

HappyML

1.0

Generated by Doxygen 1.8.11

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	9
5.1	happyml Namespace Reference	9
5.1.1	Detailed Description	10
5.1.2	Typedef Documentation	10
5.1.2.1	Input	10
5.1.3	Function Documentation	10
5.1.3.1	greet()	10
5.1.3.2	sgn(double x)	10
5.1.3.3	sigmoid(double x)	11
5.1.4	Variable Documentation	11
5.1.4.1	version	11
5.2	happyml::colors Namespace Reference	11
5.2.1	Variable Documentation	12

5.2.1.1	BLUE	12
5.2.1.2	CYAN	12
5.2.1.3	GREEN	12
5.2.1.4	PINK	12
5.2.1.5	RED	12
5.2.1.6	RESET	12
5.3	happyml::tools Namespace Reference	12
5.3.1	Typedef Documentation	12
5.3.1.1	dictionary	12
5.3.2	Function Documentation	13
5.3.2.1	modelToDot(const LinearModel &lm, const string &filename, bool latex=false)	13
5.3.2.2	modelToDot(const LinearModel &lm, ostream &out, bool latex=false)	14
5.3.2.3	substitute(const string &in, const dictionary &dic)	14
5.3.3	Variable Documentation	14
5.3.3.1	edgeTemplate	14
5.3.3.2	linearModelTemplate	15
6	Class Documentation	17
6.1	happyml::Classifier Class Reference	17
6.1.1	Detailed Description	17
6.2	happyml::DataSet Class Reference	17
6.2.1	Detailed Description	18
6.2.2	Constructor & Destructor Documentation	18
6.2.2.1	DataSet(unsigned dim=0, unsigned size=0)	18
6.2.3	Member Function Documentation	18
6.2.3.1	read(istream &stream)	18
6.2.3.2	write(ostream &stream) const	19
6.2.4	Member Data Documentation	19
6.2.4.1	d	19
6.2.4.2	N	19
6.2.4.3	X	19

6.2.4.4	y	19
6.3	happyml::LinearModel Class Reference	20
6.3.1	Detailed Description	20
6.3.2	Constructor & Destructor Documentation	20
6.3.2.1	LinearModel(unsigned d=0)	20
6.3.2.2	LinearModel(const vec &weights)	21
6.3.2.3	LinearModel(const LinearModel &lm)	21
6.3.2.4	~LinearModel()	21
6.3.3	Member Function Documentation	21
6.3.3.1	getWeights() const	21
6.3.4	Member Data Documentation	21
6.3.4.1	w	21
6.4	happyml::LinearRegression Class Reference	22
6.4.1	Constructor & Destructor Documentation	22
6.4.1.1	LinearRegression(unsigned d=0)	22
6.4.1.2	LinearRegression(const vec &weights)	22
6.4.1.3	LinearRegression(const LinearModel &lm)	23
6.4.2	Member Function Documentation	23
6.4.2.1	error(const DataSet &data) const	23
6.4.2.2	predict(const Input &x) const	23
6.4.2.3	train(const DataSet &data)	23
6.4.2.4	train(const DataSet &data, double lambda)	24
6.5	happyml::LogisticRegression Class Reference	24
6.5.1	Constructor & Destructor Documentation	24
6.5.1.1	LogisticRegression(unsigned d=0)	24
6.5.1.2	LogisticRegression(const vec &weights)	25
6.5.1.3	LogisticRegression(const LinearModel &lm)	25
6.5.2	Member Function Documentation	25
6.5.2.1	error(const DataSet &data) const	25
6.5.2.2	predict(const Input &x) const	25

6.5.2.3	train(const DataSet &data, unsigned iter=1000, double learning_rate=0.1)	25
6.6	happyml::NeuralNetwork Class Reference	26
6.6.1	Constructor & Destructor Documentation	26
6.6.1.1	NeuralNetwork(unsigned layers...)	26
6.6.1.2	NeuralNetwork(const NeuralNetwork &nn)	27
6.6.1.3	~NeuralNetwork()	27
6.6.2	Member Function Documentation	27
6.6.2.1	getWeights() const	27
6.6.2.2	predict(const Input &x) const	27
6.6.2.3	train(const DataSet &dataset, unsigned iter=500, float learning_rate=0.1, float lambda=0)	27
6.7	happyml::Perceptron Class Reference	28
6.7.1	Constructor & Destructor Documentation	28
6.7.1.1	Perceptron(unsigned d=0)	28
6.7.1.2	Perceptron(const vec &weights)	29
6.7.1.3	Perceptron(const LinearModel &lm)	29
6.7.2	Member Function Documentation	29
6.7.2.1	error(const DataSet &data) const	29
6.7.2.2	predict(const Input &x) const	29
6.7.2.3	train(const DataSet &data, unsigned iter)	29
6.8	happyml::Predictor Class Reference	30
6.8.1	Detailed Description	30
6.8.2	Member Function Documentation	30
6.8.2.1	error(const Input &x, double y) const	30
6.8.2.2	error(const DataSet &dataset) const	31
6.8.2.3	predict(const Input &x) const =0	31
6.8.2.4	saveSampling(const string &filename, double minx_1, double maxx_1, unsigned samples_1, double minx_2, double maxx_2, unsigned samples_2, const Transformer &t=Transformer()) const	31
6.8.2.5	saveSampling(const string &filename, double minx, double maxx, unsigned samples, const Transformer &t=Transformer()) const	32
6.9	happyml::Serializable Class Reference	33

6.9.1	Detailed Description	33
6.9.2	Member Function Documentation	33
6.9.2.1	load(const string &filename)	33
6.9.2.2	read(istream &stream)=0	33
6.9.2.3	save(const string &filename) const	34
6.9.2.4	write(ostream &stream) const =0	34
6.10	happyml::SVM Class Reference	34
6.10.1	Detailed Description	35
6.10.2	Constructor & Destructor Documentation	35
6.10.2.1	SVM()	35
6.10.3	Member Function Documentation	35
6.10.3.1	error(const DataSet &data) const	35
6.10.3.2	predict(const Input &x) const	35
6.10.3.3	train(DataSet &data, double C=1, unsigned iter=5, double tolerance=0.001)	36
6.11	happyml::Transformer Class Reference	36
6.11.1	Detailed Description	37
6.11.2	Constructor & Destructor Documentation	37
6.11.2.1	Transformer()	37
6.11.2.2	~Transformer()	37
6.11.3	Member Function Documentation	37
6.11.3.1	addAddition(unsigned feature, double n, bool create_new=false)	37
6.11.3.2	addPower(unsigned feature, double power, bool create_new=true)	37
6.11.3.3	addProduct(unsigned feature, double n, bool create_new=false)	37
6.11.3.4	apply(DataSet &dataset)	38
6.11.3.5	apply(const Input &input) const	38
6.11.3.6	normalize()	38
6.11.3.7	pca(int k)	38
6.11.3.8	pcaMinVariance(double pcaVar)	38
6.11.3.9	remove(unsigned feature)	39

7 File Documentation	41
7.1 include/happyml.h File Reference	41
7.2 include/happyml/dataset.h File Reference	42
7.3 include/happyml/happytools.h File Reference	42
7.4 include/happyml/linear_model.h File Reference	43
7.5 include/happyml/linear_regression/linear_regression.h File Reference	43
7.6 include/happyml/logistic_regression/logistic_regression.h File Reference	43
7.7 include/happyml/neural_network/neural_network.h File Reference	44
7.8 include/happyml/perceptron/perceptron.h File Reference	44
7.9 include/happyml/predictor.h File Reference	44
7.10 include/happyml/serializable.h File Reference	45
7.11 include/happyml/svm/svm.h File Reference	45
7.12 include/happyml/transformer.h File Reference	46
7.13 include/happyml/types.h File Reference	46
7.14 include/happyml/utils.h File Reference	46
7.15 src/dataset.cpp File Reference	47
7.16 src/happyml.cpp File Reference	47
7.17 src/happytools.cpp File Reference	48
7.18 src/linear_regression/linear_regression.cpp File Reference	48
7.19 src/logistic_regression/logistic_regression.cpp File Reference	48
7.20 src/neural_network/neural_network.cpp File Reference	49
7.21 src/perceptron/perceptron.cpp File Reference	49
7.22 src/predictor.cpp File Reference	49
7.23 src/serializable.cpp File Reference	50
7.24 src/svm/svm.cpp File Reference	50
7.25 src/transformer.cpp File Reference	50
7.26 src/utils.cpp File Reference	50
Index	51

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

happyml	HappyML library namespace	9
happyml::colors	11
happyml::tools	12

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

happyml::LinearModel	20
happyml::LinearRegression	22
happyml::LogisticRegression	24
happyml::Perceptron	28
happyml::Predictor	30
happyml::Classifier	17
happyml::LinearRegression	22
happyml::LogisticRegression	24
happyml::NeuralNetwork	26
happyml::Perceptron	28
happyml::SVM	34
happyml::Serializable	33
happyml::DataSet	17
happyml::Transformer	36

Chapter 3

Class Index

3.1 Class List

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

happyml::Classifier	Abstract class that represent an algorithm that classifies an input in classes	17
happyml::DataSet	Generic collection of inputs and outputs	17
happyml::LinearModel	Hypothesis of the form $w_0 \cdot x_0 + w_1 \cdot x_1 + \dots + w_d \cdot x_d$	20
happyml::LinearRegression	22
happyml::LogisticRegression	24
happyml::NeuralNetwork	26
happyml::Perceptron	28
happyml::Predictor	Abstract class that represent an algorithm that predict an output or classifies an input vector . .	30
happyml::Serializable	Abstract class that represent an object that can be loaded from a file and saved to a file	33
happyml::SVM	Support vector machine with linear kernel	34
happyml::Transformer	Class that applies linear and non-linear transformations to an input or to a whole dataset . . .	36

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

include/happyml.h	41
include/happyml/dataset.h	42
include/happyml/happytools.h	42
include/happyml/linear_model.h	43
include/happyml/predictor.h	44
include/happyml/serializable.h	45
include/happyml/transformer.h	46
include/happyml/types.h	46
include/happyml/utils.h	46
include/happyml/linear_regression/linear_regression.h	43
include/happyml/logistic_regression/logistic_regression.h	43
include/happyml/neural_network/neural_network.h	44
include/happyml/perceptron/perceptron.h	44
include/happyml/svm/svm.h	45
src/dataset.cpp	47
src/happyml.cpp	47
src/happytools.cpp	48
src/predictor.cpp	49
src/serializable.cpp	50
src/transformer.cpp	50
src/utils.cpp	50
src/linear_regression/linear_regression.cpp	48
src/logistic_regression/logistic_regression.cpp	48
src/neural_network/neural_network.cpp	49
src/perceptron/perceptron.cpp	49
src/svm/svm.cpp	50

Chapter 5

Namespace Documentation

5.1 happymml Namespace Reference

HappyML library namespace.

Namespaces

- [colors](#)
- [tools](#)

Classes

- class [Classifier](#)
Abstract class that represent an algorithm that classifies an input in classes.
- class [DataSet](#)
Generic collection of inputs and outputs.
- class [LinearModel](#)
Hypothesis of the form $w_0 \cdot x_0 + w_1 \cdot x_1 + \dots + w_d \cdot x_d$.
- class [LinearRegression](#)
- class [LogisticRegression](#)
- class [NeuralNetwork](#)
- class [Perceptron](#)
- class [Predictor](#)
Abstract class that represent an algorithm that predict an output or classifies an input vector.
- class [Serializable](#)
Abstract class that represent an object that can be loaded from a file and saved to a file.
- class [SVM](#)
Support vector machine with linear kernel.
- class [Transformer](#)
Class that applies linear and non-linear transformations to an input or to a whole dataset.

Typedefs

- typedef vec [Input](#)
Vector of inputs features.

Functions

- void `greet` ()
Prints a greeting to the standard output.
- int `sgn` (double x)
Default sign function.
- double `sigmoid` (double x)
Implementation of the sigmoid math function: $g(x) = \frac{1}{1 + e^{-x}}$.

Variables

- const string `version` = HAPPY_ML_VERSION
String with the library version.

5.1.1 Detailed Description

HappyML library namespace.

All in this library is under this namespace.

5.1.2 Typedef Documentation

5.1.2.1 typedef vec happyml::Input

Vector of inputs features.

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_d \end{pmatrix}$$

5.1.3 Function Documentation

5.1.3.1 void happyml::greet ()

Prints a greeting to the standard output.

This library it's not only a **happy** library, it's also a **polite** library :)

5.1.3.2 int happyml::sgn (double x)

Default sign function.

$$\text{sgn}(x) := \begin{cases} -1 & \text{if } x < 0, \\ 1 & \text{if } x \geq 0. \end{cases}$$

Parameters

x	Input number.
-----	---------------

Returns

Sign of x .

5.1.3.3 double happymml::sigmoid (double x)

Implementation of the sigmoid math function: $g(x) = \frac{1}{1 + e^{-x}}$.

Parameters

x	Input number.
-----	---------------

Returns

Sigmoid value.

5.1.4 Variable Documentation

5.1.4.1 const string happymml::version = HAPPY_ML_VERSION

String with the library version.

5.2 happymml::colors Namespace Reference

Variables

- const string **CYAN** = "\e[46m"
- const string **GREEN** = "\e[32m"
- const string **PINK** = "\e[45m\e[37m"
- const string **BLUE** = "\e[44m\e[37m"
- const string **RED** = "\e[41m\e[37m"
- const string **RESET** = "\e[m"

5.2.1 Variable Documentation

5.2.1.1 `const string happymml::colors::BLUE = "\e[44m\e[37m"`

5.2.1.2 `const string happymml::colors::CYAN = "\e[46m"`

5.2.1.3 `const string happymml::colors::GREEN = "\e[32m"`

5.2.1.4 `const string happymml::colors::PINK = "\e[45m\e[37m"`

5.2.1.5 `const string happymml::colors::RED = "\e[41m\e[37m"`

5.2.1.6 `const string happymml::colors::RESET = "\e[m"`

5.3 happymml::tools Namespace Reference

Typedefs

- typedef `map< string, string >` [dictionary](#)
Dictionary containing variables names as keys and his values.

Functions

- void [modelToDot](#) (const [LinearModel](#) &lm, const string &filename, bool latex=false)
Creates a DOT (graph description language) file of the linear model.
- void [modelToDot](#) (const [LinearModel](#) &lm, ostream &out, bool latex=false)
- string [substitute](#) (const string &in, const [dictionary](#) &dic)
Returns an new string created by substituting vars in the input string.

Variables

- string [linearModelTemplate](#)
This string is a template for a DOT file.
- string [edgeTemplate](#)

5.3.1 Typedef Documentation

5.3.1.1 `typedef map<string, string> happymml::tools::dictionary`

Dictionary containing variables names as keys and his values.

See also

[substitute](#)

5.3.2 Function Documentation

5.3.2.1 `void happymtl::tools::modelToDot (const LinearModel & lm, const string & filename, bool latex = false)`

Creates a DOT (graph description language) file of the linear model.

You can generate a png image of a file with the next command:

```
dot -Tpng -o<OUTPUT-FILE.PNG> -Gsize=9,15\! -Gdpi=100 <INPUT-FILE.DOT>
```

The DOT file can also be generated for Latex (formulas can be included in nodes or edges names). For generating a DOT-Latex files use dot2tex. First generate the tex file from your DOT file and then compile it with pdflatex:

```
dot2tex <MODEL.DOT> > file.tex
pdflatex file.tex
```

Parameters

<i>lm</i>	Linear model to represent.
<i>filename</i>	Name of the file where the DOT info will be saved.
<i>latex</i>	True to use latex in the labels. False by default.

5.3.2.2 `void happym::tools::modelToDot (const LinearModel & lm, ostream & out, bool latex = false)`

See also

[modelToDot\(const LinearModel&, const string&, bool\)](#)

5.3.2.3 `string happym::tools::substitute (const string & in, const dictionary & dic)`

Returns an new string created by substituting vars in the input string.

The syntax for the vars in the 'in' string is {{var_name}}.

Thanks Potatoswatter (StackOverflow user) for the code of this function.

Parameters

<i>in</i>	Input string with vars names to substitute.
<i>dic</i>	Dictionary with variables names as keys and another string as value.

Returns

A new string with the variables names substituted by its values.

See also

[dictionary](#)

5.3.3 Variable Documentation

5.3.3.1 `string happym::tools::edgeTemplate`

Initial value:

```
= "x{{i}} -> output [label=\"{{weight}}\", \"
    \"color=\"0 0 {{color}}\",{{color_int}}\"
    \"fontcolor=\"0 0 {{color}}\";\"
```

5.3.3.2 string happym!::tools::linearModelTemplate

This string is a template for a DOT file.

In this template you can substitute 3 variables:

title: Title of the graph. nodes: x_1 x_2 ... x_d edges: x_1 -> output; x_2 -> output ... or see edgeTemplate.

See also

[edgeTemplate](#)

Chapter 6

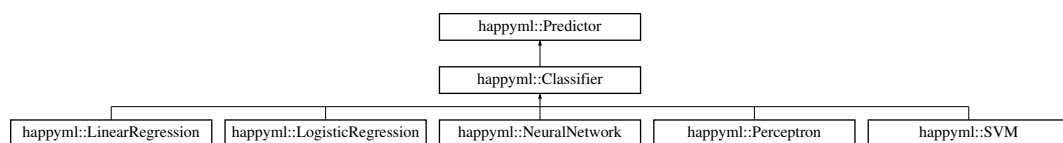
Class Documentation

6.1 happymml::Classifier Class Reference

Abstract class that represent an algorithm that classifies an input in classes.

```
#include <predictor.h>
```

Inheritance diagram for happymml::Classifier:



Additional Inherited Members

6.1.1 Detailed Description

Abstract class that represent an algorithm that classifies an input in classes.

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

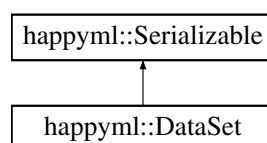
- `include/happymml/predictor.h`

6.2 happymml::DataSet Class Reference

Generic collection of inputs and outputs.

```
#include <dataset.h>
```

Inheritance diagram for happymml::DataSet:



Public Member Functions

- `DataSet` (unsigned dim=0, unsigned size=0)
Creates a generic collection of inputs and outputs.
- void `read` (istream &stream)
Creates a dataset from a text input stream with the following format:
- void `write` (ostream &stream) const
Write to an output stream the next data:

Public Attributes

- unsigned `d`
Dimension of the inputs vectors of this dataset (d).
- unsigned `N`
Number of pairs (\mathbf{x} , y) in the dataset.
- mat `X`
Matrix with all the inputs.
- mat `y`
Expected outputs.

6.2.1 Detailed Description

Generic collection of inputs and outputs.

We denote it with \mathcal{D} .

6.2.2 Constructor & Destructor Documentation

6.2.2.1 `happymml::DataSet::DataSet (unsigned dim = 0, unsigned size = 0) [inline]`

Creates a generic collection of inputs and outputs.

Parameters

<i>dim</i>	Dimension d of the inputs.
<i>size</i>	Number of inputs (N) in the dataset.

6.2.3 Member Function Documentation

6.2.3.1 `void happymml::DataSet::read (istream & stream) [virtual]`

Creates a dataset from a text input stream with the following format:

$x_{01}, x_{02}, \dots, x_{0d}, y_0$

$$\begin{matrix} x_{11}, x_{12}, \dots, x_{1d}, y_1 \\ \vdots, \vdots, \dots, \vdots, \vdots \\ x_{(N-1)1}, x_{(N-1)2}, \dots, x_{(N-1)d}, y_{(N-1)} \end{matrix}$$

Implements [happymml::Serializable](#).

6.2.3.2 void happymml::DataSet::write (ostream & stream) const [virtual]

Write to an output stream the next data:

$$\begin{matrix} x_{01}, x_{02}, \dots, x_{0d}, y_0 \\ x_{11}, x_{12}, \dots, x_{1d}, y_1 \\ \vdots, \vdots, \dots, \vdots, \vdots \\ x_{(N-1)1}, x_{(N-1)2}, \dots, x_{(N-1)d}, y_{(N-1)} \end{matrix}$$

Implements [happymml::Serializable](#).

6.2.4 Member Data Documentation

6.2.4.1 unsigned happymml::DataSet::d

Dimension of the inputs vectors of this dataset (d).

All the points has the same dimension.

6.2.4.2 unsigned happymml::DataSet::N

Number of pairs (\mathbf{x}, y) in the dataset.

We denote it with N .

6.2.4.3 mat happymml::DataSet::X

Matrix with all the inputs.

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_0^T \\ \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_{N-1}^T \end{pmatrix} = \begin{pmatrix} x_{00} & \dots & x_{0d} \\ x_{10} & \dots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{(N-1)0} & \dots & x_{(N-1)d} \end{pmatrix}$$

6.2.4.4 mat happymml::DataSet::y

Expected outputs.

$$\mathbf{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix}$$

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

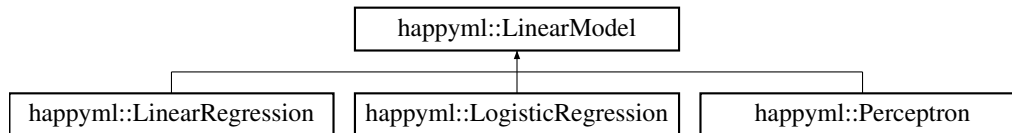
- include/happymml/[dataset.h](#)
- src/[dataset.cpp](#)

6.3 happymml::LinearModel Class Reference

Hypothesis of the form $w_0 \cdot x_0 + w_1 \cdot x_1 + \dots + w_d \cdot x_d$.

```
#include <linear_model.h>
```

Inheritance diagram for happymml::LinearModel:



Public Member Functions

- [LinearModel](#) (unsigned d=0)
Creates a linear model with the indicated input size.
- [LinearModel](#) (const vec &weights)
Creates a linear model with the indicated weights.
- [LinearModel](#) (const [LinearModel](#) &lm)
Creates a linear model from the weights of another linear model.
- [~LinearModel](#) ()
Destroys the linear model (sets the weights size to 0).
- vec [getWeights](#) () const
Get a copy of the model weights.

Protected Attributes

- vec [w](#)
Vector of weights.

6.3.1 Detailed Description

Hypothesis of the form $w_0 \cdot x_0 + w_1 \cdot x_1 + \dots + w_d \cdot x_d$.

In other words, each input feature has a weight associated.

This class contains a protected weight vector and a public getter.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 happymml::LinearModel::LinearModel (unsigned d = 0) [inline]

Creates a linear model with the indicated input size.

Parameters

d	Dimension d of the input feature vectors.
-----	---------------------------------------------

6.3.2.2 `happymml::LinearModel::LinearModel (const vec & weights)` `[inline]`

Creates a linear model with the indicated weights.

Parameters

<i>weights</i>	Weight vector with $d + 1$ dimensions.
----------------	----------------------------------------

6.3.2.3 `happymml::LinearModel::LinearModel (const LinearModel & lm)` `[inline]`

Creates a linear model from the weights of another linear model.

Parameters

<i>lm</i>	Linear model from which weights are copied.
-----------	---------------------------------------------

6.3.2.4 `happymml::LinearModel::~LinearModel ()` `[inline]`

Destroys the linear model (sets the weights size to 0).

6.3.3 Member Function Documentation

6.3.3.1 `vec happymml::LinearModel::getWeights () const` `[inline]`

Get a copy of the model weights.

Returns

Copy of the model weights.

6.3.4 Member Data Documentation

6.3.4.1 `vec happymml::LinearModel::w` `[protected]`

Vector of weights.

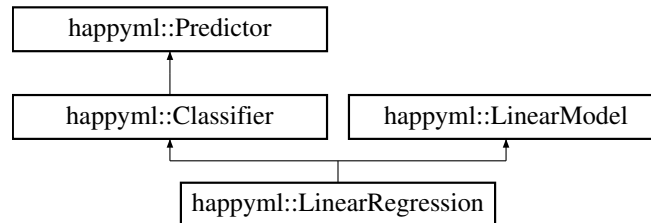
The documentation for this class was generated from the following file:

- `include/happymml/linear_model.h`

6.4 happymml::LinearRegression Class Reference

```
#include <linear_regression.h>
```

Inheritance diagram for happymml::LinearRegression:



Public Member Functions

- [LinearRegression](#) (unsigned $d=0$)
Creates a linear regression algorithm with the indicated input size.
- [LinearRegression](#) (const vec &weights)
Creates a linear regression algorithm with the indicated weights.
- [LinearRegression](#) (const [LinearModel](#) &lm)
- double [train](#) (const [DataSet](#) &data)
Train the linear regression.
- double [train](#) (const [DataSet](#) &data, double lambda)
Train the linear regression.
- double [predict](#) (const [Input](#) &x) const
Predict the output of an input vector.
- double [error](#) (const [DataSet](#) &data) const
Compute the mean squared error of the linear regression algorithm on the given dataset.

Additional Inherited Members

6.4.1 Constructor & Destructor Documentation

6.4.1.1 happymml::LinearRegression::LinearRegression (unsigned $d = 0$) [inline]

Creates a linear regression algorithm with the indicated input size.

Parameters

d	Dimension d , number of features of the input vectors.
-----	----------------------------------------------------------

6.4.1.2 happymml::LinearRegression::LinearRegression (const vec &weights) [inline]

Creates a linear regression algorithm with the indicated weights.

Parameters

<i>weights</i>	Weight vector with $d + 1$ size.
----------------	----------------------------------

6.4.1.3 `happymml::LinearRegression::LinearRegression (const LinearModel & lm)` `[inline]`

6.4.2 Member Function Documentation

6.4.2.1 `double happymml::LinearRegression::error (const DataSet & data) const` `[virtual]`

Compute the mean squared error of the linear regression algorithm on the given dataset.

Parameters

<i>data</i>	Dataset with the correct output.
-------------	----------------------------------

Returns

Error of classify the given dataset. It's value is grater than 0.

Reimplemented from [happymml::Predictor](#).

6.4.2.2 `double happymml::LinearRegression::predict (const Input & x) const` `[virtual]`

Predict the output of an input vector.

Returns

Estimated real output.

Implements [happymml::Predictor](#).

6.4.2.3 `double happymml::LinearRegression::train (const DataSet & data)`

Train the linear regression.

Parameters

<i>data</i>	Training set.
-------------	---------------

Returns

Returns the final error.

6.4.2.4 double happymml::LinearRegression::train (const DataSet & data, double lambda)

Train the linear regression.

Parameters

<i>data</i>	Training set.
<i>lambda</i>	Regularization paramiter λ .

Returns

Returns the final error.

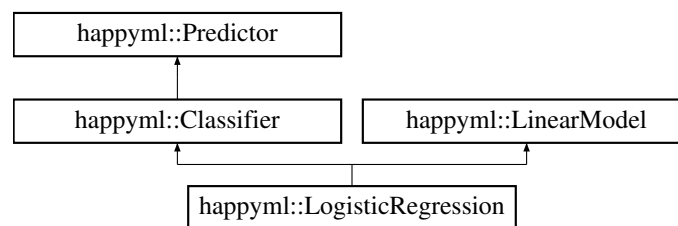
The documentation for this class was generated from the following files:

- include/happymml/linear_regression/linear_regression.h
- src/linear_regression/linear_regression.cpp

6.5 happymml::LogisticRegression Class Reference

```
#include <logistic_regression.h>
```

Inheritance diagram for happymml::LogisticRegression:



Public Member Functions

- [LogisticRegression](#) (unsigned d=0)
Creates a logistic regression algorithm with the indicated input size.
- [LogisticRegression](#) (const vec &weights)
Creates a logistic regression algorithm with the indicated weights.
- [LogisticRegression](#) (const [LinearModel](#) &lm)
- double [train](#) (const [DataSet](#) &data, unsigned iter=1000, double learning_rate=0.1)
Train the logistic regression until the training loops over the data nTimes, or classifies all correctly.
- double [predict](#) (const [Input](#) &x) const
Classifies an input vector.
- double [error](#) (const [DataSet](#) &data) const
Compute the error of the logistic regression on the given dataset.

Additional Inherited Members

6.5.1 Constructor & Destructor Documentation

6.5.1.1 happymml::LogisticRegression::LogisticRegression (unsigned d = 0) [inline]

Creates a logistic regression algorithm with the indicated input size.

Parameters

<i>d</i>	Dimension d of the input feature vectors.
----------	---------------------------------------------

6.5.1.2 `happymml::LogisticRegression::LogisticRegression (const vec & weights)` `[inline]`

Creates a logistic regression algorithm with the indicated weights.

Parameters

<i>weights</i>	Weight vector with $d + 1$ size.
----------------	----------------------------------

6.5.1.3 `happymml::LogisticRegression::LogisticRegression (const LinearModel & lm)` `[inline]`

6.5.2 Member Function Documentation

6.5.2.1 `double happymml::LogisticRegression::error (const DataSet & data) const` `[virtual]`

Compute the error of the logistic regression on the given dataset.

Parameters

<i>data</i>	Dataset with the correct output.
-------------	----------------------------------

Returns

Error of classify the given dataset. It's value is grater than 0.

Reimplemented from [happymml::Predictor](#).

6.5.2.2 `double happymml::LogisticRegression::predict (const Input & x) const` `[virtual]`

Classifies an input vector.

Returns

Probability of the input to belong to +1 class. Output in the interval $[0, 1]$.

Implements [happymml::Predictor](#).

6.5.2.3 `double happymml::LogisticRegression::train (const DataSet & data, unsigned iter = 1000, double learning_rate = 0.1)`

Train the logistic regression until the training loops over the data n Times, or classifies all correctly.

Data must contain the $x_0 = 1$ property.

Parameters

<i>data</i>	Training set.
<i>iter</i>	Maximun number of iterations (1000 by default).
<i>learning_rate</i>	Learning rate (0.1 by default.).

Returns

Returns the error of the best weights found.

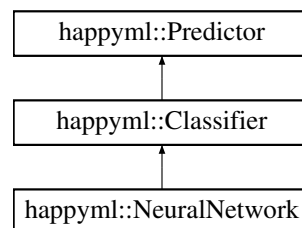
The documentation for this class was generated from the following files:

- [include/happymml/logistic_regression/logistic_regression.h](#)
- [src/logistic_regression/logistic_regression.cpp](#)

6.6 happymml::NeuralNetwork Class Reference

```
#include <neural_network.h>
```

Inheritance diagram for happymml::NeuralNetwork:



Public Member Functions

- [NeuralNetwork](#) (unsigned layers...)

Creates a neural network with the layers and number of neurons indicated in the arguments.
- [NeuralNetwork](#) (const [NeuralNetwork](#) &nn)

Creates a copy of a neural network.
- [~NeuralNetwork](#) ()
- const vector< mat > [getWeights](#) () const

Returns a copy of all the weights of the neural network.
- double [train](#) (const [DataSet](#) &dataset, unsigned iter=500, float learning_rate=0.1, float lambda=0)

Train the neural network until the training loops over the data iter times.
- double [predict](#) (const [Input](#) &x) const

Uses the forward propagation algorithm to predict an output.

6.6.1 Constructor & Destructor Documentation

6.6.1.1 happymml::NeuralNetwork::NeuralNetwork (unsigned layers...)

Creates a neural network with the layers and number of neurons indicated in the arguments.

Initialize the weight to a random values.

Parameters

<i>layers</i>	Number of layers, number of neurons on the input layer, number of neurons on the second layer...
---------------	--------------------------------------------------------------------------------------------------

6.6.1.2 happymml::NeuralNetwork::NeuralNetwork (const NeuralNetwork & nn)

Creates a copy of a neural network.

6.6.1.3 happymml::NeuralNetwork::~~NeuralNetwork ()

6.6.2 Member Function Documentation

6.6.2.1 const vector<mat> happymml::NeuralNetwork::getWeights () const [inline]

Returns a copy of all the weights of the neural network.

Returns

Copy of all the weights of the neural network.

6.6.2.2 double happymml::NeuralNetwork::predict (const Input & x) const [virtual]

Uses the forward propagation algorithm to predict an output.

Parameters

<i>x</i>	Input vector.
----------	---------------

Returns

Predicted value

Implements [happymml::Predictor](#).

6.6.2.3 double happymml::NeuralNetwork::train (const DataSet & dataset, unsigned iter = 500, float learning_rate = 0.1, float lambda = 0)

Train the neural network until the training loops over the data iter times.

Parameters

<i>dataset</i>	Training set \mathcal{D} .
<i>learningRate</i>	Learning rate η .
<i>iter</i>	Number of iterations.
<i>lambda</i>	Regularization parameter λ .

Returns

Returns the error after the training.

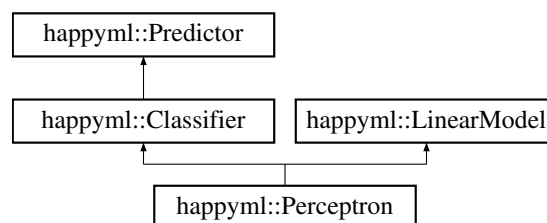
The documentation for this class was generated from the following files:

- include/happymml/neural_network/neural_network.h
- src/neural_network/neural_network.cpp

6.7 happymml::Perceptron Class Reference

```
#include <perceptron.h>
```

Inheritance diagram for happymml::Perceptron:

**Public Member Functions**

- [Perceptron](#) (unsigned d=0)
Creates a perceptron with the indicated input size.
- [Perceptron](#) (const vec &weights)
Creates a perceptron with the indicated weights.
- [Perceptron](#) (const [LinearModel](#) &lm)
- double [train](#) (const [DataSet](#) &data, unsigned iter)
Train the perceptron until the training loops over the data nTimes, or classifies all correctly.
- double [predict](#) (const [Input](#) &x) const
Classifies an input vector.
- double [error](#) (const [DataSet](#) &data) const
Compute the error of the perceptron on the given dataset.

Additional Inherited Members

6.7.1 Constructor & Destructor Documentation

6.7.1.1 happymml::Perceptron::Perceptron (unsigned d = 0) [inline]

Creates a perceptron with the indicated input size.

Parameters

<i>d</i>	Dimension d of the input feature vectors.
----------	---------------------------------------------

6.7.1.2 `happymml::Perceptron::Perceptron (const vec & weights)` `[inline]`

Creates a perceptron with the indicated weights.

Parameters

<i>weights</i>	Weight vector with $d + 1$ dimensions.
----------------	----------------------------------------

6.7.1.3 `happymml::Perceptron::Perceptron (const LinearModel & lm)` `[inline]`

6.7.2 Member Function Documentation

6.7.2.1 `double happymml::Perceptron::error (const DataSet & data) const` `[virtual]`

Compute the error of the perceptron on the given dataset.

Parameters

<i>data</i>	Dataset with the correct output.
-------------	----------------------------------

Returns

Error of classify the given dataset. It's in the interval $[0, 1]$.

Reimplemented from [happymml::Predictor](#).

6.7.2.2 `double happymml::Perceptron::predict (const Input & x) const` `[virtual]`

Classifies an input vector.

Returns

-1 or $+1$.

Implements [happymml::Predictor](#).

6.7.2.3 `double happymml::Perceptron::train (const DataSet & data, unsigned iter)`

Train the perceptron until the training loops over the data n Times, or classifies all correctly.

Data must contain the $x_0 = 1$ property.

Parameters

<i>data</i>	Training set.
<i>iter</i>	Maximun number of iterations.

Returns

Returns the error of the best perceptron found.

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

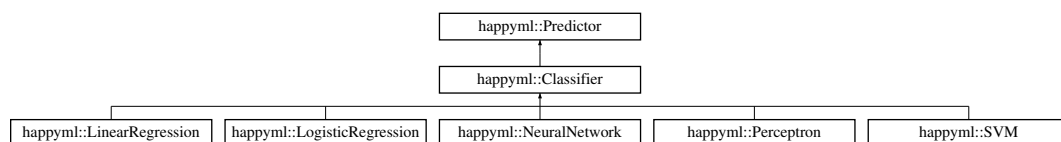
- include/happymml/perceptron/[perceptron.h](#)
- src/perceptron/[perceptron.cpp](#)

6.8 happymml::Predictor Class Reference

Abstract class that represent an algorithm that predict an output or classifies an input vector.

```
#include <predictor.h>
```

Inheritance diagram for happymml::Predictor:



Public Member Functions

- virtual double [predict](#) (const [Input](#) &x) const =0
Predict an output from an input vector.
- virtual double [error](#) (const [Input](#) &x, double y) const
Compute the error of the predictor in a given input.
- virtual double [error](#) (const [DataSet](#) &dataset) const
Compute the error of the predictor on the given dataset.
- virtual void [saveSampling](#) (const string &filename, double minx_1, double maxx_1, unsigned samples_1, double minx_2, double maxx_2, unsigned samples_2, const [Transformer](#) &t=[Transformer](#)()) const
Saves the value of predicting the output of the predictor on the rectangular area defined between $(minx_1, minx_2)$ and $(maxx_1, maxx_2)$ points.
- virtual void [saveSampling](#) (const string &filename, double minx, double maxx, unsigned samples, const [Transformer](#) &t=[Transformer](#)()) const
Saves the value of predicting the output of the predictor in the segment $[minx, maxx]$.

6.8.1 Detailed Description

Abstract class that represent an algorithm that predict an output or classifies an input vector.

6.8.2 Member Function Documentation

6.8.2.1 double happymml::Predictor::error (const [Input](#) &x, double y) const [virtual]

Compute the error of the predictor in a given input.

By default it computes the square diff.

Parameters

x	The input vector.
y	Correct output.

Returns

Error of the predicted value compare to y .

6.8.2.2 `double happymml::Predictor::error (const DataSet & dataset) const` `[virtual]`

Compute the error of the predictor on the given dataset.

Parameters

<i>data</i>	Dataset with the correct output.
-------------	----------------------------------

Returns

Error of predicting the output of the given dataset.

Reimplemented in [happymml::LinearRegression](#), [happymml::SVM](#), [happymml::LogisticRegression](#), and [happymml::Perceptron](#).

6.8.2.3 `virtual double happymml::Predictor::predict (const Input & x) const` `[pure virtual]`

Predict an output from an input vector.

Parameters

x	An input vector.
-----	------------------

Returns

Returns a real value corresponding with the prediction done by the current predictor.

Implemented in [happymml::NeuralNetwork](#), [happymml::LinearRegression](#), [happymml::SVM](#), [happymml::LogisticRegression](#), and [happymml::Perceptron](#).

6.8.2.4 `void happymml::Predictor::saveSampling (const string & filename, double minx_1, double maxx_1, unsigned samples_1, double minx_2, double maxx_2, unsigned samples_2, const Transformer & t = Transformer()) const` `[virtual]`

Saves the value of predicting the output of the predictor on the rectangular area defined between $(minx_1, minx_2)$ and $(maxx_1, maxx_2)$ points.

The CSV output file has the following structure:

```
minx1, maxx1, samples1
minx2, maxx2, samples2
h1 1, h1 2, ..., h1 samples1
⋮, ⋮, ..., ⋮
hsamples1 1, hsamples2 2, ..., hsamples2 samples1
```

Use 'happyplot' command line tool to visualize the output.

Parameters

<i>filename</i>	Output filename.
<i>minx_1</i>	Start value of the first feature.
<i>maxx_1</i>	End value of the first feature.
<i>samples_1</i>	Number of samples of the first feature.
<i>minx_2</i>	Start value of the second feature.
<i>maxx_2</i>	End value of the second feature.
<i>samples_2</i>	Number of samples of the second feature.
<i>t</i>	Transformer that transform the inputs to predict. By default a void transformed is used.

6.8.2.5 `void happymml::Predictor::saveSampling (const string & filename, double minx, double maxx, unsigned samples, const Transformer & t = Transformer()) const` [virtual]

Saves the value of predicting the output of the predictor in the segment $[minx, maxx]$.

The CSV output file has the following structure:

```
minx, maxx, samples
h1, h2, ..., hsamples
```

Use 'happyplot' command line tool to visualize the output.

Parameters

<i>filename</i>	Output filename.
<i>minx</i>	Start value of the first feature.
<i>maxx</i>	End value of the first feature.
<i>samples</i>	Number of samples of the first feature.
<i>t</i>	Transformer that transform the inputs to predict. By default a void transformed is used.

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

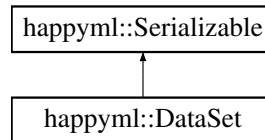
- include/happymml/[predictor.h](#)
- src/[predictor.cpp](#)

6.9 happymml::Serializable Class Reference

Abstract class that represent an object that can be loaded from a file and saved to a file.

```
#include <serializable.h>
```

Inheritance diagram for happymml::Serializable:



Public Member Functions

- virtual void **load** (const string &filename)
Fills an object using the data in the given file.
- virtual void **save** (const string &filename) const
Saves the object to a specific file.
- virtual void **read** (istream &stream)=0
Read the object from an input stream.
- virtual void **write** (ostream &stream) const =0
Write the object to an output stream.

6.9.1 Detailed Description

Abstract class that represent an object that can be loaded from a file and saved to a file.

6.9.2 Member Function Documentation

6.9.2.1 void happymml::Serializable::load (const string & *filename*) [virtual]

Fills an object using the data in the given file.

Parameters

<i>filename</i>	Name of the file to load.
-----------------	---------------------------

See also

save(const string&)

6.9.2.2 virtual void happymml::Serializable::read (istream & *stream*) [pure virtual]

Read the object from an input stream.

Parameters

<i>stream</i>	Input stream.
---------------	---------------

Implemented in [happyml::DataSet](#).

6.9.2.3 void happyml::Serializable::save (const string & *filename*) const [virtual]

Saves the object to a specific file.

After this call you can load new objects using the load method and that filename.

Parameters

<i>filename</i>	Output filename.
-----------------	------------------

See also

[load\(const string&\)](#)

6.9.2.4 virtual void happyml::Serializable::write (ostream & *stream*) const [pure virtual]

Write the object to an output stream.

Parameters

<i>stream</i>	Output stream.
---------------	----------------

Implemented in [happyml::DataSet](#).

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

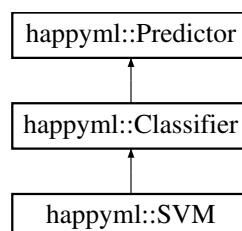
- include/happyml/[serializable.h](#)
- src/[serializable.cpp](#)

6.10 happyml::SVM Class Reference

Support vector machine with linear kernel.

```
#include <svm.h>
```

Inheritance diagram for happyml::SVM:



Public Member Functions

- [SVM](#) ()
Creates a [SVM](#) without any support vector.
- double [train](#) ([DataSet](#) &data, double C=1, unsigned iter=5, double tolerance=0.001)
Train the [SVM](#) using the given dataset and the given parameters.
- double [predict](#) (const [Input](#) &x) const
Classifies an input vector.
- double [error](#) (const [DataSet](#) &data) const
Compute the error of the [SVM](#) on the given dataset.

6.10.1 Detailed Description

Support vector machine with linear kernel.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 `happymml::SVM::SVM () [inline]`

Creates a [SVM](#) without any support vector.

Needs to be trained.

6.10.3 Member Function Documentation

6.10.3.1 `double happymml::SVM::error (const DataSet & data) const [virtual]`

Compute the error of the [SVM](#) on the given dataset.

Parameters

<i>data</i>	Dataset with the correct output.
-------------	----------------------------------

Returns

Error of classify the given dataset.

Reimplemented from [happymml::Predictor](#).

6.10.3.2 `double happymml::SVM::predict (const Input & x) const [virtual]`

Classifies an input vector.

Returns

-1 or +1.

Implements [happymml::Predictor](#).

6.10.3.3 `double happyml::SVM::train (DataSet & data, double C = 1, unsigned iter = 5, double tolerance = 0.001)`

Train the [SVM](#) using the given dataset and the given parameters.

Parameters

<i>data</i>	Training set.
<i>C</i>	SVM regularization parameter.
<i>iter</i>	Maximum number of iterations that the SMO algorithm does all over the dataset.
<i>tolerance</i>	Tolerance that checks if a change in any alpha is significant to continue with other iteration.

Returns

Returns the error of the [SVM](#).

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

- include/happyml/svm/[svm.h](#)
- src/svm/[svm.cpp](#)

6.11 happyml::Transformer Class Reference

Class that applies linear and non-linear transformations to an input or to a whole dataset.

```
#include <transformer.h>
```

Public Member Functions

- [Transformer](#) ()
- [~Transformer](#) ()
- void [addPower](#) (unsigned feature, double power, bool create_new=true)
Raise the feature number feature to the power of power.
- void [addAddition](#) (unsigned feature, double n, bool create_new=false)
Adds (or subtract) a value to the indicated feature.
- void [addProduct](#) (unsigned feature, double n, bool create_new=false)
Multiply (or divide) the indicated feature by n.
- void [remove](#) (unsigned feature)
Deletes a feature.
- void [normalize](#) ()
Normalize the dataset when you call to apply.
- void [pca](#) (int k)
PCA extractor.
- void [pcaMinVariance](#) (double pcaVar)
PCA extractor.
- void [apply](#) (DataSet &dataset)
Applies all the transformations in the given dataset.
- [Input apply](#) (const [Input](#) &input) const
Applies all the transformations in the given input.

6.11.1 Detailed Description

Class that applies linear and non-linear transformations to an input or to a whole dataset.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 `happymml::Transformer::Transformer ()` `[inline]`

6.11.2.2 `happymml::Transformer::~~Transformer ()` `[inline]`

6.11.3 Member Function Documentation

6.11.3.1 `void happymml::Transformer::addAddition (unsigned feature, double n, bool create_new = false)`

Adds (or subtract) a value to the indicated feature.

Doesn't add a new feature.

Parameters

<i>feature</i>	<i>n</i> will be added to this feature number.
<i>n</i>	Number to be added.
<i>create_new</i>	Create a new feature or not.

6.11.3.2 `void happymml::Transformer::addPower (unsigned feature, double power, bool create_new = true)`

Raise the feature number *feature* to the power of *power*.

Parameters

<i>feature</i>	Feature to raise.
<i>power</i>	Exponent.
<i>create_new</i>	Create a new feature or not.

6.11.3.3 `void happymml::Transformer::addProduct (unsigned feature, double n, bool create_new = false)`

Multiply (or divide) the indicated feature by *n*.

Doesn't add a new feature.

Parameters

<i>feature</i>	<i>n</i> will be added to this feature number.
<i>n</i>	Number to be added.
<i>create_new</i>	Create a new feature or not.

6.11.3.4 void happymml::Transformer::apply (DataSet & *dataset*)

Applies all the transformations in the given dataset.

Parameters

<i>dataset</i>	Dataset to transform.
----------------	-----------------------

6.11.3.5 Input happymml::Transformer::apply (const Input & *input*) const

Applies all the transformations in the given input.

Parameters

<i>input</i>	Input to transform.
--------------	---------------------

Returns

A transformed version of the input.

6.11.3.6 void happymml::Transformer::normalize ()

Normalize the dataset when you call to apply.

The normalization is done after all the other transformations are performed.

6.11.3.7 void happymml::Transformer::pca (int *k*)

PCA extranctor.

Parameters

<i>k</i>	Number of dimensions to choose.
----------	---------------------------------

6.11.3.8 void happymml::Transformer::pcaMinVariance (double *pcaVar*)

PCA extractor.

Parameters

<i>pcaVar</i>	Choose <i>k</i> dimensions untils total variance is at least <i>pcaVar</i> .
---------------	------------------------------------------------------------------------------

6.11.3.9 void happymml::Transformer::remove (unsigned *feature*)

Deletes a feature.

Parameters

<i>feature</i>	Feature that will be removed.
----------------	-------------------------------

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

- include/happymml/[transformer.h](#)
- src/[transformer.cpp](#)

Chapter 7

File Documentation

7.1 include/happyml.h File Reference

```
#include <string>
#include <armadillo>
#include "happyml/types.h"
#include "happyml/utils.h"
#include "happyml/transformer.h"
#include "happyml/predictor.h"
#include "happyml/perceptron/perceptron.h"
#include "happyml/logistic_regression/logistic_regression.h"
#include "happyml/linear_regression/linear_regression.h"
#include "happyml/neural_network/neural_network.h"
#include "happyml/svm/svm.h"
#include "happyml/happytools.h"
```

Namespaces

- [happyml](#)
HappyML library namespace.

Functions

- void [happyml::greet](#) ()
***Prints a greeting** to the standard output.*

Variables

- const string [happyml::version](#) = HAPPY_ML_VERSION
String with the library version.

7.2 include/happyml/dataset.h File Reference

```
#include <armadillo>
#include "happyml/serializable.h"
```

Classes

- class [happyml::DataSet](#)
Generic collection of inputs and outputs.

Namespaces

- [happyml](#)
HappyML library namespace.

7.3 include/happyml/happytools.h File Reference

```
#include <map>
#include "happyml/types.h"
#include "happyml/predictor.h"
#include "happyml/linear_model.h"
```

Namespaces

- [happyml](#)
HappyML library namespace.
- [happyml::tools](#)

Typedefs

- typedef map< string, string > [happyml::tools::dictionary](#)
Dictionary containing variables names as keys and his values.

Functions

- void [happyml::tools::modelToDot](#) (const LinearModel &lm, const string &filename, bool latex=false)
Creates a DOT (graph description language) file of the linear model.
- void [happyml::tools::modelToDot](#) (const LinearModel &lm, ostream &out, bool latex=false)
- string [happyml::tools::substitute](#) (const string &in, const dictionary &dic)
Returns an new string created by substituting vars in the input string.

7.4 include/happymml/linear_model.h File Reference

```
#include <armadillo>
```

Classes

- class [happymml::LinearModel](#)
Hypothesis of the form $w_0 \cdot x_0 + w_1 \cdot x_1 + \dots + w_d \cdot x_d$.

Namespaces

- [happymml](#)
HappyML library namespace.

7.5 include/happymml/linear_regression/linear_regression.h File Reference

```
#include "happymml/types.h"  
#include "happymml/utils.h"  
#include "happymml/predictor.h"  
#include "happymml/linear_model.h"
```

Classes

- class [happymml::LinearRegression](#)

Namespaces

- [happymml](#)
HappyML library namespace.

7.6 include/happymml/logistic_regression/logistic_regression.h File Reference

```
#include "happymml/types.h"  
#include "happymml/utils.h"  
#include "happymml/predictor.h"  
#include "happymml/linear_model.h"
```

Classes

- class [happymml::LogisticRegression](#)

Namespaces

- [happymml](#)

HappyML library namespace.

7.7 include/happymml/neural_network/neural_network.h File Reference

```
#include "happymml/types.h"
#include "happymml/utis.h"
#include "happymml/predictor.h"
```

Classes

- class [happymml::NeuralNetwork](#)

Namespaces

- [happymml](#)

HappyML library namespace.

7.8 include/happymml/perceptron/perceptron.h File Reference

```
#include "happymml/types.h"
#include "happymml/predictor.h"
#include "happymml/linear_model.h"
```

Classes

- class [happymml::Perceptron](#)

Namespaces

- [happymml](#)

HappyML library namespace.

7.9 include/happymml/predictor.h File Reference

```
#include "happymml/types.h"
#include "happymml/transformer.h"
```

Classes

- class [happyml::Predictor](#)
Abstract class that represent an algorithm that predict an output or classifies an input vector.
- class [happyml::Classifier](#)
Abstract class that represent an algorithm that classifies an input in classes.

Namespaces

- [happyml](#)
HappyML library namespace.

7.10 include/happyml/serializable.h File Reference

```
#include <string>
```

Classes

- class [happyml::Serializable](#)
Abstract class that represent an object that can be loaded from a file and saved to a file.

Namespaces

- [happyml](#)
HappyML library namespace.

7.11 include/happyml/svm/svm.h File Reference

```
#include "happyml/types.h"  
#include "happyml/predictor.h"
```

Classes

- class [happyml::SVM](#)
Support vector machine with linear kernel.

Namespaces

- [happyml](#)
HappyML library namespace.

7.12 include/happymml/transformer.h File Reference

```
#include <queue>
#include <vector>
#include "happymml/types.h"
```

Classes

- class [happymml::Transformer](#)

Class that applies linear and non-linear transformations to an input or to a whole dataset.

Namespaces

- [happymml](#)

HappyML library namespace.

7.13 include/happymml/types.h File Reference

```
#include <armadillo>
#include <istream>
#include <ostream>
#include "happymml/dataset.h"
```

Namespaces

- [happymml](#)

HappyML library namespace.

Typedefs

- typedef vec [happymml::Input](#)

Vector of inputs features.

7.14 include/happymml/utils.h File Reference

```
#include <string>
```

Namespaces

- [happymml](#)

HappyML library namespace.

- [happymml::colors](#)

Functions

- int `happyml::sgn` (double x)

Default sign function.

- double `happyml::sigmoid` (double x)

Implementation of the sigmoid math function: $g(x) = \frac{1}{1 + e^{-x}}$.

Variables

- const string `happyml::colors::CYAN` = "\e[46m"
- const string `happyml::colors::GREEN` = "\e[32m"
- const string `happyml::colors::PINK` = "\e[45m\e[37m"
- const string `happyml::colors::BLUE` = "\e[44m\e[37m"
- const string `happyml::colors::RED` = "\e[41m\e[37m"
- const string `happyml::colors::RESET` = "\e[m"

7.15 src/dataset.cpp File Reference

```
#include "happyml/dataset.h"
```

Namespaces

- `happyml`

HappyML library namespace.

7.16 src/happyml.cpp File Reference

```
#include <iostream>
#include <string>
#include "happyml.h"
```

Namespaces

- `happyml`

HappyML library namespace.

Functions

- void `happyml::greet` ()

Prints a greeting to the standard output.

7.17 src/happytools.cpp File Reference

```
#include "happyml/happytools.h"
#include <fstream>
#include <sstream>
#include <iomanip>
```

Namespaces

- [happyml](#)
HappyML library namespace.
- [happyml::tools](#)

Functions

- void [happyml::tools::modelToDot](#) (const LinearModel &lm, const string &filename, bool latex=false)
Creates a DOT (graph description language) file of the linear model.
- void [happyml::tools::modelToDot](#) (const LinearModel &lm, ostream &out, bool latex=false)
- string [happyml::tools::substitute](#) (const string &in, const dictionary &dic)
Returns an new string created by substituting vars in the input string.

Variables

- string [happyml::tools::linearModelTemplate](#)
This string is a template for a DOT file.
- string [happyml::tools::edgeTemplate](#)

7.18 src/linear_regression/linear_regression.cpp File Reference

```
#include "happyml/linear_regression/linear_regression.h"
#include <iomanip>
```

Namespaces

- [happyml](#)
HappyML library namespace.

7.19 src/logistic_regression/logistic_regression.cpp File Reference

```
#include "happyml/logistic_regression/logistic_regression.h"
#include <iomanip>
```


Namespaces

- [happyml](#)

HappyML library namespace.

7.20 src/neural_network/neural_network.cpp File Reference

```
#include "happyml/neural_network/neural_network.h"
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <iomanip>
#include <cstdlib>
```

Namespaces

- [happyml](#)

HappyML library namespace.

7.21 src/perceptron/perceptron.cpp File Reference

```
#include "happyml/perceptron/perceptron.h"
#include <iomanip>
#include "happyml/utils.h"
```

Namespaces

- [happyml](#)

HappyML library namespace.

7.22 src/predictor.cpp File Reference

```
#include "happyml/predictor.h"
```

Namespaces

- [happyml](#)

HappyML library namespace.

7.23 src/serializable.cpp File Reference

```
#include "happymml/serializable.h"
#include <fstream>
```

Namespaces

- [happymml](#)

HappyML library namespace.

7.24 src/svm/svm.cpp File Reference

```
#include "happymml/svm/svm.h"
#include <math.h>
#include "happymml/utils.h"
```

Namespaces

- [happymml](#)

HappyML library namespace.

7.25 src/transformer.cpp File Reference

```
#include "happymml/ttransformer.h"
```

Namespaces

- [happymml](#)

HappyML library namespace.

7.26 src/utils.cpp File Reference

```
#include "happymml/utils.h"
#include <math.h>
```

Namespaces

- [happymml](#)

HappyML library namespace.

Functions

- int [happymml::sgn](#) (double x)

Default sign function.

- double [happymml::sigmoid](#) (double x)

Implementation of the sigmoid math function: $g(x) = \frac{1}{1 + e^{-x}}$.

Index

- ~LinearModel
 - happyml::LinearModel, [21](#)
- ~NeuralNetwork
 - happyml::NeuralNetwork, [27](#)
- ~Transformer
 - happyml::Transformer, [37](#)
- addAddition
 - happyml::Transformer, [37](#)
- addPower
 - happyml::Transformer, [37](#)
- addProduct
 - happyml::Transformer, [37](#)
- apply
 - happyml::Transformer, [38](#)
- BLUE
 - happyml::colors, [12](#)
- CYAN
 - happyml::colors, [12](#)
- d
 - happyml::DataSet, [19](#)
- DataSet
 - happyml::DataSet, [18](#)
- dictionary
 - happyml::tools, [12](#)
- edgeTemplate
 - happyml::tools, [14](#)
- error
 - happyml::LinearRegression, [23](#)
 - happyml::LogisticRegression, [25](#)
 - happyml::Perceptron, [29](#)
 - happyml::Predictor, [30](#), [31](#)
 - happyml::SVM, [35](#)
- GREEN
 - happyml::colors, [12](#)
- getWeights
 - happyml::LinearModel, [21](#)
 - happyml::NeuralNetwork, [27](#)
- greet
 - happyml, [10](#)
- happyml, [9](#)
 - greet, [10](#)
 - Input, [10](#)
 - sgn, [10](#)
 - sigmoid, [11](#)
 - version, [11](#)
- happyml::Classifier, [17](#)
- happyml::DataSet, [17](#)
 - d, [19](#)
 - DataSet, [18](#)
 - N, [19](#)
 - read, [18](#)
 - write, [19](#)
 - X, [19](#)
 - y, [19](#)
- happyml::LinearModel, [20](#)
 - ~LinearModel, [21](#)
 - getWeights, [21](#)
 - LinearModel, [20](#), [21](#)
 - w, [21](#)
- happyml::LinearRegression, [22](#)
 - error, [23](#)
 - LinearRegression, [22](#), [23](#)
 - predict, [23](#)
 - train, [23](#)
- happyml::LogisticRegression, [24](#)
 - error, [25](#)
 - LogisticRegression, [24](#), [25](#)
 - predict, [25](#)
 - train, [25](#)
- happyml::NeuralNetwork, [26](#)
 - ~NeuralNetwork, [27](#)
 - getWeights, [27](#)
 - NeuralNetwork, [26](#), [27](#)
 - predict, [27](#)
 - train, [27](#)
- happyml::Perceptron, [28](#)
 - error, [29](#)
 - Perceptron, [28](#), [29](#)
 - predict, [29](#)
 - train, [29](#)
- happyml::Predictor, [30](#)
 - error, [30](#), [31](#)
 - predict, [31](#)
 - saveSampling, [31](#), [32](#)
- happyml::SVM, [34](#)
 - error, [35](#)
 - predict, [35](#)
 - SVM, [35](#)
 - train, [35](#)
- happyml::Serializable, [33](#)
 - load, [33](#)
 - read, [33](#)
 - save, [34](#)

- write, 34
- happyml::Transformer, 36
 - ~Transformer, 37
 - addAddition, 37
 - addPower, 37
 - addProduct, 37
 - apply, 38
 - normalize, 38
 - pca, 38
 - pcaMinVariance, 38
 - remove, 38
 - Transformer, 37
- happyml::colors, 11
 - BLUE, 12
 - CYAN, 12
 - GREEN, 12
 - PINK, 12
 - RESET, 12
 - RED, 12
- happyml::tools, 12
 - dictionary, 12
 - edgeTemplate, 14
 - linearModelTemplate, 14
 - modelToDot, 13, 14
 - substitute, 14
- include/happyml.h, 41
- include/happyml/dataset.h, 42
- include/happyml/happytools.h, 42
- include/happyml/linear_model.h, 43
- include/happyml/linear_regression/linear_regression.h, 43
- include/happyml/logistic_regression/logistic_regression.h, 43
- include/happyml/neural_network/neural_network.h, 44
- include/happyml/perceptron/perceptron.h, 44
- include/happyml/predictor.h, 44
- include/happyml/serializable.h, 45
- include/happyml/svm/svm.h, 45
- include/happyml/transformer.h, 46
- include/happyml/types.h, 46
- include/happyml/utils.h, 46
- Input
 - happyml, 10
- LinearModel
 - happyml::LinearModel, 20, 21
- linearModelTemplate
 - happyml::tools, 14
- LinearRegression
 - happyml::LinearRegression, 22, 23
- load
 - happyml::Serializable, 33
- LogisticRegression
 - happyml::LogisticRegression, 24, 25
- modelToDot
 - happyml::tools, 13, 14
- N
 - happyml::DataSet, 19
- NeuralNetwork
 - happyml::NeuralNetwork, 26, 27
- normalize
 - happyml::Transformer, 38
- PINK
 - happyml::colors, 12
- pca
 - happyml::Transformer, 38
- pcaMinVariance
 - happyml::Transformer, 38
- Perceptron
 - happyml::Perceptron, 28, 29
- predict
 - happyml::LinearRegression, 23
 - happyml::LogisticRegression, 25
 - happyml::NeuralNetwork, 27
 - happyml::Perceptron, 29
 - happyml::Predictor, 31
 - happyml::SVM, 35
- RESET
 - happyml::colors, 12
- RED
 - happyml::colors, 12
- read
 - happyml::DataSet, 18
 - happyml::Serializable, 33
- remove
 - happyml::Transformer, 38
- SVM
 - happyml::SVM, 35
- save
 - happyml::Serializable, 34
- saveSampling
 - happyml::Predictor, 31, 32
- sgn
 - happyml, 10
- sigmoid
 - happyml, 11
- src/dataset.cpp, 47
- src/happyml.cpp, 47
- src/happytools.cpp, 48
- src/linear_regression/linear_regression.cpp, 48
- src/logistic_regression/logistic_regression.cpp, 48
- src/neural_network/neural_network.cpp, 49
- src/perceptron/perceptron.cpp, 49
- src/predictor.cpp, 49
- src/serializable.cpp, 50
- src/svm/svm.cpp, 50
- src/transformer.cpp, 50
- src/utils.cpp, 50
- substitute
 - happyml::tools, 14
- train

- happyml::LinearRegression, [23](#)
- happyml::LogisticRegression, [25](#)
- happyml::NeuralNetwork, [27](#)
- happyml::Perceptron, [29](#)
- happyml::SVM, [35](#)
- Transformer
 - happyml::Transformer, [37](#)
- version
 - happyml, [11](#)
- w
 - happyml::LinearModel, [21](#)
- write
 - happyml::DataSet, [19](#)
 - happyml::Serializable, [34](#)
- X
 - happyml::DataSet, [19](#)
- y
 - happyml::DataSet, [19](#)