Package 'rpud'

October 1, 2016

Type Package

Title GPU computation in R

Version 0.6.1

Date 2016-10-01

Author Chi Yau <chi.yau@r-tutor.com>

Maintainer Chi Yau <chi.yau@r-tutor.com>

URL http://www.r-tutor.com/gpu-computing

LazyData yes

Depends R (>= 3.0.0)

SystemRequirements NVIDIA CUDA Toolkit 8.0

Description This package provides R functions for performing statistical computation on GPU. The implementation is based on the NVIDIA CUDA Toolkit. Besides the core functionality, it also aims for robustness, performance, and scalability.

License GPL-3

R topics documented:

`rpud-package` *GPU computation in R*

Description

This package provides R functions for performing statistical computation on GPU. The implementation is based on the NVIDIA CUDA Toolkit. Besides the core functionality, it also aims for robustness, performance, and scalability.

Details

Package:	rpud
Type:	Package

rpuDist	Distance matrix
rpuHclust	Hierarchical clustering
rpucor	Kendall's tau-b correlation
rpucor.test	Significant test of Kendall's tau-b
rpusvm	Support vector machine modelling
rvbm	Variational Bayesian multiclass probit regression
rhierLinearModel	Hierarchical Linear Model
rhierMnlRwMixture	Hierarchical Multinomial Logit Model
rpudl	Deep Learning Model

Author(s)

Maintainer: Chi Yau <chi.yau@r-tutor.com>

`plot.rpusvm` *Plot an SVM Model Trained by* `rpusvm`

Description

The `plot.rpusvm` function creates a scatter plot of the input data of a `rpusvm` fit for classification models by highlighting the classes and support vectors. Optionally, draws a filled contour plot of the class regions.

Usage

```
## S3 method for class 'rpusvm'
plot(x, data, formula, fill = TRUE, grid = 50, slice = list(),
symbolPalette = palette(), svSymbol = "x", dataSymbol = "o", ...)
```

Arguments

`x`	An S3 object that inherits from `rpusvm` class.
`data`	data to visualize. Should be the same used for fitting.
`formula`	formula selecting the visualized two dimensions. Only needed if more than two input variables are used.
`fill`	switch indicating whether a contour plot for the class regions should be added.
`grid`	granularity for the contour plot.
`slice`	a list of named values for the dimensions held constant (only needed if more than two variables are used). The defaults for unspecified dimensions are 0 (for numeric variables) and the first level (for factors). Factor levels can either be specified as factors or character vectors of length 1.
`symbolPalette`	Color palette used for the class the data points and support vectors belong to.
`svSymbol`	Symbol used for support vectors.
`dataSymbol`	Symbol used for data points (other than support vectors).
`...`	additional graphics parameters passed to `filled.contour` and `plot`.

Note

This is a clone of the `plot.svm` function in `e1071`.

Author(s)

Chi Yau (based on plot.svm in e1071 by David Meyer)
<chi.yau@r-tutor.com>

See Also

rpusvm, e1071::plot.svm

Examples

```
## Not run:
# copied from e1071
library(rpud)

data(cats)
m <- rpusvm(Sex~., data = cats)
plot(m, cats)

## more than two variables: fix 2 dimensions
data(iris)
m2 <- rpusvm(Species~., data = iris)
plot(m2, iris, Petal.Width ~ Petal.Length,
     slice = list(Sepal.Width = 3, Sepal.Length = 4))

## plot with custom symbols and colors
plot(m, cats, svSymbol = 1, dataSymbol = 2, symbolPalette = rainbow(4),
color.palette = terrain.colors)



## End(Not run)
```

---

plot.rvbm *Diagnostics Plots for Variational Bayesian Multiclass Probit Regression*

Description

The plot.rvbm method plots the history of marginal likelihood lower bound, predictive likelihood, test error and estimated covariance parameters during the iterations of the Gaussian process computation.

Usage

```
## S3 method for class 'rvbm'
plot(x, ...)
```

Arguments

x	An S3 object that inherits from the class rvbm.
...	Not used.

Note

This is a clone of the plotDiagnostics method in vbmp.

Author(s)

Chi Yau
<chi.yau@r-tutor.com>

See Also

rvbm, vbmp

Examples

```
## Not run:
library(rpud)

x <- rvbm.sample.train$X
y <- rvbm.sample.train$t.class
model.rvbm <- rvbm(
    x, y, x, y,
    theta = rep(1, ncol(x)),
    control = list(
            sKernelType="gaussian",
            bThetaEstimate=TRUE,
            bMonitor=TRUE,
            maxIts=12,
            InfoLevel=1)
)

plot(model.rvbm)


## End(Not run)
```

---

predict.rpudl *Predictions of an RPUDL Model*

Description

Predict the classification outcome of a collection of input vectors based on a trained RPUDL model.

Usage

```
## S3 method for class 'rpudl'
predict(
    model,
    x,
    ...,
    log.level = 1
    )
```

Arguments

model	a trained rpudl model object for prediction
x	An m by n matrix, where m is the size of each input vector, and n is the number of input vectors
...	further arguments to be passed to or from methods
log.level	level of the method output (0 = silent, 1 = basic output, 2 = detailed output)

Value

A vector whose length is the number of input vectors. the 'i-th' entry in the vector is the prediction class of the 'i-th' input vector. The probability output of each input vector is stored in the "decision.values".

Author(s)

Chi Yau
<chi.yau@r-tutor.com>

See Also

rpudl,
rpudlDataSource,
rpudlTestDataSample,
rpudlTrain

Examples

```
## Not run:
# create data source
ds <- rpudlDataSource(
        data.format="lmdb",
        data.dir="data/mnist",
        train.data="mnist-official-data_train_lmdb",
        test.data="mnist-official-data_test_lmdb",
        data.shape=c(28, 28)
)

# create model
model <- rpudl(
```

```
        "mnist_mpl_lenet.prototxt",
        data.source=ds
)

# train model
model <- rpudlTrain(model, batch=100, iter=1000)

# extract some test samples
num <- 12
obj <- rpudlTestDataSample(ds, c(1, num))

# predictions
res <- predict(model, obj$x)

# find num of errors
sum((obj$y+1) == res)

## End(Not run)
```

`predict.rpusvm` *Predict Method for Support Vector Machines*

Description

The `predict.rpusvm` function predicts values based upon an SVM model trained by `rpusvm` in the non-free `rpudplus` add-on.

This method is not supported in the free `rpud` package.

Usage

```
## S3 method for class 'rpusvm'
predict(object, newdata, decision.values = FALSE,
probability = FALSE, ..., na.action = na.omit, verbose = TRUE)
```

Arguments

`object`	An S3 object that inherits from `rpusvm` class.
`newdata`	An object containing the new input data: either a matrix or a sparse matrix (object of class `Matrix` provided by the **Matrix** package, or of class `matrix.csr` provided by the **SparseM** package. A vector will be transformed to a n x 1 matrix.
`decision.values`	Logical controlling whether the decision values of all binary classifiers computed in multiclass classification shall be computed and returned.
`probability`	Logical indicating whether class probabilities should be computed and returned. Only possible if the model was fitted with the `probability` option enabled.

na.action	A function to specify the action to be taken if 'NA's are found. The default action is na.omit, which leads to rejection of cases with missing values on any required variable. An alternative is na.fail, which causes an error if NA cases are found. (NOTE: If given, this argument must be named.)
...	Currently not used.
verbose	logical indicating whether progress information should be displayed. (default: TRUE)

Value

A vector of predicted values (for classification: a vector of labels, for density estimation: a logical vector). If decision.value is TRUE, the vector gets a "decision.values" attribute containing a n x c matrix (n number of predicted values, c number of classifiers) of all c binary classifiers' decision values. There are k * (k - 1) / 2 classifiers (k number of classes). The colnames of the matrix indicate the labels of the two classes. If probability is TRUE, the vector gets a "probabilities" attribute containing a n x k matrix (n number of predicted values, k number of classes) of the class probabilities.

Note

If the training set was scaled by rpusvm (done by default), the new data is scaled accordingly using scale and center of the training data.

Author(s)

Chi Yau (based on R doc of predict.svm in e1071 by David Meyer)
<chi.yau@r-tutor.com>

See Also

rpusvm, e1071::predict.svm

Examples

```
## Not run:
# copied from e1071
library(rpud)

data(iris)
attach(iris)

## classification mode
# default with factor response:
model <- rpusvm(Species ~ ., data = iris)

# alternatively the traditional interface:
x <- subset(iris, select = -Species)
y <- Species
model <- rpusvm(x, y, probability = TRUE)
```

```
print(model)
summary(model)

# test with train data
pred <- predict(model, x)
# (same as:)
pred <- fitted(model)

# compute decision values and probabilites
pred <- predict(model, x, decision.values = TRUE, probability = TRUE)
attr(pred, "decision.values")[1:4,]
attr(pred, "probabilities")[1:4,]

## try regression mode on two dimensions

# create data
x <- seq(0.1, 5, by = 0.05)
y <- log(x) + rnorm(x, sd = 0.2)

# estimate model and predict input values
m   <- rpusvm(x, y)
new <- predict(m, x)

# visualize
plot   (x, y)
points (x, log(x), col = 2)
points (x, new, col = 4)

## density-estimation

# create 2-dim. normal with rho=0:
X <- data.frame(a = rnorm(1000), b = rnorm(1000))
attach(X)

# traditional way:
m <- rpusvm(X, gamma = 0.1)

# formula interface:
m <- rpusvm(~., data = X, gamma = 0.1)
# or:
m <- rpusvm(~ a + b, gamma = 0.1)

# test:
newdata <- data.frame(a = c(0, 4), b = c(0, 4))
predict (m, newdata)

# visualize:
plot(X, col = 1:1000 %in% m$index + 1, xlim = c(-5,5), ylim=c(-5,5))
points(newdata, pch = "+", col = 2, cex = 5)

## End(Not run)
```

predict.rvbm *Predict Method for Variational Bayesian Multiclass Probit Regression*

Description

The predict.rvbm method predicts class membership of given test data based on the Gaussian process model created by rvbm.

Usage

```
## S3 method for class 'rvbm'
predict(object, X.TEST=NULL, ...)
```

Arguments

object	An S3 object that inherits from the class rvbm.
X.TEST	A matrix of new test data. If NULL, the feature matrix in obj is used instead.
...	Not used.

Value

An vector containing the predicted class membership based on the maximum posterior probability. An attribute named Ptest contains the predicted posterior probability.

Note

This replaces the predictCPP and predClass methods in vbmp.

Author(s)

Chi Yau
<chi.yau@r-tutor.com>

See Also

rvbm, vbmp

Examples

```
## Not run:
library(rpud)

x <- rvbm.sample.train$X
y <- rvbm.sample.train$t.class
model.rvbm <- rvbm(
    x, y, x, y,
    theta = rep(1, ncol(x)),
    control = list(
```

```
            sKernelType="gaussian",
            bThetaEstimate=TRUE,
            bMonitor=TRUE,
            InfoLevel=1)
)

res <- predict(model.rvbm, rvbm.test$X)
err <- sum(res != rvbm.test$t.class); err
rate <- err/length(rvbm.test$t.class)*100; rate

## End(Not run)
```

`read.svm.data` *Reading SVM Training Data from a File*

Description

The `read.svm.data` reads a training file in the LIBSVM data format, and is a replacement of the `read.matrix.csr` function in `e1071`.

In the non-free `rpudplus` add-on, the method redirects to a native C++ implementation with enhanced robustness and performance.

This method is not supported in the free `rpud` package.

Usage

```
read.svm.data(file, fac = TRUE, sparse = TRUE)
```

Arguments

`file`	The path name of the file in LIBSVM training data format.
`fac`	If `TRUE` and y-values are stored in the file, the values are interpreted as factor levels.
`sparse`	If `TRUE`, the x-values are loaded into a `matrix.csr` object provided by the `SparseM` package. If `FALSE`, the x-values are loaded into a data matrix instead.

Value

The method returns a list with two components:

`x`	object of class `matrix.csr` or a data matrix, depending on the value of `sparse`.
`y`	vector of numeric values or factor levels, depending on the value of `fac`.

Author(s)

Chi Yau (based on R doc of `read.matrix.csr` in `e1071` by David Meyer)
<chi.yau@r-tutor.com>

See Also

matrix.csr

Examples

```
## Not run:

library(rpud)
library(SparseM)

filepath <- file.path(.path.package(package="rpud"), "runit/data/heart_scale")
hs <- read.svm.data(filepath, fac=TRUE)

x <- hs$x
y <- hs$y

model <- rpusvm(x, y)
summary(model)


## End(Not run)
```

`rhierLinearModel` *Gibbs Sampler for Hierarchical Linear Model*

Description

`rhierLinearModel` implements a Gibbs Sampler for hierarchical linear models with a normal prior.

Usage

```
rhierLinearModel(Data, Prior, Mcmc, output)
```

Arguments

`Data`	list(regdata,Z) (Z optional).
`Prior`	list(Deltabar,A,nu.e,ssq,nu,V) (optional).
`Mcmc`	list(R,keep,chains) (R required).
`output`	binary output format.

Details

Model: length(regdata) regression equations.
$y_i = X_i beta_i + e_i$. $e_i \sim N(0, tau_i)$. nvar X vars in each equation.

Priors:
$tau_i \sim$ nu.e$*ssq_i/\chi^2_{nu.e}$. tau_i is the variance of e_i.

$beta_i \sim$ N(ZDelta[i,],V_{beta}).
Note: ZDelta is the matrix Z * Delta; [i,] refers to ith row of this product.

$vec(Delta)$ given $V_{beta} \sim N(vec(Deltabar), V_{beta}(x)A^{-1})$.
$V_{beta} \sim IW(nu, V)$.
$Delta, Deltabar$ are nz x nvar. A is nz x nz. V_{beta} is nvar x nvar.

Note: The default of Z is iota(nreg x 1).

List arguments contain:

- `regdata` list of lists with X,y matrices for each of length(regdata) regressions
- `regdata[[i]]$X` X matrix for equation i
- `regdata[[i]]$y` y vector for equation i
- `Deltabar` nz x nvar matrix of prior means (def: 0)
- `A` nz x nz matrix for prior precision (def: .01I)
- `nu.e` d.f. parm for regression error variance prior (def: 3)
- `ssq` scale parm for regression error var prior (def: var(y_i))
- `nu` d.f. parm for Vbeta prior (def: nvar+3)
- `V` Scale location matrix for Vbeta prior (def: nu*I)
- `R` number of MCMC draws
- `keep` MCMC thinning parm: keep every keepth draw (def: 1)
- `chains` number of MCMC simulation chains for `CODA` diagnostics
- `output` supported binary output format: `default`, `bayesm`, or `coda`.

Value

For `default` format: a list containing

`betadraw`	R/keep x nreg x nvar array of individual regression coef draws
`taudraw`	R/keep x nreg array of error variance draws
`Deltadraw`	R/keep x nz x nvar array of Deltadraws
`Vbetadraw`	R/keep x nvar*nvar array of Vbeta draws

For `bayesm` format: a list containing

`betadraw`	nreg x nvar x R/keep array of individual regression coef draws
`taudraw`	R/keep x nreg array of error variance draws
`Deltadraw`	R/keep x nz x nvar array of Deltadraws
`Vbetadraw`	R/keep x nvar*nvar array of Vbeta draws

For `CODA` output format: A list of CODA compatible simulation chains in default format. Only the upper triangle of each VbetaDraw sample is kept for coda diagnostics.

Author(s)

Chi Yau (based on R doc of rhierLinearModel in bayesm by Peter Rossi)
<chi.yau@r-tutor.com>

References

Rossi, Allenby and McCulloch:
Bayesian Statistics and Marketing Ch 3
http://www.perossi.org/home/bsm-1

Examples

```
## Not run:
library(rpud)
data.path <- file.path(path.package(package="rpud"), "runit/data")

##################################################
# sample data

Z <- read.table(
        file.path(data.path, "rhierlm-n100-z.txt"),
        header=FALSE)
Z <- as.matrix(Z)

data <- read.table(
        file.path(data.path, "rhierlm-n100-data.txt"),
        header=FALSE)
data <- as.matrix(data)

regkey <- unique(data[,1])
colvar <- 3:ncol(data)

regdata <- NULL
for (reg in regkey) {

    filter <- (data[,1]==reg)
    y <- data[filter,2]
    X <- data[filter,colvar]

    regdata[[reg]] <- list(y=y,X=X)
}

# parameters
R <- 1000
keep <- 1

Data <- list(regdata=regdata,Z=Z)
Mcmc <- list(R=R,keep=keep)

set.seed(66)
out<-rpud::rhierLinearModel(Data=Data,Mcmc=Mcmc,output="bayesm")
```

```
library(bayesm)
cat("\n*** Summary of Delta Draws ***",fill=TRUE)
summary(out$Deltadraw)

cat("\n*** Summary of Vbeta Draws ***",fill=TRUE)
summary(out$Vbetadraw)

## End(Not run)
```

`rhierMnlRwMixture` *MCMC Algorithm for Hierarchical Multinomial Logit with Mixture of Normals Heterogeneity*

Description

`rhierMnlRwMixture` is a MCMC algorithm for a hierarchical multinomial logit with a mixture of normals heterogeneity distribution. This is a hybrid Gibbs Sampler with a RW Metropolis step for the MNL coefficients for each panel unit.

Usage

```
rhierMnlRwMixture(Data, Prior, Mcmc, output)
```

Arguments

`Data`	list(p,lgtdata,Z) (Z is optional)
`Prior`	list(a,deltabar,Ad,mubar,Amu,nu,V,ncomp) (all but ncomp are optional)
`Mcmc`	list(s,w,R,keep,chains) (R required)
`output`	binary output format.

Details

Model:
$y_i \sim MNL(X_i, beta_i)$. i=1,..., length(lgtdata). $theta_i$ is nvar x 1.

$beta_i$= ZDelta[i,] + u_i.
Note: here ZDelta refers to Z%*%D, ZDelta[i,] is ith row of this product.
Delta is an nz x nvar array.

$u_i \sim N(mu_{ind}, Sigma_{ind})$. $ind \sim$ multinomial(pvec).

Priors:
$pvec \sim$ dirichlet (a)
$delta = vec(Delta) \sim N(deltabar, A_d^{-1})$
$mu_j \sim N(mubar, Sigma_j(x) Amu^{-1})$
$Sigma_j \sim$ IW(nu,V)

Lists contain:

- p p is number of choice alternatives
- lgtdatalist of lists with each cross-section unit MNL data
- lgtdata[[i]]$y n_i vector of multinomial outcomes (1,. . . ,m)
- lgtdata[[i]]$X n_i*p by nvar design matrix for ith unit
- avector of length ncomp of Dirichlet prior parms (def: rep(5,ncomp))
- deltabarnz*nvar vector of prior means (def: 0)
- Ad prior prec matrix for vec(D) (def: .01I)
- mubar nvar x 1 prior mean vector for normal comp mean (def: 0)
- Amu prior precision for normal comp mean (def: .01I)
- nu d.f. parm for IW prior on norm comp Sigma (def: nvar+3)
- V pds location parm for IW prior on norm comp Sigma (def: nuI)
- ncomp number of components used in normal mixture
- s scaling parm for RW Metropolis (def: 2.93/sqrt(nvar))
- w fractional likelihood weighting parm (def: .1)
- R number of MCMC draws
- keep MCMC thinning parm: keep every keepth draw (def: 1)
- chains number of MCMC simulation chains for CODA diagnostics
- output supported binary output format: default, bayesm, or coda.

Value

For default format: a list containing

llkdraw	R/keep array of log likelikelhood of draws
betadraw	R/keep x nvar x nlgt array of draws of betas
Deltadraw	R/keep x nvar x nz array of Delta draws, first row is initial value
probdraw	R/keep x ncomp array of draws of probs of mixture components
mudraw	R/keep x nvar x ncomp array of draws of mean parameter of beta coefficients
rootdraw	R/keep x nvar x nvar x ncomp array of error variances of beta coefficients

For bayesm format: a list containing

Deltadraw	R/keep x nz*nvar matrix of draws of Delta, first row is initial value
betadraw	nlgt x nvar x R/keep array of draws of betas
nmix	list of 3 components, probdraw, NULL, compdraw
loglike	log-likelihood for each kept draw (length R/keep)

For CODA output format: A list of CODA compatible simulation chains in default format.

Note

More on `probdraw` component of nmix list in `bayesm` output format:
R/keep x ncomp matrix of draws of probs of mixture components (pvec)
More on `compdraw` component of return value list:

- compdraw[[i]] the ith draw of components for mixtures
- compdraw[[i]][[j]] ith draw of the jth normal mixture comp
- compdraw[[i]][[j]][[1]] ith draw of jth normal mixture comp mean vector
- compdraw[[i]][[j]][[2]] ith draw of jth normal mixture cov parm (rooti)

Note: Z should **not** include an intercept and is centered for ease of interpretation.

Be careful in assessing prior parameter, Amu. .01 is too small for many applications. See Rossi et al, chapter 5 for full discussion.

Note: as of version 2.0-2 of `bayesm`, the fractional weight parameter has been changed to a weight between 0 and 1. w is the fractional weight on the normalized pooled likelihood. This differs from what is in Rossi et al chapter 5, i.e.

$like_i^(1-w) x like_p ooled^(n_i/N) * w)$

Large R values may be required (>20,000).

Author(s)

Chi Yau (based on R doc of `rhierMnlRwMixture` in `bayesm` by Peter Rossi)
<chi.yau@r-tutor.com>

References

Rossi, Allenby and McCulloch:
Bayesian Statistics and Marketing Ch 5
http://www.perossi.org/home/bsm-1

Examples

```
## Not run:

library(rpud)
data.path <- file.path(path.package(package="rpud"), "runit/data")

# load p, Delta, Z, and simlgtdata
load(file.path(data.path, "rhierMnl-n2x6x5-rd.txt.gz"))

## set parms for priors and Z
R <- 5000
keep <- 5
seed <- 66
```

```
Prior <- list(ncomp=4)
Mcmc <- list(R=R, keep=keep)
Data <- list(p=p, lgtdata=simlgtdata, Z=Z)

set.seed(seed)

system.time(
out <- rpud::rhierMnlRwMixture(
        Data=Data,
        Prior=Prior,
        Mcmc=Mcmc,
        output="bayesm"
))

library(bayesm)

cat("Summary of Delta draws", fill=TRUE)
summary(out$Deltadraw, tvalues=as.vector(Delta))

cat("Summary of Normal Mixture Distribution", fill=TRUE)
summary(out$nmix)

if(0) {
    ## plotting examples
    plot(out$betadraw)
    plot(out$nmix)
}


## End(Not run)
```

`rmultireg` *Draw from the Posterior of a Multivariate Regression*

Description

`rmultireg` draws from the posterior of a Multivariate Regression model with a natural conjugate prior.

Usage

```
rmultireg(Y, X, Bbar, A, nu, V, rep)
```

Arguments

`Y`	n x m matrix of observations on m dep vars
`X`	n x k matrix of observations on indep vars (supply intercept)
`Bbar`	k x m matrix of prior mean of regression coefficients
`A`	k x k Prior precision matrix

nu	d.f. parameter for Sigma
V	m x m pdf location parameter for prior on Sigma
rep	number of posterior sample draws

Details

Model: $Y = XB + U$. $cov(u_i) = Sigma$. B is k x m matrix of coefficients. $Sigma$ is m x m covariance.

Priors: $beta$ given $Sigma \sim N(betabar, Sigma(x)A^{-1})$. $betabar = vec(Bbar)$; $beta = vec(B)$ $Sigma \sim$ IW(nu,V).

Value

A list of the components of draws from the posterior

betadraw	draws of regression coefficient matrix
Sigmadraw	draws of Sigma

Author(s)

Chi Yau (based on R doc of rmultireg in bayesm by Peter Rossi)
<chi.yau@r-tutor.com>

References

Rossi, Allenby and McCulloch:
Bayesian Statistics and Marketing Ch 2
http://www.perossi.org/home/bsm-1

Examples

```
## Not run:
library(rpud)

data.path <- file.path(path.package(package="rpud"), "runit/data")

# sample X
X <- read.table(
        file.path(data.path, "rmultireg-n100x2-x.txt"),
        header=FALSE)
X <- as.matrix(X)

# sample Y
Y <- read.table(
        file.path(data.path, "rmultireg-n100x2-y.txt"),
        header=FALSE)
Y <- as.matrix(Y)

# parameters
n <- nrow(X)
```

```
k <- ncol(X)
m <- ncol(Y)

Bbar <- matrix(c(1, 0, -1, 1),nrow=k,ncol=m)    # prior conjugate (normal) mean of B
A <- matrix(c(0.1, -0.05, -0.05, 0.1),nrow=k)   # prior conjugate (normal) precision of B
nu <- 3; V <- nu*diag(m)                         # prior Wishart params of Sigma

# mcmc params
R <- 2000

set.seed(66)
out <- rpud::rmultireg(Y, X, Bbar, A, nu, V, rep=R)

# true params
B <- matrix(c(1,2,-1,3),ncol=m)
Sigma <- matrix(c(1,.5,.5,1),ncol=m)

cat(" Beta draws ",fill=TRUE)
mat <- apply(out$betadraw,2,quantile,probs=c(.01,.05,.5,.95,.99))
mat <- rbind(as.vector(B),mat); rownames(mat)[1]="beta"
print(mat)

cat(" Sigma draws",fill=TRUE)
mat <- apply(out$Sigmadraw,2,quantile,probs=c(.01,.05,.5,.95,.99))
mat <- rbind(as.vector(Sigma),mat); rownames(mat)[1]="Sigma"
print(mat)

## End(Not run)
```

`rpuchol` *GPU Accelerated Cholesky Decomposition*

Description

The `rpuchol` method computes the upper triangular Cholesky decomposition of a real positive definite symmetric matrix using GPU.

Usage

```
rpuchol(x, ...)
```

Arguments

`x`	A positive symmetric data matrix.
`...`	Not used.

Value

The upper triangular Cholesky decomposition.

Author(s)

Chi Yau
<chi.yau@r-tutor.com>

See Also

chol

Examples

```
## Not run:
library(rpud)

N <- 20
x <- matrix(runif(N*5), ncol=N)
A <- t(x)
rpuchol(A)


## End(Not run)
```

---

rpucor *Kendall's Tau-b Correlation Coefficient on GPU*

Description

This function computes the correlation matrix of two multivariate statistics.

Usage

```
## Default S3 method:
rpucor(x, y = NULL,
    use = "everything",
    method = c("pearson", "spearman", "kendall", "gamma"), ...)
```

Arguments

x	a numeric vector, matrix or data frame.
y	a vector, matrix or data frame with compatible dimensions to 'x'. The default 'NULL' value is equivalent to 'y = x'.
use	a string representing the method to handle missing values. Currently supported values are: "everything"(default), "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs".
method	a string representing the name of the correlation to be computed. The supported values are: "pearson"(default), "spearman", "kendall", and "gamma".
...	further arguments to be passed to or from methods.

Details

Perform the Kendall correlation with GPU using the rpudplus add-on for rpud. The Pearson and Spearman correlation are just stubs that invoke the default implementation of R. In absense of rpudplus, rpud computes the Kendall correlation also with the stock R implementation, and there will be no speed gain.

Value

The correlation matrix of the input statistics. An attribute named "method" contains the name of the correlation, and another attribute named "use" contains the method to handle missing values.

See Also

cor, gpuCor

Examples

```
## Not run:

num <- 5
dim1 <- 6
dim2 <- 8
x <- matrix(runif(num*dim1), num, dim1)
y <- matrix(runif(num*dim2), num, dim2)
rpucor(x, y, method = "kendall")

# introduce missing values
x[3,5] <- NA
y[4,1] <- NA
rpucor(x, y, method = "kendall", use="pairwise.complete.obs")


## End(Not run)
```

---

rpucor.test *Compute the p-values of the correlation matrix*

Description

This function computes the p-values of the correlation matrix of two multivariate statistics. Current implementation supports only finding p-values of the Kendall Tau-b coefficient.

Usage

```
rpucor.test(x, ...)

## Default S3 method:
rpucor.test(x, y = NULL,
    alternative = c("two.sided", "less", "greater"),
    use = "everything", method = "kendall", ...)
```

Arguments

x	a numeric vector, matrix or data frame.
y	a vector, matrix or data frame with compatible dimensions to 'x'. The default 'NULL' value is equivalent to 'y = x'.
alternative	a string representing the alternative hypothesis. It must be one of the following: "two.sided"(default), "less", or "greater".
use	a string representing the method to handle missing values. Currently supported values are: "everything"(default), "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs".
method	a string representing the name of the correlation to be computed. Currently, the only supported value is "kendall".
...	further arguments to be passed to or from methods.

Details

Compute the p-values of the Kendall Tau-b correlation with GPU using the rpudplus add-on for rpud.

Value

A list of class "rpucor.test" containing the following:

statistic	matrix of the Z statistics.
p.value	matrix of the p-values.
estimate	matrix of the correlation coefficient.
null.value	the zero value for the null hypothesis.
alternative	a string representing the alternative hypothesis.
method	the correlation method.
use	the method to handle the missing values.
data.name	a string representing the input data.

See Also

rpucor, cor.test

Examples

```
## Not run:

num <- 5
dim1 <- 6
dim2 <- 8
x <- matrix(runif(num*dim1), num, dim1)
y <- matrix(runif(num*dim2), num, dim2)
rpucor.test(x, y, method = "kendall")
```

```
# introduce missing values
x[3,5] <- NA
y[4,1] <- NA
rpucor.test(x, y, method = "kendall", use="pairwise.complete.obs")


## End(Not run)
```

---

rpuDist *Compute the Distance Matrix on GPU*

Description

This function computes the distance between each vector of the 'points' argument using the metric specified by 'method'.

Usage

```
rpuDist(points, method = "euclidean", diag = FALSE, upper = FALSE, p = 2)
```

Arguments

points	a matrix of floating point numbers in which each row is a vector in R^n space where n is ncol(points).
method	a string representing the name of the metric to use to calculate the distance between the vectors of 'points'. Currently supported values are: "binary", "canberra", "euclidean", "manhattan", "maximum", and "minkowski".
diag	logical value indicating whether the diagonal of the distance matrix should be printed by print.dist.
upper	logical value indicating whether the upper triangle of the distance matrix should be printed by print.dist.
p	The power of the Minkowski distance.

Details

Compute the distance matrix with GPU. The processing capacity of rpud is confined by the GPU device memory by default. With the rpudplus add-on, rpud will make use of the system RAM, and can handle much larger data sets.

Value

a class of type "dist" containing floating point numbers representing the distances between vectors from the 'points' argument.

See Also

dist, gpuDist

Examples

```
## Not run:
numVectors <- 5
dimension <- 10
Vectors <- matrix(runif(numVectors*dimension), numVectors, dimension)
rpuDist(Vectors, "euclidean")
rpuDist(Vectors, "maximum")
rpuDist(Vectors, "manhattan")
rpuDist(Vectors, "canberra")
rpuDist(Vectors, "binary")
rpuDist(Vectors, "minkowski")

## End(Not run)
```

`rpudl` *RPUDL Model of Deep Learning*

Description

Creates an S3 object that represents a `RPUDL` model. It contains the necessary information for model training afterwards.

Usage

```
## Default S3 method:
rpudl(
    model.file,
    data.source,
    ...,
    log.level = 1
    )
```

Arguments

`model.file`	path location of a text file that defines an `RPUDL` model in the codeprotobuf text format
`data.source`	a `RPUDL` object that represents the training and testing data
`...`	further arguments to be passed to or from methods
`log.level`	level of the method output (0 = silent, 1 = basic output, 2 = detailed output)

Details

An `RPUDL` model is described in Google Protocol Buffers (`protobuf`) message format. The .proto file that defines the message type is posted online at http://www.r-tutor.com. The `RPUDL` runtime supports save/resume incremental training, and autoencoders for pre-training. It uses `LMDB` database as native data input format.

Value

An S3 rpudl object that represents an RPUDL model, containing:

model.file	path location of the RPUDL model in protobuf text format
model.spec	text description of the RPUDL model file in protobuf text format
data.format	format of the data set
data.dir	path location of the data set
data.shape	the width and height of each data item
data.channels	number of channels of each data item
train.data	name of the training LMDB database
test.data	name of the testing LMDB database
param.data	parametric values of individual layers in the model
mean.data	mean value of the training data items
model.classes	number of classification classes of the model
learning.rate	manual override of the learning rate, ignored if zero or negative
cost.log	accumulative log of the model cost after each training iteration
cost	model cost evaluated against the test data
loss	prediction error ratio evaluated against the test data

Author(s)

Chi Yau
<chi.yau@r-tutor.com>

See Also

rpudlCreateDataSource,
rpudlTrain,
rpudlPretrain,
predict.rpudl

Examples

```
## Not run:
# create data source
ds <- rpudlCreateDataSource(
        data.format="lmdb",
        data.dir="data/mnist",
        train.data="mnist-official-data_train_lmdb",
        test.data="mnist-official-data_test_lmdb",
        data.shape=c(28, 28)
)

# create model
model <- rpudl(
        "mnist_mpl_lenet.prototxt",
```

```
        data.source=ds
)

# train model
model <- rpudlTrain(model, batch=100, iter=1000)

## End(Not run)
```

`rpudlCreateDataSource` *RPUDL Data Source for Deep Learning*

Description

Creates an S3 object that contains training and testing data.

Usage

```
rpudlCreateDataSource(
        data.format = c("lmdb", "cifar10", "cifar100", "minst"),
        data.dir,
        data.shape = NULL,
        data.channels = 1,
        train.data = NULL,
        test.data = NULL,
        url = NULL
    )
```

Arguments

`data.format`	format of the data set
`data.dir`	path location of the data set
`data.shape`	the width and height of each data item
`data.channels`	number of channels of each data item
`train.data`	name of the training `LMDB` database
`test.data`	name of the testing `LMDB` database
`url`	if non-null, specifies alternative URL for downloading `MNIST` or `CIFAR` data

Details

The supported data formats include `MNIST`, `CIFAR`, and `LMDB`. If the data format is `MNIST` or `CIFAR`, it will automatically download the data from its origin. For `LMDB` data, the database must locally exists beforehand. To use custom data for model training, it must be imported into an `LMDB` database. Each record in the database is a `protobuf` message with the following type: `message Datum { required bytes data = 1; required uint32 label = 2; }`

Value

An S3 object that inherits the class `rpudlDataSource` containing:

`data.format`	format of the data set
`data.dir`	path location of the data set
`data.shape`	the width and height of each data item
`data.channels`	number of channels of each data item
`train.data`	name of the training `LMDB` database
`test.data`	name of the testing `LMDB` database

Author(s)

Chi Yau
<chi.yau@r-tutor.com>

See Also

rpudl,
predict.rpudl,
rpudlTrain,
rpudlPretrain,
rpudlGetTestingDataSamples,
rpudlFindTrainingDataMean,
rpudlWriteLMDB

Examples

```
## Not run:
# create data source
ds <- rpudlCreateDataSource(
        data.format="lmdb",
        data.dir="data/mnist",
        train.data="mnist-official-data_train_lmdb",
        test.data="mnist-official-data_test_lmdb",
        data.shape=c(28, 28)
)

# create model
model <- rpudl(
        "mnist_mpl_lenet.prototxt",
        data.source=ds
)

# train model
model <- rpudlTrain(model, batch=100, iter=1000)

## End(Not run)
```

rpudlFindTrainingDataMean *Mean Input Values of the Training Data*

Description

Finding mean input values of the training data in an RPUDL data source. The result is to be saved to an output file.

Usage

```
rpudlFindTrainingDataMean(
    data.source,
    filename = "data.mean",
    ...,
    log.level = 1
    )
```

Arguments

data.source	An S3 object that inherits the class rpudl.data.source
filename	Name of the output file, which is located in the same path folder as the data set.
...	further arguments to be passed to or from methods
log.level	level of the method output (0 = silent, 1 = basic output, 2 = detailed output)

Author(s)

Chi Yau
<chi.yau@r-tutor.com>

See Also

rpudlCreateDataSource

Examples

```
## Not run:
ds <- rpudlCreateDataSource(
        data.format="lmdb",
        data.dir=file.path(data.dir, "cifar10"),
        train.data="samples_train_lmdb",
        test.data="samples_test_lmdb",
        data.shape=c(32, 32),
        data.channels=3
)

rpudlFindTrainingDataMean(ds, filename="samples.mean")

## End(Not run)
```

rpudlGetLayerWeights *Weights of a Layer Element in an RPUDL model*

Description

Retrieve weights of a layer element in an rpudl object

Usage

```
rpudlGetLayerWeights(model, layer.name)
```

Arguments

model	a rpudl model object
layer.name	name of a layer element in the model

Value

A 4 dimensional array that inherits the class rpudlGetLayerWeights and contains the weight parameters of a layer element. The first and second dimensions are the number of rows and columns of the weight parameters. The third dimension is the number of input channels, and the fourth dimension is the number of output channels.

Author(s)

Chi Yau
<chi.yau@r-tutor.com>

See Also

rpudl,
rpudlTrain,
rpudlPlotLayerWeights

Examples

```
## Not run:
# create data source
ds <- rpudlCreateDataSource(
        data.format="lmdb",
        data.dir="data/mnist",
        train.data="mnist-official-data_train_lmdb",
        test.data="mnist-official-data_test_lmdb",
        data.shape=c(28, 28)
)

# create model
model <- rpudl(
```

```
        "mnist_mpl_lenet.prototxt",
        data.source=ds
)

# train model
model <- rpudlTrain(model, batch=100, iter=1000)

# extract weights
weights <- rpudlGetLayerWeights(model, "conv1"); str(weights)

# plot weights
library(gg2plot)
rpudlPlotLayerWeights(weights)

## End(Not run)
```

`rpudlGetTestingDataSamples` *Test Data Sample Extraction*

Description

Extract a sequence of data items from the test database.

Usage

```
rpudlGetTestingDataSamples(
    data.source,
    data.interval,
    label.offset = 1
    )
```

Arguments

`data.source`	An S3 object that inherits the class `rpudl.data.source`
`data.interval`	An integer pair that denotes the range of test data to be retrieved.
`label.offset`	Integer offset for having 1-based vector index of the labels.

Value

A list of the following:

`x`	m by n matrix where m is the size of each input data, n is the number of test data.
`y`	classification values from the test data

Author(s)

Chi Yau
<chi.yau@r-tutor.com>

See Also

rpudl,
rpudlCreateDataSource,
rpudlTrain,
predict.rpudl

Examples

```
## Not run:
# create data source
ds <- rpudlCreateDataSource(
        data.format="lmdb",
        data.dir="data/mnist",
        train.data="mnist-official-data_train_lmdb",
        test.data="mnist-official-data_test_lmdb",
        data.shape=c(28, 28)
)

# create model
model <- rpudl(
        "mnist_mpl_lenet.prototxt",
        data.source=ds
)

# train model
model <- rpudlTrain(model, batch=100, iter=1000)

# extract some test samples
num <- 12
obj <- rpudlGetTestingDataSamples(ds, c(1, num))

# predictions
res <- predict(model, obj$x)

# find num of errors
sum((obj$y+1) == res)

## End(Not run)
```

---

rpudlPlotLayerWeights *Plots of Weights in a Layer Element*

Description

Plotting the weight parameters of a layer element in an RPUDL model.

Usage

```
rpudlPlotLayerWeights(obj)
```

Arguments

obj	S3 object that inherits the class rpudl.weights

Details

There is a separate plot for each individual output channel. The plot is monochrome if there is only one input channel, in RGB color if there are three input channels. The plot is not available if the number of input channels is other than one or three.

Author(s)

Chi Yau
<chi.yau@r-tutor.com>

See Also

rpudl,
rpudlTrain,
rpudlGetLayerWeights

Examples

```
## Not run:
# create data source
ds <- rpudlCreateDataSource(
        data.format="lmdb",
        data.dir="data/mnist",
        train.data="mnist-official-data_train_lmdb",
        test.data="mnist-official-data_test_lmdb",
        data.shape=c(28, 28)
)

# create model
model <- rpudl(
        "mnist_mpl_lenet.prototxt",
        data.source=ds
)

# train model
model <- rpudlTrain(model, batch=100, iter=1000)

# extract weights
weights <- rpudlGetLayerWeights(model, "conv1"); str(weights)

# plot weights
library(gg2plot)
rpudlPlotLayerWeights(weights)
```

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

`rpudlPretrain` *RPUDL Model Pre-training*

Description

Pre-train a `RPUDL` deep learning model.

Usage

```
rpudlPretrain(
    model,
    learning.rate = 0,
    batch.size = 128,
    iterations = 128,
    display = 100,
    seed = 0,
    ...,
    log.level = 1
    )
```

Arguments

`model`	a `rpudl` model object to be pre-trained
`learning.rate`	manual override of the learning rate, ignored if zero or negative
`batch.size`	the size of a data batch for each training iteration
`iterations`	maximum number of training iterations
`display`	a logical to display training progress
`seed`	seed of random number generation
`...`	further arguments to be passed to or from methods
`log.level`	level of the method output (0 = silent, 1 = basic output, 2 = detailed output)

Details

Only the fully-connected layers will be pre-trained via one of the following mechanisms:

- Masked autoencoder
- Gaussian autoencoder
- Restricted Boltzmann Machine

Value

An `rpudl` object with pre-trained parameters.

Author(s)

Chi Yau
<chi.yau@r-tutor.com>

See Also

rpudl,
rpudlTrain,
rpudlPredict,

Examples

```
## Not run:
# create data source
ds <- rpudlCreateDataSource(
        data.format="lmdb",
        data.dir="data/mnist",
        train.data="mnist-official-data_train_lmdb",
        test.data="mnist-official-data_test_lmdb",
        data.shape=c(28, 28)
)

# create model
model <- rpudl(
        "masked_autoencoder.prototxt",
        data.source=ds
)

# pre-train
model <- rpudlPretrain(model, batch=100, iter=1000)

## End(Not run)
```

---

rpudlTrain *RPUDL Model Training*

Description

Optimizes a RPUDL deep learning model with training data.

Usage

```
rpudlTrain(
    model,
    learning.rate = 0,
    batch.size = 128,
    iterations = 128,
    display = 100,
```

```
top.n = 1,
seed = 0,
...,
log.level = 1
)
```

Arguments

model	a rpudl model object to be trained
learning.rate	manual override of the learning rate, ignored if zero or negative
batch.size	the size of a data batch for each training iteration
iterations	maximum number of training iterations
display	a logical to display training progress
top.n	a classification predict is considered an error if its top n results are all incorrect
seed	seed of random number generation
...	further arguments to be passed to or from methods
log.level	level of the method output (0 = silent, 1 = basic output, 2 = detailed output)

Details

The optimization training process supports the following algorithms:

- Stochastic Gradient Descent
- RMSProp
- AdaGrad
- Adam

Value

A new rpudl object with trained parameters.

Author(s)

Chi Yau
<chi.yau@r-tutor.com>

See Also

rpudl,
rpudlPretrain,
predict.rpudl

Examples

```
## Not run:
# create data source
ds <- rpudlCreateDataSource(
        data.format="lmdb",
        data.dir="data/mnist",
        train.data="mnist-official-data_train_lmdb",
        test.data="mnist-official-data_test_lmdb",
        data.shape=c(28, 28)
)

# create model
model <- rpudl(
        "mnist_mpl_lenet.prototxt",
        data.source=ds
)

# train model
model <- rpudlTrain(model, batch=100, iter=1000)

## End(Not run)
```

`rpudlWriteLMDB` *Export Supported Data Sets to LMDB*

Description

Export `CIFAR` and `MNIST` data sets to `LMDB`.

Usage

```
rpudlWriteLMDB(
    data.source,
    prefix = "rpudl",
    ...,
    log.level = 1
)
```

Arguments

`data.source`	An S3 object that inherits the class `rpudl.data.source`
`prefix`	Prefix of the training and testing LMDB databases. For the default prefix "rpudl", the training database would be "rpudl_train_lmdb", and the testing database would be "rpudl_test_lmdb". The directories must not exists beforehand. They will be located in the same path folder as the data set.
`...`	further arguments to be passed to or from methods
`log.level`	level of the method output (0 = silent, 1 = basic output, 2 = detailed output)

Author(s)

Chi Yau
<chi.yau@r-tutor.com>

See Also

rpudlCreateDataSource

Examples

```
## Not run:
ds <- rpudlCreateDataSource(
        data.format="mnist",
        data.dir="data/mnist"
)

rpudlWriteLMDB(ds, prefix="mnist-official-data")

## End(Not run)
```

---

rpuGetDevice *ID of the GPU device in use by the active host thread*

Description

Query the ID of the GPU device on which the active host thread executes the device code.

Usage

```
rpuGetDevice()
```

Value

Returns the ID of the GPU device on which the active host thread executes the device code.

Examples

```
rpuGetDevice()
```

rpuHclust *Hierarchical Clustering*

Description

This function finds hierarchical clusters from a distance matrix based an agglomerative algorithms.

Usage

```
rpuHclust(d, method="complete", members=NULL)
```

Arguments

d	a dissimilarity structure as produced by dist or rpuDist.
method	the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid".
members	NULL or a vector with length size of d. See the 'Details' section.

Details

Perform hierarchical clusterings with the rpudplus add-on for rpud. In absense of rpudplus, rpud perform the cluster analysis with the default R implementation, and there will be no speed gain.

Value

An object of class **hclust** which describes the tree produced by the clustering process.

See Also

hclust, gpuHclust

Examples

```
## Not run:
    numVectors <- 5
    dimension <- 10
    Vectors <- matrix(runif(numVectors*dimension), numVectors, dimension)
    distMat <- rpuDist(Vectors, "euclidean")
    myClust <- rpuHclust(distMat, "single")
    plot(myClust)

## End(Not run)
```

rpuScale *Scaling SVM Training Data*

Description

The rpuScale method performs scaling on SVM training data. It is not supported in the free rpud package.

Usage

```
rpuScale(x, y=NULL, scale=TRUE)
```

Arguments

x	a data matrix, a vector, or a sparse matrix (object of class Matrix provided by the **Matrix** package, or of class matrix.csr provided by the **SparseM** package.
y	a response vector with one label for each row/component of x. Can be either a factor (for classification tasks) or a numeric vector (for regression).
scale	It can be a numeric vector or a single logical value. If it is a numeric vector, the first and second vector elements will be the lower and upper bounds of each attributes in the x component after scaling. The third and fourth optional vector elements will be the lower and upper bounds of the y component after scaling. This is ignored if y is of factor type. If scale is a single logical TRUE value, the default scale vector c(0, 1, -1, 1) will be applied.

Value

rpuScale returns a list with components:

x	a scaled version of the x input.
y	a scaled version of the y input.
x.scale	The lower and upper bounds of the attributes in the scaled x.
y.scale	The lower and upper bounds of the scaled vector y.
x.bound	The lower and upper bounds of the attributes in the original x input matrix.
y.bound	The lower and upper bounds of the original y input vector.
r2	Scaling factor for the mean square error.

Author(s)

Chi Yau (based on svm in e1071 by David Meyer)
<chi.yau@r-tutor.com>

See Also

matrix.csr

Examples

```
## Not run:

library(rpud)
library(SparseM)

cadata1.path <- file.path(.path.package(package="rpud"), "runit/data/cadata-sample")
cadata1.data <- read.svm.data(cadata1.path, fac=FALSE)
cadata1.rpusvm <- rpusvm(cadata1.data$x, cadata1.data$y, type=type)
cadata1.fitted <- fitted(cadata1.rpusvm)

cadata1.scaled <- rpuScale(cadata1.data$x, cadata1.data$y)
cadata2.rpusvm <- rpusvm(cadata1.scaled$x, cadata1.scaled$y, type=type, scale=FALSE)
cadata2.fitted <- fitted(cadata2.rpusvm)

cadata1.yrange <- cadata1.scaled$y.bound[2]-cadata1.scaled$y.bound[1]
cadata1.yscale <- cadata1.scaled$y.scale[2]-cadata1.scaled$y.scale[1]

cadata2.fitted.scaled <-
        (cadata2.fitted - cadata1.scaled$y.scale[1])*cadata1.yrange/cadata1.yscale +
        cadata1.scaled$y.bound[1]
all.equal(cadata1.fitted, cadata2.fitted.scaled, tol=50)



## End(Not run)
```

---

rpuSetDevice *Select GPU for use by the current thread*

Description

Select the GPU of the device ID for use by the current thread.

Usage

```
rpuSetDevice(deviceId)
```

Arguments

deviceId	Device ID to be used by the current thread.

Value

Returns the active GPU of the current thread.

rpusvm *Support Vector Machines on GPU*

Description

The rpusvm method trains an SVM model.

In the non-free rpudplus add-on, the rpusvm method is implemented in NVIDIA CUDA, and assumes necessary double precision CUDA hardware support. The SVM model thus trained assumes ascending classification label ordering, and has an independent sigma coefficient for probabilistic regression. It also includes possible scaling parameters. Hence it is incompatible with the SVM model created by e1071.

Despite the incompatibility of the SVM models, rpusvm supports equivalent LIBSVM functionality in e1071.

This method is not supported in the free rpud package.

Usage

```
## S3 method for class 'formula'
rpusvm(formula, data = NULL, ..., subset, na.action =
na.omit, scale = TRUE, verbose = TRUE)
## Default S3 method:
rpusvm(x, y = NULL, scale = TRUE, type = NULL, kernel =
"radial", degree = 3, gamma = if (is.vector(x)) 1 else 1 / ncol(x),
coef0 = 0, cost = 1, nu = 0.5,
class.weights = NULL, cachesize = 100, tolerance = 0.001, epsilon = 0.1,
shrinking = TRUE, cross = 0, probability = FALSE, fitted = TRUE, seed = 0,
..., subset, na.action = na.omit, verbose = TRUE)
```

Arguments

formula	a symbolic description of the model to be fit.
data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which 'rpusvm' is called from.
x	a data matrix, a vector, or a sparse matrix (object of class Matrix provided by the **Matrix** package, or of class matrix.csr provided by the **SparseM** package.
y	a response vector with one label for each row/component of x. Can be either a factor (for classification tasks) or a numeric vector (for regression).
scale	It can be a numeric vector or a single logical value. If it is a numeric vector, the first and second vector elements will be the lower and upper bounds of each attributes in the x component after scaling. The third and fourth optional vector elements will be the lower and upper bounds of the y component after scaling. This is ignored if y is of factor type. If scale is a single logical TRUE value, the default scale vector c(0, 1, -1, 1) will be applied.

`type` rpusvm can be used as a classification machine, as a regression machine, or for novelty detection. Depending of whether y is a factor or not, the default setting for type is C-classification or eps-regression, respectively, but may be overwritten by setting an explicit value.
Valid options are:

- C-classification
- nu-classification
- one-classification (for novelty detection)
- eps-regression
- nu-regression

`kernel` the kernel used in training and predicting. You might consider changing some of the following parameters, depending on the kernel type.

linear: $u'v$
polynomial: $(\gamma u'v + c)^d$
radial basis: $exp(-\gamma|u - v|^2)$
sigmoid: $tanh(\gamma u'v + c)$

`degree` parameter needed for kernel of type polynomial (default: 3)

`gamma` parameter needed for all kernels except linear (default: 1/(data dimension))

`coef0` parameter needed for kernels of type polynomial and sigmoid (default: 0)

`cost` cost of constraints violation (default: 1)—it is the 'C'-constant of the regularization term in the Lagrange formulation.

`nu` parameter needed for nu-classification, nu-regression, and one-classification

`class.weights` a named vector of weights for the different classes, used for asymmetric class sizes. Not all factor levels have to be supplied (default weight: 1). All components have to be named.

`cachesize` set cache memory in MB (default 100), which maybe constrained by the GPU device memory in rpudplus

`tolerance` tolerance of termination criterion (default: 0.001)

`epsilon` epsilon in the insensitive-loss function (default: 0.1)

`shrinking` option whether to use the shrinking-heuristics (default: TRUE)

`cross` if a integer value k>0 is specified, a k-fold cross validation on the training data is performed to assess the quality of the model: the accuracy rate for classification and the Mean Squared Error for regression

`probability` logical indicating whether the model should allow for probability predictions.

`fitted` logical indicating whether the fitted values should be computed and included in the model or not (default: TRUE)

`seed` integer indicating the seed of random number generator used in cross-validation and probabilistic inference (default: 1)

`...` additional parameters for the low level fitting function rpusvm.default

`subset` An index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)

na.action	A function to specify the action to be taken if NAs are found. The default action is na.omit, which leads to rejection of cases with missing values on any required variable. An alternative is na.fail, which causes an error if NA cases are found. (NOTE: If given, this argument must be named.)
verbose	logical indicating whether progress information should be displayed. (default: TRUE)

Details

For multiclass-classification with k levels, k>2, libsvm uses the 'one-against-one'-approach, in which k(k-1)/2 binary classifiers are trained; the appropriate class is found by a voting scheme.

libsvm internally uses a sparse data representation, which is also high-level supported by the package **SparseM**.

If the predictor variables include factors, the formula interface must be used to get a correct model matrix.

plot.rpusvm allows a simple graphical visualization of classification models.

The probability model for classification fits a logistic distribution using maximum likelihood to the decision values of all binary classifiers, and computes the a-posteriori class probabilities for the multi-class problem using quadratic optimization. The probabilistic regression model assumes (zero-mean) laplace-distributed errors for the predictions, and estimates the scale parameter using maximum likelihood.

Value

An object of class "rpusvm" containing the fitted model, including:

SV	The resulting support vectors (possibly scaled).
index	The index of the resulting support vectors in the data matrix. Note that this index refers to the preprocessed data (after the possible effect of na.omit and subset)
coefs	The corresponding coefficients times the training labels.
rho	The negative intercept.
sigma	In case of a probabilistic regression model, the scale parameter of the hypothesized (zero-mean) laplace distribution estimated by maximum likelihood.
probA, probB	numeric vectors of length k(k-1)/2, k number of classes, containing the parameters of the logistic distributions fitted to the decision values of the binary classifiers (1 / (1 + exp(a x + b))).

Note

Data are scaled internally, usually yielding better results.

Parameters of SVM-models usually *must* be tuned to yield sensible results!

Author(s)

Chi Yau (based on R doc of svm in e1071 by David Meyer)
<chi.yau@r-tutor.com>

References

- Chih-Chung Chang and Chih-Jen Lin:
LIBSVM: a library for Support Vector Machines
http://www.csie.ntu.edu.tw/~cjlin/libsvm
- Chih-Chung Chang and Chih-Jen Lin:
LIBSVM: a library for Support Vector Machines
http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.ps.gz
- Rong-En Fan, Pai-Hsune Chen and Chih-Jen Lin:
Working Set Selection Using the Second Order Information for Training SVM
http://www.csie.ntu.edu.tw/~cjlin/papers/quadworkset.pdf
- Austin Carpenter:
CuSVM: A CUDA implementation of support vector classification and regression
http://patternsonascreen.net/cuSVMDesc.pdf

See Also

predict.rpusvm plot.rpusvm matrix.csr (in package **SparseM**)

Examples

```
## Not run:
library(rpud)

data(iris)
attach(iris)

## classification mode
# default with factor response:
model <- rpusvm(Species ~ ., data = iris)

# alternatively the traditional interface:
x <- subset(iris, select = -Species)
y <- Species
model <- rpusvm(x, y)

print(model)
summary(model)

# test with train data
pred <- predict(model, x)
# (same as:)
pred <- fitted(model)

# Check accuracy:
table(pred, y)

# compute decision values and probabilities:
pred <- predict(model, x, decision.values = TRUE)
attr(pred, "decision.values")[1:4,]
```

```
# visualize (classes by color, SV by crosses):
plot(cmdscale(dist(iris[,-5])),
     col = as.integer(iris[,5]),
     pch = c("o","+")[1:150 %in% model$index + 1])

## try regression mode on two dimensions

# create data
x <- seq(0.1, 5, by = 0.05)
y <- log(x) + rnorm(x, sd = 0.2)

# estimate model and predict input values
m   <- rpusvm(x, y)
new <- predict(m, x)

# visualize
plot(x, y)
points(x, log(x), col = 2)
points(x, new, col = 4)

## density-estimation

# create 2-dim. normal with rho=0:
X <- data.frame(a = rnorm(1000), b = rnorm(1000))
attach(X)

# traditional way:
m <- rpusvm(X, gamma = 0.1)

# formula interface:
m <- rpusvm(~., data = X, gamma = 0.1)
# or:
m <- rpusvm(~ a + b, gamma = 0.1)

# test:
newdata <- data.frame(a = c(0, 4), b = c(0, 4))
predict (m, newdata)

# visualize:
plot(X, col = 1:1000 %in% m$index + 1, xlim = c(-5,5), ylim=c(-5,5))
points(newdata, pch = "+", col = 2, cex = 5)

# weights: (example not particularly sensible)
i2 <- iris
levels(i2$Species)[3] <- "versicolor"
summary(i2$Species)
wts <- 100 / table(i2$Species)
wts
m <- rpusvm(Species ~ ., data = i2, class.weights = wts)

## End(Not run)
```

rvbm *GPU Accelerated Variational Bayesian Multiclass Probit Regression*

Description

The rvbm method trains a Bayesian classification model with Gaussian process priors. It is based on the variational sparse approximation technique used by Girolami and Rogers, as well as the vbmp package implemented by Lama and Girolami.

In the non-free rpudplus add-on, the rvbm method is implemented in NVIDIA CUDA, and assumes necessary double precision CUDA hardware support.

While there is no GPU-acceleration in the free rpud package, it includes optimized R code nonetheless.

Usage

```
## Default S3 method:
rvbm(X, t.class, X.TEST, t.class.TEST,
        theta = rep(1, ncol(X)), control = list())
```

Arguments

X	Feature matrix for the model to fit.
t.class	Response vector for the feature matrix.
X.TEST	Out-of-sample test matrix for prediction.
t.class.TEST	Out-of-sample response values.
theta	Covariance parameters for each feature dimension.
control	A list of control parameters including the following:

- InfoLevel: Trace level
- sFILE.TRACE: Path of trace file
- bThetaEstimate: Enable covariance parameter estimate
- sKernelType: Gaussian process kernel, default is gaussian:
  - linear: $u'v$
  - polynomial: $(u'v + c)^d$
  - cauchy: $(1 + |u - v|^2)^{-d}$
  - gaussian: $exp(-|u - v|^2)$
  - laplacian: $exp(-|u - v|)$
  - sigmoid: $tanh(u'v + c)$
- kDegree: coef0 parameter for polynomial and cauchy kernels
- kCoef0: degree parameter for polynomial and sigmoid kernels, usually 0 or 1
- maxIts: Maximum number of EM steps allowed (default 50).
- Thresh: Convergence threshold on marginal likelihood lowerbound (default 1e-4).

- method: Numerical integration method:
  - quadrature: Gaussian quadrature (default)
  - classic: Naive Monte Carlo
- nNodesQuad: Number of quadrature nodes (default 49).
- nSampsTG: Number of number of truncated Gaussian samples (default 1000).
- nSampsIS: Importance sampler population (default 1000).
- nSmallNo: Numerical underflow tolerance (default 1e-10).
- parGammaTau: Location of Gamma prior over covariance parameters (default 1e-6).
- parGammaSigma: Scale of Gamma prior over covariance parameters (default 1e-6).
- bMonitor: TRUE to collect monitor convergence diagnostics (default FALSE).
- bPlotFitting: TRUE to plot test performance results at each iteration (default FALSE).
- bInitM: Optionally initializes latent Gaussian process random variables. Set as TRUE to exactly match results from vbmp (default FALSE).

Details

Sharing a single covariance function across all classes.

Value

An S3 object of class "rvbm" with the following components:

Kc	Number of response categories
Ptest	Matrix of predicted posterior probability
X	Feature matrix of the Gaussian process model
cholPHI	Upper triangular Cholesky factor of the symmetric kernel matrix
Y	Matrix of auxiliary variables
M	Matrix of latent Gaussian process random variables
THETA	covariance kernel hyperparameters
sKernelType	Kernel type of the Gaussian process model
testErr	Percentage of posterior prediction error.
PL	Predictive log likelihood during each model fitting iteration.
lowerBound	Lower bound estimates of marginal likelihood during each model fitting iteration.

Author(s)

Chi Yau (based on the vbmp R doc by Lama and Girolami)
<chi.yau@r-tutor.com>

References

- Girolami M, Rogers S, *Variational Bayesian Multinomial Probit Regression with Gaussian Process Priors*, Neural Computation 18, 1790-1817 (2006).

- Lama N, Girolami M *vbmp: Variational Bayesian Multinomial Probit Regression for multi-class classification in R*, Bioinformatics 24(1):135-136 (2008). http://bioinformatics.oxfordjournals.org/cgi/content/short/btm535v1

See Also

predict.rvbm plot.rvbm rpusvm vbmp

Examples

```
## Not run:
library(rpud)

x <- rvbm.sample.train$X
y <- rvbm.sample.train$t.class
model.rvbm <- rvbm(
    x, y, x, y,
    theta = rep(1, ncol(x)),
    control = list(
            sKernelType="gaussian",
            bThetaEstimate=TRUE,
            bMonitor=TRUE,
            maxIts=12,
            InfoLevel=1)
)

summary(model.rvbm)


## End(Not run)
```

---

rvbm.sample.train *Example Data Sets for Variational Bayesian Multiclass Probit Regression*

Description

The sample data sets are for demonstration of the rvbm method.

Usage

```
rvbm.sample.train
rvbm.sample.test
```

Format

There are two components in each data set:

- X: Data points in a six dimensional Euclidean space. The first two coordinates are generated randomly in a 2-dimensional Euclidean space. The other four are noise dimensions.
- t.class: Classifies the data points of `X` into 3 different categories according to the squared sum of the first two coordinates of the data points.

Note

The data sets are created with the R code found in the `vbmp` vignette. The only difference between `rvbm.sample.train` and `rvbm.sample.test` are the number of data points being generated.

Examples

```
## Not run:
library(rpud)

x1 <- rvbm.sample.train$X[, 1]
x2 <- rvbm.sample.train$X[, 2]
tc <- rvbm.sample.train$t.class

plot(x1, x2, type="n", xlab="X1", ylab="X2")
points(x1[tc==1], x2[tc==1], type="p", col="blue", pch=19)
points(x1[tc==2], x2[tc==2], type="p", col="red")
points(x1[tc==3], x2[tc==3], type="p", col="green", pch=24)


## End(Not run)
```

---

summary.rvbm — *Summary Statistics of Variational Bayesian Multiclass Probit Regression*

Description

This method gives summary statistics of the Gaussian process model created with `rvbm`.

Usage

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

Arguments

`object`	An S3 object that inherits from the class `rvbm`.
`...`	Not used.

Value

An S3 object of type `summary.rvbm` that contains the following:

- Kc: Number of response categories.
- covParams: Estimated theta covariance parameters.
- predLik: Predictive log likelihood of the fitted model.
- predError: Percentage of the prediction errors.
- predClass: Predicted class membership of the test data.

Note

This replaces the `covParams`, `predLik`, `predError`, and `predClass` methods in `vbmp`.

Author(s)

Chi Yau
<chi.yau@r-tutor.com>

See Also

`rvbm`, vbmp

Examples

```
## Not run:
library(rpud)

x <- rvbm.sample.train$X
y <- rvbm.sample.train$t.class
model.rvbm <- rvbm(
    x, y, x, y,
    theta = rep(1, ncol(x)),
    control = list(
            sKernelType="gaussian",
            bThetaEstimate=TRUE,
            bMonitor=TRUE,
            maxIts=12,
            InfoLevel=1)
)

summary(model.rvbm)


## End(Not run)
```

write.rpusvm *SVM Model File Export*

Description

The wite.rpusvm method exports a SVM model created by rpusvm to a file.

In the non-free rpudplus add-on, the rpusvm method is implemented in NVIDIA CUDA, and assumes necessary double precision CUDA hardware support. The SVM model thus trained assumes ascending classification label ordering, and has an independent sigma coefficient for probabilistic regression. It also includes possible scaling parameters. Hence it is incompatible with the SVM model created by e1071.

Despite the incompatibility of the SVM models, rpusvm supports equivalent LIBSVM functionality in e1071.

This method is not supported in the free rpud package.

Usage

```
write.rpusvm(object, rpusvm.file = "Rdata.rpusvm")
```

Arguments

object	An S3 object that inherits from rpusvm class.
rpusvm.file	filename to export the rpusvm object to.

Details

Note: The SVM model files created by e1071 and rpusvm are *incompatible*.

Author(s)

Chi Yau (based on R doc of write.svm in e1071 by Tomomi TAKASHINA)
<chi.yau@r-tutor.com>

See Also

rpusvm

Examples

```
## Not run:
data(iris)
attach(iris)

## classification mode
# default with factor response:
model <- rpusvm (Species~., data=iris)
```

```
# export SVM object to file
write.rpusvm(model, rpusvm.file = "iris-classifier.rpusvm")


## End(Not run)
```

Index