

Classification Algorithms System

V0.1

Generated by Doxygen 1.8.13

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	Classifier Class Reference	7
4.1.1	Member Function Documentation	9
4.1.1.1	evaluate()	9
4.1.1.2	getSteps()	9
4.1.1.3	getUpdates()	10
4.1.1.4	setSamples()	10
4.1.1.5	train()	10
4.2	Data Class Reference	10
4.2.1	Detailed Description	12
4.2.2	Constructor & Destructor Documentation	12
4.2.2.1	Data() [1/2]	12
4.2.2.2	Data() [2/2]	13
4.2.3	Member Function Documentation	13
4.2.3.1	changeXVector()	13
4.2.3.2	copy()	13

4.2.3.3	copyZero()	14
4.2.3.4	getDim()	14
4.2.3.5	getFeaturesNames()	14
4.2.3.6	getIndex()	14
4.2.3.7	getNumberNegativePoints()	15
4.2.3.8	getNumberPositivePoints()	15
4.2.3.9	getPoint()	15
4.2.3.10	getPoints()	15
4.2.3.11	getSize()	16
4.2.3.12	getStatistics()	16
4.2.3.13	insertFeatures()	16
4.2.3.14	insertPoint() [1/2]	16
4.2.3.15	insertPoint() [2/2]	17
4.2.3.16	isEmpty()	17
4.2.3.17	isNormalized()	17
4.2.3.18	join()	18
4.2.3.19	load()	18
4.2.3.20	normalize() [1/2]	18
4.2.3.21	normalize() [2/2]	19
4.2.3.22	removeFeatures()	19
4.2.3.23	removePoint()	19
4.2.3.24	removePoints()	20
4.2.3.25	setClasses()	20
4.2.3.26	setDim()	20
4.2.3.27	setFeaturesNames()	20
4.2.3.28	setPoint()	21
4.2.3.29	write()	21
4.3	DualClassifier Class Reference	22
4.3.1	Member Function Documentation	23
4.3.1.1	getSolution()	23

4.3.1.2	setKernel()	23
4.4	FeatureSelection Class Reference	23
4.5	Gnuplot Class Reference	24
4.5.1	Member Function Documentation	26
4.5.1.1	is_valid()	26
4.5.1.2	operator<<()	26
4.5.1.3	plot_equation()	27
4.5.1.4	plot_equation3d()	27
4.5.1.5	plot_image()	27
4.5.1.6	plot_x()	28
4.5.1.7	plot_xy()	28
4.5.1.8	plot_xy_err()	28
4.5.1.9	plotfile_x()	28
4.5.1.10	plotfile_xy()	28
4.5.1.11	plotfile_xy_err()	29
4.5.1.12	plotfile_xyz()	29
4.5.1.13	replot()	29
4.5.1.14	set_contour()	29
4.5.1.15	set_GNUPlotPath()	29
4.5.1.16	set_hidden3d()	30
4.5.1.17	set_legend()	30
4.5.1.18	set_multiplot()	30
4.5.1.19	set_smooth()	31
4.5.1.20	set_style()	31
4.5.1.21	set_surface()	31
4.5.1.22	set_terminal_std()	31
4.5.1.23	set_title()	32
4.5.1.24	set_xautoscale()	32
4.5.1.25	set_yautoscale()	32
4.5.1.26	set_zautoscale()	33

4.5.1.27	unset_contour()	33
4.5.1.28	unset_hidden3d()	33
4.5.1.29	unset_legend()	34
4.5.1.30	unset_multiplot()	34
4.5.1.31	unset_smooth()	34
4.5.1.32	unset_surface()	35
4.5.1.33	unset_title()	35
4.5.1.34	unset_xlogscale()	35
4.5.1.35	unset_ylogscale()	36
4.5.1.36	unset_zlogscale()	36
4.6	GnuplotException Class Reference	36
4.6.1	Detailed Description	37
4.7	Kernel Class Reference	38
4.7.1	Detailed Description	38
4.7.2	Constructor & Destructor Documentation	38
4.7.2.1	Kernel()	38
4.7.3	Member Function Documentation	39
4.7.3.1	compute()	39
4.7.3.2	function()	39
4.7.3.3	getKernelMatrix()	39
4.7.3.4	norm()	40
4.7.3.5	setKernelMatrix()	40
4.7.3.6	setParam()	40
4.7.3.7	setType()	41
4.8	PerceptronDual Class Reference	41
4.8.1	Detailed Description	42
4.8.2	Member Function Documentation	42
4.8.2.1	evaluate()	43
4.8.2.2	train()	43
4.9	PerceptronFixedMarginDual Class Reference	43

4.9.1	Detailed Description	45
4.9.2	Member Function Documentation	45
4.9.2.1	evaluate()	45
4.9.2.2	train()	45
4.10	PerceptronFixedMarginPrimal Class Reference	46
4.10.1	Detailed Description	47
4.10.2	Member Function Documentation	47
4.10.2.1	evaluate()	48
4.10.2.2	train()	48
4.11	PerceptronPrimal Class Reference	48
4.11.1	Detailed Description	50
4.11.2	Member Function Documentation	50
4.11.2.1	evaluate()	50
4.11.2.2	train()	50
4.12	Point Class Reference	51
4.12.1	Detailed Description	51
4.12.2	Member Function Documentation	51
4.12.2.1	dot()	51
4.12.2.2	norm()	52
4.13	PrimalClassifier Class Reference	52
4.13.1	Member Function Documentation	53
4.13.1.1	getSolution()	54
4.13.1.2	setNorm()	54
4.14	Solution Class Reference	54
4.15	Statistics Class Reference	55
4.15.1	Detailed Description	56
4.15.2	Member Function Documentation	56
4.15.2.1	getDistCenters()	56
4.15.2.2	getDistCentersWithoutFeats()	57
4.15.2.3	getFeatureMean()	57

4.15.2.4	getFeatureStdev()	57
4.15.2.5	getRadius()	58
4.15.2.6	mean()	58
4.15.2.7	stdev()	59
4.15.2.8	variance() [1/2]	59
4.15.2.9	variance() [2/2]	59
4.16	Validation Class Reference	60
4.16.1	Detailed Description	60
4.16.2	Member Function Documentation	60
4.16.2.1	partTrainTest()	60
4.17	ValidationSolution Class Reference	61
4.17.1	Detailed Description	62
4.18	Visualisation Class Reference	62
4.18.1	Detailed Description	62
4.18.2	Member Function Documentation	62
4.18.2.1	plot2D()	62
4.18.2.2	plot2DwithHyperplane()	63
4.18.2.3	plot3D()	63
4.18.2.4	plot3DwithHyperplane()	64
4.18.2.5	setSample()	64
4.18.2.6	setStyle()	64
4.18.2.7	setTitle()	65
5	File Documentation	67
5.1	includes/Utils.hpp File Reference	67
5.1.1	Detailed Description	68
5.1.2	Function Documentation	68
5.1.2.1	dtoa()	68
5.1.2.2	is_number()	69
5.1.2.3	itos()	69
5.1.2.4	maxAbsElement()	69
5.1.2.5	stodn()	71
5.1.2.6	stoin()	71
5.2	src/Data.cpp File Reference	71
5.2.1	Detailed Description	72
5.3	src/Point.cpp File Reference	72
5.3.1	Detailed Description	73
5.4	src/Utils.cpp File Reference	73
5.4.1	Detailed Description	74
5.4.2	Function Documentation	75
5.4.2.1	dtoa()	75
5.4.2.2	itos()	75
Index		77

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Classifier	7
DualClassifier	22
PerceptronDual	41
PerceptronFixedMarginDual	43
PrimalClassifier	52
PerceptronFixedMarginPrimal	46
PerceptronPrimal	48
Data	10
FeatureSelection	23
Gnuplot	24
Kernel	38
Point	51
runtime_error	
GnuplotException	36
Solution	54
ValidationSolution	61
Statistics	55
Validation	60
Visualisation	62

Chapter 2

Class Index

2.1 Class List

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

Classifier	7
Data	
Wrapper for the dataset data	10
DualClassifier	22
FeatureSelection	23
Gnuplot	24
GnuplotException	
A C++ interface to gnuplot	36
Kernel	
Class for the kernel computations	38
PerceptronDual	
Wrapper for the implementation of the Perceptron dual algorithm	41
PerceptronFixedMarginDual	
Wrapper for the implementation of the Perceptron dual with fixed margin algorithm	43
PerceptronFixedMarginPrimal	
Wrapper for the implementation of the Perceptron primal with fixed margin algorithm	46
PerceptronPrimal	
Wrapper for the implementation of the Perceptron primal algorithm	48
Point	
Class of a Point of doubles in a space of n dimensions	51
PrimalClassifier	52
Solution	54
Statistics	
Class with methods for statistical computations	55
Validation	
Class of methods for the validation of ML algorithms	60
ValidationSolution	
Solution for the validation of a ML method	61
Visualisation	
Class for visualize data using gnuplot	62

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

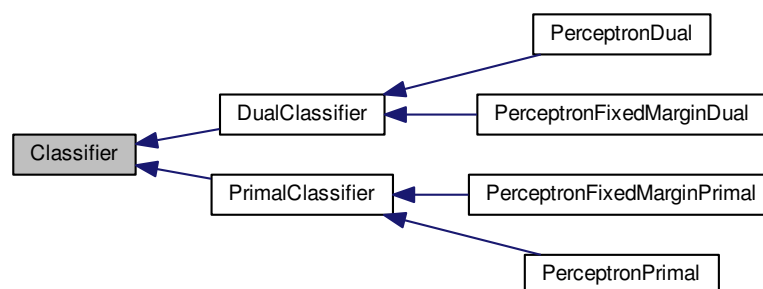
includes/ Classifier.hpp	??
includes/ Data.hpp	??
includes/ DualClassifier.hpp	??
includes/ FeatureSelection.hpp	??
includes/ gnuplot_i.hpp	??
includes/ Kernel.hpp	??
includes/ MLToolkit.hpp	??
includes/ Perceptron.hpp	??
includes/ Point.hpp	??
includes/ PrimalClassifier.hpp	??
includes/ Solution.hpp	??
includes/ Statistics.hpp	??
includes/ Utils.hpp	67
includes/ Validation.hpp	??
includes/ ValidationSolution.hpp	??
includes/ Visualisation.hpp	??
src/ Data.cpp	
Implementation of the Data class methods	71
src/ Point.cpp	
Implementation of the Point class methods	72
src/ Utils.cpp	
Implementation of methods for general use in the system	73

Chapter 4

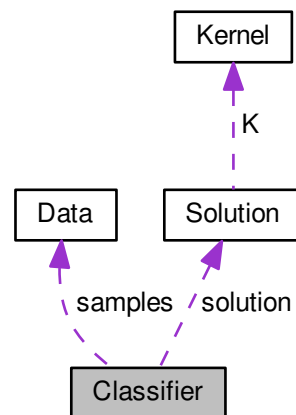
Class Documentation

4.1 Classifier Class Reference

Inheritance diagram for Classifier:



Collaboration diagram for Classifier:



Public Member Functions

- virtual bool **train** ()=0
Function that execute the training phase of a classification algorithm.
- virtual double **evaluate** (Point p)=0
Returns the class of a feature point based on the trained classifier.
- virtual void **setSamples** (Data *samples)
setSamples Set the samples used in the classifier.
- void **setStartTime** (double start_time)
- void **setMaxTime** (double max_time)
- void **setEPS** (double EPS)
- void **setMaxIterations** (int MAX_IT)
- void **setMaxUpdates** (int MAX_UP)
- int **getSteps** ()
getSteps Returns the number of steps through the data by the classifier.
- int **getUpdates** ()
getUpdates Returns the number of updates needed to get to the the solution.

Protected Attributes

- Data * **samples**
- std::vector< Point > **svs**
Support vectors points.
- Solution **solution**
Classifier solution.
- double **rate** = 0.5
Learning rate.
- double **start_time** = 0
Initial time.

- double `max_time` = 200
Maximum time of training.
- int `steps` = 0
Number of steps in the data.
- int `ctot` = 0
Number of updates of the weights.
- double `EPS` = 1E-9
Max precision.
- int `MAX_IT` = 1E9
- int `MAX_UP` = 1E9

4.1.1 Member Function Documentation

4.1.1.1 `evaluate()`

```
virtual double Classifier::evaluate (  
    Point p ) [pure virtual]
```

Returns the class of a feature point based on the trained classifier.

Parameters

<code>Point</code>	x (???) Features point to be evaluated.
--------------------	---

Returns

int

Implemented in [PerceptronFixedMarginDual](#), [PerceptronDual](#), [PerceptronFixedMarginPrimal](#), and [PerceptronPrimal](#).

4.1.1.2 `getSteps()`

```
int Classifier::getSteps ( )
```

`getSteps` Returns the number of steps through the data by the classifier.

Returns

int

4.1.1.3 getUpdates()

```
int Classifier::getUpdates ( )
```

getUpdates Returns the number of updates needed to get to the the solution.

Returns

int

4.1.1.4 setSamples()

```
void Classifier::setSamples (
    Data * samples ) [virtual]
```

setSamples Set the samples used in the classifier.

Parameters

<i>samples</i>	Samples to be used.
----------------	---------------------

4.1.1.5 train()

```
virtual bool Classifier::train ( ) [pure virtual]
```

Function that execute the training phase of a classification algorithm.

Returns

void

Implemented in [PerceptronFixedMarginDual](#), [PerceptronDual](#), [PerceptronFixedMarginPrimal](#), and [PerceptronPrimal](#).

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

- includes/Classifier.hpp
- src/Classifier.cpp

4.2 Data Class Reference

Wrapper for the dataset data.

```
#include <Data.hpp>
```

Public Member Functions

- [Data](#) (const char *pos_class="1", const char *neg_class="-1")
Constructor for empty data.
- [Data](#) (std::string dataset, const char *pos_class="1", const char *neg_class="-1")
Data constructor to load a dataset from a file.
- void [write](#) (std::string fname, std::string ext)
write Write the data to a file with the given extension.
- int [getSize](#) ()
Returns the size of the dataset.
- int [getDim](#) ()
Returns the dimension of the dataset.
- void [setDim](#) (int dim)
setDim Set the dimension of the points.
- [Point](#) [getPoint](#) (int index)
Returns the point with the given index.
- void [setPoint](#) (int index, [Point](#) p)
setPoint Set the point in a position of the data.
- std::vector< [Point](#) > [getPoints](#) ()
Returns the vector of Points of the sample.
- std::vector< int > [getFeaturesNames](#) ()
Returns the features names.
- void [setFeaturesNames](#) (std::vector< int > fnames)
setFeaturesNames Set the name of the features of the data.
- [Statistics](#) [getStatistics](#) ()
Returns a class with the statistics info of the sample.
- std::vector< int > [getIndex](#) ()
Returns the vector of indexes.
- int [getNumberPositivePoints](#) ()
Return the number of positive points.
- int [getNumberNegativePoints](#) ()
Return the number of negative points.
- void [setClasses](#) (std::string pos, std::string neg)
setClasses Set the classes of the dataset.
- bool [isEmpty](#) ()
Returns if there's a dataset loaded.
- bool [isNormalized](#) ()
Returns if the dataset is normalized.
- bool [load](#) (std::string file)
Load a dataset from a file.
- void [clear](#) ()
clear Clear the data.
- [Data](#) [copy](#) ()
Returns a copy of the data.
- [Data](#) [copyZero](#) ()
Returns a copy of the data with zero points.
- void [join](#) ([Data](#) data)
Merge one dataset with another.
- bool [insertPoint](#) ([Data](#) sample, int id)
Insert a point to the data from another sample.
- bool [insertPoint](#) ([Point](#) p)

- Insert a point to the end of vector points.*
 - `std::vector< bool > removePoints (std::vector< int > ids)`
- Remove several points from the sample.*
 - `bool removePoint (int pid)`
- Remove a point from the data.*
 - `Data insertFeatures (std::vector< int > ins_feat)`
- insertFeatures Returns Data object with only features in array.*
 - `bool removeFeatures (std::vector< int > feats)`
- Remove several features from the sample.*
 - `void changeXVector (std::vector< int > index)`
- Change the x vector of a sample.*
 - `void normalize (double p=2)`
- normalize Normalize the dataset using a Lp-norm.*
 - `void operator= (const Data &)`

Static Public Member Functions

- static void `normalize (std::vector< double > &p, double q)`
normalize Normalize a vector using a Lp-norm.

Friends

- `std::ostream & operator<< (std::ostream &output, const Data &data)`

4.2.1 Detailed Description

Wrapper for the dataset data.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Data() [1/2]

```
Data::Data (
    const char * pos_class = "1",
    const char * neg_class = "-1" )
```

Constructor for empty data.

Parameters

<i>pos_class</i>	String representing the positive class on the dataset.
<i>neg_class</i>	String representing the negative class on the dataset.

4.2.2.2 Data() [2/2]

```

Data::Data (
    std::string dataset,
    const char * pos_class = "1",
    const char * neg_class = "-1" )

```

[Data](#) constructor to load a dataset from a file.

Parameters

<i>dataset</i>	(???) Path to the dataset to be loaded.
<i>pos_class</i>	String representing the positive class on the dataset.
<i>neg_class</i>	String representing the negative class on the dataset.

4.2.3 Member Function Documentation

4.2.3.1 changeXVector()

```

void Data::changeXVector (
    std::vector< int > index )

```

Change the x vector of a sample.

Parameters

<i>index</i>	(???) Indexes of the change to be made.
--------------	---

Returns

void

4.2.3.2 copy()

```

Data Data::copy ( )

```

Returns a copy of the data.

Returns

[Data](#)

4.2.3.3 copyZero()

```
Data Data::copyZero ( )
```

Returns a copy of the data with zero points.

Returns

[Data](#)

4.2.3.4 getDim()

```
int Data::getDim ( )
```

Returns the dimension of the dataset.

Returns

int

4.2.3.5 getFeaturesNames()

```
vector< int > Data::getFeaturesNames ( )
```

Returns the features names.

Returns

std::vector<int>

4.2.3.6 getIndex()

```
vector< int > Data::getIndex ( )
```

Returns the vector of indexes.

Returns

std::vector<int>

4.2.3.7 `getNumberNegativePoints()`

```
int Data::getNumberNegativePoints ( )
```

Return the number of negative points.

Returns

int

4.2.3.8 `getNumberPositivePoints()`

```
int Data::getNumberPositivePoints ( )
```

Return the number of positive points.

Returns

int

4.2.3.9 `getPoint()`

```
Point Data::getPoint (
    int index )
```

Returns the point with the given index.

Parameters

<i>index</i>	Position of a point in the points array.
--------------	--

Returns

std::vector<Points>

4.2.3.10 `getPoints()`

```
vector< Point > Data::getPoints ( )
```

Returns the vector of Points of the sample.

Returns

std::vector<Points>

4.2.3.11 getSize()

```
int Data::getSize ( )
```

Returns the size of the dataset.

Returns

int

4.2.3.12 getStatistics()

```
Statistics Data::getStatistics ( )
```

Returns a class with the statistics info of the sample.

Returns

[Statistics](#)

4.2.3.13 insertFeatures()

```
Data Data::insertFeatures (
    std::vector< int > ins_feat )
```

insertFeatures Returns [Data](#) object with only features in array.

Parameters

<i>ins_feat</i>	(???) Array with features that will be in the Data object.
-----------------	--

Returns

[Data](#) If the object is empty something wrong happened.

4.2.3.14 insertPoint() [1/2]

```
bool Data::insertPoint (
    Data sample,
    int id )
```

Insert a point to the data from another sample.

Parameters

<i>sample</i>	(???) Sample with the point to be added.
<i>id</i>	(???) Index of the point to be added.

Returns

bool

4.2.3.15 insertPoint() [2/2]

```
bool Data::insertPoint (
    Point p )
```

Insert a point to the end of vector points.

Parameters

<i>p</i>	(???) Point to be inserted.
----------	---

Returns

bool

4.2.3.16 isEmpty()

```
bool Data::isEmpty ( )
```

Returns if there's a dataset loaded.

Returns

bool

4.2.3.17 isNormalized()

```
bool Data::isNormalized ( )
```

Returns if the dataset is normalized.

Returns

bool

4.2.3.18 join()

```
void Data::join (
    Data data )
```

Merge one dataset with another.

Parameters

<i>data</i>	(???) Dataset to be joined.
-------------	-----------------------------

Returns

bool

4.2.3.19 load()

```
bool Data::load (
    std::string file )
```

Load a dataset from a file.

Parameters

<i>file</i>	(???) Path to dataset file.
-------------	-----------------------------

Returns

bool

4.2.3.20 normalize() [1/2]

```
void Data::normalize (
    double p = 2 )
```

normalize Normalize the dataset using a Lp-norm.

Parameters

<i>p</i>	Norm to be utilized.
----------	----------------------

4.2.3.21 normalize() [2/2]

```
static void Data::normalize (
    std::vector< double > & p,
    double q ) [static]
```

normalize Normalize a vector using a Lp-norm.

Parameters

<i>q</i>	Norm to be utilized.
<i>p</i>	Vector to be normalized.

4.2.3.22 removeFeatures()

```
bool Data::removeFeatures (
    std::vector< int > feats )
```

Remove several features from the sample.

Parameters

<i>feats</i>	(???) Names of the features to be removed (must be sorted).
--------------	---

Returns

boolean informing if all features were succesfully removed.

4.2.3.23 removePoint()

```
bool Data::removePoint (
    int pid )
```

Remove a point from the data.

Parameters

<i>pid</i>	(???) Index of the point to be removed.
------------	---

Returns

bool

4.2.3.24 removePoints()

```
vector< bool > Data::removePoints (
    std::vector< int > ids )
```

Remove several points from the sample.

Parameters

<i>ids</i>	(???) Ids of the points to be removed (must be sorted).
------------	---

Returns

booleans informing which points were removed succesfully.

4.2.3.25 setClasses()

```
void Data::setClasses (
    std::string pos,
    std::string neg )
```

setClasses Set the classes of the dataset.

Parameters

<i>pos</i>	Positive class.
<i>neg</i>	Negative class.

4.2.3.26 setDim()

```
void Data::setDim (
    int dim )
```

setDim Set the dimension of the points.

Parameters

<i>dim</i>	(???) Dimension to be set.
------------	----------------------------

4.2.3.27 setFeaturesNames()

```
void Data::setFeaturesNames (
    std::vector< int > fnames )
```

setFeaturesNames Set the name of the features of the data.

Parameters

<i>fnames</i>	(???) Name of the features.
---------------	-----------------------------

4.2.3.28 setPoint()

```
void Data::setPoint (
    int index,
    Point p )
```

setPoint Set the point in a position of the data.

Parameters

<i>index</i>	(???) Index of the point that will be set.
<i>p</i>	(???) Point to be set.

4.2.3.29 write()

```
void Data::write (
    std::string fname,
    std::string ext )
```

write Write the data to a file with the given extension.

Parameters

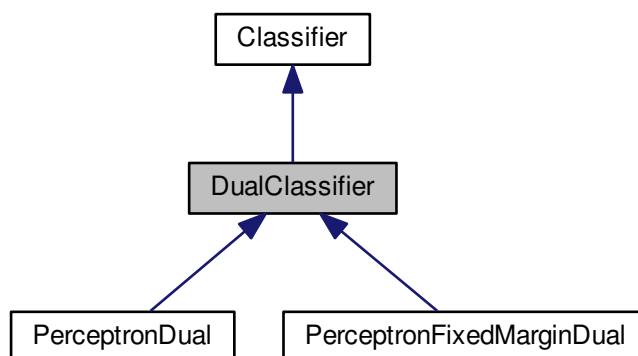
<i>fname</i>	Name of the file.
<i>ext</i>	Extention of the file.

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

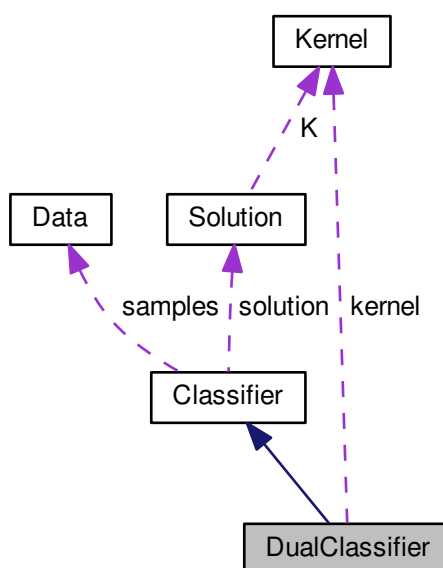
- includes/Data.hpp
- src/[Data.cpp](#)

4.3 DualClassifier Class Reference

Inheritance diagram for DualClassifier:



Collaboration diagram for DualClassifier:



Public Member Functions

- void [setKernel](#) ([Kernel](#) K)

setKernel Set the kernel used by the dual classifier.

- [Solution](#) `getSolution` ()

getSolution Returns the solution of the primal classifier.

Protected Attributes

- `std::vector< double >` [alpha](#)

Alphas vector.

- [Kernel](#) `kernel`

Object for kernel computations.

4.3.1 Member Function Documentation

4.3.1.1 `getSolution()`

[Solution](#) `DualClassifier::getSolution` ()

`getSolution` Returns the solution of the primal classifier.

Returns

[Solution](#)

4.3.1.2 `setKernel()`

```
void DualClassifier::setKernel (
    Kernel K )
```

`setKernel` Set the kernel used by the dual classifier.

Parameters

<code>q</code>	Norm that will be used by the classifier.
----------------	---

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

- `includes/DualClassifier.hpp`
- `src/DualClassifier.cpp`

4.4 FeatureSelection Class Reference

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

- `includes/FeatureSelection.hpp`

4.5 Gnuplot Class Reference

Public Member Functions

- [Gnuplot](#) (const std::string &style="points")
set a style during construction
- [Gnuplot](#) (const std::vector< double > &x, const std::string &title="", const std::string &style="points", const std::string &labelx="x", const std::string &labely="y")
plot a single std::vector at one go
- [Gnuplot](#) (const std::vector< double > &x, const std::vector< double > &y, const std::string &title="", const std::string &style="points", const std::string &labelx="x", const std::string &labely="y")
plot pairs std::vector at one go
- [Gnuplot](#) (const std::vector< double > &x, const std::vector< double > &y, const std::vector< double > &z, const std::string &title="", const std::string &style="points", const std::string &labelx="x", const std::string &labely="y", const std::string &labelz="z")
plot triples std::vector at one go
- [~Gnuplot](#) ()
destructor: needed to delete temporary files
- [Gnuplot & cmd](#) (const std::string &cmdstr)
send a command to gnuplot
- [Gnuplot & operator<<](#) (const std::string &cmdstr)
Sends a command to an active gnuplot session, identical to [cmd\(\)](#) send a command to gnuplot using the << operator.
- [Gnuplot & showonscreen](#) ()
sets terminal type to terminal_std
- [Gnuplot & savetops](#) (const std::string &filename="gnuplot_output")
saves a gnuplot session to a postscript file, filename without extension
- [Gnuplot & set_style](#) (const std::string &stylestr="points")
- [Gnuplot & set_smooth](#) (const std::string &stylestr="csplines")
- [Gnuplot & unset_smooth](#) ()
unset smooth attention: smooth is not set by default
- [Gnuplot & set_pointsize](#) (const double pointsize=1.0)
scales the size of the points used in plots
- [Gnuplot & set_grid](#) ()
turns grid on/off
- [Gnuplot & unset_grid](#) ()
grid is not set by default
- [Gnuplot & set_multiplot](#) ()
- [Gnuplot & unset_multiplot](#) ()
- [Gnuplot & set_samples](#) (const int samples=100)
set sampling rate of functions, or for interpolating data
- [Gnuplot & set_isosamples](#) (const int isolines=10)
set isoline density (grid) for plotting functions as surfaces (for 3d plots)
- [Gnuplot & set_hidden3d](#) ()
- [Gnuplot & unset_hidden3d](#) ()
- [Gnuplot & set_contour](#) (const std::string &position="base")
- [Gnuplot & unset_contour](#) ()
- [Gnuplot & set_surface](#) ()
- [Gnuplot & unset_surface](#) ()

- [Gnuplot & set_legend](#) (const std::string &position="default")
- [Gnuplot & unset_legend](#) ()
Switches legend off attention:legend is set by default.
- [Gnuplot & set_title](#) (const std::string &title="")
sets and clears the title of a gnuplot session
- [Gnuplot & unset_title](#) ()
Clears the title of a gnuplot session The title is not set by default.
- [Gnuplot & set_ylabel](#) (const std::string &label="x")
set x axis label
- [Gnuplot & set_xlabel](#) (const std::string &label="y")
set y axis label
- [Gnuplot & set_zlabel](#) (const std::string &label="z")
set z axis label
- [Gnuplot & set_xrange](#) (const double iFrom, const double iTo)
set axis - ranges
- [Gnuplot & set_yrange](#) (const double iFrom, const double iTo)
set y-axis - ranges
- [Gnuplot & set_zrange](#) (const double iFrom, const double iTo)
set z-axis - ranges
- [Gnuplot & set_xautoscale](#) ()
- [Gnuplot & set_yautoscale](#) ()
- [Gnuplot & set_zautoscale](#) ()
- [Gnuplot & set_xlogscale](#) (const double base=10)
turns on/off log scaling for the specified xaxis (logscale is not set by default)
- [Gnuplot & set_ylogscale](#) (const double base=10)
turns on/off log scaling for the specified yaxis (logscale is not set by default)
- [Gnuplot & set_zlogscale](#) (const double base=10)
turns on/off log scaling for the specified zaxis (logscale is not set by default)
- [Gnuplot & unset_xlogscale](#) ()
- [Gnuplot & unset_ylogscale](#) ()
- [Gnuplot & unset_zlogscale](#) ()
- [Gnuplot & set_cbrange](#) (const double iFrom, const double iTo)
set palette range (autoscale by default)
- [Gnuplot & plotfile_x](#) (const std::string &filename, const unsigned int column=1, const std::string &title="")
- `template<typename X >`
[Gnuplot & plot_x](#) (const X &x, const std::string &title="")
from std::vector
- [Gnuplot & plotfile_xy](#) (const std::string &filename, const unsigned int column_x=1, const unsigned int column_y=2, const std::string &title="")
- `template<typename X , typename Y >`
[Gnuplot & plot_xy](#) (const X &x, const Y &y, const std::string &title="")
from data
- [Gnuplot & plotfile_xy_err](#) (const std::string &filename, const unsigned int column_x=1, const unsigned int column_y=2, const unsigned int column_dy=3, const std::string &title="")
- `template<typename X , typename Y , typename E >`
[Gnuplot & plot_xy_err](#) (const X &x, const Y &y, const E &dy, const std::string &title="")
from data
- [Gnuplot & plotfile_xyz](#) (const std::string &filename, const unsigned int column_x=1, const unsigned int column_y=2, const unsigned int column_z=3, const std::string &title="")
- `template<typename X , typename Y , typename Z >`
[Gnuplot & plot_xyz](#) (const X &x, const Y &y, const Z &z, const std::string &title="")
from std::vector

- [Gnuplot & plot_slope](#) (const double a, const double b, const std::string &title="")
plot an equation of the form: $y = ax + b$, you supply a and b
- [Gnuplot & plot_equation](#) (const std::string &equation, const std::string &title="")
- [Gnuplot & plot_equation3d](#) (const std::string &equation, const std::string &title="")
- [Gnuplot & plot_image](#) (const unsigned char *ucPicBuf, const unsigned int iWidth, const unsigned int iHeight, const std::string &title="")
plot image
- [Gnuplot & replot](#) (void)
replot repeats the last plot or splot command. this can be useful for viewing a plot with different set options, or when generating the same plot for several devices (showonscreen, savetops)
- [Gnuplot & reset_plot](#) ()
resets a gnuplot session (next plot will erase previous ones)
- [Gnuplot & reset_all](#) ()
resets a gnuplot session and sets all variables to default
- void [remove_tmpfiles](#) ()
deletes temporary files
- bool [is_valid](#) ()
Is the gnuplot session valid ??

Static Public Member Functions

- static bool [set_GNUPlotPath](#) (const std::string &path)
optional function: set [Gnuplot](#) path manual attention: for windows: path with slash '/' not backslash '\'
- static void [set_terminal_std](#) (const std::string &type)

4.5.1 Member Function Documentation

4.5.1.1 is_valid()

```
bool Gnuplot::is_valid ( ) [inline]
```

Is the gnuplot session valid ??

Parameters

—	
---	--

Returns

true if valid, false if not

4.5.1.2 operator<<()

```
Gnuplot& Gnuplot::operator<< (
    const std::string & cmdstr ) [inline]
```

Sends a command to an active gnuplot session, identical to `cmd()` send a command to gnuplot using the `<<` operator.

Parameters

<i>cmdstr</i>	→ the command string
---------------	----------------------

Returns

← a reference to the gnuplot object

4.5.1.3 plot_equation()

```
Gnuplot & Gnuplot::plot_equation (
    const std::string & equation,
    const std::string & title = "" )
```

plot an equation supplied as a `std::string` `y=f(x)`, write only the function `f(x)` not `y=` the independent variable has to be `x` binary operators: `**` exponentiation, `*` multiply, `/` divide, `+` add, `-` subtract, `%` modulo unary operators: `-` minus, `!` factorial elementary functions: `rand(x)`, `abs(x)`, `sgn(x)`, `ceil(x)`, `floor(x)`, `int(x)`, `imag(x)`, `real(x)`, `arg(x)`, `sqrt(x)`, `exp(x)`, `log(x)`, `log10(x)`, `sin(x)`, `cos(x)`, `tan(x)`, `asin(x)`, `acos(x)`, `atan(x)`, `atan2(y,x)`, `sinh(x)`, `cosh(x)`, `tanh(x)`, `asinh(x)`, `acosh(x)`, `atanh(x)` special functions: `erf(x)`, `erfc(x)`, `inverf(x)`, `gamma(x)`, `igamma(a,x)`, `lgamma(x)`, `ibeta(p,q,x)`, `besj0(x)`, `besj1(x)`, `besy0(x)`, `besy1(x)`, `lambertw(x)` statistical fuctions: `norm(x)`, `invnorm(x)`

4.5.1.4 plot_equation3d()

```
Gnuplot & Gnuplot::plot_equation3d (
    const std::string & equation,
    const std::string & title = "" )
```

plot an equation supplied as a `std::string` `z=f(x,y)`, write only the function `f(x,y)` not `z=` the independent variables have to be `x` and `y`

4.5.1.5 plot_image()

```
Gnuplot & Gnuplot::plot_image (
    const unsigned char * ucPicBuf,
    const unsigned int iWidth,
    const unsigned int iHeight,
    const std::string & title = "" )
```

plot image

- note that this function is not valid for versions of GNUPlot below 4.2

4.5.1.6 plot_x()

```
template<typename X >
Gnuplot & Gnuplot::plot_x (
    const X & x,
    const std::string & title = "" )
```

from std::vector

Plots a 2d graph from a list of doubles: x.

4.5.1.7 plot_xy()

```
template<typename X , typename Y >
Gnuplot & Gnuplot::plot_xy (
    const X & x,
    const Y & y,
    const std::string & title = "" )
```

from data

Plots a 2d graph from a list of doubles: x y.

4.5.1.8 plot_xy_err()

```
template<typename X , typename Y , typename E >
Gnuplot & Gnuplot::plot_xy_err (
    const X & x,
    const Y & y,
    const E & dy,
    const std::string & title = "" )
```

from data

plot x,y pairs with dy errorbars

4.5.1.9 plotfile_x()

```
Gnuplot & Gnuplot::plotfile_x (
    const std::string & filename,
    const unsigned int column = 1,
    const std::string & title = "" )
```

plot a single std::vector: x from file

4.5.1.10 plotfile_xy()

```
Gnuplot & Gnuplot::plotfile_xy (
    const std::string & filename,
    const unsigned int column_x = 1,
    const unsigned int column_y = 2,
    const std::string & title = "" )
```

plot x,y pairs: x y from file

4.5.1.11 `plotfile_xy_err()`

```
Gnuplot & Gnuplot::plotfile_xy_err (
    const std::string & filename,
    const unsigned int column_x = 1,
    const unsigned int column_y = 2,
    const unsigned int column_dy = 3,
    const std::string & title = "" )
```

plot x,y pairs with dy errorbars: x y dy from file

4.5.1.12 `plotfile_xyz()`

```
Gnuplot & Gnuplot::plotfile_xyz (
    const std::string & filename,
    const unsigned int column_x = 1,
    const unsigned int column_y = 2,
    const unsigned int column_z = 3,
    const std::string & title = "" )
```

plot x,y,z triples: x y z from file

4.5.1.13 `replot()`

```
Gnuplot& Gnuplot::replot (
    void ) [inline]
```

replot repeats the last plot or splot command. this can be useful for viewing a plot with different set options, or when generating the same plot for several devices (showonscreen, savetops)

Parameters

—	
---	--

Returns

—

4.5.1.14 `set_contour()`

```
Gnuplot & Gnuplot::set_contour (
    const std::string & position = "base" )
```

enables/disables contour drawing for surfaces (for 3d plot) base, surface, both

4.5.1.15 `set_GNUPlotPath()`

```
bool Gnuplot::set_GNUPlotPath (
    const std::string & path ) [static]
```

optional function: set [Gnuplot](#) path manual attention: for windows: path with slash '/' not backslash '\'

Parameters

<i>path</i>	→ the gnuplot path
-------------	--------------------

Returns

true on success, false otherwise

4.5.1.16 set_hidden3d()

```
Gnuplot& Gnuplot::set_hidden3d ( ) [inline]
```

enables/disables hidden line removal for surface plotting (for 3d plot)

Parameters

—	
---	--

Returns

← reference to the gnuplot object

4.5.1.17 set_legend()

```
Gnuplot & Gnuplot::set_legend (
    const std::string & position = "default" )
```

switches legend on/off position: inside/outside, left/center/right, top/center/bottom, nobox/box

4.5.1.18 set_multiplot()

```
Gnuplot& Gnuplot::set_multiplot ( ) [inline]
```

set the mulitplot mode

Parameters

—	
---	--

Returns

← reference to the gnuplot object

4.5.1.19 `set_smooth()`

```
Gnuplot & Gnuplot::set_smooth (
    const std::string & stylestr = "csplines" )
```

interpolation and approximation of data, arguments: csplines, bezier, acsplines (for data values > 0), sbezier, unique, frequency (works only with plot_x, plot_xy, plotfile_x, plotfile_xy (if smooth is set, set_style has no effect on data plotting)

4.5.1.20 `set_style()`

```
Gnuplot & Gnuplot::set_style (
    const std::string & stylestr = "points" )
```

set line style (some of these styles require additional information): lines, points, linespoints, impulses, dots, steps, fsteps, histeps, boxes, histograms, filledcurves

4.5.1.21 `set_surface()`

```
Gnuplot & Gnuplot::set_surface ( ) [inline]
```

enables/disables the display of surfaces (for 3d plot)

Parameters

—	
---	--

Returns

← reference to the gnuplot object

4.5.1.22 `set_terminal_std()`

```
void Gnuplot::set_terminal_std (
    const std::string & type ) [static]
```

optional: set standart terminal, used by showonscreen defaults: Windows - win, Linux - x11, Mac - aqua

Parameters

<i>type</i>	→ the terminal type
-------------	---------------------

Returns

—

4.5.1.23 `set_title()`

```
Gnuplot& Gnuplot::set_title (
    const std::string & title = "" ) [inline]
```

sets and clears the title of a gnuplot session

Parameters

<i>title</i>	→ the title of the plot [optional, default == ""]
--------------	---

Returns

← reference to the gnuplot object

4.5.1.24 `set_xautoscale()`

```
Gnuplot& Gnuplot::set_xautoscale ( ) [inline]
```

autoscale axis (set by default) of xaxis

Parameters

—	
---	--

Returns

← reference to the gnuplot object

4.5.1.25 `set_yautoscale()`

```
Gnuplot& Gnuplot::set_yautoscale ( ) [inline]
```

autoscale axis (set by default) of yaxis

Parameters

—	
---	--

Returns

← reference to the gnuplot object

4.5.1.26 set_zautoscale()

```
Gnuplot& Gnuplot::set_zautoscale ( ) [inline]
```

autoscale axis (set by default) of zaxis

Parameters

—	
---	--

Returns

<— reference to the gnuplot object

4.5.1.27 unset_contour()

```
Gnuplot& Gnuplot::unset_contour ( ) [inline]
```

contour is not set by default, it disables contour drawing for surfaces

Parameters

—	
---	--

Returns

<— reference to the gnuplot object

4.5.1.28 unset_hidden3d()

```
Gnuplot& Gnuplot::unset_hidden3d ( ) [inline]
```

hidden3d is not set by default

Parameters

—	
---	--

Returns

<— reference to the gnuplot object

4.5.1.29 unset_legend()

`Gnuplot& Gnuplot::unset_legend () [inline]`

Switches legend off attention: legend is set by default.

Parameters

—	
---	--

Returns

<— reference to the gnuplot object

4.5.1.30 unset_multiplot()

`Gnuplot& Gnuplot::unset_multiplot () [inline]`

unsets the mulitplot mode

Parameters

—	
---	--

Returns

<— reference to the gnuplot object

4.5.1.31 unset_smooth()

`Gnuplot& Gnuplot::unset_smooth () [inline]`

unset smooth attention: smooth is not set by default

Parameters

—	
---	--

Returns

<— a reference to a gnuplot object

4.5.1.32 unset_surface()

```
Gnuplot& Gnuplot::unset_surface ( ) [inline]
```

surface is set by default, it disables the display of surfaces (for 3d plot)

Parameters

—	
---	--

Returns

<— reference to the gnuplot object

4.5.1.33 unset_title()

```
Gnuplot& Gnuplot::unset_title ( ) [inline]
```

Clears the title of a gnuplot session The title is not set by default.

Parameters

—	
---	--

Returns

<— reference to the gnuplot object

4.5.1.34 unset_xlogscale()

```
Gnuplot& Gnuplot::unset_xlogscale ( ) [inline]
```

turns off log scaling for the x axis

Parameters

—	
---	--

Returns

<— reference to the gnuplot object

4.5.1.35 unset_ylogscale()

```
Gnuplot& Gnuplot::unset_ylogscale ( ) [inline]
```

turns off log scaling for the y axis

Parameters

—	
---	--

Returns

<— reference to the gnuplot object

4.5.1.36 unset_zlogscale()

```
Gnuplot& Gnuplot::unset_zlogscale ( ) [inline]
```

turns off log scaling for the z axis

Parameters

—	
---	--

Returns

<— reference to the gnuplot object

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

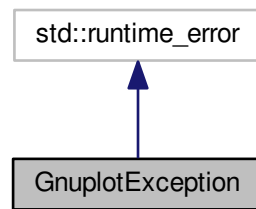
- includes/gnuplot_i.hpp
- src/gnuplot_i.cpp

4.6 GnuplotException Class Reference

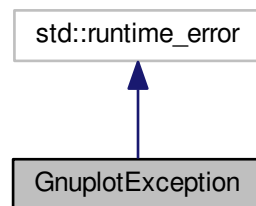
A C++ interface to gnuplot.

```
#include <gnuplot_i.hpp>
```

Inheritance diagram for GnuplotException:



Collaboration diagram for GnuplotException:



Public Member Functions

- **GnuplotException** (const std::string &msg)

4.6.1 Detailed Description

A C++ interface to gnuplot.

The interface uses pipes and so won't run on a system that doesn't have POSIX pipe support Tested on Windows (MinGW and Visual C++) and Linux (GCC)

Version history: 0. C interface by N. Devillard (27/01/03)

1. C++ interface: direct translation from the C interface by Rajarshi Guha (07/03/03)
2. corrections for Win32 compatibility by V. Chyzhdzenka (20/05/03)
3. some member functions added, corrections for Win32 and Linux compatibility by M. Burgis (10/03/08)
4. Move function definition into gnuplot_i.cpp by X. BROQUERE (25/10/11)

Requirements:

- gnuplot has to be installed (<http://www.gnuplot.info/download.html>)
- for Windows: set Path-Variable for [Gnuplot](#) path (e.g. C:/program files/gnuplot/bin) or set [Gnuplot](#) path with:
`Gnuplot::set_GNUPlotPath(const std::string &path);`

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

- includes/gnuplot_i.hpp

4.7 Kernel Class Reference

Class for the kernel computations.

```
#include <Kernel.hpp>
```

Public Member Functions

- [Kernel](#) ()
Class constructor.
- [Kernel](#) (dMatrix K)
Class constructor.
- void [setType](#) (int type)
setType Set the kernel type used in the kernel computations.
- void [setParam](#) (int param)
setParam Set the kernel parameter used in the kernel computations.
- void [setKernelMatrix](#) (dMatrix K)
setKernelMatrix Set a pre computed kernel matrix.
- dMatrix [getKernelMatrix](#) ()
getKernelMatrix Get the kernel matrix.
- void [compute](#) ([Data](#) samples)
compute Compute the kernel matrix with the given type and parameter.
- double [function](#) ([Point](#) one, [Point](#) two, int dim)
function Compute the kernel function between two points.
- double [norm](#) ([Data](#) data)
norm Computes norm in dual variables.

4.7.1 Detailed Description

Class for the kernel computations.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Kernel()

```
Kernel::Kernel (
    dMatrix K )
```

Class constructor.

Parameters

<i>K</i>	Kernel matrix to be set in initialization.
----------	--

4.7.3 Member Function Documentation

4.7.3.1 compute()

```
void Kernel::compute (
    Data samples )
```

compute Compute the kernel matrix with the given type and parameter.

Parameters

<i>samples</i>	Data used to compute the kernel matrix.
----------------	---

4.7.3.2 function()

```
double Kernel::function (
    Point one,
    Point two,
    int dim )
```

function Compute the kernel function between two points.

Parameters

<i>one</i>	first point.
<i>two</i>	second point.
<i>dim</i>	Dimension of the points.

Returns

double

4.7.3.3 getKernelMatrix()

```
dMatrix Kernel::getKernelMatrix ( )
```

getKernelMatrix Get the kernel matrix.

Returns

`std::vector<std::vector<double>>`

4.7.3.4 norm()

```
double Kernel::norm (
    Data data )
```

norm Computes norm in dual variables.

Parameters

<i>data</i>	Dataset to compute norm.
-------------	--------------------------

Returns

double

4.7.3.5 setKernelMatrix()

```
void Kernel::setKernelMatrix (
    dMatrix K )
```

setKernelMatrix Set a pre computed kernel matrix.

Parameters

<i>K</i>	Kernel matrix to be set.
----------	--------------------------

4.7.3.6 setParam()

```
void Kernel::setParam (
    int param )
```

setParam Set the kernel parameter used in the kernel computations.

Parameters

<i>param</i>	parameter to be set.
--------------	----------------------

4.7.3.7 setType()

```
void Kernel::setType (
    int type )
```

setType Set the kernel type used in the kernel computations.

Parameters

<i>type</i>	Kernel type.
-------------	--------------

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

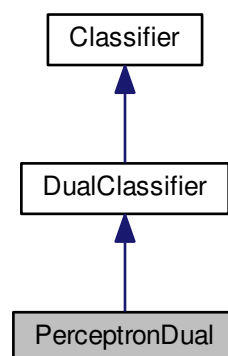
- includes/Kernel.hpp
- src/Kernel.cpp

4.8 PerceptronDual Class Reference

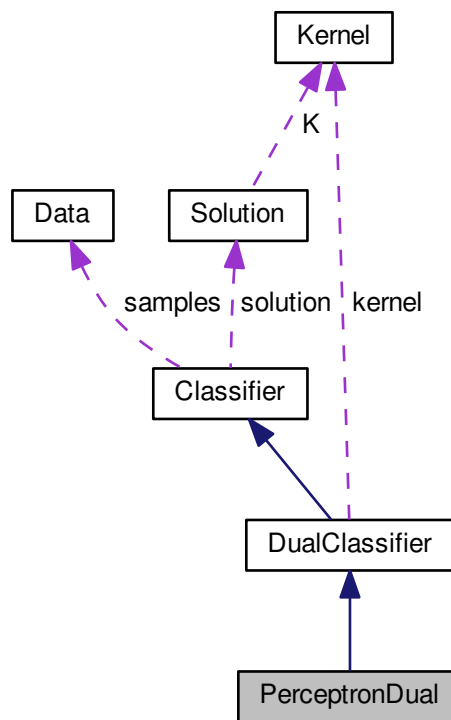
Wrapper for the implementation of the Perceptron dual algorithm.

```
#include <Perceptron.hpp>
```

Inheritance diagram for PerceptronDual:



Collaboration diagram for PerceptronDual:



Public Member Functions

- **PerceptronDual** ([Data](#) *samples=NULL, double [rate](#)=0.5, [Kernel](#) *K=NULL, [Solution](#) *initial_solution=NULL)
- bool [train](#) ()
Function that execute the training phase of a classification algorithm.
- double [evaluate](#) ([Point](#) p)
Returns the class of a feature point based on the trained classifier.

Additional Inherited Members

4.8.1 Detailed Description

Wrapper for the implementation of the Perceptron dual algorithm.

4.8.2 Member Function Documentation

4.8.2.1 evaluate()

```
double PerceptronDual::evaluate (
    Point p ) [virtual]
```

Returns the class of a feature point based on the trained classifier.

Parameters

Point	x (???) Features point to be evaluated.
-----------------------	---

Returns

int

Implements [Classifier](#).

4.8.2.2 train()

```
bool PerceptronDual::train ( ) [virtual]
```

Function that execute the training phase of a classification algorithm.

Returns

void

Implements [Classifier](#).

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

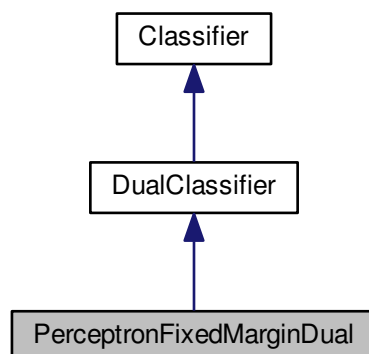
- includes/Perceptron.hpp
- src/Perceptron.cpp

4.9 PerceptronFixedMarginDual Class Reference

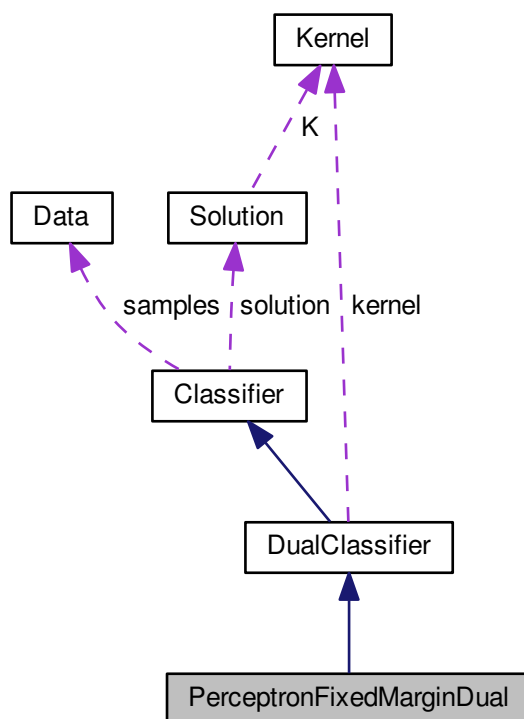
Wrapper for the implementation of the Perceptron dual with fixed margin algorithm.

```
#include <Perceptron.hpp>
```

Inheritance diagram for PerceptronFixedMarginDual:



Collaboration diagram for PerceptronFixedMarginDual:



Public Member Functions

- **PerceptronFixedMarginDual** ([Data](#) *samples=NULL, double gamma=1.0, double [rate](#)=0.5, [Kernel](#) *K=NULL, [Solution](#) *initial_solution=NULL)
- bool [train](#) ()
Function that execute the training phase of a classification algorithm.
- double [evaluate](#) ([Point](#) p)
Returns the class of a feature point based on the trained classifier.

Additional Inherited Members

4.9.1 Detailed Description

Wrapper for the implementation of the Perceptron dual with fixed margin algorithm.

4.9.2 Member Function Documentation

4.9.2.1 [evaluate\(\)](#)

```
double PerceptronFixedMarginDual::evaluate (
    Point p ) [virtual]
```

Returns the class of a feature point based on the trained classifier.

Parameters

Point	x (???) Features point to be evaluated.
-----------------------	---

Returns

int

Implements [Classifier](#).

4.9.2.2 [train\(\)](#)

```
bool PerceptronFixedMarginDual::train ( ) [virtual]
```

Function that execute the training phase of a classification algorithm.

Returns

void

Implements [Classifier](#).

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

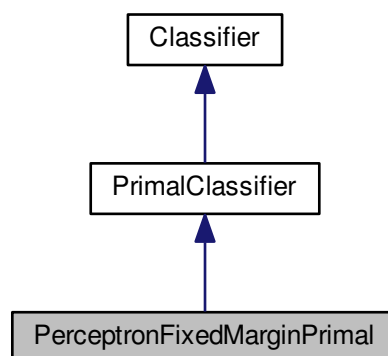
- includes/Perceptron.hpp
- src/Perceptron.cpp

4.10 PerceptronFixedMarginPrimal Class Reference

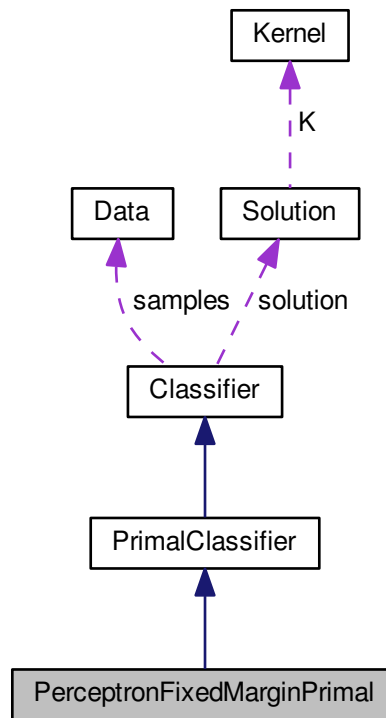
Wrapper for the implementation of the Perceptron primal with fixed margin algorithm.

```
#include <Perceptron.hpp>
```

Inheritance diagram for PerceptronFixedMarginPrimal:



Collaboration diagram for PerceptronFixedMarginPrimal:



Public Member Functions

- **PerceptronFixedMarginPrimal** ([Data](#) *samples=NULL, double gamma=1.0, double [q](#)=2, double [rate](#)=0.5, [Solution](#) *initial_solution=NULL)
- bool [train](#) ()
Function that execute the training phase of a classification algorithm.
- double [evaluate](#) ([Point](#) p)
Returns the class of a feature point based on the trained classifier.

Additional Inherited Members

4.10.1 Detailed Description

Wrapper for the implementation of the Perceptron primal with fixed margin algorithm.

4.10.2 Member Function Documentation

4.10.2.1 evaluate()

```
double PerceptronFixedMarginPrimal::evaluate (
    Point p ) [virtual]
```

Returns the class of a feature point based on the trained classifier.

Parameters

Point	x (???) Features point to be evaluated.
-----------------------	---

Returns

int

Implements [Classifier](#).

4.10.2.2 train()

```
bool PerceptronFixedMarginPrimal::train ( ) [virtual]
```

Function that execute the training phase of a classification algorithm.

Returns

void

Implements [Classifier](#).

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

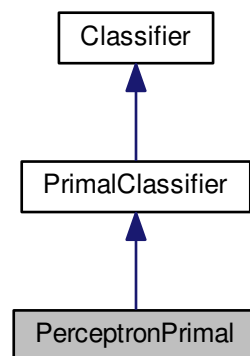
- includes/Perceptron.hpp
- src/Perceptron.cpp

4.11 PerceptronPrimal Class Reference

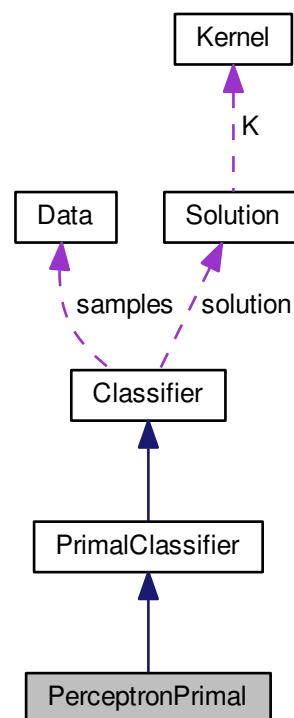
Wrapper for the implementation of the Perceptron primal algorithm.

```
#include <Perceptron.hpp>
```


Inheritance diagram for PerceptronPrimal:



Collaboration diagram for PerceptronPrimal:



Public Member Functions

- **PerceptronPrimal** ([Data](#) *samples=NULL, double [q](#)=2, double [rate](#)=0.5, [Solution](#) *initial_solution=NULL)

- bool [train](#) ()
Function that execute the training phase of a classification algorithm.
- double [evaluate](#) ([Point](#) p)
Returns the class of a feature point based on the trained classifier.

Additional Inherited Members

4.11.1 Detailed Description

Wrapper for the implementation of the Perceptron primal algorithm.

4.11.2 Member Function Documentation

4.11.2.1 [evaluate\(\)](#)

```
double PerceptronPrimal::evaluate (  
    Point p ) [virtual]
```

Returns the class of a feature point based on the trained classifier.

Parameters

Point	x (???) Features point to be evaluated.
-----------------------	---

Returns

int

Implements [Classifier](#).

4.11.2.2 [train\(\)](#)

```
bool PerceptronPrimal::train ( ) [virtual]
```

Function that execute the training phase of a classification algorithm.

Returns

void

Implements [Classifier](#).

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

- includes/Perceptron.hpp
- src/Perceptron.cpp

4.12 Point Class Reference

Class of a [Point](#) of doubles in a space of n dimensions.

```
#include <Point.hpp>
```

Public Member Functions

- **Point** (int dim, int val=0)
- double [dot](#) (std::vector< double > p)
Computes the dot product with a vector.
- double [norm](#) (int p=2)
Returns the p-norm of the point.

Public Attributes

- std::vector< double > [x](#)
Features values.
- double [y](#) = 0
[Point](#) classification.
- double **alpha** = 0.0
- int [id](#) = 0
[Point](#) identification.

Friends

- std::ostream & **operator**<< (std::ostream &output, const [Point](#) &p)

4.12.1 Detailed Description

Class of a [Point](#) of doubles in a space of n dimensions.

4.12.2 Member Function Documentation

4.12.2.1 dot()

```
double Point::dot (  
    std::vector< double > p )
```

Computes the dot product with a vector.

Parameters

p	(???)
-----	-------

Returns

double

4.12.2.2 norm()

```
double Point::norm (
    int p = 2 )
```

Returns the p-norm of the point.

Parameters

p	(???) p of the norm (euclidean norm is the default).
-----	--

Returns

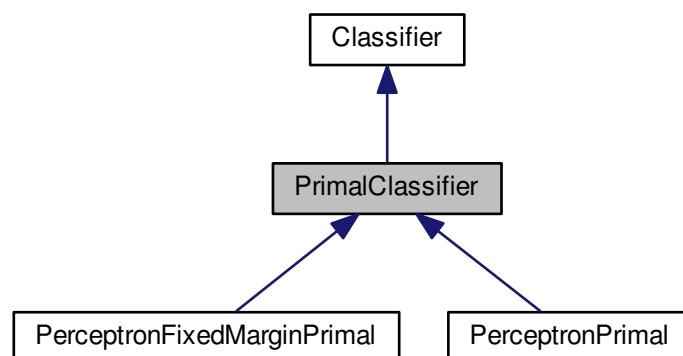
double

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

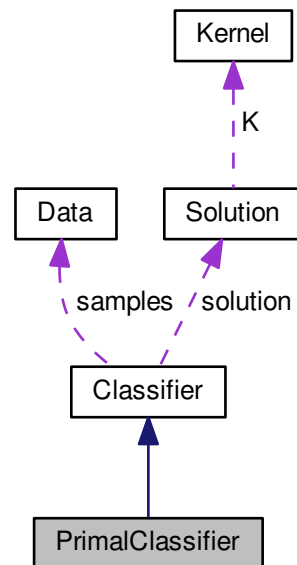
- includes/Point.hpp
- src/[Point.cpp](#)

4.13 PrimalClassifier Class Reference

Inheritance diagram for PrimalClassifier:



Collaboration diagram for PrimalClassifier:



Public Member Functions

- void [setNorm](#) (double [q](#))
setNorm Set the norm used by the classifier. (Euclidean norm is the default)
- [Solution](#) [getSolution](#) ()
getSolution Returns the solution of the primal classifier.

Protected Attributes

- `std::vector< double > w`
Weights vector.
- double [q](#) = 2
Norm used in the classification. (Euclidean Norm is the default)
- double [flexible](#) = 0.0
Flexibilidade.

4.13.1 Member Function Documentation

4.13.1.1 `getSolution()`

`Solution` PrimalClassifier::getSolution ()

`getSolution` Returns the solution of the primal classifier.

Returns

`Solution`

4.13.1.2 `setNorm()`

```
void PrimalClassifier::setNorm (
    double q )
```

`setNorm` Set the norm used by the classifier. (Euclidean norm is the default)

Parameters

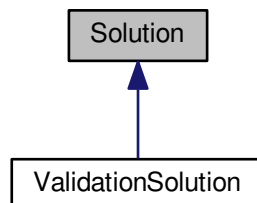
<i>q</i>	Norm that will be used by the classifier.
----------	---

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

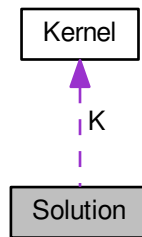
- includes/PrimalClassifier.hpp
- src/PrimalClassifier.cpp

4.14 Solution Class Reference

Inheritance diagram for `Solution`:



Collaboration diagram for Solution:



Public Attributes

- `std::vector< double > w`
Weights vector.
- `Kernel K`
Kernel for Dual methods.
- `std::vector< double > alpha`
Alpha Vector for Dual methods.
- `double bias = 0`
Bias of the solution.
- `std::vector< int > fnames`
Features names of the resulting solution.
- `double margin = 0`
Margin generated from the classifier that generated the solution.
- `double norm = 0`
Norm of the solution.

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

- `includes/Solution.hpp`

4.15 Statistics Class Reference

Class with methods for statistical computations.

```
#include <Statistics.hpp>
```

Static Public Member Functions

- static double `mean` (`std::vector< double > p`)
Compute the mean (average) of a vector.
- static double `getFeatureMean` (`Data data`, `int index`)
Computes the mean of a feature in the sample.
- static double `variance` (`std::vector< double > p`)
Compute the variance of a vector.
- static double `variance` (`Data data`, `int index`)
Compute the variance of a sample.
- static double `stdev` (`std::vector< double > p`)
Compute the standard deviation of a vector.
- static double `getFeatureStdev` (`Data data`, `int index`)
Computes the standard deviation of a feature.
- static double `getRadius` (`Data data`, `int index`, `double q`)
Returns radius of the ball that circ. the data.
- static double `getDistCenters` (`Data data`, `int index`)
Returns distance of centers of the classes.
- static double `getDistCentersWithoutFeats` (`Data data`, `std::vector< int > feats`, `int index`)
Returns distance of centers of the classes without given features.

Friends

- class `Data`

4.15.1 Detailed Description

Class with methods for statistical computations.

4.15.2 Member Function Documentation

4.15.2.1 `getDistCenters()`

```
double Statistics::getDistCenters (
    Data data,
    int index ) [static]
```

Returns distance of centers of the classes.

Parameters

<i>data</i>	Dataset to compute the distance.
<i>index</i>	Feature to be ignored (-1 uses all features).

Returns

double

4.15.2.2 getDistCentersWithoutFeats()

```
double Statistics::getDistCentersWithoutFeats (
    Data data,
    std::vector< int > feats,
    int index ) [static]
```

Returns distance of centers of the classes without given features.

Parameters

<i>data</i>	Dataset to compute the distance.
<i>feats</i>	Features to be excluded from the computation.
<i>index</i>	Feature to be ignored (-1 uses all features).

Returns

double

4.15.2.3 getFeatureMean()

```
double Statistics::getFeatureMean (
    Data data,
    int index ) [static]
```

Computes the mean of a feature in the sample.

Parameters

<i>data</i>	(???) Sample where the feature is located.
<i>index</i>	(???) Index of the feature to compute the mean.

Returns

double

4.15.2.4 getFeatureStdev()

```
double Statistics::getFeatureStdev (
    Data data,
    int index ) [static]
```

Computes the standard deviation of a feature.

Parameters

<i>data</i>	(???) Sample where the feature is located.
<i>index</i>	(???) Index of teh feature to compute the standard deviation.

Returns

double

4.15.2.5 getRadius()

```
double Statistics::getRadius (
    Data data,
    int index,
    double q ) [static]
```

Returns radius of the ball that circ. the data.

Parameters

<i>data</i>	Dataset to compute the radius.
<i>index</i>	Feature to be ignored (-1 uses all features).
<i>q</i>	Lp-Norm to be used.

Returns

double

4.15.2.6 mean()

```
double Statistics::mean (
    std::vector< double > p ) [static]
```

Compute the mean (average) of a vector.

Parameters

<i>p</i>	(???) Point to compute the mean.
----------	--

Returns

double

4.15.2.7 stdev()

```
double Statistics::stdev (
    std::vector< double > p ) [static]
```

Compute the standard deviation of a vector.

Parameters

<i>p</i>	(???) Point to compute stdev.
----------	---

Returns

double

4.15.2.8 variance() [1/2]

```
static double Statistics::variance (
    std::vector< double > p ) [static]
```

Compute the variance of a vector.

Parameters

<i>p</i>	(???) Vector to compute the variance.
----------	---------------------------------------

Returns

double

4.15.2.9 variance() [2/2]

```
double Statistics::variance (
    Data data,
    int index ) [static]
```

Compute the variance of a sample.

Parameters

<i>data</i>	(???) Sample to compute the variance.
<i>index</i>	(???) Index of the feature to be ignored. (-1 dont ignore any feature)

Returns

double

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

- includes/Statistics.hpp
- src/Statistics.cpp

4.16 Validation Class Reference

Class of methods for the validation of ML algorithms.

```
#include <Validation.hpp>
```

Public Member Functions

- **Validation** ([Data](#) *sample, [Classifier](#) *classifier=NULL)
- void [partTrainTest](#) (int fold, unsigned int seed)
Divide sample into train and test.
- double **kFold** (int fold, int seed)
- void **validation** (int fold, int qtde)
- [Data](#) **getTestSample** ()
- [Data](#) **getTrainSample** ()

4.16.1 Detailed Description

Class of methods for the validation of ML algorithms.

4.16.2 Member Function Documentation

4.16.2.1 [partTrainTest\(\)](#)

```
void Validation::partTrainTest (
    int fold,
    unsigned int seed )
```

Divide sample into train and test.

Parameters

<i>fold</i>	Number of folds.
<i>seed</i>	Seed to feed the pseudo random number generator.

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

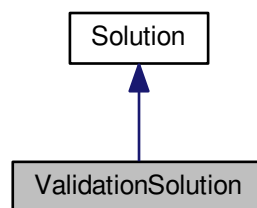
- includes/Validation.hpp
- src/Validation.cpp

4.17 ValidationSolution Class Reference

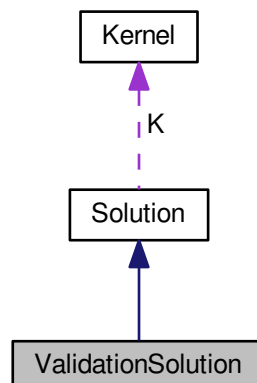
[Solution](#) for the validation of a ML method.

```
#include <ValidationSolution.hpp>
```

Inheritance diagram for ValidationSolution:



Collaboration diagram for ValidationSolution:



Additional Inherited Members

4.17.1 Detailed Description

[Solution](#) for the validation of a ML method.

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

- includes/ValidationSolution.hpp

4.18 Visualisation Class Reference

Class for visualize data using gnuplot.

```
#include <Visualisation.hpp>
```

Public Member Functions

- **Visualisation** ([Data](#) *sample)
- void [setSample](#) ([Data](#) sample)
Set sample to be visualized.
- void [setTitle](#) (std::string title)
Set plot title.
- void [setStyle](#) (std::string style)
Set plot style. (points, lines, etc.)
- void [plot2D](#) (int x, int y)
Plot the selected features in 2D.
- void [plot3D](#) (int x, int y, int z)
Plot the selected features in 3D.
- void [plot2DwithHyperplane](#) (int x, int y, [Solution](#) w)
Plot the data in 2D with separated by the hyperplane in the solution.
- void [plot3DwithHyperplane](#) (int x, int y, int z, [Solution](#) w)
Plot the data in 2D with separated by the hyperplane in the solution.

4.18.1 Detailed Description

Class for visualize data using gnuplot.

4.18.2 Member Function Documentation

4.18.2.1 [plot2D\(\)](#)

```
void Visualisation::plot2D (  
    int x,  
    int y )
```

Plot the selected features in 2D.

Parameters

<i>x</i>	(???) Feature to be used in the x-axis.
<i>y</i>	(???) Feature to be used in the y-axis.

Returns

void

4.18.2.2 plot2DwithHyperplane()

```
void Visualisation::plot2DwithHyperplane (
    int x,
    int y,
    Solution w )
```

Plot the data in 2D with separated by the hyperplane in the solution.

Parameters

<i>x</i>	(???) Feature to be used in the x-axis.
<i>y</i>	(???) Feature to be used in the y-axis.

Returns

void

4.18.2.3 plot3D()

```
void Visualisation::plot3D (
    int x,
    int y,
    int z )
```

Plot the selected features in 3D.

Parameters

<i>x</i>	(???) Feature to be used in the x-axis.
<i>y</i>	(???) Feature to be used in the y-axis.
<i>z</i>	(???) Feature to be used in the z-axis.

Returns

void

4.18.2.4 plot3DwithHyperplane()

```
void Visualisation::plot3DwithHyperplane (
    int x,
    int y,
    int z,
    Solution w )
```

Plot the data in 2D with separated by the hyperplane in the solution.

Parameters

<i>x</i>	(???) Feature to be used in the x-axis.
<i>y</i>	(???) Feature to be used in the y-axis.
<i>z</i>	(???) Feature to be used in the z-axis.

Returns

void

4.18.2.5 setSample()

```
void Visualisation::setSample (
    Data sample )
```

Set sample to be visualized.

Parameters

<i>sample</i>	(???) Data to set for visualization.
---------------	--

Returns

void

4.18.2.6 setStyle()

```
void Visualisation::setStyle (
    std::string style )
```

Set plot style. (points, lines, etc.)

Parameters

<i>style</i>	(???) Style to be set.
--------------	------------------------

Returns

void

4.18.2.7 setTitle()

```
void Visualisation::setTitle (
    std::string title )
```

Set plot title.

Parameters

<i>title</i>	(???) Plot title.
--------------	-------------------

Returns

void

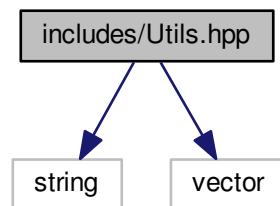
The documentation for this class was generated from the following files:

- includes/Visualisation.hpp
- src/Visualisation.cpp

File Documentation

```
#include <string>
#include <vector>
```

Include dependency graph for Utils.hpp:



```

graph TD
    includes_Utils_hpp[includes/Utils.hpp]
    src_Utils_cpp[src/Utils.cpp]
    includes_Point_hpp[includes/Point.hpp]
    src_Point_cpp[src/Point.cpp]
    includes_Statistics_hpp[includes/Statistics.hpp]
    includes_Data_hpp[includes/Data.hpp]
    src_Data_cpp[src/Data.cpp]
    includes_Visualisation_hpp[includes/Visualisation.hpp]
    includes_FeatureSelection_hpp[includes/FeatureSelection.hpp]
    includes_MLToolkit_hpp[includes/MLToolkit.hpp]
    includes_Kernel_hpp[includes/Kernel.hpp]
    includes_Solution_hpp[includes/Solution.hpp]
    includes_ValidationSolution_hpp[includes/ValidationSolution.hpp]
    includes_Classifier_hpp[includes/Classifier.hpp]
    includes_Validation_hpp[includes/Validation.hpp]
    includes_PrimalClassifier_hpp[includes/PrimalClassifier.hpp]
    includes_DualClassifier_hpp[includes/DualClassifier.hpp]
    includes_Perception_hpp[includes/Perception.hpp]

    includes_Utils_hpp --> includes_Point_hpp
    includes_Utils_hpp --> includes_Statistics_hpp
    includes_Utils_hpp --> includes_Data_hpp
    includes_Utils_hpp --> includes_Visualisation_hpp
    includes_Utils_hpp --> includes_FeatureSelection_hpp
    includes_Utils_hpp --> includes_MLToolkit_hpp
    includes_Utils_hpp --> includes_Kernel_hpp
    includes_Utils_hpp --> includes_Solution_hpp
    includes_Utils_hpp --> includes_ValidationSolution_hpp
    includes_Utils_hpp --> includes_Classifier_hpp
    includes_Utils_hpp --> includes_Validation_hpp
    includes_Utils_hpp --> includes_PrimalClassifier_hpp
    includes_Utils_hpp --> includes_DualClassifier_hpp
    includes_Utils_hpp --> includes_Perception_hpp

    src_Utils_cpp --> includes_Utils_hpp
    includes_Point_hpp --> src_Point_cpp
    src_Point_cpp --> includes_Point_hpp
    includes_Statistics_hpp --> includes_Data_hpp
    includes_Data_hpp --> src_Data_cpp
    src_Data_cpp --> includes_Data_hpp
    includes_Data_hpp --> includes_Visualisation_hpp
    includes_Data_hpp --> includes_FeatureSelection_hpp
    includes_Data_hpp --> includes_MLToolkit_hpp
    includes_Visualisation_hpp --> includes_MLToolkit_hpp
    includes_FeatureSelection_hpp --> includes_MLToolkit_hpp
    includes_MLToolkit_hpp --> includes_Kernel_hpp
    includes_MLToolkit_hpp --> includes_Solution_hpp
    includes_MLToolkit_hpp --> includes_ValidationSolution_hpp
    includes_MLToolkit_hpp --> includes_Classifier_hpp
    includes_MLToolkit_hpp --> includes_Validation_hpp
    includes_MLToolkit_hpp --> includes_PrimalClassifier_hpp
    includes_MLToolkit_hpp --> includes_DualClassifier_hpp
    includes_MLToolkit_hpp --> includes_Perception_hpp
    includes_Kernel_hpp --> includes_Solution_hpp
    includes_Kernel_hpp --> includes_ValidationSolution_hpp
    includes_Kernel_hpp --> includes_Classifier_hpp
    includes_Kernel_hpp --> includes_Validation_hpp
    includes_Kernel_hpp --> includes_PrimalClassifier_hpp
    includes_Kernel_hpp --> includes_DualClassifier_hpp
    includes_Kernel_hpp --> includes_Perception_hpp
    includes_Solution_hpp --> includes_ValidationSolution_hpp
    includes_ValidationSolution_hpp --> includes_Classifier_hpp
    includes_ValidationSolution_hpp --> includes_Validation_hpp
    includes_ValidationSolution_hpp --> includes_PrimalClassifier_hpp
    includes_ValidationSolution_hpp --> includes_DualClassifier_hpp
    includes_ValidationSolution_hpp --> includes_Perception_hpp
    includes_Classifier_hpp --> includes_Validation_hpp
    includes_Classifier_hpp --> includes_PrimalClassifier_hpp
    includes_Classifier_hpp --> includes_DualClassifier_hpp
    includes_Classifier_hpp --> includes_Perception_hpp
    includes_Validation_hpp --> includes_PrimalClassifier_hpp
    includes_Validation_hpp --> includes_DualClassifier_hpp
    includes_Validation_hpp --> includes_Perception_hpp
    includes_PrimalClassifier_hpp --> includes_DualClassifier_hpp
    includes_PrimalClassifier_hpp --> includes_Perception_hpp
    includes_DualClassifier_hpp --> includes_Perception_hpp
    includes_Perception_hpp --> includes_Perception_hpp
  
```

Macros

- `#define INF 1E8`

Typedefs

- `typedef std::vector< std::vector< double > > dMatrix`

Enumerations

- `enum NormType { NORM_LINF = 0, NORM_L1 = 1, NORM_L2 = 2 }`

Functions

- `bool is_number (std::string str)`
Verify if the string is a number.
- `int stoin (std::string str)`
Converts the string to an integer.
- `double stodn (std::string str)`
Converts the string to a double.
- `double maxAbsElement (std::vector< double > x)`
Returns the max absolute element.
- `std::string itos (int n)`
itos Integer to string conversion.
- `std::string dtoa (double n)`
dtoa Double to string conversion.

5.1.1 Detailed Description

Utils functions

Author

Mateus Coutinho Marim

5.1.2 Function Documentation

5.1.2.1 `dtoa()`

```
std::string dtoa (  
    double n )
```

`dtoa` Double to string conversion.

Parameters

<i>n</i>	Double to be converted.
----------	-------------------------

Returns

string

5.1.2.2 is_number()

```
bool is_number (
    std::string str )
```

Verify if the string is a number.

Parameters

<i>str</i>	String to be tested.
------------	----------------------

Returns

bool

5.1.2.3 itos()

```
std::string itos (
    int n )
```

itos Integer to string conversion.

Parameters

<i>n</i>	Integer to be converted.
----------	--------------------------

Returns

string

5.1.2.4 maxAbsElement()

```
double maxAbsElement (
    std::vector< double > x )
```

Returns the max absolute element.

Parameters

<i>x</i>	The vector used to obtain the max element.
----------	--

Returns

The max absolute element found.

5.1.2.5 stodn()

```
double stodn (
    std::string str )
```

Converts the string to a double.

Parameters

<i>str</i>	The string to be converted.
------------	-----------------------------

Returns

The double resulted from the conversion.

5.1.2.6 stoin()

```
int stoin (
    std::string str )
```

Converts the string to an integer.

Parameters

<i>str</i>	String to be converted.
------------	-------------------------

Returns

The integer resulted from the conversion.

5.2 src/Data.cpp File Reference

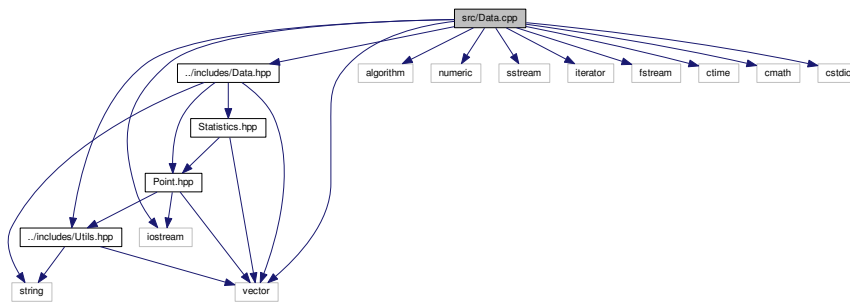
Implementation of the [Data](#) class methods.

```

#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>
#include <sstream>
#include <iterator>
#include <fstream>
#include <ctime>
#include <cmath>
#include <cstdio>
#include "../includes/Data.hpp"
#include "../includes/Utils.hpp"

```

Include dependency graph for Data.cpp:



Functions

- ostream & **operator**<< (ostream &output, const [Data](#) &data)

5.2.1 Detailed Description

Implementation of the [Data](#) class methods.

5.3 src/Point.cpp File Reference

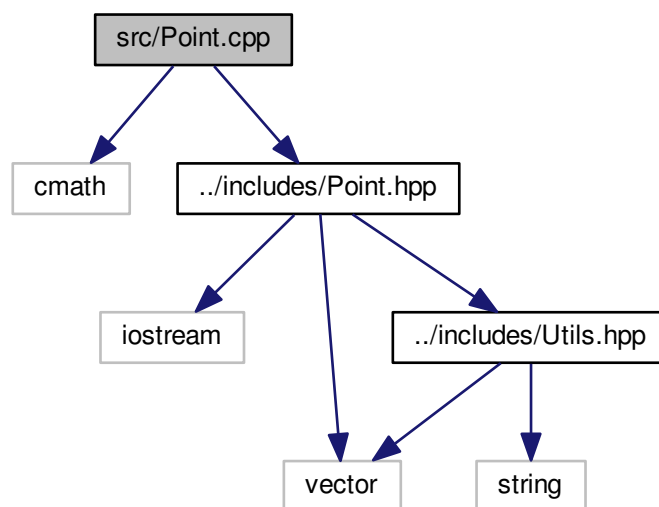
Implementation of the [Point](#) class methods.

```

#include <cmath>
#include "../includes/Point.hpp"

```


Include dependency graph for Point.cpp:



Functions

- ostream & **operator**<< (ostream &output, const [Point](#) &p)

5.3.1 Detailed Description

Implementation of the [Point](#) class methods.

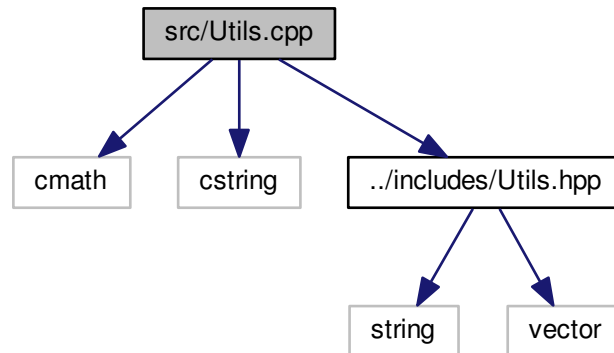
5.4 src/Utils.cpp File Reference

Implementation of methods for general use in the system.

```
#include <cmath>
#include <cstring>
```

```
#include "../includes/Utils.hpp"
```

Include dependency graph for Utils.cpp:



Macros

- `#define PRECISION 1E-9`
- `#define MAX_NUMBER_STRING_SIZE 32`

Functions

- `bool is_number (string str)`
- `int stoin (string str)`
- `double stodn (string str)`
- `string itos (int n)`
itos Integer to string conversion.
- `void reverse (char *str, int len)`
- `int intToStr (int x, char str[], int d)`
- `string dtoa (double n)`
dtoa Double to string conversion.
- `double maxAbsElement (vector< double > x)`

5.4.1 Detailed Description

Implementation of methods for general use in the system.

Utils functions

Author

Mateus Coutinho Marim

5.4.2 Function Documentation

5.4.2.1 dtoa()

```
string dtoa (
    double n )
```

dtoa Double to string conversion.

Parameters

<i>n</i>	Double to be converted.
----------	-------------------------

Returns

string

5.4.2.2 itos()

```
string itos (
    int n )
```

itos Integer to string conversion.

Parameters

<i>n</i>	Integer to be converted.
----------	--------------------------

Returns

string

Index

- changeXVector
 - Data, [13](#)
- Classifier, [7](#)
 - evaluate, [9](#)
 - getSteps, [9](#)
 - getUpdates, [9](#)
 - setSamples, [10](#)
 - train, [10](#)
- compute
 - Kernel, [39](#)
- copy
 - Data, [13](#)
- copyZero
 - Data, [13](#)
- Data, [10](#)
 - changeXVector, [13](#)
 - copy, [13](#)
 - copyZero, [13](#)
 - Data, [12](#)
 - getDim, [14](#)
 - getFeaturesNames, [14](#)
 - getIndex, [14](#)
 - getNumberNegativePoints, [14](#)
 - getNumberPositivePoints, [15](#)
 - getPoint, [15](#)
 - getPoints, [15](#)
 - getSize, [15](#)
 - getStatistics, [16](#)
 - insertFeatures, [16](#)
 - insertPoint, [16](#), [17](#)
 - isEmpty, [17](#)
 - isNormalized, [17](#)
 - join, [17](#)
 - load, [18](#)
 - normalize, [18](#)
 - removeFeatures, [19](#)
 - removePoint, [19](#)
 - removePoints, [19](#)
 - setClasses, [20](#)
 - setDim, [20](#)
 - setFeaturesNames, [20](#)
 - setPoint, [21](#)
 - write, [21](#)
- dot
 - Point, [51](#)
- dtoa
 - Utils.cpp, [75](#)
 - Utils.hpp, [68](#)
- DualClassifier, [22](#)
 - getSolution, [23](#)
 - setKernel, [23](#)
- evaluate
 - Classifier, [9](#)
 - PerceptronDual, [42](#)
 - PerceptronFixedMarginDual, [45](#)
 - PerceptronFixedMarginPrimal, [47](#)
 - PerceptronPrimal, [50](#)
- FeatureSelection, [23](#)
- function
 - Kernel, [39](#)
- getDim
 - Data, [14](#)
- getDistCenters
 - Statistics, [56](#)
- getDistCentersWithoutFeats
 - Statistics, [57](#)
- getFeatureMean
 - Statistics, [57](#)
- getFeatureStddev
 - Statistics, [57](#)
- getFeaturesNames
 - Data, [14](#)
- getIndex
 - Data, [14](#)
- getKernelMatrix
 - Kernel, [39](#)
- getNumberNegativePoints
 - Data, [14](#)
- getNumberPositivePoints
 - Data, [15](#)
- getPoint
 - Data, [15](#)
- getPoints
 - Data, [15](#)
- getRadius
 - Statistics, [58](#)
- getSize
 - Data, [15](#)
- getSolution
 - DualClassifier, [23](#)
 - PrimalClassifier, [53](#)
- getStatistics
 - Data, [16](#)
- getSteps
 - Classifier, [9](#)
- getUpdates

- Classifier, 9
- Gnuplot, 24
 - is_valid, 26
 - operator<<, 26
 - plot_equation, 27
 - plot_equation3d, 27
 - plot_image, 27
 - plot_x, 27
 - plot_xy, 28
 - plot_xy_err, 28
 - plotfile_x, 28
 - plotfile_xy, 28
 - plotfile_xy_err, 28
 - plotfile_xyz, 29
 - replot, 29
 - set_GNUPlotPath, 29
 - set_contour, 29
 - set_hidden3d, 30
 - set_legend, 30
 - set_multiplot, 30
 - set_smooth, 30
 - set_style, 31
 - set_surface, 31
 - set_terminal_std, 31
 - set_title, 31
 - set_xautoscale, 32
 - set_yautoscale, 32
 - set_zautoscale, 32
 - unset_contour, 33
 - unset_hidden3d, 33
 - unset_legend, 33
 - unset_multiplot, 34
 - unset_smooth, 34
 - unset_surface, 34
 - unset_title, 35
 - unset_xlogscale, 35
 - unset_ylogscale, 35
 - unset_zlogscale, 36
- GnuplotException, 36
- includes/Utils.hpp, 67
- insertFeatures
 - Data, 16
- insertPoint
 - Data, 16, 17
- is_number
 - Utils.hpp, 69
- is_valid
 - Gnuplot, 26
- isEmpty
 - Data, 17
- isNormalized
 - Data, 17
- itos
 - Utils.cpp, 75
 - Utils.hpp, 69
- join
 - Data, 17
- Kernel, 38
 - compute, 39
 - function, 39
 - getKernelMatrix, 39
 - Kernel, 38
 - norm, 40
 - setKernelMatrix, 40
 - setParam, 40
 - setType, 40
- load
 - Data, 18
- maxAbsElement
 - Utils.hpp, 69
- mean
 - Statistics, 58
- norm
 - Kernel, 40
 - Point, 52
- normalize
 - Data, 18
- operator<<
 - Gnuplot, 26
- partTrainTest
 - Validation, 60
- PerceptronDual, 41
 - evaluate, 42
 - train, 43
- PerceptronFixedMarginDual, 43
 - evaluate, 45
 - train, 45
- PerceptronFixedMarginPrimal, 46
 - evaluate, 47
 - train, 48
- PerceptronPrimal, 48
 - evaluate, 50
 - train, 50
- plot2DwithHyperplane
 - Visualisation, 63
- plot2D
 - Visualisation, 62
- plot3DwithHyperplane
 - Visualisation, 64
- plot3D
 - Visualisation, 63
- plot_equation
 - Gnuplot, 27
- plot_equation3d
 - Gnuplot, 27
- plot_image
 - Gnuplot, 27
- plot_x
 - Gnuplot, 27
- plot_xy
 - Gnuplot, 28

- plot_xy_err
 - Gnuplot, 28
- plotfile_x
 - Gnuplot, 28
- plotfile_xy
 - Gnuplot, 28
- plotfile_xy_err
 - Gnuplot, 28
- plotfile_xyz
 - Gnuplot, 29
- Point, 51
 - dot, 51
 - norm, 52
- PrimalClassifier, 52
 - getSolution, 53
 - setNorm, 54
- removeFeatures
 - Data, 19
- removePoint
 - Data, 19
- removePoints
 - Data, 19
- replot
 - Gnuplot, 29
- set_GNUPlotPath
 - Gnuplot, 29
- set_contour
 - Gnuplot, 29
- set_hidden3d
 - Gnuplot, 30
- set_legend
 - Gnuplot, 30
- set_multiplot
 - Gnuplot, 30
- set_smooth
 - Gnuplot, 30
- set_style
 - Gnuplot, 31
- set_surface
 - Gnuplot, 31
- set_terminal_std
 - Gnuplot, 31
- set_title
 - Gnuplot, 31
- set_xautoscale
 - Gnuplot, 32
- set_yautoscale
 - Gnuplot, 32
- set_zautoscale
 - Gnuplot, 32
- setClasses
 - Data, 20
- setDim
 - Data, 20
- setFeaturesNames
 - Data, 20
- setKernel
 - DualClassifier, 23
- setKernelMatrix
 - Kernel, 40
- setNorm
 - PrimalClassifier, 54
- setParam
 - Kernel, 40
- setPoint
 - Data, 21
- setSample
 - Visualisation, 64
- setSamples
 - Classifier, 10
- setStyle
 - Visualisation, 64
- setTitle
 - Visualisation, 65
- setType
 - Kernel, 40
- Solution, 54
- src/Data.cpp, 71
- src/Point.cpp, 72
- src/Utils.cpp, 73
- Statistics, 55
 - getDistCenters, 56
 - getDistCentersWithoutFeats, 57
 - getFeatureMean, 57
 - getFeatureStdev, 57
 - getRadius, 58
 - mean, 58
 - stdev, 59
 - variance, 59
- stdev
 - Statistics, 59
- stodn
 - Utils.hpp, 71
- stoin
 - Utils.hpp, 71
- train
 - Classifier, 10
 - PerceptronDual, 43
 - PerceptronFixedMarginDual, 45
 - PerceptronFixedMarginPrimal, 48
 - PerceptronPrimal, 50
- unset_contour
 - Gnuplot, 33
- unset_hidden3d
 - Gnuplot, 33
- unset_legend
 - Gnuplot, 33
- unset_multiplot
 - Gnuplot, 34
- unset_smooth
 - Gnuplot, 34
- unset_surface
 - Gnuplot, 34
- unset_title

- Gnuplot, [35](#)
- unset_xlogscale
 - Gnuplot, [35](#)
- unset_ylogscale
 - Gnuplot, [35](#)
- unset_zlogscale
 - Gnuplot, [36](#)
- Utils.cpp
 - dtoa, [75](#)
 - itos, [75](#)
- Utils.hpp
 - dtoa, [68](#)
 - is_number, [69](#)
 - itos, [69](#)
 - maxAbsElement, [69](#)
 - stodn, [71](#)
 - stoin, [71](#)
- Validation, [60](#)
 - partTrainTest, [60](#)
- ValidationSolution, [61](#)
- variance
 - Statistics, [59](#)
- Visualisation, [62](#)
 - plot2DwithHyperplane, [63](#)
 - plot2D, [62](#)
 - plot3DwithHyperplane, [64](#)
 - plot3D, [63](#)
 - setSample, [64](#)
 - setStyle, [64](#)
 - setTitle, [65](#)
- write
 - Data, [21](#)