

# Implémentation de l'algorithme de rétropropagation sur un réseau neuronal

Gaspard Coulet - Félix Guyard - Areski Himeur - Lucas Picasarri-Arrieta - Jean Maurice Raboude



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	Archi.DrawNN Class Reference . . . . .	5
3.2	FonctionActivation Class Reference . . . . .	5
3.2.1	Detailed Description . . . . .	6
3.2.2	Constructor & Destructor Documentation . . . . .	6
3.2.2.1	FonctionActivation() . . . . .	6
3.2.2.2	FonctionActivation(EnumFonctionActivation fonctionActivation) . . . . .	6
3.2.3	Member Function Documentation . . . . .	6
3.2.3.1	getValeurActivation(double sum, double k=0) const . . . . .	6
3.2.3.2	getValeurDerivee(double sum, double k=0) const . . . . .	6
3.2.3.3	setFonctionActivation(EnumFonctionActivation fonctionActivation) . . . . .	7
3.3	InputLayer Class Reference . . . . .	7
3.3.1	Constructor & Destructor Documentation . . . . .	8
3.3.1.1	InputLayer(int, FonctionActivation::EnumFonctionActivation fct) . . . . .	8
3.3.2	Member Function Documentation . . . . .	8
3.3.2.1	fire(std::vector< double >, double) . . . . .	8
3.4	Archi.Layer Class Reference . . . . .	9
3.5	Layer Class Reference . . . . .	9

3.5.1	Constructor & Destructor Documentation . . . . .	10
3.5.1.1	Layer() . . . . .	10
3.5.1.2	Layer(TypeLayer, int, int, FonctionActivation::EnumFonctionActivation) . . . . .	10
3.5.2	Member Function Documentation . . . . .	10
3.5.2.1	fire(std::vector< double >, double) . . . . .	10
3.5.2.2	getInput() . . . . .	11
3.5.2.3	getNbNeurones() . . . . .	11
3.5.2.4	getNeurone(int index) . . . . .	11
3.5.2.5	printWeight() . . . . .	11
3.6	Archi.NeuralNetwork Class Reference . . . . .	12
3.7	Archi.Neuron Class Reference . . . . .	12
3.8	Neurone Class Reference . . . . .	13
3.8.1	Constructor & Destructor Documentation . . . . .	14
3.8.1.1	Neurone() . . . . .	14
3.8.1.2	Neurone(int taille, FonctionActivation::EnumFonctionActivation fct) . . . . .	14
3.8.1.3	Neurone(int taille, std::vector< double > *x, FonctionActivation::EnumFonctionActivation fct) . . . . .	14
3.8.2	Member Function Documentation . . . . .	14
3.8.2.1	aleaWeights() . . . . .	14
3.8.2.2	derive_activate(double sum, double k) const . . . . .	14
3.8.2.3	fire(std::vector< double > x, double k) const . . . . .	15
3.8.2.4	fw_activate(double sum, double k) const . . . . .	15
3.8.2.5	fw_sum(std::vector< double > x) const . . . . .	15
3.8.2.6	getNbPoids() . . . . .	16
3.8.2.7	getWeight() . . . . .	16
3.8.2.8	learn(std::vector< double > x, double o, double k, double mu) . . . . .	16
3.8.2.9	printWeight() . . . . .	16
3.9	NeuroneB Class Reference . . . . .	17
3.9.1	Constructor & Destructor Documentation . . . . .	18
3.9.1.1	NeuroneB(int taille, FonctionActivation::EnumFonctionActivation fct) . . . . .	18
3.9.1.2	NeuroneB(int taille, std::vector< double > *x, double biais_w) . . . . .	18

3.9.2	Member Function Documentation	18
3.9.2.1	aleaWeights()	18
3.9.2.2	fw_sum(std::vector< double > x) const	18
3.9.2.3	getNbPoids()	19
3.9.2.4	learn(std::vector< double > x, double o, double k, double mu)	19
3.9.2.5	printWeight()	19
3.10	Option Class Reference	19
3.10.1	Constructor & Destructor Documentation	20
3.10.1.1	Option()	20
3.10.2	Member Function Documentation	20
3.10.2.1	getOptionID() const	20
3.10.2.2	getOptionInt() const	20
3.10.2.3	getOptionRac() const	20
3.10.2.4	getOptionType() const	20
3.10.2.5	print(std::ostream &os) const	20
3.10.2.6	setOptionInt(std::string &intitul)	21
3.10.2.7	setOptionRac(std::string &rac)	21
3.10.2.8	setOptionType(Type::TypeEnum t)	21
3.11	OptionTab Class Reference	21
3.11.1	Constructor & Destructor Documentation	22
3.11.1.1	OptionTab()	22
3.11.2	Member Function Documentation	22
3.11.2.1	addOption(const Option &opt)	22
3.11.2.2	getArgument(const std::string &opt) const	22
3.11.2.3	getIntitul(const std::string &opt) const	22
3.11.2.4	getOptionID(const std::string &opt) const	22
3.11.2.5	getRaccour(const std::string &opt) const	23
3.11.2.6	printOptions() const	23
3.12	Reseau Class Reference	23
3.12.1	Constructor & Destructor Documentation	23

3.12.1.1	<code>Reseau(int, std::vector&lt; int &gt;, double, double, FonctionActivation::Enum↵ FonctionActivation fct)</code>	23
3.12.2	Member Function Documentation	24
3.12.2.1	<code>backPropagation(std::vector&lt; double &gt; erreurs)</code>	24
3.12.2.2	<code>fire_all(std::vector&lt; double &gt; input)</code>	24
3.12.2.3	<code>learn(std::vector&lt; std::vector&lt; std::vector&lt; double &gt; &gt; &gt; jeuxTest, unsigned int nbPasDescenteGradient)</code>	24
3.12.2.4	<code>printWeight()</code>	24
3.13	Settings Class Reference	25
3.13.1	Constructor & Destructor Documentation	25
3.13.1.1	<code>Settings(char const *fileName)</code>	25
3.13.2	Member Function Documentation	25
3.13.2.1	<code>getArchi()</code>	25
3.13.2.2	<code>getDifferentOutputs()</code>	25
3.13.2.3	<code>getNbrHiddenLayers()</code>	26
3.14	Type Class Reference	26
3.14.1	Constructor & Destructor Documentation	26
3.14.1.1	<code>Type()</code>	26
3.14.1.2	<code>Type(Type::TypeEnum type)</code>	26
3.14.2	Member Function Documentation	26
3.14.2.1	<code>getType() const</code>	27
3.14.2.2	<code>print(std::ostream &amp;os) const</code>	27
3.14.2.3	<code>setType(Type::TypeEnum type)</code>	27

# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

Archi.DrawNN . . . . .	5
FonctionActivation . . . . .	5
Archi.Layer . . . . .	9
Layer . . . . .	9
InputLayer . . . . .	7
Archi.NeuralNetwork . . . . .	12
Archi.Neuron . . . . .	12
Neurone . . . . .	13
NeuroneB . . . . .	17
Option . . . . .	19
OptionTab . . . . .	21
Reseau . . . . .	23
Settings . . . . .	25
Type . . . . .	26





## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">Archi.DrawNN</a>	5
<a href="#">FonctionActivation</a>	5
<a href="#">InputLayer</a>	7
<a href="#">Archi.Layer</a>	9
<a href="#">Layer</a>	9
<a href="#">Archi.NeuralNetwork</a>	12
<a href="#">Archi.Neuron</a>	12
<a href="#">Neurone</a>	13
<a href="#">NeuroneB</a>	17
<a href="#">Option</a>	19
<a href="#">OptionTab</a>	21
<a href="#">Reseau</a>	23
<a href="#">Settings</a>	25
<a href="#">Type</a>	26



## Chapter 3

# Class Documentation

### 3.1 Archi.DrawNN Class Reference

#### Public Member Functions

- def **\_\_init\_\_** (self, neural\_network)
- def **draw** (self)

#### Public Attributes

- **neural\_network**

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

- Archi.py

### 3.2 FonctionActivation Class Reference

```
#include <FonctionActivation.h>
```

#### Public Types

- enum **EnumFonctionActivation** {  
    **IDENTITY**, **BINARYSTEP**, **SIGMOID**, **TAN**,  
    **SIN**, **RELU**, **LRELU**, **PRELU**,  
    **ELU** }

#### Public Member Functions

- [FonctionActivation](#) ()
- [FonctionActivation](#) (EnumFonctionActivation fonctionActivation)
- void [setFonctionActivation](#) (EnumFonctionActivation fonctionActivation)
- double [getValeurActivation](#) (double sum, double k=0) const
- double [getValeurDerivee](#) (double sum, double k=0) const

### 3.2.1 Detailed Description

L'objectif de cette classe est d'être appelée pour l'activation et donc changer plus facilement la fonction d'activation pour le test.

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 `FonctionActivation::FonctionActivation ( )`

Constructeur par défauts [FonctionActivation](#)

##### Parameters

<i>fonctionActivation</i>	Fonction activation à utiliser parmi l'énumération
---------------------------	--

#### 3.2.2.2 `FonctionActivation::FonctionActivation ( EnumFonctionActivation fonctionActivation )`

Constructeur [FonctionActivation](#)

##### Parameters

<i>fonctionActivation</i>	Fonction activation à utiliser parmi l'énumération
---------------------------	--

### 3.2.3 Member Function Documentation

#### 3.2.3.1 `double FonctionActivation::getValeurActivation ( double x, double k = 0 ) const`

`getValeurActivation`

##### Parameters

<i>x</i>	Valeur de la variable
<i>k</i>	Constante de changement

##### Returns

Valeur d'activation

#### 3.2.3.2 `double FonctionActivation::getValeurDerivee ( double x, double k = 0 ) const`

`getValeurDerivee`

## Parameters

$x$	Valeur de la variable
$k$	Constante de changement

## Returns

Valeur de la derivée

3.2.3.3 void FonctionActivation::setFonctionActivation ( EnumFonctionActivation *fonctionActivation* )

[FonctionActivation::setFonctionActivation](#)

## Parameters

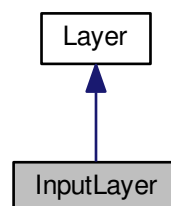
<i>fonctionActivation</i>	Fonction activation à utiliser parmi l'enumération
---------------------------	--

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

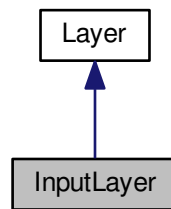
- Neurone/FonctionActivation.h
- Neurone/FonctionActivation.cpp

## 3.3 InputLayer Class Reference

Inheritance diagram for InputLayer:



Collaboration diagram for InputLayer:



## Public Member Functions

- [InputLayer](#) (int, FonctionActivation::EnumFonctionActivation fct)
- `std::vector< double >` [fire](#) (`std::vector< double >`, double)

## Additional Inherited Members

### 3.3.1 Constructor & Destructor Documentation

#### 3.3.1.1 `InputLayer::InputLayer ( int taille, FonctionActivation::EnumFonctionActivation fct )`

Constructeur par taille [InputLayer::InputLayer](#)

##### Parameters

<i>taille</i>	Nombre de neurones dans le layer
---------------	----------------------------------

### 3.3.2 Member Function Documentation

#### 3.3.2.1 `std::vector< double > InputLayer::fire ( std::vector< double > input, double k )` [virtual]

Méthode de propagation en avant [InputLayer::fire](#)

##### Parameters

<i>input</i>	Vecteur en entrées
<i>k</i>	Coefficient de sigmoid

#### Returns

Valeur d'activation

Reimplemented from [Layer](#).

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

- Layer/InputLayer.h
- Layer/InputLayer.cpp

## 3.4 Archi.Layer Class Reference

#### Public Member Functions

- def **\_\_init\_\_** (self, network, number\_of\_neurons, number\_of\_neurons\_in\_widest\_layer)
- def **draw** (self, layerType=0)

#### Public Attributes

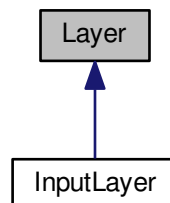
- **vertical\_distance\_between\_layers**
- **horizontal\_distance\_between\_neurons**
- **neuron\_radius**
- **number\_of\_neurons\_in\_widest\_layer**
- **previous\_layer**
- **y**
- **neurons**

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

- Archi.py

## 3.5 Layer Class Reference

Inheritance diagram for Layer:



## Public Types

- enum **TypeLayer** { **INPUT**, **OUTPUT**, **HIDDEN** }

## Public Member Functions

- [Layer](#) ()
- [Layer](#) (TypeLayer, int, int, FonctionActivation::EnumFonctionActivation)
- [Neurone](#) \* [getNeurone](#) (int index)
- virtual std::vector< double > [fire](#) (std::vector< double >, double)
- int [getNbNeurones](#) ()
- std::vector< double > [getInput](#) ()
- std::vector< double > [getOutput](#) ()
- void [printWeight](#) ()

## Protected Attributes

- double **k**
- int **nbNeurone**
- std::vector< [Neurone](#) \* > **membres**
- std::vector< double > **input**
- std::vector< double > **output**
- TypeLayer **type**

### 3.5.1 Constructor & Destructor Documentation

#### 3.5.1.1 [Layer::Layer](#) ( )

Constructeur par défauts [Layer::Layer](#)

#### 3.5.1.2 [Layer::Layer](#) ( TypeLayer *type*, int *nbneur*, int *nbinput*, FonctionActivation::EnumFonctionActivation *fct* )

Constructeur avec arguments [Layer::Layer](#)

##### Parameters

<i>type</i>	<a href="#">Type</a> de layer
<i>nbneur</i>	Nombre de neurone
<i>nbinput</i>	Nombre d'entrée par neurone

### 3.5.2 Member Function Documentation

#### 3.5.2.1 std::vector< double > [Layer::fire](#) ( std::vector< double > *input*, double *k* ) [virtual]

Propagation en avant [Layer::fire](#)



## Parameters

<i>input</i>	Vecteur en entrées
<i>k</i>	Coefficient de sigmoid

## Returns

Vecteur des valeurs d'activations

Reimplemented in [InputLayer](#).

3.5.2.2 `std::vector< double > Layer::getInput ( )`

[Layer::getInput](#)

## Returns

Retourne le vecteur d'entrées

3.5.2.3 `int Layer::getNbNeurones ( )`

[Layer::getNbNeurones](#)

## Returns

Nombre de neurones

3.5.2.4 `Neurone * Layer::getNeurone ( int index )`

[Layer::getNeurone](#)

## Parameters

<i>index</i>	index
--------------	-------

## Returns

[Neurone](#)

3.5.2.5 `void Layer::printWeight ( )`

Affiche les poids [Layer::printWeight](#)

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

- Layer/Layer.h
- Layer/Layer.cpp

## 3.6 Archi.NeuralNetwork Class Reference

### Public Member Functions

- def **\_\_init\_\_** (self, number\_of\_neurons\_in\_widest\_layer)
- def **add\_layer** (self, number\_of\_neurons)
- def **draw** (self)

### Public Attributes

- **number\_of\_neurons\_in\_widest\_layer**
- **layers**
- **layertype**

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

- Archi.py

## 3.7 Archi.Neuron Class Reference

### Public Member Functions

- def **\_\_init\_\_** (self, x, y)
- def **draw** (self, neuron\_radius)

### Public Attributes

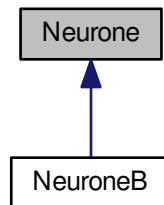
- **x**
- **y**

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

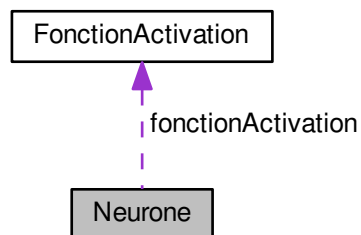
- Archi.py

## 3.8 Neurone Class Reference

Inheritance diagram for Neurone:



Collaboration diagram for Neurone:



### Public Member Functions

- [Neurone](#) ()
- [Neurone](#) (int taille, [FonctionActivation::EnumFonctionActivation](#) fct)
- [Neurone](#) (int taille, [std::vector< double > \\*x](#), [FonctionActivation::EnumFonctionActivation](#) fct)
- virtual double [fw\\_sum](#) ([std::vector< double > x](#)) const
- double [fw\\_activate](#) (double sum, double k) const
- double [derive\\_activate](#) (double sum, double k) const
- virtual double [fire](#) ([std::vector< double > x](#), double k) const
- virtual void [learn](#) ([std::vector< double > x](#), double o, double k, double mu)
- virtual void [printWeight](#) ()
- virtual void [aleaWeights](#) ()
- [std::vector< double > \\* getWeight](#) ()
- virtual int [getNbPoids](#) ()

## Protected Attributes

- `std::vector< double > * w`
- `int n`
- [FonctionActivation](#) `fonctionActivation`

### 3.8.1 Constructor & Destructor Documentation

#### 3.8.1.1 `Neurone::Neurone ( )`

Constructeur par defaults [Neurone::Neurone](#)

#### 3.8.1.2 `Neurone::Neurone ( int n, FonctionActivation::EnumFonctionActivation fct )`

Constructeur par nombre d'entrées [Neurone::Neurone](#)

##### Parameters

<i>n</i>	Nombre d'entrées
<i>n</i>	Fonction d'Activation

#### 3.8.1.3 `Neurone::Neurone ( int n, std::vector< double > * x, FonctionActivation::EnumFonctionActivation fct )`

Constructeur par arguments [Neurone::Neurone](#)

##### Parameters

<i>taille</i>	Nombre d'entrées
<i>x</i>	Vecteur de poids

### 3.8.2 Member Function Documentation

#### 3.8.2.1 `void Neurone::aleaWeights ( ) [virtual]`

Méthode de génération de poids aléatoire selon une loi normale centrée réduite [Neurone::aleaWeights](#)

Reimplemented in [NeuroneB](#).

#### 3.8.2.2 `double Neurone::derive_activate ( double sum, double k ) const`

Méthode de dérivation [Neurone::derive\\_activate](#)

## Parameters

<i>sum</i>	Valeur de la somme du vecteur (cf <a href="#">Neurone::fw_sum</a> )
<i>k</i>	Valeur du coefficient de sigmoid k

## Returns

Valeur de dérivation

**3.8.2.3** `double Neurone::fire ( std::vector< double > in, double k ) const` `[virtual]`

Méthode de propagation en avant [Neurone::fire](#)

## Parameters

<i>in</i>	Vecteur d'entrées
<i>k</i>	Coefficient de sigmoid k

## Returns

Valeur d'activation

**3.8.2.4** `double Neurone::fw_activate ( double sum, double k ) const`

Méthode d'activation [Neurone::fw\\_activate](#)

## Parameters

<i>sum</i>	Valeur de la somme du vecteur (cf <a href="#">Neurone::fw_sum</a> )
<i>k</i>	Valeur du biais

## Returns

Valeur d'activation

**3.8.2.5** `double Neurone::fw_sum ( std::vector< double > x ) const` `[virtual]`

Méthode de somme des valeurs du vecteur [Neurone::fw\\_sum](#)

## Parameters

<i>x</i>	Vecteur
----------	---------

**Returns**

Somme

Reimplemented in [NeuroneB](#).

**3.8.2.6** `int Neurone::getNbPoids ( )` [virtual]

[Neurone::getNbPoids](#)

**Returns**

Nombre de poids

Reimplemented in [NeuroneB](#).

**3.8.2.7** `std::vector< double > * Neurone::getWeight ( )`

[Neurone::getWeight](#)

**Returns**

Vecteur de poids

**3.8.2.8** `void Neurone::learn ( std::vector< double > x, double o, double k, double mu )` [virtual]

Méthode d'apprentissage (cf [Neurone::fw\\_sum](#), cf [Neurone::fw\\_activate](#), cf [Neurone::derive\\_activate](#)) [Neurone↔::learn](#)

**Parameters**

<i>x</i>	Vecteur de poids
<i>o</i>	Valeur attendue
<i>k</i>	Valeur de l'hyperparamètre
<i>mu</i>	Taux d'apprentissage   Learning rate

Reimplemented in [NeuroneB](#).

**3.8.2.9** `void Neurone::printWeight ( )` [virtual]

Affichage des poids [Neurone::printWeight](#)

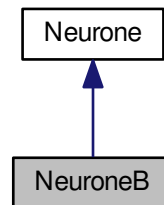
Reimplemented in [NeuroneB](#).

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

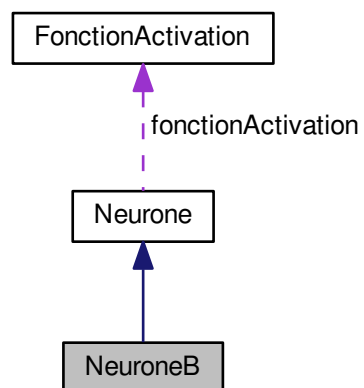
- Neurone/Neurone.h
- Neurone/Neurone.cpp

## 3.9 NeuroneB Class Reference

Inheritance diagram for NeuroneB:



Collaboration diagram for NeuroneB:



### Public Member Functions

- `NeuroneB` (int taille, FonctionActivation::EnumFonctionActivation fct)
- `NeuroneB` (int taille, std::vector< double > \*x, double biais\_w)
- void `printWeight` ()
- void `aleaWeights` ()
- void `learn` (std::vector< double > x, double o, double k, double mu)
- double `fw_sum` (std::vector< double > x) const
- int `getNbPoids` ()

## Additional Inherited Members

### 3.9.1 Constructor & Destructor Documentation

#### 3.9.1.1 `NeuroneB::NeuroneB ( int taille, FonctionActivation::EnumFonctionActivation fct )`

Constructeur du [Neurone](#) avec biais [NeuroneB::NeuroneB](#)

##### Parameters

<i>taille</i>	Taille du neurone
---------------	-------------------

#### 3.9.1.2 `NeuroneB::NeuroneB ( int taille, std::vector< double > * x, double biais_w )`

Constructeur du [Neurone](#) avec biais [NeuroneB::NeuroneB](#)

##### Parameters

<i>taille</i>	Taille du neurone
<i>x</i>	Vecteur de poids
<i>biais_w</i>	Valeur du biais

### 3.9.2 Member Function Documentation

#### 3.9.2.1 `void NeuroneB::aleaWeights ( )` [virtual]

Génération aléatoire des poids [NeuroneB::aleaWeights](#)

Reimplemented from [Neurone](#).

#### 3.9.2.2 `double NeuroneB::fw_sum ( std::vector< double > x ) const` [virtual]

Méthode de somme du vecteur de poids [NeuroneB::fw\\_sum](#)

##### Parameters

<i>x</i>	Vecteur de poids
----------	------------------

##### Returns

Somme des poids

Reimplemented from [Neurone](#).



3.9.2.3 `int NeuroneB::getNbPoids ( ) [virtual]`

[NeuroneB::getNbPoids](#)

Returns

Nombre de poids

Reimplemented from [Neurone](#).

3.9.2.4 `void NeuroneB::learn ( std::vector< double > x, double o, double k, double mu ) [virtual]`

Méthode d'apprentissage (cf [Neurone::fw\\_sum](#), cf [Neurone::fw\\_activate](#), cf [Neurone::derive\\_activate](#)) [NeuroneB::learn](#)

Parameters

<i>x</i>	Vecteur de poids
<i>o</i>	Valeur attendue
<i>k</i>	Valeur du biais
<i>mu</i>	Taux d'apprentissage   Learning rate

Reimplemented from [Neurone](#).

3.9.2.5 `void NeuroneB::printWeight ( ) [virtual]`

Affichage des poids [NeuroneB::printWeight](#)

Reimplemented from [Neurone](#).

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

- Neurone/NeuroneB.h
- Neurone/NeuroneB.cpp

## 3.10 Option Class Reference

### Public Member Functions

- [Option](#) ()
- **Option** (int id, const std::string &intitul, const std::string &m\_rac, const Type::TypeEnum type, const std::string &m\_description)
- int [getOptionID](#) () const
- Type::TypeEnum [getOptionType](#) () const
- std::string [getOptionInt](#) () const
- std::string [getOptionRac](#) () const
- void [setOptionType](#) (Type::TypeEnum t)
- void [setOptionInt](#) (std::string &intitul)
- void [setOptionRac](#) (std::string &rac)
- void [print](#) (std::ostream &os) const

### 3.10.1 Constructor & Destructor Documentation

#### 3.10.1.1 Option::Option ( )

Constructeur d'option vide [Option::Option](#)

### 3.10.2 Member Function Documentation

#### 3.10.2.1 int Option::getOptionID ( ) const

Option::GetOptionId

Returns

ID de l'option

#### 3.10.2.2 string Option::getOptionInt ( ) const

Option::GetOptionInt

Returns

Intitulé de l'option

#### 3.10.2.3 string Option::getOptionRac ( ) const

Option::GetOptionRac

Returns

Raccourci de l'option

#### 3.10.2.4 Type::TypeEnum Option::getOptionType ( ) const

Option::GetOptionType

Returns

[Type](#) d'option

#### 3.10.2.5 void Option::print ( std::ostream & os ) const

Méthode d'affichage en complément de << print

## Parameters

<i>os</i>	Flux
-----------	------

3.10.2.6 void Option::setOptionInt ( std::string & *intitul* )

Option::SetOptionInt

## Parameters

<i>intitul</i>	Nouveau Intitulé
----------------	------------------

3.10.2.7 void Option::setOptionRac ( std::string & *rac* )

Option::SetOptionRac

## Parameters

<i>rac</i>	Nouveau Raccourci
------------	-------------------

3.10.2.8 void Option::setOptionType ( Type::TypeEnum *t* )

Option::SetOptionType

## Parameters

<i>t</i>	Nouveau type d'option
----------	-----------------------

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

- Option/Option.h
- Option/Option.cpp

## 3.11 OptionTab Class Reference

### Public Member Functions

- [OptionTab](#) ()
- void [addOption](#) (const [Option](#) &opt)
- void [printOptions](#) () const
- int [getOptionID](#) (const std::string &opt) const
- Type::TypeEnum [getArgument](#) (const std::string &opt) const
- std::string [getRaccour](#) (const std::string &opt) const
- std::string [getTypeOption](#) (const std::string &opt) const
- std::string [getIntitul](#) (const std::string &opt) const

### 3.11.1 Constructor & Destructor Documentation

#### 3.11.1.1 OptionTab::OptionTab ( )

Constructeur par défauts [OptionTab::OptionTab](#)

### 3.11.2 Member Function Documentation

#### 3.11.2.1 void OptionTab::addOption ( const Option & *opt* )

Ajoute une option OptionTab::AddOption

##### Parameters

<i>opt</i>	[description]
------------	---------------

#### 3.11.2.2 Type::TypeEnum OptionTab::getArgument ( const std::string & *opt* ) const

OptionTab::GetArgument

##### Parameters

<i>opt</i>	Nom de l'option
------------	-----------------

##### Returns

Argument de cette option

#### 3.11.2.3 std::string OptionTab::getIntitul ( const std::string & *opt* ) const

[OptionTab::getIntitul](#)

##### Parameters

<i>opt</i>	Nom de l'option
------------	-----------------

##### Returns

Intitulé de cette option

#### 3.11.2.4 int OptionTab::getOptionID ( const std::string & *opt* ) const

OptionTab::GetOptionID

## Parameters

<i>opt</i>	Nom de l'option
------------	-----------------

## Returns

ID de cette option

3.11.2.5 `std::string OptionTab::getRaccour ( const std::string & opt ) const`

[OptionTab::getRaccour](#)

## Parameters

<i>opt</i>	Nom de l'option
------------	-----------------

## Returns

Raccourci de cette option

3.11.2.6 `void OptionTab::printOptions ( ) const`

Affiche les options `OptionTab::PrintOptions`

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

- Option/OptionTab.h
- Option/OptionTab.cpp

## 3.12 Reseau Class Reference

### Public Member Functions

- [Reseau](#) (int, std::vector< int >, double, double, FonctionActivation::EnumFonctionActivation fct)
- std::vector< double > [fire\\_all](#) (std::vector< double > input)
- void [learn](#) (std::vector< std::vector< std::vector< double > > > jeuxTest, unsigned int nbPasDescente↵ Gradient)
- void [backPropagation](#) (std::vector< double > erreurs)
- void [printWeight](#) ()

### 3.12.1 Constructor & Destructor Documentation

3.12.1.1 `Reseau::Reseau ( int nbLayers, std::vector< int > layerInformation, double k, double eta, FonctionActivation::EnumFonctionActivation fct )`

Constructeur [Reseau::Reseau](#)

## Parameters

<i>nbLayers</i>	Nombre de couche
<i>layerInformation</i>	Vecteur descriptif de chaque layer
<i>k</i>	constante k dont depends la sigmoide
<i>eta</i>	coefficient d'evolution

## 3.12.2 Member Function Documentation

3.12.2.1 void Reseau::backPropagation ( std::vector< double > *erreurs* )

Méthode de propagation en arrière [Reseau::backPropagation](#)

## Parameters

<i>output</i>	Sortie
<i>k</i>	Valeur du coefficient
<i>eta</i>	Valeur d'eta

3.12.2.2 std::vector< double > Reseau::fire\_all ( std::vector< double > *input* )

Fire général [Reseau::fire\\_all](#)

## Parameters

<i>input</i>	Vecteur d'entrée
--------------	------------------

## Returns

Valeur d'activation

3.12.2.3 void Reseau::learn ( std::vector< std::vector< std::vector< double > > > *jeuxTest*, unsigned int *nbPasDescenteGradient* )

Méthode d'apprentissage [Reseau::learn](#)

## Parameters

<i>jeuxTest</i>	Vecteur de vecteur de vecteur : décrivant en <code>jeuxTest[i][0]</code> le vecteur d'entrée et en <code>jeuxTest[i][1]</code> la sortie attendue
-----------------	---

## 3.12.2.4 void Reseau::printWeight ( )

Affichage des poids [Reseau::printWeight](#)

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

- Reseau/Reseau.h
- Reseau/Reseau.cpp

## 3.13 Settings Class Reference

### Public Member Functions

- [Settings](#) (char const \*fileName)
- int [getNbrHiddenLayers](#) ()
- std::vector< int > \* [getArchi](#) ()
- std::vector< double > \* [getDifferentOutputs](#) ()

### 3.13.1 Constructor & Destructor Documentation

#### 3.13.1.1 Settings::Settings ( char const \* *fileName* )

Constructeur settings::settings

##### Parameters

<i>fileName</i>	Nom du fichier
-----------------	----------------

Counter

Parser

strinsplit à partir du caractère ','

### 3.13.2 Member Function Documentation

#### 3.13.2.1 std::vector< int > \* Settings::getArchi ( )

settings::getArchi

##### Returns

Architecture

#### 3.13.2.2 std::vector< double > \* Settings::getDifferentOutputs ( )

settings::getDifferentOutputs

##### Returns

DifferentOutputs

### 3.13.2.3 int Settings::getNbrHiddenLayers ( )

settings::getNbrHiddenLayers

#### Returns

Nombre de Layers "caché"

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

- Settings/Settings.h
- Settings/Settings.cpp

## 3.14 Type Class Reference

### Public Types

- enum **TypeEnum** { **NONE**, **INT**, **FLOAT**, **STRING** }

### Public Member Functions

- [Type](#) ()
- [Type](#) (Type::TypeEnum type)
- void [print](#) (std::ostream &os) const
- void [setType](#) (Type::TypeEnum type)
- Type::TypeEnum [getType](#) () const

### 3.14.1 Constructor & Destructor Documentation

#### 3.14.1.1 Type::Type ( )

Constructeur par défaut [Type::Type](#)

#### 3.14.1.2 Type::Type ( Type::TypeEnum type )

Constructeur avec arguments [Type::Type](#)

#### Parameters

<i>type</i>	Nom du type
-------------	-------------

### 3.14.2 Member Function Documentation



#### 3.14.2.1 `Type::TypeEnum Type::getType ( ) const`

[Type::getType](#)

Returns

[Type](#)

#### 3.14.2.2 `void Type::print ( std::ostream & os ) const`

Affiche dans le flux une chaine de caractère décrivant le type [Type::print](#)

Parameters

<i>os</i>	Flux
-----------	------

#### 3.14.2.3 `void Type::setType ( Type::TypeEnum type )`

[Type::setType](#)

Parameters

<i>type</i>	Nouveau type
-------------	--------------

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

- Option/Type.h
- Option/Type.cpp



# Index

- addOption
  - OptionTab, 22
- aleaWeights
  - Neurone, 14
  - NeuroneB, 18
- Archi.DrawNN, 5
- Archi.Layer, 9
- Archi.NeuralNetwork, 12
- Archi.Neuron, 12
- backPropagation
  - Reseau, 24
- derive\_activate
  - Neurone, 14
- fire
  - InputLayer, 8
  - Layer, 10
  - Neurone, 15
- fire\_all
  - Reseau, 24
- FonctionActivation, 5
  - FonctionActivation, 6
  - getValeurActivation, 6
  - getValeurDerivee, 6
  - setFonctionActivation, 7
- fw\_activate
  - Neurone, 15
- fw\_sum
  - Neurone, 15
  - NeuroneB, 18
- getArchi
  - Settings, 25
- getArgument
  - OptionTab, 22
- getDifferentOutputs
  - Settings, 25
- getInput
  - Layer, 11
- getIntitul
  - OptionTab, 22
- getNbNeurones
  - Layer, 11
- getNbPoids
  - Neurone, 16
  - NeuroneB, 18
- getNbrHiddenLayers
  - Settings, 25
- getNeurone
  - Layer, 11
- getOptionID
  - Option, 20
  - OptionTab, 22
- getOptionInt
  - Option, 20
- getOptionRac
  - Option, 20
- getOptionType
  - Option, 20
- getRaccour
  - OptionTab, 23
- getType
  - Type, 26
- getValeurActivation
  - FonctionActivation, 6
- getValeurDerivee
  - FonctionActivation, 6
- getWeight
  - Neurone, 16
- InputLayer, 7
  - fire, 8
  - InputLayer, 8
- Layer, 9
  - fire, 10
  - getInput, 11
  - getNbNeurones, 11
  - getNeurone, 11
  - Layer, 10
  - printWeight, 11
- learn
  - Neurone, 16
  - NeuroneB, 19
  - Reseau, 24
- Neurone, 13
  - aleaWeights, 14
  - derive\_activate, 14
  - fire, 15
  - fw\_activate, 15
  - fw\_sum, 15
  - getNbPoids, 16
  - getWeight, 16
  - learn, 16
  - Neurone, 14
  - printWeight, 16
  - NeuroneB, 17

- aleaWeights, 18
- fw\_sum, 18
- getNbPoids, 18
- learn, 19
- NeuroneB, 18
- printWeight, 19
- Option, 19
  - getOptionID, 20
  - getOptionInt, 20
  - getOptionRac, 20
  - getOptionType, 20
  - Option, 20
  - print, 20
  - setOptionInt, 21
  - setOptionRac, 21
  - setOptionType, 21
- OptionTab, 21
  - addOption, 22
  - getArgument, 22
  - getIntitul, 22
  - getOptionID, 22
  - getRaccour, 23
  - OptionTab, 22
  - printOptions, 23
- print
  - Option, 20
  - Type, 27
- printOptions
  - OptionTab, 23
- printWeight
  - Layer, 11
  - Neurone, 16
  - NeuroneB, 19
  - Reseau, 24
- Reseau, 23
  - backPropagation, 24
  - fire\_all, 24
  - learn, 24
  - printWeight, 24
  - Reseau, 23
- setFonctionActivation
  - FonctionActivation, 7
- setOptionInt
  - Option, 21
- setOptionRac
  - Option, 21
- setOptionType
  - Option, 21
- setType
  - Type, 27
- Settings, 25
  - getArchi, 25
  - getDifferentOutputs, 25
  - getNbrHiddenLayers, 25
  - Settings, 25
- Type, 26
  - getType, 26
  - print, 27
  - setType, 27
  - Type, 26