Neural Network Simulator

Documentation

Najib Ghadri

2016

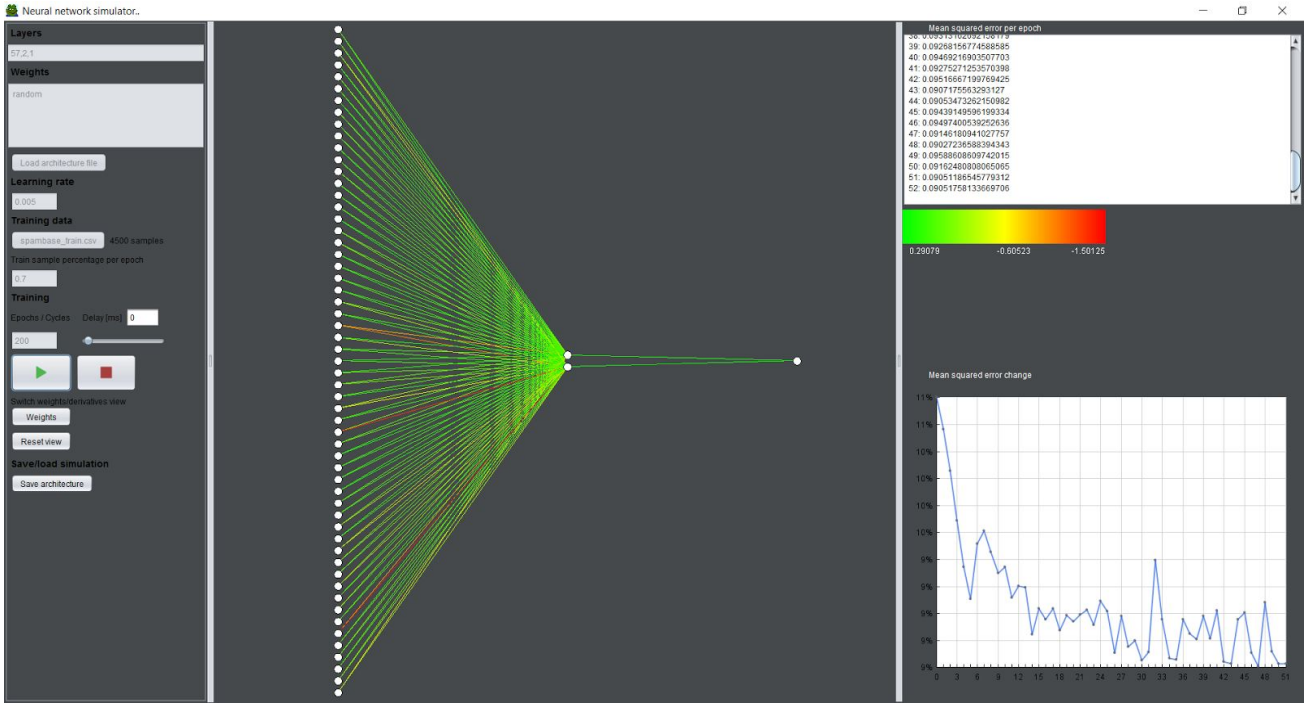
https://github.com/najibg96/NeuralNetworkSimulator

Description

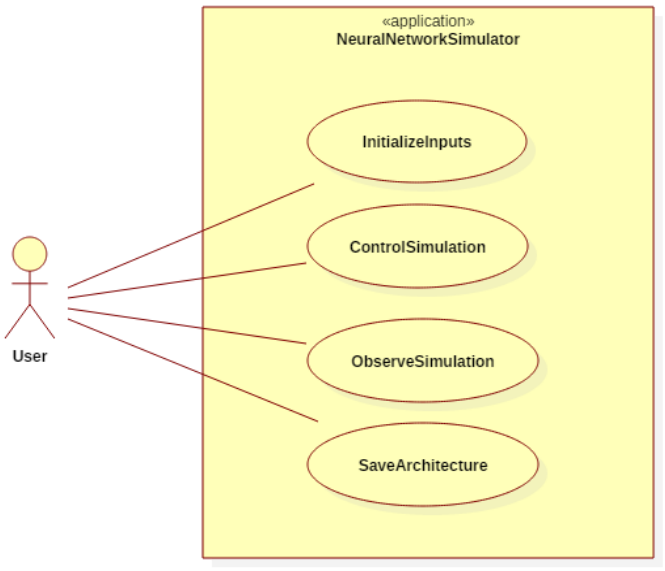
The objective of this software is to provide a good visualization of how neural networks work.
It not obective to create a good market product of a neural network implementation, neither it is to make any industrial use of it.
It is not the objective to explain how nerual networks work.

The software implements a simple multilayer perceptron (MLP), and visually simulates the calculation procedure on a very esthetic and ergonomic interface.

The software was written in Java, using SWING for gui purposes. Big emphasis was put on the reusability and well documentedness of the source code. Thus the architecture of the software is approached by an MVC (model-view-controller) or rather MVP (presenter) pattern, as you may see in the source code.
The user of the software is free to do anything under the MIT license.



Uses, Use-cases



Initializing inputs

Layers

Describes the architecture of the network. Numbers separated with commas, resolve to the number and width of the network. First number is the input dimension, last is the output.

After the user updates the fields with correct values, the network will show in the graphics canvas.

Weights

Initializes the weights of the network. If “random” is written then every weight will be assigned a random value using normal distribution centering 0, dispersing by 0.1.

The weights are given in csv format, such that every line represents the a neuron in the network. Neurons come ordered after each other layer after layer. Values in a line repsresent the weights of the incoming connections and a last value for bias of the neuron.

The weights of the network is shown in colorized lines, according to the color scale.

Learning parameters, epochs, train percentage, learning rate

Epochs mean the number of learning iterations.
Train percentage is the percentage of the sample database used for training.
Learning rate is the rate by which the network will change its weights (>0.1 not recommended).

Training samples

User can load **training samples for the learning/testing simulation.**
The samples should be in the following format:
Every line means a sample, where the first n numbers separated by comma will mean the input and the last m numbers separated by comma will mean the correct output vector.

If the n and m values do not match the networks input and output dimensions the simulation will abort.

Controlling the simulation

After setting the inputs, the user can **start, pause and stop** the simulation from the control panel.
With *Delay* field the user is able to change the **speed** of the simulation.
The user can **zoom and drag** the canvas at any time to see more of the network.
Switch between **derivative view and weight view**: which values to show on colorized edges.
Reset the view.

Observing the simulation

After starting the simulation, the user can observe the gradual change of the weights of the network toward a state of minimum error.
The **mean squared error of every epoch is readable on a graph** on the rigth panel.
The network is drawn in the middle panel.

Saving the architecture

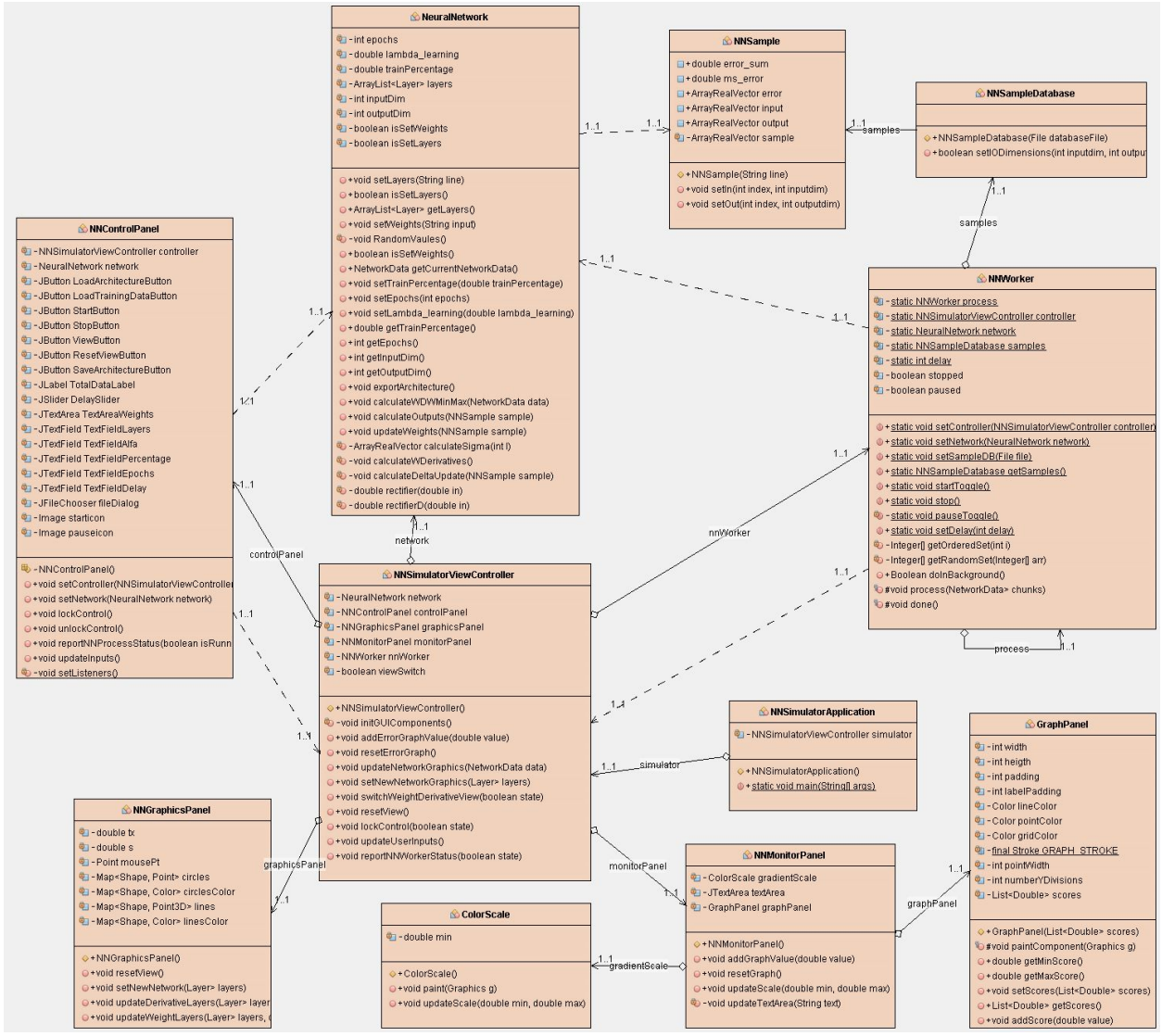
Optionally the user can **save the actual weight configuration of the network** into a file called ***architecture.txt*** which is in the same format as the input.

Class diagram

The implementation is based on the Model-View Controller architecture pattern or (MVP), where NeuralNetwork (NNSampleDatabase, NNSample) are the the model, NNGraphicsPanel, NNMonitorPanel (ColorScale, GraphPanel) is the view and NNControlPanel and NNSimulatorViewController are the controllers. Obviously MVC is loosely applied.
NNSimulatorApplication is the starter class.
NNWorker uses a Singleton pattern.

Documentation of classes available at. https://najibg96.github.io/NeuralNetworkSimulator/

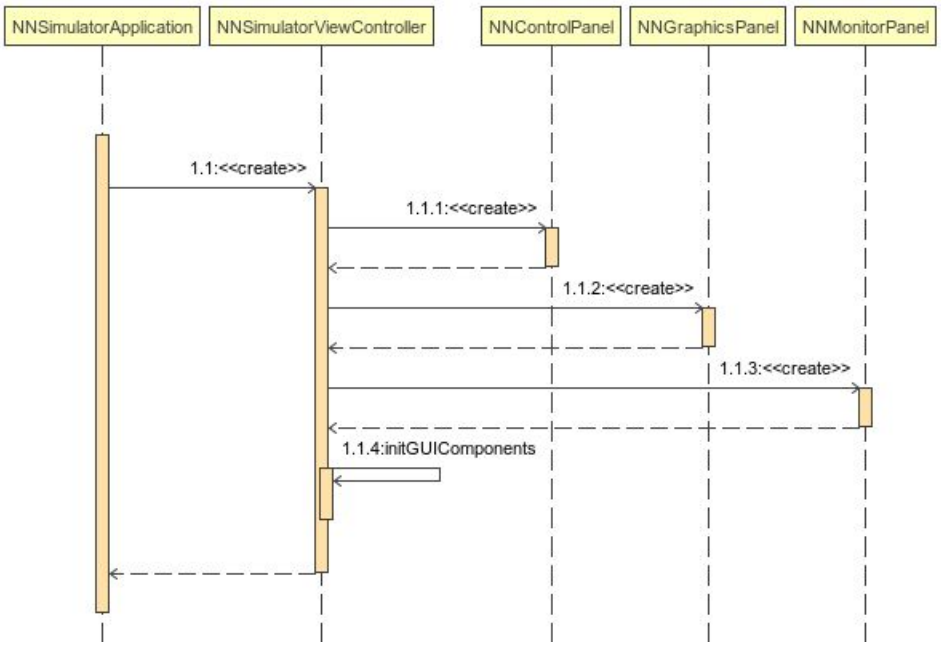
Class diagram



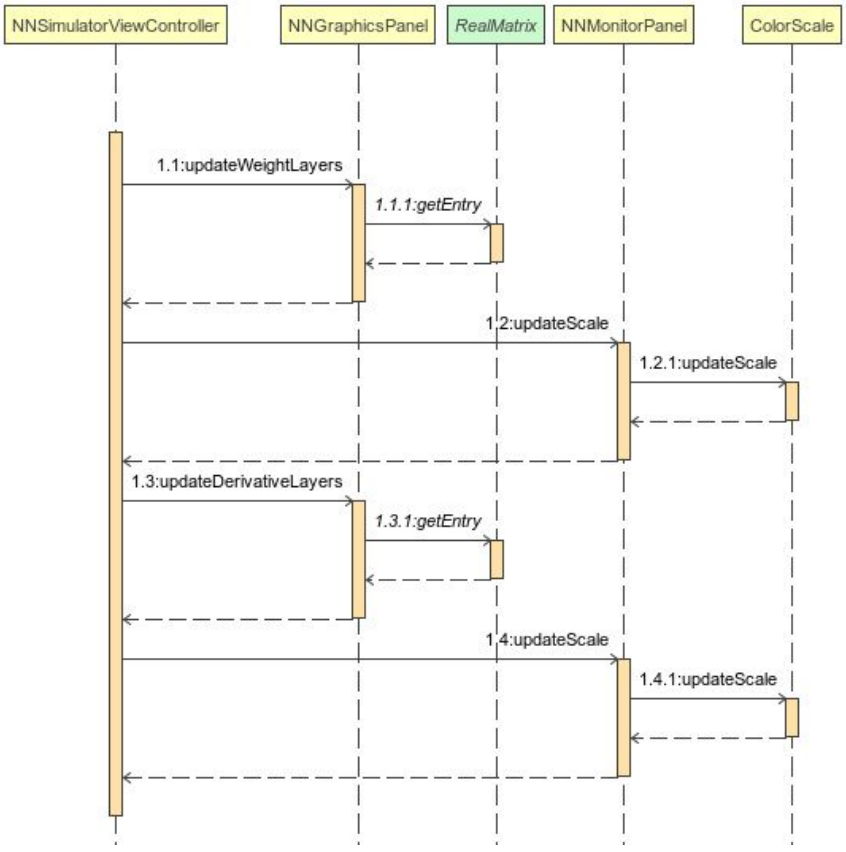
Sequence diagramms

Of main functionality

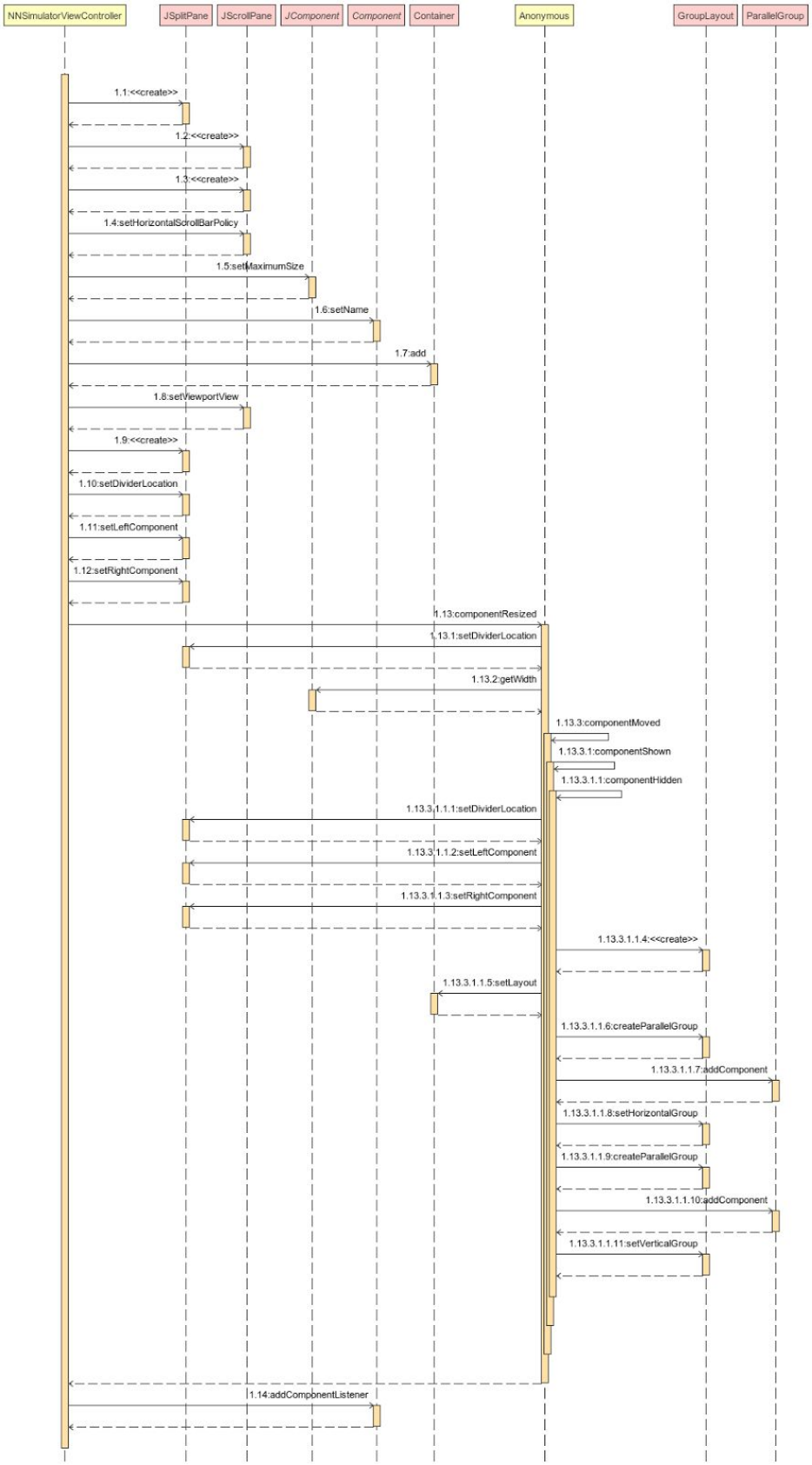
Initialize application



Update graphics panel



Create controlpanel


NNSimulatorViewController
JSplitPane
JScrollPane
JComponent
Component
Container
Anonymous
GroupLayout
ParallelGroup
1.1:<<create>>
1.2:<<create>>
1.3:<<create>>
1.4:setHorizontalScrollBarPolicy
1.5:setMaximumSize
1.6:setName
1.7:add
1.8:setViewportView
1.9:<<create>>
1.10:setDividerLocation
1.11:setLeftComponent
1.12:setRightComponent
1.13:componentResized
1.13.1:setDividerLocation
1.13.2:getWidth
1.13.3:componentMoved
1.13.3.1:componentShown
1.13.3.1.1:componentHidden
1.13.3.1.1.1:setDividerLocation
1.13.3.1.1.2:setLeftComponent
1.13.3.1.1.3:setRightComponent
1.13.3.1.1.4:<<create>>
1.13.3.1.1.5:setLayout
1.13.3.1.1.6:createParallelGroup
1.13.3.1.1.7:addComponent
1.13.3.1.1.8:setHorizontalGroup
1.13.3.1.1.9:createParallelGroup
1.13.3.1.1.10:addComponent
1.13.3.1.1.11:setVerticalGroup
1.14:addComponentListener

One calculation step