (K x N), `O[i,j] = P(Y_t=j | X_t=i)`
`scale`	Scaling factors from `Forward` (length T_len)

Value

Matrix (K x T_len) of scaled backward probabilities

`BaumWelch` *Baum-Welch Algorithm (EM) for HMM*

Description

Estimates HMM parameters (T, O, pi) from observation sequences using the Expectation-Maximization algorithm.

Usage

```
BaumWelch(
  Y,
  K,
  N,
  initT = NULL,
  initO = NULL,
  initPi = NULL,
  num.iter = 100L,
  thr = 1e-06,
  verbose = FALSE
)
```

Arguments

`Y`	Integer vector of observations (values in 1:N), or a list of integer vectors for multiple sequences.
`K`	Number of hidden states
`N`	Number of distinct observation symbols
`initT`	Initial transition matrix (K x K). If NULL, random initialization.
`initO`	Initial emission matrix (K x N). If NULL, random initialization.
`initPi`	Initial state distribution (length K). If NULL, uniform.
`num.iter`	Maximum EM iterations (default: 100)
`thr`	Convergence threshold on log-likelihood relative change (default: 1e-6)
`verbose`	Logical (default: FALSE)

Value

A list with components:

T_mat Estimated transition matrix (K x K)

O Estimated emission matrix (K x N)

pi0 Estimated initial distribution (length K)

loglik Vector of log-likelihoods per iteration

converged Logical

iter Number of iterations

Forward *Forward Algorithm for HMM*

Description

Computes forward probabilities $\alpha_t(i) = P(Y_1, \ldots, Y_t, X_t = i)$ with log-scaling to avoid underflow.

Usage

```
Forward(Y, T_mat, O, pi0)
```

Arguments

`Y`	Integer vector of observations (values in 1:N)
`T_mat`	Transition matrix (K x K), `T_mat[i,j] = P(X_t=j | X_{t-1}=i)`
`O`	Emission matrix (K x N), `O[i,j] = P(Y_t=j | X_t=i)`
`pi0`	Initial state distribution (length K)

Value

A list with components:

alpha Matrix (K x T_len) of scaled forward probabilities

loglik Log-likelihood of the observation sequence

scale Vector of scaling factors (length T_len)

HMM *HMM Parameter Estimation via Matrix/Tensor Decomposition*

Description

Estimates HMM parameters by decomposing the observation co-occurrence matrix/tensor. Supports multiple decomposition solvers.

Usage

```
HMM(
  Y,
  K,
  N = NULL,
  solver = c("symNMF", "SVD", "CP", "TT"),
  Beta = 2,
  order = 2L,
```

```
  lag = 1L,
  smooth = 1e-10,
  num.iter = 100L,
  thr = 1e-10,
  verbose = FALSE
)
```

Arguments

`Y`	Integer vector of observations (values in 1:N), or a list of integer vectors for multiple sequences.
`K`	Number of hidden states
`N`	Number of distinct observation symbols. If NULL, inferred from data.
`solver`	Decomposition method: `"symNMF"` Symmetric NMF via `symTensor::symNMF` (default) `"SVD"` Truncated SVD of the co-occurrence matrix `"CP"` CP decomposition of the 3rd-order co-occurrence tensor via `rTensor::cp` `"TT"` Tensor-Train approximation (via Tucker + reshape)
`Beta`	Beta-divergence parameter for symNMF (default: 2). Ignored for other solvers.
`order`	Co-occurrence order for Seq2Prob: 2 (default) or 3. `order = 3` is required for `solver = "CP"`.
`lag`	Lag for pairwise co-occurrence (default: 1)
`smooth`	Laplace smoothing pseudo-count (default: 1e-10)
`num.iter`	Maximum iterations for iterative solvers (default: 100)
`thr`	Convergence threshold (default: 1e-10)
`verbose`	Logical (default: FALSE)

Details

The pipeline:

1. Convert observations to co-occurrence matrix $\Omega = P_{2,1}$ via `Seq2Prob`
2. Decompose Ω using the chosen solver
3. Recover HMM parameters (T, O, pi) from the decomposition

For NMF-based methods, $\Omega \approx M \Theta M^\top$ where M relates to the emission matrix O and Θ to `diag(pi) %*% T`.

Value

A list with components:

T_mat Estimated transition matrix (K x K)

O Estimated emission matrix (K x N)

pi0 Estimated initial distribution (length K)

Omega Co-occurrence matrix/tensor used

decomp Raw decomposition result

solver Solver used

RecError Reconstruction error (if available)

Examples

```
set.seed(42)
toy <- toyModel(type = "simple")
result <- HMM(toy$Y, K = toy$K, N = toy$N, solver = "symNMF")
result$T_mat
```

`Seq2Prob` *Convert Observation Sequences to Co-occurrence Matrix/Tensor*

Description

Constructs empirical co-occurrence statistics from observation sequences. These form the basis for matrix/tensor decomposition approaches to HMM.

Usage

```
Seq2Prob(Y, N = NULL, order = 2L, lag = 1L, smooth = 0)
```

Arguments

`Y`	Integer vector of observations (values in 1:N), or a list of integer vectors for multiple sequences.
`N`	Number of distinct observation symbols. If NULL, inferred from data.
`order`	Co-occurrence order: 2 (pairwise, default) or 3 (triple). Order 2 gives an N x N matrix $P_{2,1}(i,j) = P(Y_2 = i, Y_1 = j)$. Order 3 gives an N x N x N tensor $P_3(i,j,k) = P(Y_3 = i, Y_2 = j, Y_1 = k)$.
`lag`	Lag for pairwise co-occurrence (default: 1). $\Omega^{(\tau)}(i,j) = P(Y_t = i, Y_{t+\tau} = j)$. Only used when `order` = 2.
`smooth`	Laplace smoothing pseudo-count (default: 0). Adds `smooth` to all counts before normalization.

Value

For `order` = 2: an N x N matrix (normalized). For `order` = 3: an rTensor Tensor object of dimension N x N x N.

Examples

```
Y <- c(1, 2, 3, 1, 2, 3, 1, 2, 3)
P2 <- Seq2Prob(Y, N = 3, order = 2)
P3 <- Seq2Prob(Y, N = 3, order = 3)
```

`toyModel` *Generate Toy HMM Data*

Description

Creates synthetic HMM data with visually clear structure for demonstrations and testing.

Usage

```
toyModel(type = c("simple", "weather", "leftright"), T_len = 500L, seed = NULL)
```

Arguments

`type`	Type of toy model: `"simple"` 2 states, 3 observations. States alternate with high probability. Clear emission separation. `"weather"` 3 states (Sunny/Cloudy/Rainy), 4 observations (Walk/Shop/Clean/Stay). Classic weather HMM. `"leftright"` 3 states with left-to-right transitions only. Models sequential processes.
`T_len`	Length of observation sequence (default: 500)
`seed`	Random seed (default: NULL)

Value

A list with components:

Y Integer vector of observations

X Integer vector of true hidden states

T_mat True transition matrix

O True emission matrix

pi0 True initial distribution

K Number of hidden states

N Number of observation symbols

type Model type

Examples

```
toy <- toyModel("simple", T_len = 200, seed = 42)
table(toy$X)
table(toy$Y)
```

`Viterbi` *Viterbi Algorithm for HMM*

Description

Finds the most likely state sequence given the observations using the Viterbi algorithm (log-space).

Usage

```
Viterbi(Y, T_mat, O, pi0)
```

Arguments

`Y`	Integer vector of observations (values in 1:N)
`T_mat`	Transition matrix (K x K), `T_mat[i,j] = P(X_t=j | X_{t-1}=i)`
`O`	Emission matrix (K x N), `O[i,j] = P(Y_t=j | X_t=i)`
`pi0`	Initial state distribution (length K)

Value

A list with components:

path Integer vector of most likely states (length T_len)

loglik Log-likelihood of the best path

Index