NNDefToolkit - Usage HOWTO

Version 1.0.1, Feb-10-2004

The purpose of this document is to describe the usage of NNDefRun application. NNDefRun is used to evaluate trained neural networks saved in NNDEF XML files applying user-defined input values. The library currently supports Multilayer Perceptrons and Radial Basis Function networks.

---

1. Prerequisits
2. Command Line Execution
3. Library Binding

---

1. Prerequisits:

1.1 Input NNDefXML file. You can download a sample file from here. You can generate your own XML file from within Matlab environment using NNDef Generator or manually using any text/xml editor.

1.2 The XML file should conform to NNDef DTD (Document Type Definition). Make sure *SYSTEM* value (located within *DOCTYPE* declaration) points to available DTD location.

1.3 Of course, you should have NNDefRun library (packed as a single JAR file) as well as Java Runtime Environment installed on your PC.

---

2. Command Line Execution

You can use either of the following methods to execute NNDef from command line:

2.1 Without command line parameters.

Call java bytecode interpreter from within the directory, containing NNDefRun.jar (don't forget -jar option):

```
java -jar NNDefRun.jar
```

2.2 User -log option for testing and debugging:

```
java -jar NNDefRun.jar -log
```

In both cases you'll be prompted for NNDef XML file name (see 1.1) following by Input values.

2.3 Specifying NNDEF XML file name from within command line.

File name should always be the first command line parameter (or second if '-log ' exists):

```
java -jar NNDefRun.jar nndef_sample.xml
or
java -jar NNDefRun.jar -log nndef_sample.xml
```

2.4 Additionally, you can pass input values from the command line:

```
java -jar NNDefRun.jar -log nndef_sample.xml 0.1 male
```

In this example '0.1' and 'male' are the inputs to selected model. The first value is a *simple numeric input*. Simple numeric input is described in NNDEF XML like that:

```
<Input OrderId="2" Name="Height"/>
```

While an option input looks like:

```
<Input OrderId="3" Name="Gender">
   <Option ActualValue="male"   TransValue="0"/>
   <Option ActualValue="female" TransValue="1"/>
</Input>
```

For this input the only valid values are 'male' or 'female'.
If you enter invalid values, you'll be prompted for correction.

2.5 Additonally, you can specify some parameters from command line and some during execution. All values will be striclty validated.

2.6 As an alternative for entering inputs you can pass the name of input file as a command line parameter. This text file confirms following simple rules:

- comment lines are starting with "#"
- all the inputs are stored on the first non-comment line and separated by spaces

To invoke the runner with input file, use the argument: "-input <in_file_name>"
Example:

```
java -jar NNDefRun.jar -log nndef_sample.xm -input input_file.txt
```

---

3. Library Binding

NNDef library can be used in both single and multi threaded modes.

Following step by step instruction will guide you through the process :

3.1 Place NNDefRun.jar within your classpath or current directory.

3.2 Import NNDefRun package as well as java.io.IOException :

```
import java.io.IOException;
import com.makhfi.NN.NNDefRun;
```

3.3 Create NNDefRun object. And specify NN model described in NNDEF XML file:

```
NNDefRun r = new NNDefRun();
r.setXMLFile("nndef_sample.xml");
```

Alternatively, you can combine these 2 lines by invoking another constructor of that object:

```
NNDefRun r = new NNDefRun(XMLFName);
```

3.4 Set your input values:

```
String[] inputs={"0.25", "0.7", "male"};
r.setDirectInputs(inputs);
```

Note that all values are represented as strings. Numeric values represent simple inputs, while option inputs are represented by their string values. See 2.4. for option inputs description.

3.5 Start NNDefRun calculation engine.

3.5.1 For single threaded mode use:

```
r.execute();
```

3.5.2 Alternatively, you can use:

```
r.start();
```

This will run the task in a separate thread

3.6 Use following methods to retrieve the resulting values and corresponding output names:

```
String[] outputs = r.getDirectOutputs();
String[] outputNames = r.getOutputNames();

for (int i=0; i<outputs.length; i++){
  System.out.println("Output "+outputNames[i]+" has value "+outputs[i]);
}
```

Alternatively you can bypass any data translation use:

```
r.setInputs(doubleInputs[]);
and
doubleOunputs[] = r.getOutnputs();
```

More to follow . . .