Time Series Autocorrelation

DUDEKULA USENI - AHILAN R
r171099@rguktrkv.ac.in - ra1atcuk@gmail.com
FWC22098-FWC22090 - IITH Future Wireless Communication (FWC)

February 24, 2023

1 Abstract

Time series autocorrelation

We would like to experiment and see can Raspberry Pi and/or Jetson Nano do time series autocorrelation for large data sets. Data set usually will contain about 40e6 to 1.5e8 points and will be sampled uniformly

2 Methods used

Given measurements, $Y_1, Y_2, ..., Y_N$ at time $X_1, X_2, ..., X_N$, the lag k autocorrelation function is defined as

$$r_k = \frac{\sum_{i=1}^{N-k}(Y_i - \bar{Y})(Y_{i+k} - \bar{Y})}{\sum_{i=1}^{N}(Y_i - \bar{Y})^2}$$

2.1 Python only implementation

This is a Python-only method without any external dependencies for calculating the autocorrelation.

2.2 Statsmodels

Statsmodels is a great library for statistics and it provides a simple interface for computing the autocorrelation.

2.3 numpy correlate

Numpy provides a simple function for calculating correlation. Since autocorrelation is basically correlation for a data set with itself, we will use numpy.correlate to calculate autocorrelation.

2.4 Fourier Transform Implementation

It turns out that autocorrelation is the real part of the inverse Fourier transform of the power spectrum. This follows from the Wiener–Khinchin theorem. We can exploit this, and write the following simple algorithm

(a) Take the Fourier transform of our data set.

(b) Calculate the corresponding power spectrum.

(c) Take the inverse Fourier transform of the power spectrum and you get the autocorrelation.

2.5 Verification of Results

All the methods produced the same outputs. The output plots are given in the python notebook "acf-codes.ipynb". These results are verified with the standard definition of autocorrelation as convolution of same signal, provide in the verification "verification.ipynb". The output plots are given in the next page.

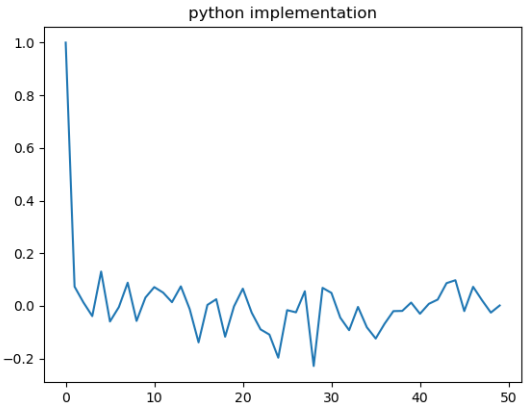

Figure 1:

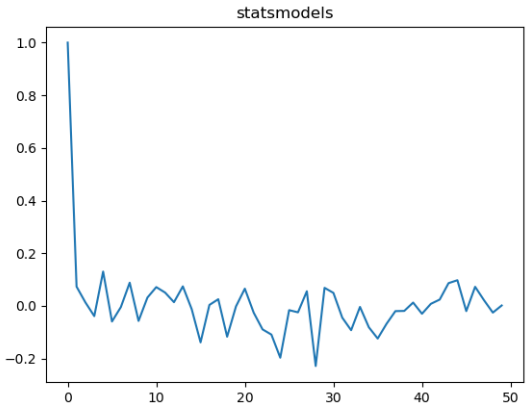

Figure 2:

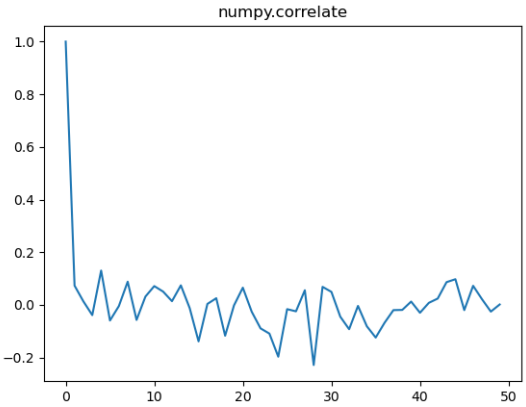

Figure 3:

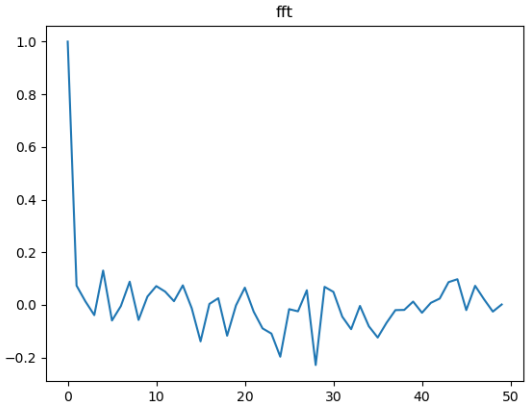

Figure 4:

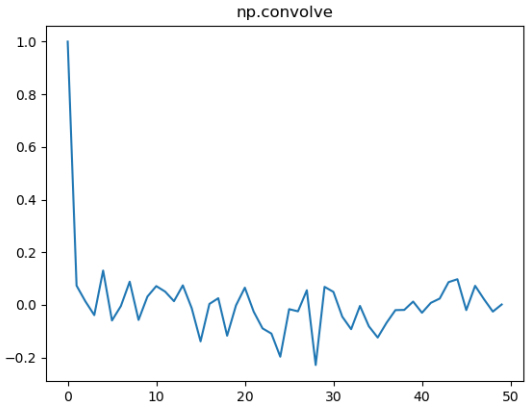

Figure 5: Autocorrelation through convolution

3 Perfomance Comparision Tables Of PI And JETSON

METHOD-1 PYTHON IMPLEMENTATION		
SAMPLES	**PI**	**JETSON**
50K	36 minutes	9 minutes
100K	2.69 hrs	52.67 minutes
200K	limit time exceed	limit time exceed

METHOD-2 STATSMODEL		
SAMPLES	**PI**	**JETSON**
50K	1.58 sec	9.69 sec
100K	2.06 sec	28.90 sec
200K	2.29 sec	1.88 minutes
1M	3.57 sec	51.58 minutes
10M	26.03 sec	limit time exceed
20M	Killed	killed

METHOD-3 NUMPY.CORRELATE		
SAMPLES	**PI**	**JETSON**
50K	4.701 sec	6.80 sec
100K	11.37 sec	27.18 sec
200K	1.89 minutes	1.89 minutes
1M	1.90 minutes	52.96 minutes
10M	limit time exceed	limit time exceed
20M	Killed	killed

METHOD-4 FOURIER TRANSFORM		
SAMPLES	**PI**	**JETSON**
50K	0.08 sec	1.49 sec
100K	0.16 sec	2.70 sec
200K	0.39 sec	5.27 sec
1M	0.51 sec	24.98 sec
10M	44.20 sec	52.19 sec
20M	Killed	killed

4 Conclusion

Overall, RaspberryPi is faster to execute the autocorrelation than Jetson Nano. However, both have limitaions depending on the number of samples taken and the method of implementation. For example, when 20 million samples were taken both output produced "killed".

5 Software

Download the following code

```
https://github.com/ahilan22/fwc/tree/main/jetson-pi
```