RPROP

1.0

Generated by Doxygen 1.8.8

Mon May 25 2015 20:17:06

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Data Structure Documentation

3.1 ann_t Struct Reference

Struct representing a neural network.

Data Fields

- double **x** [Nx]
- double **y** [Nx]
- double **delta** [Nx]
- double **prevGrad** [Nx][Nx]
- double **currGrad** [Nx][Nx]
- double **updateValue** [Nx][Nx]
- double **wDelta** [Nx][Nx]
- double **w** [Nx][Nx]
- double **dv** [**Nou**]

3.1.1 Detailed Description

Struct representing a neural network.

Container for a complex MLP structure.

3.1.2 Field Documentation

3.1.2.1 double currGrad[Nx][Nx]

Weight current error gradient

3.1.2.2 double delta[Nx]

Delta value on neurons

3.1.2.3 double dv[Nou]

Target value on output neurons

3.1.2.4 double prevGrad[Nx][Nx]

Weight error gradient from last step

3.1.2.5 double updateValue[Nx][Nx]

Update value according to RPROP algorithm for each weight

3.1.2.6 double w[Nx][Nx]

Weights matrix

3.1.2.7 double wDelta[Nx][Nx]

Delta of weights change

3.1.2.8 double x[Nx]

Input of neurons

3.1.2.9 double y[Nx]

Output of neurons

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

- **rprop.c**

Chapter 4

File Documentation

4.1 _rprop.c File Reference

File containing python wrapper for a C MLP with RPROP.

```
#include <Python.h>
#include <numpy/arrayobject.h>
#include "rprop.c"
```

Functions

- static PyObject ∗ **rprop_learn2** (PyObject ∗self, PyObject ∗args)

  Function wrapping a learning process of a neural network.
- static PyObject ∗ **rprop_run2** (PyObject ∗self, PyObject ∗args)

  Function wrapping a function that runs a neural network forward.
- static PyObject ∗ **rprop_init** (PyObject ∗self, PyObject ∗args)

  Function wrapping a initialization of a network.
- PyMODINIT_FUNC **init_rprop** (void)

  Initializes python wrapping functions.

Variables

- static PyMethodDef **module_methods** [ ]

4.1.1 Detailed Description

File containing python wrapper for a C MLP with RPROP.

Author

Jan Gamec

Date

24 May 2015 This file serves as a wrapper for functionality in **rprop.c** (p. 9) . After building this script, it can be imported as a standalone python module. The module can be build by following command: python setup.py build_ext –inplace

The setup.py file **is required** to be in the same directory as this script!. In order to change configuration of network, please see documentation for **rprop.c** (p. 9) file.

See also

rprop.c (p. 9)

4.1.2 Function Documentation

4.1.2.1 static PyObject ∗ rprop_init (PyObject ∗ *self,* PyObject ∗ *args*) `[static]`

Function wrapping a initialization of a network.

Function in python has no input arguments. It just creates new network with random weights initialization and return this weights as a numpy array.

Parameters

self	Object pointer
args	Holds all arguments that can be passed to function in python

Returns

A numpy array holding new weights of the created network

4.1.2.2 static PyObject ∗ rprop_learn2 (PyObject ∗ *self,* PyObject ∗ *args*) `[static]`

Function wrapping a learning process of a neural network.

Function in python accepts following arguments:

- **num_of_epochs** Number of training epochs
- **patternSet** A numpy array of training patterns
- **weights** A numpy array of neural network weights

  Parameters

self	Object pointer
args	Holds all arguments that can be passed to function in python

  Returns

  A numpy array holding new weights

4.1.2.3 static PyObject ∗ rprop_run2 (PyObject ∗ *self,* PyObject ∗ *args*) `[static]`

Function wrapping a function that runs a neural network forward.

It runs network forward and return values on output neurons. Function in python accepts following arguments:

- **pattern** A numpy array representing one input pattern
- **weights** A numpy array of neural network weights

  Parameters

self	Object pointer
args	Holds all arguments that can be passed to function in python

  Returns

  An Q-value on the output neuron with double precission

4.1.3 Variable Documentation

4.1.3.1 PyMethodDef module_methods[ ] [static]

Initial value:

```
= {
  {"learn", rprop_learn2, METH_VARARGS, learn_docstring},
  {"run", rprop_run2, METH_VARARGS, run_docstring},
  {"init", rprop_init, METH_VARARGS, init_docstring},
  {NULL, NULL, 0, NULL}
}
```

Definition of all functions available after the build.

4.2 rprop.c File Reference

File containing implementation of MLP with RPROP learning.

```
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/timeb.h>
```

Data Structures

- struct **ann_t**

  Struct representing a neural network.

Macros

- #define **Nin** 5
- #define **Nh1** 10
- #define **Nh2** 10
- #define **Nou** 1

Functions

- int **sign** (double x)

  Returns the sign of given double.
- void **shuffle** (double ∗∗array, int n)

  Shuffles the given 2D array.
- void **ann_initRprop** (**ann_t** ∗ann)

  Rprop variables initialization. Current weights gradient is set to 0 and udpate value to 0.1.
- void **ann_resetDelta** (**ann_t** ∗ann)

  Resets all delta values on neurons.
- void **ann_rndinit** (**ann_t** ∗ann, double min, double max)

  Randomly initializes weights matrix withit given interval.
- void **ann_init** (**ann_t** ∗ann, double ∗∗weights)

  Initializes a network from a give weights matrix.
- static void **layer_run** (blk_t(ann))

  Calculates an output on one layer using output from previous.

- void **MLP2_run** (**ann_t** ∗ann)

  Simple runs of network in a forward direction Calculate output running whole network forward.
- void **calculate_gradients** (blk_t(ann), int out)

  Calculate weights gradients between 2 layers.
- double ∗ **rprop_run** (**ann_t** ∗ann, double ∗pattern)

  Runs a network in a forward direction with a given input pattern Calculate output running whole network forward.
- void **rprop_update** (blk_t(ann))

  Implementation of RPROP learning algorithm according to paper Update rules and equations are described in work. This function updates weights between 2 layers.
- void **rprop_learning_step** (**ann_t** ∗ann, int num_of_patterns, double ∗∗patternSet)

  RPROP learning step. Implementation of RPROP algorithm according to paper. This method makes one forward run throught network calculating learning variables and updating weights after then.
- void **test_net** (**ann_t** ∗ann, int num_of_patterns, double ∗∗patternSet)

  Tests a network for an error against training set.
- void **rprop_learn** (**ann_t** ∗ann, int num_of_epochs, int num_of_patterns, double ∗∗patternSet)

  Manages a learning process Repeats learning procedure for the given number of epochs and shuffles the training set. It tests the network after then.

4.2.1 Detailed Description

File containing implementation of MLP with RPROP learning.

Author

Jan Gamec

Date

24 May 2015 This module contain functions for initialization, running and training Multilayer Perceptron with RPROP learning algorithm. This file cannot be run independently, but is used as a library for python wrapper. File can be combiled: gcc -Wall -std=gnu99 -O3 -ffast-math -funroll-loops -s -o rprop_standalone **rprop.c** (p. 9) -lm

4.2.2 Macro Definition Documentation

4.2.2.1 #define Nh1 10

Defines number of neurons in first hidden layer. This needs to be changed according to task.

4.2.2.2 #define Nh2 10

Defines number of neurons in second hidden layer. This needs to be changed according to task.

4.2.2.3 #define Nin 5

Defines number of input neurons. This needs to be changed according to task.

4.2.2.4 #define Nou 1

Defines number of output neurons. This needs to be changed according to task.

4.2.3 Function Documentation

4.2.3.1 void ann_init (ann_t $*$ *ann,* double $**$ *weights*)

Initializes a network from a give weights matrix.

Parameters

in,out	*ann*	Neural network structure
	weights	Initializatioon weights matrix

4.2.3.2 void ann_initRprop (ann_t ∗ *ann*)

Rprop variables initialization. Current weights gradient is set to 0 and udpate value to 0.1.

Parameters

in,out	*ann*	Neural network structure

4.2.3.3 void ann_resetDelta (ann_t ∗ *ann*)

Resets all delta values on neurons.

Parameters

in,out	*ann*	Neural network structure

4.2.3.4 void ann_rndinit (ann_t ∗ *ann,* double *min,* double *max*)

Randomly initializes weights matrix withit given interval.

Parameters

in,out	*ann*	Neural network structure
	min	Bottom weight bound
	max	Upper weight bound

4.2.3.5 void calculate_gradients (blk_t(ann) , int *out*)

Calculate weights gradients between 2 layers.

Parameters

blk_t(ann)	Macro representing separated 2 layers
out	Signalizes whether the layer is hidden or output/input

4.2.3.6 static void layer_run (blk_t(ann)) [static]

Calculates an output on one layer using output from previous.

Parameters

in,out	*blk_t(ann)*	Macro separating 2 layers from ann structure

4.2.3.7 void MLP2_run (ann_t ∗ *ann*)

Simple runs of network in a forward direction Calculate output running whole network forward.

Parameters

in,out	*ann*	Neural network structure

4.2.3.8 void rprop_learn (ann_t ∗ *ann,* int *num_of_epochs,* int *num_of_patterns,* double ∗∗ *patternSet*)

Manages a learning process Repeats learning procedure for the given number of epochs and shuffles the training set. It tests the network after then.

Parameters

in,out	*ann*	Neural network structure
	num_of_epochs	Number of training epochs
	num_of_patterns	Number of training patterns
	patternSet	Training set represented by a 2D array

4.2.3.9 void rprop_learning_step (ann_t ∗ *ann,* int *num_of_patterns,* double ∗∗ *patternSet*)

RPROP learning step. Implementation of RPROP algorithm according to paper. This method makes one forward run throught network calculating learning variables and updating weights after then.

Parameters

in,out	*ann*	Neural network structure

4.2.3.10 double∗ rprop_run (ann_t ∗ *ann,* double ∗ *pattern*)

Runs a network in a forward direction with a given input pattern Calculate output running whole network forward.

Parameters

in,out	*ann*	Neural network structure
	pattern	Training pattern

Returns

Vector of values on the output neurons

4.2.3.11 void rprop_update (blk_t(ann))

Implementation of RPROP learning algorithm according to paper Update rules and equations are described in work. This function updates weights between 2 layers.

Parameters

blk_t(ann)	Macro separating 2 following layers

4.2.3.12 void shuffle (double ∗∗ *array,* int *n*)

Shuffles the given 2D array.

Parameters

in,out	*array*	Array to be shuffled
	n	length of an array

4.2.3.13 int sign (double *x*)

Returns the sign of given double.

Parameters

x	Double precission number

4.2.3.14 void test_net (ann_t $*$ *ann,* int *num_of_patterns,* double $**$ *patternSet*)

Tests a network for an error against training set.

Parameters

in,out	*ann*	Neural network structure
	num_of_pattern	Number of pattern in training set
	patternSet	Training set represented by 2D array