Package ‘s525’

December 13, 2017

Type Package

Title Implementation of common Bayesian models

Version 1.0

Date 2017-11-26

Author Daniel Bourgeois

Maintainer Daniel Bourgeois <dcb10@rice.edu>

Description This package contains implementations of common Bayesian models.

Imports truncnorm, MASS, coda

License MIT License

R topics documented:

s525-package *Implementation of common Bayesian models*

Description

This package contains implementations of common Bayesian models.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

Author(s)

Daniel Bourgeois

Maintainer: Daniel Bourgeois <dcb10@rice.edu>

Examples

```
# simple examples of the most important functions
```

factor_analysis *Runs gibbs sampler for a factor model.*

Description

The model is as follows:

$$y_i = \Lambda \eta_i + \epsilon_i$$

$$\epsilon_i \sim N_p(0, \Sigma)$$

where Σ = diag($\sigma_1^2, ..., \sigma_p^2$).

See Joyee Ghosh & David B. Dunson (2009) Default Prior Distributions and Efficient Posterior Computation in Bayesian Factor Analysis, Journal of Computational and Graphical Statistics, 18:2, 306-320, DOI: 10.1198/jcgs.2009.07145 for conditional posteriors.

Usage

```
factor_analysis <- function(
  Y,
  k,
  niter = 1000,
  shape_psi = 1/2,
  rate_psi = 1/2,
  shape_sigma2 = 1,
  rate_sigma2 = 0.2,
  nonzero_structure = NULL)
```

Arguments

`Y`	n by p matrix
`k`	number of factors
`niter`	number of iterations for the gibbs sampler to run.
`shape_psi`	shape parameter for psi. Can be a scalar or a k vector
`rate_psi`	rate parameter for psi. Can be a k vector
`shape_sigma2`	shape parameter for sigma2. Can be a p vector
`rate_sigma2`	rate parameter for sigma2. Can be a p vector
`nonzero_structure`	A boolean p x k matrix. If the i, jth spot is TRUE, then λ_{ij} is free. If the i, jth spot is FALSE, then λ_{ij} is zero. If not set, then a lower triangular matrix is used.

---

factor_analysis_with_regression

Runs gibbs sampler for a factor model with regression on the factors.

Description

The model is as follows:

$$y_i = \alpha + \Lambda\eta_i + \epsilon_i,$$

$$\eta_i = Bx_i + \delta_i$$

$$\epsilon_i \sim N_p(0, \tau^{-1})$$

$$\delta_i \sim N_k(0, I)$$

where τ = diag($\tau_1, ..., \tau_p$).

See Joyee Ghosh & David B. Dunson (2009) Default Prior Distributions and Efficient Posterior Computation in Bayesian Factor Analysis, Journal of Computational and Graphical Statistics, 18:2, 306-320, DOI: 10.1198/jcgs.2009.07145.

Usage

```
factor_analysis_with_regression <- function(
  Y,
  X,
  k,
  niter = 1,
  shape_psi = 1/2,
  rate_psi = 1/2,
  shape_tau = 1,
  rate_tau = 0.2,
  coef_multiplier = 10,
  nonzero_structure = NULL)
```

Arguments

Y	n by p matrix
X	n by f matrix
k	number of factors
niter	number of iterations for the gibbs sampler to run.
shape_psi	shape parameter for psi. Can be a scalar or a k vector
rate_psi	rate parameter for psi. Can be a k vector
shape_tau	shape parameter for sigma2. Can be a p vector
rate_tau	rate parameter for sigma2. Can be a p vector
nonzero_structure	A boolean p x k matrix. If the i, jth spot is TRUE, then λ_{ij} is free. If the i, jth spot is FALSE, then λ_{ij} is zero. If not set, then a lower triangular matrix is used.

`horseshoe_regression` *Gibbs Sampler for Horseshoe Regression*

Description

The horseshoe regression model is given by

$$[y_i|x_i, \beta, \sigma] \sim N(x_i^t\beta, \sigma^2), i = 1, ..., n,$$

$$[\beta_j|\sigma, \lambda_j] \sim N(0, \sigma^2\lambda_j^2),$$

$$[\lambda_j|A] \sim C^+(0, A),$$

$$A \sim Uniform(0, 10),$$

$$p(\sigma^2) \propto \frac{1}{\sigma^2}.$$

The half-Cauchy parameter expansion is used; given by

$$[\eta_j|\gamma_j] \sim Gamma(\frac{1}{2}, \gamma_j),$$

$$[\gamma_j] \sim Gamma(\frac{1}{2}, \frac{1}{A^2}).$$

Let $\eta_j = \lambda_j^{-2}$, $\tau_A = A^{-2}$, $\tau = \frac{1}{\sigma^2}$ and $\Lambda = diag(\eta_1, ..., \eta_p)$. The full conditionals are given by:

$$[\beta|Y, X, \eta, \tau] \sim \mathcal{N}((X'X + \Lambda)^{-1}X'Y, \tau^{-1}(X'X + \Lambda)^{-1}),$$

$$[\eta_j|\beta_j, \gamma_j, \tau] \sim \exp(\frac{\tau\beta_j^2}{2} + \gamma_j),$$

$$[\gamma_j|\eta_j, \tau_A] \sim \exp(\eta_j + \tau_A),$$

$$[\tau_A|\gamma] \sim \mathrm{Gamma}(\frac{p-1}{2}, \sum \gamma_i)\mathrm{I}_{(\frac{1}{100},\infty)},$$

$$[\tau|Y, X, \beta, \eta] \sim \mathrm{Gamma}(\frac{n+p}{2}, \frac{(y - X\beta)'(y - X\beta) + \beta'\Lambda\beta}{2}).$$

Usage

```
horseshoe_regression <- function(
  Y,
  X,
  niter = 10000)
```

Arguments

Y	n by 1
X	n by p predictor matrix
niter	number of gibbs sampling iterations

Details

This function returns the generated parameters from the gibbs sampling markov chain.

Value

beta	An niter x p matrix
lambda	An niter x p matrix
sigma	An niter x 1 matrix

Examples

```
# Load the data
 prostate.data = "https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data"
prostate = read.table(file = prostate.data, sep="", header = TRUE)
# Training data:
prostate_train = prostate[which(prostate$train),-10]
# Testing data:
prostate_test = prostate[which(!prostate$train),-10]
# Response:
y = prostate_train$lpsa
# Center and scale the data:
```

```
y = scale(y)
# And the predictors
X = scale(prostate_train[,names(prostate_train) != "lpsa"])

gibbs_hs <- horseshoe_regression(y, X, niter=10000)
shrinkage_regression_plot(gibbs_hs$beta, y, X)
```

---

mat_apply *mat apply*

Description

Apply a function to each element in an input and return an arbitrary dimensioned array.

Usage

```
mat_apply <- function(vec, fun)
```

Arguments

vec	vector of inputs fed into fun
fun	a function to call for each input of vec

Details

This function returns an array with first dimension of length(vec) provided length(vec) > 1–the first dimension indexes over the length(vec) outputs.

Examples

```
# returns a 10 x 3 x 3 array. ret[i,,] contains i*diag(3).
ret <- mat_apply(1:10, function(i){ i*diag(3) })
```

---

metropolis *Metropolis MCMC*

Description

The metropolis algorithm is a special case of the metropolis-hastings algorithm, namely where the proposal distribution is symmetric.

Usage

```
metropolis <- function(
  rproposal,
  prob,
  niter,
  init,
  log_prob = FALSE)
```

Arguments

rproposal	rproposal(previous_val) generates a value from the proposal distribution
prob	prob(val_proposed)/prob(val_t_minus_1) forms the acceptance probability
niter	number of iterations to perform
init	vector of initial values
log_prob	whether or not prob function specifies log probabilities

Details

This function returns a niter x d matrix of values where d is the dimension of init and the dimension of each element from rproposal. The returned matrix contains all generated values of the metropolis walk.

Examples

```
vals <- metropolis(
  rproposal = function(val){ rnorm(1, mean = val, sd = 1){} },
  prob = posterior_prob,
  niter = 10000,
  init = 0)

library(coda)
plot(as.mcmc(vals))
```

---

probit_horseshoe_regression

Gibbs Sampler for Probit Regression with Horseshoe Prior

Description

The probit regression model with horseshoe prior is given by

$$y_i|\pi_i \sim Bernoulli(\pi_i),$$

$$\pi_i = \Phi(x_i^t\beta),$$
$$[\beta_j|\lambda_j] \sim N(0, \lambda_j^2), j = 2, ..., p,$$
$$p(\beta_1) \propto 1,$$
$$[\lambda_j|A] \sim C^+(0, A), j = 2, ..., p,$$
$$A \sim Uniform(0, 10).$$

where Φ is given by the gaussian CDF.

The implemented parameter-expanded model is given by

$$y_i^* = x_i^t\beta + \epsilon_i,$$

$$\epsilon_i \sim N(0,1),$$

$$y_i = I(y_i^* > 0).$$

The half-Cauchy parameter expansion is also used; given by

$$[\eta_j|\gamma_j] \sim Gamma(\frac{1}{2}, \gamma_j),$$

$$[\gamma_j] \sim Gamma(\frac{1}{2}, \frac{1}{A^2})$$

and $\eta_j = \lambda_j^{-2}$, $\tau_A = A^{-2}$, The full conditionals are given by:

$$[y_i^*|y_i, \beta, X] \sim sgn(y_i, y_i^*)N(x_i^t\beta, 1)$$

where sqn is 1 if both arguments are of the same sign and zero otherwise,

$$[\beta|Y^*, X, \eta] \sim \mathcal{N}(Q^{-1}l, Q^{-1})$$

where $Q = X'X + diag(0, 1/\eta_2, ..., 1/\eta_p)$ and $l = X'Y^*$,

$$[\eta_j|\beta_j, \gamma_j] \sim \exp(\frac{\beta_j^2}{2} + \gamma_j),$$

$$[\gamma_j|\eta_j, \tau_A] \sim \exp(\eta_j + \tau_A),$$

$$[\tau_A|\gamma] \sim \mathrm{Gamma}(\frac{p-2}{2}, \sum \gamma_i)\mathrm{I}_{(\frac{1}{100}, \infty)}.$$

Usage

```
probit_horseshoe_regression <- function(
  Y,
  X,
  niter,
  init = NULL)
```

Arguments

`Y`	n by 1 vector of ones and zeros
`X`	n by p predictor matrix, where p > 1 and the first column of X is all 1.
`niter`	number of gibbs sampling iterations
`init`	Initial starting values for beta. If NULL, beta is set to zero.

Details

This function returns a niter x p matrix of values where p is the second dimension of the predictor matrix X. The returned matrix contains all generated values of the gibbs sampling markov chain.

Examples

```
print("TODO")
```

probit_regression *Gibbs Sampler for Probit Regression*

Description

The probit regression model is given by

$$y_i|\pi_i \sim Bernoulli(\pi_i),$$
$$\pi_i = \Phi(x_i^t\beta),$$
$$\beta_j \sim N(0, A^2).$$

where Φ is given by the gaussian CDF.

The implemented parameter-expanded model is given by

$$y_i^* = x_i^t\beta + \epsilon_i,$$
$$\epsilon_i \sim N(0, 1),$$
$$y_i = I(y_i^* > 0).$$

The full conditional distributions are given by

$$[\beta|y, y^*, X] \sim N(Q^{-1}l, Q^{-1})$$

where $Q = X^tX + A^{-2}I$ and $l = X^ty^*$,

$$[y_i^*|y_i, \beta, X] \sim sgn(y_i, y_i^*)N(x_i^t\beta, 1)$$

where sqn is 1 if both arguments are of the same sign and zero otherwise.

If A is a vector, $Q = X^tX + diag(A)$. A flat prior on the intercept at position 1 is thus given by A = c(Inf, rep(3, p)).

Usage

```
probit_regression <- function(
  Y,
  X,
  niter,
  A = 3,
  init = NULL)
```

Arguments

Y	n by 1 vector of ones and zeros
X	n by p predictor matrix
niter	number of gibbs sampling iterations
A	A parameter. The default value is chosen to provide a reasonable range of π. If A is a vector, then different variances are given for each intercept.
init	Initial starting values for beta. If NULL, beta is set to zero.

Details

This function returns a niter x p matrix of values where p is the second dimension of the predictor matrix X. The returned matrix contains all generated values of the gibbs sampling markov chain.

Examples

```
library(LearnBayes)
library(s525)

data(donner)
y = donner$survival
X = cbind(1, donner$age, donner$male)
niter <- 2000

gibbs_results <- probit_regression(y, X, niter)
```

`ridge_regression` *Gibbs Sampler for Ridge Regression*

Description

The ridge regression model coeffecients are given by

$$\arg\min_{\beta} \parallel Y - X\beta \parallel^2 + \lambda \parallel \beta \parallel^2 .$$

This is the implementation of the bayesian interpretation. Namely,

$$[y_i|x_i, \beta, \tau] \sim N(x_i^t\beta, \tau^{-1}), i = 1, ..., n,$$

$$[\beta|\eta] \sim N(0, \eta^{-1}I),$$

$$\tau \sim Gamma(\frac{1}{100}, \frac{1}{100}),$$

$$\sigma_\beta \sim unif(0, A)$$

where $A = 1000$ and $\tau = \frac{1}{\sigma_\beta^2}$.

Usage

```
ridge_regression <- function(
  Y,
  X,
  niter = 10000)
```

Arguments

`Y`	n by 1
`X`	n by p predictor matrix
`niter`	number of gibbs sampling iterations

Details

This function returns the generated parameters from the gibbs sampling markov chain.

Value

beta	An niter by p matrix
sigma	An niter by 1 matrix
sigma_beta	An niter by 1 matrix

Examples

```
# Load the data
 prostate.data = "https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data"
prostate = read.table(file = "prostate.data", sep="", header = TRUE)
# Training data:
prostate_train = prostate[which(prostate$train),-10]
# Testing data:
prostate_test = prostate[which(!prostate$train),-10]
# Response:
y = prostate_train$lpsa
# Center and scale the data:
y = scale(y)
# And the predictors
X = scale(prostate_train[,names(prostate_train) != "lpsa"])

gibbs_ridge <- ridge_regression(y, X, niter=10000)
shrinkage_regression_plot(gibbs_ridge$beta, y, X, main = "Ridge Prior")
```

---

rnorm_qinv_l *Sample from multivariate normal distribution*

Description

Sample from multivariate normal distribution with mean $Q^{-1}l$ and covariance matrix Q^{-1}.

Usage

```
rnorm_qinv_l <- function(
  n,
  Q,
  l)
```

Arguments

n	number of elements to generate
Q	p by p precision matrix.
l	p by 1 vector

Details

The algorithm is as follows

1. Cholesky decomposition of Q into LL^t.
2. Sample z from rnorm(p). Let $y = Lz + l$.
3. Solve for x in $LL^t x = y$ and return.

Value

beta	An p x 1 vector if $n = 1$ otherwise a n by p matrix

---

rpost_regression_coef *Generate posterior for regression coefecients*

Description

Generate a $N(\mu, \Sigma)$ random variable where

$$\Sigma = (X^t X + D^{-1})^{-1},$$
$$\mu = \Sigma X^t \alpha.$$

The algorithm is $O(np^2)$; for large p it performs fast.

Usage

```
rpost_regression_coef <- function(
  X,
  D,
  alpha,
  u = NULL)
```

Arguments

X	n by p matrix
D	p by p matrix
alpha	n by 1 vector
u	Optional. If specified, don't generate $u \sim N(0, D)$.

Details

The algorithm is from

Bhattacharya, Anirban, Antik Chakraborty, and Bani K. Mallick. "Fast sampling with Gaussian scale mixture priors in high-dimensional regression." Biometrika (2016): asw042.

The algorithm is as follows:

1. Sample $u \sim N(0, D), d \sim N(0, I_n)$
2. Set $v = Xu + d$
3. Solve $(XDX^t + I_n)w = (\alpha - v)$
4. Return $\beta = u + DX^t w$

Value

beta An p x 1 vector

---

shrinkage_regression_plot
Plot the MC chain of regression coeffecients compared to OLS.

Description

Plot the MC chain of regression coeffecients comparet to OLS using 95% HPD confidence intervals.

Usage

```
horseshoe_regression_plot <- function(
  beta,
  Y,
  X,
  main = "Horseshoe Prior",
  ylim = NULL)
```

Arguments

beta	n x p MCMC chain of beta coeffecients.
Y	Y values used to generate beta; used as input to OLS regression
X	X values used to generate beta; used as input to OLS regression
main	Title of the plot
ylim	y range of plot

Examples

```
# Load the data
 prostate.data = "https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data"
prostate = read.table(file = "prostate.data", sep="", header = TRUE)
# Training data:
prostate_train = prostate[which(prostate$train),-10]
# Testing data:
prostate_test = prostate[which(!prostate$train),-10]
# Response:
y = prostate_train$lpsa
# Center and scale the data:
y = scale(y)
# And the predictors
X = scale(prostate_train[,names(prostate_train) != "lpsa"])

gibbs_hs <- horseshoe_regression(y, X, niter=10000)
shrinkage_regression_plot(gibbs_hs$beta, y, X)
```

Index