1 Autocorrelation

1.1 Definition

Autocorrelation is a correlation of a signal with a delayed copy of itself as a function of delay and it is mostly used for finding repeating patterns.

Autocorrelation:

$$R(s.t) = \frac{E[(X_t - \mu_t)(X_s - \mu_s)]}{\sigma_t \sigma_s} \tag{1}$$

where X_t is the given run of a process. If the mean exists and the variance is not zero R must be in the range of $[-1, +1]$; $+1$ for perfect correlation and -1 for perfect anti-correlation.

Expressing autocorrelation as a time lag

$$R(\tau) = \frac{E[(X_t - \mu)(X_{t+\tau} - \mu)]}{\sigma^2} \tag{2}$$

which is an even function; $R(\tau) = R(-\tau)$.

Properties can be found in `https://en.wikipedia.org/wiki/Autocorrelation`.

1.2 Efficient calculation

Autocorrelation can be computed direct from the definition but is order n^2, while there are several algorithms of order $nlog(n)$ for example the Wiener–Khinchin theorem.

`https://en.wikipedia.org/wiki/Wiener%E2%80%93Khinchin_theorem`

Autocorrelation function and the Wiener-Khinchin theorem:

`https://www.itp.tu-berlin.de/fileadmin/a3233/grk/pototskyLectures2012/pototsky_lectures_part1.pdf`

Wiener–Khinchin theorem uses Fast Fourier Transforms (FFT and IFFT).

$$F_R(f) = FFT[(X_t)] \tag{3}$$

$$S(f) = F_R(f) F_R^*(f) \tag{4}$$

$$R(\tau) = IFFT[S(f)] \tag{5}$$

2 Program for autocorrelation

2.1 Development in general

There are a lot of codes on the net in proportion with growing complexity with increasing number of bugs. Generally speaking I always try the algorithms in scripting languages (Matlab or Python) and when they work, I develop the C++ code and compare the results.

FFT source programs:

`http://s.pudn.com/search_hot_en.asp?k=fft+ifft`

For Fast Fourier Transform in C++ I choose the GNU Scientific Library;

`https://www.gnu.org/software/gsl/doc/html/index.html`

2.2 Structure of the Program

The structure of the program is straightforward.

The GSL radix-2 algorithms use the Cooley-Tukey algorithm to compute in-place complex FFTs for lengths which are a power of 2. Advantage: no additional storage is required.

The mixed-radix functions work for FFTs of any length. They are a reimplementation of Paul Swarztraubers Fortran FFTPACK library.

2.3 Programs

Simple konsole application under Linux.

C++ code for autocorrelation using WienerKhinchin theorem:

autocorrelation_example.cpp

makefile (OpenCV libraries are not necessary in this case only GSL)

lew.dat (input data for checking, it can be downloaded)

`http://www.itl.nist.gov/div898/handbook/eda/section3/eda35c.htm`

reference.dat (reference data for comparing results, it can be downloaded)

`http://www.itl.nist.gov/div898/handbook/eda/section4/eda4251.htm`

out.dat (result, output from the code)

Python 2.x code for Rapid Application Development/checkig/testing using NORMAL Python autocorrelation function:

autocorr_ex2.py (it uses NORMAL Python autocorrelation function)