Package 'lrgs'

December 2, 2017

Type Package

Title Linear Regression by Gibbs Sampling

Version 0.5.3

Date 2016-12-15

Author Adam Mantz

Maintainer Adam Mantz <amantz@slac.stanford.edu>

Description Implements a Gibbs sampler to do linear regression with multiple covariates, multiple responses, Gaussian measurement errors on covariates and responses, Gaussian intrinsic scatter, and a covariate prior distribution which is given by either a Gaussian mixture of specified size or a Dirichlet process with a Gaussian base distribution.

License MIT + file LICENSE

LazyLoad yes

Imports mvtnorm

URL https://github.com/abmantz/lrgs

NeedsCompilation no

Repository CRAN

Date/Publication 2017-12-01 23:14:51 UTC

R topics documented:

`lrgs-package` *Linear Regression by Gibbs Sampling*

Description

Implements a Gibbs sampler to do linear regression with multiple covariates, multiple responses, Gaussian measurement errors on covariates and responses, Gaussian intrinsic scatter, and a covariate prior distribution which is given by either a Gaussian mixture of specified size or a Dirichlet process with a Gaussian base distribution.

Details

Package:	lrgs
Type:	Package
Version:	0.5.3
Date:	2016-12-15
License:	MIT
LazyLoad:	yes

See help for Gibbs.regression

Author(s)

Adam Mantz <amantz@slac.stanford.edu>

`Gibbs.post2dataframe` *Linear Regression by Gibbs Sampling*

Description

Transforms a set of posterior samples produced by Gibbs.regression into a data frame for more straightforward analysis.

Usage

```
Gibbs.post2dataframe(p)
```

Arguments

`p`	an object returned from Gibbs.regression.

Value

A data frame with one column corresponding to each parameter stored in post.

Author(s)

Adam Mantz

See Also

Gibbs.regression

`Gibbs.regression` *Linear Regression by Gibbs Sampling*

Description

Runs a Gibbs sampler to simulate the posterior distribution of a linear model with (potentially) multiple covariates and response variables. Throughout this help file, we use the following notation: there are n data points, m response variables and p covariates.

Usage

```
Gibbs.regression(x.in, y.in, M, Nsamples, Ngauss = 1, dirichlet=FALSE,
 M.inv = NULL, intercept = TRUE, trace = "bs", fix = "", start = list(),
 B.prior.mean=NULL, B.prior.cov=NULL, Sigma.prior.scale=NULL,
 Sigma.prior.dof=-1, dp.prior.alpha=NULL, dp.prior.beta=NULL,
 mention.every=NA, save.every=NA, save.to=NA)
```

Arguments

`x.in`	the measured covariates. Either an n*p matrix, a vector (if p=1) or NULL (if p=0, i.e. a the model is constant).
`y.in`	the measured responses. Either an n*m matrix or a vector (if m=1).
`M`	a (p+m)*(p+m)*n array holding the n measurement covariance matrices, with covariates ordered before responses within each matrix. If both M and M.inv are NULL, all measurement errors are taken to be zero.
`Nsamples`	the number of iterations of the Gibbs sampler to run.
`Ngauss`	number of Gaussian mixture components describing the distribution of covariates. Pass 0 for the special case of a uniform distribution.
`dirichlet`	if TRUE, the prior on covariates is given not by a Gaussian mixture of size Ngauss, but by a Dirichlet process with a Gaussian base distribution (see details). This can be thought of as a Gaussian mixture where Ngauss is learned from the data and marginalized over.
`M.inv`	as an alternative to M, the inverses of the n measurement covariance matrices in the same format. Note that the M argument takes precedence; pass M=NULL to use M.inv. If both M and M.inv are NULL, all measurement errors are taken to be zero.
`intercept`	if TRUE, the model includes constant (intercept) terms for each response. Otherwise, the intercepts are fixed to zero.

`trace`	determines which variables are returned at the end of the call. See details.
`fix`	determines which parameters are NOT varied. See details.
`start`	a list containing starting values for any of the model parameters. These are optional, with the exception of Sigma and Tau if they are fixed. See details.
`B.prior.mean`	optional vector giving the mean of a Gaussian prior to be applied to the coefficients. The order is B11, B21, ..., B12, B22, ..., where the first index refers to the covariate and the second to the response. Assumed to be zero if it is not specified but B.prior.cov is.
`B.prior.cov`	optional matrix giving the covariance of a Gaussian prior to be applied to the coefficients. The parameter order is the same as in B.prior.mean. If not specified, an improper uniform prior is used.
`Sigma.prior.scale`	optional matrix giving the scale parameter of an inverse-Wishart prior to be applied to the intrinsic covariance. Default is zero.
`Sigma.prior.dof`	number of degrees of freedom for the inverse-Wishart prior on the intrinsic covariance. Default is -1, which corresponds to the Jeffreys (i.e. minimally informative) prior when the default scale matrix is used. Using -(m+1) degrees of freedom with the default scale matrix corresponds to a prior that is uniform in det(Sigma).
`dp.prior.alpha`	shape paramerer of a Gamma prior to be applied to the Dirichlet process concentration parameter, if dirichlet=TRUE.
`dp.prior.beta`	rate paramerer of a Gamma prior to be applied to the Dirichlet process concentration parameter, if dirichlet=TRUE.
`mention.every`	if set to a positive integer N, a message will be printed after every N iterations to confirm that something is happening.
`save.every`	if set to a positive integer N, the result will be saved as an object named "res" to the file named in the save.to argument every N iterations.
`save.to`	the name of a file to periodically save the results to.

Details

An in-depth description of the model and algorithm can be found in the references below. The full set of parameters is

X: true values of all covariates, arranged as a design matrix

Y: true values of all responses

B: matrix of intercepts and coefficients

Sigma: intrisic covariance matrix of the responses

G: vector encoding which Gaussian mixture component of the covariate distribution model each data point belongs to

pi: vector of relative proportions of the mixture components

mu: matrix holding the mean of each mixture component

Tau: covariance matrices for each mixture component

mu0: mean of the Gaussian hyper-prior applied to mu

U: covariance matrix of the hyper-prior for mu

W: matrix defining the Wishart hyper-prior for Tau

alpha: the Dirichlet process concentration parameter

Note that mu0, U and W are meaningful only if Ngauss>1. If Ngauss=0 these, as well as G, pi, mu and Tau, are ignored.

If a Dirichlet process prior on the covariates is used instead of a Gaussian mixture, then mu and Tau represent the hyperparameters of the Gaussian base distribution of the process. In this case, pi mu0, U and W are not sampled, but G can still be interpretted as giving which cluster each data point belongs to. alpha is only meaningful if the Dirichlet process is being used.

In the trace and fix arguments, these parameters (in the order above) are indicated by the characters "xybsgpmtzuwa". So, for example, the default value trace="bs" means that only the coefficients and intrisic scatter are returned to the user, while fix="x" would fix the values of X (typically to the measured input values, x.in).

The contents of the list optionally passed to start should be similar to those returned by this function, with one fewer dimension (corresponding to the multiple samples returned). That is, if the $X item returned by this function has dimension c(n,p+1,Nsamples), a valid value for start$X has dimensions c(n,p+1).

Value

A list containing samples of the model parameters specified by the trace argument. The parameters are defined above; the dimensionality of the result will be

`X`	n,p+1,Nsamples
`Y`	n,m,Nsamples
`B`	p+1,m,Nsamples
`Sigma`	m,m,Nsamples
`G`	n,Nsamples
`pi`	Ngauss,Nsamples
`mu`	Ngauss,p,Nsamples
`Tau`	p,p,Ngauss,Nsamples
`mu0`	p,Nsamples
`U`	p,p,Nsamples
`W`	p,p,Nsamples
`alpha`	Nsamples

This assumes intercept=TRUE; otherwise, replace p+1 with p everywhere. This list can be transformed into a data frame using the Gibbs.post2dataframe function.

Note

The output is a Monte Carlo Markov Chain, and therefore may take some time to converge to the true posterior distribution. Traces of the parameters of interest should be examined in order to identify and remove this "burn-in" period.

If your data set is missing the occasional covariate or response measurement, it should be sufficient to set the corresponding variance in M to a very large number. (Alternatively, this case can be handled exactly by passing the inverse-covariances through M.inv, and setting the appropriate elements to zero.) Similarly, if a subset of values are measured with no error, use a very small number for the variance (but do not attempt to pass zero covariance or Inf in the inverse-covariance).

Author(s)

Adam Mantz

References

Mantz (2016; MNRAS 457:1279; arXiv:1509.00908; doi:10.1093/mnras/stv3008) for this function, Kelly (2007, ApJ 665:1489; arXiv:0705.2774; doi:10.1086/519947) for the same approach with a single response variable and a Gaussian mixture of covariates, Neal (2000, Journal of Computational and Graphical Statistics 9:249) for the Dirichlet process algorithm employed.

See Also

lm for classical linear regression.

Examples

```
## example using the default Ngauss=1 with no measurement errors
x <- rnorm(500, 0, 5)
y <- pi*x + rnorm(length(x), 0, 0.1)
post <- Gibbs.regression(x, y, NULL, 50, trace='bsmt', fix='xy')
m <- lm(y~x)
plot(post$B[1,1,-(1:10)], col=4); abline(h=m$coefficients[1], lty=2, col=2)
plot(post$B[2,1,-(1:10)], col=4); abline(h=m$coefficients[2], lty=2, col=2)
plot(post$Sigma[1,1,-(1:10)], col=4); abline(h=var(m$residuals), lty=2, col=2)
plot(post$mu[1,1,-(1:10)], col=4); abline(h=mean(x), lty=2, col=2)
plot(post$Tau[1,1,1,-(1:10)], col=4); abline(h=var(x), lty=2, col=2)

## verbose example using a Dirichlet process, including measurement errors
## in practice, you would would want a longer chain, i.e. larger nmc
xx <- rnorm(100, c(-15,0,15), 1)
yy <- xx + rnorm(length(xx)) + rnorm(length(xx), 0, 3)
xx <- xx + rnorm(length(xx))
M <- array(0, dim=c(2,2,length(xx)))
M[1,1,] <- 1
M[2,2,] <- 1
nmc <- 10
post <- Gibbs.regression(xx, yy, M, nmc, dirichlet=TRUE, trace='bsgmta', mention.every=1)
plot(xx, yy, col=post$G[,nmc]) # plot clusters at the last iteration
m <- lm(yy~xx)
```

```
plot(post$B[1,1,-1], col=4); abline(h=m$coefficients[1], lty=2, col=2)
plot(post$B[2,1,-1], col=4); abline(h=m$coefficients[2], lty=2, col=2)
plot(post$Sigma[1,1,-1], col=4); abline(h=var(m$residuals), lty=2, col=2)
plot(post$mu[1,1,-1], col=4); abline(h=mean(xx), lty=2, col=2)
plot(post$Tau[1,1,1,-1], col=4); abline(h=var(xx), lty=2, col=2)
plot(post$alpha[-1], col=4)
```

Index