

Package ‘stochmod’

October 26, 2009

Title Stochastic Modeling

Version 1.2.1

Author Artem Sokolov

Description Learning and inference algorithms for a variety of probabilistic models

Maintainer Artem Sokolov <artem.sokolov@gmail.com>

License GPL (>= 2)

Depends R (>= 2.4.0), mvtnorm

URL <http://www.cs.colostate.edu/~sokolov/stochmod.html>

Repository CRAN

Date/Publication 2009-10-26 20:27:41

R topics documented:

feat-lagdata	2
gmm-gendata	2
gmm-learn	3
gmm-ll	4
gmm-make	5
gmm-resp	6
hmm-fwbk	6
hmm-gendata	7
hmm-learn	8
hmm-ll	9
hmm-make	9
hmm-obs	10
lda-test	11
lda-train	11
qda-test	12
qda-train	13
sm-options	14
sm-output	14

Index**16**

 feat-lagdata *Data Lagging*

Description

Time-lagged embedding of a time series

Usage

```
FEAT.lagData( x, k=1 )
```

Arguments

x	Either a matrix or a list of matrices containing observation sequences, with one sample per row
k	Number of lags

Details

Time-delayed or time-lagged embedding appends a time-series to a lagged version of itself, thereby increasing dimensionality from p to $k*p$ and reducing the number of samples from n to $n-k$.

Value

$n-k$ by $k*p$ matrix of lagged time series

Author(s)

Artem Sokolov (Artem.Sokolov@gmail.com)

 gmm-gendata *Gaussian Mixture Models*

Description

Generate data from a provided model.

Usage

```
GMM.genData( N, gmm )
```

Arguments

N	Number of samples to generate
gmm	The model of dimensionality p

Details

Mixture coefficients are used to decide which component each datapoint is generated from. Component mean and covariance matrix is then employed to generate the actual sample.

Value

$N \times p$ matrix of generated data samples

Author(s)

Artem Sokolov <Artem.Sokolov@gmail.com>

gmm-learn

Gaussian Mixture Models

Description

Expectation Maximization algorithm for Gaussian Mixture Models

Usage

```
GMM.learn( xL, K, vL = NULL, gmm.init = NULL, cov.reg = 0.0, tol = 1e-03, LLstop = Inf, min.iter = 3, max.iter = Inf )
```

Arguments

<code>xL</code>	Either a matrix or a list of matrices containing training observation sequences, with one sample per row
<code>K</code>	Desired number of components
<code>vL</code>	Either a matrix or a list of matrices containing validation observation sequences, with one sample per row
<code>gmm.init</code>	Optional initial model, can be partially specified
<code>cov.reg</code>	Covariance matrix regularization (towards identity), value must be in [0, 1]
<code>tol</code>	Stopping criterion: relative tolerance on the log-likelihood
<code>LLstop</code>	Stopping criterion: hard bound on the log-likelihood value
<code>min.iter</code>	At least this number of EM iterations is preformed before validation and tolerance stopping criteria are triggered
<code>max.iter</code>	Stoppint criterion: maximum number of iterations

Details

Learns a maximum likelihood GMM given the data

Value

A Gaussian Mixture Model defined by:

<code>mu</code>	[K x p] matrix of component means
<code>sigma</code>	[K x p x p] array of component covariance matrices
<code>pi</code>	[K x 1] vector of mixture coefficients

Author(s)

Artem Sokolov (Artem.Sokolov@gmail.com)

`gmm-ll`*Gaussian Mixture Models*

Description

Computes log-likelihood of new observation sequences given a previously trained model

Usage

```
GMM.ll ( x, gmm )
```

Arguments

<code>x</code>	A matrix of a list of matrices containing one or more observation sequences
<code>gmm</code>	A Gaussian mixture model computed by, e.g., <code>GMM.learn</code>

Value

If `x` is a single matrix, the function returns the log-likelihood of that observation sequence as a single value. If `x` is a list of observation sequences, the log-likelihood values are computed for each sequence and returned together in a list.

Author(s)

Artem Sokolov (Artem.Sokolov@gmail.com)

`gmm-make`*Gaussian Mixture Models*

Description

Generate an instance of GMM parameters.

Usage

```
GMM.make( K, p, mu=rep(0, p), sigma=diag(p) )
```

Arguments

<code>K</code>	Number of components
<code>p</code>	Dimensionality
<code>mu</code>	General center of the data being modeled
<code>sigma</code>	General orientation of the data being modeled

Details

Component centers are generated from a multivariate Normal distribution with the provided mean `mu` and covariance matrix `sigma`. Component covariance matrices are all set to the provided `sigma`. Prior distribution is uniform across all states.

Value

A Gaussian Mixture Model defined by:

<code>mu</code>	[K x p] matrix of component means
<code>sigma</code>	[K x p x p] array of component covariance matrices
<code>pi</code>	[K x 1] vector of mixture coefficients

Author(s)

Artem Sokolov <Artem.Sokolov@gmail.com>

gmm-resp

Gaussian Mixture Models

Description

Compute component responsibilities and posterior probabilities

Usage

```
GMM.resp( x, gmm )
```

Arguments

x	[N x p] matrix of N samples in p dimensions
gmm	Gaussian mixture model for K components with dimensionality p

Value

resp	[N x K] matrix of component responsibilities
LL	[N x 1] matrix of log-likelihood values for each sample

Author(s)

Artem Sokolov <Artem.Sokolov@gmail.com>

hmm-fwbk

Hidden Markov Models

Description

Forward-Backward procedure

Usage

```
HMM.fwbk( x, hmm )
```

Arguments

x	[N x p] matrix of N samples in p dimensions
hmm	An HMM

Details

The E-step of the EM algorithm for Hidden Markov Models

Value

alpha	[N x K] matrix of forward variable values
beta	[N x K] matrix of backward variable values
c	[N x 1] vector of scaling coefficients
gamma	[N x K] matrix of transition probabilities from state i: $\gamma[t,i] = P(Q_t = i \mid O, \text{hmm})$
xi	[N-1 x K x K] array of transition probabilities from state i to state j: $\xi[t,i,j] = P(Q_t = i, Q_{t+1} = j \mid O, \text{hmm})$

Author(s)

Artem Sokolov <Artem.Sokolov@gmail.com>

hmm-gendata

Hidden Markov Models

Description

Generate data from a provided model.

Usage

```
HMM.genData( N, hmm )
```

Arguments

N	Number of samples to generate
hmm	The model of dimensionality p

Details

Hidden state value is first generated from prior distribution (first sample) or transition matrix (second sample and on). The state mean and covariance matrix are then used to generate the actual sample.

Value

N x p matrix of generated data samples

Author(s)

Artem Sokolov <Artem.Sokolov@gmail.com>

hmm-learn

*Hidden Markov Models***Description**

Maximum Likelihood learning via EM

Usage

```
HMM.learn( xL, K, vL=NULL, hmm.init=NULL, tol=1e-03, LLstop=Inf,
min.iter=3, max.iter=Inf )
```

Arguments

<code>xL</code>	Either a matrix or a list of matrices containing training observation sequences, with one sample per row
<code>K</code>	Desired number of states
<code>vL</code>	Either a matrix or a list of matrices containing validation observation sequences, with one sample per row
<code>hmm.init</code>	Optional initial model, can be partially specified
<code>tol</code>	Stopping criterion: relative tolerance on the log-likelihood
<code>LLstop</code>	Stopping criterion: hard bound on the log-likelihood value
<code>min.iter</code>	At least this number of EM iterations is preformed before validation and tolerance stopping criteria are triggered
<code>max.iter</code>	Stoppint criterion: maximum number of iterations

Details

Learns a maximum likelihood HMM given the data

Value

A Hidden Markov Model defined by:

<code>mu</code>	[K x p] matrix of component means
<code>sigma</code>	[K x p x p] array of component covariance matrices
<code>pi</code>	[K x 1] vector of mixture coefficients
<code>A</code>	[K x K] state transition matrix with element (i,j) referring to transition from state i to state j

Author(s)

Artem Sokolov <Artem.Sokolov@gmail.com>

`hmm-ll`*Hidden Markov Models*

Description

Computes log-likelihood of new observation sequences given a previously trained model

Usage

```
HMM.ll( x, hmm )
```

Arguments

<code>x</code>	A matrix of a list of matrices containing one or more observation sequences
<code>hmm</code>	A hidden markov model created by functions like <code>HMM.learn</code> or <code>HMM.make</code>

Value

If `x` is a single matrix, the function returns the log-likelihood of that observation sequence as a single value. If `x` is a list of observation sequences, the log-likelihood values are computed for each sequence and returned together in a list.

Author(s)

Artem Sokolov <Artem.Sokolov@gmail.com>

`hmm-make`*Hidden Markov Models*

Description

Generate an instance of HMM parameters.

Usage

```
HMM.make( K, p, mu=rep(0, p), sigma=diag(p) )
```

Arguments

<code>K</code>	Number of components
<code>p</code>	Dimensionality
<code>mu</code>	General center of the data being modeled
<code>sigma</code>	General orientation of the data being modeled

Details

Component centers are generated from a multivariate Normal distribution with the provided mean μ and covariance matrix σ . Component covariance matrices are all set to the provided σ . Prior distribution is uniform across all states. Rows of the transition matrix are random multinomial distributions.

Value

A Hidden Markov Model defined by:

<code>mu</code>	[K x p] matrix of component means
<code>sigma</code>	[K x p x p] array of component covariance matrices
<code>pi</code>	[K x 1] vector of mixture coefficients
<code>A</code>	[K x K] state transition matrix with element (i,j) referring to transition from state i to state j

Author(s)

Artem Sokolov <Artem.Sokolov@gmail.com>

hmm-obs

Hidden Markov Models

Description

Computes observation probabilities

Usage

```
HMM.obs ( x, hmm )
```

Arguments

<code>x</code>	[N x p] matrix of N samples in p dimensions
<code>hmm</code>	An HMM

Details

Computes observation probability for each (sample, state) pair

Value

<code>B</code>	[N x K] matrix of probabilities
<code>outliers</code>	indices of points that had zero probability for all states

Author(s)

Artem Sokolov <Artem.Sokolov@gmail.com>

`lda-test`*Linear Discriminant Analysis*

Description

Test method for LDA.

Usage

```
LDA.test( x, clsf )
```

Arguments

<code>x</code>	N x p data matrix of N samples in p dimensions
<code>clsf</code>	LDA classifier object produced by LDA.train

Details

Classifies new test data using LDA discriminant functions computed during training.

Value

N x 1 vector of predicted labels

Author(s)

Artem Sokolov <Artem.Sokolov@gmail.com>

See Also

[LDA.train](#)

`lda-train`*Linear Discriminant Analysis*

Description

Training method for LDA.

Usage

```
LDA.train( x, y, cov.reg = 0.0 )
```

Arguments

<code>x</code>	<code>N x p</code> data matrix of <code>N</code> samples in <code>p</code> dimensions
<code>y</code>	<code>N x 1</code> vector of labels
<code>cov.reg</code>	Covariance matrix regularization (towards identity), value must be in <code>[0, 1]</code>

Details

Models each class as a single (multivariate) Gaussian and uses a single common covariance matrix across all classes. Computes the distribution parameters, the Bayesian class priors, and the discriminant functions. LDA is insensitive to temporal structure of the data and, therefore, only needs to work with a single observation sequence. This, in turn, requires a label for each sample.

Value

An LDA classifier defined by:

<code>labels</code>	Vector of unique class labels
<code>priors</code>	<code>K x 1</code> vector of priors, estimated as fraction of points from each class
<code>means</code>	<code>K x p</code> matrix of means approximated from the data
<code>covmat</code>	The common <code>p x p</code> covariance matrix
<code>weights</code>	<code>K x (p+1)</code> matrix of weights and the bias term for each of the <code>K</code> discriminant functions

Author(s)

Artem Sokolov <Artem.Sokolov@gmail.com>

qda-test

Quadratic Discriminant Analysis

Description

Test method for QDA.

Usage

```
QDA.test( x, clsf )
```

Arguments

<code>x</code>	<code>N x p</code> data matrix of <code>N</code> samples in <code>p</code> dimensions
<code>clsf</code>	QDA classifier object produced by <code>QDA.train</code>

Details

Classifies new test data using QDA discriminant functions computed during training.

Value

N x 1 vector of predicted labels

Author(s)

Artem Sokolov <Artem.Sokolov@gmail.com>

See Also

[QDA.train](#)

qda-train

Quadratic Discriminant Analysis

Description

Training method for QDA.

Usage

```
QDA.train( x, y, cov.reg = 0.0 )
```

Arguments

x	N x p data matrix of N samples in p dimensions
y	N x 1 vector of labels
cov.reg	Covariance matrix regularization (towards identity), value must be in [0, 1]

Details

Models each class as a single (multivariate) Gaussian. Relaxes the common covariance matrix constraint of LDA. Computes the distribution parameters, the Bayesian class priors, and the discriminant functions. QDA is insensitive to temporal structure of the data and, therefore, only needs to work with a single observation sequence. This, in turn, requires a label for each sample.

Value

An QDA classifier defined by:

labels	Vector of unique class labels
priors	K x 1 vector of priors, estimated as fraction of points from each class
means	K x p matrix of means approximated from the data
covmats	K x p x p array of covariance matrices esimated from the data
icovmats	K x p x p array of inverse covariance matrices
bias	K x 1 vector of bias terms for discriminant function computations

Author(s)

Artem Sokolov <Artem.Sokolov@gmail.com>

sm-options

Manipulation of Stochastic Modeling Options

Description

These functions provide a means to get/set options that affect behavior of many modeling routines

Usage

```
SM.setOption( name, value = TRUE )
SM.getOption( name )
SM.getOptions()
```

Arguments

name	Argument name. Currently supported options are:
	debug.output: Amount of debug output. (integer)
	pure.R: whether exclusively R code is to be used. (logical)
value	Argument value

Value

'SM.setOption' returns the previous argument value. 'SM.getOption' returns the current value. 'SM.getOptions' returns a list of current option assignments.

Author(s)

Artem Sokolov <Artem.Sokolov@gmail.com>

sm-output

Output of Debug Information

Description

Versions of cat and print that depend on debug.output option

Usage

```
d.cat( lvl, ... )
d.print( lvl, ... )
```

Arguments

<code>lvl</code>	When debug.output option is above (or equal to) this level, output is displayed
<code>...</code>	Arguments that get passed to the actual output function

Value

'd.cat' returns the same value as cat 'd.print' return the same value as print

Author(s)

Artem Sokolov (Artem.Sokolov@gmail.com)

Index

*Topic **IO**

sm-output, 14

*Topic **models**

feat-lagdata, 1

gmm-gendata, 2

gmm-learn, 3

gmm-ll, 4

gmm-make, 4

gmm-resp, 5

hmm-fwbk, 6

hmm-gendata, 6

hmm-learn, 7

hmm-ll, 8

hmm-make, 9

hmm-obs, 10

lda-test, 10

lda-train, 11

qda-test, 12

qda-train, 13

*Topic **utilities**

sm-options, 14

d.cat (sm-output), 14

d.print (sm-output), 14

feat-lagdata, 1

FEAT.lagData (feat-lagdata), 1

gmm-gendata, 2

gmm-learn, 3

gmm-ll, 4

gmm-make, 4

gmm-resp, 5

GMM.genData (gmm-gendata), 2

GMM.learn (gmm-learn), 3

GMM.ll (gmm-ll), 4

GMM.make (gmm-make), 4

GMM.resp (gmm-resp), 5

hmm-fwbk, 6

hmm-gendata, 6

hmm-learn, 7

hmm-ll, 8

hmm-make, 9

hmm-obs, 10

HMM.fwbk (hmm-fwbk), 6

HMM.genData (hmm-gendata), 6

HMM.learn (hmm-learn), 7

HMM.ll (hmm-ll), 8

HMM.make (hmm-make), 9

HMM.obs (hmm-obs), 10

lda-test, 10

lda-train, 11

LDA.test (lda-test), 10

LDA.train, 11

LDA.train (lda-train), 11

qda-test, 12

qda-train, 13

QDA.test (qda-test), 12

QDA.train, 12

QDA.train (qda-train), 13

sm-options, 14

sm-output, 14

SM.getOption (sm-options), 14

SM.getOptions (sm-options), 14

SM.setOption (sm-options), 14