# Package 'HPDGLM'

August 1, 2014

**Type** Package

**Title** Distributed Regression for Big Data

**Version** 0.7.0

**Date** 2014-01-22

**Author** Arash Fard

**Maintainer** Arash Fard <afard@vertica.com>

**Depends** R (>= 3.0.1), distributedR

**Description** It provides distributed Generalized Linear Model. It is written based on the infrastructure created in HP-Lab for distributed computing in R.

**License** GPL (>= 2) | file LICENSE

## R topics documented:

## HPDGLM-package *HPDGLM - Distributed Regression for Big Data*

### Description

**HPDGLM** provides distributed Generalized Linear Model. It is written based on the infrastructure created in HP-Lab for distributed computing in R.

### Details

| | |
|---|---|
| Package: | HPDGLM |
| Type: | Package |
| Version: | 0.7.0 |
| Date: | 2014-01-22 |

Main Functions:

- hpdglm: It is a distributed version of glm.
- v.hpdglm: This function is implemented for evaluating a model built by hpdglm using Split-Sample-Validation method.
- cv.hpdglm: This function is implemented for evaluating a model built by hpdglm using Cross-Validation method.

### Author(s)

Arash Fard <afard@vertica.com>

## cv.hpdglm *Cross-Validation Method for hpdglm Models*

### Description

This function is implemented for evaluating a model built by hpdglm using Cross-Validation method. In simple words, the data is randomly divided into K folds, and building model and testing that is repeated K times. Every iteration, one fold is reserved as test data and the rest is used for training. Finally, the prediction costs of the new models on all folds are aggregated.

### Usage

```
cv.hpdglm(responses, predictors, hpdglmfit, cost = hpdCost,
              K = 10, sampling_threshold = 1e+06)
```

## Arguments

| | |
|---|---|
| responses | the darray that contains the vector of responses. |
| predictors | the darray that contains the vector of predictors. |
| hpdglmfit | a built model of type hpdglm. |
| cost | an optional cost function for validation. |
| K | number of folds in cross validation. |
| sampling_threshold | threshold for the method of sampling (centralized or distributed). It should be always smaller than 1e9. When (blockSize > sampling_threshold \|\| nSample > 1e9 \|\| nSample/K > sampling_threshold), the distributed sampling is selected in which each block is divided to K folds. Here, blockSize is the number of samples in each partition of predictors, and nSample is the total number of samples in predictors. |

## Details

In order to randomly select the validation set, sample.int function of R is used. This function does not support sample space bigger than 1e9; moreover, it is slow for big numbers. Therefore, when the number of samples in each block of darray becomes bigger than sampling_threshold, instead of purely random selection on the master side, each block will randomly contribute its portion to validation set in a distributed fashion. When the ratio of number of samples in each block to the total number of blocks is huge, the skew of randomness is negligible, but performance will be improved.

## Value

| | |
|---|---|
| call | the original call to cv.hpdglm |
| K | number of folds used at the input |
| delta | a vector of length two. The first component is the raw validation estimate of prediction error. The second component is the adjusted validation estimate. Lower cost indicates better fitting. In more detail, the 1st-cost is prediction cost of new model on test data, and the 2nd-cost is [1st-cost + (cost of the old model on all data - the cost of the new model on all data)] |
| seed | the value of .Random.seed when cv.hpdglm was called. |

## Author(s)

Arash Fard

## Examples

```
 ## Not run:
    require(distributedR)
    distributedR_start()
    Y <- as.darray(as.matrix(faithful$eruptions),
                   c(ceiling(length(faithful$eruption)/4),1))
    X <- as.darray(as.matrix(faithful$waiting),
                   c(ceiling(length(faithful$waiting)/4),1))

    myModel <- hpdglm(Y, X)
    testCV <- cv.hpdglm(Y, X, myModel)

## End(Not run)
```

## d.pow *Power Function for darray*

### Description

It makes the elements of a darray to a power

### Usage

```
d.pow(X, power)
```

### Arguments

| | |
|---|---|
| X | a darray |
| power | a numerical value |

### Value

the input darray which has its elements have raised to a power.

### Author(s)

Arash Fard

### Examples

```
 ## Not run:
    require(distributedR)
    distributedR_start()
    Y <- as.darray(as.matrix(faithful$eruptions),
                   c(ceiling(length(faithful$eruption)/4),1))
    Y <- d.pow(Y, 2)

## End(Not run)
```

## family.hpdglm *family of an hpdglm model*

### Description

Returns the family used for building an hpdglm model.

### Usage

```
family.hpdglm(object, ...)
```

### Arguments

| | |
|---|---|
| object | an hpdglm model |

### Value

the family of the model

## hpdCost *Prediction Cost*

### Description

The defualt distributed cost function for v.hpdglm and cv.hpdglm

### Usage

```
hpdCost(y, yhat, mask = NULL)
```

### Arguments

| | |
|---|---|
| y | darray of the real response |
| yhat | darray of the predicted response |
| mask | mask indicates which samples (rows) should be considered in the calculation. It should be a darray with a single column, and 0 or 1 as the value of its elements. |

### Details

equivalent sequential: mean((y[mask==1,]-yhat[mask==1,])^2)

### Value

a real value corresponding to the cost

### Author(s)

Arash Fard

## hpdglm *Distributed Generalized Linear Models*

### Description

hpdglm function is intended to be a distributed alternative for glm function.

### Usage

```
hpdglm(responses, predictors, family=gaussian, weights=NULL,
na_action="exclude", start=NULL, etastart=NULL, mustart=NULL,
offset=NULL, control=list(...), method="hpdglm.fit.Newton", ...)
```

## Arguments

| | |
|---|---|
| `responses` | the darray that contains the vector of responses. |
| `predictors` | the darray that contains the vector of predictors. hpdglm() cannot accept a predictor with constant value. Moreover, a categorical predictor should be decoded (converted to several predictors) before applying hpdglm(). |
| `family` | it specifies the family function for regression. The supported family-links at the time of this writing are gaussian(identity), binomial(logit), and poisson(log). The mentioned links are the default ones for their families; so, specifying them is optional. The default family is Gaussian. |
| `weights` | it is an optional darray of 'prior weights' to be used in the fitting process. It has a single column. The number of rows and its number of blocks should be the same as responses. The values should not be negative (greater than or equal to zero). Weight zero on a sample makes it be ignored. |
| `na_action` | it indicates what should happen when the data contain missed values. Values of NA, NaN, and Inf in samples are treated as missed values. There are two options for this argument exclude and fail. When exclude is selected (the default choice), the weight of any sample with missed values will become zero, and that sample will be ignored in the fitting process. In the darray which will be created for residuals, the value corresponding to these samples will be NA. When fail is selected, the function will stop in the case of any missed value in the dataset. |
| `start` | starting values for coefficients. It is optional. |
| `etastart` | starting values for parameter 'eta' which is used for computing deviance. It should be of type darray. It is optional. |
| `mustart` | starting values for mu 'parameter' which is used for computing deviance. It should be of type darray. It is optional. |
| `offset` | an optional darray which can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. |
| `control` | an optional list of controlling arguments. The optional elements of the list and their default values are: epsilon = 1e-8, maxit = 25, trace = FALSE, rigorous = FALSE. |
| `method` | this argument reserved for the future improvement. The only available fitting method at the moment is "hpdglm.fit.Newton". In the future, if we have new developed algorithms, this argument can be used to switch between them. |
| `...` | arguments to be used to form the default `control` argument if it is not supplied directly. |

## Details

predictors and responses must align with each other (have the same number of rows and the same number of blocks).

## Value

| | |
|---|---|
| `coefficients` | calculated coefficients |
| `d.residuals` | the working residuals, that is the residuals in the final iteration of the IWLS fit. Since cases with zero weights are omitted, their working residuals are NA. It is of type darray. |

`d.fitted.values`
: the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function. It is of type darray.

`family`
: the family function used for regression

`d.linear.predictors`
: the linear fit on link scale. It is of type darray.

`deviance`
: up to a constant, minus twice the maximized log-likelihood.

`aic`
: a version of Akaike's An Information Criterion, minus twice the maximized log-likelihood plus twice the number of parameters, computed by the aic component of the family.

`null.deviance`
: the deviance for the null model, comparable with deviance.

`iter`
: the number of iterations of IWLS used.

`prior.weights`
: the weights initially supplied. All of its values are 1 if no initial weights used. It is of type darray. The value of weight will become 0 for the samples with invalid data (NA, NaN, Inf).

`weights`
: the working weights, that is the weights in the final iteration of the IWLS fit. It is of type darray. In order to save memory and execution time, no new darray will be created for weights when the initial weights are all 0 or 1, and it will simply be a reference to prior.weights.

`df.residual`
: the residual degrees of freedom.

`df.null`
: the residual degrees of freedom for the null model.

`converged`
: logical. Was the IWLS algorithm judged to have converged?

`boundary`
: logical. Is the fitted value on the boundary of the attainable values?

`responses`
: the darray of responses.

`predictors`
: the darray of predictors.

`na_action`
: this item exists only when a few samples are excluded because of missed data. It is a list containing type "exclude" and the number of excluded samples.

`call`
: the matched call.

`offset`
: the offset darray used.

`control`
: the value of the control argument used.

`method`
: the name of the fitter function used, currently always "hpdglm.fit.Newton".

## Author(s)

Arash Fard

## Examples

```
## Not run:
   ## Example for logistic regression
   require(distributedR)
   distributedR_start()
   # selecting 4 partitions
   Y <- as.darray(as.matrix(mtcars$am),c(ceiling(length(mtcars$am)/4),1))
   colnames(Y) <- "am"
   X <- as.darray(as.matrix(cbind(mtcars$wt,mtcars$hp)),
                  c(ceiling(length(mtcars$hp)/4),2))
```

```
colnames(X) <- c("wt", "hp")
# building logistic regression model
myModel <- hpdglm(Y, X, binomial)
summary(myModel)

## Example for linear regression
require(distributedR)
distributedR_start()
require(MASS)
y <- matrix(Boston$medv,ncol=1)
n <- nrow(y)
x <- matrix(c(Boston$rad,Boston$crim,Boston$ptratio,Boston$dis),ncol=4)
# selecting the number of partitions the same as the number of R-executors
nparts <- sum(distributedR_status()$Inst)
block_row <- ceiling(n/nparts)
# creating the darray of response
Y <- as.darray(y, c(block_row,1))
colnames(Y) <- c("medv")
# creating the darray of predictors
X <- as.darray(x, c(block_row,4))
colnames(X) <- c("rad","crim","ptratio","dis")
# building linear regression model
reg <- hpdglm(Y,X)
summary(reg)

## End(Not run)
```

---

`hpdglm.control` *Auxiliary for Controlling hpdglm Fitting*

---

## Description

Auxiliary function for `hpdglm` fitting. Typically only used internally by `hpdglm.fit`, but may be used to construct a `control` argument to either function.

## Usage

```
hpdglm.control(epsilon = 1e-08, maxit = 25, trace = FALSE,
               rigorous = FALSE)
```

## Arguments

| | |
|---|---|
| `epsilon` | It is used to adjust desired accuracy of the result. |
| `maxit` | It is the maximu, number of iterations before achieving the desired accuracy. |
| `trace` | When this argument is true, intermediate steps of the progress are displayed. |
| `rigorous` | When this argument is true, some extra checks are performed during fitting procedure. For example, mu and eta may be validating in each iteration to check if the fitted values are outside of the domain. Usually these checks are time consuming; therefore, the default value for this argument is FALSE. |

## Value

A list with components named as the arguments.

### Examples

```
## Not run:
   require(distributedR)
   distributedR_start()
   Y <- as.darray(as.matrix(mtcars$am),c(ceiling(length(mtcars$am)/4),1))
   X <- as.darray(as.matrix(cbind(mtcars$wt,mtcars$hp)),
                  c(ceiling(length(mtcars$hp)/4),2))

   myModel <- hpdglm(Y, X, binomial, control=list(epsilon=1e-02, maxit=5,
                     trace=FALSE, rigorous=TRUE)

## End(Not run)
```

---

## predict.hpdglm *Predict Method for HPDGLM Fits*

---

### Description

It produces predicted values, obtained by evaluating the regression function on provided new data.

### Usage

```
predict.hpdglm(object, newdata, type = c("link", "response"),
               na.action = na.pass, mask = NULL, trace = TRUE, ...)
```

### Arguments

| | |
|---|---|
| object | a built model of type hpdglm. |
| newdata | a matrix or a darray containing predictors of new samples. |
| type | the type of prediction required which can be "link" or "response". |
| na.action | a function to determine what should be done with missing values. At this version it is always na.pass (reserved for future improvement). |
| mask | a darray with a single column, and 0 or 1 as the value of its elements. It indicates which samples (rows) should be considered in the calculation. |
| trace | when this argument is true, intermediate steps of the progress are displayed. |
| ... | further arguments passed to or from other methods. |

### Details

This function produces predicted values, obtained by evaluating the regression function on provided new data. New data can be either a darray or a normal matrix.

### Value

The output is a matrix or a darray, depending to the type of newdata, which contains predicted values for response.

### Author(s)

Arash Fard

### Examples

```
## Not run:
   require(distributedR)
   distributedR_start()
   Y <- as.darray(as.matrix(faithful$eruptions),
                  c(ceiling(length(faithful$eruption)/4),1))
   X <- as.darray (as.matrix(faithful$waiting),
                  c(ceiling(length(faithful$waiting)/4),1))

   myModel <- hpdglm(Y, X)
   newSamples <- matrix(c(1:3),,1)
   predict(myModel, newSamples, "link")

## End(Not run)
```

---

## residuals.hpdglm *Extract Residuals of an hpdglm Model*

---

### Description

This function extracts model residuals of an hpdglm model in a darray. The abbreviated function is resid.

### Usage

```
residuals.hpdglm(object, type = c("deviance", "pearson",
                 "working", "response", "partial"), ...)
```

### Arguments

| | |
|---|---|
| object | an hpdglm model |
| type | can be "deviance", "pearson", "working", "response", or "partial". |

### Value

darray of residuals

### Examples

```
## Not run:
   require(distributedR)
   distributedR_start()
   Y <- as.darray(as.matrix(mtcars$am),
                  c(ceiling(length(mtcars$am)/4),1))
   X <- as.darray(as.matrix(cbind(mtcars$wt,mtcars$hp)),
                  c(ceiling(length(mtcars$hp)/4),2))

   myModel <- hpdglm(responses=Y, predictors=X, family= binomial(logit))
   res <- resid(myModel)

## End(Not run)
```

summary.hpdglm *Summarizing The Model Made by hpdglm*

## Description

This function prints a summary of the learned model.

## Usage

```
summary.hpdglm(object, dispersion = NULL, correlation = FALSE,
               symbolic.cor = FALSE, ...)
```

## Arguments

| | |
|---|---|
| object | a result of a call to hpdglm. |
| dispersion | the dispersion parameter for the family used. Either a single numerical value or NULL (the default), when it is inferred from object). |
| correlation | logical; if TRUE, the correlation matrix of the estimated parameters is returned and printed. |
| symbolic.cor | logical. If TRUE, print the correlations in a symbolic form (see symnum) rather than as numbers. |

## Value

summary.hpdglm returns an object of class "summary.hpdglm", a list with components

| | |
|---|---|
| call | the component from object. |
| family | the component from object. |
| deviance | the component from object. |
| df.residual | the component from object. |
| null.deviance | the component from object. |
| df.null | the component from object. |
| deviance.resid | the deviance residuals: see residuals.hpdglm. |
| coefficients | the matrix of coefficients, standard errors, z-values and p-values. Aliased coefficients are omitted. |
| aliased | named logical vector showing if the original coefficients are aliased. |
| dispersion | either the supplied argument or the inferred/estimated dispersion if the latter is NULL. |
| df | a 3-vector of the rank of the model and the number of residual degrees of freedom, plus number of non-aliased coefficients. |
| cov.unscaled | the unscaled (dispersion = 1) estimated covariance matrix of the estimated coefficients. |
| cov.scaled | ditto, scaled by dispersion. |
| correlation | (only if correlation is true.) The estimated correlations of the estimated coefficients. |
| symbolic.cor | (only if correlation is true.) The value of the argument symbolic.cor. |

### Author(s)

Arash Fard

### Examples

```
## Not run:
   require(distributedR)
   distributedR_start()
   Y <- as.dobject(as.matrix(faithful$eruptions),
                   c(ceiling(length(faithful$eruption)/4),1))
   X <- as. dobject (as.matrix(faithful$waiting),
                   c(ceiling(length(faithful$waiting)/4),1))

   myModel <- hpdglm(responses=Y, predictors=X)
   summary(myModel)

## End(Not run)
```

---

## v.hpdglm *Split-Sample-Validation Method for hpdglm Models*

---

### Description

This function is implemented for evaluating a model built by hpdglm using Split-Sample-Validation method. In simple words, a percent of the data is randomly selected as test data, and a new model is built using the rest. Finally, the prediction cost of the new model on the test data is measured.

### Usage

```
v.hpdglm(responses, predictors, hpdglmfit, cost = hpdCost,
         percent = 30, sampling_threshold = 1e+06)
```

### Arguments

| | |
|---|---|
| `responses` | the darray that contains the vector of responses. |
| `predictors` | the darray that contains the vector of predictors. |
| `hpdglmfit` | a built model of type hpdglm. |
| `cost` | an optional cost function for validation. |
| `percent` | the percent of data which should be set aside for validation. |
| `sampling_threshold` | threshold for the method of sampling (centralized or distributed). It should be always smaller than 1e9. When (blockSize > sampling_threshold ‖ nSample > 1e9), the distributed sampling is selected in which a percent of each block is selected for test data. Here, blockSize is the number of samples in each partition of predictors, and nSample is the total number of samples in predictors. |

## Details

In order to randomly select the validation set, sample.int function of R is used. This function does not support sample space bigger than 1e9; moreover, it is slow for big numbers. Therefore, when the number of samples in each block of darray becomes bigger than sampling_threshold, instead of purely random selection on the master side, each block will randomly contribute its portion to validation set in a distributed fashion. When the ratio of number of samples in each block to the total number of blocks is huge, the skew of randomness is negligible, but performance will be improved.

## Value

| | |
|---|---|
| `call` | the original call to v.hpdglm |
| `percent` | the percent value used at the input |
| `delta` | a vector of length two. The first component is the raw validation estimate of prediction error. The second component is the adjusted validation estimate. Lower cost indicates better fitting. In more detail, the 1st-cost is prediction cost of new model on test data, and the 2nd-cost is [1st-cost + (cost of the old model on all data - the cost of the new model on all data)] |
| `seed` | the value of .Random.seed when v.hpdglm was called. |

## Author(s)

Arash Fard

## Examples

```
 ## Not run:
    require(HPDGLM)
    distributedR_start()
    Y <- as.darray(as.matrix(faithful$eruptions),
                   c(ceiling(length(faithful$eruption)/4),1))
    X <- as.darray(as.matrix(faithful$waiting),
                   c(ceiling(length(faithful$waiting)/4),1))

    myModel <- hpdglm(Y, X)
    testV <- v.hpdglm(Y, X, myModel)

## End(Not run)
```

# Index