tfg_big_data_algorithms Documentation

Release 0.1

garciparedes

Jul 24, 2017

Contents:

CHAPTER 1

tf_G

Name tf_G

Description Python's Tensorflow Graph Library

Website https://github.com/garciparedes/tf_G

Author @garciparedes

Version 0.1

Description

This work consists of a study of a set of techniques and strategies related with algorithm's design, whose purpose is the resolution of problems on massive data sets, in an efficient way. This field is known as Algorithms for Big Data. In particular, this work has studied the Streaming Algorithms, which represents the basis of the data structures of sublinear order o(n) in space, known as Sketches. In addition, it has deepened in the study of problems applied to Graphs on the Semi-Streaming model. Next, the PageRank algorithm was analyzed as a concrete case study. Finally, the development of a library for the resolution of graph problems, implemented on the top of the intensive mathematical computation platform known as TensorFlow has been started.

Content

- Final Degree Project: Summary

How to install

If you have git installed, you can try:

```
$ pip install git+https://github.com/garciparedes/tf_G.git
```

If you get any installation or compilation errors, make sure you have the latest pip and setuptools:

```
$ pip install --upgrade pip setuptools
```

How to run the tests

Install in editable mode and call *pytest*:

```
$ pip install -e .
$ pytest
```

CHAPTER 2

tf_G

tf_G module

This module represents the parent module of the *tf_G* library.

CHAPTER 3

tf_G.graph

tf_G.graph Module

It contains the graph implementations on *tf_G* module.

Graph

class tf_G.graph.graph.**Graph**(*sess: tensorflow.python.client.session.Session*, *name: str*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *edges_np: numpy.ndarray = None*, *n: int = None*, *is_sparse: bool = False*) → None

Graph class implemented in the top of TensorFlow.

The class codifies the graph using an square matrix of 2-D shape and provides functionality operating with this matrix.

sess
: tf.Session – This attribute represents the session that runs the TensorFlow operations.

name
: *str* – This attribute represents the name of the object in TensorFlow's op Graph.

writer
: tf.summary.FileWriter – This attribute represents a TensorFlow's Writer, that is used to obtain stats. The default value is *None*.

_listeners
: set – The set of objects that will be notified when an edge modifies it weight.

n
: *int* – Represents the cardinality of the vertex set as Python *int*.

n_tf
: tf.Tensor – Represents the cardinality of the vertex set as 0-D Tensor.

m
int – Represents the cardinality of the edge set as Python *int*.

A_tf
`tf.Tensor` – Represents the Adjacency matrix of the graph as 2-D Tensor with shape [n,n].

out_degrees_tf
`tf.Tensor` – Represents the out-degrees of the vertices of the graph as 2-D Tensor with shape [n, 1]

in_degrees_tf
`tf.Tensor` – Represents the in-degrees of the vertices of the graph as 2-D Tensor with shape [1, n]

A_tf_vertex(*vertex: int*) → tensorflow.python.framework.ops.Tensor
Method that returns the adjacency of an individual vertex.

This method extracts the corresponding row referred to the *vertex* passed as parameter. It constructs a vector that contains the weight of the edge between *vertex* (obtained as parameter) and the vertex at position *i* in the vector.

Parameters **vertex** (*int*) – The index of the vertex that wants the degree.

Returns

A 1-D Tensor with the same length as the cardinality of the vertex set.

Return type (`tf.Tensor`)

L_pseudo_inverse_tf
Method that returns the pseudo inverse of the Laplacian matrix.

This method calculates the pseudo inverse matrix of the Laplacian of the Graph. It generates a matrix of the same shape as the Laplacian matrix, i.e. [n, n] where n is the cardinality of the vertex set.

Returns

A 2-D square Tensor with the he same length as cardinality of the vertex set representing the laplacian pseudo inverse.

Return type (`tf.Tensor`)

L_tf
This method returns the Laplacian of the graph.

The method generates a 2-D Array containing the laplacian matrix of the graph

Returns

A 2-D Tensor with [n,n] shape where n is the cardinality of the vertex set

Return type (`tf.Tensor`)

__init__(*sess: tensorflow.python.client.session.Session*, *name: str*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *edges_np: numpy.ndarray = None*, *n: int = None*, *is_sparse: bool = False*) → None
Class Constructor of the Graph

This method is called to construct a Graph object. This block of code initializes all the variables necessaries for this class to properly works.

This class can be initialized using an edge list, that fill the graph at this moment, or can be construct it from the cardinality of vertices set given by *n* parameter.

Parameters

- **sess** (`tf.Session`) – This attribute represents the session that runs the TensorFlow operations.

- **name** (*str*) – This attribute represents the name of the object in TensorFlow's op Graph.
- **writer** (tf.summary.FileWriter, optional) – This attribute represents a TensorFlow's Writer, that is used to obtain stats. The default value is *None*.
- **edges_np** (np.ndarray, optional) – The edge set of the graph codifies as *edges_np[:,0]* represents the sources and *edges_np[:,1]* the destinations of the edges. The default value is *None*.
- **n** (*int, optional*) – Represents the cardinality of the vertex set. The default value is *None*.
- **is_sparse** (*bool, optional*) – Use sparse Tensors if it's set to *True*. The default value is False' Not implemented yet. Show the Todo for more information.

Todo

•Implement variables as sparse when it's possible. Waiting to TensorFlow for it.

---

append(*src: int*, *dst: int*) → None

Append an edge to the graph.

This method process an input edge adding it to the graph updating all the variables necessaries to maintain the graph in correct state.

Parameters

- **src** (*int*) – The id of the source vertex of the edge.
- **dst** (*int*) – The id of the destination vertex of the edge.

Returns This method returns nothing.

edge_list_np

Method that returns the edge set of the graph as list.

This method return all the edges of the graph codified as 2-D matrix in which the first dimension represents each edge and second dimension the source and destination vertices of each edge.

Returns

A 2-D Array with the he same length as cardinality of the edge set in the first dimension and 2 in the second.

Return type (np.ndarray)

edge_list_tf

Method that returns the edge set of the graph as list.

This method return all the edges of the graph codified as 2-D matrix in which the first dimension represents each edge and second dimension the source and destination vertices of each edge.

Returns

A 2-D Tensor with the he same length as cardinality of the edge set in the first dimension and 2 in the second.

Return type (tf.Tensor)

in_degrees_np

This method returns the in-degree of all vertex as vector.

The method generates a 1-D Array containing the in-degree of the vertex *i* at position *i*

Returns

A 1-D Array with the same length as cardinality of the vertex set.

Return type (np.ndarray)

in_degrees_tf_vector
The in-degrees of the vertices of the graph

Method that returns the in-degrees of the vertices of the graph as 1-D Tensor with shape [n]

Returns

A 1-D Tensor with the same length as the cardinality of the vertex set.

Return type (tf.Tensor)

is_not_sink_tf
This method returns if a vertex is a sink vertex as vector.

The method generates a 1-D Tensor containing the boolean values that indicates if the vertex at position *i* is a sink vertex.

Returns

A 1-D Tensor with the same length as cardinality of the vertex set.

Return type (tf.Tensor)

is_not_sink_tf_vertex(*vertex: int*) → TF_type
This method returns if a vertex is a sink vertex as vector.

The method generates a 1-D Tensor containing the boolean values that indicates if the vertex at position *i* is a sink vertex.

Parameters **vertex** (*int*) – The index of the vertex that wants to know if is sink.

Returns

A 0-D Tensor that represents if a vertex is a sink vertex

Return type (tf.Tensor)

out_degrees_np
This method returns the degree of all vertex as vector.

The method generates a 1-D Array containing the out-degree of the vertex *i* at position *i*

Returns

A 1-D Array with the same length as cardinality of the vertex set.

Return type (np.ndarray)

out_degrees_tf_vector
The out-degrees of the vertices of the graph

Method that returns the out-degrees of the vertices of the graph as 1-D Tensor with shape [n]

Returns A 1-D Tensor with the same length as the cardinality of the vertex set.

Return type (tf.Tensor)

out_degrees_tf_vertex(*vertex: int*) → tensorflow.python.framework.ops.Tensor
This method returns the degree of all vertex as vector.

The method generates a 0-D Array containing the out-degree of the vertex i.

Parameters **vertex** (*int*) – The index of the vertex that wants the degree.

Returns

A 1-D Array with the same length as cardinality of the vertex set.

Return type (np.ndarray)

remove(*src: int*, *dst: int*) → None
Remove an edge to the graph.

This method process an input edge deleting it to the graph updating all the variables necessaries to maintain the graph in correct state.

Parameters

- **src** (*int*) – The id of the source vertex of the edge.
- **dst** (*int*) – The id of the destination vertex of the edge.

Returns This method returns nothing.

GraphConstructor

class tf_G.graph.graph_constructor.**GraphConstructor**
Class that helps to construct a Graph object.

This class contains a set of static methods that helps in the task of graph construction and initialization. It provides the generation of a Graph from an edge list, and allows to generate empty Graphs, sparsifier Graphs an also random Graphs.

static as_naive_sparsifier(*sess: tensorflow.python.client.session.Session*, *graph: tf_G.graph.graph.Graph*, *p: float*, *is_sparse: bool = False*) → tf_G.graph.graph.Graph
Generates a sparsifier graph of the given graph.

The method picks the edges with probability uniform probability *p* from edge set of the graph given as parameter. This does not provide any guarantee from the structure of the original graph.

Parameters

- **sess** (tf.Session) – This attribute represents the session that runs the TensorFlow operations.
- **graph** (tf_G.Graph) – The input graph to pick the edges
- **p** (*float*) – The picking probability value. It must be in the [0,1] interval.
- **is_sparse** (*bool*) – Use sparse Tensors if it's set to True. Not implemented yet.

Returns

The resulting graph with less edges than the original graph.

Return type (tf_G.Graph)

classmethod as_sparsifier(*sess*, *graph: tf_G.graph.graph.Graph*, *p: float*, *is_sparse=False*)
Generates a sparsifier graph from the given graph.

The method picks the edges with probability uniform probability *p* from edge set of the graph given as parameter. The sparsifier uses an heuristic to picks the more edges from the vertices with big out_degree to try to maintain the structure of the graph.

Parameters

- **sess** (tf.Session) – This attribute represents the session that runs the TensorFlow operations.
- **graph** (tf_G.Graph) – The input graph to pick the edges.
- **p** (*float*) – The picking probability value. It must be in the [0,1] interval.
- **is_sparse** (*bool*) – Use sparse Tensors if it's set to True. Not implemented yet.

Returns

The resulting graph sparsifier with less edges than the original graph.

Return type (tf_G.Graph)

static empty(*sess: tensorflow.python.client.session.Session*, *name: str*, *n: int*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *sparse: bool = False*) → tf_G.graph.graph.Graph

Generates an empty Graph.

This method generates an empty graph with the number of vertex fixed at the construction. The graph allows addition and deletion of edges.

Parameters

- **sess** (tf.Session) – This attribute represents the session that runs the TensorFlow operations.
- **name** (*str*) – This attribute represents the name of the object in TensorFlow's op Graph.
- **n** (*int*) – The cardinality of vertex set of the empty graph.
- **writer** (tf.summary.FileWriter, optional) – This attribute represents a TensorFlow's Writer, that is used to obtain stats. The default value is *None*.
- **is_sparse** (*bool, optional*) – Use sparse Tensors if it's set to True. Not implemented yet. Show the Todo. The default value is *False*.

Returns

A empty graph that allows additions and deletions of edges from vertex in the interval [0,n].

Return type (tf_G.Graph)

static empty_sparsifier(*sess: tensorflow.python.client.session.Session*, *name: str*, *n: int*, *p: float*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *is_sparse: bool = False*) → tf_G.graph.graph_sparsifier.GraphSparsifier

Generates an empty Sparsifier Graph.

This method generates an empty sparsifier graph with the number of vertex fixed at the construction. The graph allows addition and deletion of edges. The sparsifier means that the graph will not add all edges. Only a subset of it to improve the performance of the algorithms.

Parameters

- **sess** (tf.Session) – This attribute represents the session that runs the TensorFlow operations.
- **name** (*str*) – This attribute represents the name of the object in TensorFlow's op Graph.
- **n** (*int*) – The cardinality of vertex set of the empty graph.
- **p** (*float*) – The picking probability value. It must be in the [0,1] interval.
- **writer** (tf.summary.FileWriter, optional) – This attribute represents a TensorFlow's Writer, that is used to obtain stats. The default value is *None*.

- **is_sparse** (*bool, optional*) – Use sparse Tensors if it's set to True. Not implemented yet. Show the Todo. The default value is *False*.

Returns

A empty graph that allows additions and deletions of edges from vertex in the interval [0,n].

Return type (tf_G.GraphSparsifier)

static from_edges(*sess: tensorflow.python.client.session.Session*, *name: str*, *edges_np: numpy.ndarray*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *is_sparse: bool = False*) → tf_G.graph.graph.Graph

Generates a graph from a set of edges.

This method acts as interface between the Graph constructor and the rest exterior.

Parameters

- **sess** (tf.Session) – This attribute represents the session that runs the TensorFlow operations.
- **name** (*str*) – This attribute represents the name of the object in TensorFlow's op Graph.
- **edges_np** (np.ndarray) – The edge set of the graph codifies as *edges_np[:,0]* represents the sources and *edges_np[:,1]* the destinations of the edges.
- **writer** (tf.summary.FileWriter, optional) – This attribute represents a TensorFlow's Writer, that is used to obtain stats. The default value is *None*.
- **is_sparse** (*bool, optional*) – Use sparse Tensors if it's set to True. Not implemented yet. Show the Todo. The default value is *False*.

Returns

A graph containing all the edges passed as input in *edges_np*.

Return type (tf_G.Graph)

static unweighted_random(*sess: tensorflow.python.client.session.Session*, *name: str*, *n: int*, *m: int*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *is_sparse: bool = False*) → tf_G.graph.graph.Graph

Generates a random unweighted graph.

This method generates a random unweighted graph with *n* vertex and *m* edges. The edge set is generated using a uniform distribution.

Parameters

- **sess** (tf.Session) – This attribute represents the session that runs the TensorFlow operations.
- **name** (*str*) – This attribute represents the name of the object in TensorFlow's op Graph.
- **n** (*int*) – The cardinality of vertex set of the random graph.
- **m** (*int*) – The cardinality of edge set of the random graph.
- **writer** (tf.summary.FileWriter, optional) – This attribute represents a TensorFlow's Writer, that is used to obtain stats. The default value is *None*.
- **is_sparse** (*bool, optional*) – Use sparse Tensors if it's set to True. Not implemented yet. Show the Todo. The default value is *False*.

Returns

A empty graph that allows additions and deletions of edges from vertex in the interval [0,n].

Return type (tf_G.GraphSparsifier)

GraphSparsifier

class tf_G.graph.graph_sparsifier.**GraphSparsifier**(*sess: tensorflow.python.client.session.Session*, *p: float*, *graph: tf_G.graph.graph.Graph = None*, *is_sparse: bool = False*, *name: str = None*, *n: int = None*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*) → None

The graph sparsifier class implemented in the top of TensorFlow.

This class inherits the Graph class and modifies it functionality adding a level of randomness on edge additions and deletions, that improves the performance of the results.

sess
tf.Session – This attribute represents the session that runs the TensorFlow operations.

name
str – This attribute represents the name of the object in TensorFlow's op Graph.

writer
tf.summary.FileWriter – This attribute represents a TensorFlow's Writer, that is used to obtain stats. The default value is *None*.

n
int – Represents the cardinality of the vertex set as Python *int*.

p
float – The default probability to pick an edge and add it to the GraphSparsifier.

n_tf
tf.Tensor – Represents the cardinality of the vertex set as 0-D Tensor.

m
int – Represents the cardinality of the edge set as Python *int*.

A_tf
tf.Tensor – Represents the Adjacency matrix of the graph as 2-D Tensor with shape [n,n].

out_degrees_tf
tf.Tensor – Represents the out-degrees of the vertices of the graph as 2-D Tensor with shape [n, 1].

in_degrees_tf
tf.Tensor – Represents the in-degrees of the vertices of the graph as 2-D Tensor with shape [1, n].

__init__(*sess: tensorflow.python.client.session.Session*, *p: float*, *graph: tf_G.graph.graph.Graph = None*, *is_sparse: bool = False*, *name: str = None*, *n: int = None*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*) → None
Class Constructor of the GraphSparsfiier

This method is called to construct a Graph object. This block of code initializes all the variables necessaries for this class to properly works.

This class can be initialized using an edge list, that fill the graph at this moment, or can be construct it from the cardinality of vertices set given by *n* parameter.

Parameters

- **sess** (`tf.Session`) – This attribute represents the session that runs the TensorFlow operations.
- **p** (*float*) – The default probability to pick an edge and add it to the GraphSparsifier.
- **graph** (`tf_G.Graph`, optional) – The input graph to pick the edges. The default value is *None*.
- **writer** (`tf.summary.FileWriter`, optional) – This attribute represents a TensorFlow's Writer, that is used to obtain stats. The default value is *None*.
- **name** (*str, optional*) – This attribute represents the name of the object in TensorFlow's op Graph.
- **n** (*int, optional*) – Represents the cardinality of the vertex set. The default value is *None*.
- **is_sparse** (*bool, optional*) – Use sparse Tensors if it's set to *True*. The default value is False' Not implemented yet. Show the Todo for more information.

Todo

•Implement variables as sparse when it's possible. Waiting to TensorFlow for it.

---

append(*src: int*, *dst: int*)

Append an edge to the graph.

This method overrides it parent's functionality adding a certain grade of probability.

This method process an input edge adding it to the graph updating all the variables necessaries to maintain the graph in correct state. The additions works with some probability, so there the addition is not guaranteed.

Parameters

- **src** (*int*) – The id of the source vertex of the edge.
- **dst** (*int*) – The id of the destination vertex of the edge.

Returns This method returns nothing.

remove(*src: int*, *dst: int*)

Remove an edge to the graph.

This method overrides it parent's functionality adding a certain grade of probability.

This method process an input edge deleting it to the graph updating all the variables necessaries to maintain the graph in correct state. The deletions works with some probability, so there the deletion is not guaranteed.

Parameters

- **src** (*int*) – The id of the source vertex of the edge.
- **dst** (*int*) – The id of the destination vertex of the edge.

Returns This method returns nothing.

CHAPTER 4

tf_G.algorithms

tf_G.algorithms Module

This module contains a set of submodules that represents the implementation of a set graph algorithms.

CHAPTER 5

tf_G.algorithms.pagerank

tf_G.algorithms.pagerank Module

This module contains a set of PageRank's algorithm implementations on the *tf_G* module.

PageRank

class `tf_G.algorithms.pagerank.pagerank.`**`PageRank`**(*sess: tensorflow.python.client.session.Session*, *name: str*, *graph: tf_G.graph.graph.Graph*, *beta: float*, *T: tf_G.algorithms.pagerank.transition.transition.Transition*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *is_sparse: bool = False*) → None

PageRank base class.

This class model the PageRank algorithm as Abstract Class containing all methods that the heir classes need to implements. Also, this class provides a set of attributes that helps to implement the algorithm.

The PageRank algorithm calculates the rank of each vertex in a graph based on the relational structure from them and giving more importance to the vertices that connects with edges to vertices with very high in-degree recursively.

This class depends on the TensorFlow library, so it's necessary to install it to properly work.

`sess`
`tf.Session` – This attribute represents the session that runs the TensorFlow operations.

`name`
str – This attribute represents the name of the object in TensorFlow's op Graph.

G
tf_G.Graph – The graph on witch it will be calculated the algorithm. It will be treated as Directed Weighted Graph.

beta
float – The reset probability of the random walks, i.e. the probability that a user that surfs the graph an decides to jump to another vertex not connected to the current.

T
tf_G.Transition – The transition matrix that provides the probability distribution relative to the walk to another node of the graph.

v
tf.Variable – The stationary distribution vector. It contains the normalized probability to stay in each vertex of the graph. So represents the PageRank ranking of the graph.

writer
tf.summary.FileWriter – This attribute represents a TensorFlow's Writer, that is used to obtain stats.

is_sparse
bool – Use sparse Tensors if it's set to True. Not implemented yet.

__init__(*sess: tensorflow.python.client.session.Session*, *name: str*, *graph: tf_G.graph.graph.Graph*, *beta: float*, *T: tf_G.algorithms.pagerank.transition.transition.Transition*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *is_sparse: bool = False*) → None
The constructor of the class.

This method initializes all the attributes needed to compute the PageRank of the graph.

Parameters

- **sess** (tf.Session) – This attribute represents the session that runs the TensorFlow operations.
- **name** (*str*) – This attribute represents the name of the object in TensorFlow's op Graph.
- **beta** (*float*) – The reset probability of the random walks, i.e. the probability that a user that surfs the graph an decides to jump to another vertex not connected to the current.
- **T** (tf_G.Transition) – The transition matrix that provides the probability distribution relative to the walk to another node of the graph.
- **v** (tf.Variable) – The stationary distribution vector. It contains the normalized probability to stay in each vertex of the graph. So represents the PageRank ranking of the graph.
- **writer** (tf.summary.FileWriter) – This attribute represents a TensorFlow's Writer, that is used to obtain stats.
- **is_sparse** (*bool*) – Use sparse Tensors if it's set to True. Not implemented yet.

error_vector_compare_np(*other_pr: tf_G.algorithms.pagerank.pagerank.PageRank*, *k: int = -1*) → numpy.ndarray
The comparison method between two PageRank algorithm results.

This method compares the *self* PageRank with another one passed as parameter of the function. The comparison is based on the difference of the Norm One of each *v* vector.

The method also provides a *k* parameter as option to base the comparison only the *k* better ranked vertices.

Parameters

- **other_pr** (tf_G.PageRank) – Another PageRank object to compare the resulting ranking.
- **k** (*int, optional*) – An additional parameter that allows to base the comparison only on the *k* better vertices. Not implemented yet.

Returns

A *np.ndarray* with 0-D shape, that represents the difference between the two rankings using the Norm One.

Return type (np.ndarray)

error_vector_compare_tf(*other_pr: tf_G.algorithms.pagerank.pagerank.PageRank*, *k: int = -1*) → tensorflow.python.framework.ops.Tensor

The comparison method between two PageRank algorithm results.

This method compares the *self* PageRank with another one passed as parameter of the function. The comparison is based on the difference of the Norm One of each *v* vector.

The method also provides a *k* parameter as option to base the comparison only the *k* better ranked vertices.

Parameters

- **other_pr** (tf_G.PageRank) – Another PageRank object to compare the resulting ranking.
- **k** (*int, optional*) – An additional parameter that allows to base the comparison only on the *k* better vertices. Not implemented yet.

Returns

A *tf.Tensor* with 0-D shape, that represents the difference between the two rankings using the Norm One.

Return type (tf.Tensor)

Todo

•Implement ranking based only on the *k* better ranked vertices.

---

pagerank_vector_np(*convergence: float = 1.0*, *steps: int = 0*, *topics: typing.List[int] = None*, *c_criterion=<function ConvergenceCriterion.<lambda>>*) → numpy.ndarray

The Method that runs the PageRank algorithm

This method returns a Numpy Array that contains the result of running the PageRank algorithm customized by the parameters passed to it.

This method acts as interface between the algorithm and the external classes, so it contains a set of parameters that in some implementations of PageRank algorithms will not be needed. All the parameters is defined as optional for this reason.

Parameters

- **convergence** (*float, optional*) – A float between 0 and 1 that represents the convergence rate that allowed to finish the iterative implementations of the algorithm to accept the solution. It has more preference than the *steps* parameter. Default to *1.0*.
- **steps** (*int, optional*) – A positive integer that sets the number of iterations that the iterative implementations will run the algorithm until finish. It has less preference than the *convergence* parameter. Default to *0*.

- **topics** (list of int, optional) – A list of integers that represent the set of vertex where the random jumps arrives. If this parameter is used, the uniform distribution over all vertices of the random jumps will be modified to jump only to this vertex set. Default to *None*.
- **c_criterion** (function, optional) – The function used to calculate if the Convergence Criterion of the iterative implementations is reached. Default to *tf_G.ConvergenceCriterion.ONE*.

Returns

A 1-D *np.ndarray* of [n] shape, where *n* is the cardinality of the graph vertex set. It contains the normalized rank of vertex *i* at position *i*.

Return type (np.ndarray)

pagerank_vector_tf(*convergence: float = 1.0*, *steps: int = 0*, *topics: typing.List[int] = None*, *topics_decrement: bool = False*, *c_criterion=<function ConvergenceCriterion.<lambda>>*) → tensorflow.python.framework.ops.Tensor

The Method that runs the PageRank algorithm

This method generates a TensorFlow graph of operations needed to calculate the PageRank Algorithm and sets to it different parameters passed as parameters.

This method acts as interface between the algorithm and the external classes, so it contains a set of parameters that in some implementations of PageRank algorithms will not be needed. All the parameters is defined as optional for this reason.

Parameters

- **convergence** (*float, optional*) – A float between 0 and 1 that represents the convergence rate that allowed to finish the iterative implementations of the algorithm to accept the solution. It has more preference than the *steps* parameter. Default to *1.0*.
- **steps** (*int, optional*) – A positive integer that sets the number of iterations that the iterative implementations will run the algorithm until finish. It has less preference than the *convergence* parameter. Default to *0*.
- **topics** (list of int, optional) – A list of integers that represent the set of vertex where the random jumps arrives. If this parameter is used, the uniform distribution over all vertices of the random jumps will be modified to jump only to this vertex set. Default to *None*.
- **topics_decrement** (*bool, optional*) – If topics is not None and topics_decrement is *True* the topics will be casted to 0-Index. Default ‘ to False‘.
- **c_criterion** (function, optional) – The function used to calculate if the Convergence Criterion of the iterative implementations is reached. Default to *tf_G.ConvergenceCriterion.ONE*.

Returns

A 1-D *tf.Tensor* of [n] shape, where *n* is the cardinality of the graph vertex set. It contains the normalized rank of vertex *i* at position *i*.

Return type (tf.Tensor)

ranks_np(*convergence: float = 1.0*, *steps: int = 0*, *topics: typing.List[int] = None*, *topics_decrement: bool = False*) → numpy.ndarray

Generates a ranked version of PageRank results.

This method returns the PageRank ranking of the graph sorted by the position of each vertex in the rank. So it generates a 2-D matrix with shape [n,2] where n is the cardinality of the vertex set of the graph, and

at the first column it contains the index of vertex and the second column contains it normalized rank. The *i* row is referred to the vertex with *i* position in the rank.

Parameters

- **convergence** (*float, optional*) – A float between 0 and 1 that represents the convergence rate that allowed to finish the iterative implementations of the algorithm to accept the solution. It has more preference than the *steps* parameter. Default to *1.0*.
- **steps** (*int, optional*) – A positive integer that sets the number of iterations that the iterative implementations will run the algorithm until finish. It has less preference than the *convergence* parameter. Default to *0*.
- **topics** (list of int, optional) – A list of integers that represent the set of vertex where the random jumps arrives. If this parameter is used, the uniform distribution over all vertices of the random jumps will be modified to jump only to this vertex set. Default to *None*.
- **topics_decrement** (*bool, optional*) – If topics is not None and topics_decrement is *True* the topics will be casted to 0-Index. Default ‘ to False‘.

Returns

A 2-D *np.ndarray* than represents a sorted PageRank ranking of the graph.

Return type (np.ndarray)

update_edge(*edge: numpy.ndarray*, *change: float*) → None

The callback to receive notifications about edge changes in the graph.

This method is called from the Graph when an addition or deletion is produced on the edge set. So probably is necessary to recompute the PageRank ranking.

Parameters

- **edge** (np.ndarray) – A 1-D *np.ndarray* that represents the edge that changes in the graph, where *edge[0]* is the source vertex, and *edge[1]* the destination vertex.
- **change** (*float*) – The variation of the edge weight. If the final value is 0.0 then the edge is removed.

Returns This method returns nothing.

AlgebraicPageRank

class tf_G.algorithms.pagerank.algebraic_pagerank.**AlgebraicPageRank**(*sess: tensorflow.python.client.session.Session*, *name: str*, *graph: tf_G.graph.graph.Graph*, *beta: float*, *writer: tensorflow.python.summary.writer.writer.Fi = None*, *is_sparse: bool = False*) → None

The Algebraic PageRank implementation.

This class acts as the algebraic algorithm to obtain the PageRank ranking of a graph.

The PageRank algorithm calculates the rank of each vertex in a graph based on the relational structure from them and giving more importance to the vertices that connects with edges to vertices with very high in-degree recursively.

This class depends on the TensorFlow library, so it's necessary to install it to properly work.

sess
: tf.Session – This attribute represents the session that runs the TensorFlow operations.

name
: *str* – This attribute represents the name of the object in TensorFlow's op Graph.

beta
: *float* – The reset probability of the random walks, i.e. the probability that a user that surfs the graph an decides to jump to another vertex not connected to the current.

T
: tf_G.Transition – The transition matrix that provides the probability distribution relative to the walk to another node of the graph.

v
: tf.Variable – The stationary distribution vector. It contains the normalized probability to stay in each vertex of the graph. So represents the PageRank ranking of the graph.

writer
: tf.summary.FileWriter – This attribute represents a TensorFlow's Writer, that is used to obtain stats.

is_sparse
: *bool* – Use sparse Tensors if it's set to True. Not implemented yet.

__init__(*sess: tensorflow.python.client.session.Session*, *name: str*, *graph: tf_G.graph.graph.Graph*, *beta: float*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *is_sparse: bool = False*) → None

Constructor of the class.

This method initializes the attributes needed to run the Algebraic version of PageRank algorithm. It uses the *tf_G.TransitionMatrix* as transition matrix.

Parameters

- **sess** (tf.Session) – This attribute represents the session that runs the TensorFlow operations.
- **name** (*str*) – This attribute represents the name of the object in TensorFlow's op Graph.
- **G** (tf_G.Graph) – The graph on witch it will be calculated the algorithm. It will be treated as Directed Weighted Graph.
- **beta** (*float*) – The reset probability of the random walks, i.e. the probability that a user that surfs the graph an decides to jump to another vertex not connected to the current.
- **v** (tf.Variable) – The stationary distribution vector. It contains the normalized probability to stay in each vertex of the graph. So represents the PageRank ranking of the graph.
- **writer** (tf.summary.FileWriter) – This attribute represents a TensorFlow's Writer, that is used to obtain stats.
- **is_sparse** (*bool*) – Use sparse Tensors if it's set to True. Not implemented yet.

update_edge(*edge: numpy.ndarray*, *change: float*) → None

The callback to receive notifications about edge changes in the graph.

This method is called from the Graph when an addition or deletion is produced on the edge set. So probably is necessary to recompute the PageRank ranking.

Parameters

- **edge** (np.ndarray) – A 1-D *np.ndarray* that represents the edge that changes in the graph, where *edge[0]* is the source vertex, and *edge[1]* the destination vertex.
- **change** (*float*) – The variation of the edge weight. If the final value is 0.0 then the edge is removed.

Returns This method returns nothing.

IterativePageRank

class tf_G.algorithms.pagerank.iterative_pagerank.**IterativePageRank**(*sess: tensorflow.python.client.session.Session*, *name: str*, *graph: tf_G.graph.graph.Graph*, *beta: float*, *writer: tensorflow.python.summary.writer.writer.Fi = None*, *is_sparse: bool = False*) → None

The Iterative PageRank implementation.

This class acts as the iterative algorithm to obtain the PageRank ranking of a graph.

The PageRank algorithm calculates the rank of each vertex in a graph based on the relational structure from them and giving more importance to the vertices that connects with edges to vertices with very high in-degree recursively.

This class depends on the TensorFlow library, so it's necessary to install it to properly work.

sess
`tf.Session` – This attribute represents the session that runs the TensorFlow operations.

name
str – This attribute represents the name of the object in TensorFlow's op Graph.

beta
float – The reset probability of the random walks, i.e. the probability that a user that surfs the graph an decides to jump to another vertex not connected to the current.

T
`tf_G.Transition` – The transition matrix that provides the probability distribution relative to the walk to another node of the graph.

v
`tf.Variable` – The stationary distribution vector. It contains the normalized probability to stay in each vertex of the graph. So represents the PageRank ranking of the graph.

writer
`tf.summary.FileWriter` – This attribute represents a TensorFlow's Writer, that is used to obtain stats.

is_sparse
bool – Use sparse Tensors if it's set to True. Not implemented yet.

iter
`tf.Tensor` – The operation that will be repeated in each iteration of the algorithm.

__init__(*sess: tensorflow.python.client.session.Session*, *name: str*, *graph: tf_G.graph.graph.Graph*, *beta: float*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *is_sparse: bool = False*) → None
Constructor of the class.

This method initializes the attributes needed to run the Algebraic version of PageRank algorithm. It uses the *tf_G.TransitionResetMatrix* as transition matrix between vertex.

Parameters

- **sess** (`tf.Session`) – This attribute represents the session that runs the TensorFlow operations.
- **name** (*str*) – This attribute represents the name of the object in TensorFlow's op Graph.
- **G** (`tf_G.Graph`) – The graph on witch it will be calculated the algorithm. It will be treated as Directed Weighted Graph.
- **beta** (*float*) – The reset probability of the random walks, i.e. the probability that a user that surfs the graph an decides to jump to another vertex not connected to the current.
- **v** (`tf.Variable`) – The stationary distribution vector. It contains the normalized probability to stay in each vertex of the graph. So represents the PageRank ranking of the graph.
- **writer** (`tf.summary.FileWriter`) – This attribute represents a TensorFlow's Writer, that is used to obtain stats.
- **is_sparse** (*bool*) – Use sparse Tensors if it's set to True. Not implemented yet.

update_edge(*edge: numpy.ndarray*, *change: float*) → None
The callback to receive notifications about edge changes in the graph.

This method is called from the Graph when an addition or deletion is produced on the edge set. So probably is necessary to recompute the PageRank ranking.

Parameters

- **edge** (np.ndarray) – A 1-D *np.ndarray* that represents the edge that changes in the graph, where *edge[0]* is the source vertex, and *edge[1]* the destination vertex.
- **change** (*float*) – The variation of the edge weight. If the final value is 0.0 then the edge is removed.

Returns This method returns nothing.

CHAPTER 6

tf_G.algorithms.pagerank.transition

tf_G.algorithms.pagerank.transition Module

This module contains a set of classes that represents the transition matrix of a graph, that is used in PageRank algorithms.

Transition

class `tf_G.algorithms.pagerank.transition.transition.`**`Transition`**(*sess: tensorflow.python.client.session.Session*, *name: str*, *graph: tf_G.graph.graph.Graph*, *writer: tensorflow.python.summary.writer.writer.FileWrit = None*, *is_sparse: bool = False*) → None

Transition Base Class

This class acts as base class of transition behavior between vertices of the graph. This class is used to use as base type that provides this functionality and also to store the common attributes that uses all Transition implementations.

The heiress classes need to implement the *get_tf()* method that provides the transitions.

`sess`
`tf.Session` – This attribute represents the session that runs the TensorFlow operations.

`name`
str – This attribute represents the name of the object in TensorFlow's op Graph.

`writer`
`tf.summary.FileWriter` – This attribute represents a TensorFlow's Writer, that is used to obtain stats.

is_sparse
bool – Use sparse Tensors if it's set to True. Not implemented yet. Show the Todo.

_listeners
`set` – The set of objects that will be notified when an edge modifies it weight.

G
`tf_G.Graph` – The graph on which the transition is referred.

__init__(*sess: tensorflow.python.client.session.Session*, *name: str*, *graph: tf_G.graph.graph.Graph*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *is_sparse: bool = False*) → None
Constructor of the class.

This method is called to create a new instance of Transition class.

Parameters

- **sess** (`tf.Session`) – This attribute represents the session that runs the TensorFlow operations.
- **name** (*str*) – This attribute represents the name of the object in TensorFlow's op Graph.
- **graph** (`tf_G.Graph`) – The graph on which the transition is referred.
- **writer** (`tf.summary.FileWriter`) – This attribute represents a TensorFlow's Writer, that is used to obtain stats.
- **is_sparse** (*bool*) – Use sparse Tensors if it's set to True. Not implemented yet. Show the Todo.

get_tf(**args*, ***kwargs*)
The method that returns the transition Tensor.

This method will return the transition matrix of the graph.

Parameters

- ***args** – The args of the *get_tf()* method.
- ****kwargs** – The kwargs of the *get_tf()* method.

Returns

A *tf.Tensor* that contains the distribution of transitions over vertices of the graph.

Return type (`tf.Tensor`)

TransitionMatrix

class tf_G.algorithms.pagerank.transition.transition_matrix.**TransitionMatrix**(*sess: tensorflow.python.client.sessio name: str, graph: tf_G.graph.graph.Grap writer: tensorflow.python.summary.w = None, is_sparse: bool = False*) → None

Transition Matrix Class

This class implements the functionality of a 2-D matrix that represents the probability distribution of walk between the vertices of the graph.

sess
tf.Session – This attribute represents the session that runs the TensorFlow operations.

name
str – This attribute represents the name of the object in TensorFlow's op Graph.

writer
tf.summary.FileWriter – This attribute represents a TensorFlow's Writer, that is used to obtain stats.

is_sparse
bool – Use sparse Tensors if it's set to True. Not implemented yet. Show the Todo.

G
tf_G.Graph – The graph on which the transition is referred.

transition
tf.Variable – The 2-D *tf.Tensor* with the same shape as adjacency matrix of the graph, that represents the probabilities to move from one vertex to another.

__init__(*sess: tensorflow.python.client.session.Session*, *name: str*, *graph: tf_G.graph.graph.Graph*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *is_sparse: bool = False*) → None

Constructor of the class.

This method is called to create a new instance of Transition class.

Parameters

- **sess** (tf.Session) – This attribute represents the session that runs the TensorFlow operations.
- **name** (*str*) – This attribute represents the name of the object in TensorFlow's op Graph.

- **graph** (tf_G.Graph) – The graph on which the transition is referred.
- **writer** (tf.summary.FileWriter) – This attribute represents a TensorFlow's Writer, that is used to obtain stats.
- **is_sparse** (*bool*) – Use sparse Tensors if it's set to True. Not implemented yet. Show the Todo.

get_tf(**args*, ***kwargs*)
The method that returns the transition Tensor.

This method will return the transition matrix of the graph.

Parameters

- ***args** – The args of the *get_tf()* method.
- ****kwargs** – The kwargs of the *get_tf()* method.

Returns

A *tf.Tensor* that contains the distribution of transitions over vertices of the graph.

Return type (tf.Tensor)

update_edge(*edge: numpy.ndarray*, *change: float*) → None
The callback to receive notifications about edge changes in the graph.

This method is called from the Graph when an addition or deletion is produced on the edge set. So probably is necessary to recompute the transition matrix.

Parameters

- **edge** (np.ndarray) – A 1-D *np.ndarray* that represents the edge that changes in the graph, where *edge[0]* is the source vertex, and *edge[1]* the destination vertex.
- **change** (*float*) – The variation of the edge weight. If the final value is 0.0 then the edge is removed.

Returns This method returns nothing.

TransitionResetMatrix

class tf_G.algorithms.pagerank.transition.transition_reset_matrix.**TransitionResetMatrix**(*sess: tensorflow.python.client.session.Session*, *name: str*, *graph: tf_G.graph.graph.Graph*, *beta: float*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *is_sparse: bool = False*) → None

Transition Matrix Class

This class implements the functionality of a 2-D matrix that represents the probability distribution of walk between the vertices of the graph.

sess
tf.Session – This attribute represents the session that runs the TensorFlow operations.

name
str – This attribute represents the name of the object in TensorFlow's op Graph.

writer
tf.summary.FileWriter – This attribute represents a TensorFlow's Writer, that is used to obtain stats.

is_sparse
bool – Use sparse Tensors if it's set to True. Not implemented yet. Show the Todo.

G
tf_G.Graph – The graph on which the transition is referred.

transition
tf.Variable – The 2-D *tf.Tensor* with the same shape as adjacency matrix of the graph, that represents the probabilities to move from one vertex to another.

beta
float – The reset probability of the random walks, i.e. the probability that a user that surfs the graph an decides to jump to another vertex not connected to the current.

__init__(*sess: tensorflow.python.client.session.Session*, *name: str*, *graph: tf_G.graph.graph.Graph*, *beta: float*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *is_sparse: bool = False*) → None
Constructor of the class.

This method is called to create a new instance of Transition class.

Parameters

- **sess** (tf.Session) – This attribute represents the session that runs the TensorFlow operations.
- **name** (*str*) – This attribute represents the name of the object in TensorFlow's op Graph.
- **graph** (tf_G.Graph) – The graph on which the transition is referred.
- **beta** (*float*) – The reset probability of the random walks, i.e. the probability that a user that surfs the graph an decides to jump to another vertex not connected to the current.
- **writer** (tf.summary.FileWriter) – This attribute represents a TensorFlow's Writer, that is used to obtain stats.
- **is_sparse** (*bool*) – Use sparse Tensors if it's set to True. Not implemented yet. Show the Todo.

get_tf(**args*, ***kwargs*)

The method that returns the transition Tensor.

This method will return the transition matrix of the graph.

Parameters

- ***args** – The args of the *get_tf()* method.
- ****kwargs** – The kwargs of the *get_tf()* method.

Returns

A *tf.Tensor* that contains the distribution of transitions over vertices of the graph.

Return type (tf.Tensor)

update_edge(*edge: numpy.ndarray*, *change: float*) → None

The callback to receive notifications about edge changes in the graph.

This method is called from the Graph when an addition or deletion is produced on the edge set. So probably is necessary to recompute the transition matrix.

Parameters

- **edge** (np.ndarray) – A 1-D *np.ndarray* that represents the edge that changes in the graph, where *edge[0]* is the source vertex, and *edge[1]* the destination vertex.
- **change** (*float*) – The variation of the edge weight. If the final value is 0.0 then the edge is removed.

Returns This method returns nothing.

CHAPTER 7

tf_G.utils

tf_G.utils Module

It contains a set of utilities that is used for another classes of *tf_G* module.

DataSets

class `tf_G.utils.datasets.`**`DataSets`**

DataSets class represents some data sets included in the package.

The class provides *compose_from_path* method to import personal sets.

static `compose_from_path` (*path: str*, *index_decrement: bool*) → numpy.ndarray

This method generates a data set from a given path.

The method obtains the data from the given path, then decrements its values if is necessary and permutes the resulting data set.

The decrement option is offered because of in some cases the data set treats the initial node as 1 but many data structures in python are 0-indexed, so decrementing the values improves space performance.

Parameters

- **`path`** (*str*) – The path of the file of data set csv.
- **`index_decrement`** (*bool*) – Decrements all valus by one if True, do nothing otherwise.

Returns The data set that represents the Graph.

Return type (`np.ndarray`)

static `naive_4` () → numpy.ndarray

This method returns the naive_4 data set.

The data set is obtained from Cornell University guide lecture of PageRank algorithm.

This graph contains 4 vertices and 8 edges.

Url: http://www.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture3/lecture3.html

Returns The data set that represents the Graph.

Return type (np.ndarray)

static naive_6() → numpy.ndarray
This method returns the naive_6 data set.

The data set is obtained from mathworks study of PageRank algorithm.

This graph contains 6 vertices and 9 edges.

Url: https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/moler/exm/chapters/pagerank.pdf

Returns The data set that represents the Graph.

Return type (np.ndarray)

static permute_edges(*edges_np: numpy.ndarray*) → numpy.ndarray
Method that permutes the rows order of given the input set.

Parameters **edges_np** (np.ndarray) – The input data set.

Returns The input data set permuted in rows

Return type (np.ndarray)

TensorFlowObject

class tf_G.utils.tensorflow_object.**TensorFlowObject**(*sess: tensorflow.python.client.session.Session*, *name: str*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *is_sparse: bool = False*) → None
This class gives represents a TensorFlow object in the package.

It acts as Parent class of many classes that uses the TensorFlow library and needs to execute code, so it's necessary to have a session and other attributes.

__init__(*sess: tensorflow.python.client.session.Session*, *name: str*, *writer: tensorflow.python.summary.writer.writer.FileWriter = None*, *is_sparse: bool = False*) → None
The constructor of class.

It assign the input parameters to the class objects and no more.

Parameters

- **sess** (tf.Session) – This attribute represents the session that runs the TensorFlow operations.
- **name** (*str*) – This attribute represents the name of the object in TensorFlow's op Graph.
- **writer** (tf.summary.FileWriter) – This attribute represents a TensorFlow's Writer, that is used to obtain stats.
- **is_sparse** (*bool*) – Use sparse Tensors if it's set to True. Not implemented yet. Show the Todo.

Todo

•Implement variables as sparse when it's possible. Waiting to TensorFlow for it.

run_tf (*input_to_run*)
Run method to execute TensorFlow operations

Parameters **input_to_run** – This parameter represents a TensorFlow operation.

Returns The result of the operation as numpy array

Utils

class tf_G.utils.utils.**Utils**
Utils class of the tf_G package.

This class contains static methods that will be used by another classes around the package.

static ranked (*x: numpy.ndarray*) → numpy.ndarray
This method sorts the array indices given by its values.

It can be used to generate a ranking based on the values of an array in which the value represents the score and the index the object identifier.

Parameters **x** (np.ndarray) – A 2-D *np.ndarray* to rank the results by rows.

Returns

An array containing the indices of the input array sorted in decremental order.

Return type (np.ndarray)

static save_ranks (*filename: str*, *array: numpy.ndarray*, *index_increment: bool = True*) → None
This method will save the input array in the filesystem.

The method creates a file in the filesystem with name *filename* and puts the array content inside it.

This method provides the *index_increment* that increments the first column by one if True. The reason of this option is that the method is created to store results of graph operations, and in some cases the graph vertices is represented as 0-indexed and anothers as 1-indexed.

Parameters

- **filename** (*str*) – The name of the file that will be created.
- **array** (np.ndarray) – The array that contains the data that will be saved.
- **index_increment** (*bool, optional*) – Increments the first column of the array input if True, do nothing otherwise.

Returns This method returns nothing.

CHAPTER 8

tf_G.utils.callbacks

tf_G.utils.callbacks Module

It contains a set of classes that helps on notifications of graph changes.

UpdateEdgeListener

class `tf_G.utils.callbacks.update_edge_listener.`**`UpdateEdgeListener`**

Listen notifications related with a change in graph edges.

The graph (or another class that wants to receive notifications from a edge change) inherits this class and when an edge changes it will receive notifications from the change in edge set of a graph.

The classes that inherit this class need to implement *update_edge(edge,change)* method.

`update_edge`(*edge: numpy.ndarray*, *change: float*)

The callback to receive notifications about edge changes in the graph.

This method is called from the Graph when an addition or deletion is produced on the edge set. So probably is necessary to recompute the PageRank ranking.

Parameters

- **`edge`** (`np.ndarray`) – A 1-D *np.ndarray* that represents the edge that changes in the graph, where *edge[0]* is the source vertex, and *edge[1]* the destination vertex.
- **`change`** (*`float`*) – The variation of the edge weight. If the final value is 0.0 then the edge is removed.

Returns This method returns nothing.

UpdateEdgeNotifier

class tf_G.utils.callbacks.update_edge_notifier.**UpdateEdgeNotifier**

This class is used to notify another classes that a change in graph edges.

The graph (or another class that wants to notify an edge change) inherits this class and when an edge changes it will notify this change to all the attached objects.

The objects attached to this class need to implement *update_edge(edge,change)* method.

_listeners

set – The set of objects that will be notified when an edge modifies it weight.

__init__()

Constructor of UpdateEdgeNotifier

The set of listeners is initialised.

attach(*listener: tf_G.utils.callbacks.update_edge_listener.UpdateEdgeListener*)

Method to attach objects from this class notifications.

Parameters **listener** (tf_G.UpdateEdgeListener) – An object that will start being notified when the graph changes its edge set.

Returns This method returns nothing.

detach(*listener: tf_G.utils.callbacks.update_edge_listener.UpdateEdgeListener*)

Method to detach objects from this clas notifications.

Parameters **listener** (tf_G.UpdateEdgeListener) – An object that will stop being notified when the graph changes its edge set.

Returns This method returns nothing.

CHAPTER 9

tf_G.utils.math

tf_G.utils.math Module

It contains some math utilities needed to another classes of the tf_G module.

ConvergenceCriterion

class `tf_G.utils.math.convergence_criterion.`**`ConvergenceCriterion`**

Enum class that contains some convergence criteria.

The class has lambda functions that can be accessed as static class methods because it inherits Enum class.

The methods operates with two vectors and a convergence bound value.

`INFINITY`(*i=None*, *x=None*, *y=None*, *c=None*, *n=None*, *dist=None*)

INFINITY convergence criterion.

The one convergence criterion uses the norm infinity to gets the difference between two vectors and the compare it with the convergence bound c.

Returns

Tensor that returns True if the difference of the vectors is inside the convergence criterion interval, and False otherwise.

Return type (`tf.Tensor`)

`ONE`(*i=None*, *x=None*, *y=None*, *c=None*, *n=None*, *dist=None*)

ONE convergence criterion.

The one convergence criterion uses the norm one to gets the difference between two vectors and the compare it with the convergence bound c.

Returns

Tensor that returns True if the difference of the vectors is inside the convergence criterion interval, and False otherwise.

Return type (tf.Tensor)

VectorNorm

class tf_G.utils.math.vector_norm.**VectorNorm**

Enum class that contains some vectorial norms.

The class has lambda functions that can be accessed as static class methods because it inherits Enum class.

The methods operates with a vector with 1-D rank.

EUCLIDEAN(*x*)

EUCLIDEAN norm.

The euclidean norm of the vector x.

Returns Tensor that contains the eucledian norm of the vector.

Return type (tf.Tensor)

INFINITY(*x*)

INFINITY norm.

The infinity norm of the vector x.

Returns Tensor that contains the infinity norm of the vector.

Return type (tf.Tensor)

ONE(*x*)

ONE norm.

The one norm of the vector x.

Returns Tensor that contains the one norm of the vector.

Return type (tf.Tensor)

P_NORM(*x*, *p*)

P-NORM norm.

The p-norm of the vector x.

Returns Tensor that contains the p-norm of the vector.

Return type (tf.Tensor)

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index

Symbols

A

B

C

D

E