Accelerating autoregressive integrated moving average models using sparse matrix representations

Context

Statistical time series models (such as autoregressive integrated moving average, ARIMA) are a staple forecasting method for time series. Even with the advent of more modern forecasting methods (e.g., long-short term memory (LSTM) neural networks and Facebook Prophet), they remain an interesting baseline that has the advantage of being interpretable.

Modern ARIMA implementations rely on Kalman filters for fitting and forecasting purposes (notably because Kalman filters and state-space representations are well mastered mathematical tools and support missing observations). This is also the case for seasonal time series models (i.e., seasonal autoregressive integrated moving average, SARIMA) [Brockwell, 2002] [Durbin, 2012].

Interestingly, state-space models of high-order ARIMA models happen to be sparse (which means that the state transition and observation matrices mostly consist of zero coefficients); sparsity levels get even higher in the case of SARIMA models.

However, despite the sparse nature of most of the state-space matrices, it appears that many implementations (e.g., Python statsmodels) do not leverage it and instead rely on standard dense matrix representations for computations.

Objectives and steps

The goal of this master's thesis is to program a custom, high-performance fit-and-forecast program for ARIMA models, relying on sparse matrix representations to reduce computing load. Steps are *i)* a review of state-space models for ARIMA models and the associated fitting and forecasting methods, *ii)* implementing a fit-and-forecast program that leverages sparse matrix representations and *iii)* compare the performance of the developed program to that of Python's statsmodels (in terms of accuracy and computing time), using real and/or synthetic time series. Very motivated students can go further and *i)* extend the program to SARIMA models, or *ii)* implement the program in CUDA (for graphical processing unit (GPU) programming) or *iii)* rely on template meta-programming for implementing compile-time sparse matrices. The BEAMS-EE department possesses various computing platforms (notably a consumer-grade computing tower with a GPU and a rack server with dual Xeon Gold (64 cores in total) and two RTX A6000 GPUs), which are available to the student.

Student profile

Ideally, the student has experience in C++ programming and is skilled in mathematics (mostly linear algebra, although basic probability theory and optimization theory are important as well). Knowledge of BLAS, LAPACK and/or the Eigen library is a plus. Having followed the course "Microprocessor architectures" is also a (minor) plus.

References

[Brockwell, 2002] Brockwell, Peter J., and Richard A. Davis, eds. *Introduction to time series and forecasting*. New York, NY: Springer New York, 2002.
[Durbin, 2012] Durbin, James, and Siem Jan Koopman. *Time series analysis by state space methods*. Vol. 38. OUP Oxford, 2012.

Contact

Jean-François Determe, jean-francois.determe@ulb.be

Solbosch campus, building U, level. 5, BEAMS-EE department