UseInPython

Joseph Lizier edited this page 5 days ago · 8 revisions

How to use the toolkit in Python

Introduction

The java code from this toolkit can easily be used in Python. First we describe calling java code from Python, then specifically describe the use of this toolkit.

Several longer examples of using the toolkit in Python can be viewed at PythonExamples.

Using Java objects in Python

There are several alternatives for using Java objects in Python, e.g. JPype and Jython.

Here, we focus on using JPype for several reasons, including:

- It is run as an import to ordinary python code in the standard interpreter (jython runs as an alternative python interpreter, and so is not always up to date with the main python interpreter)
- It is packaged for ubuntu (which is my platform of choice) as `python-jpype` As such, here we will only describe use via JPype, though of course you could use an alternative method quite easily.

Using JPype

The first step is to install JPype. On ubuntu for Python 2, one simply installs `python-jpype` via the ubuntu package manager. On other platforms, install as you would other python extensions (e.g. via `pip install jpype`).

For Python 3, you may need to install a slightly different version of JPype:

- On ubuntu linux it has been reported that `pip3 install jpype1` (You might need to `sudo` that) will install a python 3 compatible JPype (thanks to Michael Wibral). `jpype1-py3` is another available package, however it seems that `jpype1` is more efficient.

- On windows it has been reported that this works inside anaconda: `pip install -i https://pypi.anaconda.org/pypi/simple jpype1` (thanks to Jonathan de Vries)

You can then run your java code fairly simply in python with JPype:

1. Import the relevant packages from jPype, e.g.: `from jpype import *`
2. Start the JVM and tell it where our jar file is, e.g.: `startJVM(getDefaultJVMPath(), "-Djava.class.path=" + jarLocation)`
3. Create a reference to the package and class that you wish to create an instance of, e.g.: `teCalcClass = JPackage("infodynamics.measures.discrete").TransferEntropyCalculatorDiscrete`
4. Create an instance of the class, e.g.: `teCalc = teCalcClass(2,1)`
5. Use the object, e.g.: `teCalc.initialise()`
6. Shutdown the JVM when it's no longer required, e.g.: `shutdownJVM()`

Array conversion between python and java requires some extra code, but is perfectly do-able - see sections 6 and 7 of the JPype user guide. *For example*, to convert a one-dimensional python array `myArray` to a java one dimensional array of doubles, use `JArray(JDouble, 1)(myArray)` -- see e.g. Example 3 in PythonExamples.

Numpy Alternatively, if you use numpy.array objects these are (*somewhat*) directly accepted by the java methods:

- For doubles (continuous variables), yes you can pass numpy arrays straight through -- see e.g. Example 6 and Example 9 in PythonExamples -- these work in both python 2 and python 3, and indeed I think this is preferable at least in python3 with jpype1.
- For ints (discrete variables), you can only pass numpy arrays straight through in python 2 with jpype, whereas with python 3 with jpype1 you need to convert the numpy array to a list and then a jpype.JArray -- see e.g. the code and comments in Example 1

For static methods, you can call them on the reference to the class itself.

Several longer examples of using the toolkit in Python can be viewed at PythonExamples.

JIDT -- Java Information Dynamics Toolkit -- Joseph Lizier *et al.*