Package 'parglm'

August 10, 2020

Type Package

Title Parallel GLM

Version 0.1.6

Description Provides a parallel estimation method for generalized
linear models without compiling with a multithreaded LAPACK or BLAS.

License GPL-2

Encoding UTF-8

LazyData true

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, Matrix

SystemRequirements C++11

Suggests testthat, SuppDists, knitr, rmarkdown, speedglm,
microbenchmark, R.rsp

RoxygenNote 6.1.1

VignetteBuilder R.rsp

NeedsCompilation yes

Author Benjamin Christoffersen [cre, aut],
Anthony Williams [cph],
Boost developers [cph]

Maintainer Benjamin Christoffersen <boennecd@gmail.com>

Repository CRAN

Date/Publication 2020-08-10 17:10:10 UTC

R topics documented:

parglm *Fitting Generalized Linear Models in Parallel*

Description

Function like glm which can make the computation in parallel. The function supports most families listed in family. See "vignette("parglm","parglm")" for run time examples.

Usage

```
parglm(formula, family = gaussian, data, weights, subset, na.action,
  start = NULL, offset, control = list(...), contrasts = NULL,
  model = TRUE, x = FALSE, y = TRUE, ...)

parglm.fit(x, y, weights = rep(1, NROW(x)), start = NULL,
  etastart = NULL, mustart = NULL, offset = rep(0, NROW(x)),
  family = gaussian(), control = list(), intercept = TRUE, ...)
```

Arguments

formula	an object of class formula.
family	a family object.
data	an optional data frame, list or environment containing the variables in the model.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs.
start	starting values for the parameters in the linear predictor.
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting.
control	a list of parameters for controlling the fitting process. For parglm.fit this is passed to parglm.control.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
model	a logical value indicating whether model frame should be included as a component of the returned value.
x, y	For parglm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For parglm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.
...	For parglm: arguments to be used to form the default control argument if it is not supplied directly. For parglm.fit: unused.

etastart	starting values for the linear predictor. Not supported.
mustart	starting values for the vector of means. Not supported.
intercept	logical. Should an intercept be included in the null model?

Details

The current implementation uses min(as.integer(n / p),nthreads) threads where n is the number observations, p is the number of covariates, and nthreads is the nthreads element of the list returned by parglm.control. Thus, there is likely little (if any) reduction in computation time if p is almost equal to n. The current implementation cannot handle p > n.

Value

glm object as returned by glm but differs mainly by the qr element. The qr element in the object returned by parglm(.fit) only has the R matrix from the QR decomposition.

Examples

```
# small example from `help('glm')`. Fitting this model in parallel does
# not matter as the data set is small
clotting <- data.frame(
  u = c(5,10,15,20,30,40,60,80,100),
  lot1 = c(118,58,42,35,27,25,21,19,18),
  lot2 = c(69,35,26,21,18,16,13,12,12))
f1 <- glm   (lot1 ~ log(u), data = clotting, family = Gamma)
f2 <- parglm(lot1 ~ log(u), data = clotting, family = Gamma,
             control = parglm.control(nthreads = 2L))
all.equal(coef(f1), coef(f2))
```

---

parglm.control *Auxiliary for Controlling GLM Fitting in Parallel*

Description

Auxiliary function for parglm fitting.

Usage

```
parglm.control(epsilon = 1e-08, maxit = 25, trace = FALSE,
  nthreads = 1L, block_size = NULL, method = "LINPACK")
```

Arguments

epsilon	positive convergence tolerance.
maxit	integer giving the maximal number of IWLS iterations.
trace	logical indicating if output should be produced doing estimation.

nthreads	number of cores to use. You may get the best performance by using your number of physical cores if your data set is sufficiently large. Using the number of physical CPUs/cores may yield the best performance (check your number e.g., by calling `parallel::detectCores(logical = FALSE)`).
block_size	number of observation to include in each parallel block.
method	string specifying which method to use. Either `"LINPACK"`, `"LAPACK"`, or `"FAST"`.

Details

The `LINPACK` method uses the same QR method as `glm.fit` for the final QR decomposition. This is the `dqrdc2` method described in `qr`. All other QR decompositions but the last are made with `DGEQP3` from `LAPACK`. See Wood, Goude, and Shaw (2015) for details on the QR method.

The `FAST` method computes the Fisher information and then solves the normal equation. This is faster but less numerically stable.

Value

A list with components named as the arguments.

References

Wood, S.N., Goude, Y. & Shaw S. (2015) Generalized additive models for large datasets. Journal of the Royal Statistical Society, Series C 64(1): 139-155.

Examples

```
# use one core
clotting <- data.frame(
 u = c(5,10,15,20,30,40,60,80,100),
 lot1 = c(118,58,42,35,27,25,21,19,18),
 lot2 = c(69,35,26,21,18,16,13,12,12))
f1 <- parglm(lot1 ~ log(u), data = clotting, family = Gamma,
             control = parglm.control(nthreads = 1L))

# use two cores
f2 <- parglm(lot1 ~ log(u), data = clotting, family = Gamma,
             control = parglm.control(nthreads = 2L))
all.equal(coef(f1), coef(f2))
```

Index