# HappyML

1.0

Generated by Doxygen 1.8.11

# Contents

# Chapter 1

# Namespace Index

## 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

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

# Chapter 3

# Class Index

## 3.1 Class List

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

# Chapter 4

# File Index

## 4.1 File List

Here is a list of all files with brief descriptions:

# Chapter 5

# Namespace Documentation

## 5.1 happyml 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* $w_0 \cdot x_0 + w_1 \cdot x_1 + \cdots + w_d \cdot x_d$.
- 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 happyml::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 happyml::version = HAPPY_ML_VERSION

String with the library version.

## 5.2 happyml::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 happyml::colors::BLUE = "\e[44m\e[37m"

#### 5.2.1.2 const string happyml::colors::CYAN = "\e[46m"

#### 5.2.1.3 const string happyml::colors::GREEN = "\e[32m"

#### 5.2.1.4 const string happyml::colors::PINK = "\e[45m\e[37m"

#### 5.2.1.5 const string happyml::colors::RED = "\e[41m\e[37m"

#### 5.2.1.6 const string happyml::colors::RESET = "\e[m"

## 5.3 happyml::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> happyml::tools::dictionary

Dictionary containing variables names as keys and his values.

**See also**

substitute

### 5.3.2 Function Documentation

#### 5.3.2.1 void happyml::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**

| | |
|---|---|
| *lm* | Linear model to represent. |
| *filename* | Name of the file where the DOT info will be saved. |
| *latex* | True to use latex in the labels. False by default. |

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

**See also**

modelToDot(const LinearModel&, const string&, bool)

#### 5.3.2.3 string happyml::tools::substitute ( const string & *in*, const dictionary & *dic* )

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

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

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

**Parameters**

| | |
|---|---|
| *in* | Input string with vars names to substitute. |
| *dic* | 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 happyml::tools::edgeTemplate

**Initial value:**

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

#### 5.3.3.2 string happyml::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

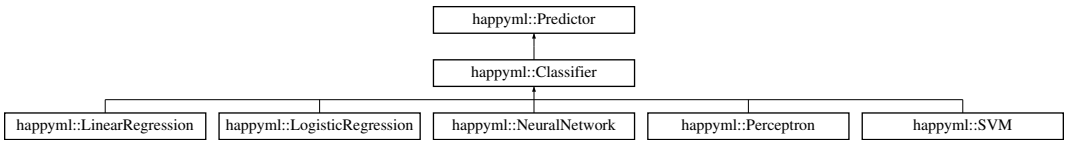
# Class Documentation

## 6.1 happyml::Classifier Class Reference

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

`#include <predictor.h>`

Inheritance diagram for happyml::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:

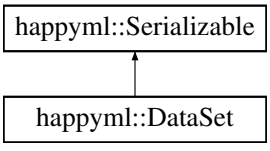
- include/happyml/predictor.h

## 6.2 happyml::DataSet Class Reference

Generic collection of inputs and outputs.

`#include <dataset.h>`

Inheritance diagram for happyml::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 happyml::DataSet::DataSet ( unsigned *dim* = 0, unsigned *size* = 0 ) [inline]

Creates a generic collection of inputs and outputs.

**Parameters**

| | |
|---|---|
| *dim* | Dimension $d$ of the inputs. |
| *size* | Number of inputs ( $N$) in the dataset. |

## 6.2.3 Member Function Documentation

### 6.2.3.1 void happyml::DataSet::read ( istream & *stream* ) [virtual]

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

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

$x_{11}, x_{12}, \cdots, x_{1d}, y_1$
$\vdots, \vdots, \cdots, \vdots, \vdots$
$x_{(N-1)1}, x_{(N-1)2}, \cdots, x_{(N-1)d}, y_{(N-1)}$

Implements happyml::Serializable.

#### 6.2.3.2 void happyml::DataSet::write ( ostream & *stream* ) const [virtual]

Write to an output stream the next data:

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

Implements happyml::Serializable.

### 6.2.4 Member Data Documentation

#### 6.2.4.1 unsigned happyml::DataSet::d

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

All the points has the same dimension.

#### 6.2.4.2 unsigned happyml::DataSet::N

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

We denote it with $N$.

#### 6.2.4.3 mat happyml::DataSet::X

Matrix with all the inputs.

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

#### 6.2.4.4 mat happyml::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:

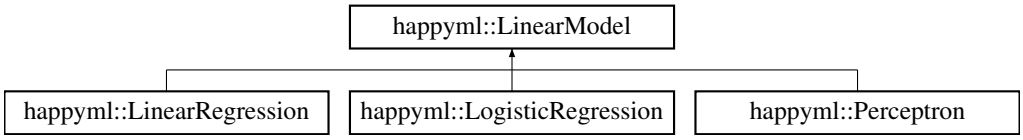
- include/happyml/dataset.h
- src/dataset.cpp

## 6.3 happyml::LinearModel Class Reference

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

`#include <linear_model.h>`

Inheritance diagram for happyml::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 $w_0 \cdot x_0 + w_1 \cdot x_1 + \cdots + w_d \cdot x_d$.

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

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

### 6.3.2 Constructor & Destructor Documentation

#### 6.3.2.1 happyml::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 happyml::LinearModel::LinearModel ( const vec & *weights* ) [inline]

Creates a linear model with the indicated weights.

**Parameters**

| | |
|---|---|
| *weights* | Weight vector with $d + 1$ dimensions. |

#### 6.3.2.3 happyml::LinearModel::LinearModel ( const LinearModel & *lm* ) [inline]

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

**Parameters**

| | |
|---|---|
| *lm* | Linear model from which weights are copied. |

#### 6.3.2.4 happyml::LinearModel::~LinearModel ( ) [inline]

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

### 6.3.3 Member Function Documentation

#### 6.3.3.1 vec happyml::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 happyml::LinearModel::w [protected]

Vector of weights.

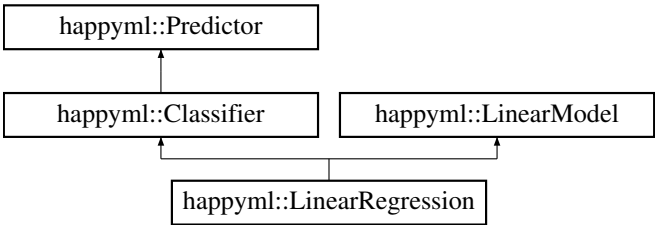
The documentation for this class was generated from the following file:

- include/happyml/linear_model.h

## 6.4 happyml::LinearRegression Class Reference

`#include <linear_regression.h>`

Inheritance diagram for happyml::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 happyml::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 happyml::LinearRegression::LinearRegression ( const vec & *weights* ) `[inline]`

Creates a linear regression algorithm with the indicated weights.

**Parameters**

| | |
|---|---|
| *weights* | Weight vector with $d + 1$ size. |

#### 6.4.1.3 happyml::LinearRegression::LinearRegression ( const LinearModel & *lm* ) [inline]

### 6.4.2 Member Function Documentation

#### 6.4.2.1 double happyml::LinearRegression::error ( const DataSet & *data* ) const [virtual]

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

**Parameters**

| | |
|---|---|
| *data* | Dataset with the correct output. |

**Returns**

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

Reimplemented from happyml::Predictor.

#### 6.4.2.2 double happyml::LinearRegression::predict ( const Input & *x* ) const [virtual]

Predict the output of an input vector.

**Returns**

Estimated real output.

Implements happyml::Predictor.

#### 6.4.2.3 double happyml::LinearRegression::train ( const DataSet & *data* )

Train the linear regression.

**Parameters**

| | |
|---|---|
| *data* | Training set. |

**Returns**

Returns the final error.

#### 6.4.2.4 double happyml::LinearRegression::train ( const DataSet & *data*, double *lambda* )

Train the linear regression.

**Parameters**

| | |
|---|---|
| *data* | Training set. |
| *lambda* | Regularization paramiter $\lambda$. |

**Returns**

Returns the final error.

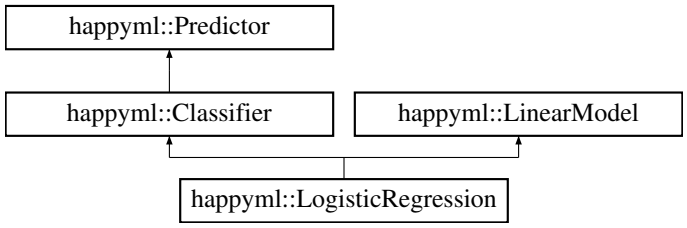
The documentation for this class was generated from the following files:

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

## 6.5 happyml::LogisticRegression Class Reference

`#include <logistic_regression.h>`

Inheritance diagram for happyml::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 happyml::LogisticRegression::LogisticRegression ( unsigned *d* = `0` ) [inline]

Creates a logistic regression algorithm with the indicated input size.

**Parameters**

| | |
|---|---|
| *d* | Dimension $d$ of the input feature vectors. |

#### 6.5.1.2 happyml::LogisticRegression::LogisticRegression ( const vec & *weights* ) [inline]

Creates a logistic regression algorithm with the indicated weights.

**Parameters**

| | |
|---|---|
| *weights* | Weight vector with $d + 1$ size. |

#### 6.5.1.3 happyml::LogisticRegression::LogisticRegression ( const LinearModel & *lm* ) [inline]

### 6.5.2 Member Function Documentation

#### 6.5.2.1 double happyml::LogisticRegression::error ( const DataSet & *data* ) const [virtual]

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

**Parameters**

| | |
|---|---|
| *data* | Dataset with the correct output. |

**Returns**

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

Reimplemented from happyml::Predictor.

#### 6.5.2.2 double happyml::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 happyml::Predictor.

#### 6.5.2.3 double happyml::LogisticRegression::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.

Data must contain the $x_0 = 1$ property.

**Parameters**

| | |
|---|---|
| *data* | Training set. |
| *iter* | Maximun number of iterations ( $1000$ by default). |
| *learning_rate* | Learning rate ( $0.1$ by default.). |

**Returns**

Returns the error of the best weights found.

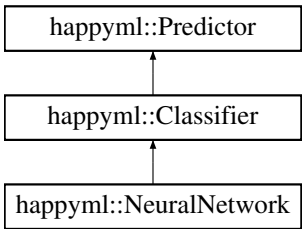
The documentation for this class was generated from the following files:

- include/happyml/logistic_regression/logistic_regression.h
- src/logistic_regression/logistic_regression.cpp

## 6.6 happyml::NeuralNetwork Class Reference

`#include <neural_network.h>`

Inheritance diagram for happyml::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 happyml::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**

| | |
|---|---|
| *layers* | Number of layers, number of neurons on the input layer, number of neurons on the second layer... |

#### 6.6.1.2 happyml::NeuralNetwork::NeuralNetwork ( const NeuralNetwork & *nn* )

Creates a copy of a neural network.

#### 6.6.1.3 happyml::NeuralNetwork::~NeuralNetwork ( )

### 6.6.2 Member Function Documentation

#### 6.6.2.1 const vector<mat> happyml::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 happyml::NeuralNetwork::predict ( const Input & *x* ) const [virtual]

Uses the forward propagation algorithm to predict an output.

**Parameters**

| | |
|---|---|
| *x* | Input vector. |

**Returns**

Predicted value

Implements happyml::Predictor.

#### 6.6.2.3 double happyml::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**

| | |
|---|---|
| *dataset* | Training set $\mathcal{D}$. |
| *learningRate* | Learning rate $\eta$. |
| *iter* | Number of iterations. |
| *lambda* | Regularization paramiter $\lambda$. |

**Returns**

Returns the error after the training.

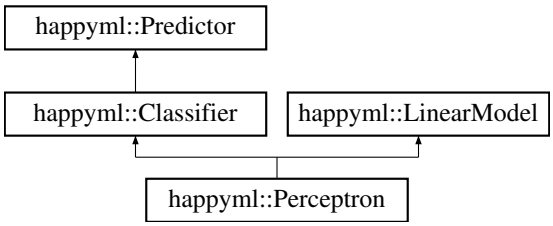
The documentation for this class was generated from the following files:

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

## 6.7 happyml::Perceptron Class Reference

`#include <perceptron.h>`

Inheritance diagram for happyml::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 happyml::Perceptron::Perceptron ( unsigned *d* = `0` ) `[inline]`

Creates a perceptron with the indicated input size.

**Parameters**

| | |
|---|---|
| *d* | Dimension $d$ of the input feature vectors. |

#### 6.7.1.2 happyml::Perceptron::Perceptron ( const vec & *weights* ) [inline]

Creates a perceptron with the indicated weights.

**Parameters**

| | |
|---|---|
| *weights* | Weight vector with $d + 1$ dimensions. |

#### 6.7.1.3 happyml::Perceptron::Perceptron ( const **LinearModel** & *lm* ) [inline]

### 6.7.2 Member Function Documentation

#### 6.7.2.1 double happyml::Perceptron::error ( const **DataSet** & *data* ) const [virtual]

Compute the error of the perceptron on the given dataset.

**Parameters**

| | |
|---|---|
| *data* | Dataset with the correct output. |

**Returns**

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

Reimplemented from happyml::Predictor.

#### 6.7.2.2 double happyml::Perceptron::predict ( const **Input** & *x* ) const [virtual]

Classifies an input vector.

**Returns**

$-1$ or $+1$.

Implements happyml::Predictor.

#### 6.7.2.3 double happyml::Perceptron::train ( const **DataSet** & *data*, unsigned *iter* )

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

Data must contain the $x_0 = 1$ property.

**Parameters**

| | |
|---|---|
| *data* | Training set. |
| *iter* | Maximun number of iterations. |

**Returns**

Returns the error of the best perceptron found.

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

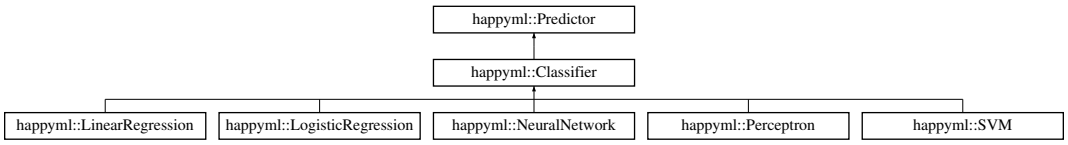
- include/happyml/perceptron/perceptron.h
- src/perceptron/perceptron.cpp

## 6.8 happyml::Predictor Class Reference

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

`#include <predictor.h>`

Inheritance diagram for happyml::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 predicter 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 predicter 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 happyml::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 comparate to y.

#### 6.8.2.2 double happyml::Predictor::error ( const DataSet & *dataset* ) const [virtual]

Compute the error of the predictor on the given dataset.

**Parameters**

| | |
|---|---|
| *data* | Dataset with the correct output. |

**Returns**

Error of predicting the output of the given dataset.

Reimplemented in happyml::LinearRegression, happyml::SVM, happyml::LogisticRegression, and happyml::Perceptron.

#### 6.8.2.3 virtual double happyml::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 happyml::NeuralNetwork, happyml::LinearRegression, happyml::SVM, happyml::LogisticRegression, and happyml::Perceptron.

#### 6.8.2.4 void happyml::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 predicter on the rectangular area defined between $(minx_1, minx_2)$ and $(maxx_1, maxx_2)$ points.

The CSV output file has the following structure:

$minx_1, maxx_1, samples_1$
$minx_2, maxx_2, samples_2$
$h_{1\ 1}, h_{1\ 2}, \cdots, h_{1\ samples_1}$
$\vdots, \vdots, \cdots, \vdots$
$h_{samples\ 1}, h_{samples\ 2}, \cdots, h_{samples_2\ samples_1}$

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

**Parameters**

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

#### 6.8.2.5 void happyml::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 predicter in the segment $[minx, maxx]$.

The CSV output file has the following structure:

$minx, maxx, samples$
$h_1, h_2, \cdots, h_{samples}$

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

**Parameters**

| | |
|---|---|
| *filename* | Output filename. |
| *minx* | Start value of the first feature. |
| *maxx* | End value of the first feature. |
| *samples* | Number of samples of the first feature. |
| *t* | 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:

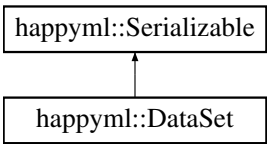
- include/happyml/predictor.h
- src/predictor.cpp

## 6.9 happyml::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 happyml::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 happyml::Serializable::load ( const string & *filename* ) [virtual]

Fills an object using the data in the given file.

**Parameters**

| | |
|---|---|
| *filename* | Name of the file to load. |

**See also**

save(const string&)

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

Read the object from an input stream.

**Parameters**

| | |
|---|---|
| *stream* | 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**

| | |
|---|---|
| *filename* | 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**

| | |
|---|---|
| *stream* | Output stream. |

Implemented in happyml::DataSet.

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

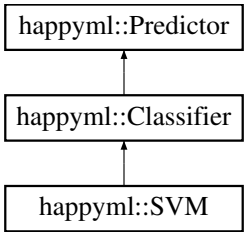
- include/happyml/serializable.h
- src/serializable.cpp

## 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 happyml::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 happyml::SVM::error ( const DataSet & *data* ) const [virtual]

Compute the error of the SVM on the given dataset.

**Parameters**

| | |
|---|---|
| *data* | Dataset with the correct output. |

**Returns**

Error of classify the given dataset.

Reimplemented from happyml::Predictor.

### 6.10.3.2 double happyml::SVM::predict ( const Input & *x* ) const [virtual]

Classifies an input vector.

**Returns**

$-1$ or $+1$.

Implements happyml::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**

| | |
|---|---|
| *data* | Training set. |
| *C* | SVM regularization parameter. |
| *iter* | Maximun number of iterations that the SMO algorithm does all over the dataset. |
| *tolerance* | 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 substract) 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 extranctor.*
- 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 happyml::Transformer::Transformer ( ) [inline]

#### 6.11.2.2 happyml::Transformer::~Transformer ( ) [inline]

### 6.11.3 Member Function Documentation

#### 6.11.3.1 void happyml::Transformer::addAddition ( unsigned *feature,* double *n,* bool *create_new* = false )

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

Doesn't add a new feature.

**Parameters**

| | |
|---|---|
| *feature* | $n$ will be added to this feature number. |
| *n* | Number to be added. |
| *create_new* | Create a new feature or not. |

#### 6.11.3.2 void happyml::Transformer::addPower ( unsigned *feature,* double *power,* bool *create_new* = true )

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

**Parameters**

| | |
|---|---|
| *feature* | Feature to raise. |
| *power* | Exponent. |
| *create_new* | Create a new feature or not. |

#### 6.11.3.3 void happyml::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**

| | |
|---|---|
| *feature* | $n$ will be added to this feature number. |
| *n* | Number to be added. |
| *create_new* | Create a new feature or not. |

#### 6.11.3.4 void happyml::Transformer::apply ( DataSet & *dataset* )

Applies all the transformations in the given dataset.

**Parameters**

| | |
|---|---|
| *dataset* | Dataset to transform. |

#### 6.11.3.5 Input happyml::Transformer::apply ( const Input & *input* ) const

Applies all the transformations in the given input.

**Parameters**

| | |
|---|---|
| *input* | Input to transform. |

**Returns**

A transformated version of the input.

#### 6.11.3.6 void happyml::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 happyml::Transformer::pca ( int *k* )

PCA extranctor.

**Parameters**

| | |
|---|---|
| *k* | Number of dimensions to choose. |

#### 6.11.3.8 void happyml::Transformer::pcaMinVariance ( double *pcaVar* )

PCA extractor.

**Parameters**

| | |
|---|---|
| *pcaVar* | Choose $k$ dimensions ultils total variance is at least `pcaVar`. |

#### 6.11.3.9 void happyml::Transformer::remove ( unsigned *feature* )

Deletes a feature.

**Parameters**

| | |
|---|---|
| *feature* | Feature that will be removed. |

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

- include/happyml/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/happyml/linear_model.h File Reference

```
#include <armadillo>
```

### Classes

- class happyml::LinearModel

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

### Namespaces

- happyml

  *HappyML library namespace.*

## 7.5 include/happyml/linear_regression/linear_regression.h File Reference

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

### Classes

- class happyml::LinearRegression

### Namespaces

- happyml

  *HappyML library namespace.*

## 7.6 include/happyml/logistic_regression/logistic_regression.h File Reference

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

### Classes

- class happyml::LogisticRegression

### Namespaces

- happyml

  *HappyML library namespace.*

## 7.7 include/happyml/neural_network/neural_network.h File Reference

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

### Classes

- class happyml::NeuralNetwork

### Namespaces

- happyml

  *HappyML library namespace.*

## 7.8 include/happyml/perceptron/perceptron.h File Reference

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

### Classes

- class happyml::Perceptron

### Namespaces

- happyml

  *HappyML library namespace.*

## 7.9 include/happyml/predictor.h File Reference

```
#include "happyml/types.h"
#include "happyml/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/happyml/transformer.h File Reference

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

### Classes

- class happyml::Transformer

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

### Namespaces

- happyml

  *HappyML library namespace.*

## 7.13 include/happyml/types.h File Reference

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

### Namespaces

- happyml

  *HappyML library namespace.*

### Typedefs

- typedef vec happyml::Input

  *Vector of inputs features.*

## 7.14 include/happyml/utils.h File Reference

```
#include <string>
```

### Namespaces

- happyml

  *HappyML library namespace.*
- happyml::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 <cstdarg>
```

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

### Namespaces

- happyml

  *HappyML library namespace.*

## 7.24 src/svm/svm.cpp File Reference

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

### Namespaces

- happyml

  *HappyML library namespace.*

## 7.25 src/transformer.cpp File Reference

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

### Namespaces

- happyml

  *HappyML library namespace.*

## 7.26 src/utils.cpp File Reference

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

### Namespaces

- happyml

  *HappyML library namespace.*

### Functions

- int happyml::sgn (double x)

  *Default sign function.*
- double happyml::sigmoid (double x)

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

# Index