Package 'Rnanoflann'

July 21, 2025

Type Package

Title Extremely Fast Nearest Neighbor Search

Version 0.0.3

Date 2024-05-17

Maintainer Manos Papadakis <papadakm95@gmail.com>

Description Finds the k nearest neighbours for every point in a given dataset using Jose Luis' 'nanoflann' library. There is support for exact searches, fixed radius searches with 'kd' trees and two distances, the 'Euclidean' and 'Manhattan'. For more information see <https://github.com/jlblancoc/nanoflann>. Also, the 'nanoflann' library is exported and ready to be used via the linking to mechanism.

License GPL (>= 3)

Imports Rcpp (>= 1.0.11)

LinkingTo Rcpp, RcppArmadillo

BugReports https://github.com/ManosPapadakis95/Rnanoflann/issues

URL https://github.com/ManosPapadakis95/Rnanoflann

NeedsCompilation yes

Author Manos Papadakis [aut, cre, cph],
Jose Luis Blanco [aut, cph],
Michail Tsagris [ctb]

Repository CRAN

Date/Publication 2024-05-17 14:10:02 UTC

Contents

Rnanoflann-package *Extremely Fast Nearest Neighbor Search*

Description

Finds the k nearest neighbours for every point in a given dataset using Jose Luis' 'nanoflann' library. There is support for exact searches, fixed radius searches with 'kd' trees and two distances, the 'Euclidean' and 'Manhattan'. For more information see <https://github.com/jlblancoc/nanoflann>. Also, the 'nanoflann' library is exported and ready to be used via the linking to mechanism.

Details

Package:	Rnanoflann
Type:	Package
Version:	0.0.3
Date:	2024-05-17
License:	GPL (>= 3)

Author(s)

Authors: Manos Papadakis [aut, cre, cph], Jose Luis Blanco [aut, cph], Michail Tsagris [ctb]

Maintainer: Manos Papadakis <papadakm95@gmail.com>

nn *k-nearest neighbours search*

Description

Uses a kd-tree to find the k nearest neighbours for each point in a given dataset.

Usage

```
nn(data, points, k = nrow(data), method = "euclidean", search = "standard",
eps = 0.0, square = FALSE, sorted = FALSE, radius = 0.0, trans = TRUE,
leafs = 10L, p = 0.0, parallel = FALSE, cores = 0L)
```

Arguments

data	A numerical matrix. The k nearest points will be extracted from this matrix.
points	A numerical matrix. The function will find the nearest neighbours of each row of this matrix.
k	The number of nearest neighbours to search for.
method	The type of distance.See details for the supported metrics.
search	The type of search. Apart from the "standard" there is the "radius" option. It searches only for neighbours within a specified radius of the point. If there are no neighbours then the value "indices" will contain 0 and distances will contain 1.340781e+154 for that point.
eps	The accuracy of the search. When this is equal to 0, the function will return the exact k neighbours. If higher values are supplied, the function will return k approximate neighbours.
square	If you choose "euclidean" as the method, then you can have the option to return the squared Euclidean distances by setting this argument to TRUE. Default is FALSE.
sorted	Should the distances be sorted? This works only when search = "radius".
radius	The radius of the search, when search = "radius".
trans	Should the return matrices be transposed? The default value is TRUE.
p	This is for the the Minkowski, the power of the metric.
leafs	Number of divided points. Default is 10. Large values mean that the tree will be built faster (since the tree will be smaller), but each query will be slower (since the linear search in the leaf is to be done over more points). Small values will build the tree much slower (there will be many tree nodes), but queries will be faster... up to some point, since the "tree-part" of the search (logarithmic complexity) still has a significant cost.
parallel	Should the computations take place in parallel? The default value is FALSE.
cores	Number of threads for parallel version. The default is 0 which means all the available threads.

Details

The target of this function is to calculate the distances between xnew and x without having to calculate the whole distance matrix of xnew and x. The latter does extra calculations, which can be avoided.

- euclidean : $\sum \sqrt{(\sum |P_i - Q_i|^2)}$
- manhattan : $\sum \sum |P_i - Q_i|$
- minimum : $\sum \min |P_i - Q_i|$
- maximum : $\sum \max |P_i - Q_i|$
- minkowski : $\sum (\sum |P_i - Q_i|^p)^{\frac{1}{p}}$

- bhattacharyya : $\sum -ln \sum \sqrt{(P_i * Q_i)}$
- hellinger : $\sum 2 * \sqrt{(1 - \sum \sqrt{(P_i * Q_i)})}$
- kullback_leibler : $\sum \sum P_i * log(\frac{P_i}{Q_i})$
- jensen_shannon : $\sum 0.5 * (\sum P_i * log(2 * \frac{P_i}{Q_i} + Q_i) + \sum Q_i * log(2 * \frac{Q_i}{P_i} + Q_i))$
- canberra : $\sum \sum \frac{|P_i - Q_i|}{P_i + Q_i}$
- chi_square X^2 : $\sum \sum (\frac{(P_i - Q_i)^2}{P_i + Q_i})$
- soergel : $\sum \frac{\sum |P_i - Q_i|}{\sum \max(P_i, Q_i)}$
- sorensen : $\sum \frac{\sum |P_i - Q_i|}{\sum (P_i + Q_i)}$
- cosine : $\sum \frac{\sum (P_i * Q_i)}{\sqrt{(\sum P_i^2)} * \sqrt{(\sum Q_i^2)}}$
- wave_hedges : $\sum \frac{\sum |P_i - Q_i|}{\max(P_i, Q_i)}$
- motyka : $\sum \sum \frac{\min(P_i, Q_i)}{(P_i + Q_i)}$
- harmonic_mean : $\sum 2 * \frac{\sum P_i * Q_i}{P_i + Q_i}$
- jeffries_matusita : $\sum \sqrt{(2 - 2 * \sum \sqrt{(P_i * Q_i)})}$
- gower : $\sum \frac{1}{d} * \sum |P_i - Q_i|$
- kulczynski : $\sum \frac{\sum |P_i - Q_i|}{\sum \min(P_i, Q_i)}$
- itakura_saito : $\sum \frac{P_i}{Q_i} - log(\frac{P_i}{Q_i}) - 1$

Value

A list with 2 fields.

`indices`	A matrix with the indices of each nearest neighbour for each of the rows of the matrix "points".
`distances`	A matrix with the distances between each nearest neighbour and each of the rows of the matrix "points".

Examples

```
x <- as.matrix(iris[1:140, 1:4])
xnew <- as.matrix(iris[141:150, 1:4])
nn(data = x, points = xnew, k = 10)
```

Index