DetectingInteractionLags

Demo to show how to run the transfer entropy calculators for a source-destination lag.

examples, octave, matlab

Updated Today (moments ago) by joseph.lizier

Demos > Detecting interaction lags

Detecting interaction lags

This demo shows how to use the transfer entropy calculators to investigate a source-destination lag that is different to 1 (the default).

As described below, this software was used to make the comparisons of transfer entropy (TE) and momentary information transfer (MIT) in the following paper (see Test cases Ia and Ib therein):

- M. Wibral, N. Pampu, V. Priesemann, F. Siebenhühner, H. Seiwert, M. Lindner, J.T. Lizier and R. Vicente, "Measuring information-transfer delays", *PLOS ONE*, vol. **8**, no. 2, e55809, 2013; doi: 10.1371/journal.pone.0055809.

Momentary information transfer was presented in the following paper:

- B. Pompe and J. Runge, "Momentary information transfer as a coupling measure of time series", Physical Review E 83, 051122 (2011).

The results from Wibral et al. lead to the conclusions that:

1. the TE is most suitable for investigating source-destination lags, showing that the MIT can be deceived by source memory, and also
2. that symbolic TE can miss important components of information in an interaction.

This demo is run in Matlab or Octave.

It is found at `demos/octave/DetectingInteractionLags` in the svn or full distribution.

Test case Ia -- Source with memory

As described in the paper above, we use the following model to construct a sample source-destination interaction for discrete data in the presence of source memory:

1. `X_{n} = f( X_{n-1} )`, where `f()` is a noisy function (defined below), with the noise representing new information entering the source;
2. `Y_{n+1} = X_{n}`, `Y_{n+1}` copies directly from `X_{n}`.

X and Y are sampled from the set `{0,1,2,3}`, and f() is defined in terms of `p(X_{n+1} | X_{n})` as a function of noise η (ref. Table 1 of Wibral et al. above):

		p(X_{n+1} | X_{n})		
X_{n}	**X_{n+1}=0**	**X_{n+1}=1**	**X_{n+1}=2**	**X_{n+1}=3**
0	(1-η)/2	η/2	(1-η)/2	η/2
1	(1-η)/2	η/2	(1-η)/2	η/2
2	η/2	(1-η)/2	η/2	(1-η)/2
3	η/2	(1-η)/2	η/2	(1-η)/2

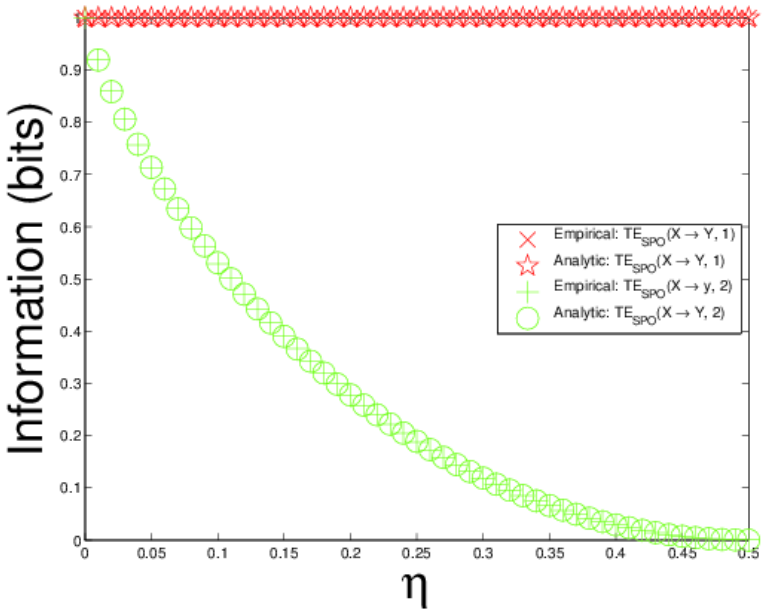
We **expect** to find the connection `X_{n} -> Y_{n+1}` dominating, since it is causal, if our measure of transfer is correct. Of course, `Y_{n+1}` is correlated to `X_{n-1}`, but these are connected only via `X_{n}`. Indeed, the paper of Wibral et al. above contains a proof that states that transfer entropy will be maximised from `X_{n}`.

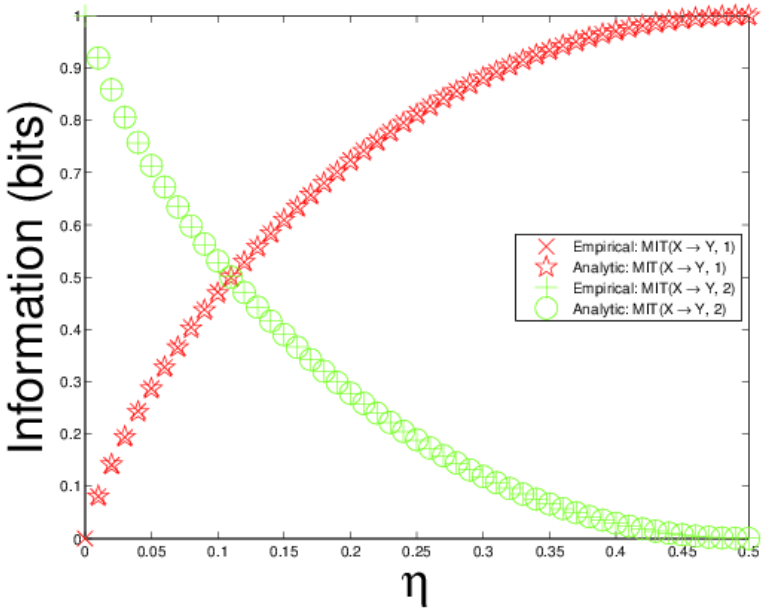
To investigate this example, run:

```
transferWithSourceMemory(savePlot)
```

If savePlot is set to true, the plots will be saved in `.eps` format. This script takes 1-2 minutes to run.

The script generates plots of transfer entropy and momentary information transfer from `X_{n} -> Y_{n+1}` (i.e. lag 1) and `X_{n-1} -> Y_{n+1}` (i.e. lag 2), over a range of noise parameters (which alter the amount of memory in the source):





As expected, this shows that TE for `X_{n} -> Y_{n+1}` dominates `X_{n-1} -> Y_{n+1}` over all values of noise. However, the Momentary information transfer is actually maximised for lag 2: `X_{n-1} -> Y_{n+1}`, for many noise levels instead (because for `X_{n} -> Y_{n+1}` it *conditions out* the memory from `X_{n-1}`); this shows that it is not maximised for the correct delay even in such simple unidirectional coupling in the presence of source memory. Further discussion is contained in the Wibral et al. paper above.

Notice in the code, that for the discrete TE calculator here, one needs to manually shift the source time-series fed into the calculator in order to examine a lag (this is not the case for the continuous TE calculators, see below).

NOTE: You may need to increase the Java heap space in Matlab for this to work (you will get a "`java.lang.OutOfMemoryError: Java heap space`" error if this is a problem).

Test case Ib -- Interaction delay in Coupled logistic map

Next, the Wibral et al. paper revisits transfer entropy TE(X->Y) for coupled logistic maps X and Y, where the process is as defined in Example V.A of Pompe and Runge above. We rewrite this process as follow:

```
fmod1(x) := 4 .* (x mod 1) .* (1 - (x mod 1));
gYToX = cYToX .* Y(t - delayYtoX) + (1-cYToX) .* X(t - 1);
X(t) = fmod1(gYToX);
gXToY = cXToY .* X(t - delayXtoY) + (1-cXToY) .* Y(t - 1);
Y(t) = fmod1(gXToY);
```

We use `delayYtoX = 5; delayXtoY = 2; cYToX = 0.2; cXToY = 0.5`; time-series length 512 and embedded history length `k=1` to match parameters in the papers above. We investigate `TE(X->Y, delay 1)` and `TE(X ->Y, delay 2)`, because Pompe and Runge reported that they had found `TE(X->Y, delay 1)` was larger than `TE(X ->Y, delay 2)` despite the X -> Y coupling delay being 2. This is *contrary* to what we would expect however, since `Y(t)` is causally determined from `X(t-2)` and `Y(t-1)`, we would expect to find more information in `X(t-2)` about `Y(t)` given `Y(t-1)` than from `X(t-1)`. Pompe and Runge had used a *symbolic* TE estimator though, which we suspected was missing important information in the interaction, since it only examples ordinal relationships. (Further commentary is provided in the footnote below).

As such, we re-investigate this example with the best of breed Kraskov-Stoegbauer-Grassberger (KSG) estimator,

To investigate this example, run:

```
coupledLogisticMap()
```

For the full 1000 repeats, the calculation takes 2-5 minutes. (Notice in the code, that for the continuous TE calculator here, one can simply call a more complete `initialise()` method to supply the source-destination lag to investigate, e.g. `teCalc.initialise(k,1,1,1,3);` for lag 3.)

Sample results are as follows:

```
TE(X->Y,delay=1) = 0.5735 nats (+/- std 0.0324, stderr 0.0010) or 0.8274 bits
TE(X->Y,delay=2) = 1.4703 nats (+/- std 0.0242, stderr 0.0008) or 2.1212 bits
TE(X->Y,delay=3) = 0.4772 nats (+/- std 0.0358, stderr 0.0011) or 0.6885 bits
```

As hypothesised, we find that *this more accurate estimator is indeed able to accurately identify the correct lag of 2 as having larger transfer entropy here*. This corresponds to the theoretical proof in Wibral et al. that the TE will be maximised at the correct source-lag delay for this type of bidirectional coupling.

Footnote: In the Wibral et al. paper, we conclude that the Pompe and Runge result was likely to be *"an artifact of their symbolic mapping approach: symbolic mapping may be a useful technique to handle small data sets, but it certainly removes parts of the information about the processes, and this information may well be relevant."* The main clue is that their measurement of MIT was significantly non-zero with lag 1, whilst intuitively it should be zero for lag 1, since that source variable should not be able to add any information about the destination (which is completely determined by its past and the source at lag 2, both of which MIT conditions out). The source at lag 1 is only seen to add information because they estimated it using ordinal values, which for this process appears to leave uncertainty regarding the ordinal position of the next state of the destination given the ordinal relationships amongst its past and the source at lag 2. Some information regarding this uncertainty seems to be provided by the far past of the destination which is directly coupled to the source at lag 1 via a lag 5 relationship. This information is useful because of the *memory* within the destination variable. In a similar fashion, the transfer entropy at lag 1 contains much information about what the source causally added at lag 2 (because of strong memory in the source). But it also contains more information about the far past of Y (beyond the history or embedding length 1 used for the measure) which can be helpful to decode the process Y symbolically (when there may be uncertainty remaining in the previous step of Y). If one is using ordinal relationships, it appears that this information is more strongly contained in the source at lag 1 than at lag 2.