Network manual

Gabriele Pernici

December 2023

1 Introduction

This document contains the info on how to use the network program used in *insert thesis name*. Section 2 describes how to correctly generate a command file, in order to use the program. Section 3 describes in details how the program acts and the meaning of each parameter. Section 4 describes the format of the files correlated to the program. More specifically the examples file, the network's structure Json file and the output file are described.

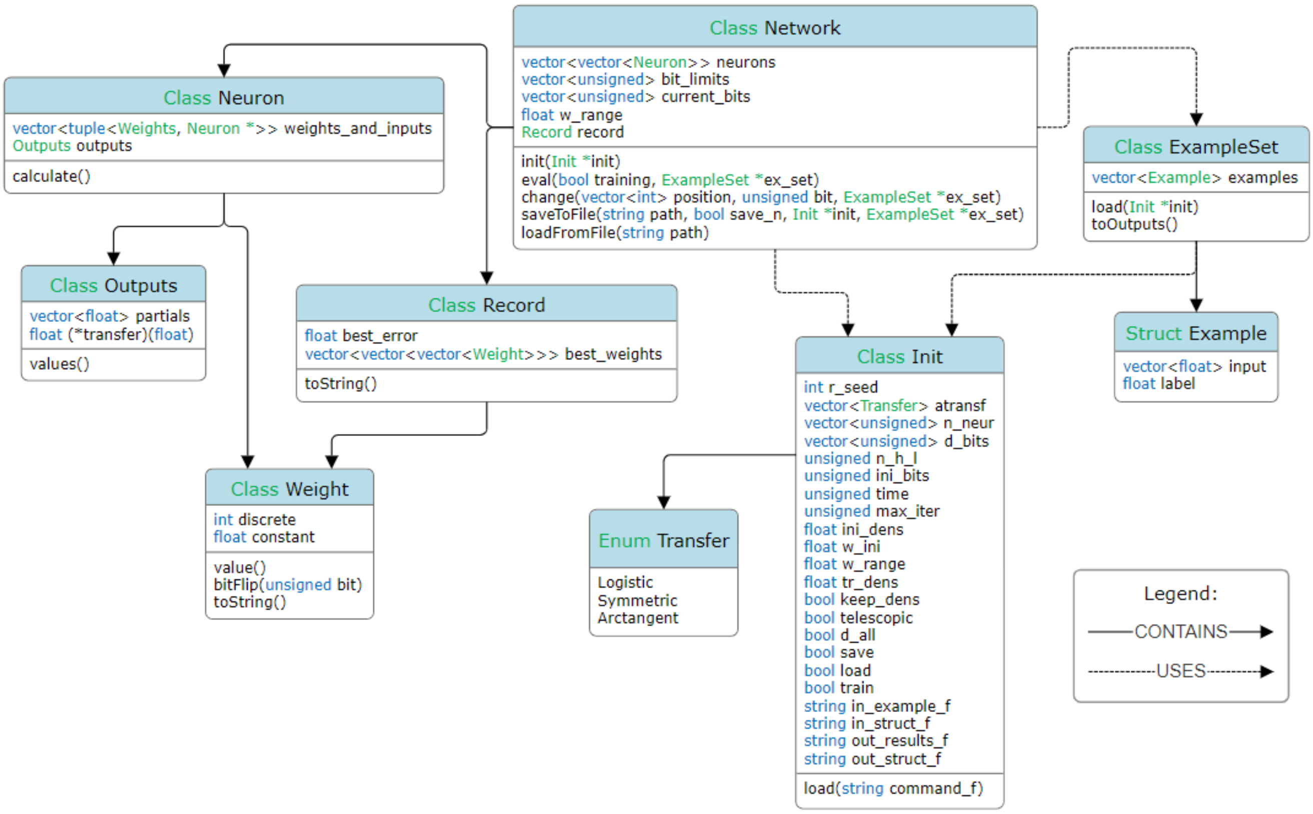

Figure 1: This image shows the network's diagram of implementation.

2 Command file

The command file should be a text file (preferably with extension .cmd) with a combination of the following parameters. Each parameter to be valid has to be inserted as *ParameterName ParameterValue1 ... ParameterValueN*. Each parameter should be on a different line of the file, however the order is not important. Every required parameter is in bald (if loading a network from file the only needed parameters are in italic) and every optional parameter has a default value. Moreover, for each parameter the number of values needed is stated. If a parameter has zero values required it means that the parameter name's presence is enough. Lastly, the type of value required is specified.

Name	**nVal**	**Type**	**Def**	**Description**
r_seed	1	N	1	Random seed to generate the initial weights
n_h_l	1	N		Number of hidden layers in the network
n_neur	n	N		Number of neuron per hidden layer
atransf	n	0,1,2	0	Type of transfer function of each layer. Each number meaning is described in section 3
d_bits	n	N		Max number of bits to use for discrete weights in each layer
ini_bits	1	N	d_bits	Number of bits to start discretization from
time	1	N	∞	Time limit for training
max_iter	1	N	∞	Iterations limit for training
ini_dens	1	0<x<1	1	Fraction of nonzero weight in the initialization phase
w_ini	1	x>0		Weight range in initialization phase
w_range	1	x>0		Weight range in training phase
tr_dens	1	0<x<1	1	Fraction of examples used in training
keep_dens	0			If present the program tries to keep the value of ini_dens during training
telescopic	0			If present the program uses the telescopic mode during training
d_all	0			If present the program looks for the best improving move during training
save	0			If present the program saves the network's structure after training
load	0			If present the program loads the netowrk structure from in_struct_f
train	0			If present the program trains the network, otherwise it's evaluated
in_example_f	1	String		Path of the examples file. Always needed
in_struct_f	1	String		Path of the network's structure file
out_results_f	1	String	./out.exa	Path where the results are saved
out_struct_f	1	String	./net.json	Path where the network's structure is saved

3 Program functionalities

3.1 Transfer functions details

As stated in the previous section, the ***atransf*** parameter can assume 3 values that each represent a different type of transfer function. 0 represents the standard logistic function, 1 represents the symmetric logistic function (aka the hyperbolic tangent) and 2 represents the arctangent function.

3.2 Program operations

The program's main functionalities are to train or evaluate the specified network. The ***train*** parameter in the command file specifies which of the two the program will do. Moreover, the program can save the network's structure after the end of the main operations, using the ***save*** parameter. This is fundamental in order to save the weights achieved during training, or to stop (using the iterations or the time limit) a training process and continue it later, without having to start from scratch. Lastly, the program can load a previously saved network in order to evaluate, or keep training it, by using the ***load*** parameter. When doing so the program file is greatly shortened as the only needed parameters are: *load, in_example_f, in_struct_f.* The following parameters are optional and their value, if not present, will be set either accordingly to the json file (where present) or as their default: *d_bits, time, max_iter, tr_dens, save, train, out_results_f, out_struct_f.* All other parameters are useless and, if present, will be ignored. Any combination of the *train, save, load* parameters is accepted.

4 Files

4.1 Sample/Output file

The suggested filename extension is .exa. In some cases, output values are not used (e.g., when evaluating a trained network). In such case, they can be set to zero. Files containing the examples or the outputs have the following format:

n_inp n_{inputs}							
n_out $n_{outputs}$							
n_patt $n_{patterns}$							
pattern x_1	x_2	$\dots$	$x_{n_{inputs}}$	y_1	y_2	$\dots$	$y_{n_{outputs}}$
...							
pattern x_1	x_2	$\dots$	$x_{n_{inputs}}$	y_1	y_2	$\dots$	$y_{n_{outputs}}$

4.2 Network structure file

The following is the format of the network's structure json file. The first layer of neurons is not saved (as can be seen by the first ”pos” parameter) as it is used to represent the input examples, so it's useless to save. In the file are saved the info regarding the network's structure and training commands (that will be ignored in case of evaluation).

```
{
    ”neurons”: [
        {
            ”pos”:[1,0],
            ”weights”: [
                {
                    ”index”:0,
                    ”discrete”:value,
                    ”constant”:value
                },

                ...

                {
                    ”index”:n,
                    ”discrete”:value,
                    ”constant”:value
                }
            ]
        },

        ...

        {
            ”pos”:[x,y],
            ”weights”: [
                {
                    ”index”:0,
                    ”discrete”:value,
                    ”constant”:value
                },

                ...

                {
                    ”index”:n,
                    ”discrete”:value,
                    ”constant”:value
                }
            ]
        }
    ],
    ”bit_limits”: [values],
    ”current_bits”: [values],
    ”w_range”:value,
    ”atransf”:[values],
    ”commands”: {
        ”keep_dens”:boolean,
        ”telescopic”:boolean,
        ”d_all”:boolean
    }
}
```