

FNN++

1

Generated by Doxygen 1.8.9.1

Sun May 10 2015 22:16:51



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>Namespace Documentation</b>	<b>7</b>
4.1	fnn Namespace Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
<b>5</b>	<b>Class Documentation</b>	<b>9</b>
5.1	fnn::DataPoint Struct Reference . . . . .	9
5.2	fnn::DataSet Class Reference . . . . .	9
5.2.1	Constructor & Destructor Documentation . . . . .	9
5.2.1.1	DataSet . . . . .	9
5.2.2	Member Function Documentation . . . . .	10
5.2.2.1	calcError . . . . .	10
5.2.2.2	calculateErrors . . . . .	10
5.2.2.3	DataSet::operator[] . . . . .	10
5.2.2.4	Load . . . . .	10
5.2.2.5	Shuffle . . . . .	11
5.2.2.6	size . . . . .	11
5.3	fnn::Experiment Class Reference . . . . .	11
5.3.1	Constructor & Destructor Documentation . . . . .	11
5.3.1.1	Experiment . . . . .	11
5.3.2	Member Function Documentation . . . . .	12
5.3.2.1	Run . . . . .	12
5.3.2.2	RunAsThread . . . . .	12
5.3.2.3	toggleThread . . . . .	12
5.4	fnn::FNNTrainer Class Reference . . . . .	12

5.4.1	Detailed Description	12
5.4.2	Constructor & Destructor Documentation	13
5.4.2.1	FNNTrainer	13
5.4.3	Member Function Documentation	13
5.4.3.1	Bound	13
5.4.3.2	Train	13
5.5	fnn::Log Class Reference	14
5.5.1	Detailed Description	14
5.5.2	Constructor & Destructor Documentation	14
5.5.2.1	Log	14
5.5.2.2	Log	14
5.5.3	Member Function Documentation	15
5.5.3.1	GetContent	15
5.5.3.2	GetName	15
5.5.3.3	IsVerbose	15
5.5.3.4	Push	15
5.6	fnn::Loggable Class Reference	16
5.6.1	Detailed Description	16
5.6.2	Constructor & Destructor Documentation	16
5.6.2.1	Loggable	16
5.6.3	Member Function Documentation	17
5.6.3.1	AddLog	17
5.6.3.2	GetLogs	17
5.6.3.3	GetName	17
5.6.3.4	Initialize	17
5.6.3.5	IsVerbose	19
5.6.3.6	Log	19
5.6.3.7	SetVerbose	19
5.7	fnn::LogManager Class Reference	20
5.7.1	Detailed Description	20
5.7.2	Constructor & Destructor Documentation	20
5.7.2.1	LogManager	20
5.7.3	Member Function Documentation	20
5.7.3.1	Print	20
5.7.3.2	Register	21
5.7.3.3	Save	21
5.8	fnn::Math Class Reference	21
5.8.1	Detailed Description	22
5.8.2	Member Function Documentation	22
5.8.2.1	DataSort	22

5.8.2.2	Factorial	23
5.8.2.3	GaussianReal	23
5.8.2.4	GaussJordan	23
5.8.2.5	LagrangeInterpolation	24
5.8.2.6	LERP	24
5.8.2.7	Mean	24
5.8.2.8	NIntegrate	24
5.8.2.9	NIntegrate	25
5.8.2.10	PolyMult	25
5.8.2.11	SSpline	26
5.8.2.12	StdDev	26
5.8.2.13	UniformReal	26
5.9	fnn::Network Class Reference	27
5.9.1	Detailed Description	27
5.9.2	Constructor & Destructor Documentation	28
5.9.2.1	Network	28
5.9.3	Member Function Documentation	28
5.9.3.1	AddLayer	28
5.9.3.2	BackPropagate	28
5.9.3.3	FeedForward	28
5.9.3.4	NudgeWeights	29
5.9.3.5	SetActivation	29
5.9.3.6	Train	29
5.9.4	Member Data Documentation	29
5.9.4.1	Activator	29
5.10	fnn::Sigmoid Class Reference	30
5.10.1	Constructor & Destructor Documentation	30
5.10.1.1	Sigmoid	30
5.10.1.2	Sigmoid	30
5.10.2	Member Function Documentation	31
5.10.2.1	Linear	31
5.10.2.2	Logistic	31
5.10.2.3	operator()	31
5.10.2.4	prime	31
5.10.2.5	Tanh	32
5.11	fnn::WeightSurface Class Reference	32
5.11.1	Constructor & Destructor Documentation	32
5.11.1.1	WeightSurface	32
5.11.2	Member Function Documentation	33
5.11.2.1	GetCoefficient	33

5.11.2.2	GetSizeX . . . . .	33
5.11.2.3	GetSizeY . . . . .	33
5.11.2.4	Nudge . . . . .	33
5.11.2.5	operator() . . . . .	34
<b>Index</b>		<b>35</b>

# Chapter 1

## Namespace Index

### 1.1 Namespace List

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

<a href="#">fnn</a>	Different sigmoid activation functions. . . . .	<a href="#">7</a>
---------------------	---	-------------------





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

fnn::DataPoint . . . . .	9
fnn::DataSet . . . . .	9
fnn::Experiment . . . . .	11
fnn::FNNTTrainer . . . . .	12
fnn::Log . . . . .	14
fnn::Loggable . . . . .	16
fnn::Network . . . . .	27
fnn::LogManager . . . . .	20
fnn::Math . . . . .	21
fnn::Sigmoid . . . . .	30
fnn::WeightSurface . . . . .	32



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">fnn::DataPoint</a>	9
<a href="#">fnn::DataSet</a>	9
<a href="#">fnn::Experiment</a>	11
<a href="#">fnn::FNNTrainer</a>	
The Trainer for the <a href="#">Network</a>	12
<a href="#">fnn::Log</a>	
A log.	14
<a href="#">fnn::Loggable</a>	
A loggable.	16
<a href="#">fnn::LogManager</a>	
Manager for logs.	20
<a href="#">fnn::Math</a>	
The main mathematics helper class for FNNLIB	21
<a href="#">fnn::Network</a>	
The main class of operation on the functional neural networks.	27
<a href="#">fnn::Sigmoid</a>	30
<a href="#">fnn::WeightSurface</a>	32



## Chapter 4

# Namespace Documentation

### 4.1 fnn Namespace Reference

Different sigmoid activation functions.

#### Classes

- struct [DataPoint](#)
- class [DataSet](#)
- class [Experiment](#)
- class [FNNTrainer](#)  
*The Trainer for the [Network](#)*
- class [Log](#)  
*A log.*
- class [Loggable](#)  
*A loggable.*
- class [LogManager](#)  
*Manager for logs.*
- class [Math](#)  
*The main mathematics helper class for FNNLIB*
- class [Network](#)  
*The main class of operation on the functional neural networks.*
- class [Sigmoid](#)
- class [WeightSurface](#)

#### 4.1.1 Detailed Description

Different sigmoid activation functions.

=====



## Chapter 5

# Class Documentation

### 5.1 fnn::DataPoint Struct Reference

#### Public Attributes

- `std::function< double(double)>` **input**
- `std::function< double(double)>` **desired**

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

- FNN++/FNNLib/FNNDataPoint.h

### 5.2 fnn::DataSet Class Reference

#### Public Member Functions

- [DataSet](#) ()  
*Constructor that initializes the class*
- virtual void [Load](#) ()=0  
*Loads the [DataSet](#)*
- void [Shuffle](#) ()  
*Shuffles this [DataSet](#)*
- `std::vector< double(double)>` [calculateErrors](#) ([Network](#) &nn, double step=-1)  
*Calculates the errors.*
- double [calcError](#) ([Network](#) &nn, double step=-1)  
*Calculates the error.*
- int [size](#) ()  
*Gets the size.*
- [DataPoint](#) & [DataSet::operator\[\]](#) (int index)  
*Array indexer operator.*

#### 5.2.1 Constructor & Destructor Documentation

##### 5.2.1.1 fnn::DataSet::DataSet ( void )

Constructor that initializes the class

```
=====
=====
```

Phillip Kuznetsov, 5/6/2015.

Constructor that initializes the class.

Phillip Kuznetsov, 5/6/2015.

## 5.2.2 Member Function Documentation

### 5.2.2.1 `double fnn::DataSet::calcError ( Network & nn, double step = -1 )`

Calculates the error.

Phillip Kuznetsov, 5/6/2015.

#### Parameters

<i>nn</i>	The nn.
<i>step</i>	Amount to increment by.

#### Returns

The calculated error.

### 5.2.2.2 `std::vector<double(double)> fnn::DataSet::calculateErrors ( Network & nn, double step = -1 )`

Calculates the errors.

Phillip Kuznetsov, 5/6/2015.

#### Parameters

<i>n</i>	The <a href="#">Network</a> to process.
<i>step</i>	Amount to increment by.

#### Returns

The calculated errors.

### 5.2.2.3 `DataPoint& fnn::DataSet::DataSet::operator[] ( int index )`

Array indexer operator.

Phillip Kuznetsov, 5/6/2015.

#### Parameters

<i>index</i>	Zero-based index of the.
--------------	--------------------------

#### Returns

The indexed value from DataPoints.

### 5.2.2.4 `virtual void fnn::DataSet::Load ( ) [pure virtual]`

Loads the [DataSet](#)

Phillip Kuznetsov, 5/6/2015.



## 5.2.2.5 void fnn::DataSet::Shuffle ( )

Shuffles this [DataSet](#)

Phillip Kuznetsov, 5/6/2015.

## 5.2.2.6 int fnn::DataSet::size ( void )

Gets the size.

Phillip Kuznetsov, 5/6/2015.

## Returns

An int size of the [DataSet](#)

Phillip Kuznetsov, 5/6/2015.

## Returns

An int size of the [DataSet](#).

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

- FNN++/FNNLib/FNNDataSet.h
- FNN++/FNNLib/FNNDataSet.cpp

## 5.3 fnn::Experiment Class Reference

### Public Member Functions

- [Experiment](#) ([DataSet](#) &trainingSet, [DataSet](#) &testingSet)  
*Constructor.*
- virtual void [Run](#) ()=0  
*Runs this object.*
- bool [RunAsThread](#) ()  
*Executes as thread operation.*
- void [toggleThread](#) ()  
*Toggles whether the thread is running or not*

### 5.3.1 Constructor & Destructor Documentation

#### 5.3.1.1 fnn::Experiment::Experiment ( [DataSet](#) & trainingSet, [DataSet](#) & testingSet )

Constructor.

Phillip Kuznetsov, 5/8/2015.

## Parameters

<i>trainingSet</i>	Set the training belongs to.
<i>testingSet</i>	Set the testing belongs to.

### 5.3.2 Member Function Documentation

#### 5.3.2.1 virtual void fnn::Experiment::Run ( ) [pure virtual]

Runs this object.

Phillip Kuznetsov, 5/8/2015.

#### 5.3.2.2 bool fnn::Experiment::RunAsThread ( )

Executes as thread operation.

Phillip Kuznetsov, 5/8/2015.

#### Returns

true if it succeeds, false if it fails.

#### 5.3.2.3 void fnn::Experiment::toggleThread ( )

Toggles whether the thread is running or not

Toggles whether the thread is running or not.

Phillip Kuznetsov, 5/8/2015.

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

- FNN++/FNNLib/FNNExperiment.h
- FNN++/FNNLib/FNNExperiment.cpp

## 5.4 fnn::FNNTrainer Class Reference

The Trainer for the [Network](#)

```
#include <FNNTrainer.h>
```

### Public Member Functions

- [FNNTrainer](#) ([Network](#) &network, [fnn::DataSet](#) &trainingSet, [fnn::DataSet](#) &testingSet)  
*[FNNTrainer](#) that takes a 2D vector for the trainingSet and testingSet.*
- int [Train](#) (int epochs, double minError, std::vector< double > &learningParameters, bool nudging=false)  
*Trains the network according to the parameters*
- double [Bound](#) (double val, double min, double max)  
*Bounds a double to a range of min and max*

#### 5.4.1 Detailed Description

The Trainer for the [Network](#)

=====

Phillip Kuznetsov, 5/6/2015.

## 5.4.2 Constructor & Destructor Documentation

5.4.2.1 `fnn::FNNTrainer::FNNTrainer ( Network & network, fnn::DataSet & trainingSet, fnn::DataSet & testingSet )`

[FNNTrainer](#) that takes a 2D vector for the *trainingSet* and *testingSet*.

=====

Phillip Kuznetsov, 5/6/2015.

Parameters

<i>network</i>	The <a href="#">Network</a> being operated on.
<i>trainingSet</i>	The training dataset.
<i>testingSet</i>	The traing dataset.

## 5.4.3 Member Function Documentation

5.4.3.1 `double fnn::FNNTrainer::Bound ( double val, double min, double max )`

Bounds a double to a range of min and max

Bounds a double to a range of min and max.

Phillip Kuznetsov, 5/6/2015.

Parameters

<i>val</i>	The value.
<i>min</i>	The minimum.
<i>max</i>	The maximum.

Returns

A double.

5.4.3.2 `int fnn::FNNTrainer::Train ( int epochs, double minError, std::vector< double > & learningParameters, bool nudging = false )`

Trains the network according to the parameters

=====

Phillip Kuznetsov, 5/6/2015.

Parameters

<i>epochs</i>	The epochs.
<i>minError</i>	The minimum error.
<i>learningParameters</i>	Options for controlling the learning.
<i>nudging</i>	Whether the trainer uses nudging

An int.

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

- FNN++/FNNLib/FNNTrainer.h
- FNN++/FNNLib/FNNTrainer.cpp

## 5.5 fnn::Log Class Reference

A log.

```
#include <FNNLog.h>
```

### Public Member Functions

- [Log \(\)](#)  
*Default constructor.*
- [Log](#) (string name, bool verbose)  
*Constructs the log with a name and a vocality.*
- void [Push](#) (string message)  
*Pushes an object onto this log.*
- list< string > \* [GetContent](#) ()  
*Gets the content of the log.*
- string [GetName](#) ()  
*Gets the name.*
- bool [IsVerbose](#) ()  
*Query if this object is verbose.*

### 5.5.1 Detailed Description

A log.

=====

William, 4/29/2015.

### 5.5.2 Constructor & Destructor Documentation

#### 5.5.2.1 fnn::Log::Log ( void )

Default constructor.

=====

William, 4/29/2015.

#### 5.5.2.2 fnn::Log::Log ( string name, bool verbose )

Constructs the log with a name and a vocality.

=====

=====

William, 4/29/2015.

#### Parameters

<i>name</i>	The name.
-------------	-----------

true to vocal.

Constructs the log with a name and a vocality.

William, 4/29/2015.

Parameters

<i>name</i>	The name.
-------------	-----------

true to vocal.

Constructs the log.

William, 4/29/2015.

Parameters

<i>name</i>	The name.
-------------	-----------

true to verbose.

### 5.5.3 Member Function Documentation

#### 5.5.3.1 list< string > \* fnn::Log::GetContent ( void )

Gets the content of the log.

=====

William, 4/29/2015.

null if it fails, else the content.

#### 5.5.3.2 string fnn::Log::GetName ( void )

Gets the name.

=====

William, 4/29/2015.

The name.

#### 5.5.3.3 bool fnn::Log::IsVerbose ( void )

Query if this object is verbose.

=====

William, 4/29/2015.

true if verbose, false if not.

#### 5.5.3.4 void fnn::Log::Push ( string message )

Pushes an object onto this log.

=====

William, 4/29/2015.

The message to push.

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

- FNN++/FNNLib/FNNLog.h
- FNN++/FNNLib/FNNLog.cpp

## 5.6 fnn::Loggable Class Reference

A loggable.

```
#include <FNNLoggable.h>
```

Inheritance diagram for fnn::Loggable:

### Public Member Functions

- [Loggable](#) ()  
*Default constructor.*
- void [Initialize](#) ([LogManager](#) \*lm, string name, bool verbose)  
*Constructs the logger.*
- void [Log](#) (string log, string message, bool verbose=false)  
*Logs a message to a specific log.*
- void [AddLog](#) (string name, bool verbose)  
*Adds a log.*
- void [SetVerbose](#) (bool verbose)  
*Sets a verbose.*
- bool [IsVerbose](#) ()  
*Query if this object is verbose.*
- std::vector< [fnn::Log](#) \* > [GetLogs](#) ()  
*Gets the logs.*
- string [GetName](#) ()  
*Gets the name.*

### 5.6.1 Detailed Description

A loggable.

=====

William, 4/29/2015.

### 5.6.2 Constructor & Destructor Documentation

#### 5.6.2.1 fnn::Loggable::Loggable ( void )

Default constructor.

=====

William, 5/2/2015.

### 5.6.3 Member Function Documentation

#### 5.6.3.1 void fnn::Loggable::AddLog ( string *name*, bool *verbose* )

Adds a log.

=====

William, 4/29/2015.

The name.

#### 5.6.3.2 std::vector< fnn::Log \* > fnn::Loggable::GetLogs ( void )

Gets the logs.

=====

William, 4/29/2015.

null if it fails, else the logs.

#### 5.6.3.3 string fnn::Loggable::GetName ( void )

Gets the name.

=====

William, 4/29/2015.

The name.

#### 5.6.3.4 void fnn::Loggable::Initialize ( LogManager \* *lm*, string *name*, bool *verbose* )

Constructs the logger.

=====

William, 4/29/2015.

Parameters

<i>lm</i>	[in,out] If non-null, the lm.
<i>name</i>	The name.

true to verbose.

Constructs the logger.

William, 4/29/2015.

The name.

Initializes this object.

William, 5/2/2015.



## Parameters

<i>lm</i>	[in,out] If non-null, the lm.
<i>name</i>	The name.

true to verbose.

## 5.6.3.5 bool fnn::Loggable::IsVerbose ( void )

Query if this object is verbose.

=====

William, 4/29/2015.

true if verbose, false if not.

## 5.6.3.6 void fnn::Loggable::Log ( string log, string message, bool verbose = false )

Logs a message to a specific log.

=====

William, 4/29/2015.

## Parameters

<i>log</i>	The log.
------------	----------

The message.

=====

William, 4/29/2015.

## Parameters

<i>log</i>	The log.
<i>message</i>	The message.

Whether or not (upon creation of a new log the log is verbose)

## 5.6.3.7 void fnn::Loggable::SetVerbose ( bool verbose )

Sets a verbose.

=====

William, 4/29/2015.

true to verbose.

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

- FNN++/FNNLib/FNNLoggable.h
- FNN++/FNNLib/FNNLoggable.cpp

## 5.7 fnn::LogManager Class Reference

Manager for logs.

```
#include <FNNLogManager.h>
```

### Public Member Functions

- [LogManager](#) ()  
*Default constructor.*
- void [Register](#) ([Loggable](#) \*logger, string loggerName, bool verbose)  
*Registers a log with the [LogManager](#).*
- void [Save](#) (string directory)  
*Saves the set of all loggers under a directory and a sub directory. Consider is main logger property.*
- void [Print](#) ([Loggable](#) \*logger, string log, string message)  
*TODO: Consider adding loading functionality.*

### 5.7.1 Detailed Description

Manager for logs.

=====

William, 4/29/2015.

### 5.7.2 Constructor & Destructor Documentation

#### 5.7.2.1 fnn::LogManager::LogManager ( void )

Default constructor.

=====

=====

William, 4/29/2015.

Default constructor.

William, 4/29/2015.

### 5.7.3 Member Function Documentation

#### 5.7.3.1 void fnn::LogManager::Print ( [Loggable](#) \* logger, string log, string message )

TODO: Consider adding loading functionality.

Prints a message to the verbose log.

=====

William, 5/3/2015.

## Parameters

<i>logger</i>	[in,out] If non-null, the logger.
<i>log</i>	The log.

The message.

5.7.3.2 void fnn::LogManager::Register ( Loggable \* *logger*, string *loggerName*, bool *verbose* )

Registers a log with the [LogManager](#).

=====

William, 4/29/2015.

## Parameters

<i>logger</i>	[in,out] The logger.
<i>loggerName</i>	Name of the logger.

true to verbose.

5.7.3.3 void fnn::LogManager::Save ( string *dir* )

Saves the set of all loggers under a directory and a sub directory. Consider is main logger property.

=====

William, 4/29/2015.

The directory to which to save.

=====

William, 4/29/2015.

The directory to which to save.

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

- FNN++/FNNLib/FNNLogManager.h
- FNN++/FNNLib/FNNLogManager.cpp

## 5.8 fnn::Math Class Reference

The main mathematics helper class for FNNLIB

```
#include <FNNMath.h>
```

### Static Public Member Functions

- static double [NIntegrate](#) (std::function< double(double)> &f, double a, double b, double eps)  
*Numerically integrates any integrable function on a compact Hausdorff space. Integration occurs using Simpson's rule.*
- static double [NIntegrate](#) (std::function< double(double)> &f, double a, double b)  
*Numerically integrates any integrable function using Simpson's rule with auto scaling.*

- static double [UniformReal](#) (double min=0.0, double max=0.0)  
*Uniform real.*
- static double [GaussianReal](#) (double mean=0.0, double dev=1.0)  
*Gaussian real.*
- static std::vector< double > [PolyMult](#) (std::vector< double > &poly1, std::vector< double > &poly2)  
*A polynomial multiplication helper*
- static std::function< double(double)> [LERP](#) (std::vector< std::vector< double >> &data)  
*A linear interpolation algorithm*
- static std::function< double(double)> [LagrangeInterpolation](#) (std::vector< std::vector< double >> &data)  
*A polynomial interpolation algorithm using the Lagrange Interpolation Polynomial according to [http://en.wikipedia.org/wiki/Polynomial\\_interpolation](http://en.wikipedia.org/wiki/Polynomial_interpolation)*
- static int [Factorial](#) (int n)  
*Factorial implementation.*
- static std::vector< double > [GaussJordan](#) (std::vector< std::vector< double >> &matrix)  
*Gauss Jordan elimination for matrices.*
- static std::function< double(double)> [SSpline](#) (std::vector< std::vector< double >> &data)  
*A simple spline interpolation algorithm as described in [http://www.geos.ed.ac.uk/~yliu23/docs/lecture\\_spline.pdf](http://www.geos.ed.ac.uk/~yliu23/docs/lecture_spline.pdf). Makes the assumption that the second derivative at the boundaries is equal to 0.*
- static void [DataSort](#) (std::vector< std::vector< double >> &data)  
*Data sort algorithm to sort by x-values of the data. Useful for the interpolation algorithms.*
- static double [Mean](#) (std::vector< double > &data)  
*Determines the mean of the input vector.*
- static double [StdDev](#) (std::vector< double > &data)  
*Standard deviation of the data.*

### 5.8.1 Detailed Description

The main mathematics helper class for FNNLIB

=====

William Guss, 4/6/2015.

### 5.8.2 Member Function Documentation

#### 5.8.2.1 void fnn::Math::DataSort ( std::vector< std::vector< double >> & data ) [static]

Data sort algorithm to sort by x-values of the data. Useful for the interpolation algorithms.

=====

Phillip Kuznetsov, 4/29/2015.

#### Parameters

<i>data</i>	2D vector of input data points. Each row is a point.
-------------	--

=====

Phillip Kuznetsov, 4/29/2015.

#### Parameters

\_\_\_\_\_

<i>data</i>	2D vector of input data points. Each row is a point.
-------------	--

A 2D vector of the same points sorted.

5.8.2.2 `int fnn::Math::Factorial ( int n ) [static]`

Factorial implementation.

=====

William, 4/26/2015.

Parameters

<i>n</i>	The int to process.
----------	---------------------

An int.

5.8.2.3 `double fnn::Math::GaussianReal ( double mean = 0.0, double dev = 1.0 ) [static]`

Gaussian real.

=====

William Guss, 4/11/2015.

Parameters

<i>mean</i>	The mean.
-------------	-----------

A double.

5.8.2.4 `std::vector< double > fnn::Math::GaussJordan ( std::vector< std::vector< double >> & matrix ) [static]`

Gauss Jordan elimination for matrices.

=====

Phillip Kuznetsov, 4/29/2015.

Parameters

<i>matrix</i>	The systems of equation augmented matrix.
---------------	---

A vector of the variable values solved by completed Gauss-Jordan elimination.

=====

Phillip Kuznetsov, 4/29/2015.

Parameters

<i>matrix</i>	The systems of equation augmented matrix. Passes by reference.
---------------	--

A vector of the variable values solved by completed Gauss-Jordan elimination.

**5.8.2.5** `std::function< double(double)> fnn::Math::LagrangeInterpolation ( std::vector< std::vector< double >> & data )`  
`[static]`

A polynomial interpolation algorithm using the Lagrange Interpolation Polynomial according to [http://en.wikipedia.org/wiki/Polynomial\\_interpolation](http://en.wikipedia.org/wiki/Polynomial_interpolation)

=====

Phillip Kuznetsov, 4/19/2015.

#### Parameters

<i>data</i>	2D vector of input data points. Each row is a point.
-------------	--

A polynomial interpolation function

**5.8.2.6** `std::function< double(double)> fnn::Math::LERP ( std::vector< std::vector< double >> & data )` `[static]`

A linear interpolation algorithm

=====

Phillip Kuznetsov, 4/18/2015.

#### Parameters

<i>data</i>	2D vector of input data points
-------------	--------------------------------

A linear interpolation function

**5.8.2.7** `double fnn::Math::Mean ( std::vector< double > & data )` `[static]`

Determines the mean of the input vector.

Phillip Kuznetsov, 5/6/2015.

#### Parameters

<i>data</i>	The data.
-------------	-----------

#### Returns

The mean of the data

Phillip Kuznetsov, 5/6/2015.

#### Parameters

<i>data</i>	The data.
-------------	-----------

#### Returns

The mean of the data.

**5.8.2.8** `double fnn::Math::NIntegrate ( std::function< double(double)> & f, double a, double b, double eps )` `[static]`

Numerically integrates any integrable function on a compact Hausdorff space. Integration occurs using Simpson's rule.

=====

William Guss, 4/6/2015.

## Parameters

<i>f</i>	The function to integrate.
<i>a</i>	The lower bound of integration.
<i>b</i>	The upper bound of integration.
<i>eps</i>	The step-size of integration using Simpson's rule.

The result.

=====

William Guss, 4/6/2015.

## Parameters

<i>f</i>	The function to integrate.
<i>a</i>	The lower bound of integration.
<i>b</i>	The upper bound of integration.
<i>eps</i>	The step-size of integration using Simpson's rule.

The result.

**5.8.2.9** `double fnn::Math::NIntegrate ( std::function< double(double)> &f, double a, double b ) [static]`

Numerically integrates any integrable function using Simpson's rule with auto scaling.

=====

William Guss, 4/6/2015.

## Parameters

<i>f</i>	The function to integrate.
<i>a</i>	The lower bound of integration.
<i>b</i>	The upper bound of integration.

The result.

**5.8.2.10** `std::vector< double > fnn::Math::PolyMult ( std::vector< double > &poly1, std::vector< double > &poly2 ) [static]`

A polynomial multiplication helper

=====

Phillip Kuznetsov, 4/19/2015.

## Parameters

<i>poly1</i>	A vector of the coefficients. Each index is the power which x is raised to
--------------	--

## Parameters

<i>poly2</i>	A vector of the second polynomial coefficients./param>
--------------	--

A vector of coefficients for hte polynomial

5.8.2.11 `std::function< double(double)> fnn::Math::SSpline ( std::vector< std::vector< double >> & data ) [static]`

A simple spline interpolation algorithm as described in [http://www.geos.ed.ac.uk/~yliu23/docs/lecture\\_spline.pdf](http://www.geos.ed.ac.uk/~yliu23/docs/lecture_spline.pdf). Makes the assumption that the second derivative at the boundaries is equal to 0.

=====

Phillip Kuznetsov, 4/29/2015.

#### Parameters

<i>data</i>	2D vector of input data points. Each row is a point.
-------------	--

#### A polynomial interpolation function

=====

Phillip Kuznetsov, 4/29/2015.

2D vector of input data points. Each row is a point.

k: the vector of three coordinates that the current spline is working with. Dimensions are (x,y)

The coefficients of the equation  $d^2f(k-1)/dx^2 \text{ coef}[0] + d^2f(k)/dx^2 \text{ coef}[1] + d^2f(k+1)/dx^2 \text{ coef}[2]$

The coefs index counter. The for loop goes through the row and fills them with the three coefficients of the equation set.

This is why we initialized the size to `data.size()+1`.

5.8.2.12 `double fnn::Math::StdDev ( std::vector< double > & data ) [static]`

Standard deviation of the data.

The population standard deviation of the data.

Phillip Kuznetsov, 5/6/2015.

#### Parameters

<i>data</i>	The data.
-------------	-----------

#### Returns

The standard deviation of the data

Phillip Kuznetsov, 5/6/2015.

#### Parameters

<i>data</i>	The data.
-------------	-----------

#### Returns

The standard deviation of the data.

5.8.2.13 `double fnn::Math::UniformReal ( double min = 0.0, double max = 0.0 ) [static]`

Uniform real.

=====

William Guss, 4/11/2015.



## Parameters

<i>min</i>	The minimum.
<i>max</i>	The maximum.

A double.

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

- FNN++/FNNLib/FNNMath.h
- FNN++/FNNLib/FNNMath.cpp

## 5.9 fnn::Network Class Reference

The main class of operation on the functional neural networks.

```
#include <FNNNetwork.h>
```

Inheritance diagram for fnn::Network:

### Public Member Functions

- [Network](#) ()  
*Constructs a functional neural network..*
- `std::function< double(double)>` [FeedForward](#) (`std::function< double(double)>` )  
*Runs the network using the fast feedforward algorithm. The algorithm caches following that described in the paper.*
- `double` [BackPropagate](#) (`std::function< double(double)>` )  
*Back propagate using the Super Pro Algo developed by William Guss and Patrick Chen.*
- `void` [SetActivation](#) ([Sigmoid](#) activator)  
*Sets an activation.*
- `void` [AddLayer](#) (int x, int y)  
*Adds a layer to 'y'.*
- `double` [Train](#) ([DataPoint](#) &dp, `std::vector< double >` learningParameters)  
*Trains the network*
- `void` [NudgeWeights](#) ()  
*Nudge weights.*

### Public Attributes

- [Sigmoid Activator](#)  
*The primary activator type for the neural network.*

#### 5.9.1 Detailed Description

The main class of operation on the functional neural networks.

William, 4/9/2015.

## 5.9.2 Constructor & Destructor Documentation

### 5.9.2.1 fnn::Network::Network ( )

Constructs a functional neural network..

William, 4/10/2015.

The layer count.

Constructs a functional neural network.

William, 4/10/2015.

The layer count.

## 5.9.3 Member Function Documentation

### 5.9.3.1 void fnn::Network::AddLayer ( int x, int y )

Adds a layer to 'y'.

William Guss, 4/12/2015.

Parameters

x	The x coordinate.
---	-------------------

The y coordinate.

### 5.9.3.2 double fnn::Network::BackPropagate ( std::function< double(double)> )

Back propagate using the Super Pro Algo developed by William Guss and Patrick Chen.

William Guss, 5/6/2015.

Parameters

	The desired function delta .
--	------------------------------

The total integrated error over the last interval.

### 5.9.3.3 std::function< double(double)> fnn::Network::FeedForward ( std::function< double(double)> )

Runs the network using the fast feedforward algorithm. The algorithm caches following that described in the paper.

William, 4/10/2015.

## Parameters

	The input, .
--	--------------

The output, F[].

#### 5.9.3.4 void fnn::Network::NudgeWeights ( void )

Nudge weights.

Phillip Kuznetsov, 5/8/2015.

#### 5.9.3.5 void fnn::Network::SetActivation ( Sigmoid activator )

Sets an activation.

=====

William Guss, 4/11/2015.

The activator.

#### 5.9.3.6 double fnn::Network::Train ( DataPoint & dp, std::vector< double > learningParameters )

Trains the network

Trains the network.

Phillip Kuznetsov, 5/8/2015.

## Parameters

<i>ds</i>	The current datapoint
<i>learning↔ Parameters</i>	Options for controlling the learning.

## Returns

A double.

Phillip Kuznetsov, 5/8/2015.

## Parameters

<i>dp</i>	The current datapoint.
<i>learning↔ Parameters</i>	Options for controlling the learning.

## Returns

A double.

## 5.9.4 Member Data Documentation

### 5.9.4.1 Sigmoid fnn::Network::Activator

The primary activator type for the neural network.

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

- FNN++/FNNLib/FNNNetwork.h
- FNN++/FNNLib/FNNNetwork.cpp

## 5.10 fnn::Sigmoid Class Reference

### Public Member Functions

- [Sigmoid](#) (std::function< double(double)> f, std::function< double(double)> fprime)  
*Constructs a sigmoid function*
- [Sigmoid](#) ()  
*Default constructor.*
- double [prime](#) (double x)  
*Evaluates the derivative of the activation function.*
- double [operator\(\)](#) (double x)  
*Evaluates the sigmoid.*

### Static Public Member Functions

- static [Sigmoid Linear](#) ()  
*Gets the linear sigmoid activation.*
- static [Sigmoid Logistic](#) ()  
*The logistic activation function.*
- static [Sigmoid Tanh](#) ()  
*Gets the hyperbolic tangent activation function.*

### 5.10.1 Constructor & Destructor Documentation

#### 5.10.1.1 fnn::Sigmoid::Sigmoid ( std::function< double(double)> g, std::function< double(double)> gprime )

Constructs a sigmoid function

Default constructor. Do nothing.t

=====

William, 4/10/2015.

Parameters

<i>f</i>	The std::function<double(double)> to process.
----------	---

The fprime.

=====

William Guss, 4/12/2015.

#### 5.10.1.2 fnn::Sigmoid::Sigmoid ( )

Default constructor.

Default constructor. Do nothing.t

William Guss, 4/12/2015.

## 5.10.2 Member Function Documentation

### 5.10.2.1 fnn::Sigmoid fnn::Sigmoid::Linear ( void ) [static]

Gets the linear sigmoid activation.

William, 4/10/2015.

A **Sigmoid**.

### 5.10.2.2 fnn::Sigmoid fnn::Sigmoid::Logistic ( void ) [static]

The logistic activation function.

William, 4/10/2015.

A **Sigmoid**.

### 5.10.2.3 double fnn::Sigmoid::operator() ( double x )

Evaluates the sigmoid.

William, 4/10/2015.

Parameters

x	The x coordinate.
---	-------------------

The result of the operation.

### 5.10.2.4 double fnn::Sigmoid::prime ( double x )

Evaluates the derivative of the activation function.

William, 4/10/2015.

Parameters

x	The x coordinate.
---	-------------------

A double.

Evaluates the derivative of the activation function.

William, 4/10/2015.

## Parameters

x	The x coordinate.
---	-------------------

A double.

#### 5.10.2.5 fnn::Sigmoid fnn::Sigmoid::Tanh ( void ) [static]

Gets the hyperbolic tangent activation function.

=====

William, 4/10/2015.

A [Sigmoid](#).

=====

William, 4/10/2015. </remaqrks>

A [Sigmoid](#).

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

- FNN++/FNNLib/FNNSigmoid.h
- FNN++/FNNLib/FNNSigmoid.cpp

## 5.11 fnn::WeightSurface Class Reference

### Public Member Functions

- [WeightSurface](#) (int x, int y)  
*Default constructor.*
- double [operator\(\)](#) (double i, double j)  
*Function call operator.*
- double [GetCoefficient](#) (int x, int y)  
*Gets a coefficient.*
- void [Nudge](#) ()  
*Nudges the weight surface coefficients.*
- int [GetSizeX](#) ()  
*Gets size x coordinate.*
- int [GetSizeY](#) ()  
*Gets size y coordinate.*

### 5.11.1 Constructor & Destructor Documentation

#### 5.11.1.1 fnn::WeightSurface::WeightSurface ( int x, int y )

Default constructor.

=====

=====

William Guss, 4/11/2015.

Default constructor.

William Guss, 4/11/2015.

Parameters

x	The x depth.
---	--------------

The y depth.

## 5.11.2 Member Function Documentation

### 5.11.2.1 double fnn::WeightSurface::GetCoefficient ( int x, int y )

Gets a coefficient.

=====

William Guss, 4/11/2015.

Parameters

x	The x coordinate.
y	The y coordinate.

The coefficient.

### 5.11.2.2 int fnn::WeightSurface::GetSizeX ( void )

Gets size x coordinate.

=====

William Guss, 4/11/2015.

The size x coordinate.

### 5.11.2.3 int fnn::WeightSurface::GetSizeY ( void )

Gets size y coordinate.

=====

William Guss, 4/11/2015.

The size y coordinate.

### 5.11.2.4 void fnn::WeightSurface::Nudge ( void )

Nudges the weight surface coefficients.

=====

William Guss, 4/11/2015.

#### 5.11.2.5 `double fnn::WeightSurface::operator() ( double i, double j )`

Function call operator.

=====

William Guss, 4/11/2015.

##### Parameters

<i>i</i>	Zero-based index of the .
<i>j</i>	The double to process.

The result of the operation.

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

- FNN++/FNNLib/FNNWeightSurface.h
- FNN++/FNNLib/FNNWeightSurface.cpp



# Index

- Activator
  - fnn::Network, [29](#)
- AddLayer
  - fnn::Network, [28](#)
- AddLog
  - fnn::Loggable, [17](#)
- BackPropagate
  - fnn::Network, [28](#)
- Bound
  - fnn::FNNTrainer, [13](#)
- calcError
  - fnn::DataSet, [10](#)
- calculateErrors
  - fnn::DataSet, [10](#)
- DataSet
  - fnn::DataSet, [9](#)
- DataSet::operator[]
  - fnn::DataSet, [10](#)
- DataSort
  - fnn::Math, [22](#)
- Experiment
  - fnn::Experiment, [11](#)
- FNNTrainer
  - fnn::FNNTrainer, [13](#)
- Factorial
  - fnn::Math, [23](#)
- FeedForward
  - fnn::Network, [28](#)
- fnn, [7](#)
- fnn::DataPoint, [9](#)
- fnn::DataSet, [9](#)
  - calcError, [10](#)
  - calculateErrors, [10](#)
  - DataSet, [9](#)
  - DataSet::operator[], [10](#)
  - Load, [10](#)
  - Shuffle, [10](#)
  - size, [11](#)
- fnn::Experiment, [11](#)
  - Experiment, [11](#)
  - Run, [12](#)
  - RunAsThread, [12](#)
  - toggleThread, [12](#)
- fnn::FNNTrainer, [12](#)
  - Bound, [13](#)
  - FNNTrainer, [13](#)
- Train, [13](#)
- fnn::Log, [14](#)
  - GetContent, [15](#)
  - GetName, [15](#)
  - IsVerbose, [15](#)
  - Log, [14](#)
  - Push, [15](#)
- fnn::LogManager, [20](#)
  - LogManager, [20](#)
  - Print, [20](#)
  - Register, [21](#)
  - Save, [21](#)
- fnn::Loggable, [16](#)
  - AddLog, [17](#)
  - GetLogs, [17](#)
  - GetName, [17](#)
  - Initialize, [17](#)
  - IsVerbose, [19](#)
  - Log, [19](#)
  - Loggable, [16](#)
  - SetVerbose, [19](#)
- fnn::Math, [21](#)
  - DataSort, [22](#)
  - Factorial, [23](#)
  - GaussJordan, [23](#)
  - GaussianReal, [23](#)
  - LERP, [24](#)
  - LagrangeInterpolation, [23](#)
  - Mean, [24](#)
  - NIntegrate, [24](#), [25](#)
  - PolyMult, [25](#)
  - SSpline, [25](#)
  - StdDev, [26](#)
  - UniformReal, [26](#)
- fnn::Network, [27](#)
  - Activator, [29](#)
  - AddLayer, [28](#)
  - BackPropagate, [28](#)
  - FeedForward, [28](#)
  - Network, [28](#)
  - NudgeWeights, [29](#)
  - SetActivation, [29](#)
  - Train, [29](#)
- fnn::Sigmoid, [30](#)
  - Linear, [31](#)
  - Logistic, [31](#)
  - operator(), [31](#)
  - prime, [31](#)
  - Sigmoid, [30](#)

- Tanh, 32
- fnn::WeightSurface, 32
  - GetCoefficient, 33
  - GetSizeX, 33
  - GetSizeY, 33
  - Nudge, 33
  - operator(), 33
  - WeightSurface, 32
- GaussJordan
  - fnn::Math, 23
- GaussianReal
  - fnn::Math, 23
- GetCoefficient
  - fnn::WeightSurface, 33
- GetContent
  - fnn::Log, 15
- GetLogs
  - fnn::Loggable, 17
- GetName
  - fnn::Log, 15
  - fnn::Loggable, 17
- GetSizeX
  - fnn::WeightSurface, 33
- GetSizeY
  - fnn::WeightSurface, 33
- Initialize
  - fnn::Loggable, 17
- IsVerbose
  - fnn::Log, 15
  - fnn::Loggable, 19
- LERP
  - fnn::Math, 24
- LagrangeInterpolation
  - fnn::Math, 23
- Linear
  - fnn::Sigmoid, 31
- Load
  - fnn::DataSet, 10
- Log
  - fnn::Log, 14
  - fnn::Loggable, 19
- LogManager
  - fnn::LogManager, 20
- Loggable
  - fnn::Loggable, 16
- Logistic
  - fnn::Sigmoid, 31
- Mean
  - fnn::Math, 24
- NIntegrate
  - fnn::Math, 24, 25
- Network
  - fnn::Network, 28
- Nudge
  - fnn::WeightSurface, 33
- NudgeWeights
  - fnn::Network, 29
- operator()
  - fnn::Sigmoid, 31
  - fnn::WeightSurface, 33
- PolyMult
  - fnn::Math, 25
- prime
  - fnn::Sigmoid, 31
- Print
  - fnn::LogManager, 20
- Push
  - fnn::Log, 15
- Register
  - fnn::LogManager, 21
- Run
  - fnn::Experiment, 12
- RunAsThread
  - fnn::Experiment, 12
- SSpline
  - fnn::Math, 25
- Save
  - fnn::LogManager, 21
- SetActivation
  - fnn::Network, 29
- SetVerbose
  - fnn::Loggable, 19
- Shuffle
  - fnn::DataSet, 10
- Sigmoid
  - fnn::Sigmoid, 30
- size
  - fnn::DataSet, 11
- StdDev
  - fnn::Math, 26
- Tanh
  - fnn::Sigmoid, 32
- toggleThread
  - fnn::Experiment, 12
- Train
  - fnn::FNNTrainer, 13
  - fnn::Network, 29
- UniformReal
  - fnn::Math, 26
- WeightSurface
  - fnn::WeightSurface, 32