Package ‘varbvs’

February 20, 2015

Type Package
Title Variational inference for Bayesian variable selection
Version 1.0
Date 2012-03-22
Author Peter Carbonetto
Maintainer Peter Carbonetto <pcarbo@uchicago.edu>
Description Implements the variational inference procedure for Bayesian variable selection, as described in the \"Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies\" (Bayesian Analysis 7, March 2012, pages 73-108). This software has been used to implement Bayesian variable selection for large problems with over a million variables and thousands of samples.
Suggests grid, ggplot2
LazyLoad yes
License GPL (>= 3)
Repository CRAN
Date/Publication 2012-04-10 16:26:08
NeedsCompilation yes

R topics documented:

  varbvs-package .................................................. 2
  create.data ..................................................... 2
  create.snps ..................................................... 4
  grid3d ............................................................. 5
  updatestats ...................................................... 5
  varbvsbinoptimize ............................................. 6
  varbvsoptimize ................................................ 8
  varsimbvs ......................................................... 10

Index 12
varbvs-package  Variational inference for Bayesian variable selection

Description

The varbvs package implements the variational inference procedure for Bayesian variable selection, as described in the "Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies" (Bayesian Analysis 7, March 2012, pages 73-108). This software has been used to implement Bayesian variable selection for large problems with over a million variables and thousands of samples.

Details

The main functions that implement the variational inference for Bayesian variable selection are varbvsoptimize (for linear regression) and varbvsbinoptimize (for logistic regression). Function varsimbvs demonstrates how the variational approximation can be combined with importance sampling to compute the posterior distributions for coefficients and hyperparameters. Accompanying functions that might be useful to users include grid3d, create.snps and create.data.

To see a full example of how the variational inference procedure can be used to compute posterior inclusion probabilities and posterior distributions of the hyperparameters in a linear regression setting, run demo(example1).

To see more details about the varbvs package, including the license and a list of available functions, use help(package="varbvs").

Author(s)

Peter Carbonetto <pcarbo@uchicago.edu>

References


create.data  Simulate SNP data

Description

Generates samples of the genotypes and quantitative trait (i.e. a continuous outcome Y) according to the specified SNP minor allele frequencies and additive effects.

Usage

create.data(maf, beta, sigma, n)
create.data

Arguments

maf vector giving minor allele frequencies of the SNPs.
beta vector of the same length giving the additive effects of the SNPs.
sigma positive scalar specifying the variance of residual.
n number of samples to generate.

Details

Genotypes are generated from a binomial distribution with success rates given by the minor allele frequencies. Observations about the quantitative trait are generated according to \( y = X\beta + e \), where the residual \( e \) is normal with mean zero and covariance \( \sigma \) times the identity matrix.

Value

Returns a list containing two components:

- \( X \) matrix of genotype data with centered columns so that the mean of each column is zero.
- \( y \) vector of quantitative trait data, centered so that the mean of the vector entries is zero.

Author(s)

Peter Carbonetto

See Also

create.snps

Examples

```
## Randomly generate genotypes and quantitative trait measurements for
## 1000 SNPs and 200 individuals, in which the variance of the residual
## error is 4. Of these SNPs, 10 have nonzero additive effects on the
## trait.
se <- 4
snps <- create.snps(1000,10)
data <- create.data(snps$maf,snps$beta,se,200)

## Calculate the proportion of variance in the quantitative trait
## explained by the SNPs.
sz <- var(c(data$X %*% snps$beta))
sz/(sz + se)
```
create.snps

Generate SNP Info

Description
Generates minor allele frequencies and additive effects for genetic loci (specifically, these are single nucleotide polymorphisms, or SNPs for short).

Usage
create.snps(p, n)

Arguments
p  number of SNPs.
n  number of causal SNPs.

Details
Additive effects are generated from the standard normal, and minor allele frequencies are uniform between 0.05 and 0.5.

Value
Returns a list containing two components:
maf  vector with minor allele frequencies of SNPs.
beta  vector with additive effects of SNPs.

Author(s)
Peter Carbonetto

See Also
create.data

Examples
## Generate minor allele frequencies and additive effects for 1000 SNPs; of these SNPs, 10 have nonzero additive effects on the trait.
snps <- create.snps(1000, 10)
grid3d
Generates possible combinations of 3 variables

Description
It is equivalent to ndgrid(x, y, z) in MATLAB.

Usage
grid3d(x, y, z)

Arguments
x, y, z vectors specifying the domains for variables X, Y and Z

Value
Returns a list containing three components, X, Y and Z. Each is an array with one element for every possible combination of X, Y and Z, so that c(X[i], Y[i], Z[i]) is a possible assignment to variables X, Y and Z.

Author(s)
Peter Carbonetto

Examples

## Generate all possible combinations of three binary variables.
grid <- grid3d(0:1, 0:1, 0:1)

updatestats
Statistics for updating variational parameters eta.

Description
updatestats calculates useful quantities for updating the variational approximation to the logistic regression factors. This function should be called whenever the free parameters eta are modified.

Usage
updatestats(X, y, eta)
Arguments

\( x \)  Matrix of observations about the variables (or features). It has \( n \) rows and \( p \) columns, where \( n \) is the number of samples, and \( p \) is the number of variables.

\( y \)  Vector of observations about the binary outcome. It is a vector of length \( n \).

\( \eta \)  Free parameters specifying the variational lower bound on the logistic regression factors. It is a vector of length \( n \).

Value

Returns a list containing five components: \( xy = t(x) \times yhat \), \( xu = t(x) \times u \), \( d = diag(t(x) \times uhat \times x) \), and \( u = \text{sigmoid}(\eta - 0.5)/\eta \), the slope of the conjugate to the log-sigmoid function at \( X \), times 2. For a definition of vectors \( yhat \) and matrix \( uhat \), see the Bayesian Analysis paper.

Author(s)

Peter Carbonetto

Description

\texttt{varbvsbinoptimize} implements the fully-factorized variational approximation for Bayesian variable selection in logistic regression. It finds the "best" fully-factorized variational approximation to the posterior distribution of the coefficients in a logistic regression model of a binary outcome (e.g. disease status in a case-control study), with spike and slab priors on the coefficients. By "best", we mean the approximating distribution that locally minimizes the Kullback-Leibler divergence between the approximating distribution and the exact posterior.

\texttt{varbvsbinupdate} runs a single iteration of the coordinate ascent updates to maximize the variational lower bound or, equivalently, to minimize the Kullback-Leibler divergence objective.

Usage

\begin{verbatim}
varbvsbinoptimize(X, y, sa, logodds, alpha0 = NULL, mu0 = NULL, 
                  eta0 = NULL, fixed.eta = FALSE, verbose = TRUE)

varbvsbinupdate(X, sa, logodds, stats, alpha0, mu0, Xr0, S)
\end{verbatim}

Arguments

\( X \)  Matrix of observations about the variables (or features). It has \( n \) rows and \( p \) columns, where \( n \) is the number of samples, and \( p \) is the number of variables.

\( sa \)  Prior variance of the regression coefficients.
Prior log-odds of inclusion for each variable. It is equal to $\logodds = \log(q/(1-q))$, where $q$ is the prior probability that each variable is included in the linear model of $Y$. It may either be a scalar, in which case all the variables have the same prior inclusion probability, or it may be a vector of length $p$.

Initial variational estimate of posterior inclusion probabilities. It is a vector of length $p$. If $\alpha_0 = NULL$, the variational parameters are initialized at random.

Initial variational estimate of posterior mean coefficients. It is a vector of length $p$. If $\mu_0 = NULL$, the variational parameters are randomly initialized.

Vector of observations about the binary outcome. It is a vector of length $n$.

Initial free parameters specifying the variational lower bound on the logistic regression factors. If $\eta_0 = NULL$, the free parameters are initialized to a vector of ones.

Set $\text{fixed.eta} = TRUE$ to prevent ETA from being updated.

Set $\text{verbose} = FALSE$ to turn off reporting the algorithm’s progress.

This is the return value of $\text{updatestats}(X, y, \eta)$. See $\text{updatestats}$ for more information on this argument.

Equal to $X \%\% (\alpha_0*\mu_0)$.

Order in which the coordinates are updated. It is a vector of any length. Each entry of $S$ must be an integer between 1 and $p$.

Details

$\alpha$, $\mu$ and $s$ are the parameters of the variational approximation and, equivalently, variational estimates of posterior quantities: under the variational approximation, the $i$th regression coefficient is normal with probability $\alpha[i]$; $\mu[i]$ and $s[i]$ are the mean and variance of the coefficient given that it is included in the model. $\alpha$, $\mu$ and $s$ are always column vectors of length $p$.

Unlike $\text{varbvsoptimize}$, $y$ and $X$ must not be centered. Instead, we will account for the intercept as we update the variational approximation.

Also note that the residual variance parameter $\sigma$ is not needed to model a binary trait.

The computational complexity of running $\text{varbvsbinupdate}$ is $O(n*\text{length}(S))$. For efficient computation, most of the work is done by $\text{varbvsbinupdateR}$, a function implemented in C. The call to the C function in the shared library is made using .C.

Value

$\text{varbvsbinoptimize}$ returns a list containing five components: variational parameters $\alpha$, $\mu$ and $s$, the vector of free parameters $\eta$, and the variational estimate of the marginal log-likelihood $lnZ$.

$\text{varbvsbinupdate}$ returns a list containing three components: the updated variational parameters $\alpha$ and $\mu$, and the updated matrix-vector product $Xr$.

Author(s)

Peter Carbonetto
varbvsoptimize

Coordinate ascent for variational approximation to Bayesian variable selection in linear regression

Description

varbvsoptimize implements the fully-factorized variational approximation for Bayesian variable selection in linear regression. It finds the "best" fully-factorized variational approximation to the posterior distribution of the coefficients in a linear regression model of a continuous outcome (quantitative trait), with spike and slab priors on the coefficients. By "best", we mean the approximating distribution that locally minimizes the Kullback-Leibler divergence between the approximating distribution and the exact posterior.

varbvsupdate runs a single iteration of the coordinate ascent updates to maximize the variational lower bound or, equivalently, to minimize the Kullback-Leibler divergence objective.

Usage

```r
varbvsoptimize(x, y, sigma, sa, logodds, alphaP = NULL, muP = NULL, verbose = TRUE)

varbvsupdate(x, sigma, sa, logodds, xy, d, alphaP, muP, XrP, S)
```

Arguments

- **X**: Matrix of observations about the variables (or features). It has \(n\) rows and \(p\) columns, where \(n\) is the number of samples, and \(p\) is the number of variables.
- **sigma**: Scalar giving the variance of the residual.
- **sa**: \(sa \times \text{sigma}\) is the prior variance of the regression coefficients.
- **logodds**: Prior log-odds of inclusion for each variable. It is equal to \(\logodds = \log(q/(1-q))\), where \(q\) is the prior probability that each variable is included in the linear model of \(Y\). It may either be a scalar, in which case all the variables have the same prior inclusion probability, or it may be a vector of length \(p\).
- **alphaP**: Initial variational estimate of posterior inclusion probabilities. It is a vector of length \(p\). If \(alphaP = \text{NULL}\), the variational parameters are initialized at random.
- **muP**: Initial variational estimate of posterior mean coefficients. It is a vector of length \(p\). If \(muP = \text{NULL}\), the variational parameters are randomly initialized.
- **y**: Vector of observations about the outcome. It is a vector of length \(n\).
- **verbose**: Set \(verbose = \text{FALSE}\) to turn off reporting the algorithm’s progress.
- **xy**: Equal to \(t(X) \times y\), where \(y\) is the vector of observations about the outcome.
- **d**: Equal to \(\text{diag}(t(X) \times X)\).
- **XrP**: Equal to \(X \times (alphaP \times muP)\).
- **S**: Order in which the coordinates are updated. It is a vector of any length. Each entry of \(S\) must be an integer between 1 and \(p\).
Details

alpha, \( \mu \) and \( s \) are the parameters of the variational approximation and, equivalently, variational estimates of posterior quantities: under the variational approximation, the \( i \)th regression coefficient is normal with probability \( \alpha[i] \); \( \mu[i] \) and \( s[i] \) are the mean and variance of the coefficient given that it is included in the model. \( \alpha, \mu \) and \( s \) are always column vectors of length \( p \).

To account for an intercept, \( y \) and \( X \) must be centered beforehand so that \( y \) and each column of \( X \) has a mean of zero.

The computational complexity of running \( \text{varbvsupdate} \) is \( O(n \times \text{length}(S)) \). For efficient computation, most of the work is done by \( \text{varbvsupdater} \), a function implemented in C. The call to the C function in the shared library is made using \( .C \).

Value

\( \text{varbvsoptimize} \) returns a list containing four components: variational parameters \( \alpha, \mu \) and \( s \), and the variational estimate of the marginal log-likelihood \( \ln Z \).

\( \text{varbvsupdate} \) returns a list containing three components: the updated variational parameters \( \alpha \) and \( \mu \), and the updated matrix-vector product \( XR \).

Author(s)

Peter Carbonetto

See Also

\( \text{varsimbvs} \)

Examples

```r
## Randomly generate genotypes and quantitative trait measurements for
## 1000 SNPs and 500 individuals, in which the variance of the residual
## error is 4. Of these SNPs, 10 have nonzero additive effects on the
## trait.
se <- 4
snps <- create.snps(1000,10)
data <- create.data(snps$maf,snps$beta,se,500)

## Compute the variational approximation given (appropriate) choices for
## the hyperparameters.
result <- varbvsoptimize(data$x,data$y,4,1/4,log(0.01/0.99))

## View the posterior inclusion probabilities for the (true) causal SNPs.
S <- which(snps$beta != 0)
cbind(snps$beta[S],result$alpha[S])

## View the largest posterior inclusion probability for a SNP that has
## no effect on the quantitative trait.
S <- which(snps$beta == 0)
i <- S[which.max(result$alpha[S])]
cbind(snps$beta[i],result$alpha[i])
```
Description

varsimbvs is a special implementation of the variational inference procedure used in the two simulation studies for the Bayesian Analysis paper. The main distinguishing feature of this procedure is the choice of priors for the hyperparameters of the variable selection model. In addition, we also avoid erratic behaviour in the variational approximation by first searching for a good initialization of the variational parameters. This inference procedure involves an inner loop and an outer loop. The inner loop consists of running a coordinate ascent algorithm to tighten the variational lower bound given a setting of the hyperparameters (this inner loop is implemented by varbvsoptimize). The outer loop computes importance weights for all combinations of the hyperparameters.

Usage

varsimbvs(X, y, sigma, sa, log10q, a, b, ca)

Arguments

X  Matrix of observations about the variables (or features). It has $n$ rows and $p$ columns, where $n$ is the number of samples, and $p$ is the number of variables.

y  Vector of observations about the outcome. It is a vector of length $n$.

sigma, sa, log10q  These arguments specify the combinations of the hyperparameter settings. These inputs must be vectors of the same length. For each combination of the hyperparameters, we compute an importance weight. sigma is the residual variance, $\sigma^*sa$ is the prior variance of the regression coefficients, and $\log_{10}q$ is the (base 10) logarithm of the prior inclusion probability.

a, b, ca  These arguments are all positive scalars specifying the priors on the hyperparameters. $a$ and $b$ are the prior sample sizes for the beta prior on the prior inclusion probability. We assume a uniform prior on the "proportion of variance explained", except that we replace the prior inclusion probability in the proportion of variance explained by a constant, $ca$. This is done purely for convenience, so that hyperparameter $sa$ does not depend on the prior inclusion probability a priori, making it easier to implement the Markov chain Monte Carlo (MCMC) method (see the Bayesian Analysis article for details). We assume the standard noninformative prior on the residual variance $\sigma^2$. 

Value

Returns a list containing three components:

- `alpha`: variational estimates of the posterior inclusion probabilities, averaged over settings of the hyperparameters.
- `mu`: variational estimates of the posterior mean coefficients, averaged over settings of the hyperparameters.
- `w`: normalized importance weights for all settings of the hyperparameters.

Author(s)

Peter Carbonetto

See Also

- `varbvsoptimize`

Examples

```r
## Run the demonstration R Script.
## Not run: demo(example1)
```
Index

.C. 7, 9
create.data. 2, 2, 4
create.snps. 2, 3, 4
grid3d. 2, 5
updatestats. 5, 7
varbvs (varbvs-package). 2
varbvs-package. 2
varbvsbinoptimize. 2, 6
varbvsbinupdate (varbvsbinoptimize). 6
varbvsoptimize. 2, 7, 8, 10, 11
varbvsupdate (varbvsoptimize). 8
varsimbvs. 2, 9, 10