information-dynamics-toolkit

JIDT: Java Information Dynamics Toolkit for studying information-theoretic measures of computation in complex systems

UseInOctaveMatlab

How to use the toolkit in Octave or Matlab
octave, matlab, Phase-Deploy

Updated Today (moments ago) by joseph.lizier

Introduction

The java code from this toolkit can easily be used in Octave or Matlab. This page contains basic information on how to get up and running with the toolkit:

- Matlab instructions
- Octave instructions

Several longer examples of using the toolkit in Octave/Matlab can be viewed at OctaveMatlabExamples.

Use in Matlab

The ability to use Java from within Matlab comes with Matlab out of the box. Full info is available at:

- http://au.mathworks.com/help/matlab/using-java-libraries-in-matlab.html
- http://au.mathworks.com/help/matlab/matlab_external/product-overview.html

You can run your Java code inside Matlab fairly simply:

1. Tell Matlab about your jar, e.g.: `javaaddpath('../Information Dynamics/infoDynamics.jar');`
2. Create an instance of a class, e.g. by direct instantiation `miCalc = infodynamics.measures.continuous.kernel.MutualInfoCalculatorMultiVariateWithDiscreteKernel();` or using `miCalc = javaObject('infodynamics.measures.continuous.kernel.MutualInfoCalculatorMultiVariateWithDiscreteKernel')`
3. Use the object: `miCalc.initialise(4, 2, 0.5);`
4. Note that Matlab automatically makes conversion between its own and Java arrays.

For *static* methods, note that you can simply call it on the class itself, or on an object of the class - see http://www.mathworks.com.au/help/techdoc/matlab_external/f46719.html#f23705

If you get any `java.lang.OutOfMemoryError` errors, then you need to increase the heap space allocated by Matlab (it is usually set to a rather low value) -- doing this is described e.g. here.

Finally, sometimes Matlab can give warnings such as `Warning: Objects of infodynamics/measures/continuous/kraskovTransferEntropyCalculatorMultiVariateKraskov class exist - not clearing java`. To get rid of these errors, then you can clear the relevant Java variables in Matlab, e.g. `clear teCalc`, or clear all Java variables: `clear java`.

See longer code examples at OctaveMatlabExamples.

Use in Octave

Here I will describe material I found useful to get started with Java inside Octave, then how I installed octave-java, then how to run the code.

For octave 3.8 onwards, java functionality is natively available (this is the version carried for ubuntu 14.04, which I'm currently using).

Otherwise, material that I found useful regarding using Java code inside Octave:

- http://sourceforge.net/mailarchive/forum.php?thread_name=128f38bd0911170856i2d9dc349x28e02b3fe3333011%40mail.gmail.com&forum_name=octave-dev
- http://octave.1599824.n4.nabble.com/How-do-I-install-the-java-and-jhandles-packages-td1632116.html
- http://www.octave.org/wiki/index.php?title=Java_package

Here is a list of how I got octave-java installed (on ubuntu 12.04):

1. Set-up steps:
   1. Set up an environment variable for JAVA_HOME - for me `/usr/lib/jvm/java-6-openjdk-amd64`
   2. Make sure that `$JAVA_HOME/jre/lib/<ARCH>/client/libjvm.so` exists - on 64-bit machines, the required file exists in `server` directory, not `client`. We can get around this by adding a symlink - I created the directory `client` and symlink from inside the `client` directory to the file in the `server` directory. I've been told that an alternative fix is to change the `_java_.cc` file as described here.
2. Install octave-java from the Octave-forge project either:

1. through your linux package manager (this didn't work properly for me on ubuntu 12.04), or
2. via directly downloading the `.tar.gz` file from http://octave.sourceforge.net/java/ then starting octave in root mode (`sudo octave`), and inside octave run the installation, e.g.: `pkg install -verbose java-1.2.8.tar.gz`. To uninstall at a later date, you can run: `pkg uninstall java`. After running the installation in root mode, you will need to add read access for yourself to the `doc.info` file for running the `doc java` command.

Once all that is set up, you can run your java code just as simply as in Matlab:

1. Tell octave-java about your jar, e.g.: `javaaddpath('../Information Dynamics/infoDynamics.jar');`
2. Create an instance of a class, e.g.: `miCalc = javaObject("infodynamics.measures.continuous.kernel.MutualInfoCalculatorMultiVariateWithDiscreteKernel");`
3. Use the object: `miCalc.initialise(4, 2, 0.5);`
4. While Matlab automatically makes conversion between its own and Java arrays, Octave does not. To facilitate such conversion, scripts are provided for converting arrays back and forth between these environments in the `demos/octave` directory; see further discussion at OctaveJavaArrayConversion and example uses in OctaveMatlabExamples.

For *static* methods, you can call them on an object of the class itself, or using the javaMethod function - http://octave.sourceforge.net/java/function/javaMethod.html

See longer code examples at OctaveMatlabExamples.