Package ‘RcppHNSW’

September 20, 2019

Title  'Rcpp' Bindings for 'hnswlib', a Library for Approximate Nearest Neighbors
Version  0.2.0
Description  'Hnswlib' is a C++ library for Approximate Nearest Neighbors. This package provides a minimal R interface by relying on the 'Rcpp' package. See <https://github.com/nmslib/hnswlib> for more on 'hnswlib'. 'hnswlib' is released under Version 2.0 of the Apache License.
License  GPL-3
URL  https://github.com/jlmelville/rcpphnsw
BugReports  https://github.com/jlmelville/rcpphnsw/issues
Encoding  UTF-8
LazyData  true
Imports  methods, Rcpp (>= 0.11.3)
LinkingTo  Rcpp
RoxygenNote  6.1.1
Suggests  testthat, covr
NeedsCompilation  yes
Author  James Melville [aut, cre],
        Aaron Lun [ctb]
Maintainer  James Melville <jlmelville@gmail.com>
Repository  CRAN
Date/Publication  2019-09-20 16:40:12 UTC

R topics documented:

RcppHnsw-package ........................................................................... 2
hnsw_build .................................................................................. 2
hnsw_knn ...................................................................................... 3
hnsw_search .................................................................................. 5

Index  6
RcppHnsw-package

Rcpp bindings for the hnswlib C++ library for approximate nearest neighbors.

Description

hnswlib is a library implementing the Hierarchical Navigable Small World method for approximate nearest neighbor search.

Details

Details about hnswlib are available at the reference listed below.

Author(s)

James Melville for the R interface; Yury Malkov for hnswlib itself.
Maintainer: James Melville <jmelville@gmail.com>

References

https://github.com/nmslib/hnsw

hnsw_build

Build an hnswlib nearest neighbor index

Description

Build an hnswlib nearest neighbor index

Usage

hnsw_build(X, distance = "euclidean", M = 16, ef = 200, verbose = FALSE)

Arguments

X

distance

a numeric matrix of data to add. Each of the n rows is an item in the index.
Type of distance to calculate. One of:
- "l2" Squared L2, i.e. squared Euclidean.
- "euclidean" Euclidean.
- "cosine" Cosine.
hnsw_knn

- "ip" Inner product: 1 - sum(ai * bi), i.e. the cosine distance where the vectors are not normalized. This can lead to negative distances and other non-metric behavior.

M Controls the number of bi-directional links created for each element during index construction. Higher values lead to better results at the expense of memory consumption. Typical values are 2 - 100, but for most datasets a range of 12 - 48 is suitable. Can't be smaller than 2.

ef Size of the dynamic list used during construction. A larger value means a better quality index, but increases build time. Should be an integer value between 1 and the size of the dataset.

verbose If TRUE, log progress to the console.

Value
an instance of a HnswL2, HnswCosine or HnswIp class.

Examples

```r
irism <- as.matrix(iris[, -5])
ann <- hnsw_build(irism)
iris_nn <- hnsw_search(irism, ann, k = 5)
```

Description
A k-nearest neighbor algorithm using the hnswlib library (https://github.com/nmslib/hnswlib).

Usage

```r
hnsw_knn(X, k = 10, distance = "euclidean", M = 16,
     ef_construction = 200, ef = 10, verbose = FALSE)
```

Arguments

- X a numeric matrix of data to add. Each of the n rows is an item in the index.
- k Number of neighbors to return.
- distance Type of distance to calculate. One of:
  - "l2" Squared L2, i.e. squared Euclidean.
  - "euclidean" Euclidean.
  - "cosine" Cosine.
  - "ip" Inner product: 1 - sum(ai * bi), i.e. the cosine distance where the vectors are not normalized. This can lead to negative distances and other non-metric behavior.
hnsw_knn

\( M \)
Controls the number of bi-directional links created for each element during index construction. Higher values lead to better results at the expense of memory consumption. Typical values are \( 2 \sim 100 \), but for most datasets a range of \( 12 \sim 48 \) is suitable. Can’t be smaller than 2.

\( \text{ef\_construction} \)
Size of the dynamic list used during construction. A larger value means a better quality index, but increases build time. Should be an integer value between 1 and the size of the dataset.

\( \text{ef} \)
Size of the dynamic list used during search. Higher values lead to improved recall at the expense of longer search time. Can take values between \( k \) and the size of the dataset and may be greater or smaller than \( \text{ef\_construction} \). Typical values are \( 100 \sim 2000 \).

\( \text{verbose} \)
If TRUE, log progress to the console.

Value

a list containing:

- \( \text{idx} \) an n by k matrix containing the nearest neighbor indices.
- \( \text{dist} \) an n by k matrix containing the nearest neighbor distances.

Every item in the dataset is considered to be a neighbor of itself, so the first neighbor of item \( i \) should always be \( i \) itself. If that isn’t the case, then any of \( M, \text{ef\_construction} \) and \( \text{ef} \) may need increasing.

Hnswlib Parameters

Some details on the parameters used for index construction and search, based on [https://github.com/nmslib/hnswlib/blob/master/ALGO_PARAMS.md](https://github.com/nmslib/hnswlib/blob/master/ALGO_PARAMS.md):

- \( M \) Controls the number of bi-directional links created for each element during index construction. Higher values lead to better results at the expense of memory consumption, which is around \( M \times 8\sim10 \) bytes per bytes per stored element. High intrinsic dimensionalities will require higher values of \( M \). A range of \( 2 \sim 100 \) is typical, but \( 12 \sim 48 \) is ok for most use cases.

- \( \text{ef\_construction} \) Size of the dynamic list used during construction. A larger value means a better quality index, but increases build time. Should be an integer value between 1 and the size of the dataset. A typical range is \( 100 \sim 2000 \). Beyond a certain point, increasing \( \text{ef\_construction} \) has no effect. A sufficient value of \( \text{ef\_construction} \) can be determined by searching with \( \text{ef} = \text{ef\_construction} \), and ensuring that the recall is at least 0.9.

- \( \text{ef} \) Size of the dynamic list used during index search. Can differ from \( \text{ef\_construction} \) and be any value between \( k \) (the number of neighbors sought) and the number of elements in the index being searched.

References

hnsw_search

Examples

iris_nn_data <- hnsw_knn(as.matrix(iris[, -5]), k = 10)

hnsw_search

Search an hnswlib nearest neighbor index

Description

Search an hnswlib nearest neighbor index

Usage

hnsw_search(X, ann, k, ef = 10, verbose = FALSE)

Arguments

X  
A numeric matrix of data to search for neighbors.

ann  
an instance of a HnswL2, HnswCosine or HnswIp class.

k  
Number of neighbors to return. This can’t be larger than the number of items that were added to the index ann. To check the size of the index, call ann$size().

ef  
Size of the dynamic list used during search. Higher values lead to improved recall at the expense of longer search time. Can take values between k and the size of the dataset. Typical values are 100 - 2000.

verbose  
If TRUE, log progress to the console.

Value

a list containing:

- idx an n by k matrix containing the nearest neighbor indices.
- dist an n by k matrix containing the nearest neighbor distances.

Every item in the dataset is considered to be a neighbor of itself, so the first neighbor of item i should always be i itself. If that isn’t the case, then any of M, ef_construction and ef may need increasing.

Examples

irism <- as.matrix(iris[, -5])
ann <- hnsw_build(iris)
iris_nn <- hnsw_search(irism, ann, k = 5)
Index

hnsw_build, 2
hnsw_knn, 3
hnsw_search, 5
HnswCosine (RcppHnsw-package), 2
HnswIp (RcppHnsw-package), 2
HnswL2 (RcppHnsw-package), 2

Rcpp_HnswCosine-class
   (RcppHnsw-package), 2
Rcpp_HnswIp-class (RcppHnsw-package), 2
Rcpp_HnswL2-class (RcppHnsw-package), 2
RcppHnsw-package, 2