

Package ‘hsmm’

April 17, 2009

Version 0.3-5

Date 2008-06-15

Title Hidden Semi Markov Models

Author Jan Bulla <Jan.Bulla@mcs.vuw.ac.nz>, Ingo Bulla <ingobulla@gmx.de>, Oleg Nenadic <onenadi@uni-goettingen.de>

Maintainer Ingo Bulla <ingobulla@gmx.de>

Depends R (>= 2.0.0)

Description A package for computation of hidden semi markov models

License GPL

Repository CRAN

Date/Publication 2008-06-16 12:10:20

R topics documented:

hsmm	1
hsmm.sim	5
hsmm.smooth	7
hsmm.viterbi	9

Index	11
--------------	-----------

Description

Fitting a hidden semi-Markov model with conditional distribution `od` and runlength distribution `rd` to the observations `x`.

Usage

```
hsmm(x,
      od,
      od.par,
      rd      = "nonp",
      rd.par  = list(np = matrix(0.1, nrow = 10, ncol = 2)),
      pi.par  = c(0.5, 0.5),
      tpm.par = matrix(c(0, 1, 1, 0), 2),
      M      = NA,
      Q.max   = 500,
      epsilon = 1e-08,
      censoring = 1,
      prt     = TRUE,
      detailed = FALSE,
      r.lim   = c(0.01, 100),
      p.log.lim = c(0.001, 0.999),
      nu.lim  = c(0.01, 100))
```

Arguments

`x` The observations as a vector of length T .

`od` Character with the name of the conditional distribution of the observations. The following distributions are currently implemented:

```
"bern" = Bernoulli
"norm" = Normal
"pois" = Poisson
"t"    = Student-t
```

`rd` Character with the name of the runlength distribution (or sojourn time, dwell time distribution). The following distributions are currently implemented:

```
"nonp" = Non-parametric
"geom" = Geometric
"nbinom" = Negative Binomial
"log" = Logarithmic
"pois" = Poisson
```

<code>pi.par</code>	Vector of length J with the initial values for the initial probabilities of the semi-Markov chain.
<code>tpm.par</code>	Matrix of dimension $J \times J$ with the initial values for the transition probability matrix of the embedded Markov chain. The diagonal entries must all be zero; absorbing states are not permitted.
<code>rd.par</code>	List with the initial values for the parameters of the runlength distributions. See further details below (section 'List Objects rd.par and od.par').
<code>od.par</code>	List with the initial values for the parameters of the conditional observation distributions. See further details below (section 'List Objects rd.par and od.par').
<code>M</code>	Positive integer containing the maximum runlength.
<code>Q.max</code>	Positive integer containing the maximum number of iterations.
<code>epsilon</code>	Positive scalar giving the tolerance at which the relative change of log-likelihood is considered close enough to zero to terminate the algorithm.
<code>censoring</code>	Integer. If equal to 1, the last visited state contributes to the likelihood. If equal to 0, the partial likelihood estimator, which ignores the contribution of the last visited state, is used. For details see Guedon (2003).
<code>prt</code>	Logical. if TRUE, the log-likelihood and number of iterations carried out are printed for each iteration.
<code>detailed</code>	Logical. if TRUE, a list of the parameters at every iteration step is written into the <code>ctrl</code> list.
<code>r.lim</code>	Upper and lower bound for the r parameter of the negative binomial distribution in the M-step, bisection is applied to determine this parameter.
<code>p.log.lim</code>	Upper and lower bound for the parameter of the logarithmic distribution in the M-step, bisection is applied to determine this parameter.
<code>nu.lim</code>	Upper and lower bound for the degrees of freedom of parameter of the t distribution in the M-step, bisection is applied to determine this parameter.

Details

The function `hsmm` fits a hidden semi-Markov model using the EM algorithm for parameter estimation. The estimation algorithms are based on the right-censored approach initially described in Guedon (2003). This model does not assume that the last observation coincides with an exit from the last visited state. The EM algorithm is an iterative procedure and requires initial values. The results may depend on the initial values selected, because convergence to local maxima is a common phenomenon. Details on the algorithm utilized for the package `hsmm` are also presented by Bulla (2006).

Default model

The default model is a two-state HSMM with a non-parametric runlength distribution. Thus, the TPM does not require any initial values (for models with $J > 2$ states, the TPM may be initialized by the value $1/(J - 1)$ for the off-diagonal elements). The non-parametric runlength distribution is implemented as default distribution and initialized by a uniform distribution on the first ten runlengths. Similarly, the initial probabilities for π follow a uniform distribution. There is no default for the conditional distribution of the observations, because it should not be selected without investigating the data. We would like to point out that the non-parametric runlength distribution

often requires a very high number of observations. Sansom and Thomson (2001), e.g., obtained satisfactory results with series of length 20000 and longer.

Value

<code>call</code>	The matched call.
<code>iter</code>	Positive integer containing the number of iterations carried out.
<code>logl</code>	Double containing log-likelihood of the fitted model.
<code>para</code>	List object containing the parameter estimates.
<code>ctrl</code>	List object containing additional control variables. These are <code>solution.reached</code> , <code>error</code> , and <code>details</code> . <code>solution.reached</code> is TRUE, if the stopping criterion is fulfilled. <code>error</code> returns an error code: 0 = no error, 1 = internal probability less or equal to zero, 2 = memory exception, 3 = file error (internal output from C routine, disabled by default). <code>details</code> contains the parameter values of every iteration.

List Objects `rd.par` and `od.par`

The list objects `rd.par` and `od.par` contain parameter values for the runlength and conditional observation distribution, respectively. For a model with J states, the length of all parameter vectors is equal to J . For non-parametric runlength distribution, the corresponding entry is a matrix of dimension $M \times J$. The names of the list entries have to be as follows.

`od.par`:

```
"bern" (Bernoulli): "b"
"norm" (Normal):   "mean", "var"
"pois" (Poisson):  "lambda"
"t"     (Student-t): "mean", "var", "df"
```

`rd.par`:

```
"nonp" (Non-parametric): "np"
"geom" (Geometric):      "p"
"nbinom" (Negative Binomial): "r", "pi"
"log"   (Logarithmic):    "p"
"pois"  (Poisson):        "lambda"
```

References

- Bulla, J. (2006), Stylized facts of financial time series and hidden semi-Markov models. Ph.D. thesis, Goettingen.
- Guedon, Y. (2003), Estimating Hidden Semi-Markov Chains From Discrete Sequences. JCGS, 12 (3), pp 604-639.
- Sansom, J. and Thomson, P. (2001), Fitting hidden semi-Markov models to breakpoint rainfall data. J. Appl. Probab., 38A, pp 142-157

See Also

[hsmm.smooth](#), [hsmm.viterbi](#), [hsmm.sim](#)

Examples

```
# Simulating observations:
# (see hsmm.sim for details)
pipar <- rep(1/3, 3)
tpmpar <- matrix(c(0, 0.5, 0.5,
                  0.7, 0, 0.3,
                  0.8, 0.2, 0), 3, byrow = TRUE)
rdpar <- list(p = c(0.98, 0.98, 0.99))
odpar <- list(mean = c(-1.5, 0, 1.5), var=c(0.5, 0.6, 0.8))
sim <- hsmm.sim(n = 2000, od = "norm", rd = "log",
               pi.par = pipar, tpm.par = tpmpar,
               rd.par = rdpar, od.par = odpar, seed = 3539)

# Executing the EM algorithm:
fit <- hsmm(sim$obs, od = "norm", rd = "log",
            pi.par = pipar, tpm.par = tpmpar,
            od.par = odpar, rd.par = rdpar)

# The log-likelihood:
fit$logl

# The estimated parameters:
fit$para

# For comparison, the estimated parameters separately together with the true parameter value
# are given below.
# Transition probability matrix:
tpmpar
fit$para$tpm
# Observation distribution:
odpar
fit$para$od
# Runlength distribution:
rdpar
fit$para$rd
```

Description

Simulation of sequences of observations and underlying paths for hidden semi-Markov models.

Usage

```
hsmm.sim(n,
         od,
         rd,
         pi.par,
         tpm.par,
         od.par,
         rd.par,
         M = NA,
         seed = NULL)
```

Arguments

<code>n</code>	Positive integer containing the number of observations to simulate.
<code>od</code>	Character containing the name of the conditional distribution of the observations. For details see hsmm .
<code>rd</code>	Character containing the name of the runlength distribution (or sojourn time, dwell time distribution). See hsmm for details.
<code>pi.par</code>	Vector of length J containing the values for the initial probabilities of the semi-Markov chain.
<code>tpm.par</code>	Matrix of dimension $J \times J$ containing the parameter values for the transition probability matrix of the embedded Markov chain. The diagonal entries must all be zero, absorbing states are not permitted.
<code>rd.par</code>	List with the values for the parameters of the runlength distributions. For details see hsmm .
<code>od.par</code>	List with the values for the parameters of the conditional observation distributions. For details see hsmm .
<code>M</code>	Positive integer containing the maximum runlength.
<code>seed</code>	Seed for the random number generator (integer).

Details

The function `hsmm.sim` simulates the observations and the underlying state sequence of a hidden semi-Markov model. The simulation requires the specification of the runlength and the conditional observation distributions as well as all corresponding parameters.

Note: The simulation of t-distributed conditional observations is performed by the functions `rmt` and `rvm`, extracted from the package `csampling` and `CircStats`, respectively.

Value

<code>call</code>	The matched call.
<code>obs</code>	A vector of length n containing the simulated observations.
<code>path</code>	A vector of length n containing the simulated underlying semi-Markov chain.

See Also

[hsmm](#), [hsmm.smooth](#), [hsmm.viterbi](#)

Examples

```

# Simulation of sequences of observations and hidden states from a
# 3-state HSMM with a logarithmic runlength distribution and a
# conditional Gaussian distributions.

### Setting up the parameter values:
# Initial probabilities of the semi-Markov chain:
pipar <- rep(1/3, 3)
# Transition probabilities:
# (Note: For two states, the matrix degenerates, taking 0 for the
# diagonal and 1 for the off-diagonal elements.)
tpmpar <- matrix(c(0, 0.5, 0.5,
                  0.7, 0, 0.3,
                  0.8, 0.2, 0), 3, byrow = TRUE)
# Runlength distribution:
rdpar <- list(p = c(0.98, 0.98, 0.99))
# Observation distribution:
odpar <- list(mean = c(-1.5, 0, 1.5), var=c(0.5, 0.6, 0.8))

# Invoking the simulation:
sim <- hsmm.sim(n = 2000, od = "norm", rd = "log",
               pi.par = pipar, tpm.par = tpmpar,
               rd.par = rdpar, od.par = odpar, seed = 3539)

# The first 15 simulated observations:
round(sim$obs[1:15], 3)

# The first 15 simulated states:
sim$path[1:15]

```

hsmm.smooth

Hidden Semi-Markov Models

Description

Inference on the hidden states for given observations and model specifications of a hidden semi-Markov model. For every observation, the function calculates the probability of being in a particular state.

Usage

```

hsmm.smooth(x,
            od,
            rd,
            pi.par,
            tpm.par,
            od.par,
            rd.par,
            M = NA)

```

Arguments

<code>x</code>	The observations as a vector of length T .
<code>od</code>	Character containing the name of the conditional distribution of the observations. For details see hsmm .
<code>rd</code>	Character containing the name of the runlength distribution (or sojourn time, dwell time distribution). For details see hsmm .
<code>pi.par</code>	Vector of length J containing the values for the initial probabilities of the semi-Markov chain.
<code>tpm.par</code>	Matrix of dimension $J \times J$ containing the parameter values for the transition probability matrix of the embedded Markov chain. The diagonal entries must all be zero, absorbing states are not permitted.
<code>rd.par</code>	List with the values for the parameters of the runlength distributions. For details see hsmm .
<code>od.par</code>	List with the values for the parameters of the conditional observation distributions. For details see hsmm .
<code>M</code>	Positive integer containing the maximum runlength.

Details

The function `hsmm.smooth` calculates the so-called smoothed probabilities

$$P(S_t = i | X_1, \dots, X_T)$$

for all $t \in 1, \dots, T$ and $i \in 1, \dots, J$. This procedure is often termed 'local decoding'. The sequence of the most probable states follows directly. Note that this sequence is not necessarily the most probable state sequence, which is determined by the Viterbi algorithm. Also note that local decoding may ignore restrictions imposed by the transition probability matrix, such as forbidden transitions, because it optimizes locally for every single observation.

Value

<code>call</code>	The matched call.
<code>smooth.prob</code>	A matrix of dimension $J \times T$ containing the smoothing probabilities.
<code>path</code>	A vector of length T containing the sequence of the states with the highest probabilities.

See Also

[hsmm](#), [hsmm.sim](#), [hsmm.viterbi](#)

Examples

```
# Simulating observations:
# (see hsmm.sim for details)
pipar <- rep(1/3, 3)
tpmpar <- matrix(c(0, 0.5, 0.5,
                  0.7, 0, 0.3,
```

```

                                0.8, 0.2, 0), 3, byrow = TRUE)
rdpar <- list(p = c(0.98, 0.98, 0.99))
odpar <- list(mean = c(-1.5, 0, 1.5), var=c(0.5, 0.6, 0.8))
sim <- hsmm.sim(n = 2000, od = "norm", rd = "log",
               pi.par = pipar, tpm.par = tpmpar,
               rd.par = rdpar, od.par = odpar, seed = 3539)

# Computation of the smoothing probabilities:
fit.sm <- hsmm.smooth(sim$obs, od = "norm", rd = "log",
                     pi.par = pipar, tpm.par = tpmpar,
                     od.par = odpar, rd.par = rdpar)

# The first 15 smoothing probabilities:
round(fit.sm$sm[, 1:15], 2)

# The first 15 values of the resulting path:
fit.sm$path[1:15]

# For comparison, the real/simulated path (first 15 values):
sim$path[1:15]

```

hsmm.viterbi

Hidden Semi-Markov Models

Description

Inference on the hidden states for given observations and model specifications of a hidden semi-Markov model. The Viterbi algorithm determines the most probable sequence of hidden states.

Usage

```

hsmm.viterbi(x,
             od,
             rd,
             pi.par,
             tpm.par,
             od.par,
             rd.par,
             M = NA)

```

Arguments

- `x` The observed process, a vector of length T .
- `od` Character containing the name of the conditional distribution of the observations. For details see [hsmm](#).
- `rd` Character containing the name of the runlength distribution (or sojourn time, dwell time distribution). For details see [hsmm](#).

<code>pi.par</code>	Vector of length J containing the values for the initial probabilities of the semi-Markov chain.
<code>tpm.par</code>	Matrix of dimension $J \times J$ containing the parameter values for the transition probability matrix of the embedded Markov chain. The diagonal entries must all be zero, absorbing states are not permitted.
<code>rd.par</code>	List with the values for the parameters of the runlength distributions. For details see hsmm .
<code>od.par</code>	List with the values for the parameters of the conditional observation distributions. For details see hsmm .
<code>M</code>	Positive integer containing the maximum runlength.

Details

The function `hsmm.viterbi` carries out the Viterbi algorithm. It derives the most probable state sequence by a dynamic programming technique. This procedure is often termed 'global decoding'.

Value

<code>call</code>	The matched call.
<code>path</code>	Vector of length T containing the most probable path of the underlying states.

See Also

[hsmm](#), [hsmm.sim](#), [hsmm.smooth](#)

Examples

```
# Simulating observations:
# (see hsmm.sim for details)
pipar <- rep(1/3, 3)
tpmpar <- matrix(c(0, 0.5, 0.5,
                  0.7, 0, 0.3,
                  0.8, 0.2, 0), 3, byrow = TRUE)
rdpar <- list(p = c(0.98, 0.98, 0.99))
odpar <- list(mean = c(-1.5, 0, 1.5), var=c(0.5, 0.6, 0.8))
sim <- hsmm.sim(n = 2000, od = "norm", rd = "log",
               pi.par = pipar, tpm.par = tpmpar,
               rd.par = rdpar, od.par = odpar, seed = 3539)

# Executing the Viterbi algorithm:
fit.vi <- hsmm.viterbi(sim$obs, od = "norm", rd = "log",
                      pi.par = pipar, tpm.par = tpmpar,
                      od.par = odpar, rd.par = rdpar)

# The first 15 values of the resulting path:
fit.vi$path[1:15]

# For comparison, the real/simulated path (first 15 values):
sim$path[1:15]
```

Index

*Topic **ts**

hsmm, [1](#)

hsmm.sim, [5](#)

hsmm.smooth, [7](#)

hsmm.viterbi, [9](#)

hsmm, [1](#), [6–10](#)

hsmm.sim, [4](#), [5](#), [8](#), [10](#)

hsmm.smooth, [4](#), [6](#), [7](#), [10](#)

hsmm.viterbi, [4](#), [6](#), [8](#), [9](#)