Package 'Rcublas'

May 2, 2016

Type Package

Title GPU enabled BLAS functions and random number generators

Version 0.1

Date 2016-1-07

Author Yuan Li, Hua Zhou

Maintainer Yuan Li <yli16@ncsu.edu>

Description This package provides GPU-accelerated algebra and random numbergenerator functions by wrapping CUDA library. It also includes some self-defined GPU algebra functions which are unavailable from CUDA library.

License BSD

URL https://github.ncsu.edu/yli16/R_CUBLAS

RoxygenNote 5.0.1

R topics documented:

`addGPU` *addGPU*

Description

This function computes the element-wise sum of two given vectors or matrices (x + y) by using CUDA cublas function cublasDgeam

Usage

```
addGPU(x, y)
```

Arguments

`x`	list consisting of R external GPU pointer and dimension
`y`	list consisting of R external GPU pointer and dimension

Value

sum of vectors or matrices (x + y), a list consisting of

- ptr: GPU pointer
- m: number of rows
- n: number of columns

See Also

`subtractGPU`

Examples

```
a <- 1:4
b <- 2:5
a_gpu <- createGPU(a)
b_gpu <- createGPU(b)
addGPU(a_gpu, b_gpu)->c_gpu
gatherGPU(c_gpu)
```

createGPU *createGPU*

Description

Create a GPU vector by copying from the input R vector

Usage

```
createGPU(input)
```

Arguments

input	R vector to be copied (numerical)

Details

This function creates a vector in GPU by calling the CUDA cudamalloc function, and then copys the values of vector from input R vector. The output of this function is a list consisting of the GPU pointer and vector length .

Value

a list consisting of

- ptr: GPU pointer
- m: vector length
- n: always 1 as vector

Note

output is a R external GPU pointer and can only be used in Rcublas functions

Author(s)

Yuan Li

See Also

gatherGPU

Examples

```
a <- 1:4
a_gpu <- createGPU(a)
gatherGPU(a_gpu)
```

createGPUmat *createGPUmat*

Description

Create a GPU matrix by copying from the input R vector

Usage

```
createGPUmat(input, nrow, ncol)
```

Arguments

input	R vector (matrix stored in column-major format) to be copied (numerical)
nrow	number of rows
ncol	number of columns

Details

This function creates a matrix in GPU by calling the CUDA cudamalloc function, and then copys the values of matrix from input R vector (matrix stored in column-major format). The output is a list consisting of the GPU pointer and matrix dimension

Value

a list consisting of

- ptr: GPU pointer
- m: number of rows
- n: number of columns

Note

output is a R external GPU pointer and can only be used in Rcublas functions

Author(s)

Yuan Li

See Also

gatherGPU createGPU

Examples

```
a <- 1:4
am_gpu <- createGPUmat(a, 2, 2)
gatherGPU(am_gpu)
```

`divideGPU` *divideGPU*

Description

This function computes the element-wise division of two given vectors or matrices (x / y) by using CUDA function

Usage

```
divideGPU(x, y)
```

Arguments

`x`	list consisting of R external GPU pointer and dimension
`y`	list consisting of R external GPU pointer and dimension

Value

element-wise division of vectors or matrices (x / y), a list consisting of

- ptr: GPU pointer
- m: number of rows
- n: number of columns

See Also

`multiplyGPU`

Examples

```
a <- 1:4
b <- 2:5
a_gpu <- createGPU(a)
b_gpu <- createGPU(b)
divideGPU(a_gpu, b_gpu)->c_gpu
gatherGPU(c_gpu)
```

dotGPU *dotGPU*

Description

This function computes the dot product of two given vectors by using CUDA cublas function cublasDdot

Usage

```
dotGPU(x, y)
```

Arguments

x	list consisting of R external GPU pointer and dimension
y	list consisting of R external GPU pointer and dimension

Value

the resulting dot product

See Also

norm2GPU

Examples

```
a <- 1:4
b <- 2:5
a_gpu <- createGPU(a)
b_gpu <- createGPU(b)
dotGPU(a_gpu, b_gpu)
```

expGPU *expGPU*

Description

This function computes the exponential of given vector or matrix by using CUDA function

Usage

```
expGPU(input)
```

Arguments

input	list consisting of R external GPU pointer and dimension

Value

exponential vector, a list consisting of

- ptr: GPU pointer
- m: number of rows
- n: number of columns

See Also

`scaleGPU`

Examples

```
a <- 1:4
a_gpu <- createGPU(a)
expGPU(a_gpu)->b_gpu
gatherGPU(b_gpu)
```

---

gatherGPU *gatherGPU*

Description

Copy GPU vector to R vector

Usage

```
gatherGPU(input)
```

Arguments

`input`	list consisting of R external GPU pointer and dimension

Details

This function copys GPU vector/matrix to R vector

Value

R vector

Note

output is R vector and can be used by any R functions

Author(s)

Yuan Li

See Also

gatherGPU createGPU

Examples

```
a <- 1:4
am_gpu <- createGPUmat(a, 2, 2)
gatherGPU(am_gpu)
```

GPUobject *GPUobject*

Description

classify the input as GPU vector and assign its dimension This function classifies the input object as GPU vector and assign its dimension. The output of this function is a list consisting of the GPU pointer and its dimension.

Usage

```
GPUobject(input, length1, length2)
```

Arguments

input	R external pointer
length1	vector length
length2	always 1 as vector

Value

a list consisting of

- ptr: GPU pointer
- m: vector length
- n: always 1 as vector

Note

output is a R external GPU pointer and can only be used in Rcublas functions

Author(s)

Yuan Li

See Also

gatherGPU

gpuQuery *gpuQuery*

Description

This function returns the information of available GPU device in system

Usage

```
gpuQuery()
```

See Also

createGPU

Examples

```
gpuQuery()
```

inverseGPU *inverseGPU*

Description

This function computes the inverse of given matrix (square) by using CUDA cublas function cublasDgetrfBatched and cublasDgetriBatched (LU decomposition)

Usage

```
inverseGPU(X)
```

Arguments

X	input matrix; list of R external GPU pointer and dimension

Value

matrix inverse, a list consisting of

- ptr: GPU pointer
- m: matrix X's number of rows
- n: matrix X's number of columns

See Also

mmGPU createGPUmat

Examples

```
a <- 1:9
a_gpu <- createGPUmat(a,3,3)
inverseGPU(a_gpu)->c_gpu
gatherGPU(c_gpu)
```

`logGPU` *logGPU*

Description

This function computes the natural logarithms of given vector or matrix by using CUDA function

Usage

```
logGPU(input)
```

Arguments

`input`	list consisting of R external GPU pointer and dimension

Value

natural logarithms vector, a list consisting of

- ptr: GPU pointer
- m: number of rows
- n: number of columns

See Also

`expGPU`

Examples

```
a <- 1:4
a_gpu <- createGPU(a)
logGPU(a_gpu)->b_gpu
gatherGPU(b_gpu)
```

`lognormRNGGPU` *log normal random number generator*

Description

This function generates log-normally distributed numbers by using CUDA curand function CURAND_RNG_PSEUDO_DEFAULT and curandGenerateLogNormalDouble

Usage

```
lognormRNGGPU(n, mean = 0, sd = 1, seed = 1)
```

Arguments

`n`	number of random numbers
`mean`	mean of log-normal distribution; default value 0
`sd`	standard deviation of log-normal distribution; default value 1
`seed`	random number generator seed

Value

random numbers vector, a list consisting of

- ptr: GPU pointer
- m: matrix X's number of rows
- n: matrix X's number of columns

See Also

`normRNGGPU`

Examples

```
a_gpu <- lognormRNGGPU(100, 0, 1, 15)
gatherGPU(a_gpu)
```

maxGPU *maxGPU*

Description

finds the (smallest) index of the element with the maximum magnitude of given vector This function finds the (smallest) index of the element with the maximum magnitude of given vector by using CUDA cublas function cublasIdamin

Usage

```
maxGPU(input)
```

Arguments

input	list consisting of R external GPU pointer and dimension

Value

the resulting index

See Also

minGPU

Examples

```
a <- 1:4
a_gpu <- createGPU(a)
maxGPU(a_gpu)
```

---

minGPU *minGPU*

Description

finds the (smallest) index of the element with the minimum magnitude of given vector This function finds the (smallest) index of the element with the minimum magnitude of given vector by using CUDA cublas function cublasIdamin

Usage

```
minGPU(input)
```

Arguments

input	list consisting of R external GPU pointer and dimension

Value

the resulting index

See Also

`maxGPU`

Examples

```
a <- 1:4
a_gpu <- createGPU(a)
minGPU(a_gpu)
```

`mmGPU` *mmGPU*

Description

This function computes the matrix-matrix multiplication (X * Y) by using CUDA cublas function cublasDgemm

Usage

```
mmGPU(X, Y)
```

Arguments

`X`	input matrix; list of R external GPU pointer and dimension
`Y`	input matrix; list of R external GPU pointer and dimension

Value

matrix-matrix multiplication (X * Y), a list consisting of

- ptr: GPU pointer
- m: matrix X's number of rows
- n: matrix Y's number of columns

See Also

`mmGPU` `createGPUmat`

Examples

```
a <- 1:6
b <- 2:7
a_gpu <- createGPUmat(a,2,3)
b_gpu <- createGPUmat(b,3,2)
mmGPU(a_gpu, b_gpu)->c_gpu
gatherGPU(c_gpu)
```

multiplyGPU *multiplyGPU*

Description

This function computes the element-wise multiplication of two given vectors or matrices (x * y) by using CUDA cublas function cublasDdgmm

Usage

```
multiplyGPU(x, y)
```

Arguments

x	list consisting of R external GPU pointer and dimension
y	list consisting of R external GPU pointer and dimension

Value

element-wise multiplication of vectors or matrices (x * y), a list consisting of

- ptr: GPU pointer
- m: number of rows
- n: number of columns

See Also

divideGPU

Examples

```
a <- 1:4
b <- 2:5
a_gpu <- createGPU(a)
b_gpu <- createGPU(b)
multiplyGPU(a_gpu, b_gpu)->c_gpu
gatherGPU(c_gpu)
```

mvGPU *mvGPU*

Description

This function computes the matrix-vector multiplication (X * y) by using CUDA cublas function cublasDgemv

Usage

```
mvGPU(X, y)
```

Arguments

X	input matrix; list of R external GPU pointer and dimension
y	input vector; list of R external GPU pointer and dimension

Value

matrix-vector multiplication (X * y), a list consisting of

- ptr: GPU pointer
- m: matrix X's number of rows
- n: matrix X's number of columns; vector y's number of elements

See Also

mmGPU createGPUmat

Examples

```
a <- 1:4
b <- 2:3
a_gpu <- createGPUmat(a,2,2)
b_gpu <- createGPU(b)
mvGPU(a_gpu, b_gpu)->c_gpu
gatherGPU(c_gpu)
```

norm2GPU *norm2GPU*

Description

Compute the Euclidean norm of given vector

Usage

```
norm2GPU(input)
```

Arguments

input	list consisting of R external GPU pointer and dimension

Details

This function computes Euclidean norm of given vector by using CUDA cublas function cublasDnrm2

Value

vector Euclidean norm, a non-negative number.

Author(s)

Yuan Li

See Also

gatherGPU createGPU

Examples

```
a <- 1:4
a_gpu <- createGPU(a)
norm2GPU(a_gpu)
```

normRNGGPU *normal random number generator*

Description

This function generates normally distributed numbers by using CUDA curand function CURAND_RNG_PSEUDO_DEFAUI and curandGenerateNormalDouble

Usage

```
normRNGGPU(n, mean = 0, sd = 1, seed = 1)
```

Arguments

n	number of random numbers
mean	mean of normal distribution; default value 0
sd	standard deviation of normal distribution; default value 1
seed	random number generator seed

Value

random numbers vector, a list consisting of

- ptr: GPU pointer
- m: matrix X's number of rows
- n: matrix X's number of columns

See Also

lognormRNGGPU

Examples

```
a_gpu <- normRNGGPU(100, 0, 1, 15)
gatherGPU(a_gpu)
```

`poissonRNGGPU` *Poisson random number generator*

Description

This function generates Poisson distributed numbers by using CUDA curand function CURAND_RNG_PSEUDO_DEFAULT and curandGeneratePoisson

Usage

```
poissonRNGGPU(n, lambda = 1, seed = 1)
```

Arguments

`n`	number of random numbers
`seed`	random number generator seed
`mean`	mean of Poisson distribution; default value 1

Value

random numbers vector, a list consisting of

- ptr: GPU pointer
- m: matrix X's number of rows
- n: matrix X's number of columns

See Also

`normRNGGPU`

Examples

```
a_gpu <- poissonRNGGPU(100, 1)
gatherGPU(a_gpu)
```

`powerGPU` *powerGPU*

Description

This function computes the power of given vector or matrix by using CUDA function

Usage

```
powerGPU(input, alpha)
```

Arguments

`input`	list consisting of R external GPU pointer and dimension
`alpha`	power factor

Value

powered vector, a list consisting of

- ptr: GPU pointer
- m: number of rows
- n: number of columns

See Also

`sqrtGPU`

Examples

```
a <- 1:4
b <- 2
a_gpu <- createGPU(a)
powerGPU(a_gpu, b)->b_gpu
gatherGPU(b_gpu)
```

`Rcublas` *GPU enabled BLAS functions and random number generators*

Description

This package provides GPU-accelerated algebra and random numbergenerator functions by wrapping CUDA library. It also includes some self-defined GPU algebra functions which are unavailable from CUDA library.

Details

Package:	Rcublas
Type:	Package
Version:	1.0
Date:	2016-01-07
License:	BSD

~~ An overview of how to use the package, including the most important ~~ ~~ functions ~~

Author(s)

Yuan Li, Hua Zhou

Maintainer: Yuan Li <yli16@ncsu.edu>

`scaleGPU` *scaleGPU*

Description

This function scales the given vector by a scalar by using CUDA cublas function cublasDcopy

Usage

```
scaleGPU(input, alpha)
```

Arguments

`input`	list consisting of R external GPU pointer and dimension
`alpha`	scale factor

Value

scaled vector, a list consisting of

- ptr: GPU pointer
- m: number of rows
- n: number of columns

See Also

`exp_cuda`

Examples

```
a <- 1:4
b <- 2
a_gpu <- createGPU(a)
scaleGPU(a_gpu, b)->b_gpu
gatherGPU(b_gpu)
```

sqrtGPU *sqrtGPU*

Description

This function computes the square root of given vector or matrix by using CUDA function

Usage

```
sqrtGPU(input)
```

Arguments

input	(non-negative vector) list consisting of R external GPU pointer and dimension

Value

square root vector, a list consisting of

- ptr: GPU pointer
- m: number of rows
- n: number of columns

See Also

expGPU

Examples

```
a <- 1:4
a_gpu <- createGPU(a)
sqrtGPU(a_gpu)->b_gpu
gatherGPU(b_gpu)
```

subtractGPU *subtractGPU*

Description

This function computes the element-wise subtraction of two given vectors or matrices (x - y) by using CUDA cublas function cublasDgeam

Usage

```
subtractGPU(a, b)
```

Arguments

x	list consisting of R external GPU pointer and dimension
y	list consisting of R external GPU pointer and dimension

Value

subtraction of vectors or matrices (x - y), a list consisting of

- ptr: GPU pointer
- m: number of rows
- n: number of columns

See Also

addGPU

Examples

```
a <- 1:4
b <- 2:5
a_gpu <- createGPU(a)
b_gpu <- createGPU(b)
subtractGPU(a_gpu, b_gpu)->c_gpu
gatherGPU(c_gpu)
```

---

sumGPU *sumGPU*

Description

Compute the sum of given vector

Usage

```
sumGPU(x)
```

Arguments

x	list consisting of R external GPU pointer and dimension

Details

This function computes sum of given vector by using CUDA vector reduction

Value

vector sum

Author(s)

Yuan Li

See Also

`gatherGPU` `createGPU`

Examples

```
a <- createGPU(1:4)
sumGPU(a)
```

`tGPU` *tGPU*

Description

This function transposes the given matrix by using CUDA cublas cublasDgeam

Usage

```
tGPU(X)
```

Arguments

`X`	input matrix; list of R external GPU pointer and dimension

Value

matrix transpose, a list consisting of

- ptr: GPU pointer
- m: matrix X's number of rows
- n: matrix X's number of columns

See Also

`createGPUmat`

Examples

```
a <- 1:12
a_gpu <- createGPUmat(a,3,4)
tGPU(a_gpu)->c_gpu
gatherGPU(c_gpu)
```

`uniformRNGGPU` *standard uniform random number generator*

Description

This function generates uniformly distributed numbers between 0 and 1 by using CUDA curand function CURAND_RNG_PSEUDO_DEFAULT and curandGenerateUniformDouble

Usage

```
uniformRNGGPU(n, seed = 1)
```

Arguments

`n`	number of random numbers
`seed`	random number generator seed

Value

random numbers vector, a list consisting of

- ptr: GPU pointer
- m: matrix X's number of rows
- n: matrix X's number of columns

See Also

`createGPU` `createGPUmat`

Examples

```
a_gpu <- uniformRNGGPU(100, 15)
gatherGPU(a_gpu)
```