Estimating embedding dimension with confidence probability

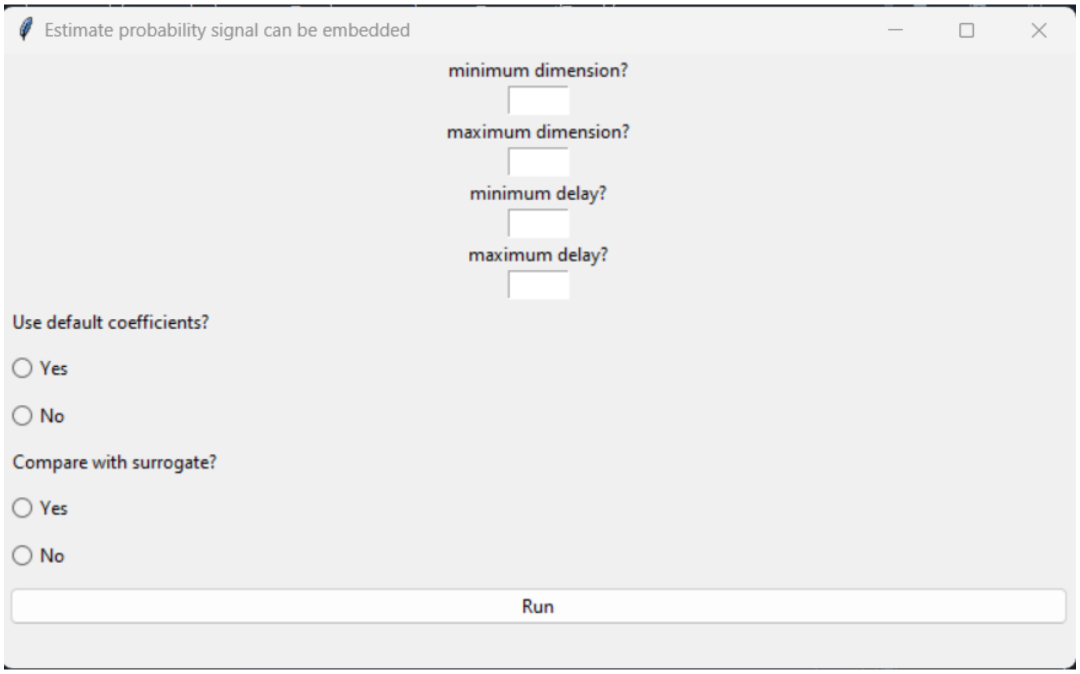
This Python code takes a single column ascii time series file and estimates the probability that it can be embedded in D dimensions with a delay of T. the method is based on T. L. Carroll and J. M. Byers, "Dimension from Covariance Matrices", Chaos, vol. 27, 023101 (2017).

The files included are the main program, "estimate_embedding_dim.py", and the additional files "embedding_prob_func_coeff.py" and "default_coeff.py". An additional code file "embedding_prob_func_coeff.py" is included in case the user wants to calculate their own fit coefficients for the probability distributions, but it is not necessary.

The Python libraries used for this code are numpy, scipy, array, tkinter, matplotlib,csv, math, cmath and random.

Quick start

Start the main program "estimate_embedding_dim.py". The following window appears:



Enter the minimum and maximum dimensions to be tested. The minimum dimension should be at least 2. Maximum dimensions greater than five or six are not very useful. A common rule is that embedding a D dimensional object requires at least 10^D points, so embedding a six dimensional object requires one million points, and the required number of points are even higher for higher dimensions. By default this program will go up to ten dimensions, but that is not recommended. Going to higher dimensions requires checking the "No" box under "Use default coefficients".

Enter the minimum and maximum embedding delays in the minimum and maximum delay boxes. Then choose under "Use default coefficients?". For now choose yes.

“Compare with surrogate” compares the probability that the signal can be embedded with the probability for a random surrogate generated from the same signal. The surrogate has the same spectrum as the input signal but has randomized phases. Sometimes filtered noise can look like a low dimensional signal, so this setting guards against that probability. If you know your signal is deterministic, check “No”.

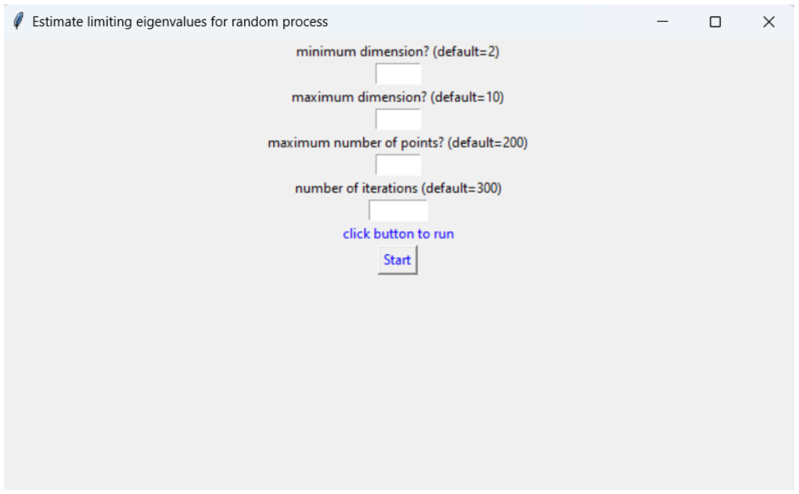
Once the selections have been made click “Run”. A window will now appear allowing you to choose the input signal. The input should be a single column ascii file. Next a window will tell you to specify the output file. The output is saved as a CSV file, so the file you create should have the “.csv” extension.

Other options

A detailed description of the algorithm is in the paper cited above. This method embeds the input signal with a particular delay T and dimension D and then clusters the embedded signal into local clusters. The clustering begins with choosing a small number of points within some radius around a randomly chosen center. Statistical principles are used to determine if clustering the points in this way contains more information than a same size set of points that are distributed uniformly. If the cluster contains no more information than a uniform distribution, the radius is increased until the cluster contains more information than a set of uniformly distributed points.

A covariance matrix is then created from this set of points and the eigenvalues of the covariance matrix are calculated. For a random process, the probability distribution of the eigenvalues of a cluster of points follows a known probability distribution- unfortunately, this distribution only holds for infinite numbers of points in infinite dimensions.

This is where the code “fit_covariance_eigenvalues.py” comes in. This code generates random matrices with a covariance matrix consisting of the identity matrix, drawn from a Wishart distribution for D dimensional processes with N points. The eigenvalues for each matrix are found and recorded. For each D and N combination the code creates a number of matrices determined by the “number of iterations?” parameter. The minimum and maximum dimensions and the maximum number of points are also chosen in the program window.


Estimate limiting eigenvalues for random process

minimum dimension? (default=2)

maximum dimension? (default=10)

maximum number of points? (default=200)

number of iterations (default=300)

click button to run

For each D and N the program saves the maximum and minimum values of the covariance eigenvalues. These values represent the maximum and minimum covariance eigenvalues for a random process with N points in D dimensions. This Monte Carlo simulation is repeated the full range of N and D.

The curves of the maximum and minimum eigenvalues as a function of N for each dimension are then fit with a polynomial. The output of the program will be the fit coefficients for each dimension.

After clicking the “Start” button a window opens to choose the output file. The data is saved in the “.npz” format, so the file extension should be “.npz”. In the main program “estimate_embedding_dim.py” this file may be chosen by choosing “No” under “Use default coefficients?”. The default coefficients were calculated for 10,000 iterations, a maximum of 1000 points and a range of dimensions from 2 to 10.