Package 'reglogit'

November 19, 2017

Type Package

Title Simulation-Based Regularized Logistic Regression

Version 1.2-5

Date 2017-11-17

Author Robert B. Gramacy <rbg@vt.edu>

Maintainer Robert B. Gramacy <rbg@vt.edu>

Depends R (>= 2.14.0), methods, mvtnorm, boot, Matrix

Suggests plgp

Description Regularized (polychotomous) logistic regression
by Gibbs sampling. The package implements subtly different
MCMC schemes with varying efficiency depending on the data type
(binary v. binomial, say) and the desired estimator (regularized maximum
likelihood, or Bayesian maximum a posteriori/posterior mean, etc.) through a
unified interface.

License LGPL

URL http://bobby.gramacy.com/r_packages/reglogit

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-11-19 18:22:50 UTC

R topics documented:

reglogit-package 2
pima 3
predict.reglogit 4
reglogit 6

Index **11**

reglogit-package *Simulation-based Regularized Logistic Regression*

Description

Regularized (polychotomous) logistic regression by Gibbs sampling. The package implements subtly different MCMC schemes with varying efficiency depending on the data type (binary v. binomial, say) and the desired estimator (regularized maximum likelihood, or Bayesian maximum a posteriori/posterior mean, etc.) through a unified interface.

Details

Package:	reglogit
Type:	Package
Version:	1.0
Date:	2011-08-05
License:	LGPL
LazyLoad:	yes

See the documentation for the `reglogit` function

Author(s)

Robert B. Gramacy <rbg@vt.edu>

Maintainer: Robert B. Gramacy <rbg@vt.edu>

References

R.B. Gramacy, N.G. Polson. "Simulation-based regularized logistic regression". (2010); arXiv:1005.3430; http://arxiv.org/abs/1005.3430

See Also

`reglogit`, `blasso` and `regress`

Examples

```
## see the help file for the reglogit function
```

pima *Pima Indian Data*

Description

A population of women who were at least 21 years old, of Pima Indian heritage and living near Phoenix, Arizona, was tested for diabetes according to World Health Organization criteria. The data were collected by the US National Institute of Diabetes and Digestive and Kidney Diseases.

Usage

```
data(pima)
```

Format

A data frame with 768 observations on the following 9 variables.

`npreg` number of pregnancies

`glu` plasma glucose concentration in an oral glucose tolerance test

`bp` diastolic blood pressure (mm Hg)

`skin` tricepts skin fold thickness (mm)

`serum` 2-hour serum insulin (mu U/ml)

`bmi` mody mass index (weight in kg/(height in m)\^2)

`ped` diabetes pedigree function

`age` age in years

`y` classification label: `1` for diabetic

Source

Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C. and Johannes, R. S. (1988) *Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications in Medical Care (Washington, 1988)*, ed. R. A. Greenes, pp. 261-265. Los Alamitos, CA: IEEE Computer Society Press.

Ripley, B.D. (1996) *Pattern Recognition and Neural Networks.* Cambridge: Cambridge University Press.

UCI Machine Learning Repository
http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes

Examples

```
data(pima)
## see reglogit documentation for an example using this data
```

predict.reglogit *Prediction for regularized (polychotomous) logistic regression models*

Description

Sampling from the posterior predictive distribution of a regularized (multinomial) logistic regression fit, including entropy information for variability assessment

Usage

```
## S3 method for class 'reglogit'
predict(object, XX, burnin = round(0.1 * nrow(object$beta)), ...)
## S3 method for class 'regmlogit'
predict(object, XX, burnin = round(0.1 * dim(object$beta)[1]), ...)
```

Arguments

object	a "reglogit"-class object or a "regmlogit"-class object, depending on whether binary or polychotomous methods were used for fitting
XX	a matrix of predictive locations where ncol(XX) == object$ncol(XX).
burnin	a scalar positive integer indicate the number of samples of object$beta to discard as burn-in; the default is 10% of the number of samples
...	For compatibility with generic predict method; not used

Details

Applies the logit transformation (reglogit) or multinomial logit (regmlogit) to convert samples of the linear predictor at XX into a samples from a predictive posterior probability distribution. The raw probabilties, averages (posterior means), entropies, and posterior mean casses (arg-max of the average probabilities) are returned.

Value

The output is a list with components explained below. For predict.regmlogit everyhing (except entropy) is expanded by one dimension into an array or matrix as appropriate.

p	a nrow(XX) x (T-burnin) sized matrix of probabilities (of class 1) from the posterior predictive distribution.
mp	a vector of average probablities calculated over the rows of p
pc	class labels formed by rouding (or arg max for predict.regmlogit) the values in mp
ent	The posterior mean entropy given the probabilities in mp

Author(s)

Robert B. Gramacy <rbg@vt.edu>

References

R.B. Gramacy, N.G. Polson. "Simulation-based regularized logistic regression". (2012) Bayesian Analysis, 7(3), p567-590; arXiv:1005.3430; http://arxiv.org/abs/1005.3430

C. Holmes, K. Held (2006). "Bayesian Auxilliary Variable Models for Binary and Multinomial Regression". Bayesian Analysis, 1(1), p145-168.

See Also

reglogit and regmlogit

Examples

```
## see reglogit for a full example of binary classifiction complete with
## sampling from the posterior predictive distribution.

## the example here is for polychotomous classification and prediction

## Not run:
library(plgp)
x <- seq(-2, 2, length=40)
X <- expand.grid(x, x)
C <- exp2d.C(X)
xx <- seq(-2, 2, length=100)
XX <- expand.grid(xx, xx)
CC <- exp2d.C(XX)

## build cubically-expanded design matrix (with interactions)
Xe <- cbind(X, X[,1]^2, X[,2]^2, X[,1]*X[,2],
            X[,1]^3, X[,2]^3, X[,1]^2*X[,2], X[,2]^2*X[,1],
            (X[,1]*X[,2])^2)

## perform MCMC
T <- 1000
out <- regmlogit(T, C, Xe, nu=6, normalize=TRUE)

## create predictive (cubically-expanded) design matrix
XX <- as.matrix(XX)
XXe <- cbind(XX, XX[,1]^2, XX[,2]^2, XX[,1]*XX[,2],
             XX[,1]^3, XX[,2]^3, XX[,1]^2*XX[,2], XX[,2]^2*XX[,1],
             (XX[,1]*XX[,2])^2)

## predict class labels
p <- predict(out, XXe)

## make an image of the predictive surface
cols <- c(gray(0.85), gray(0.625), gray(0.4))
par(mfrow=c(1,3))
image(xx, xx, matrix(CC, ncol=length(xx)), col=cols, main="truth")
image(xx, xx, matrix(p$c, ncol=length(xx)), col=cols, main="predicted")
image(xx, xx, matrix(p$ent, ncol=length(xx)), col=heat.colors(128),
      main="entropy")
```

```
## End(Not run)
```

`reglogit` *Gibbs sampling for regularized logistic regression*

Description

Regularized (multinomial) logistic regression by Gibbs sampling implementing subtly different MCMC schemes with varying efficiency depending on the data type (binary v. binomial, say) and the desired estimator (regularized maximum likelihood, or Bayesian maximum a posteriori/posterior mean, etc.) through a unified interface.

Usage

```
reglogit(T, y, X, N = NULL, flatten = FALSE, sigma = 1, nu = 1,
      kappa = 1, icept = TRUE, normalize = TRUE, zzero = TRUE,
      powerprior = TRUE, kmax = 442, bstart = NULL, lt = NULL,
      nup = list(a = 2, b = 0.1), save.latents = FALSE, verb = 100)
regmlogit(T, y, X, flatten = FALSE, sigma = 1, nu = 1, kappa = 1,
      icept=TRUE, normalize = TRUE, zzero = TRUE, powerprior = TRUE,
      kmax = 442, bstart = NULL, lt = NULL, nup = list(a=2, b=0.1),
      save.latents = FALSE, verb=100)
```

Arguments

`T`	a positive integer scalar specifying the number of MCMC rounds
`y`	`reglogit` requires `logical` classification labels for Bernoulli data, or counts for Binomial data; for the latter, `N` must also be specified. `regmlogit` requires positive integer class labeels in `1:C` where `C` is the number of classes.
`X`	a design `matrix` of predictors; can be a typical (dense) `matrix` or a sparse `Matrix` object. When the design matrix is sparse (and is stored sparsely), this can produce a ~3x-faster execution via a more efficient update for the beta parameter. But when it is not sparse (but is stored sparsely) the execution could be much slower
`N`	an optional integer vector of total numbers of replicate trials for each `X`-`y`, i.e., for Binomial data instead of Bernoulli
`flatten`	a scalar `logical` that is only specified for Binomial data. It indicates if preprocessing code should flatten the Binomial likelihood into a Bernoulli likelihood
`sigma`	weights on the regression coefficients in the lasso penalty. The default of `1` is sensible when `normalize = TRUE` since then the estimator for `beta` is equivariant under rescaling
`nu`	a non-negative scalar indicating the initial value of the penalty parameter

kappa	a positive scalar specifying the multiplicity; kappa = 1 provides samples from the Bayesian posterior distribution. Larger values of kappa facilitates a simulated annealing approach to obtaining a regularized point estimator
icept	a scalar logical indicating if an (implicit) intercept should be included in the model
normalize	a scalar logical which, if TRUE, causes each variable is standardized to have unit L2-norm, otherwise it is left alone
zzero	a scalar logical indicating if the latent z variables to be sampled. Therefore this indicator specifies if the cdf representation (zzero = FALSE) or pdf representation (otherwise) should be used
powerprior	a scalar logical indicating if the prior should be powered up with multiplicity parameter kappa as well as the likelihood
kmax	a positive integer indicating the number replacing infinity in the sum for mixing density in the generative expression for lambda
bstart	an optional vector of length p = ncol(X) specifying initial values for the regression coefficients beta. Otherwise standard normal deviates are used
lt	an optional vector of length n = nrow(X) of initial values for the lambda latent variables. Otherwise a vector of ones is used.
nup	prior parameters =list(a, b) for the inverse Gamma distribution prior for nu, or NULL, which causes nu to be fixed
save.latents	a scalar logical indicating wether or not a trace of latent z, lambda and omega values should be saved for each iteration. Specify save.latents=TRUE for very large X in order to reduce memory swapping on low-RAM machines
verb	A positive integer indicating the number of MCMC rounds after which a progress statement is printed. Giving verb = 0 causes no statements to be printed

Details

These are the main functions in the package. They support an omnibus framework for simulation-based regularized logistic regression. The default arguments invoke a Gibbs sampling algorithm to sample from the posterior distribution of a logistic regression model with lasso-type (double-exponential) priors. See the paper by Gramacy \& Polson (2012) for details. Both cdf and pdf implementations are provided, which use slightly different latent variable representations, resulting in slightly different Gibbs samplers. These methods extend the un-regularized methods of Holmes \& Held (2006)

The kappa parameter facilitates simulated annealing (SA) implementations in order to help find the MAP, and other point estimators. The actual SA algorithm is not provided in the package. However, it is easy to string calls to this function, using the outputs from one call as inputs to another, in order to establish a SA schedule for increasing kappa values.

The regmlogit function is a wrapper around the Gibbs sampler inside reglogit, invoking C-1 linked chains for C classes, extending the polychotomous regression scheme outlined by Holmes \& Held (2006). For an example with regmlogit, see predict.regmlogit

Value

The output is a `list` object of type `"reglogit"` or `"regmlogit"` containing a subset of the following fields; for `"regmlogit"` everyhing is expanded by one dimension into an `array` or `matrix` as appropriate.

`X`	the input design `matrix`, possible adjusted by normalization or intercept
`y`	the input response variable
`beta`	a `matrix` of `T` sampled regression coefficients on the original input scale
`z`	if `zzero = FALSE` a `matrix` of latent variables for the hierarchical cdf representation of the likelihood
`lambda`	a `matrix` of latent variables for the hierarchical (cdf or pdf) representation of the likelihood
`lpost`	a vector of log posterior probabilities of the parameters
`map`	the `list` containing the maximum a' posterior parameters; `out$map$beta` is on the original scale of the data
`kappa`	the input multiplicity parameter
`omega`	a `matrix` of latent variables for the regularization prior

Author(s)

Robert B. Gramacy <rbg@vt.edu>

References

R.B. Gramacy, N.G. Polson. "Simulation-based regularized logistic regression". (2012) Bayesian Analysis, 7(3), p567-590; arXiv:1005.3430; http://arxiv.org/abs/1005.3430

C. Holmes, K. Held (2006). "Bayesian Auxilliary Variable Models for Binary and Multinomial Regression". Bayesian Analysis, 1(1), p145-168.

See Also

`predict.reglogit`, `predict.regmlogit`, `blasso` and `regress`

Examples

```
## load in the pima indian data
data(pima)
X <- as.matrix(pima[,-9])
y <- as.numeric(pima[,9])

## pre-normalize to match the comparison in the paper
one <- rep(1, nrow(X))
normx <- sqrt(drop(one %*% (X^2)))
X <- scale(X, FALSE, normx)

## compare to the GLM fit
fit.logit <- glm(y~X, family=binomial(link="logit"))
```

```
bstart <- fit.logit$coef

## do the Gibbs sampling
T <- 300 ## set low for CRAN checks; increase to >= 1000 for better results
out6 <- reglogit(T, y, X, nu=6, nup=NULL, bstart=bstart, normalize=FALSE)

## plot the posterior distribution of the coefficients
burnin <- (1:(T/10))
boxplot(out6$beta[-burnin,], main="nu=6, kappa=1", ylab="posterior",
        xlab="coefficients", bty="n", names=c("mu", paste("b", 1:8, sep="")))
abline(h=0, lty=2)

## add in GLM fit and MAP with legend
points(bstart, col=2, pch=17)
points(out6$map$beta, pch=19, col=3)
legend("topright", c("MLE", "MAP"), col=2:3, pch=c(17,19))

## simple prediction
p6 <- predict(out6, XX=X)
## hit rate
mean(p6$c == y)

##
## for a polychotomous example, with prediction,
## see ? predict.regmlogit
##

## Not run:
## now with kappa=10
out10 <- reglogit(T, y, X, kappa=10, nu=6, nup=NULL, bstart=bstart,
                           normalize=FALSE)

## plot the posterior distribution of the coefficients
par(mfrow=c(1,2))
boxplot(out6$beta[-burnin,], main="nu=6, kappa=1",  ylab="posterior",
        xlab="coefficients", bty="n",  names=c("mu", paste("b", 1:8, sep="")))
abline(h=0, lty=2)
points(bstart, col=2, pch=17)
points(out6$map$beta, pch=19, col=3)
legend("topright", c("MLE", "MAP"), col=2:3, pch=c(17,19))
boxplot(out10$beta[-burnin,], main="nu=6, kappa=10",  ylab="posterior",
        xlab="coefficients", bty="n",  names=c("mu", paste("b", 1:8, sep="")))
abline(h=0, lty=2)
## add in GLM fit and MAP with legend
points(bstart, col=2, pch=17)
points(out10$map$beta, pch=19, col=3)
legend("topright", c("MLE", "MAP"), col=2:3, pch=c(17,19))

## End(Not run)

##
## now some binomial data
##
```

```
## Not run:
## synthetic data generation
library(boot)
N <- rep(20, 100)
beta <- c(2, -3, 2, -4, 0, 0, 0, 0, 0)
X <- matrix(runif(length(N)*length(beta)), ncol=length(beta))
eta <- drop(1 + X %*% beta)
p <- inv.logit(eta)
y <- rbinom(length(N), N, p)

## run the Gibbs sampler for the logit -- uses the fast Binomial
## version; for a comparison, try flatten=FALSE
out <- reglogit(T, y, X, N)

## plot the posterior distribution of the coefficients
boxplot(out$beta[-burnin,], main="binomial data",  ylab="posterior",
       xlab="coefficients", bty="n",
       names=c("mu", paste("b", 1:ncol(X), sep="")))
abline(h=0, lty=2)

## add in GLM fit, the MAP fit, the truth, and a legend
fit.logit <- glm(y/N~X, family=binomial(link="logit"), weights=N)
points(fit.logit$coef, col=2, pch=17)
points(c(1, beta), col=4, pch=16)
points(out$map$beta, pch=19, col=3)
legend("topright", c("MLE", "MAP", "truth"), col=2:4, pch=c(17,19,16))

## also try specifying a larger kappa value to pin down the MAP

## End(Not run)
```

Index

∗Topic **classif**
predict.reglogit, 4
reglogit, 6
∗Topic **datasets**
pima, 3
∗Topic **methods**
reglogit, 6
∗Topic **models**
predict.reglogit, 4
∗Topic **package**
reglogit-package, 2

blasso, *2*, *8*

Matrix, *6*

pima, 3
predict, *4*
predict.reglogit, 4, *8*
predict.regmlogit, *7, 8*
predict.regmlogit *(*predict.reglogit*)*, 4

reglogit, *2*, *5*, 6
reglogit-package, 2
regmlogit, *5*
regmlogit *(*reglogit*)*, 6
regress, *2*, *8*