Package 'kmcudaR'

May 3, 2017

Type Package

Title 'Yinyang' K-Means and K-NN using NVIDIA CUDA

Version 1.0.0

Date 2017-04-21

Author Vadim Markovtsev, Charles Determan

Maintainer Charles Determan <cdetermanjr@gmail.com>

Description K-means implementation is based on ``Yinyang K-Means: A Drop-In Replacement of the Classic K-Means with Consistent Speedup". While it introduces some overhead and many conditional clauses which are bad for CUDA, it still shows 1.6-2x speedup against the Lloyd algorithm. K-nearest neighbors employ the same triangle inequality idea and require precalculated centroids and cluster assignments, similar to the flattened ball tree.

License MIT + file LICENSE

Depends R (>= 3.3.2)

Imports Rcpp (>= 0.12.9)

LinkingTo Rcpp, RcppEigen

Suggests testthat

RoxygenNote 5.0.1.9000

SystemRequirements CUDA 8.0 tookit, OpenMP 4.0 capable compiler

OS_type unix

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-05-03 15:51:52 UTC

R topics documented:

kmeans_cuda *K-Means Clustering using CUDA*

Description

Performs k-means clustering on a numeric matrix using a NVIDIA GPU via CUDA

Usage

```
kmeans_cuda(samples, clusters, tolerance = 0.01, init = "k-means++",
  yinyang_t = 0.1, metric = "L2", average_distance = FALSE, seed = NULL,
  device = 0L, verbosity = 0L)
```

Arguments

samples	A numeric matrix
clusters	the number of clusters
tolerance	if the relative number of reassignments drops below this value the algorithm stops
init	A character vector or numeric matrix, sets the method for centroids initialization. Options include "k-means++", "afk-mc2", "random" or numeric matrix of shape [clusters, number of features]. Default = "kmeans++"
yinyang_t	numeric value defining relative number of cluster groups. Usually 0.1 but 0 disables Yinyang refinement.
metric	Character vector specifying distance metric to use. The default is Euclidean (L2), it can be changed to "cos" for Sphereical K-means with angular distance. NOTE - the samples must be normalized in the latter case.
average_distance	logical indicating whether to calculate the average distance between cluster elements and the corresponding centroids. Useful for finding the best 'K'. Returned as third list element
seed	random generator seed for reproducible results [deprecated]
device	integer defining device to use. 1 = first device, 2 = second device, 3 = first & second devices, 0 = use all devices. Default = 0
verbosity	Integer indicating amount of output to see. 0 = silence, 1 = progress logging, 2 = all output

Value

a list consisting of

centroids	Cluster centroids
assignments	integer vector of sample-cluster associations
average_distance	average distance between cluster elements

knn_cuda *K-Nearest Neighbor Classification using CUDA*

Description

k-nearest neighbor classification using a NVIDIA GPU via CUDA backend

Usage

```
knn_cuda(k, samples, centroids, assignments, metric = "L2", device = 0,
  verbosity = 0)
```

Arguments

k	The number of neighbors to search for each sample
samples	Numeric matrix
centroids	Numeric matrix with precalculated clusters' centroids
assignments	integer vector with sample-cluster associations. Indices start from 1.
metric	character name of the distance metric to use. The default is Euclidean (L2), it can be changed to "cos" for Sphereical K-means with angular distance. NOTE - the samples must be normalized in the latter case.
device	integer defining device to use. 1 = first device, 2 = second device, 3 = first & second devices, 0 = use all devices. Default = 0
verbosity	Integer indicating amount of output to see. 0 = silence, 1 = progress logging, 2 = all output

Value

Integer matrix with neighbor indices of shape [nsamp, k].

Index