Tutorial

A brief tutorial for how to use JIDT

tutorials

The distribution (from V1.2.1) ships with a tutorial for users in the `tutorial` folder, consisting of a PDF of tutorial slides, a further guide to exercises, and sample solutions.

Further description of these components follows.

Slides

A 2-3 hour tutorial on JIDT has been presented at several conferences (see below). The slides from the tutorial are available in the `tutorial` folder of the distribution (from V1.2.1 onwards), or directly here.

The slides introduce the information-theoretic measures and estimation techniques, then briefly introduce the toolkit, walk through some of the demos, and guide the user through an exercise (expanded on below).

A shorter 30-minute presentation is available here, though obviously it does not walk the user through using JIDT in the same level of detail.

Excercise

To consolidate your understanding of the toolkit, you will complete a task of applying one of the calculators to one of the sample data sets to create some new insights. The exercise comes with some code to get started in Java, Matlab/Octave and Python.

1. Compute MI in heart-breath interaction

The demonstration set SchreiberTeDemos uses our toolkit to recreate Schreiber's original transfer entropy analysis of information flow between heart and breath rate, and then adds some further insights using our KSG (Kraskov et al.) Transfer Entropy and Active Information Storage calculators.

Your exercise task is to:

1. Compute the Mutual Information between the heart and breath rate for this data set (using samples 2350 to 3550, inclusive);
2. Using KSG Mutual Information estimator (with KSG algorithm 2) with 4 nearest neighbours.

You can either:

1. write code for this from scratch in your preferred environment (the data set is at demos/data/SFI-heartRate_breathVol_bloodOx.txt with heart rate in the first column and breath rate in the second), or
2. you can start by using one of our sample scripts which calculate Transfer Entropy on this data set as a template. These samples include code to read in the data and parse the arguments. There are samples available for Transfer Entropy here in either:
    1. Java (source file at demos/java/infodynamics/demos/schreiberTransferEntropyExamples/HeartBreathRateKraskovRunner.java and shell script to run it at [demos/java/SchreiberTransferEntropyExamples/heartRateBreathRateKraskov.sh]) or
    2. Octave (demos/octave/SchreiberTransferEntropyExamples/runHeartBreathRateKraskov.m) or
    3. Python (demos/python/SchreiberTransferEntropyExamples/runHeartBreathRateKraskov.py).
3. (**easiest way**) you can auto-generate some code to get started by using our AutoAnalyser demo. Follow the steps for using this demo described on its webpage, including using the MI AutoAnalyser, and selecting the correct data file, the columns for the variables and the properties for the calculator. Note that the auto-generated code will use the entire data set from the file; to use the smaller set of samples you would need to edit the auto-generated code.

The **answer** that you are looking for is **0.134 nats** (+/- 0.002 nats, due to use of noise addition by the calculator) if you set only the properties mentioned above (e.g. don't change addition of noise to the data, or add dynamic correlation exclusion, etc.).

HINT -- If you're getting stuck, you really should start with option 3 above. If you're keen for a bigger challenge then that but still not sure what to do, then start with one of our sample TE scripts as in step 2 above, and try to read through and understand how it is working first, before aiming to convert it to MI and set the required properties. Remember that in using any of the calculators, you should follow the following steps:

1. Construct the calculator -- which class should you be using to achieve this calculation? (Remember we want MI, by KSG estimator, algorithm 2)
2. Set relevant properties for the calculator -- what properties are available? (Check the Javadocs for this class)
3. Initialise the calculator -- what parameters should be supplied here? (Check the Javadocs for this class)
4. Supply the data to the calculator using a `set/addObservations()` method

5. Calculate the average MI.

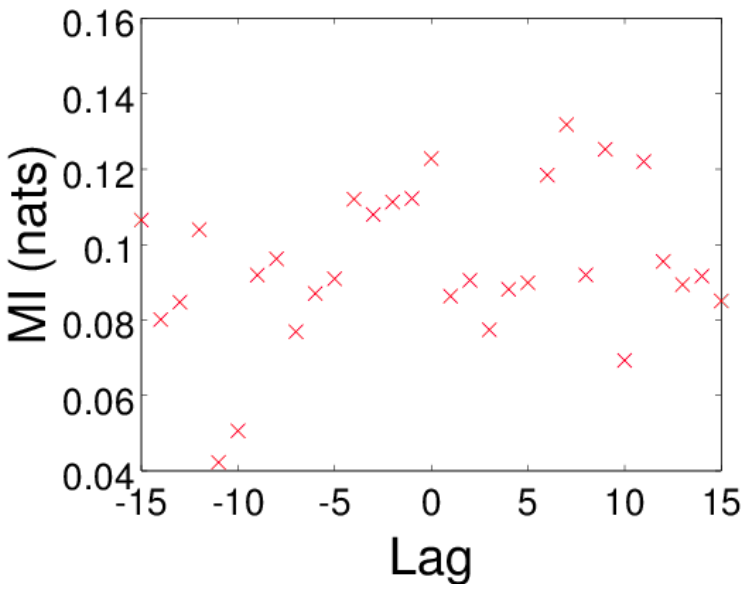
(If you are really stuck, you can compare your answer to the sample provided at tutorial/sampleExerciseSolutions/matlabOctave/runHeartBreathRateKraskovMI.m or the excerpt of this in the tutorial slides.)

2. Challenge 1. Compute MI over various lags

Extend your code from the above task to compute the Mutual Information between heart rate and breath rate, for a variety of lags between the two time-series. For instance, investigate lags of (0, 1, ..., 14, 15).

HINT: You could either shift the time-series with respect to eachother to affect the lag, or a cleaner method can be achieved if you investigate the available properties for this MI calculator in the Javadocs (*Challenge*: what to do if you want to use negative lags, i.e. a positive lag from breath to heart, with this property?)

Sample results:



What would you interpret from the results? Can you think of some logical further investigations here?

(If you are really stuck, you can compare your answer to the sample provided at tutorial/sampleExerciseSolutions/matlabOctave/runHeartBreathRateKraskovMIWithLags.m)

Where this tutorial has been run

1. European Conference on Artificial Life, July 20, 2015 -- see our page for this tutorial
2. Australian Conference on Artificial Life and Computational Intelligence, February 5, 2015