wbia-pyflann

Release latest

Nov 11, 2022

Contents:

CHAPTER 1

pyflann package

1.1 Submodules

1.2 pyflann.__main__ module

pyflann.__main__.**main**()

1.3 pyflann.exceptions module

exception pyflann.exceptions.**FLANNException**
Bases: Exception

1.4 pyflann.flann_ctypes module

class pyflann.flann_ctypes.**CustomStructure**
Bases: _ctypes.Structure

This class extends the functionality of the ctype's structure class by adding custom default values to the fields and a way of translating field types.

keys()

update(*dict*)

class pyflann.flann_ctypes.**FLANNParameters**
Bases: *pyflann.flann_ctypes.CustomStructure*

algorithm
Structure/Union member

branching
Structure/Union member

build_weight
Structure/Union member

cb_index
Structure/Union member

centers_init
Structure/Union member

checks
Structure/Union member

cores
Structure/Union member

eps
Structure/Union member

iterations
Structure/Union member

key_size_
Structure/Union member

leaf_max_size
Structure/Union member

log_level
Structure/Union member

max_neighbors
Structure/Union member

memory_weight
Structure/Union member

multi_probe_level_
Structure/Union member

random_seed
Structure/Union member

sample_fraction
Structure/Union member

sorted
Structure/Union member

table_number_
Structure/Union member

target_precision
Structure/Union member

trees
Structure/Union member

class pyflann.flann_ctypes.**FlannLib**
Bases: object

pyflann.flann_ctypes.**define_functions**(*fmtstr*)

pyflann.flann_ctypes.**ensure_2d_array**(*arr*, *flags*, ***kwargs*)

pyflann.flann_ctypes.**load_flann_library**()

CommandLine: python -c "import pyflann" –verbose

1.5 pyflann.index module

class pyflann.index.**FLANN**(***kwargs*)
Bases: object

This class defines a python interface to the FLANN lirary.

Example

```
>>> from pyflann import FLANN
>>> import numpy as np
>>> dvecs = np.random.rand(1000, 128)
>>> qvecs = np.random.rand(10, 128)
>>> flann = FLANN()
>>> params = flann.build_index(dvecs)
```

add_points(*pts*, *rebuild_threshold=2*)
Adds points to pre-built index.

Parameters

- **pts** – 2D numpy array of points.
- **rebuild_threshold** – reallocs index when it grows by factor of *rebuild_threshold*. A smaller value results is more space efficient but less computationally efficient. Must be greater than 1.

build_index(*pts*, ***kwargs*)
This builds and internally stores an index to be used for future nearest neighbor matchings. It erases any previously stored indexes, so use multiple instances of this class to work with multiple stored indices. Use nn_index(...) to find the nearest neighbors in this index.

pts is a 2d numpy array or matrix. All the computation is done in np.float32 type, but pts may be any type that is convertable to np.float32.

delete_index(***kwargs*)
Deletes the current index freeing all the momory it uses. The memory used by the dataset that was indexed is not freed unless there are no other references to those numpy arrays.

get_indexed_data()
returns all the data indexed by the FLANN object

(this returns points that have been removed but still exist in memory)

get_indexed_shape()
returns the shape of the data being indexed

hierarchical_kmeans(*pts*, *branch_size*, *num_branches*, *max_iterations=None*, *dtype=None*, ***kwargs*)
Clusters the data by using multiple runs of kmeans to recursively partition the dataset. The number of resulting clusters is given by (branch_size-1)*num_branches+1.

This method can be significantly faster when the number of desired clusters is quite large (e.g. a hundred or more). Higher branch sizes are slower but may give better results.

If dtype is None (the default), the array returned is the same type as pts. Otherwise, the returned array is of type dtype.

kmeans (*pts*, *num_clusters*, *max_iterations=None*, *dtype=None*, ***kwargs*)
Runs kmeans on pts with num_clusters centroids. Returns a numpy array of size num_clusters x dim.

If max_iterations is not None, the algorithm terminates after the given number of iterations regardless of convergence. The default is to run until convergence.

If dtype is None (the default), the array returned is the same type as pts. Otherwise, the returned array is of type dtype.

load_index (*filename*, *pts*)
Loads an index previously saved to disk.

nn (*pts*, *qpts*, *num_neighbors=1*, ***kwargs*)
Returns the num_neighbors nearest points in dataset for each point in testset.

nn_index (*qpts*, *num_neighbors=1*, ***kwargs*)
For each point in querypts, (which may be a single point), it returns the num_neighbors nearest points in the index built by calling build_index.

nn_radius (*query*, *radius*, ***kwargs*)

remove_point (*idx*)
Removes a point from a pre-built index.

remove_points (*id_list*)
Removes multiple points from the index

Params: id_list = point ids to be removed

Returns: void

save_index (*filename*)
This saves the index to a disk file.

shape

used_memory ()
Returns the amount of memory used by the index

Returns: int

used_memory_dataset ()
Returns the amount of memory used by the dataset

pyflann.index.**set_distance_type** (*distance_type*, *order=0*)
Sets the distance type used. Possible values: euclidean, manhattan, minkowski, max_dist, hik, hellinger, cs, kl.

pyflann.index.**to_bytes** (*string*)

1.6 Module contents

mkinit pyflann

class pyflann.**CustomStructure**
Bases: _ctypes.Structure

This class extends the functionality of the ctype's structure class by adding custom default values to the fields and a way of translating field types.

keys()

update(*dict*)

class pyflann.**FLANN**(***kwargs*)
Bases: object

This class defines a python interface to the FLANN lirary.

Example

```
>>> from pyflann import FLANN
>>> import numpy as np
>>> dvecs = np.random.rand(1000, 128)
>>> qvecs = np.random.rand(10, 128)
>>> flann = FLANN()
>>> params = flann.build_index(dvecs)
```

add_points(*pts*, *rebuild_threshold=2*)
Adds points to pre-built index.

Parameters

- **pts** – 2D numpy array of points.
- **rebuild_threshold** – reallocs index when it grows by factor of *rebuild_threshold*. A smaller value results is more space efficient but less computationally efficient. Must be greater than 1.

build_index(*pts*, ***kwargs*)
This builds and internally stores an index to be used for future nearest neighbor matchings. It erases any previously stored indexes, so use multiple instances of this class to work with multiple stored indices. Use nn_index(...) to find the nearest neighbors in this index.

pts is a 2d numpy array or matrix. All the computation is done in np.float32 type, but pts may be any type that is convertable to np.float32.

delete_index(***kwargs*)
Deletes the current index freeing all the momory it uses. The memory used by the dataset that was indexed is not freed unless there are no other references to those numpy arrays.

get_indexed_data()
returns all the data indexed by the FLANN object

(this returns points that have been removed but still exist in memory)

get_indexed_shape()
returns the shape of the data being indexed

hierarchical_kmeans(*pts*, *branch_size*, *num_branches*, *max_iterations=None*, *dtype=None*, ***kwargs*)
Clusters the data by using multiple runs of kmeans to recursively partition the dataset. The number of resulting clusters is given by (branch_size-1)*num_branches+1.

This method can be significantly faster when the number of desired clusters is quite large (e.g. a hundred or more). Higher branch sizes are slower but may give better results.

If dtype is None (the default), the array returned is the same type as pts. Otherwise, the returned array is of type dtype.

kmeans(*pts*, *num_clusters*, *max_iterations=None*, *dtype=None*, ***kwargs*)
Runs kmeans on pts with num_clusters centroids. Returns a numpy array of size num_clusters x dim.

If max_iterations is not None, the algorithm terminates after the given number of iterations regardless of convergence. The default is to run until convergence.

If dtype is None (the default), the array returned is the same type as pts. Otherwise, the returned array is of type dtype.

load_index(*filename*, *pts*)
Loads an index previously saved to disk.

nn(*pts*, *qpts*, *num_neighbors=1*, ***kwargs*)
Returns the num_neighbors nearest points in dataset for each point in testset.

nn_index(*qpts*, *num_neighbors=1*, ***kwargs*)
For each point in querypts, (which may be a single point), it returns the num_neighbors nearest points in the index built by calling build_index.

nn_radius(*query*, *radius*, ***kwargs*)

remove_point(*idx*)
Removes a point from a pre-built index.

remove_points(*id_list*)
Removes multiple points from the index

Params: id_list = point ids to be removed

Returns: void

save_index(*filename*)
This saves the index to a disk file.

shape

used_memory()
Returns the amount of memory used by the index

Returns: int

used_memory_dataset()
Returns the amount of memory used by the dataset

exception pyflann.**FLANNException**
Bases: Exception

class pyflann.**FLANNParameters**
Bases: *pyflann.flann_ctypes.CustomStructure*

algorithm
Structure/Union member

branching
Structure/Union member

build_weight
Structure/Union member

cb_index
Structure/Union member

centers_init
Structure/Union member

checks
Structure/Union member

cores
Structure/Union member

eps
Structure/Union member

iterations
Structure/Union member

key_size_
Structure/Union member

leaf_max_size
Structure/Union member

log_level
Structure/Union member

max_neighbors
Structure/Union member

memory_weight
Structure/Union member

multi_probe_level_
Structure/Union member

random_seed
Structure/Union member

sample_fraction
Structure/Union member

sorted
Structure/Union member

table_number_
Structure/Union member

target_precision
Structure/Union member

trees
Structure/Union member

pyflann.**FLANN_INDEX**
alias of ctypes.c_void_p

class pyflann.**FlannLib**
Bases: object

pyflann.**STRING**
alias of ctypes.c_char_p

pyflann.**define_functions**(*fmtstr*)

pyflann.**ensure_2d_array**(*arr*, *flags*, ***kwargs*)

pyflann.**index_type**
alias of numpy.int32

pyflann.**load_flann_library**()

CommandLine: python -c “import pyflann” –verbose

pyflann.**set_distance_type**(*distance_type*, *order=0*)

Sets the distance type used. Possible values: euclidean, manhattan, minkowski, max_dist, hik, hellinger, cs, kl.

pyflann.**to_bytes**(*string*)

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index