Package 'bayesreg'

November 24, 2016

Type Package

Title Bayesian Regression Models with Continuous Shrinkage Priors

Version 1.0

Date 2016-11-14

Author Daniel F. Schmidt and Enes Makalic

Maintainer Daniel F. Schmidt <dschmidt@unimelb.edu.au>

Description Fits linear or logistic regression model using Bayesian continuous shrinkage prior distributions. Handles ridge, lasso, horseshoe and horseshoe+ regression with logistic, Gaussian, Laplace or Student-t distributed targets.

License GPL-2

Depends methods (>= 2.0), BayesLogit (>= 0.6)

RoxygenNote 5.0.1.9000

NeedsCompilation no

Repository CRAN

Date/Publication 2016-11-24 11:31:05

R topics documented:

`bayesreg` *Fitting Bayesian Regression Models with Continuous Shrinkage Priors*

Description

Fit a linear or logistic regression model using Bayesian continuous shrinkage prior distributions. Handles ridge, lasso, horseshoe and horseshoe+ regression with logistic, Gaussian, Laplace or Student-t distributed targets.

Usage

```
bayesreg(formula, data, model = "normal", prior = "ridge",
     nsamples = 1000, burnin = 1000, thin = 5, tdof = 5)
```

Arguments

formula	an object of class "formula": a symbolic description of the model to be fitted using the standard R formula notation.
data	an data frame containing the variables in the model.
model	the distribution of the target (y) variable. Continuous or numeric variables can be distributed as per a Gaussian distribution (model="gaussian" or model="normal"), Laplace distribution (model = "laplace" or model = "l1") or Student-t distribution ("model" = "studentt" or "model" = "t"). For binary targets (factors with two levels) either model="logistic" or "model"="binomial" should be used.
prior	The continuous shrinkage prior distribution over the regression coefficients that is to be used. Options include ridge regression (prior="rr" or prior="ridge"), lasso regression (prior="lasso"), horseshoe regression (prior="hs" or prior="horseshoe") and horseshoe+ regression (prior="hs+" or prior="horseshoe+").
nsamples	number of posterior samples to generate.
burnin	number of burn-in samples.
thin	level of thinning.
tdof	degrees of freedom for the Student-t distribution.

Details

Draws a series of samples from the posterior distribution of a linear (Gaussian, Laplace or Student-t) or logistic regression model with specified continuous shrinkage prior distribution (ridge regression, lasso, horseshoe and horseshoe+) using Gibbs sampling. The intercept parameter is always included, and is never penalised.

While only nsamples are returned, the total number of samples generated is equal to burnin+nsamples*thin. To generate the samples of the regression coefficients, the code will use either Rue's algorithm (when the number of samples is twice the number of covariates) or the algorithm of Bhattacharya et al. as appropriate.

Value

An object with S3 class "bayesreg".

beta	Posterior samples the regression model coefficients
beta0	Posterior samples of the intercept parameter
sigma2	Posterior samples of the square of the scale parameter; for Gaussian distributed targets this is equal to the variance. For binary distributed targets this is empty.
muBeta	The mean of the posterior samples for the regression coefficients.
muBeta0	The mean of the posterior samples for the intercept parameter.

muSigma2	The mean of the posterior samples for squared scale parameter.
tau2	Posterior samples of the global shrinkage parameter.
tstat	Posterior t-statistics for each regression coefficient.
varranks	Ranking of the covariates by their importance, with "1" denoting the most important covariate.
logl	The log-likelihood at the posterior means of the model parameters
dic	The Deviance Information Criterion (DIC) score for the model

The returned object also stores the parameters/options used to run bayesreg:

formula	The object of type "formula" describing the fitted model.
model	The distribution of the target (y) variable.
prior	The shrinkage prior used to fit the model.
nsamples	The number of samples generated from the posterior distribution.
burnin	The number of burnin samples that were generated.
thin	The level of thinning.
n	The sample size of the data used to fit the model.
p	The number of covariates in the fitted model.

Note

To cite this toolbox please reference:

Makalic, E. & Schmidt, D. F. High-Dimensional Bayesian Regularised Regression with the BayesReg Package arXiv:1611.06649 [stat.CO], 2016 https://arxiv.org/pdf/1611.06649.pdf

A MATLAB implementation of the bayesreg function is also available from:

https://au.mathworks.com/matlabcentral/fileexchange/60335-bayesian-regularised-linear-and-logistic-

Author(s)

Enes Makalic and Daniel F. Schmidt

Maintainer: Daniel F. Schmidt <dschmidt@unimelb.edu.au>

References

Makalic, E. & Schmidt, D. F. High-Dimensional Bayesian Regularised Regression with the BayesReg Package arXiv:1611.06649 [stat.CO], 2016 https://arxiv.org/pdf/1611.06649.pdf

Park, T. & Casella, G. The Bayesian Lasso Journal of the American Statistical Association, Vol. 103, pp. 681-686, 2008

Carvalho, C. M.; Polson, N. G. & Scott, J. G. The horseshoe estimator for sparse signals Biometrika, Vol. 97, 465-480, 2010

Makalic, E. & Schmidt, D. F. A Simple Sampler for the Horseshoe Estimator IEEE Signal Processing Letters, Vol. 23, pp. 179-182, 2016 https://arxiv.org/pdf/1508.03884v4.pdf

Bhadra, A.; Datta, J.; Polson, N. G. & Willard, B. The Horseshoe+ Estimator of Ultra-Sparse Signals Bayesian Analysis, 2016

Polson, N. G.; Scott, J. G. & Windle, J. Bayesian inference for logistic models using Polya-Gamma latent variables Journal of the American Statistical Association, Vol. 108, 1339-1349, 2013

Rue, H. Fast sampling of Gaussian Markov random fields Journal of the Royal Statistical Society (Series B), Vol. 63, 325-338, 2001

Bhattacharya, A.; Chakraborty, A. & Mallick, B. K. Fast sampling with Gaussian scale-mixture priors in high-dimensional regression arXiv:1506.04778, 2016

See Also

predict.bayesreg and summary.bayesreg

Examples

```
# ------------------------------------------------------------------
# Example 1: Gaussian regression
X = matrix(rnorm(100*20),100,20)
b = matrix(0,20,1)
b[1:5] = c(5,4,3,2,1)
y = X %*% b + rnorm(100, 0, 1)

df <- data.frame(X,y)
rv_lm <- lm(y~.,df)                       # Regular least-squares
summary(rv_lm)

rv_hs <- bayesreg(y~.,df,prior="hs")      # Horseshoe regression
rv_hs.s <- summary(rv_hs)

# Expected squared prediction error for least-squares
coef_ls = coef(rv_lm)
as.numeric(sum( (as.matrix(coef_ls[-1]) - b)^2 ) + coef_ls[1]^2)

# Expected squared prediction error for horseshoe
as.numeric(sum( (rv_hs$muBeta - b)^2 ) + rv_hs$muBeta0^2)


# ------------------------------------------------------------------
# Example 2: Gaussian v Student-t robust regression
X = 1:10;
y = c(-0.6867, 1.7258, 1.9117, 6.1832, 5.3636, 7.1139, 9.5668, 10.0593, 11.4044, 6.1677);
df = data.frame(X,y)

# Gaussian ridge
rv_G <- bayesreg(y~., df, model = "gaussian", prior = "ridge", nsamples = 1e3)

# Student-t ridge
rv_t <- bayesreg(y~., df, model = "t", prior = "ridge", tdof = 5, nsamples = 1e3)

# Plot the different estimates with credible intervals
plot(df$X, df$y, xlab="x", ylab="y")

yhat_G <- predict(rv_G, df, bayesavg=TRUE)
```

```
lines(df$X, yhat_G[,1], col="blue", lwd=2.5)
lines(df$X, yhat_G[,3], col="blue", lwd=1, lty="dashed")
lines(df$X, yhat_G[,4], col="blue", lwd=1, lty="dashed")

yhat_t <- predict(rv_t, df, bayesavg=TRUE)
lines(df$X, yhat_t[,1], col="darkred", lwd=2.5)
lines(df$X, yhat_t[,3], col="darkred", lwd=1, lty="dashed")
lines(df$X, yhat_t[,4], col="darkred", lwd=1, lty="dashed")

legend(1,11,c("Gaussian","Student-t (dof=5)"),lty=c(1,1),col=c("blue","darkred"),
       lwd=c(2.5,2.5), cex=0.7)

## Not run:
# ------------------------------------------------------------------
# Example 3: Logistic regression on spambase
data(spambase)

# bayesreg expects binary targets to be factors
spambase$is.spam <- factor(spambase$is.spam)

# First take a subset of the data (1/10th) for training, reserve the rest for testing
spambase.tr  = spambase[seq(1,nrow(spambase),10),]
spambase.tst = spambase[-seq(1,nrow(spambase),10),]

# Fit a model using logistic horseshoe for 2,000 samples
rv <- bayesreg(is.spam ~ ., spambase.tr, model = "logistic", prior = "horseshoe", nsamples = 2e3)

# Summarise, sorting variables by their ranking importance
rv.s <- summary(rv,sortrank=TRUE)

# Make predictions about testing data -- get class predictions and class probabilities
y_pred <- predict(rv, spambase.tst, type='class')

# Check how well did our predictions did by generating confusion matrix
table(y_pred, spambase.tst$is.spam)

# Calculate logarithmic loss on test data
y_prob <- predict(rv, spambase.tst, type='prob')
cat('Neg Log-Like for no Bayes average, posterior mean estimates: ', sum(-log(y_prob[,1])))
y_prob <- predict(rv, spambase.tst, type='prob', sum.stat="median")
cat('Neg Log-Like for no Bayes average, posterior median estimates: ', sum(-log(y_prob[,1])))
y_prob <- predict(rv, spambase.tst, type='prob', bayesavg=TRUE)
cat('Neg Log-Like for Bayes average: ', sum(-log(y_prob[,1])))

## End(Not run)
```

`predict.bayesreg` *Predict method for Bayesian penalised regression (bayesreg) fits*

Description

Predict values based on Bayesian penalised regression fits

Usage

```
## S3 method for class 'bayesreg'
predict(object, df, type = "linpred", bayesavg = FALSE, sum.stat = "mean", ...)
```

Arguments

object	an object of class "bayesreg" created as a result of a call to bayesreg.
df	A data frame providing the variables from which to produce predictions.
type	The type of predictions to produce; if type="linpred" it will return the linear predictor for both binary and continuous data. For binary data, if type="prob" it will return predictive probability estimates, and if type="class" and the data is binary, it will return the best guess at the class of the target variable.
bayesavg	Whether to produce predictions using Bayesian averaging.
sum.stat	The type of summary statistic to use; either sum.stat="mean" or sum.stat="median".
...	further arguments passed to or from other methods.

Details

predict.bayesreg produces predicted values using variables from the specified data frame. The type of predictions produced depend on the value of the parameter type.

If type="linpred", the predictions that are returned will be the value of the linear predictor formed from the model coefficients and the provided data.

If type="prob", the predictions will be probabilities. If the specified data frame includes a column with the same name as the target variable on which the model was created, the predictions will then be the probability density values for these target values. For binary data, if the specified data frame does not include a column with the same name as the target variable, the predictions will be the probability of the target being equal to the second level of the factor variable.

If type="class" and the target variable is binary, the predictions will be the most likely class.

If bayesavg is FALSE the predictions will be produced by using a summary of the posterior samples of the coefficients and scale parameters as estimates for the model. If bayesavg is TRUE, the predictions will be produced by posterior averaging over the posterior samples of the coefficients and scale parameters, allowing the uncertainty in the estimation process to be explicitly taken into account in the prediction process.

If sum.stat="mean" and bayesavg is FALSE, the mean of the posterior samples will be used as point estimates for making predictions. Likewise, if sum.stat="median" and bayesavg is FALSE, the co-ordinate wise posterior medians will be used as estimates for making predictions. If bayesavg is TRUE and type!="prob", the posterior mean (median) of the predictions from each of the posterior samples will be used as predictions. The value of sum.stat has no effect if type="prob".

Value

predict.bayesreg produces a vector or matrix of predictions of the specified type. If bayesavg is FALSE a matrix with a single column pred is returned, containing the predictions.

If bayesavg is TRUE, three additional columns are returned: se(pred), which contains standard errors for the predictions, and CI 5% and CI 95% which contain 95% credible intervals for the predictions.

Author(s)

Enes Makalic and Daniel F. Schmidt

Maintainer: Daniel F. Schmidt <dschmidt@unimelb.edu.au>

See Also

The model fitting function bayesreg and summary function summary.bayesreg

Examples

```
# ------------------------------------------------------------------
# Example 1: Fitting linear models to data and generating credible intervals
X = 1:10;
y = c(-0.6867, 1.7258, 1.9117, 6.1832, 5.3636, 7.1139, 9.5668, 10.0593, 11.4044, 6.1677);
df = data.frame(X,y)

# Gaussian ridge
rv_L <- bayesreg(y~., df, model = "laplace", prior = "ridge", nsamples = 1e3)

# Plot the different estimates with credible intervals
plot(df$X, df$y, xlab="x", ylab="y")

yhat <- predict(rv_L, df, bayesavg=TRUE)
lines(df$X, yhat[,1], col="blue", lwd=2.5)
lines(df$X, yhat[,3], col="blue", lwd=1, lty="dashed")
lines(df$X, yhat[,4], col="blue", lwd=1, lty="dashed")
yhat <- predict(rv_L, df, bayesavg=TRUE, sum.stat = "median")
lines(df$X, yhat[,1], col="red", lwd=2.5)

legend(1,11,c("Posterior Mean (Bayes Average)","Posterior Median (Bayes Average)"),
       lty=c(1,1),col=c("blue","red"),lwd=c(2.5,2.5), cex=0.7)


# ------------------------------------------------------------------
# Example 2: Predictive density for continuous data
X = 1:10;
y = c(-0.6867, 1.7258, 1.9117, 6.1832, 5.3636, 7.1139, 9.5668, 10.0593, 11.4044, 6.1677);
df = data.frame(X,y)

# Gaussian ridge
rv_G <- bayesreg(y~., df, model = "gaussian", prior = "ridge", nsamples = 1e3)
```

```
# Produce predictive density for X=2
df.tst = data.frame(y=seq(-7,12,0.01),X=2)
prob_noavg_mean <- predict(rv_G, df.tst, bayesavg=FALSE, type="prob", sum.stat = "mean")
prob_noavg_med  <- predict(rv_G, df.tst, bayesavg=FALSE, type="prob", sum.stat = "median")
prob_avg        <- predict(rv_G, df.tst, bayesavg=TRUE, type="prob")

# Plot the density
plot(NULL, xlim=c(-7,12), ylim=c(0,0.14), xlab="y", ylab="p(y)")
lines(df.tst$y, prob_noavg_mean[,1],lwd=1.5)
lines(df.tst$y, prob_noavg_med[,1], col="red",lwd=1.5)
lines(df.tst$y, prob_avg[,1], col="green",lwd=1.5)

legend(-7,0.14,c("Mean (no averaging)","Median (no averaging)","Bayes Average"),
       lty=c(1,1,1),col=c("black","red","green"),lwd=c(1.5,1.5,1.5), cex=0.7)


## Not run:
# ------------------------------------------------------------------
# Example 3: Logistic regression on spambase
data(spambase)

# bayesreg expects binary targets to be factors
spambase$is.spam <- factor(spambase$is.spam)

# First take a subset of the data (1/10th) for training, reserve the rest for testing
spambase.tr  = spambase[seq(1,nrow(spambase),10),]
spambase.tst = spambase[-seq(1,nrow(spambase),10),]

# Fit a model using logistic horseshoe for 2,000 samples
rv <- bayesreg(is.spam ~ ., spambase.tr, model = "logistic", prior = "horseshoe", nsamples = 2e3)

# Summarise, sorting variables by their ranking importance
rv.s <- summary(rv,sortrank=TRUE)

# Make predictions about testing data -- get class predictions and class probabilities
y_pred <- predict(rv, spambase.tst, type='class')
y_prob <- predict(rv, spambase.tst, type='prob')

# Check how well our predictions did by generating confusion matrix
table(y_pred, spambase.tst$is.spam)

# Calculate logarithmic loss on test data
y_prob <- predict(rv, spambase.tst, type='prob')
cat('Neg Log-Like for no Bayes average, posterior mean estimates: ', sum(-log(y_prob[,1])))
y_prob <- predict(rv, spambase.tst, type='prob', sum.stat="median")
cat('Neg Log-Like for no Bayes average, posterior median estimates: ', sum(-log(y_prob[,1])))
y_prob <- predict(rv, spambase.tst, type='prob', bayesavg=TRUE)
cat('Neg Log-Like for Bayes average: ', sum(-log(y_prob[,1])))

## End(Not run)
```

summary.bayesreg *Summarize Bayesian penalised regression (bayesreg) fits*

Description

summary method for class "bayesreg".

Usage

```
## S3 method for class 'bayesreg'
summary(object, sortrank = FALSE, displayor = FALSE, ...)
```

Arguments

object	an object of class "bayesreg" created as a result of a call to bayesreg.
sortrank	logical; if TRUE, the variables in the summary will be sorted by their importance as determined by their rank estimated by the Bayesian feature ranking algorithm.
displayor	logical; if TRUE, the variables will be summarised in terms of their cross-sectional odds-ratios rather than their regression coefficients.
...	further arguments passed to or from other methods.

Details

The summary method computes a number of summary statistics and displays these for each variable in a table, along with suitable header information.

For continuous target variables, the header information includes an estimate of the residual squared error, the R^2 statistic and the Deviance information criterion (DIC) statistic. For logistic regression models, the header information includes the negative log-likelihood at the posterior mean of the regression coefficients, the pseudo R^2 score and the DIC statistic.

The main table summarises properties of the coefficients for each of the variables. The first column is the variable name. The second and third columns are either the mean and standard error of the coefficients, or the median and standard error of the cross-sectional odds-ratios if displayor=TRUE.

The fourth and fifth columns are the 95% credible intervals of the coefficients (odds-ratios). The sixth column displays the posterior t-statistic, calculated as the ratio of the posterior mean on the posterior standard deviation for the coefficient. The seventh column is the importance rank assigned to the variable by the Bayesian feature ranking algorithm.

In betwen the seventh and eighth columns are up to two asterisks indicating significance; a variable scores a first asterisk if the 75% credible interval does not include zero, and scores a second if the 95% credible interval does not include zero. The final column gives an estimate of the effective sample size for the variable, ranging from 0 to 100, which indicates how much autocorrelation is present in the sample chain for this variable (smaller values indicating high levels of autocorrelation).

Value

Returns an object with the following fields:

`logl`	The log-likelihood of the model at the posterior mean estimates of the regression coefficients.
`dic`	The deviance information criterion score of the model.
`rsqr`	For non-binary data, the R^2 statistic.
`rootmse`	For non-binary data, the estimated root-mean-squared residual error.
`p_rsqr`	For binary data, the pseudo-R^2 statistic.
`muCoef`	The posterior means of the regression coefficients.
`seCoef`	The posterior standard deviations of the regression coefficients.
`CICoef`	The posterior 95% credible interval for the regression coefficients.
`medOR`	For binary data, the posterior median of the cross-sectional odds-ratios.
`seOR`	For binary data, the posterior standard deviation of the cross-sectional odds-ratios.
`CIOR`	For binary data, the posterior 95% credible interval for the cross-sectional odds-ratios.
`tStat`	The posterior t-statistic for the coefficients.
`nStars`	The significance level for the variable (see above).
`rank`	The variable importance rank as estimated by the Bayesian feature ranking algorithm (see above).
`ESS`	The effective sample size for the variable.

Author(s)

Enes Makalic and Daniel F. Schmidt

Maintainer: Daniel F. Schmidt <dschmidt@unimelb.edu.au>

See Also

The model fitting function `bayesreg` and prediction function `predict.bayesreg`

Examples

```
X = matrix(rnorm(100*20),100,20)
b = matrix(0,20,1)
b[1:9] = c(0,0,0,0,5,4,3,2,1)
y = X %*% b + rnorm(100, 0, 1)
df <- data.frame(X,y)

rv_hs <- bayesreg(y~.,df,prior="hs")       # Horseshoe regression

# Summarise without sorting by variable rank
rv_hs.s <- summary(rv_hs)
```

```
# Summarise sorting by variable rank
rv_hs.s <- summary(rv_hs, sortrank = TRUE)
```

Index