Package: fastLogisticRegressionWrap (via r-universe)

October 8, 2024

Type Package

Title Fast Logistic Regression Wrapper

Date 2023-xx-xx

Version 1.2.1

License GPL-3

Description Provides very fast logistic regression with coefficient inferences plus other useful methods such as a forward stepwise model generator (see the benchmarks by visiting the github page at the URL below). The inputs are flexible enough to accomodate GPU computations. The coefficient estimation employs the fastLR() method in the 'RcppNumerical' package by Yixuan Qiu et al. This package allows their work to be more useful to a wider community that consumes inference.

Encoding UTF-8

Depends R (>= 4.0.0)

Imports RcppNumerical, Rcpp, checkmate, stats, MASS, methods

LinkingTo Rcpp, RcppEigen

URL https://github.com/kapelner/fastLogisticRegressionWrap

BugReports https://github.com/kapelner/fastLogisticRegressionWrap/issues

RoxygenNote 7.2.3

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

RemoteUrl https://github.com/kapelner/fastlogisticregressionwrap

RemoteRef HEAD

RemoteSha 082e350cc0f870fa44b19c0fea748b4f52d894fa

Contents

`asymmetric_cost_explorer` *Asymmetric Cost Explorer*

Description

Given a set of desired proportions of predicted outcomes, what is the error rate for each of those models?

Usage

```
asymmetric_cost_explorer(
  phat,
  ybin,
  steps = seq(from = 0.001, to = 0.999, by = 0.001),
  outcome_of_analysis = 0,
  proportions_desired = seq(from = 0.1, to = 0.9, by = 0.1),
  proportion_tolerance = 0.01
)
```

Arguments

`phat`	The vector of probability estimates to be thresholded to make a binary decision
`ybin`	The true binary responses
`steps`	All possibile thresholds which must be a vector of numbers in (0, 1). Default is `seq(from = 0.001, to = 0.999, by = 0.001)`.

`outcome_of_analysis`
: Which class do you care about performance? Either 0 or 1 for the negative class or positive class. Default is `0`.

`proportions_desired`
: Which proportions of `outcome_of_analysis` class do you wish to understand performance for?

`proportion_tolerance`
: If the model cannot match the proportion_desired within this amount, it does not return that model's performance. Default is `0.01`.

`K_folds`
: If not `NULL`, this indicates that we wish to fit the `phat` thresholds out of sample using this number of folds. Default is `NULL` for in-sample fitting.

Value

A table with column 1: `proportions_desired`, column 2: actual proportions (as close as possible), column 3: error rate, column 4: probability threshold.

Author(s)

Adam Kapelner

`asymmetric_cost_explorer_cross_validated` *Asymmetric Cost Explorer*

Description

Given a set of desired proportions of predicted outcomes, what is the error rate for each of those models?

Usage

```
asymmetric_cost_explorer_cross_validated(phat, ybin, K_CV = 5, ...)
```

Arguments

`phat`
: The vector of probability estimates to be thresholded to make a binary decision

`ybin`
: The true binary responses

`K_CV`
: We wish to fit the `phat` thresholds out of sample using this number of folds. Default is `5`.

`...`
: Other parameters to be passed into the `asymmetric_cost_explorer` function

Value

A table with column 1: `proportions_desired`, column 2: actual proportions (as close as possible), column 3: error rate, column 4: probability threshold.

Author(s)

Adam Kapelner

`confusion_results` *Binary Confusion Table and Errors*

Description

Provides a binary confusion table and error metrics

Usage

```
confusion_results(yhat, ybin, skip_argument_checks = FALSE)
```

Arguments

`yhat`	The binary predictions
`ybin`	The true binary responses
`skip_argument_checks`	If `TRUE` it does not check this function's arguments for appropriateness. It is not recommended unless you truly need speed and thus the default is `FALSE`.

Value

A list of raw results

Examples

```
library(MASS); data(Pima.te)
ybin = as.numeric(Pima.te$type == "Yes")
flr = fast_logistic_regression(
  Xmm = model.matrix(~ . - type, Pima.te),
  ybin = ybin
)
phat = predict(flr, model.matrix(~ . - type, Pima.te))
confusion_results(phat > 0.5, ybin)
```

`eigen_compute_single_entry_of_diagonal_matrix` *Compute Single Value of the Diagonal of a Symmetric Matrix's Inverse*

Description

Via the eigen package's conjugate gradient descent algorithm.

Usage

```
eigen_compute_single_entry_of_diagonal_matrix(M, j, num_cores = 1)
```

Arguments

`M`	The symmetric matrix which to invert (and then extract one element of its diagonal)
`j`	The diagonal entry of `M`'s inverse
`num_cores`	The number of cores to use. Default is 1.

Value

The value of m^-1_j,j

Author(s)

Adam Kapelner

Examples

```
n = 500
X = matrix(rnorm(n^2), nrow = n, ncol = n)
M = t(X) %*% X
j = 137
eigen_compute_single_entry_of_diagonal_matrix(M, j)
solve(M)[j, j] #to ensure it's the same value
```

`eigen_det` *A fast det(X) function*

Description

Via the eigen package

Usage

```
eigen_det(X, num_cores = 1)
```

Arguments

X	A numeric matrix of size p x p
num_cores	The number of cores to use. Unless p is large, keep to the default of 1.

Value

The determinant as a scalar numeric value

Examples

```
p = 30
eigen_det(matrix(rnorm(p^2), nrow = p))
```

---

eigen_inv *A fast solve(X) function*

Description

Via the eigen package

Usage

```
eigen_inv(X, num_cores = 1)
```

Arguments

X	A numeric matrix of size p x p
num_cores	The number of cores to use. Unless p is large, keep to the default of 1.

Value

The resulting matrix

Examples

```
p = 10
eigen_inv(matrix(rnorm(p^2), nrow = p))
```

eigen_Xt_times_diag_w_times_X

A fast Xt [times] diag(w) [times] X function

Description

Via the eigen package

Usage

```
eigen_Xt_times_diag_w_times_X(X, w, num_cores = 1)
```

Arguments

X	A numeric matrix of size n x p
w	A numeric vector of length p
num_cores	The number of cores to use. Unless p is large, keep to the default of 1.

Value

The resulting matrix

Examples

```
n = 100
p = 10
X = matrix(rnorm(n * p), nrow = n, ncol = p)
w = rnorm(p)
eigen_Xt_times_diag_w_times_X(t(X), w)
```

fastLogisticRegressionWrap

A Wrapper for FastLR

Description

A tool to find many types of a priori experimental designs

Author(s)

Adam Kapelner <kapelner@qc.cuny.edu>

References

Kapelner, A

`fast_logistic_regression` *FastLR Wrapper*

Description

Returns most of what you get from glm

Usage

```
fast_logistic_regression(
  Xmm,
  ybin,
  drop_collinear_variables = FALSE,
  lm_fit_tol = 1e-07,
  do_inference_on_var = "none",
  Xt_times_diag_w_times_X_fun = NULL,
  sqrt_diag_matrix_inverse_fun = NULL,
  num_cores = 1,
  ...
)
```

Arguments

`Xmm`	The model.matrix for X (you need to create this yourself before)
`ybin`	The binary response vector
`drop_collinear_variables`	Should we drop perfectly collinear variables? Default is `FALSE` to inform the user of the problem.
`lm_fit_tol`	When `drop_collinear_variables = TRUE`, this is the tolerance to detect collinearity among predictors. We use the default value from `base::lm.fit`'s which is 1e-7. If you fit the logistic regression and still get p-values near 1 indicating high collinearity, we recommend making this value smaller.
`do_inference_on_var`	Which variables should we compute approximate standard errors of the coefficients and approximate p-values for the test of no linear log-odds probability effect? Default is `"none"` for inference on none (for speed). If not default, then `"all"` to indicate inference should be computed for all variables. The final option is to pass one index to indicate the column number of `Xmm` where inference is desired. We have a special routine to compute inference for one variable only. It consists of a conjugate gradient descent which is another approximation atop the coefficient-fitting approximation in RcppNumerical. Note: if you are just comparing nested models using anova, there is no need to compute inference for coefficients (keep the default of `FALSE` for speed).

`Xt_times_diag_w_times_X_fun`
: A custom function whose arguments are `X` (an n x m matrix), `w` (a vector of length m) and this function's `num_cores` argument in that order. The function must return an m x m R matrix class object which is the result of the computing X^T function is not parallelized, the `num_cores` argument is ignored. Default is `NULL` which uses the function `eigen_Xt_times_diag_w_times_X` which is implemented with the Eigen C++ package and hence very fast. The only way we know of to beat the default is to use a method that employs GPUs. See README on github for more information.

`sqrt_diag_matrix_inverse_fun`
: A custom function that returns a numeric vector which is square root of the diagonal of the inverse of the inputted matrix. Its arguments are `X` (an n x n matrix) and this function's `num_cores` argument in that order. If your custom function is not parallelized, the `num_cores` argument is ignored. The object returned must further have a defined function `diag` which returns the diagonal of the matrix as a vector. Default is `NULL` which uses the function `eigen_inv` which is implemented with the Eigen C++ package and hence very fast. The only way we know of to beat the default is to use a method that employs GPUs. See README on github for more information.

`num_cores`
: Number of cores to use to speed up matrix multiplication and matrix inversion (used only during inference computation). Default is 1. Unless the number of variables, i.e. `ncol(Xmm)`, is large, there does not seem to be a performance gain in using multiple cores.

`...`
: Other arguments to be passed to `fastLR`. See documentation there.

Value

A list of raw results

Examples

```
library(MASS); data(Pima.te)
flr = fast_logistic_regression(
  Xmm = model.matrix(~ . - type, Pima.te),
  ybin = as.numeric(Pima.te$type == "Yes")
)
```

---

fast_logistic_regression_stepwise_forward

Rapid Forward Stepwise Logistic Regression

Description

Roughly duplicates the following `glm`-style code:

Usage

```
fast_logistic_regression_stepwise_forward(
  Xmm,
  ybin,
  mode = "aic",
  pval_threshold = 0.05,
  use_intercept = TRUE,
  verbose = TRUE,
  drop_collinear_variables = FALSE,
  lm_fit_tol = 1e-07,
  ...
)
```

Arguments

Xmm	The model.matrix for X (you need to create this yourself before).
ybin	The binary response vector.
mode	"aic" (default, fast) or "pval" (slow, but possibly yields a better model).
pval_threshold	The significance threshold to include a new variable. Default is 0.05. If mode == "aic", this argument is ignored.
use_intercept	Should we automatically begin with an intercept? Default is TRUE.
verbose	Print out messages during the loop? Default is TRUE.
drop_collinear_variables	Parameter used in fast_logistic_regression. Default is FALSE. See documentation there.
lm_fit_tol	Parameter used in fast_logistic_regression. Default is 1e-7. See documentation there.
...	Other arguments to be passed to fastLR. See documentation there.

Details

nullmod = glm(ybin ~ 0, data.frame(Xmm), family = binomial) fullmod = glm(ybin ~ 0 + ., data.frame(Xmm), family = binomial) forwards = step(nullmod, scope = list(lower = formula(nullmod), upper = formula(fullmod)), direction = "forward", trace = 0)

Value

A list of raw results

Examples

```
library(MASS); data(Pima.te)
flr = fast_logistic_regression_stepwise_forward(
  Xmm = model.matrix(~ . - type, Pima.te),
  ybin = as.numeric(Pima.te$type == "Yes")
)
```

general_confusion_results *General Confusion Table and Errors*

Description

Provides a confusion table and error metrics for general factor vectors. There is no need for the same levels in the two vectors.

Usage

```
general_confusion_results(yhat, yfac, proportions_scaled_by_column = FALSE)
```

Arguments

yhat	The factor predictions
yfac	The true factor responses
proportions_scaled_by_column	When returning the proportion table, scale by column? Default is FALSE to keep the probabilities unconditional to provide the same values as the function confusion_results. Set to TRUE to understand error probabilities by prediction bucket.

Value

A list of raw results

Examples

```
library(MASS); data(Pima.te)
ybin = as.numeric(Pima.te$type == "Yes")
flr = fast_logistic_regression(
  Xmm = model.matrix(~ . - type, Pima.te),
  ybin = ybin
)
phat = predict(flr, model.matrix(~ . - type, Pima.te))
yhat = array(NA, length(ybin))
yhat[phat <= 1/3] = "no"
yhat[phat >= 2/3] = "yes"
yhat[is.na(yhat)] = "maybe"
general_confusion_results(factor(yhat, levels = c("no", "yes", "maybe")), factor(ybin))
#you want the "no" to align with 0, the "yes" to align with 1 and the "maybe" to be
#last to align with nothing
```

predict.fast_logistic_regression
FastLR Wrapper Predictions

Description

Predicts returning p-hats

Usage

```
## S3 method for class 'fast_logistic_regression'
predict(object, newdata, type = "response", ...)
```

Arguments

object	The object built using the fast_logistic_regression or fast_logistic_regression_stepwise wrapper functions
newdata	A matrix of observations where you wish to predict the binary response.
type	The type of prediction required. The default is "response" which is on the response scale (i.e. probability estimates) and the alternative is "link" which is the linear scale (i.e. log-odds).
...	Further arguments passed to or from other methods

Value

A numeric vector of length nrow(newdata) of estimates of P(Y = 1) for each unit in newdata.

Examples

```
library(MASS); data(Pima.te)
flr = fast_logistic_regression(
  Xmm = model.matrix(~ . - type, Pima.te),
  ybin = as.numeric(Pima.te$type == "Yes")
)
phat = predict(flr, model.matrix(~ . - type, Pima.te))
```

predict.fast_logistic_regression_stepwise
FastLR Wrapper Predictions

Description

Predicts returning p-hats

Usage

```
## S3 method for class 'fast_logistic_regression_stepwise'
predict(object, newdata, type = "response", ...)
```

Arguments

object	The object built using the fast_logistic_regression or fast_logistic_regression_stepwise wrapper functions
newdata	A matrix of observations where you wish to predict the binary response.
type	The type of prediction required. The default is "response" which is on the response scale (i.e. probability estimates) and the alternative is "link" which is the linear scale (i.e. log-odds).
...	Further arguments passed to or from other methods

Value

A numeric vector of length nrow(newdata) of estimates of P(Y = 1) for each unit in newdata.

Examples

```
library(MASS); data(Pima.te)
flr = fast_logistic_regression_stepwise_forward(
  Xmm = model.matrix(~ . - type, Pima.te),
  ybin = as.numeric(Pima.te$type == "Yes")
)
phat = predict(flr, model.matrix(~ . - type, Pima.te))
```

---

print.fast_logistic_regression *FastLR Wrapper Print*

Description

Returns the summary table a la glm

Usage

```
## S3 method for class 'fast_logistic_regression'
print(x, ...)
```

Arguments

x	The object built using the fast_logistic_regression or fast_logistic_regression_stepwise wrapper functions
...	Other arguments to be passed to print

Value

The summary as a data.frame

Examples

```
library(MASS); data(Pima.te)
flr = fast_logistic_regression(
 Xmm = model.matrix(~ . - type, Pima.te),
 ybin = as.numeric(Pima.te$type == "Yes"))
print(flr)
```

---

print.fast_logistic_regression_stepwise

FastLR Wrapper Print

Description

Returns the summary table a la glm

Usage

```
## S3 method for class 'fast_logistic_regression_stepwise'
print(x, ...)
```

Arguments

x	The object built using the fast_logistic_regression or fast_logistic_regression_stepwise wrapper functions
...	Other arguments to be passed to print

Value

The summary as a data.frame

Examples

```
library(MASS); data(Pima.te)
flr = fast_logistic_regression_stepwise_forward(
 Xmm = model.matrix(~ . - type, Pima.te),
 ybin = as.numeric(Pima.te$type == "Yes"))
print(flr)
```

summary.fast_logistic_regression *FastLR Wrapper Summary*

Description

Returns the summary table a la glm

Usage

```
## S3 method for class 'fast_logistic_regression'
summary(object, alpha_order = TRUE, ...)
```

Arguments

object	The object built using the fast_logistic_regression or fast_logistic_regression_stepwise wrapper functions
alpha_order	Should the coefficients be ordered in alphabetical order? Default is TRUE.
...	Other arguments to be passed to summary.

Value

The summary as a data.frame

Examples

```
library(MASS); data(Pima.te)
flr = fast_logistic_regression(
 Xmm = model.matrix(~ . - type, Pima.te),
 ybin = as.numeric(Pima.te$type == "Yes"))
summary(flr)
```

summary.fast_logistic_regression_stepwise *FastLR Wrapper Summary*

Description

Returns the summary table a la glm

Usage

```
## S3 method for class 'fast_logistic_regression_stepwise'
summary(object, ...)
```

Arguments

object	The object built using the fast_logistic_regression_stepwise wrapper functions
...	Other arguments to be passed to summary.

Value

The summary as a data.frame

Examples

```
library(MASS); data(Pima.te)
flr = fast_logistic_regression_stepwise_forward(
 Xmm = model.matrix(~ . - type, Pima.te),
 ybin = as.numeric(Pima.te$type == "Yes"))
summary(flr)
```

Index