NeuroMZ Tutorial

The neural network manager Neuromz helps by creating and training Artificial Neural Network in the simplest way.

Audience

This tutorial has been prepared for computer science students who are interesting in ANN (Artificial Neural Network) and AI (Artificial Intelligence).

Prerequisites

Before proceeding with this tutorial, it’s good to have a basic understanding of Artificial Neural Network.

Copyright

Table of Contents

Why NeuroMZ

NeuroMZ is created to have the simplest form and best performance without high latency.
NeuroMZ helps to test Artificial Neural Network and to do complex Artificial Intelligence programs.

Creating Neural Network

To create a neural network in NeuroMZ just type *new* (or *-new* by argument) followed by size of each layer (count of neurons in each layer).

Example:
to build a neural network with 7 inputs and 2 hidden layer (4 and 5 nodes) and 1 output layer :

new 7 4 5 1

or passing by argument:

neuromz -new 7 4 5 1

NOTES:

All layers have the sigmoid as an activation function.
A new neural network built inside the program will have no filename and won't be saved.
Building it by argument will save it automatically, but using this command will an return error because no filename was declared. You will learn how to solve this problem in the saving neural network part.
A layer's size should be an integer greater than zero.

Load Saved Neural Network

To load a neural network that exists in the hard disk you should type *load* (or *-file* by argument) followed by the filename.

Example:
to load myfile

load myfile

or by argument:

neuromz -file myfile

NOTES:

loading file by argument without any operation will do nothing because after loadind it in memory no other operation will be applied.

Save Neural Network

To save a neural network just type *save* (or *-name* by argument) followed by the name of your file.

Example:

save myfile

or by argument:

neuromz -name myfile

but we just have filename without a neural network so this will return error.
So by argument we can create new network with name:

neuromz -new 7 4 5 3 -name myfile

NOTES:

Saving without a existing neural network will return an error.

If you load a neural network and have a filename, just type save to save the changes of the current network.

While working by argument the file will be saved when it's necessary.

We can save a file by a different name if we want.

Clear Neural Network

To clear or close a neural network from the memory by just type *clear*.

NOTES:

While using arguments, there's no need for the clear command because it will be cleared automatically before the program end its task.

A new neural network can't be opened or created if we don't clear the current network.

Test Neural Network

Testing a neural network means that a neural network is ready to its task. It takes the input, that needs to be calculated, provided

that the number of input values equal to the input layer nodes. The program will return the output values equal to the number of output layer's nodes.
To do the test just type *test* (or *-test* by argument) followed by the input values.

Example:

test 2.4 4 0.1 7

or by arguments:

neuromz -file myfile -test 2.4 4 0.1 7

NOTES:

Testing without a file will return error.

Learning Rate

Learning rate is a value between 0 and 1, it helps to specify the speed of each change of the weights inside the neural network. To change the value of the learning rate just type *learnrate* (or *-lr* by argument) followed by the new value for the learning rate.

Example:

learnrate 0.1

or by argument:

neuromz -file myfile -lr 0.1

NOTES:

Choosing a learning rate near to one will make it have better value but it becomes slower to train.
Choosing a learning rate near to zero will make it have better values but it become slower to train.

Converge value

It’s a value used by training property that indicates the end of the training, the training ends when the cost function is smaller the the converge value.
The cost function is the summation of the square power of the difference between target values for output and the calculated ones. Square being used to keep the values positive.
To change the converge value just type *conv* (or *-conv* by argument) followed by the new value.

Example:

conv 0.01

or by argument:

neuromz -file myfile -conv 0.01

NOTES:

Converge value should be positive less than 1 but not zero.

Train Neural Network

To train a neural network in NeuroMZ we have two methods:

-By training set.

-By counting the steps of trains.

Training set method:

is to have many training values simultaneously till all are correct at the same time.

This is better to use it for the basics when starting a new network. The neural network will train till the cost function is smaller than the converge value.

To training the network by training sets just type *train* (or *-train* by argument) followed by the input values where the number of the input values should be equal to the number of the input layer size. Then type *-tar* as a target flag followed by output values, where the target values should be equal to the number of the output layer size.

Example:

train 2.4 0.3 -tar 0.3

or by argument:

neuromz -train 2.4 0.3 -tar 0.3

Counting method:

It’s used when done training for the basics. It’s training the network at a specified number.

To train the neural network by the counting method, it's the same as the training sets method, with adding *rn* at the end(*n* is the number of steps).

Example:
training 20 times

train 2.4 0.5 -tar 0.3 r20

or by argument:

neuromz -train 2.4 0.5 -tar 0.3 r20

NOTES:

Every time the training set is used it will saved to the file to be used by all the other training sets.

Training by the counting method will not save any data to the training sets.

Remove Training Sets

For several reasons, the training sets might be need to be removed:

-To reduce the size of the file.

-When the neural network is trained with wrong values and needs to be reset, since all the training set are saved.

To remove the training sets just type rmset (or *-rmset* by argument).

Example:

rmset

or by argument:

neuromz -file myfile -rmset

NOTES:

It is recommended to repeat the basic training after removing the training sets, otherwise the basics might be lost.

Additional Option In NeuroMZ

show:

to show the details of the neural network, the nodes count of the input layer, and the output layer and the count of hidden layers.
And print the value of the converge and learning rate.
Keyword *show* (or *-show* by argument).

Version:

to show the version of the NeuroMZ.
Keyword *version* (or *-version* by argument).
Or just type *-v* (by command or argument).

Help:

to print help of the program:
Keyword *help* (or *--help* by argument).
Or just *-h* (by command or argument).

And can get help in general or help [command],
as help train

Exit:

to exit from command program.
Just type *exit*.
This can’t be used by argument, since the program automatically exits when finished.