Neuron-Matrix Documentation

Release 1.0

Hadrien Renaud-Lebret, Guillaume Bressan, Pierre Browne

Feb 10, 2017

CONTENTS:

CHAPTER

ONE

NEURALNETWORK MODULE

Python module for NeuralNetwork class.

It provides an implementation of a NeuralNetwork with utilitaries. It is NOT bounded to learning on images or even to learning on samples in different files.

class `neuralnet.`**`NeuralNetwork`**(*geometry*, *function=<ufunc 'tanh'>*, *function_derivate=<function inv_cosh>*, *logistic_function_param=(1, 0)*, *learning_factor=0.1*, *momentum=0*)

NeuralNetwork class.

`__call__`(*input_values*)

Apply the Neural Network to the input values.

Warning DOESN'T SAVE the result for a learning after. Use *`apply()`* in this case.

Parameters **`input_values`** – as an iterable of numeric values between 0 and 1.

`__init__`(*geometry*, *function=<ufunc 'tanh'>*, *function_derivate=<function inv_cosh>*, *logistic_function_param=(1, 0)*, *learning_factor=0.1*, *momentum=0*)

Initialisation of the NeuralNetwork.

Parameters

- **`geometry`** (*`str`*) – string describing the format of the NeuralNetwork: '456:12:24:3' will create a network with a first layer with 456 neurons, a second with 12, a third with 24 and the last with 3.
- **`function`** – vectorized function (see numpy.vectorize)
- **`function_derivate`** – its (vectorized) function
- **`logistic_function_param`** (*`tuple`*) – (mu, x0) parameters send to :iso_fonction: and :deri_iso_fonction: slope and offset of the logistic function.

`apply`(*input_values*)

Apply the NeuralNetwork to the input values.

Parameters **`input_values`** – as an iterable of numeric values between 0 and 1.

`backpropagation`(*expected_output*)

Apply the backpropagation algorithm.

`dist`(*expected_output*)

Calc the distance of the result to the expected_output.

`get_geometry`()

Return self.geometry.

Returns self.geometry modified to render like the one passed as an argument of *`__init__()`*.

Return type str

learn(*sample*, *results*, *limit_iterations=50*, *maximal_distance=0.25*)
Learning algorithm on the given examples.

First algorithm.

learn2(*sample*, *results*, *limit_iterations=50*, *maximal_distance=0.25*)
Learning algorithm on the given examples.

Method given by Hélène Milhem here.

randomize_factors()
Randomize the transition matrix.

set_transition_matrix(*matrixes*)
Set the transition_matrix to the correct values.

to_json()
Return an expression of the NeuralNetwork in json.

neuralnet.**deri_iso_fonction**(*fonction*, *mu=1*, *x0=0*)
Creator of fonctions, affinely translated, for derivative.

Compute a isometric transform of fonction with the formulae :

$\forall x, \mathrm{deri_iso_fonction}(f)(x) = \mu \times f(\mu \times x + x_0)$

Which is equivalent to, if f is the derivative of F:

$\forall x, \mathrm{deri_iso_fonction}(f, \mu, x_0)(x) = \frac{d}{dx}(\mathrm{iso_fonction(F, \mu, x_0))(x) = \mu \times \frac{dF}{dx}(\mu \times x + x_0) = \mu \times f(\mu \times x + x_0)})$

Note return the derivate of *iso_fonction()*

Parameters

- **mu** (*float*) – mutliplicative factor μ
- **x0** (*floatx0*) – additive term x_0
- **fonction** – function taking 1 numeric positionnal argument.

Returns $F : x \to \mu \times f(\mu \times x + x_0)$

neuralnet.**inv_cosh**(*x*)
Return $\frac{1}{\cosh(x)}$.

neuralnet.**iso_fonction**(*fonction*, *mu=1*, *x0=0*)
Creator of fonctions, linearly translated.

Compute a isometric transform of fonction with the formulae : :math: *forall x, mathrm{iso_fonction}(f)(x) = f(mu times x + x_0)*

Parameters

- **mu** (*float*) – mutliplicative factor μ
- **x0** (*floatx0*) – additive term x_0
- **fonction** – function taking 1 numeric positionnal argument.

Returns $F : x \to f(\mu \times x + x_0)$

neuralnet.**learning_progress_display**(***args*)
Display the progress of the learning algorithm.

`neuralnet.`**`alphabet`**

Alphabet used. The order is the most important thing. Only the n first values are considered, where n is the length of the alphabet.

`neuralnet.`**`default_values`**

CHAPTER

TWO

PROCESSSING UTILITARIES

2.1 getdata module

Module for neuron-matrix.

class `getdata.IteratorMultiple`(lengths)

Bases: `object`

Iterator through an unknown number of lists.

`getdata.get_data`(proc='1', ranges={'maximal_distance': [0.2], 'momentum': [0.5], 'limit_iterations': [50], 'learning_factor': [0.1], 'learning_algo': ['default']}, **kwargs)

Process with different parameters the sample.

`getdata.procedure1`(alphabet='abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789, ?;.:!éàè\'()+-"=', learning_algo='default', learning_directory='LearningSample', testing_directory='TestSample', learning_factor=0.1, momentum=0.5, limit_iterations=50, maximal_distance=0.2)

Function that execute procedure1.

It does: - create a new NeuralNetwork object - randomize its transition_matrix - learn on a given dataset - test on a given dataset

`getdata.default_ranges`

Ranges on which it iterate by default.

2.2 fileio module

Python module for neuron-matrix.

It implements the main input/output functions used in neuron-matrix.

`fileio.find_examples`(directory, alphabet)

Iterate through the directory to find all processable examples and return them.

`fileio.is_convertible_to_float`(string)

Function that determine if a string can be safely convert to a float.

`fileio.learn_on_folder`(neurnet, directory, alphabet, learning_algo='default', **args)

Make neurnet learn on every example in the directory.

`fileio.read_sample`(file_text)

Function reading an sample in a file and returning the corresponding matrix.

Return the result as a numpy array.

fileio.**save_image**(*matrix*, *file_name*)

Function saving matrix as an image.

fileio.**test_on_folder**(*neurnet*, *directory*, *alphabet*, ***args*)

Make neurnet test every example in the directory.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX