Package 'GPUmatrix'

April 20, 2023

Type Package

Title Basic Linear Algebra with GPU

Version 0.1.0

Description Motivation: GPU power is a great resource for computational biology specifically in statistics and linear algebra. Unfortunately, very few packages connect R with the GPU and none of them are transparent enough to perform the computations on the GPU without substantial changes to the code. Most of them lack proper maintenance: several of the previous attempts were removed from the corresponding repositories. It would be desirable to have an R package, properly maintained, that exploits the use of the GPU with minimal changes in the existing code. Results: We have developed the 'GPUMatrix' package. 'GPUMatrix' mimics the behavior of the Matrix package and extends R to use the GPU for computations. It is easy to learn and very few changes in the code are required to work on the GPU. 'GPUMatrix' relies on either 'Tensorflow' or 'Torch' R packages to perform the GPU operations. Its vignette shows some toy examples on non-negative factorization and other factorization used in 'bioinformatics'.

Depends R (>= 4.1)

Imports stats, methods

Suggests torch, tensorflow, Matrix, matrixStats, float, MASS, knitr, rmarkdown

VignetteBuilder knitr

License Artistic-2.0

RoxygenNote 7.2.1

Encoding UTF-8

NeedsCompilation no

Author Cesar Lobato-Fernandez [aut, cre], Juan A.Ferrer-Bonsoms [aut], Angel Rubio [aut, ctb]

Maintainer Cesar Lobato-Fernandez <clobatofern@unav.es>

Repository CRAN

Date/Publication 2023-04-20 16:52:38 UTC

R topics documented:

`aperm` *Array Transposition*

Description

`aperm` transposes a GPUmatrix by permuting its dimensions and optionally resizing it.

`t` returns the transpose of a matrix of two dimensions.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow'
aperm(a,perm,...)
## S4 method for signature 'gpu.matrix.torch'
aperm(a,perm,...)
```

```
## S4 method for signature 'gpu.matrix.tensorflow'
t(x)
## S4 method for signature 'gpu.matrix.torch'
t(x)
```

Arguments

x,a	A gpu.matrix.tensorflow o gpu.matrix.torch to be transposed
perm	the subscript permutation vector, usually a permutation of the integers 1:n, where n is the number of dimensions of a.
...	potential further arguments of methods.

Value

It returns a transposed version of the gpu.matrix a, with subscripts permuted as indicated by the input perm. The output is a matrix array class object.

The resulting matrix can be easily converted to a gpu.matrix class object using the gpu.matrix or the as.gpu.matrix functions.

See Also

For more information: aperm, t, gpu.matrix, and as.gpu.matrix.

Examples

```
## Not run:

  #change the first with the second subscript.
  #this example corresponds to transposing the matrix
  a <- gpu.matrix(1:9,nrow=3,ncol=3)
  b <- aperm(a,perm = c(2,1))

  t(a) #transpose of a.


## End(Not run)
```

apply *Apply Functions over 'GPUmatrix' margins*

Description

This function mimics the 'base' function `'apply'` that returns a vector or array or list of values obtained by applying a function to margins of an array or matrix.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow'
apply(X, MARGIN, FUN, ..., simplify)
## S4 method for signature 'gpu.matrix.torch'
apply(X, MARGIN, FUN, ..., simplify)
```

Arguments

`X`	A gpu.matrix.tensorflow or a gpu.matrix.torch object
`MARGIN`	1 for rows and 2 for columns
`FUN`	function to be applied in the operation
`...`	general additional parameters. Optional arguments to FUN.
`simplify`	a logical indicating whether results should be simplified if possible.

Value

It returns the output corresponding to the to `FUN` in each iteraction to the function. For more information see the help of the base function apply.

See Also

For more information see: `apply`

Examples

```
if(installTorch){

  a <- gpu.matrix(rnorm(9),3,3)

  apply(a, 1, mean) #computes the mean of each row
  apply(a, 2, mean) #computes the mean of each column

}
```

as_methods *as_methods*

Description

Function 'as.matrix' attempts to turn its argument into a matrix.

Function 'as.list' attempts to turn its argument into a list.

Function 'as.numeric' attempts to turn its argument into a numeric.

Function 'as.vector' attempts to turn its argument into a vector.

Function 'is.numeric' is a general test of an object being interpretable as numbers

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow'
as.array(x,...)
## S4 method for signature 'gpu.matrix.torch'
as.array(x,...)
## S4 method for signature 'gpu.matrix.tensorflow'
as.list(x,...)
## S4 method for signature 'gpu.matrix.torch'
as.list(x,...)
## S4 method for signature 'gpu.matrix.tensorflow'
as.matrix(x,...)
## S4 method for signature 'gpu.matrix.torch'
as.matrix(x,...)
## S4 method for signature 'gpu.matrix.tensorflow'
as.numeric(x,...)
## S4 method for signature 'gpu.matrix.torch'
as.numeric(x,...)
## S4 method for signature 'gpu.matrix.tensorflow'
as.vector(x,mode)
## S4 method for signature 'gpu.matrix.torch'
as.vector(x,mode)
## S4 method for signature 'gpu.matrix.torch'
is.numeric(x)
## S4 method for signature 'gpu.matrix.tensorflow'
is.numeric(x)
```

Arguments

x	A gpu.matrix.tensorflow o gpu.matrix.torch object
...	(generalized) vectors or matrices. These can be given as named arguments.
mode	character string naming an atomic mode or "list" or "expression" or (except for vector) "any".

See Also

numeric, array, list, matrix,

---

cbind_rbind_methods *cbind_rbind_methods*

Description

Mimics the 'cbind' and 'rbind' functions. These functions take a sequence of vectors, matrices, data-frames and/or gpu.matrix matrices and combine by columns or rows, respectively.

Usage

```
## S4 method for signature 'ANY,gpu.matrix.tensorflow'
cbind2(x,y)
## S4 method for signature 'ANY,gpu.matrix.torch'
rbind2(x,y)
## S4 method for signature 'gpu.matrix.tensorflow,ANY'
cbind2(x,y,...)
## S4 method for signature 'gpu.matrix.torch,ANY'
rbind2(x,y)
```

Arguments

x	A gpu.matrix.tensorflow o gpu.matrix.torch object
y	A gpu.matrix.tensorflow o gpu.matrix.torch object or any other matrix class
...	(generalized) vectors or matrices. These can be given as named arguments.

See Also

cbind, and rbind

Examples

```
## Not run:

a <- gpu.matrix(1:9,nrow=3,ncol=3)

#add new row
newrow <- c(1,2,3)
a <- rbind2(a,newrow)
```

```
#add new column
newcolumn <- c(1,2,3,4)
a <- cbind(a,newcolumn)

#add new rows from other gpu.marix
b <- gpu.matrix(1:16,nrow=4,ncol=4)
d <- rbind(a,b)

#add new columns from other gpu.marix
b <- gpu.matrix(1:16,nrow=4,ncol=4)
d <- cbind(a,b)



## End(Not run)
```

`concatenate_gpu.matrix` *concatenate_gpu.matrix*

Description

Mimics `'c'` basic function: function which combines its arguments.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow'
c(x,...,recursive)

## S4 method for signature 'gpu.matrix.torch'
c(x,...,recursive)

## S4 method for signature 'numMatrixLike'
c(x,...,recursive)
```

Arguments

`x`	A gpu.matrix.tensorflow o gpu.matrix.torch object
`...`	objects to be concatenated.
`recursive`	logical. If recursive = TRUE, the function recursively descends through lists (and pairlists) combining all their elements into a vector.

Value

It will return a vector with the combined values. It is equivalent to working with a matrix of class matrix.

See Also

c

Examples

```
## Not run:

#add new value
a <- gpu.matrix(1:5,nrow=1,ncol=5)
c(a,3)

#add other vector
c(a,a)

#add value to a gpu.matrix
a <- gpu.matrix(1:9,nrow=3,ncol=3)
c(a,a)
#it will return a vector as in original c function.



## End(Not run)
```

---

cor_cov *Correlation, Variance and Covariance for 'GPUmatrix' object*

Description

'cov' and 'cor' compute the covariance and correlation of x and y if these are vectors. If x and y are matrices then the covariances (or correlations) between the columns of x and the columns of y are computed.

'cov2cor' scales a covariance matrix into the corresponding correlation matrix efficiently.

If x and y are object of GPUmatrix, then the corresponding operations are computed by the GPU instead of the CPU.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow,ANY,ANY,ANY'
cor(x,y)
## S4 method for signature 'gpu.matrix.tensorflow,ANY,missing,character'
cor(x,y,method)
## S4 method for signature 'gpu.matrix.tensorflow,missing,missing,character'
cor(x,y,method)
## S4 method for signature 'ANY,gpu.matrix.tensorflow,ANY,ANY'
cor(x,y)
## S4 method for signature 'gpu.matrix.tensorflow,missing,ANY,ANY'
cor(x,y)

## S4 method for signature 'gpu.matrix.torch,ANY,ANY,ANY'
cor(x,y)
## S4 method for signature 'gpu.matrix.torch,ANY,missing,character'
cor(x,y,method)
## S4 method for signature 'gpu.matrix.torch,missing,missing,character'
cor(x,y,method)
## S4 method for signature 'ANY,gpu.matrix.torch,ANY,ANY'
cor(x,y)
## S4 method for signature 'gpu.matrix.torch,missing,ANY,ANY'
cor(x,y,method)

## S4 method for signature 'gpu.matrix.tensorflow'
cov(x,y)
## S4 method for signature 'ANY,gpu.matrix.tensorflow'
cov(x,y)
## S4 method for signature 'gpu.matrix.tensorflow,ANY'
cov(x,y)
## S4 method for signature 'gpu.matrix.tensorflow,missing'
cov(x,y)


## S4 method for signature 'gpu.matrix.torch'
cov(x,y)
## S4 method for signature 'ANY,gpu.matrix.torch'
cov(x,y)
## S4 method for signature 'gpu.matrix.torch,ANY'
cov(x,y)
## S4 method for signature 'gpu.matrix.torch,missing'
cov(x,y)

## S4 method for signature 'gpu.matrix.tensorflow'
cov2cor(V)
## S4 method for signature 'gpu.matrix.torch'
cov2cor(V)
```

Arguments

x	a numeric vector, matrix, data.frame or gpu.matrix
y	NULL (default) or a vector, matrix, data frame or gpu.matrix with compatible dimensions to x.
method	a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman".
V	symmetric numeric gpu.matrix, usually positive definite such as a covariance matrix.

Value

cor computes the correlation between x and y.

cov computes the covariance between x and y.

cov2cor scales a covariance matrix into the corresponding correlation matrix efficiently.

See Also

For more information: cor, cov, cov2cor,

Examples

```
## Not run:
a <- gpu.matrix(1:10)
b <- gpu.matrix(11:20)
cor(a,b)

#example taken from stats corresponding help page:
longley_gpu <- gpu.matrix(longley,type="tensorflow")
C1 <- cor(longley_gpu)
cov(longley_gpu)
cov2cor(cov(longley_gpu))


## End(Not run)
```

density *Kernel Density Estimation*

Description

The function 'density' mimics the function 'density' of the library 'stats'. It computes kernel density estimates. Its default method does so with the given kernel and bandwidth for univariate observations.

The function 'hist' mimics the function 'hist' of the library 'graphics'. It computes a histogram of the given data values.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow'
density(x)
## S4 method for signature 'gpu.matrix.torch'
density(x)
## S4 method for signature 'gpu.matrix.tensorflow'
hist(x,...)
## S4 method for signature 'gpu.matrix.torch'
hist(x,...)
```

Arguments

x	the GPUmatrix object from which the estimate density is to be computed or the GPUmatrix object for which the histogram is desired.
...	further arguments and graphical parameters.

Value

Same output as in the base function density. It returns "an object with class 'density' whose underlying structure is a list containing the following components.

x	the n coordinates of the points where the density is estimated.
y	the estimated density values. These will be non-negative, but can be zero.
bw	the bandwidth used.
n	the sample size after elimination of missing values.
call	the call which produced the result.
data.name	the deparsed name of the x argument.
has.na	logical, for compatibility (always FALSE).

The print method reports summary values on the x and y components."

See Also

For more information see: density, and hist

Examples

```
if(installTorch){

  a <- gpu.matrix(rnorm(20*100),20,100)

  density(a[1,]) #density information
  plot(density(a[1,])) #plot the estimated density function

  hist(a[1,]) #plot the histogram

}
```

---

det *Calculate the Determinant of a 'GPUMatrix'*

Description

Mimic of the base functions 'det' and 'determinant': 'det' calculates the determinant of a gpu.matrix. 'determinant' is a generic function that returns separately the modulus of the determinant, optionally on the logarithm scale, and the sign of the determinant.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow,logical'
determinant(x,logarithm,...)
## S4 method for signature 'gpu.matrix.tensorflow,missing'
determinant(x,logarithm,...)
## S4 method for signature 'gpu.matrix.torch,logical'
determinant(x,logarithm,...)
## S4 method for signature 'gpu.matrix.torch,missing'
determinant(x,logarithm,...)

## S4 method for signature 'gpu.matrix.tensorflow'
det(x,...)
## S4 method for signature 'gpu.matrix.torch'
det(x,...)
```

Arguments

x	numeric gpu.matrix
...	extra generic parameters of gpu.matrix function
logarithm	logical; if TRUE (default) return the logarithm of the modulus of the determinant.

Value

`det` returns the same output corresponding to the base function 'det', which is the determinant of `x`.

`determinant` returns the corresponding output of the base function 'determinant', which is a list with components:

modulus	a numeric value. The modulus (absolute value) of the determinant if logarithm is FALSE; otherwise the logarithm of the modulus.
sign	integer; either +1 or -1 according to whether the determinant is positive or negative.

See Also

For more information see: `det`

Examples

```
## Not run:

x <- gpu.matrix(1:4,nrow=2, ncol = 2)
unlist(determinant(x)) #modulus of the determinant.
det(x)#the determinant.


## End(Not run)
```

---

diag *diag*

Description

It mimics the function `'diag'` that extractx or replacex the diagonal of a matrix, or constructs a diagonal matrix.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow'
diag(x)
## S4 method for signature 'gpu.matrix.torch'
diag(x)
## S4 replacement method for signature 'gpu.matrix.tensorflow,numeric'
diag(x) <- value
## S4 replacement method for signature 'gpu.matrix.torch,numeric'
diag(x) <- value
```

Arguments

x	A gpu.matrix
value	either a single value or a vector of length equal to that of the current diagonal.

Value

Output corresponding to the base function 'diag'. If input x is a GPUmatrix then diag{x} returns the diagonal of the matrix x in a vector. The output is not a GPUmatrix object.

The replacement form diag(x) <- value sets the diagonal of the matrix x to the given value(s).

See Also

For more information see:

diag

Examples

```
if(installTorch){

  a <- gpu.matrix(rnorm(9),nrow=3,ncol=3)

  diag(a) #shows the diagonal of matrix a

  diag(a) <- c(10,0,100) #set the diagonal of matrix a
  a

}
```

dim_and_names *Number of rows and columns and its corresponding names*

Description

The dim family of functions set or get the dimension of a 'GPUmatrix'.

The rownames family of functions set or get the corresponding names of rows and columns.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow'
rownames(x)
## S4 method for signature 'gpu.matrix.torch'
rownames(x)
## S4 method for signature 'gpu.matrix.tensorflow'
colnames(x)
## S4 method for signature 'gpu.matrix.torch'
colnames(x)
## S4 method for signature 'gpu.matrix.tensorflow'
dim(x)
## S4 method for signature 'gpu.matrix.torch'
dim(x)
## S4 method for signature 'gpu.matrix.tensorflow'
dimnames(x)
## S4 method for signature 'gpu.matrix.torch'
dimnames(x)
## S4 method for signature 'gpu.matrix.tensorflow'
length(x)
## S4 method for signature 'gpu.matrix.torch'
length(x)
## S4 method for signature 'gpu.matrix.tensorflow'
ncol(x)
## S4 method for signature 'gpu.matrix.torch'
ncol(x)
## S4 method for signature 'gpu.matrix.tensorflow'
nrow(x)
## S4 method for signature 'gpu.matrix.torch'
nrow(x)

## S4 replacement method for signature 'gpu.matrix.tensorflow,vector'
dim(x) <- value
## S4 replacement method for signature 'gpu.matrix.torch,vector'
dim(x) <- value
## S4 replacement method for signature 'gpu.matrix.tensorflow,vector'
dimnames(x) <- value
## S4 replacement method for signature 'gpu.matrix.torch,vector'
```

```
dimnames(x) <- value
```

Arguments

x	A gpu.matrix
value	For dim a numeric vector of length 2 with the number of rows and number of columns. For dimnames a character or numeric vector of length 2 with the names of the rows and names of the columns.

Value

rownames returns the names of the rows of a gpu.matrix.

colnames returns the names of the columns of a gpu.matrix.

dim returns the number of rows and columns of a gpu.matrix and dim <- sets the number of rows and columns of a gpu.matrix.

dimnames returns the names of the rows and columns of a gpu.matrix and dimnames <- sets the names of the rows and columns of a gpu.matrix.

length returns the length (ncol*nrow) of a gpu.matrix.

ncol returns the number of columns of a gpu.matrix.

nrow returns the number of rows of a gpu.matrix.

See Also

For more information: rownames, colnames, dim, dim<-, dimnames, dimnames<-, length, ncol, nrow.

Examples

```
## Not run:

a <- gpu.matrix(rnorm(9))

dim(a) <- c(3,3) #sets the number of rows and columns.
dim(a) #shows the number of rows and the number of columns
ncol(a) #shows the number of columns
nrow(a) #shows the number of rows
length(a) #shows the lenght of the matrix (nrow*ncol)


dimnames(a) <- list(c("r1","r2","r3"),c("c1","c2","c3")) #sets rows and column names
dimnames(a) #shows both the row and the col names

#these functions are equivalent to the following:
rownames(a) <- c("r1","r2","r3") #adds rownames to a.
colnames(a) <- c("c1","c2","c3") #adds colnames to a.
rownames(a) #shows rownames.
colnames(a) #shows colnames.
```

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

---

expmGPU *'GPUMatrix' Exponential.*

Description

It is a mimic of the function 'expm' of the library 'Matrix'. It computes the exponential of a 'GPUmatrix'.

Usage

```
expmGPU(x)
## S4 method for signature 'gpu.matrix.tensorflow'
expmGPU(x)
## S4 method for signature 'gpu.matrix.torch'
expmGPU(x)
```

Arguments

x	a gpu.matrix build with float numbers

Details

This function works with float numbers (either float32 or float64). If the data type of gpu.matrix is integer, this function will not work. An example is shown below.

Value

the matrix exponential of x as GPUmatrix tensorflow class.

See Also

See Also as expm.

Examples

```
  ## Not run:
  #build with a matrix that contains int number. It will  not work.
x <- gpu.matrix(1:9,nrow=3,ncol = 3,dtype = "int")
x
try(expmGPU(x))
#need to be float and not int32

#by default, gpu.matrix work with float64
```

```
x <- gpu.matrix(1:9,nrow=3,ncol = 3)
try(expmGPU(x))

## End(Not run)
```

---

extract_gpu.matrix *extract_gpu.matrix*

Description

Mimics the [,[<-, [[, and [[<- base functions. These operators act on 'GPUmatrix' matrices, vectors, matrices, arrays and lists.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow,missing'
e1 - e2
## S4 method for signature 'gpu.matrix.torch,missing'
e1 - e2
## S4 method for signature 'gpu.matrix.tensorflow,index,index'
x[i,j]
## S4 method for signature 'gpu.matrix.tensorflow,index,missing'
x[i,j,...,drop = TRUE]
## S4 method for signature 'gpu.matrix.tensorflow,matrix,missing'
x[i,j,...,drop = TRUE]
## S4 method for signature 'gpu.matrix.tensorflow,missing,index'
x[i,j]
## S4 method for signature 'gpu.matrix.torch,index,index'
x[i,j]
## S4 method for signature 'gpu.matrix.torch,index,missing'
x[i,j,...,drop = TRUE]
## S4 method for signature 'gpu.matrix.torch,matrix,missing'
x[i,j,...,drop = TRUE]
## S4 method for signature 'gpu.matrix.torch,missing,index'
x[i,j]
## S4 replacement method for signature 'gpu.matrix.tensorflow,index,index'
x[i,j] <- value
## S4 replacement method for signature 'gpu.matrix.tensorflow,index,missing'
x[i,j] <- value
## S4 replacement method for signature 'gpu.matrix.tensorflow,matrix,missing'
x[i,j] <- value
## S4 replacement method for signature 'gpu.matrix.tensorflow,missing,index'
x[i,j] <- value
## S4 replacement method for signature 'gpu.matrix.torch,index,index'
x[i,j] <- value
```

```
## S4 replacement method for signature 'gpu.matrix.torch,index,missing'
x[i,j] <- value
## S4 replacement method for signature 'gpu.matrix.torch,matrix,missing'
x[i,j] <- value
## S4 replacement method for signature 'gpu.matrix.torch,missing,index'
x[i,j] <- value
## S4 method for signature 'gpu.matrix.tensorflow,index'
x[[i,j,...]]
## S4 method for signature 'gpu.matrix.torch,index'
x[[i,j,...]]
## S4 replacement method for signature 'gpu.matrix.tensorflow,index'
x[[i]] <- value
## S4 replacement method for signature 'gpu.matrix.torch,index'
x[[i]] <- value
```

Arguments

e1	a gpu.matrix
e2	a gpu.matrix
x	a gpu.matrix from which extract element(s) or in which to replace element(s)
i,j,...	indices specifying elements to extract or replace.
value	typically an array-like R object of a similar class as x
drop	For matrices and arrays. If TRUE the result is coerced to the lowest possible dimension

See Also

See Also Extract.

Examples

```
## Not run:

a <- gpu.matrix(1:9,nrow=3,ncol=3)
rownames(a) <- c("R1","R2","R3")
colnames(a) <- c("C1","C2","C3")

#return
a[3,3] # the element row 3 and column 3
a[6] # the 6th element
a[1,] # the first row
a[c(1,2),] # the first and second row
a[c(1,1),] # the first row twice
a[,1] # the first column
a[,c(1,2)] # the first and second column
a[,c(1,1)] # the first column twice
```

```
#replace
a[3,3] <- 100 # replace the element 3,3
a[1,] <- c(1,2,1) # replace the first row
a[,2] <- c(0,0,0) # replace the second column
a[c(1,2),] <- matrix(1:6,nrow = 2) # replace the first and second row



## End(Not run)
```

`fft` *Fast Discrete Fourier Transform (FFT)*

Description

Computes the fft of the rows of a matrix. It is equivalent to the base R `apply(A,1,fft)`.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow'
fft(z)
## S4 method for signature 'gpu.matrix.torch'
fft(z)
```

Arguments

`z`	a real or complex gpu.matrix containing the values to be transformed.

Value

It returns the corresponding output of the function 'fft' of the library 'stats': the unnormalized univariate discrete Fourier transform of the sequence of values in input `z`. For more details check the corresponding help of 'fft' of the library 'stats'.

See Also

For more information see: `fft`

Examples

```
if(installTorch){

  x <- gpu.matrix(1:4)
  fft(x)

}
```

gpu.matrix *create and store a matrix in the GPU*

Description

Mimic the base 'matrix' function.

gpu.matrix creates a gpu.matrix.torch or gpu.matrix.tensorflow from input data. The created matrix is stored in the GPU. This function also mimics the function 'Matrix' of the library 'Matrix'.

Usage

```
gpu.matrix(data = NA, nrow = NULL, ncol = NULL, byrow = FALSE,
                        dimnames = NULL, dtype=NULL, sparse=NULL, colnames=c(),
                        rownames=c(),device="cuda", type="torch")

as.gpu.matrix(x,...)
## S4 method for signature 'ANY'
as.gpu.matrix(x,...)
```

Arguments

data,x	a scalar, vector or matrix (both matrix or Matrix class).
nrow	Number of rows of the matrix. By default the number of rows of data if data is a matrix or a Matrix.
ncol	Number of columns of the matrix. By default the number of columns of data if data is a matrix or a Matrix.
byrow	The same as function 'matrix': "logical. If FALSE (the default) the matrix is filled by columns, otherwise the matrix is filled by rows."
dimnames	The same as in function 'matrix': "A dimnames attribute for the matrix: NULL or a list of length 2 giving the row and column names respectively. An empty list is treated as NULL, and a list of length one as row names. The list can be named, and the list names will be used as names for the dimensions."
dtype	By default "float64". User can indicate "float32" or "int" for "int64".
sparse	The same as in function 'Matrix' of the library 'Matrix': "logical or NULL, specifying if the result should be sparse or not. By default, it is made sparse when more than half of the entries are 0."
colnames	A vector with the column names.
rownames	A vector with the row names.
type	If gpu.matrix is "torch"(default) or "tensorflow."
device	By default "cuda" (alternative "cpu" only with torch). Indicates de devide to load cuda.
...	additional arguments to be passed to or from methods.

Details

gpu.matrix. mimics the functions Matrix of the library Matrix and the base function matrix. If tensorflow and/or torch are properly installed then the gpu.matrix will be stored on the GPU. The example shows how to check this.

User can apply -using the same operator- to this matrix the basic functions that can be applied to a matrix of class 'matrix' or class 'Matrix'.

It can work also with sparse matrices as the library Matrix.

Value

Returns matrix of class 'gpu.matrix' of the library GPUmatrix that can be either "gpu.matrix.tensorflow" or ""gpu.matrix.torch". For both torch and tensorflow the functions to be applied to a matrix are the same.

If the gpu.matrix is not sparse it will show on the console the matrix as it is. If the gpu.matrix is sparse, it will return to the console the position where there are number different from zero. The internal values of the matrix can be seen using the operator "@".

Even if the matrix is sparse or not, both kind of matrices works equally with all functions.

Author(s)

Cesar Lobato and Angel Rubio.

See Also

See Matrix and matrix.

Examples

```
## Not run:
## create a gpu.matrix.torch and check it is stored in the GPU.
a <- gpu.matrix(1:9,nrow=3,ncol=3)
class(a)
a@gm$is_cuda

# the output of class(a) should be:
#[1] "gpu.matrix.torch"
#attr(,"package")
#[1] "GPUmatrix"

#the output of a@gm@device should have a similar shape:
#[1] TRUE

## create a gpu.matrix.torch and check it is stored in the CPU.
a <- gpu.matrix(1:9,nrow=3,ncol=3, device="cpu")
class(a)
a@gm$is_cuda
```

```
# the output of class(a) should be:
#[1] "gpu.matrix.torch"
#attr(,"package")
#[1] "GPUmatrix"

#the output of a@gm@device should have a similar shape:
#[1] FALSE

## create a gpu.matrix.tensorflow and check it is stored in the GPU.
a <- gpu.matrix(1:9,nrow=3,ncol=3,type="tensorflow")
class(a)
a@gm$device

# the output of class(a) should be:
#[1] "gpu.matrix.tensorflow"
#attr(,"package")
#[1] "GPUmatrix"

#the output of a@gm@device should have a similar shape:
#[1] "/job:localhost/replica:0/task:0/device:GPU:0"

## End(Not run)
```

`gpu.matrix-class` *Class 'gpu.matrix' for matrix stored in GPU*

Description

GPU computational power is a great resource for computational biology specifically in statistics and linear algebra. the gpu.matrix class is an object of the GPUmatrix package, that store a matrix in the GPU.

The GPUmatrix package is based on S4 objects in R and we have created a constructor function that acts similarly to the default `matrix` constructor in R for CPU matrices. The constructor function is `gpu.matrix` and accepts the same parameters as `matrix`.

Slots

rownames the row names of the gpu.matrix

colnames the colunm names of the gpu.matrix

gm the corresponding tensor

sparse Logical: indicates if the gpu.matrix is sparse or not

type If it is tensorflow or torch

See Also

See Also `gpu.matrix`.

`installTorch` *installTorch*

Description

This variable checks if you have properly installed torch package

Usage

```
installTorch
```

`kroneker` *kroneker*

Description

Kroneker product of two 'GPUmatrix' matrices. It mimics the base function `'kronecker'`.

Usage

```
## S4 method for signature 'ANY,gpu.matrix.tensorflow'
X %x% Y
## S4 method for signature 'ANY,gpu.matrix.torch'
X %x% Y
## S4 method for signature 'gpu.matrix.tensorflow,ANY'
X %x% Y
## S4 method for signature 'gpu.matrix.torch,ANY'
X %x% Y
```

Arguments

`X`	A gpu.matrix
`Y`	A gpu.matrix object or a matrix or a numeric variable

See Also

See Also `kronecker`.

Examples

```
## Not run:

a <- gpu.matrix(1:9,nrow=3,ncol=3)
a %x% diag(1,3)



## End(Not run)
```

---

matrix-product *Matrix Products*

Description

Mimics the base function for matrix multiplication. Given the matrices x and y as inputs, return a matrix standard product. The input x and y can be of class 'GPUmatrix'. Thus, the operation will be performed by the GPU.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow,ANY'
x %*% y
## S4 method for signature 'gpu.matrix.torch,ANY'
x %*% y

## S4 method for signature 'gpu.matrix.tensorflow,ANY'
crossprod(x, y,...)
## S4 method for signature 'gpu.matrix.tensorflow,missing'
crossprod(x, y = NULL,...)

## S4 method for signature 'gpu.matrix.tensorflow,ANY'
tcrossprod(x, y,...)
## S4 method for signature 'gpu.matrix.tensorflow,missing'
tcrossprod(x, y = NULL,...)


## S4 method for signature 'gpu.matrix.torch,ANY'
crossprod(x, y,...)
## S4 method for signature 'gpu.matrix.torch,missing'
crossprod(x, y = NULL,...)

## S4 method for signature 'gpu.matrix.torch,ANY'
```

```
tcrossprod(x, y,...)
## S4 method for signature 'gpu.matrix.torch,missing'
tcrossprod(x, y = NULL,...)
```

Arguments

x	a gpu.matrix
y	a gpu.matrix, matrix or Matrix object, or for `[t]crossprod()` `NULL` (by default); the latter case is formally equivalent to `y = x`.
...	potentially more arguments passed to and from methods.

Methods

%*% `signature(x = "gpu.matrix.tensorflow", y = "ANY")`: Matrix multiplication

crossprod `signature(x = "gpu.matrix.tensorflow", y = "ANY")`: Matrix multiplication

crossprod `signature(x = "gpu.matrix.tensorflow", y = "missing")`: Matrix multiplication

tcrossprod `signature(x = "gpu.matrix.tensorflow", y = "ANY")`: Matrix multiplication

tcrossprod `signature(x = "gpu.matrix.tensorflow", y = "missing")`: Matrix multiplication

%*% `signature(x = "gpu.matrix.torch", y = "ANY")`: Matrix multiplication

crossprod `signature(x = "gpu.matrix.torch", y = "ANY")`: Matrix multiplication

crossprod `signature(x = "gpu.matrix.torch", y = "missing")`: Matrix multiplication

tcrossprod `signature(x = "gpu.matrix.torch", y = "ANY")`: Matrix multiplication

tcrossprod `signature(x = "gpu.matrix.torch", y = "missing")`: Matrix multiplication

See Also

`tcrossprod` in R's base, and `crossprod` and `%*%`. **Matrix** package `%&%` for boolean matrix product methods.

Examples

```
## Not run:
  a <- gpu.matrix.tensorflow(as.numeric(1:9),nrow=3,ncol=3)
  b <- a%*%a

  b <- tcrossprod(a)

  b <- crossprod(a,a)


## End(Not run)
```

`matrix_decomposition` *Decomposition of a matrix with GPU*

Description

These functions compute different decompositions of a matrix using the GPU.

`'eigen'` mimics the base function `'eigen'` that computes the eigenvalues and eigenvectors of a numeric (double, integer, logical) or complex matrix.

`'svd'` mimics the base function `'svd'` that computes the singular-value decomposition of a rectangular matrix.

`'qr'` mimics the base function `'qr'` that computes the QR decomposition of a matrix.

`'chol'` mimics the base function `'chol'` that computes Compute the Cholesky factorization of a real symmetric positive-definite square matrix.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow'
eigen(x)
## S4 method for signature 'gpu.matrix.torch'
eigen(x)

## S4 method for signature 'gpu.matrix.tensorflow'
svd(x)
## S4 method for signature 'gpu.matrix.torch'
svd(x)

## S4 method for signature 'gpu.matrix.tensorflow'
chol(x)
## S4 method for signature 'gpu.matrix.torch'
chol(x)

## S4 method for signature 'gpu.matrix.tensorflow'
qr(x)
## S4 method for signature 'gpu.matrix.torch'
qr(x)
```

Arguments

x	a numeric or complex gpu.matrix whose spectral decomposition or whose SVD decomposition or whose QR decomposition or whose Cholesky factorization is to be computed. For the Cholesky factorization, x must be real symmetric positive-definite square gpu.matrix.

Value

The output of these functions correspond to their equivalent base functions:

`eigen` mimics the base function `eigen` that computes the eigenvalues and eigenvectors of a numeric (double, integer, logical) or complex matrix. It returns a list with the following items:

`values`	a vector with the `P` eigenvalues of `x`
`vectors`	the eigenvectors of `x`

`svd` mimics the base function `svd` that computes the singular-value decomposition of a rectangular matrix. It returns a list with the following items:

`d`	a vector containing the singular values of `x`
`u`	a matrix whose columns contain the left singular vectors of `x`
`v`	a matrix whose columns contain the right singular vectors of `x`

`qr` mimics the base function `qr` that computes the QR decomposition of a matrix. It returns a list with the following items:

`qr`	a matrix with the same dimensions as `x`. The upper triangle contains the **R** of the decomposition and the lower triangle contains information on the **Q** of the decomposition
`qraux`	a vector of length `ncol(x)` which contains additional information on **Q**.
`rank`	the rank of `x` as computed by the decomposition(*): always full rank in the LAPACK case.
`pivot`	information on the pivoting strategy used during the decomposition.

`chol` mimics the base function `chol` that computes Compute the Cholesky factorization of a real symmetric positive-definite square matrix. It returns a GPUmatrix object with The upper triangular factor of the Cholesky decomposition, i.e., the matrix R such that R'R = x.

See Also

For more information see:

`eigen`, `svd`, `chol`, `qr`, `solve`

Examples

```
## Not run:
a <- gpu.matrix(rnorm(9),3,3)
ein <- eigen(a) #eigenvalues and eigenvectors
svd_return <- svd(a) #svd of gpu.matrix a
qr_return <- qr(a) #qr decompostion

ata <- tcrossprod(a)
#ata is a real symmetric positive-definite square matrix.
chol(ata) #cholesky decomposition.

#for an example of how to use the cholesky decomposition see
```

```
#the corresponding help for the function chol_solve.


## End(Not run)
```

`matrix_general_operators_methods`
matrix_general_operators_methods

Description

'head' and 'tail' mimic the functions 'head' and 'tail' from 'utils'. They return the first or last parts of a gpu.matrix object.

The function 'show' mimics the function 'show' of 'methods'. It display the object, by printing, plotting or whatever suits its class.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow'
tail(x,...)
## S4 method for signature 'gpu.matrix.torch'
tail(x,...)
## S4 method for signature 'gpu.matrix.tensorflow'
show(object)
## S4 method for signature 'gpu.matrix.torch'
show(object)
## S4 method for signature 'gpu.matrix.tensorflow'
head(x,...)
## S4 method for signature 'gpu.matrix.torch'
head(x,...)
```

Arguments

x,object	a GPUmatrix object
...	arguments to be passed to or from other methods.

See Also

For more information see: head, tail, and show.

Examples

```
## Not run:

a <- gpu.matrix(rnorm(20*5),20,5)

head(a) #shows the first six row of every column
tail(a) #shows the las six row of every column

show(a) #show all the object
a #equivalente to apply function show.

## End(Not run)
```

`matrix_ranges` *Get different statistics for a matrix or get the same statistics for each row or column.*

Description

Functions to summarize the values of a matrix by rows (columns): maximum value, index of maximum value, minimum value, index of minimum value, mean,variance, sum of values, ranking of values.

These functions mimic the corresponding function of `'base'`, `'matrixStats'` and `'Matrix'` libraries. See also section for more details.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow'
rowMaxs(x)
## S4 method for signature 'gpu.matrix.torch'
rowMaxs(x)
## S4 method for signature 'gpu.matrix.tensorflow'
colMaxs(x)
## S4 method for signature 'gpu.matrix.torch'
colMaxs(x)
## S4 method for signature 'gpu.matrix.tensorflow'
max(x)
## S4 method for signature 'gpu.matrix.torch'
max(x)

## S4 method for signature 'gpu.matrix.tensorflow'
rowMins(x)
## S4 method for signature 'gpu.matrix.torch'
rowMins(x)
## S4 method for signature 'gpu.matrix.tensorflow'
```

```
colMins(x)
## S4 method for signature 'gpu.matrix.torch'
colMins(x)
## S4 method for signature 'gpu.matrix.tensorflow'
min(x)
## S4 method for signature 'gpu.matrix.torch'
min(x)

## S4 method for signature 'gpu.matrix.tensorflow'
rowMeans(x)
## S4 method for signature 'gpu.matrix.torch'
rowMeans(x)
## S4 method for signature 'gpu.matrix.tensorflow'
colMeans(x)
## S4 method for signature 'gpu.matrix.torch'
colMeans(x)
## S4 method for signature 'gpu.matrix.tensorflow'
mean(x)
## S4 method for signature 'gpu.matrix.torch'
mean(x)

## S4 method for signature 'gpu.matrix.tensorflow'
rowVars(x)
## S4 method for signature 'gpu.matrix.torch'
rowVars(x)
## S4 method for signature 'gpu.matrix.tensorflow'
colVars(x)
## S4 method for signature 'gpu.matrix.torch'
colVars(x)

## S4 method for signature 'gpu.matrix.tensorflow'
rowRanks(x)
## S4 method for signature 'gpu.matrix.torch'
rowRanks(x)
## S4 method for signature 'gpu.matrix.tensorflow'
colRanks(x)
## S4 method for signature 'gpu.matrix.torch'
colRanks(x)
## S4 method for signature 'gpu.matrix.tensorflow'
rankMatrix(x)
## S4 method for signature 'gpu.matrix.torch'
rankMatrix(x)

## S4 method for signature 'gpu.matrix.tensorflow'
rowSums(x)
## S4 method for signature 'gpu.matrix.torch'
rowSums(x)
## S4 method for signature 'gpu.matrix.tensorflow'
```

```
colSums(x)
## S4 method for signature 'gpu.matrix.torch'
colSums(x)
## S4 method for signature 'gpu.matrix.tensorflow'
sum(x)
## S4 method for signature 'gpu.matrix.torch'
sum(x)
```

Arguments

x	A gpu.matrix

Value

`max`, `rowMaxs`, `colMaxs` calculate de maximum value of the gpu.matrix, of each row and of each column respectively. `which.max` determines the location of the maximum value.

`min`, `rowMinss`, `colMins` calculate de minimum value of the gpu.matrix, of each row and of each column respectively. `which.min` determines the location of the minimum value.

`mean`, `rowMeans`, `colMeans` calculate the mean (average) value of the gpu.matrix, of each row and of each column respectively.

`rowVars`, `colVars` calculate the variance of each row and of each column of a gpu.matrix respectively.

`rowRanks`, `colRanks`, `rankMatrix` get the rank of the gpu.matrix, of each row and of each column respectively.

`rowSums`, `colSums`, `sum` sum the value of a gpu.matrix, of each row and of each column respectively.

See Also

For more information:

`rowMaxs`, `colMaxs`, `max`, `which.max`.

`rowMins`, `colMins`, `min`, `which.min`.

`rowMeans`, `colMeans`, `mean`.

`rowVars`, `colVars`.

`rowRanks`, `colRanks`, `rankMatrix`

`rowSums`, `colSums`, `sum`

outer *outer*

Description

Equivalent outer product for a GPUmatrix. In mimics the 'base' functions `'outer'` and `'%o%'`.

Usage

```
## S4 method for signature 'ANY,gpu.matrix.tensorflow'
X %o% Y
## S4 method for signature 'ANY,gpu.matrix.torch'
X %o% Y
## S4 method for signature 'gpu.matrix.tensorflow,ANY'
X %o% Y
## S4 method for signature 'gpu.matrix.torch,ANY'
X %o% Y

## S4 method for signature 'ANY,gpu.matrix.tensorflow'
outer(X,Y,FUN,...)
## S4 method for signature 'ANY,gpu.matrix.torch'
outer(X,Y,FUN,...)
## S4 method for signature 'gpu.matrix.tensorflow,ANY'
outer(X,Y,FUN,...)
## S4 method for signature 'gpu.matrix.torch,ANY'
outer(X,Y,FUN,...)
```

Arguments

`X,Y`	First and second arguments for function `FUN`. Tipicalle are a vector or an array. With this implementation can be also a gpu.matrix.
`FUN`	a function to use on the outer products.
`...`	Optional arguments to be passed to FUN.

See Also

See Also `outer`.

power_of_a_matrix *Compute the kth power of a matrix.*

Description

Comput the kth power of a squere matrix, i.e., multiply the matrix by itself as many times as user indicates.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow,numeric'
x %^% k
## S4 method for signature 'gpu.matrix.torch,numeric'
x %^% k
```

Arguments

x	a gpu.matrix.tensorflow or gput.matrix.torch
k	the power of the matrix

Details

x needs to be a squere matrix.

Value

the nth power of the input gpu.matrix. The returned matrix is also a gpu.matrix.

Examples

```
## Not run:

a <- gpu.matrix(1:9,nrow=3,ncol=3)
a %^% 5



## End(Not run)
```

round *rounding of numers*

Description

It mimics the base function 'round' that rounds the values in its first argument to the specified number of decimal places (default 0).

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow,ANY'
round(x)
## S4 method for signature 'gpu.matrix.torch,missing'
round(x,digits)
## S4 method for signature 'gpu.matrix.torch,numeric'
round(x,digits)
## S4 method for signature 'gpu.matrix.tensorflow,missing'
round(x,digits)
## S4 method for signature 'gpu.matrix.tensorflow,numeric'
round(x,digits)
```

Arguments

x	a numeric GPUmatrix object.
digits	integer indicating the number of decimal places (round) or significant digits (signif) to be used.

See Also

round

Examples

```
## Not run:

a <- gpu.matrix(rnorm(9),3,3)
round(a,digits = 3) #round to the third digit



## End(Not run)
```

`solve_gpu.matrix` *Solve a System of Equations*

Description

These functions are used to solve a system of equations or to compute the inverse of a matrix.

They mimic of the 'base' function `'solve'`: This generic function solves the equation ax=b for x, where b can be either a vector or a matrix (gpu.matrix).

`ginv` mimics the function `ginv` of package MASS: it calculates the Moore-Penrose generalized inverse of a gpu.matrix.

`chol_solve` function is a GPUmatrix own function. This function uses the Cholesky decomposition to solve a system of equations.

Usage

```
## S4 method for signature 'ANY,gpu.matrix.tensorflow'
solve(a,b)
## S4 method for signature 'ANY,gpu.matrix.torch'
solve(a,b)
## S4 method for signature 'gpu.matrix.tensorflow,ANY'
solve(a,b)
## S4 method for signature 'gpu.matrix.tensorflow,missing'
solve(a)
## S4 method for signature 'gpu.matrix.torch,ANY'
solve(a,b)
## S4 method for signature 'gpu.matrix.torch,missing'
solve(a)

## S4 method for signature 'gpu.matrix.torch'
ginv(X,tol)
## S4 method for signature 'gpu.matrix.tensorflow'
ginv(X,tol)

## S4 method for signature 'ANY,gpu.matrix.torch'
chol_solve(x,y)
## S4 method for signature 'ANY,gpu.matrix.tensorflow'
chol_solve(x,y)
## S4 method for signature 'gpu.matrix.torch,ANY'
chol_solve(x,y)
## S4 method for signature 'gpu.matrix.tensorflow,ANY'
chol_solve(x,y)
```

Arguments

a,x	A square numeric or complex gpu.matrix containing the coefficients of the linear system. gpu.matrix that are logical matrices are coerced to numeric. For chol_solve, x must be the cholesky cecomposition of matrix a if a real symmetric positive-definite square gpu.matrix
b,y	a numeric or complex vector or matrix giving the right-hand side(s) of the linear system. If missing, b is taken to be an identity matrix and solve will return the inverse of a.
X	Matrix for which the Moore-Penrose inverse is required.
tol	A relative tolerance to detect zero singular values.

See Also

See also `solve` and `ginv`.

For cholesky decomposition see `chol` from base or `matrix_decomposition` from GPUmatrix.

Examples

```
## Not run:

a <- gpu.matrix(rnorm(9),nrow=3,ncol=3)
inv <- solve(a) #the inverse matrix
a %*% inv

b <- c(1,1,1)
betas <- solve(a,b)
a %*% betas


#inverse using ginv
inv_2 <- GPUmatrix::ginv(a)
a %*% inv_2


#chol_solve: it can be applies only if
# in the equation Ax=b A is real symmetric positive-definite square matrix.
a <- gpu.matrix(rnorm(9),3,3)
A <- tcrossprod(a) #A is symmetrix positive-definite
b <- gpu.matrix(rnorm(3))

x_solve <- solve(A,b) #using solve to compare results
x_chol_solve <- chol_solve(chol(A),b) #using chol_solve
#NOTE: notice that the input for chol_solve is the Cholesky decomposition
# of matrix A.


## End(Not run)
```

sort_gpu.matrix *sort_gpu.matrix*

Description

Mimics the 'base' function 'sort'. Given a 'GPUmatrix' object (partially), this function sort it into ascending or descending order.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow,logical'
sort(x,decreasing)
## S4 method for signature 'gpu.matrix.tensorflow,missing'
sort(x,decreasing)
## S4 method for signature 'gpu.matrix.torch,logical'
sort(x,decreasing)
## S4 method for signature 'gpu.matrix.torch,missing'
sort(x,decreasing)
```

Arguments

x	A gpu.matrix.torch or a gpu.matrix.tensorflow object.
decreasing	Logical. Should the sort be increasing or decreasing?

Value

Returns a vector of class numeric. In order to store the resultin vector in the GPU use the function as.gpu.matrix or the function gpu.matrix.

See Also

sort and gpu.matrix

Examples

```
## Not run:
a <- gpu.matrix(rnorm(9),nrow=3,ncol=3)
sort(a) #returns a vector with the data sorted.

## End(Not run)
```

type of gpu.matrix *Spicify type of 'GPUmatrix'*

Description

'dtype' and 'dtype<-' are functions that show or set the number of bits to use to store the number. The possible options are "float64" for float64 (default), "float32" for float32 and "int" for int64. float64 uses 64 bits, that means that float64's take up twice as much memory thatn float32, thus doing operations on them may be slower in some machine architectures. However, float64's can represent numbers much more accurately than 32 bit floats. They also allow much larger numbers to be stored.

'to_dense' is a function that transforms a sparse matrix to a dense matrix. On the other hand, 'to_sparse' transforms a dense matrix to a sparse matrix.

Usage

```
## S4 method for signature 'gpu.matrix.torch'
to_dense(x)
## S4 method for signature 'gpu.matrix.tensorflow'
to_dense(x)
## S4 method for signature 'gpu.matrix.torch'
to_sparse(x)
## S4 method for signature 'gpu.matrix.tensorflow'
to_sparse(x)


## S4 method for signature 'gpu.matrix.torch'
dtype(x)
## S4 method for signature 'gpu.matrix.tensorflow'
dtype(x)
## S4 replacement method for signature 'gpu.matrix.torch'
dtype(x) <- value
## S4 replacement method for signature 'gpu.matrix.tensorflow'
dtype(x) <- value
```

Arguments

x	A gpu.matrix.torch or a gpu.matrix.tensorflow object.
value	type of gpu.matrix object

See Also

See also gpu.matrix.

Examples

```
## Not run:

a <- gpu.matrix(rnorm(9),3,3)

dtype(a) #bits used to store the numbers: it is float64 by default.

b <- a
dtype(b) <- "float32" #change to float32
b

b <- a
dtype(b) <- "int" #change to integer64 (int64)
b

#sparse or dense matrices
A <- gpu.matrix(data=c(1,1,1,0,0,1,0,1,0),3,3)
A #A is a dense gpu.matrix

A_sparse <- to_sparse(A) #transform A to a sparse matrix.
A_sparse #this matrix stores the where number different to 0 were placed.

to_dense(A_sparse) #transform A_sparse to a dense matrix and we obtain the orginal matrix A:
A


## End(Not run)
```

`xtfrm` *Auxiliary Function for Sorting and Ranking*

Description

Mimics `'xtfrm'` 'base' function.

Usage

```
## S4 method for signature 'gpu.matrix.tensorflow'
xtfrm(x)
## S4 method for signature 'gpu.matrix.torch'
xtfrm(x)
```

Arguments

x	A GPUmatrix object

See Also

xtfrm

Index