Freeman's K$_{\mathrm{III}}$
Reference guide

William Q. Ferreira

September 13, 2017

Contents

1 Introduction 3

1.1 Terminology . . . 3

1.1.1 Model . . . 3

1.1.2 Dataset . . . 3

1.1.3 Node . . . 3

1.1.4 Training . . . 3

1.1.5 Simulation . . . 3

1.2 File manipulation . . . 4

1.2.1 Models as files . . . 4

1.2.2 Input & output . . . 4

2 The graphical interface 5

2.1 The layout . . . 5

2.1.1 No model loaded . . . 5

2.1.2 Model overview . . . 5

2.2 Model manipulation . . . 6

2.2.1 Create . . . 6

2.2.2 Save . . . 7

2.2.3 Load . . . 7

2.3 Data processing . . . 8

2.3.1 Train . . . 8

2.3.2 Simulate . . . 9

2.4 Other operations . . . 10

2.4.1 Help . . . 10

3 The text interpreter 11

3.1 Executing commands via text . . . 11

3.2 Setting model parameters . . . 12

4 Error handling 13

4.1 Invalid layer size . . . 13

4.2 Could not write to file . . . 13

4.3 Could not load model from file . . . 13

4.4 Inconsistent dataset width . . . 13

4.5 Dataset width does not match layer 0 size . . . 13

5 Authors 14

1 Introduction

1.1 Terminology

1.1.1 Model

A “model” is a computational representation of the K_{III} set proposed by Freeman’s article “Mass Action in the Neural System”, in 1975.
The model is composed of 3 layers of K_{II} models, referred to throughout the guide as “layer 0” (or “input layer”), “layer 1” and “layer 2”. The “output layer” is usually layer 2, but can be configured otherwise during model creation.

1.1.2 Dataset

A “dataset” is a text file of decimal numbers organized in a matrix-like fashion. The file must be organized according to the `.csv` file format, and all rows must have the same number of elements.
The width of a dataset is the number of elements in a given row of the dataset.

1.1.3 Node

A “node” is a K_{II} set part of a K_{III} model.
An “input node” is a node part of the input layer of a model.
A “layer n node” is a node part of the nth layer of a model.
An “output node” is a node part of the output layer of a model.

1.1.4 Training

“Training” refers to fitting the weights of the model’s connections between nodes to try and adapt to a given dataset.

1.1.5 Simulation

To “simulate” is to create a `.csv` file whose rows are the result of feeding every row of an input dataset into a model. The output file is formatted as a dataset and could be used by another model as is.

1.2 File manipulation

1.2.1 Models as files

The application offers the functionality to save a model's current state into a file for later use. If you have trained a model and wish to keep its weights for future use, you may save the model into a file.
The extension for K_{III} model files is `.k3`.

1.2.2 Input & output

Any input and output is given through datasets.
Datasets are processed line by line, each element of a row being used as a stimulus for a different input node of a given model. That means a model can only accept datasets whose width matches its number of input nodes.
Simulations will always output a dataset whose width matches the number of output nodes in the model used.

2 The graphical interface

2.1 The layout

2.1.1 No model loaded

In this state, the application cannot run most of its functionalities, as there is no model in memory for it to use.
To load a model into memory, you could either create a new one or load an existing one from a file. Both these functions are explained in the "file operations" section.
After either of those operations has been completed, the application will go into the "model overview" state.

2.1.2 Model overview

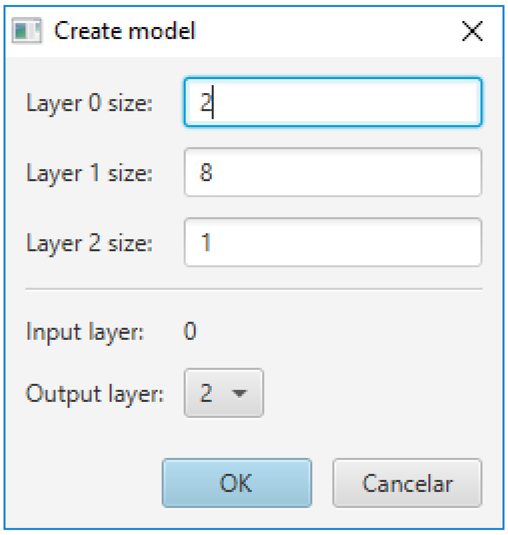
In this state, all functionalities are available for use, and any that require a model will use the one most recently created or loaded.
In the top sector, you can see a representation of the model itself. Each circle represents a K_{II} layer and their respective sizes, while the arrows between them represent the interlayer connections. The leftmost "in" arrow going into layer 0 represents the input. The upwards "out" arrow either coming out of layer 2 or branching off from another arrow represents the output of the model.
For example, if the output layer for a model is set to be layer 1, the "out" arrow is located between layers 1 and 2, and means the output dataset is going to be the state of the layer 1 nodes.
Below this visualization there are two panels of buttons, which are the different functionalities of the application.

2.2 Model manipulation

These are the functionalities present in the lower right corner of the application, and deal with models themselves instead of using them.

2.2.1 Create

This will replace the model currently in use and cannot be undone. If you wish to keep the current model for later use, be sure to save the model into a file before creating a new one.
Replaces the model loaded (if any) with a newly created one.
After selecting this option, a popup window as shown in figure 1 will appear. The three different text fields are the number of nodes in each of the layers in the new K_{III} set. The dropdown list represents the choice of which layer the K_{III} set will consider as the output when run.


Create model

Layer 0 size: 2

Layer 1 size: 8

Layer 2 size: 1

Input layer: 0

Output layer: 2

Figure 1: Model creation popup

See the example of the figure 1. If the user presses the OK button without changing any of the values, they will create a model whose layer 0 has 2 nodes, layer 1 has 8 nodes and layer 2 has 1 node, and the output of the model will be the output of layer 2 at every row of the dataset being processed.

2.2.2 Save

Can only be used if a model is currently loaded.
Stores the model structure and internal connection weights into a file. The file path is selected through a file chooser.

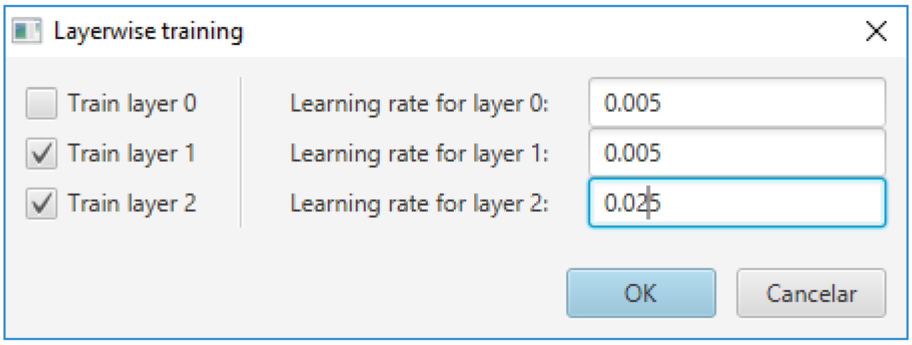
2.2.3 Load

This will replace the model currently in use and cannot be undone. If you wish to keep the current model for later use, be sure to save the model into a file before loading one.
Replaces the model loaded (if any) with one previously saved to a file. You must select the file from which to load via a file chooser.

2.3 Data processing

2.3.1 Train

Can only be used if a model is currently loaded.
While this operation takes place, the application will be completely unresponsive. Also worth noting is that training could take some time, depending on how many nodes there are in your model.
After selecting an input file via a file chooser, a popup window as shown in figure 2 will appear.



☐ Train layer 0	Learning rate for layer 0:	0.005
✓ Train layer 1	Learning rate for layer 1:	0.005
✓ Train layer 2	Learning rate for layer 2:	0.025

Figure 2: Training popup

The application requires two input values for each of the 3 layers of the model: whether a given layer should be trained at all, and the learning rate parameter for that layer.
A specific layer will be trained if and only if the associated checkbox was ticked by the time you press OK.

2.3.2 Simulate

Can only be used if a model is currently loaded.
While this operation takes place, the application will be completely unresponsive. Also worth noting is that training could take some time, depending on how many nodes there are in your model.
After selecting an input file and output file (in that order) through different file choosers, the application will start simulating the model, and will only become available for use again once the simulation is complete.

2.4 Other operations

2.4.1 Help

This will open the reference guide that was included alongside the application. The `reference.pdf` file must be in the same directory ("folder") as the `.jar` file of the application itself for it to be found. The `pdf` reader used is the one set as the default for your system.

3 The text interpreter

An alternative to the graphical user interface is a built-in command line interface. To use it, run the application's `.jar` file with the additional argument `-t`. This will make the application read commands from standard input and execute them as they are read. The process stops once EOF is reached.

3.1 Executing commands via text

Every line in standard input must contain exactly one complete command, including its required arguments separated by whitespace. The list of available commands can be found in table 1.
An example of a valid command would be

```
new 3 2 2
```

Command	Description	Arguments
`new`(1)	Creates a new model and replaces the one currently in use (if any)	Sizes of layers 0, 1 and 2, in that order, separated by whitespace
`load`	Loads a model previously saved in a file and replaces the one currently in use (if any)	Path to file to which the model was saved
`save`	Saves a model into a file	Path to file to which to save the model
`train`	Trains the model currently in use	Path to training dataset
`run`	Simulates the model currently in use	Path to input dataset and path to output dataset, in that order.
`set`(2)	Sets a parameter of the model	The parameter name, along with its new value.

Table 1: Available commands for text interpreter

(1) When created, all of the model's parameters are set to their default values as described in 3.2
(2) The available parameters and their respective value format will be discussed in section 3.2.

3.2 Setting model parameters

A valid `set` command is of the following form:

```
set parameter_name param_arg_0 param_arg_1 ...
```

Where the number of parameter arguments depends on the parameter being set.
Table 2 describes the arguments each parameter requires.

Parameter name	Description	Arguments	Example
`output_layer`	Output layer of the model	Index of new output layer(1)	`set output_layer 2`
`detect_instability`	Whether or not the simulator should try to detect and prevent instability	"`true`" or "`false`"	`set detect_instability false`
`layer_training`	Which layers should be trained	"`true`" or "`false`" for each of the 3 layers(2)	`set layer_training false true true`
`learning_rate`	Learning rate of a layer	Layer index(1) and new learning rate value	`set learning_rate 1 0.025`

Table 2: Parameter names and arguments

(1) Layer index refers to the number which uniquely identifies a layer (0, 1 or 2, with 0 being the input layer)
(2) The three values are considered to be for layers 0, 1 and 2 respectively.

4 Error handling

This section was created to minimize confusion about error messages that may appear. That being said, it is possible that some errors ask you to contact the current application maintainer. If that happens, please open an issue on the project's GitHub repository describing the error message and which functionality caused it.
The following list contains the different error messages you may find using the application:

4.1 Invalid layer size

Cause: One or more of the text fields in the model creation popup (see figure 1) was not a positive integer.

4.2 Could not write to file

Cause: File write operation failed. This could have many sources, among which are:

- The path already exists and the file is read only
- You do not have permission to write to the path specified
- Operating system could not open file

4.3 Could not load model from file

Cause: Invalid input file. This could have many sources, among which are:

- The file does not contain a valid model
- The file was moved or deleted mid operation
- You do not have permission to write to the path specified
- Operating system could not open file

4.4 Inconsistent dataset width

Cause: The dataset has rows with different lengths.

4.5 Dataset width does not match layer 0 size

Cause: The number of elements per row in the dataset is not the same as the number of nodes in the input layer.

5 Authors

Denis R. M. Piazentin	K_{III} simulator code
William Q. Ferreira	Application code