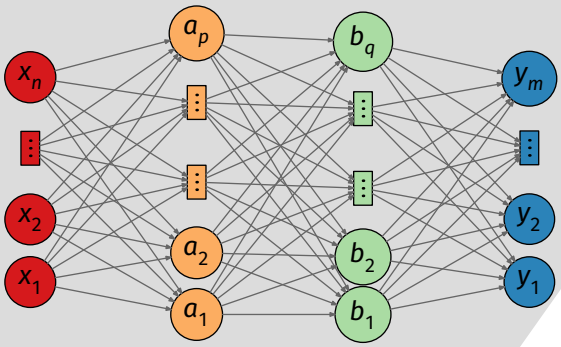
mp-neuralnetwork

drawing artificial neural networks
with METAPOST and METAOBJ



Contributor
Maxime CHUPIN
notezik@gmail.com

Version 0.1, 2025, January, 27th
https://gitlab.gutenberg-asso.fr/mchupin/mp-neuralnetwork

Abstract

This METAPOST package allows to draw artificial neural networks. It is based on the METAOBJ package which provides many tools to draw and arrange nodes.

https://gitlab.gutenberg-asso.fr/mchupin/mp-neuralnetwork

Contents

This package is in beta version—do not hesitate to report bugs, as well as requests for improvement.

1 Installation

mp-neuralnetwork is on CTAN and can also be installed via the package manager of your distribution.

https://www.ctan.org/pkg/mp-neuralnetwork

1.1 With TEXlive under Linux or macOS

To install mp-neuralnetwork with TEXLive, you will have to create the directory texmf in your home.

```
user $> mkdir ~/texmf
```

Then, you will have to place the neuralnetwork.mp and metaobj-patch.mp[1] files in

[1] Thanks to Denis Roegel, this file define a Group object to group different METAOBJ objects.

```
~/texmf/metapost/mp-neuralnetwork/
```

Once this is done, `mp-neuralnetwork` will be loaded with the classic METAPOST input code

```
input mp-neuralnetwork
```

1.2 With MikTEX and Windows

These two systems are unknown to the author of `mp-neuralnetwork`, so we refer you to the MikTEXdocumentation concerning the addition of local packages:

`http://docs.miktex.org/manual/localadditions.html`

1.3 Dependencies

`mp-neuralnetwork` depends:

- of course on METAPOST [3];
- on the packagemetaobj [2];
- if `mp-neuralnetwork` is not used with LuaLATEX and the luamplib package, on the latexmp package[1];
- and on the package colorbrewer-rgb [4].

2 Main Command

The package `mp-neuralnetwork` provides one principal command that has two possible invocations.

2.1 Finite network

`drawANN(`⟨*name1*⟩`,`⟨*size1*⟩`,`⟨*name2*⟩`,`⟨*size2*⟩`, etc.)`

Each layer of the network is described by a couple ⟨*name*⟩ and ⟨*size*⟩, where:

⟨***name***⟩**:** is the names of the layers. It a string, and this is the prefix of each node (for instance `"x"` will produce nodes x_i).

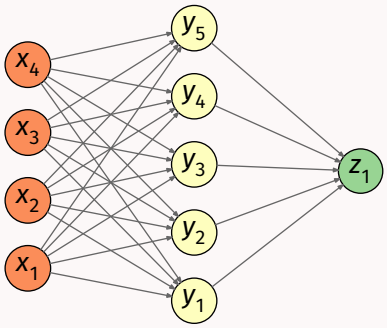
⟨***size***⟩**:** is the size (integer) of the layer (for instance if ⟨*size*⟩= 4, the layer will be composed by nodes x_1, x_2, x_3 and x_4).

Here, a simple example.

Exemple 1

```
input mp-neuralnetwork
```

```
beginfig(1);
drawANN("x",4,"y",5,"z",1);
endfig;
```



2.2 Abstract network

You can set variables for the size of each layers with the same command using the string notation for the size instead of an integer.

`drawANN(`⟨*name1*⟩`,`⟨*size1*⟩`,`⟨*name2*⟩`,`⟨*size2*⟩`, etc.)`

Each layer of the network is described by a couple ⟨*name*⟩ and ⟨*size*⟩, where:

⟨***name***⟩**:** is the names of the layers. It a string, and this is the prefix of each node (for instance `"x"` will produce nodes x_i).

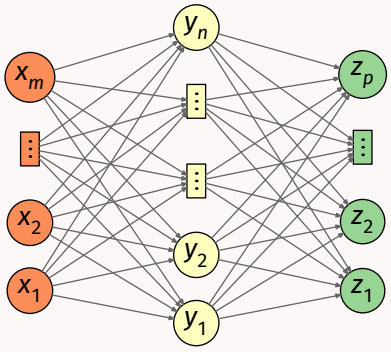
⟨***size***⟩**:** is the notation for size (string) of the layer (for instance `"p"`, the layer will be composed by nodes x_1, x_2, ... and x_p).

Here, a simple example.

Exemple 2

```
input mp-neuralnetwork

beginfig(1);
drawANN("x","m","y","n","z","p");
endfig;
```



The first and the last layers are composed with four nodes, and intern layers with five nodes.

3 Options

`mp-neuralnetwork` allows to customize the network representation. If you want to restore default behaviour, you can use the following command.

`mpneural_init_values`

3.1 Colors

We can see, in the previous examples, that the nodes are colored. If you do not want them to be colored, use the following command.

`colored_layers(⟨boolean⟩)`

⟨***boolean***⟩**:** is a METAPOST boolean (true by default) to choose if we want or not colored layers.

If the layers are colored, `mp-neuralnetwork` uses `Spectral` colors defined in colorbrewer-rgb[2].

You can also define colors for each layer using the following command.

`set_layer_colors(⟨layer number1⟩,⟨color1⟩,⟨layer number 1⟩,⟨color1⟩,etc.)`

⟨***layer number 1***⟩, ⟨***color1***⟩**:** is a couple with the integer of the layer we want to color with ⟨*color1*⟩.

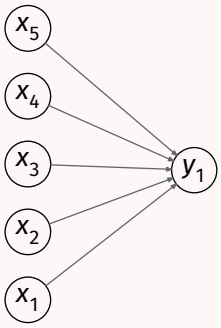
The rest of layers are set to white by default.
For example:

Exemple 3

```
input mp-neuralnetwork

colored_layers(false);
beginfig(1);
drawANN("x",5,"y",1);
endfig;
```

[2]Up to 11 colors. If there are more layers, the colors cycle.



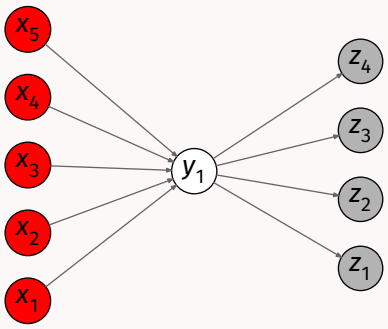
and

Exemple 4

```
input mp-neuralnetwork

colored_layers(true);
set_layer_colors(1,red,3,(0.7,0.7,0.7));

beginfig(1);
drawANN("x",5,"y",1,"z",4);
endfig;
```



Connections are colored. The default color is gray (`0.4*white`) but you can modify that with the following command.

`set_connection_color(⟨color⟩)`

⟨***color***⟩**:** is the color for drawing the connections between layers (default `0.4*white`).

3.2 Annotations

`mp-neuralnetwork` provides some tools to annotate the network. However, because we use METAOBJ behind the scene, you can always use METAOBJ tools to draw above the graph build with `mp-neuralnetwork` (see section 4).

3.2.1 Frames

You can add frames around three different blocks: input, output and hidden layers.

To draw a frame around these three types of layers (or stop the drawing of theses frames), you can use the following commands.

`framed_input(⟨boolean⟩)`

`framed_output(⟨boolean⟩)`

`framed_hidden(⟨boolean⟩)`

⟨***boolean***⟩**:** true or false (default).

3.2.2 Positioning nodes

We can adjust the vertical and horizontal distance between nodes with the following commands.

`set_vspace(⟨distance⟩)`

⟨***distance***⟩**:** is a numeric (default `1.5cm`).

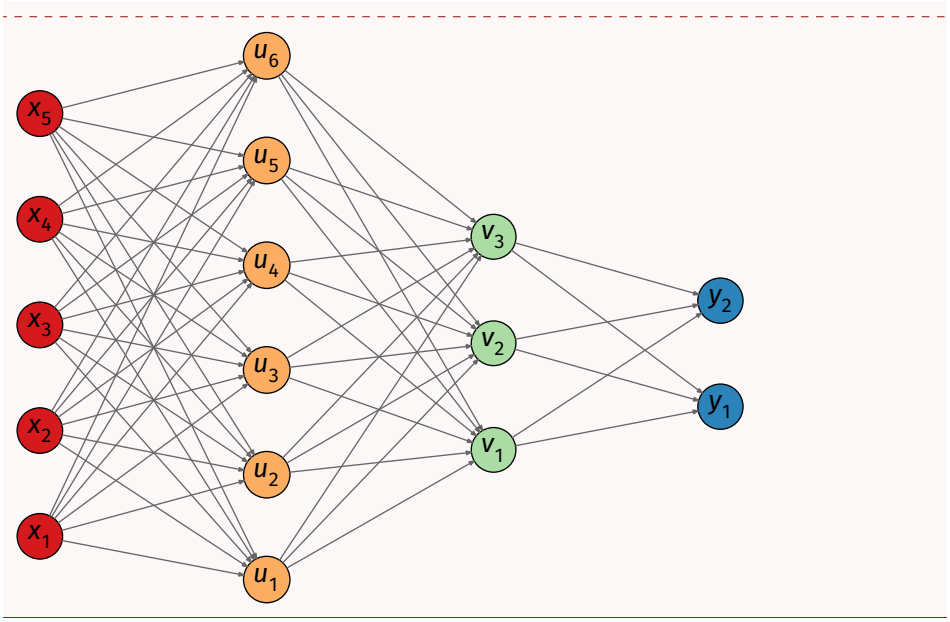
`set_hspace(⟨distance⟩)`

⟨***distance***⟩**:** is a numeric (default `2.2cm`).

Exemple 5

```
input mp-neuralnetwork

beginfig(1);
set_vspace(2cm);
set_hspace(3cm);
drawANN("x",5,"u",6,"v",3,"y",2);
endfig;
```



3.2.3 Labels and sections

First, if you do not want to label the nodes, use the following command to deactivate the function.

`show_node_labels(⟨boolean⟩)`

You can also add legends to these part of the network (independently of the frame draw). For that, you can use the following command.

`print_legends(⟨boolean⟩)`

⟨***boolean***⟩**:** true or false (default).

The default value for the legends of the three parts (input, output, and hidden), are `"Input layer"`, `"Output layer"`, and `"Hidden layer(s)"`. You can modify these string with the following commands.

`set_input_legend(⟨string⟩)`

`set_output_legend(⟨string⟩)`

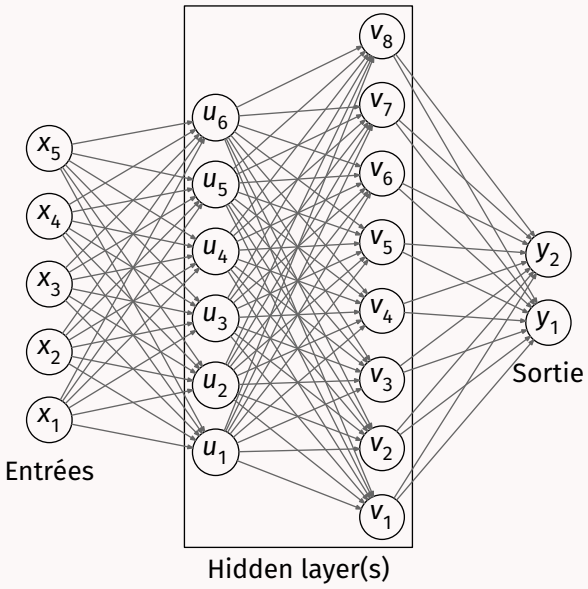
`set_hidden_legend(⟨string⟩)`

⟨***string***⟩**:** is a string for the legend (default are `"Input layer"`, `"Output layer"`, and `"Hidden layer(s)"`).

Exemple 6

```
input mp-neuralnetwork

colored_layers(false);
beginfig(1);
print_legends(true);
set_input_legend("Entrées");
set_output_legend("Sortie");
framed_hidden(true);
drawANN("x",5,"u",6,"v",8,"y",2);
endfig;
```



You may also want to print the *weights* on the network. To do that, use the following commands.

`print_weights(⟨boolean⟩)`

By default, weights are denoted by *w*, but you can change that with the following command.

`set_weights_notation(⟨string⟩)`

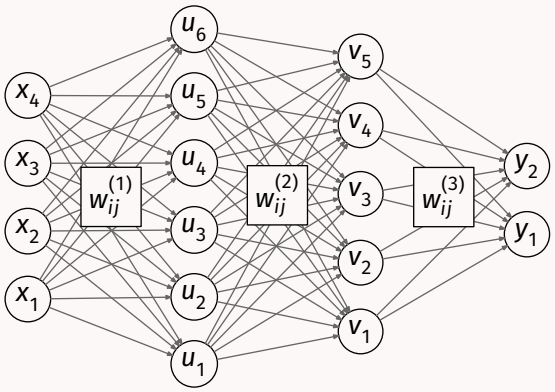
⟨***string***⟩**:** is a string for the weights (default is `"w"`). This is a prefix: for instance, with the default value, the weights are, in math mode, $w_{ij}^{(k)}$ for the *k*-th layer.

Exemple 7

```
input mp-neuralnetwork

colored_layers(false);
```

```
beginfig(1);
print_legends(false);
print_weights(true);
drawANN("x",4,"u",6,"v",5,"y",2);
endfig;
```



You can add labels positioning them with relative coordinates that is $(0, 0)$ at bottom left of the bounding box of the complete graph and $(1, 1)$ at top right. For that, you can use the following command.

`labelANN(`⟨*string*⟩`)(`⟨*xpart*⟩`,`⟨*ypart*⟩`)` ⟨*\MO/ options*⟩

⟨***string***⟩**:** is a string for the label (process by `textext` command).

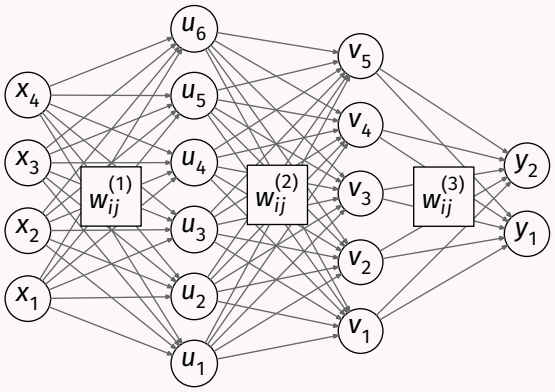
⟨***xpart***⟩,⟨***ypart***⟩**:** are numeric in the relative coordinates (but can be lower or greater than 0 and 1.).

This command build a METAOBJ box. It is the center of the box that is put at the relative coordinate. Moreover, you can add the METAOBJ options at the end to customize the box.

Exemple 8

```
input mp-neuralnetwork

colored_layers(false);
beginfig(1);
print_legends(false);
print_weights(true);
drawANN("x",4,"u",6,"v",5,"y",2);
labelANN("My fabulous network")(0.5,-0.2) "filled(true)","
      fillcolor((0.8,0.5,0.5))";
endfig;
```



My fabulous network

4 With MetaObj

Because `mp-neuralnetwork` is built above METAOBJ [2], the tools of the incredible package are available.

The nodes of the network are METAOBJ boxes, with names:

- with the prefix ANN;
- the instance number of the network (useful if you draw multiple networks);
- an underscore _;
- the identifier of the layer (specified in the `drawANN` command);
- the number of the node in the layer.

The boxes of weights are named as follows:

- with the prefix ANN;
- the instance number of the network (useful if you draw multiple networks);
- an underscore _;
- `weights`;
- the number of the left layer.

The boxes of label for input, output and hidden layers are named as follows:

- with the prefix ANN;
- the instance number of the network (useful if you draw multiple networks);
- an underscore _;
- `input`, `output` or `hidden`.

Labels generated by `labelANN` are also METAOBJ boxes named as follows:

- with the prefix ANN;
- the instance number of the network (useful if you draw multiple networks);
- an underscore _;
- `label`;
- then the number of the label.

For example, if you draw the following network:

```
input mp-neuralnetwork

beginfig(1);
drawANN("x",3,"y",2);
endfig;
```

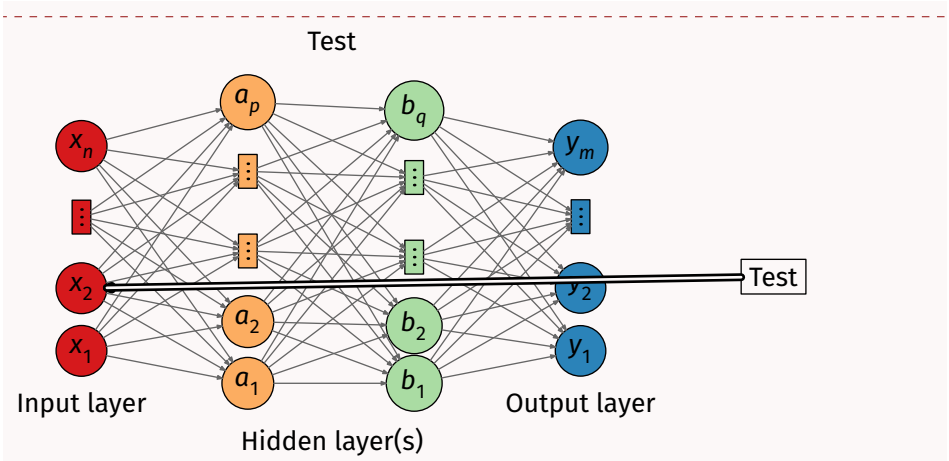
The nodes will be named `ANN1_x1`, `ANN1_x2`, `ANN1_x3`, `ANN1_y1`, and `ANN1_y2`. The weights will be named `ANN1_weights1`.

Here is a real example (this is the 10th instance of a network drawing in this document). `ncline` is a METAOBJ command.

Exemple 9

```
input mp-neuralnetwork;

beginfig(1);
  print_legends(true);
  show_node_labels(true);
  drawANN("x","n","a","p","b","q","y","m") "circmargin(3pt)";
  labelANN("Test")(1.3,0.4);
  labelANN("Test")(0.5,1.1) "framed(false)";
  ncline(ANN10_label2)(ANN10_x2)"doubleline(true)","coilwidth(2
        mm)",
  "angleA(0)",
  "linewidth(1pt)";
endfig;
```



References

[1] Jens-Uwe Morawski. *The `latexMP` package. Interface for LaTeX-based typesetting in MetaPost*. Version 1.2.1. June 21, 2020. URL: https://ctan.org/pkg/latexmp.

[2] Denis B. Roegel. *The `metaobj` package. MetaPost package providing high-level objects*. Version 0.93. June 24, 2016. URL: https://ctan.org/pkg/metaobj.

[3] The MetaPost Team and John Hobby. *The `metapost` package. A development of Metafont for creating graphics*. Aug. 26, 2021. URL: https://ctan.org/pkg/metapost.

[4] Toby Thurston. *The `metapost-colorbrewer` package. An implementation of the colorbrewer2.org colours for MetaPost*. Sept. 25, 2018. URL: https://ctan.org/pkg/metapost-colorbrewer.

Command Index