Package: sparsevb (via r-universe)

October 14, 2026

Type Package

Title Spike-and-Slab Variational Bayes for Linear and Logistic Regression

Version 0.1.1

Date 2025-1-23

Maintainer Gabriel Clara <gabriel.j.clara@gmail.com>

Description Implements variational Bayesian algorithms to perform scalable variable selection for sparse, high-dimensional linear and logistic regression models. Features include a novel prioritized updating scheme, which uses a preliminary estimator of the variational means during initialization to generate an updating order prioritizing large, more relevant, coefficients. Sparsity is induced via spike-and-slab priors with either Laplace or Gaussian slabs. By default, the heavier-tailed Laplace density is used. Formal derivations of the algorithms and asymptotic consistency results may be found in Kolyan Ray and Botond Szabo (JASA 2020) and Kolyan Ray, Botond Szabo, and Gabriel Clara (NeurIPS 2020).

BugReports https://gitlab.com/gclara/varpack/-/issues

License GPL (>= 3)

Imports Rcpp (>= 1.0.5), selectiveInference (>= 1.2.5), glmnet (>= 4.0-2), stats

LinkingTo Rcpp, RcppArmadillo, RcppEnsmallen

Encoding UTF-8

RoxygenNote 7.1.1

NeedsCompilation yes

Author Gabriel Clara [aut, cre], Botond Szabo [aut], Kolyan Ray [aut]

Repository https://cran.r-universe.dev

Date/Publication 2025-01-24 10:10:02 UTC

RemoteUrl https://github.com/cran/sparsevb

RemoteRef HEAD

RemoteSha eb538d4b3a5af28438585135340a33873b3bea95

Contents

sparsevb-package 2
svb.fit 3

Index **6**

`sparsevb-package` *sparsevb: Spike-and-Slab Variational Bayes for Linear and Logistic Regression*

Description

Implements variational Bayesian algorithms to perform scalable variable selection for sparse, high-dimensional linear and logistic regression models. Features include a novel prioritized updating scheme, which uses a preliminary estimator of the variational means during initialization to generate an updating order prioritizing large, more relevant, coefficients. Sparsity is induced via spike-and-slab priors with either Laplace or Gaussian slabs. By default, the heavier-tailed Laplace density is used. Formal derivations of the algorithms and asymptotic consistency results may be found in Kolyan Ray and Botond Szabo (2020) <doi:10.1080/01621459.2020.1847121> and Kolyan Ray, Botond Szabo, and Gabriel Clara (2020) <arXiv:2010.11665>.

Details

For details as they pertain to using the package, consult the `svb.fit` function help page. Detailed descriptions and derivations of the variational algorithms with Laplace slabs may be found in the references.

Author(s)

Maintainer: Gabriel Clara <gabriel.j.clara@gmail.com>

Authors:

- Botond Szabo
- Kolyan Ray

References

- Ray K. and Szabo B. Variational Bayes for high-dimensional linear regression with sparse priors. (2020). *Journal of the American Statistical Association*.
- Ray K., Szabo B., and Clara G. Spike and slab variational Bayes for high dimensional logistic regression. (2020). *Advances in Neural Information Processing Systems 33*.

See Also

Useful links:

- Report bugs at https://gitlab.com/gclara/varpack/-/issues

svb.fit *Fit Approximate Posteriors to Sparse Linear and Logistic Models*

Description

Main function of the sparsevb package. Computes mean-field posterior approximations for both linear and logistic regression models, including variable selection via sparsity-inducing spike and slab priors.

Usage

```
svb.fit(
  X,
  Y,
  family = c("linear", "logistic"),
  slab = c("laplace", "gaussian"),
  mu,
  sigma = rep(1, ncol(X)),
  gamma,
  alpha,
  beta,
  prior_scale = 1,
  update_order,
  intercept = FALSE,
  noise_sd,
  max_iter = 1000,
  tol = 1e-05
)
```

Arguments

X	A numeric design matrix, each row of which represents a vector of covariates/independent variables/features. Though not required, it is recommended to center and scale the columns to have norm sqrt(nrow(X)).
Y	An nrow(X)-dimensional response vector, numeric if family = "linear" and binary if family = "logistic".
family	A character string selecting the regression model, either "linear" or "logistic".
slab	A character string specifying the prior slab density, either "laplace" or "gaussian".
mu	An ncol(X)-dimensional numeric vector, serving as initial guess for the variational means. If omitted, mu will be estimated via ridge regression to initialize the coordinate ascent algorithm.
sigma	A positive ncol(X)-dimensional numeric vector, serving as initial guess for the variational standard deviations.
gamma	An ncol(X)-dimensional vector of probabilities, serving as initial guess for the variational inclusion probabilities. If omitted, gamma will be estimated via LASSO regression to initialize the coordinate ascent algorithm.

`alpha`	A positive numeric value, parametrizing the beta hyper-prior on the inclusion probabilities. If omitted, `alpha` will be chosen empirically via LASSO regression.
`beta`	A positive numeric value, parametrizing the beta hyper-prior on the inclusion probabilities. If omitted, `beta` will be chosen empirically via LASSO regression.
`prior_scale`	A numeric value, controlling the scale parameter of the prior slab density. Used as the scale parameter λ when `prior = "laplace"`, or as the standard deviation σ if `prior = "gaussian"`.
`update_order`	A permutation of `1:ncol(X)`, giving the update order of the coordinate-ascent algorithm. If omitted, a data driven updating order is used, see *Ray and Szabo (2020)* in *Journal of the American Statistical Association* for details.
`intercept`	A Boolean variable, controlling if an intercept should be included. NB: This feature is still experimental in logistic regression.
`noise_sd`	A positive numerical value, serving as estimate for the residual noise standard deviation in linear regression. If missing it will be estimated, see `estimateSigma` from the `selectiveInference` package for more details. Has no effect when `family = "logistic"`.
`max_iter`	A positive integer, controlling the maximum number of iterations for the variational update loop.
`tol`	A small, positive numerical value, controlling the termination criterion for maximum absolute differences between binary entropies of successive iterates.

Details

Suppose θ is the p-dimensional true parameter. The spike-and-slab prior for θ may be represented by the hierarchical scheme

$$\begin{aligned} w &\sim \mathrm{Beta}(\alpha, \beta), \\ z_j \mid w &\sim_{i.i.d.} \mathrm{Bernoulli}(w), \\ \theta_j \mid z_j &\sim_{ind.} (1 - z_j)\delta_0 + z_j g. \end{aligned}$$

Here, δ_0 represents the Dirac measure at 0. The slab g may be taken either as a $\mathrm{Laplace}(0, \lambda)$ or $N(0, \sigma^2)$ density. The former has centered density

$$f_\lambda(x) = \frac{\lambda}{2} e^{-\lambda |x|}.$$

Given α and β, the beta hyper-prior has density

$$b(x \mid \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 t^{\alpha-1}(1-t)^{\beta-1}\,\mathrm{d}t}.$$

A straightforward integration shows that the prior inclusion probability of a coefficient is $\frac{\alpha}{\alpha+\beta}$.

Value

The approximate mean-field posterior, given as a named list containing numeric vectors `"mu"`, `"sigma"`, `"gamma"`, and a value `"intercept"`. The latter is set to `NA` in case `intercept = FALSE`. In mathematical terms, the conditional distribution of each θ_j is given by

$$\theta_j \mid \mu_j, \sigma_j, \gamma_j \sim_{ind.} \gamma_j N(\mu_j, \sigma^2) + (1 - \gamma_j)\delta_0.$$

Examples

```
### Simulate a linear regression problem of size n times p, with sparsity level s ###

n <- 250
p <- 500
s <- 5

### Generate toy data ###

X <- matrix(rnorm(n*p), n, p) #standard Gaussian design matrix

theta <- numeric(p)
theta[sample.int(p, s)] <- runif(s, -3, 3) #sample non-zero coefficients in random locations

pos_TR <- as.numeric(theta != 0) #true positives

Y <- X %*% theta + rnorm(n) #add standard Gaussian noise

### Run the algorithm in linear mode with Laplace prior and prioritized initialization ###

test <- svb.fit(X, Y, family = "linear")

posterior_mean <- test$mu * test$gamma #approximate posterior mean

pos <- as.numeric(test$gamma > 0.5) #significant coefficients

### Assess the quality of the posterior estimates ###

TPR <- sum(pos[which(pos_TR == 1)])/sum(pos_TR) #True positive rate

FDR <- sum(pos[which(pos_TR != 1)])/max(sum(pos), 1) #False discovery rate

L2 <- sqrt(sum((posterior_mean - theta)^2)) #L_2-error

MSPE <- sqrt(sum((X %*% posterior_mean - Y)^2)/n) #Mean squared prediction error
```

Index

sparsevb, *3*
sparsevb *(*sparsevb-package*)*, 2
sparsevb-package, 2
svb.fit, *2*, 3