

kmeans

Generated by Doxygen 1.13.2

| | |
|--|-----------|
| 1 Namespace Index | 1 |
| 1.1 Namespace List | 1 |
| 2 Class Index | 3 |
| 2.1 Class List | 3 |
| 3 File Index | 5 |
| 3.1 File List | 5 |
| 4 Namespace Documentation | 7 |
| 4.1 DM Namespace Reference | 7 |
| 4.2 Kmeans Namespace Reference | 7 |
| 4.3 KmeansParser Namespace Reference | 7 |
| 5 Class Documentation | 9 |
| 5.1 DM::DistanceMetrics Class Reference | 9 |
| 5.1.1 Detailed Description | 9 |
| 5.1.2 Member Function Documentation | 9 |
| 5.1.2.1 euclideanDistance() | 9 |
| 5.2 Kmeans::Kmeans Class Reference | 10 |
| 5.2.1 Detailed Description | 10 |
| 5.2.2 Member Function Documentation | 10 |
| 5.2.2.1 add() | 10 |
| 5.2.2.2 computeDistance() | 11 |
| 5.2.2.3 computeNewCenters() | 11 |
| 5.2.2.4 div() | 11 |
| 5.2.2.5 getInitialCenters() | 12 |
| 5.2.2.6 minIndex() | 12 |
| 5.3 KmeansParser::Reader Class Reference | 13 |
| 5.3.1 Detailed Description | 13 |
| 5.3.2 Constructor & Destructor Documentation | 13 |
| 5.3.2.1 Reader() | 13 |
| 5.3.3 Member Function Documentation | 13 |
| 5.3.3.1 readAndParse() | 13 |
| 6 File Documentation | 15 |
| 6.1 app/main.cpp File Reference | 15 |
| 6.1.1 Detailed Description | 15 |
| 6.1.2 Function Documentation | 15 |
| 6.1.2.1 main() | 15 |
| 6.2 include/distanceMetrics.hpp File Reference | 16 |
| 6.2.1 Detailed Description | 16 |
| 6.3 distanceMetrics.hpp | 16 |
| 6.4 include/kmeans.hpp File Reference | 16 |

| | |
|--|-----------|
| 6.4.1 Detailed Description | 17 |
| 6.5 kmeans.hpp | 17 |
| 6.6 include/read.hpp File Reference | 17 |
| 6.6.1 Detailed Description | 18 |
| 6.7 read.hpp | 18 |
| 6.8 src/distanceMetrics.cpp File Reference | 18 |
| 6.8.1 Detailed Description | 18 |
| 6.9 src/kmeans.cpp File Reference | 19 |
| 6.9.1 Detailed Description | 19 |
| 6.10 src/read.cpp File Reference | 19 |
| 6.10.1 Detailed Description | 19 |
| Index | 21 |

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

| | | |
|------------------------------|-------|-------------------|
| DM | | 7 |
| Kmeans | | 7 |
| KmeansParser | | 7 |

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

| | |
|---|----|
| DM::DistanceMetrics | |
| Class providing distance metric functions | 9 |
| Kmeans::Kmeans | |
| Class implementing the K-Means clustering algorithm | 10 |
| KmeansParser::Reader | |
| Class to read and parse data points from a file | 13 |

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

| | | |
|---|---|----|
| app/main.cpp | Main entry point for the K-Means clustering algorithm in C++ | 15 |
| include/distanceMetrics.hpp | Header file for distance metric calculations | 16 |
| include/kmeans.hpp | Header file for the K-Means clustering algorithm implementation | 16 |
| include/read.hpp | Header file for reading and parsing input data for K-Means clustering | 17 |
| src/distanceMetrics.cpp | Implementation of distance metric functions | 18 |
| src/kmeans.cpp | Implementation of the K-Means clustering algorithm | 19 |
| src/read.cpp | Implementation of file reading and parsing for K-Means clustering | 19 |

Chapter 4

Namespace Documentation

4.1 DM Namespace Reference

Classes

- class [DistanceMetrics](#)
Class providing distance metric functions.

4.2 Kmeans Namespace Reference

Classes

- class [Kmeans](#)
Class implementing the K-Means clustering algorithm.

4.3 KmeansParser Namespace Reference

Classes

- class [Reader](#)
Class to read and parse data points from a file.

Chapter 5

Class Documentation

5.1 DM::DistanceMetrics Class Reference

Class providing distance metric functions.

```
#include <distanceMetrics.hpp>
```

Public Member Functions

- double [euclideanDistance](#) (const std::vector< double > &vec1, const std::vector< double > &vec2)
Computes the Euclidean distance between two vectors.

5.1.1 Detailed Description

Class providing distance metric functions.

5.1.2 Member Function Documentation

5.1.2.1 euclideanDistance()

```
double DM::DistanceMetrics::euclideanDistance (
    const std::vector< double > & vec1,
    const std::vector< double > & vec2)
```

Computes the Euclidean distance between two vectors.

Parameters

| | |
|-------------|----------------|
| <i>vec1</i> | First vector. |
| <i>vec2</i> | Second vector. |

Returns

Euclidean distance between the vectors.

The documentation for this class was generated from the following files:

- include/[distanceMetrics.hpp](#)
- src/[distanceMetrics.cpp](#)

5.2 Kmeans::Kmeans Class Reference

Class implementing the K-Means clustering algorithm.

```
#include <kmeans.hpp>
```

Public Member Functions

- `std::vector< std::vector< double > > getInitialCenters (const std::vector< std::vector< double > > &points, const int &k)`
Selects initial cluster centers from the data points.
- `std::vector< std::vector< double > > computeDistance (const std::vector< std::vector< double > > &points, const std::vector< std::vector< double > > ¢ers, double(DM::DistanceMetrics::*func)(const std::vector< double > &, const std::vector< double > &), DM::DistanceMetrics &obj)`
Computes distances between points and centers using a specified metric.
- `int minIndex (const std::vector< double > &distances)`
Finds the index of the minimum value in a distance vector.
- `std::vector< std::vector< double > > computeNewCenters (const std::vector< std::vector< double > > &points, const std::vector< std::vector< double > > &distances, int k)`
Computes new cluster centers based on point assignments.
- `std::vector< double > add (const std::vector< double > &a, const std::vector< double > &b)`
Adds two vectors element-wise.
- `std::vector< double > div (const std::vector< double > &a, int n)`
Divides a vector by a scalar.

5.2.1 Detailed Description

Class implementing the K-Means clustering algorithm.

5.2.2 Member Function Documentation

5.2.2.1 `add()`

```
std::vector< double > Kmeans::Kmeans::add (
    const std::vector< double > & a,
    const std::vector< double > & b)
```

Adds two vectors element-wise.

Parameters

| | |
|----------|----------------|
| <i>a</i> | First vector. |
| <i>b</i> | Second vector. |

Returns

Resulting vector from the addition.

5.2.2.2 computeDistance()

```
std::vector< std::vector< double > > Kmeans::Kmeans::computeDistance (
    const std::vector< std::vector< double > > & points,
    const std::vector< std::vector< double > > & centers,
    double (DM::DistanceMetrics::* func) (const std::vector< double > &, const std::
::vector< double > &),
    DM::DistanceMetrics & obj)
```

Computes distances between points and centers using a specified metric.

Parameters

| | |
|----------------|--|
| <i>points</i> | List of data points. |
| <i>centers</i> | Current cluster centers. |
| <i>func</i> | Pointer to the distance metric function. |
| <i>obj</i> | Reference to the DistanceMetrics object. |

Returns

A vector of distance vectors, one per point.

5.2.2.3 computeNewCenters()

```
std::vector< std::vector< double > > Kmeans::Kmeans::computeNewCenters (
    const std::vector< std::vector< double > > & points,
    const std::vector< std::vector< double > > & distances,
    int k)
```

Computes new cluster centers based on point assignments.

Parameters

| | |
|------------------|-----------------------------------|
| <i>points</i> | List of data points. |
| <i>distances</i> | Distances from points to centers. |
| <i>k</i> | Number of clusters. |

Returns

Updated cluster centers.

5.2.2.4 div()

```
std::vector< double > Kmeans::Kmeans::div (
    const std::vector< double > & a,
    int n)
```

Divides a vector by a scalar.

Parameters

| | |
|----------|---------------------|
| <i>a</i> | Vector to divide. |
| <i>n</i> | Scalar denominator. |

Returns

Resulting vector after division.

5.2.2.5 getInitialCenters()

```
std::vector< std::vector< double > > Kmeans::Kmeans::getInitialCenters (
    const std::vector< std::vector< double > > & points,
    const int & k)
```

Selects initial cluster centers from the data points.

Parameters

| | |
|---------------|--------------------------|
| <i>points</i> | List of all data points. |
| <i>k</i> | Number of clusters. |

Returns

Initial centers as a vector of vectors.

5.2.2.6 minIndex()

```
int Kmeans::Kmeans::minIndex (
    const std::vector< double > & distances)
```

Finds the index of the minimum value in a distance vector.

Parameters

| | |
|------------------|----------------------|
| <i>distances</i> | Vector of distances. |
|------------------|----------------------|

Returns

Index of the minimum distance.

The documentation for this class was generated from the following files:

- [include/kmeans.hpp](#)
- [src/kmeans.cpp](#)

5.3 KmeansParser::Reader Class Reference

Class to read and parse data points from a file.

```
#include <read.hpp>
```

Public Member Functions

- [Reader](#) (std::string fileName)
Constructor for the [Reader](#) class.
- std::vector< std::vector< double > > [readAndParse](#) ()
Reads and parses the file into a vector of points.

5.3.1 Detailed Description

Class to read and parse data points from a file.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Reader()

```
KmeansParser::Reader::Reader (  
    std::string fileName)
```

Constructor for the [Reader](#) class.

Parameters

| | |
|-----------------|---------------------------|
| <i>fileName</i> | Name of the file to read. |
|-----------------|---------------------------|

5.3.3 Member Function Documentation

5.3.3.1 readAndParse()

```
std::vector< std::vector< double > > KmeansParser::Reader::readAndParse ()
```

Reads and parses the file into a vector of points.

Returns

A vector of vectors, where each inner vector represents a point.

The documentation for this class was generated from the following files:

- include/[read.hpp](#)
- src/[read.cpp](#)

Chapter 6

File Documentation

6.1 app/main.cpp File Reference

Main entry point for the K-Means clustering algorithm in C++.

```
#include <vector>
#include <iostream>
#include <chrono>
#include "read.hpp"
#include "distanceMetrics.hpp"
#include "kmeans.hpp"
```

Functions

- `int main ()`
Main function to execute K-Means clustering.

6.1.1 Detailed Description

Main entry point for the K-Means clustering algorithm in C++.

This file drives the K-Means clustering process by reading data points from a file, initializing centers, and iterating until convergence. It measures and outputs the execution time.

6.1.2 Function Documentation

6.1.2.1 main()

```
int main ()
```

Main function to execute K-Means clustering.

Returns

0 on successful execution.

6.2 include/distanceMetrics.hpp File Reference

Header file for distance metric calculations.

```
#include <vector>
```

Classes

- class [DM::DistanceMetrics](#)
Class providing distance metric functions.

Namespaces

- namespace [DM](#)

6.2.1 Detailed Description

Header file for distance metric calculations.

Defines the DistanceMetrics class in the [DM](#) namespace, providing methods for distance computation.

6.3 distanceMetrics.hpp

[Go to the documentation of this file.](#)

```
00001
00008
00009 #pragma once
00010 #include <vector>
00011
00012 namespace DM {
00016     class DistanceMetrics {
00017     public:
00024         double euclideanDistance(const std::vector<double>& vec1, const std::vector<double>&
vec2);
00025     };
00026 }
```

6.4 include/kmeans.hpp File Reference

Header file for the K-Means clustering algorithm implementation.

```
#include <vector>
#include "distanceMetrics.hpp"
```

Classes

- class [Kmeans::Kmeans](#)
Class implementing the K-Means clustering algorithm.

Namespaces

- namespace [Kmeans](#)

6.4.1 Detailed Description

Header file for the K-Means clustering algorithm implementation.

Defines the [Kmeans](#) class in the [Kmeans](#) namespace, providing methods for clustering.

6.5 kmeans.hpp

[Go to the documentation of this file.](#)

```

00001
00007
00008 #pragma once
00009 #include <vector>
00010 #include "distanceMetrics.hpp"
00011
00012 namespace Kmeans {
00016     class Kmeans {
00017     public:
00024         std::vector<std::vector<double>> getInitialCenters(const std::vector<std::vector<double>>&
points, const int& k);
00025
00034         std::vector<std::vector<double>> computeDistance(const std::vector<std::vector<double>>&
points,
00035             const std::vector<std::vector<double>>& centers,
00036             double (DM::DistanceMetrics::*func)(const std::vector<double>>&, const
std::vector<double>>&),
00037             DM::DistanceMetrics& obj);
00038
00044         int minIndex(const std::vector<double>& distances);
00045
00053         std::vector<std::vector<double>> computeNewCenters(const std::vector<std::vector<double>>&
points,
00054             const std::vector<std::vector<double>>& distances, int k);
00055
00062         std::vector<double> add(const std::vector<double>& a, const std::vector<double>& b);
00063
00070         std::vector<double> div(const std::vector<double>& a, int n);
00071     };
00072 }

```

6.6 include/read.hpp File Reference

Header file for reading and parsing input data for K-Means clustering.

```

#include <string>
#include <vector>

```

Classes

- class [KmeansParser::Reader](#)

Class to read and parse data points from a file.

Namespaces

- namespace [KmeansParser](#)

6.6.1 Detailed Description

Header file for reading and parsing input data for K-Means clustering.

Defines the Reader class in the [KmeansParser](#) namespace, responsible for reading data points from a file.

6.7 read.hpp

[Go to the documentation of this file.](#)

```
00001
00002
00009 #pragma once
00010 #include <string>
00011 #include <vector>
00012
00013 namespace KmeansParser {
00014     class Reader {
00015     private:
00016         std::string fileName{};
00017         std::vector<std::vector<double>> all_points{};
00018     public:
00019         Reader(std::string fileName);
00020
00021         std::vector<std::vector<double>> readAndParse();
00022     };
00023 }
00024 }
```

6.8 src/distanceMetrics.cpp File Reference

Implementation of distance metric functions.

```
#include <cmath>
#include <vector>
#include "distanceMetrics.hpp"
```

Namespaces

- namespace [DM](#)

6.8.1 Detailed Description

Implementation of distance metric functions.

Provides the implementation for the DistanceMetrics class, specifically the Euclidean distance calculation.

6.9 src/kmeans.cpp File Reference

Implementation of the K-Means clustering algorithm.

```
#include <vector>
#include <limits.h>
#include "kmeans.hpp"
#include "distanceMetrics.hpp"
```

Namespaces

- namespace [Kmeans](#)

6.9.1 Detailed Description

Implementation of the K-Means clustering algorithm.

Provides the implementation for the [Kmeans](#) class, handling center initialization, distance computation, and center updates.

6.10 src/read.cpp File Reference

Implementation of file reading and parsing for K-Means clustering.

```
#include <string>
#include <fstream>
#include <vector>
#include <sstream>
#include "read.hpp"
```

Namespaces

- namespace [KmeansParser](#)

6.10.1 Detailed Description

Implementation of file reading and parsing for K-Means clustering.

Provides the implementation for the Reader class, reading data points from a file into a vector of vectors.

Index

- add
 - Kmeans::Kmeans, [10](#)
- app/main.cpp, [15](#)
- computeDistance
 - Kmeans::Kmeans, [10](#)
- computeNewCenters
 - Kmeans::Kmeans, [11](#)
- div
 - Kmeans::Kmeans, [11](#)
- DM, [7](#)
- DM::DistanceMetrics, [9](#)
 - euclideanDistance, [9](#)
- euclideanDistance
 - DM::DistanceMetrics, [9](#)
- getInitialCenters
 - Kmeans::Kmeans, [12](#)
- include/distanceMetrics.hpp, [16](#)
- include/kmeans.hpp, [16](#), [17](#)
- include/read.hpp, [17](#), [18](#)
- Kmeans, [7](#)
- Kmeans::Kmeans, [10](#)
 - add, [10](#)
 - computeDistance, [10](#)
 - computeNewCenters, [11](#)
 - div, [11](#)
 - getInitialCenters, [12](#)
 - minIndex, [12](#)
- KmeansParser, [7](#)
- KmeansParser::Reader, [13](#)
 - readAndParse, [13](#)
 - Reader, [13](#)
- main
 - main.cpp, [15](#)
- main.cpp
 - main, [15](#)
- minIndex
 - Kmeans::Kmeans, [12](#)
- readAndParse
 - KmeansParser::Reader, [13](#)
- Reader
 - KmeansParser::Reader, [13](#)
- src/distanceMetrics.cpp, [18](#)
- src/kmeans.cpp, [19](#)
- src/read.cpp, [19](#)