Embarrassingly Parallel Time Series Analysis for Large Scale Weak Memory Systems

Francois Belletti, Evan Sparks, Michael Franklin, Alexandre M. Bayen

November 19, 2015

Abstract

Second order stationary models in time series analysis are based on the analysis of essential statistics whose computations follow a common pattern. In particular, with a map-reduce nomenclature, most of these operations can be modeled as mapping a kernel that only depends on short windows of consecutive data and reducing the results produced by each computation. This computational pattern stems from the ergodicity of the model under consideration and is often referred to as weak or short memory when it comes to data indexed with respect to time. In the following we will show how studying weak memory systems can be done in a scalable manner thanks to a framework relying on specifically designed overlapping distributed data structures that enable fragmentation and replication of the data across many machines as well as parallelism in computations. This scheme has been implemented for Apache Spark but is certainly not system specific. Indeed we prove it is also adapted to leveraging high bandwidth fragmented memory blocks on GPUs.

Introduction

Classic time series analysis and in particular that of second order stationary processes has been widely studied for several decades and reached a peak of popularity in the years 1980-1990 when computational power became more common and allowed for practical implementation of varied data sets. Monographs such as [?], [?] and [?] offered the opportunity for many researchers to better understand the intricacies of time series analysis. The topics of univariate and monovariate time series analysis were also well covered in [?, ?] and many other publications. Software such as R or Matlab has given the opportunity to many practitioners to examine data, enabling even more applications of time series analysis and contributing to the interest in that field. The amount of research dedicated to time series analysis has been so important in the past decades that we do not claim here to review it holistically. However, it is necessary to contextualize the topic of the present article both in its theoretical aspects and its practical implementation.

The development of distributed programming paradigms [?], file systems [?], databases [?] and in-memory computing engines [?] for the modern commodity datacenter environment has led us to consider new applications and implementations for time series analysis. As opposed to small dataset applications such as those developed in quantitative finance [?], climate studies [?], network traffic analysis [?], hydrology [?], we are now interested in analyzing observations at scale in a data intensive environment such as [?] with a standardized distributed library similar to MLlib [?]. This creates new challenges. In particular, we are dealing with data sitting on a distributed cluster of

machines whose memory layout has an obvious bandwidth bottleneck when it comes to shuffling data across an Ethernet network. This also creates new opportunities. Time series estimation can now leverage recent theoretical developments [?] as well as improvements in convex optimization techniques [?, ?] that can help get likelihood maximization based estimators faster.

In this new programming framework, we will start adapting the implementation of the simplest class of models: linear time series models. These are the stochastic counterparts of deterministic linear systems [?]. Estimating auto-regressive and moving average models (weak memory models) will be the main focus of this document. More advanced topics such as the identification of non linear dynamics [?, ?] as well heteroscedastic systems [?] will be the subject of subsequent work.

This document focuses on the presentation of a new data structure dedicated to the representation of estimation-ready data sets for time series analysis. As opposed to other frameworks such as SparkTS, spark-timeseries or Thunder, the data is partitioned here with respect to time. An overlapping block distributed data structure has been devised which, given an appropriate padding expressed in units of time, enables the computation of M and Z estimators for second order stationary models in an embarrassingly parallel manner. This enables in particular the calibration of multivariate time series models without shuffling observations on a distributed in-memory computing engine.

Part I
Theoretical background on time series analysis, estimators, motivation

Our intent is to provide a generic programming framework for scalable time series analysis in a distributed system. In the following, we review popular time series models that should be supported by the framework and establish the corresponding computational operations the related estimators require. Focus is set on second order stationary models and the the well known ARMA family.

1   Second order stationary time series

We start with the most common time series models. These are models in which observation are regularly spaced and for which there are no missing values. Practitioners of time series analysis are familiar with missing data, outliers and irregularly spaced timestamps. In such cases, an interpolation technique (linear interpolation or last-observation-carried-forward for instance) is often used in order to align observations on a regular time index grid. We focus on time series $(X_t)_{t \in \mathbb{Z}}$ where each observation belongs in $\mathbb{R}^d$.

1.1   Second order stationarity

Second order stationary processes are common in time series analysis. Models in this family are simple, practical to estimate and yet have strong predictive power for a vast range of data sets related to economics, finance, industrial systems, data center monitoring and the climate.

Definition: Second order stationary time series A time series $(X) = (X_t)_{t \in \mathbb{Z}}$ is second order stationary if there exists an auto-covariance function $\gamma : \mathbb{Z} \to \mathbb{R}^{d \times d}$ so that for any value of t,

$$\text{Cov}(X_t, X_{t+h}) = \gamma^X(h).$$

White noise is an example of second order stationary time series where the auto-covariance function is 0 everywhere except for $h = 0$.

Definition: lag operator L Let $L : (\mathbb{R}^d)^{\mathbb{Z}} \to (\mathbb{R}^d)^{\mathbb{Z}}$ the operator such that $L((X_t)_{t \in \mathbb{Z}}) = (X_{t-1})_{t \in \mathbb{Z}}$.

1.2 Constant volatility linear time series models

A first family of models is concerned with modeling observations as the output of a linear system with constant variance perturbations.

Definition: multidimensional white noise A process $(\varepsilon_t)_{t \in \mathbb{Z}}$ in $\mathbb{R}^d$ is a multidimensional white noise if

$$\forall t \in \mathbb{Z}, \, E(\varepsilon_t) = 0$$
$$\forall t \in \mathbb{Z}, \, E(\varepsilon_t \varepsilon_t^T) = \Sigma_\varepsilon$$

and

$$\forall t, s \in \mathbb{Z} : t \neq s, \, E(\varepsilon_t \varepsilon_s^T) = 0.$$

In the following we assume white noise processes are always non-degenerate, i.e., we always have $E(\varepsilon_t \varepsilon_t^T) = \Sigma_\varepsilon$ definite positive.

Definition: Auto-regressive models (AR) A (centered) multivariate order p auto-regressive time series is defined by p matrices $(A_k)_{k \in \{1 \ldots p\}} \in \mathbb{R}^{d \times d}$ and a white noise process $(\varepsilon_t)_{t \in \mathbb{Z}}$ in $\mathbb{R}^d$ with variance $\Sigma_\varepsilon \in (\mathbb{R}^{d \times d})$ such that, for any value of t in $\mathbb{Z}$,

$$X_t = A_1 X_{t-1} + \ldots + A_p X_{t-p} + \varepsilon_t.$$

This equation can be rewritten in reduced form with the lag operator. Let $A(z) = I - A_1 z - \ldots - A_p z^p$ be the companion polynomial of the equation. A short hand for the AR equation above is

$$A(L)X = \varepsilon.$$

Equivalently, a Linear Time Invariant (LTI) system formulation of these equations is:

$$
\begin{pmatrix} X_t \\ X_{t-1} \\ \vdots \\ X_{t-p+1} \end{pmatrix} = \begin{pmatrix} A_1 & A_2 & \cdots & \cdots & A_p \\ I_d & 0 & & & \\ & I_d & \ddots & & \\ & & \ddots & \ddots & \\ & & & I_d & 0 \end{pmatrix} \begin{pmatrix} X_{t-1} \\ X_{t-2} \\ \vdots \\ X_{t-p} \end{pmatrix} + \begin{pmatrix} \varepsilon_t \\ 0 \\ \vdots \\ 0 \end{pmatrix}.
$$

In the following we will mostly use the first AR formulation as the LTI formulation has a degenerate noise structure (the noise covariance matrix is not full rank).

3

Definition: Moving average models (MA)   A (centered) multivariate order q moving average time series is defined by q matrices $(B_k)_{k \in \{1...q\}} \in \mathbb{R}^{d \times d}$ and a white noise process $(\varepsilon_t)_{t \in \mathbb{Z}}$ in $\mathbb{R}^d$ with variance $\Sigma_\varepsilon \in (\mathbb{R}^{d \times d})$ such that, for any value of t in $\mathbb{Z}$,

$$X_t = \varepsilon_t + B_1 \varepsilon_{t-1} + \ldots + B_q \varepsilon_{t-q}.$$

Letting $B(z) = I + B_1 z - \ldots + B_q z^q$ the companion polynomial of the equation. A short hand for the MA equation above is

$$B(L) X = \varepsilon.$$

Definition: Auto-regressive moving average models (ARMA)   A (centered) multivariate order p, q auto-regressive moving average time series is defined by p matrices $(A_k)_{k \in \{1...p\}} \in \mathbb{R}^{d \times d}$ and q matrices $(B_k)_{k \in \{1...q\}} \in \mathbb{R}^{d \times d}$ and a white noise process $(\varepsilon_t)_{t \in \mathbb{Z}}$ in $\mathbb{R}^d$ with variance $\Sigma_\varepsilon \in diag(\mathbb{R}^{d \times d})$ such that, for any value of t in $\mathbb{Z}$,

$$X_t = A_1 X_{t-1} + \ldots + A_p X_{t-p} + \varepsilon_t + B_1 \varepsilon_{t-1} + \ldots + B_q \varepsilon_{t-q}.$$

Such a model corresponds to a perturbed LTI system whose perturbations are auto-correlated.

Letting $A(z) = I - A_1 z - \ldots - A_p z^p$ and $B(z) = I + B_1 z - \ldots + B_q z^q$ the companion polynomials of the equation. A short hand for the ARMA equation above is

$$A(B) X = B(L) \varepsilon.$$

1.3   Identification, causality and invertibility

In the following, we will consider the conditions that guarantee stability of the model and the fact that the equations above have one and only solution.

1.3.1   Causal models:

If $\forall z : |z| < 1$, $det(A(z)) \neq 0$ i.e. if the matrix

$$\begin{pmatrix} A_1 & A_2 & \cdots & \cdots & A_p \\ I_d & 0 & & & \\ & I_d & \ddots & & \\ & & \ddots & \ddots & \\ & & & I_d & 0 \end{pmatrix}$$

has its spectrum strictly bounded in absolute value by 1, the ARMA equation has one and only stationary solution. This solution is said to be causal. Such an ARMA equation can be interpreted as equivalent to a stable LTI which is perturbed by auto-correlated noise.
In the following we will only consider causal models.

1.3.2 Invertible models:

If $\forall z : |z| < 1$, $det\,(B\,(z)) \neq 0$ i.e. if the matrix

$$\begin{pmatrix} -B_1 & -B_2 & \cdots & \cdots & -B_q \\ I_d & 0 & & & \\ & I_d & \ddots & & \\ & & \ddots & \ddots & \\ & & & I_d & 0 \end{pmatrix}$$

has its spectrum strictly bounded in absolute value by 1, for a stationary solution to the ARMA equations (X) there exists a unique white noise process (ε) such that $A\,(L)\,X = B\,(L)\,\varepsilon$.

1.4 Integrated processes and differentiation

Definition: differentiated time series For any time series $(X_t)_{t\in\mathbb{Z}}$ one may define its differentiated counterpart as $\triangle\left((X_t)_{t\in\mathbb{Z}}\right) = (X_{t+1} - X_t)_{t\in\mathbb{Z}}$.

In the case of actual and finite length data, we opt for the convention: $\triangle\left((X_t)_{t\in\{1...N\}}\right) = (X_{t+1} - X_t)_{t\in\{1...N-1\}}$.

Definition: integrated processes Formally, a time series $(X_t)_{t\in\mathbb{Z}}$ is said to be integrated of order I if $\triangle^d\,(X)$ is not second order stationary whenever $d < I$ and $\triangle^I\,(X)$ is second order stationary.

Brownian motions are famous examples of order 1 integrated processes (there difference process is a white noise).

2 Estimators of sufficient statistics of second order stationary time series

In the following we review known estimators for multivariate time series in order to highlight the similarity of their computational structure. We are concerned with a theoretical process (X) and have $(X_t)_{t\in\{1...N\}}$ consecutive observations.

2.1 Estimators of interest

The following section goes through all the sufficient statistics to estimate a second order stationary process. In particular, it is noteworthy that the causal solution to an ARMA equation has a covariance structure that is entirely determined by the parameters of the equation [?] (that is $A_1, \ldots, A_p$, $B_1, \ldots, B_q$ and Σ).

2.1.1 Mean

A consistent unbiased estimator for the mean of a second order time series (X) is

$$\widehat{\mu^X}\left((X_t)_{t\in\{1...N\}}\right) = \frac{1}{N}\sum_{k=1}^{N} X_k.$$

This estimator is asymptotically normal with variance decaying with $\frac{1}{N}$ rate.

In the following we assume that the mean has been estimated and accounted for and (X) is therefore centered.

2.1.2 Auto-covariance

A consistent unbiased estimator for the auto-covariance matrix at lag $h \in \mathbb{Z}$ for a centered second order stationary time series is

$$\widehat{\gamma^X(h)}\left((X_t)_{t \in \{1 \ldots N\}}\right) = \frac{1}{N-h-1} \sum_{k=1}^{N-h} X_k X_{k+h}^T.$$

This estimator is asymptotically normal with variance decaying with $\frac{1}{N}$ rate.

2.1.3 Auto-correlogram

Definition: auto-correlogram of a second order stationary time series Let $h \in \mathbb{Z}$, order h auto-correlation is

$$\rho_h^X = \mathrm{Cor}\left(X_t, X_{t+h}\right) = \mathrm{diag}\left(\gamma^X(0)\right)^{-\frac{1}{2}} \gamma^X(h) \, \mathrm{diag}\left(\gamma^X(0)\right)^{-\frac{1}{2}}.$$

Auto-correlation estimator: The estimator

$$\widehat{\rho_h^X} = \mathrm{diag}\left(\widehat{\gamma^X(0)}\right)^{-\frac{1}{2}} \widehat{\gamma^X(h)} \, \mathrm{diag}\left(\widehat{\gamma^X(0)}\right)^{-\frac{1}{2}}$$

is asymptotically convergent, it is in fact asymptotically normal with variance decaying with $\frac{1}{N}$ rate.

Definition: partial auto-correlogram of a second order stationary time series Let $H_X^{t-p,t}$ the span of the random variables $\{X_{t-p}, \ldots, X_t\}$ and $P_{H_X^{p,t}}$ the corresponding orthogonal projection. Let $\left(U_j^{(p)}\right)_{j=1}^p$ be defined by

$$P_{H_X^{t-p,t-1}} X_t = \sum_{j=1}^p U_j^{(p)} X_{t-j}.$$

$U_p^{(p)}$ is the partial auto-correlation matrix of order p, we will denote it $\kappa^X(p)$.

Property: from auto-correlation to partial auto-correlation The projection matrices above solve the following linear system of equations:

$$\begin{bmatrix} \gamma^X(0) & \cdots & \gamma^X(-(p-1)) \\ \vdots & \ddots & \vdots \\ \gamma^X(p-1) & \cdots & \gamma^X(0) \end{bmatrix} \begin{bmatrix} \left(U_1^{(p)}\right)^T \\ \vdots \\ \left(U_p^{(p)}\right)^T \end{bmatrix} = \begin{bmatrix} \gamma^X(1) \\ \vdots \\ \gamma^X(p) \end{bmatrix}.$$

Proof:

$P_{H_X^{t-p,t-1}}$ is an orthogonal projector therefore for any $j \in \{1 \ldots p\}$,

$$E\left[\left(X_t - P_{H_X^{t-p,t-1}} X_t\right) X_{t-j}^T\right] = 0$$

Therefore,

$$E\left[\begin{pmatrix} X_{t-1} \\ \vdots \\ X_{t-p} \end{pmatrix} (X_t)^T\right] = E\left[\begin{pmatrix} X_{t-1} \\ \vdots \\ X_{t-p} \end{pmatrix} \left(P_{H_X^{t-p,t-1}} X_t\right)^T\right] = E\left[\begin{pmatrix} X_{t-1} \\ \vdots \\ X_{t-p} \end{pmatrix} \begin{pmatrix} X_{t-1}^T & \cdots & X_{t-p}^T \end{pmatrix} \begin{pmatrix} \left(U_1^{(p)}\right)^T \\ \vdots \\ \left(U_p^{(p)}\right)^T \end{pmatrix}\right].$$

Partial auto-correlation estimator: An estimator $\widehat{\kappa^X(p)}$ can be obtained by inverting the linear system above where the actual auto-covariance is replaced by $\widehat{\gamma}$:

$$\begin{bmatrix} \widehat{\gamma^X}(0) & \cdots & \widehat{\gamma^X}(-(p-1)) \\ \vdots & \ddots & \vdots \\ \widehat{\gamma^X}(p-1) & \cdots & \widehat{\gamma^X}(0) \end{bmatrix} \begin{bmatrix} \vdots \\ \vdots \\ \left(\widehat{\kappa^X(p)}\right)^T \end{bmatrix} = \begin{bmatrix} \widehat{\gamma^X}(1) \\ \vdots \\ \widehat{\gamma^X}(p) \end{bmatrix}.$$

It is asymptotically unbiased and asymptotically normal with variance decaying with a $\frac{1}{N}$ rate.

3 AR, MA, ARMA fitting by frequentist methods

In the following we assume (X) is a centered second order stationary process.

3.1 Determining the order of an AR, MA or ARMA model by frequentist methods

It is possible to choose an appropriate value of p when estimating an AR model simply by computing the partial auto-correlation of the process. As soon as the partial auto-correlation at lag h is not significantly different from 0, an appropriate choice for p is $h-1$. Indeed, for an AR model of order p, partial auto-correlation $\kappa^X(h)$ cancels out as soon as $h > p$.

Similarly, if one considers a MA model of order q, the auto-correlation function $\gamma^X(h)$ is zero whenever $h > q$. Therefore, value of h after which the auto-correlation function is no longer significantly different from zero yields an indicator of the value of q one should choose prior to estimating the model.

For ARMA models, the analysis is more involved but only relies on estimates of auto-covariance as well. In this case, it is more common in practice to choose cutoff values for p and q based on a Bayesian information criterion (AIC or BIC).

3.2 AR estimation based on Yule-Walker equations:

Assuming

$$X_t = A_1 X_{t-1} + \ldots + A_p X_{t-p} + \varepsilon_t$$

the Yule-Walker equations give

$$
\begin{bmatrix}
\gamma^X(0) & \cdots & \gamma^X(-(p-1)) \\
\vdots & \ddots & \vdots \\
\gamma^X(p-1) & \cdots & \gamma^X(0)
\end{bmatrix}
\begin{bmatrix}
A_1^T \\
\vdots \\
A_p^T
\end{bmatrix}
=
\begin{bmatrix}
\gamma^X(1) \\
\vdots \\
\gamma^X(p)
\end{bmatrix}
$$

or equivalently

$$
\begin{bmatrix} A_1 & \cdots & A_p \end{bmatrix}
\begin{bmatrix}
\left(\gamma^X(0)\right)^T & \cdots & \left(\gamma^X(p-1)\right)^T \\
\vdots & \ddots & \vdots \\
\left(\gamma^X(-(p-1))\right)^T & \cdots & \left(\gamma^X(0)\right)^T
\end{bmatrix}
=
\begin{bmatrix} \left(\gamma^X(1)\right)^T & \cdots & \left(\gamma^X(p)\right)^T \end{bmatrix}.
$$

Solving this block Toeplitz linear system with the auto-covariance estimators given above then yields the least square estimates of $A_1^T, \ldots, A_p^T$. Right multiplying the equation above by X_t^T and computing the expectation gives an estimate of the diagonal variance matrix of the noise process:

$$\Sigma_\varepsilon = \gamma_0^X - A_1 \gamma_{-1}^X - \ldots - A_p \gamma_{-p}^X$$

The interesting point here, from a computational standpoint, is the fact that auto-covariance estimates for $h = 0 \ldots p-1$ are sufficient to obtain estimates for the parameters of the model. In the univariate case this comes down to inverting a Toeplitz matrix and is practically achieved thanks to the well known Durbin-Levinson algorithm [?] with $O\left(p^2\right)$ time complexity.

The multivariate case is more in-line with the kind of large scale analytics programming paradigm we discuss here. Indeed, what is key to the present approach is to be able to fit AR models not specifically with a large order p but more with a large number of dimensions d. One is therefore interested in solving such a system with large Toeplitz blocks (dimension d) and small order in comparison ($p \ll d$). Akaike offered a recursive method to solve such a system with $O\left(p^2 \times d\right)$ extra space and $O\left(p^2 \times d^3\right)$ in [?].

3.2.1 Issues when d becomes very large:

The fact that, in this algorithm one has to invert matrices of size (d, d) becomes problematic whenever d becomes large (10^5 or more). The time complexity of the block Toeplitz inversion procedure of Akaike is cubic with respect to d which means practically that even on modern GPUs capable of 1 TFLOPs, it would generally take at least 12 days to invert a 10^6 row square matrix. Therefore we show how to conduct multivariate analysis when d is high and the system under study features spatial stationarity thanks to a Bayesian approach.

3.3 MA estimation based on the innovation algorithm:

We want to estimate a model of the form

$$X_t = \varepsilon_t + B_1 \varepsilon_{t-1} + \ldots + B_q \varepsilon_{t-q}$$

Let us assume that their exist $(\Theta_{m,n})_{m,n\in\{1...q\}}$ such that for for any $m \in \{1...q\}$.

$$P_{H_X^{1,m}} X_{m+1} = \begin{cases} 0 & \text{if } m = 0 \\ \sum_{j=1}^{m} \Theta_{m,j} \left(X_{m-j+1} - P_{H_X^{1,m-j}} X_{m-j+1} \right) & \text{otherwise} \end{cases}.$$

The sequence $\left(X_{m-j+1} - P_{H_X^{1,m-j}} X_{m-j+1} \right)_{j\in\{1...m\}}$ is a set of orthogonal vectors obtained by a Gram-Schmidt orthonormalization procedure. Indeed, it corresponds to the series of innovations of the time series. This implies, by orthogonality and decomposition on the Gram-Schmidt basis,

$$E\left[\left(P_{H_X^{1,m}} X_{m+1} \right) \left(X_{m-j+1} - P_{H_X^{1,m-j}} X_{m-j+1} \right)^T \right] = \Theta_{m,j} \Sigma_{m-j}$$

where Σ_{m-j} is the variance matrix of the corresponding innovation process (perturbations to the linear model). $X_{m+1} - P_{H_X^{1,m}} X_{m+1}$ is orthogonal to $X_{m-j+1} - P_{H_X^{1,m-j}} X_{m-j+1}$ therefore

$$E\left(X_{m+1} \left(X_{m-j+1} - P_{H_X^{1,m-j}} X_{m-j+1} \right)^T \right) = \Theta_{m,j} \Sigma_{m-j}.$$

Substituting $P_{H_X^{1,m-j}} X_{m-j+1}$ by $\sum_{i=1}^{m-j} \Theta_{m-j,j} \left(X_{m-j-i+1} - P_{H_X^{1,m-j-i}} X_{m-j-i+1} \right)$ we get

$$E\left(X_{m+1} \left(X_{m-j+1}^T - \sum_{i=1}^{m-j} \left(X_{m-j-i+1} - P_{H_X^{1,m-j-i}} X_{m-j-i+1} \right)^T \Theta_{m-j,j}^T \right) \right) = \Theta_{m,j} \Sigma_{m-j}.$$

And, therefore,

$$\gamma_{-j}^X - \sum_{i=1}^{m-j} \Theta_{m,j+i} \Sigma_{m-j-i} \Theta_{m-j,j}^T = \Theta_{m,j} \Sigma_{m-j}.$$

Substituting j by $m - j$ gives

$$\gamma_{j-m}^X - \sum_{i=1}^{j} \Theta_{m,m-j+i} \Sigma_{j-i} \Theta_{j,m-j}^T = \Theta_{m,m-j} \Sigma_j$$

and finally substituting $j - i$ by i yields

$$\gamma_{j-m}^X - \sum_{i=0}^{j-1} \Theta_{m,m-i} \Sigma_i \Theta_{j,m-j}^T = \Theta_{m,m-j} \Sigma_j.$$

Finally, Pythagora's theorem for orthogonal projections implies that

$$\Sigma_m = \text{Var}(X_{t+1}) - \text{Var}\left(P_{H_X^{1,t}} X_{t+1} \right) = \gamma_0^X - \sum_{i=0}^{m-1} \Theta_{m,m-i} \Sigma_i \Theta_{m,m-i}^T.$$

Based on the estimates of γ^X given by the averaging procedure above, a recursive procedure, starting by $\widehat{\Sigma_0} = \widehat{\gamma_0^X}$ yields consistent estimates for $\widehat{\Sigma_m}$ and $\widehat{\Theta}_1, \ldots, \widehat{\Theta}_q$. In the case of an order q MA model, $B_1, \ldots, B_q = \Theta_1, \ldots, \Theta_q$ and therefore the we get the estimates of the parameters of the model. The complexity of the algorithm here is $O(p^2 d^3)$.

Recursive procedure: One starts with $\Sigma_0 = \widehat{\gamma}_0^X$ and then for $m \in \{1 \dots q\}$ computes

$$\forall j \in \{0 \dots m-1\}, \; \widehat{\Theta}_{m,m-j} = \left[\widehat{\gamma}_{j-m}^X - \sum_{i=0}^{j-1} \widehat{\Theta}_{m,m-i} \widehat{\Sigma}_i \widehat{\Theta}_{j,j-i}^T \right] \widehat{\Sigma}_j^{-1}$$

and

$$\widehat{\Sigma}_m = \widehat{\gamma}_0^X - \sum_{i=0}^{m-1} \Theta_{m,m-i} \Sigma_i \Theta_{m,m-i}^T.$$

3.4   ARMA model estimation:

We are now concerned with estimating the parameters of

$$X_t = A_1 X_{t-1} + \dots + A_p X_{t-p} + \varepsilon_t + B_1 \varepsilon_{t-1} + \dots + B_q \varepsilon_{t-q}.$$

We assume in the following that the model is causal. This implies in particular the existence of matrices $(\Psi_j)_{j=0}^{+\infty}$ such that $X_t = \sum_{j=0}^{\infty} \Psi_j \varepsilon_{t-j}$. As $(\varepsilon_t)_{t \in \mathbb{Z}}$ is a white noise process, necessarily

$$\begin{cases} \Psi_0 & = I \\ \Psi_j & = B_j + \sum_{i=1}^{\min(j,p)} A_i \Psi_{j-i} \end{cases}$$

where, by convention, $B_j = 0$ whenever $j > q$ and $A_j = 0$ whenever $j > p$. By construction, $(\Psi_j)_{j=1}^{p+q}$ can be estimated thanks to the innovation estimates $\left(\widehat{\Psi}_{p+q,j} \right)_{j=1}^{p+q}$ provided by the innovation algorithm above. Then one has

$$\widehat{\Psi}_{p+q,j} = \widehat{B}_j + \sum_{i=1}^{\min(j,p)} \widehat{A}_i \widehat{\Psi}_{p+q,j-i}, \; \forall j \in \{1 \dots p+q\}.$$

As by convention, $\forall j > p$, $B_j = 0$, necessarily, the estimates $\left(\widehat{A}_j \right)_{j=1}^p$ solve the following linear system:

$$\begin{bmatrix} \widehat{\Psi}_{p+q,\,q}^T & \widehat{\Psi}_{p+q,\,q-1}^T & \cdots & \widehat{\Psi}_{p+q,\,q+1-p}^T \\ \widehat{\Psi}_{p+q,\,q+1}^T & \widehat{\Psi}_{p+q,\,q}^T & & \widehat{\Psi}_{p+q,\,q+2-p}^T \\ \vdots & & \ddots & \vdots \\ \widehat{\Psi}_{p+q,\,p+q-1}^T & \widehat{\Psi}_{p+q,\,p+q-2}^T & \cdots & \widehat{\Psi}_{p+q,\,q}^T \end{bmatrix} \begin{bmatrix} \widehat{A}_1^T \\ \vdots \\ \widehat{A}_p^T \end{bmatrix} = \begin{bmatrix} \widehat{\Psi}_{p+q,\,q+1}^T \\ \vdots \\ \widehat{\Psi}_{p+q,\,p+q}^T \end{bmatrix}$$

Once this block Toeplitz system has been solved, the estimates $\widehat{B}_j$ can be determined thanks to

$$\widehat{B}_j = \widehat{\Psi}_{p+q,\,j} - \sum_{i=1}^{\min(j,p)} \widehat{A}_i \widehat{\Psi}_{p+q,\,j-i}, \; \forall j \in \{1, \dots q\}.$$

An estimate of the diagonal noise variance matrix is given by

$$\widehat{\Sigma}_{p+q} = \widehat{\gamma}_0^X - \sum_{i=0}^{p+q-1} \widehat{\Psi}_{p+q,\,p+q-i} \widehat{\Sigma}_i \widehat{\Psi}_{p+q,\,p+q-i}^T.$$

The complexity of the algorithm is $O\left((p+q)^2 d^3 \right)$.

10

4 Predictions with linear time series models

Linear models for time series, however often very simplistic, offer linear predictors for the next events to occur given a series of previous observations. Linear predictors are simple to set up and offer good guarantees on the results they provide in the form of confidence intervals. This principle can also be extended to that of featurized predictions with a linear model that feeds on non-linear features. Predictions leverage parallelism with respect to process dimensions straightforwardly and are computed in an iterative fashion with respect to time. There is no contribution here in that regard except in that we highlight that predictions in the general case an ARMA process only depend on short range previously observed values and therefore prove corresponding computations feature weak memory.

4.1 Predictions in the AR case

The auto-regressive family of models is the simplest, the one step ahead predictor of X_t, that will be denoted $\overrightarrow{X}_t^1$ is trivially

$$\overrightarrow{X}_t^1 = A_1 X_t + \ldots + A_p X_{t-p+1}$$

and predictions more than one step ahead are obtained in a recursive fashion by re-injecting shorter range projections into the linear system above. The variance of predictions can then be computed based on Σ_ε (see [?] for details).

4.2 Predictions in the ARMA case

Moving average components introduce a supplementary difficulty in predicting values based on passed observations. The aim of this section is to show that predictions can be evaluated based on a recursive procedure which, at each step, only considers previously observed values on a short range.

In order to forecast an ARMA process one runs the innovation algorithms on the observations so as to obtain an estimate of the innovations. There exists a series of projection matrices $(\Theta_{t,t'})_{t \in \{0,T\}, t' \leq t}$ such that

$$\begin{cases} \forall t \in \{0, \max{(p,q)} - 1\} & \overrightarrow{X}_t^1 = \sum_{t'=1}^{t} \Theta_{t,t'} \left(X_{t+1-t'} - \overrightarrow{X}_{t+1-t'}^1 \right) \\ \forall t \in \{\max{(p,q)}, T\} & \overrightarrow{X}_t^1 = A_1 X_t + \ldots + A_p X_{t-p+1} + B_1 \left(X_t - \overrightarrow{X}_{t-1}^1 \right) + \ldots + B_q \left(X_{t-q+1} - \overrightarrow{X}_{t-q}^1 \right) \end{cases}.$$

Therefore, at each time-step t, provided the innovation projection matrices have been computed (which has been done iteratively from 0 to t), only $\max{(p,q)}$ observations and forecasts need to be taken into account. This algorithm can be run in a streaming fashion. This algorithm can be run in an approximate parallel manner in the case of stable models in which the importance of initialization errors decays exponentially.

5 AR, MA, ARMA estimation by Bayesian methods

Let us first focus on the estimation of AR models by Bayesian methods and prove they rely on weak memory computations.

5.1 Bayesian estimation of AR models

At each time-step one considers an equation of the form

$$X_t = A_1 X_{t-1} + \ldots + A_p X_{t-p} + \varepsilon_t.$$

We assume the errors have a parametric distribution $f(\varepsilon_t, \vartheta) = \frac{1}{Z(\vartheta)} \exp(H(\varepsilon_t, \vartheta))$ where ϑ is a set of parameters in Θ (which we assume is convex), H is a function from $\mathbb{R}^d \times \Theta$ onto $\mathbb{R}$ and Z is a function from Θ onto $\mathbb{R}^{++}$. We further assume that f is log-strongly-concave with respect to its first argument. One will typically consider a centered Gaussian distribution for the white noise ε_t with variance Σ_ε. In other words

$$f(\varepsilon_t, \Sigma_\varepsilon) = (2\pi)^{-\frac{d}{2}} \det(\Sigma_\varepsilon)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\varepsilon_t^T \Sigma_\varepsilon^{-1} \varepsilon_t\right).$$

5.1.1 Iterative maximum conditional likelihood based estimation

In this section, one assumes we are given a series of samples $(X_1, \ldots, X_N)$ where $N > p$. We want to optimize the likelihood of samples considering that the first p observations are not perturbed $(\varepsilon_1 = 0, \varepsilon_2 = 0, \ldots, \varepsilon_p = 0)$. The likelihood function decomposes as follows:

$$\mathcal{L}(X_1, \ldots, X_N, A_1, A_2, \ldots, A_p, \vartheta) = \mathcal{L}(X_1, \ldots, X_p, A_1, \ldots, A_p, \vartheta) \prod_{t=p+1}^{N} f(X_t - A_1 X_{t-1} - \ldots - A_p X_{t-p}, \vartheta)$$

The log-likelihood is therefore

$$\log \mathcal{L}(X_1, \ldots, X_N, A_1, A_2, \ldots, A_p, \vartheta) = \log \mathcal{L}(X_1, \ldots, X_p, A_1, \ldots, A_p, \vartheta)$$

$$+ \sum_{t=p+1}^{N} \log f(X_t - A_1 X_{t-1} - \ldots - A_p X_{t-p}, \vartheta)$$

Our aim here is to find ϑ and $A_1, \ldots, A_p$ so as to maximize $\log \mathcal{L}(X_1, \ldots, X_N, A_1, A_2, \ldots, A_p, \vartheta)$. In the conditional likelihood framework we assume $X_1, \ldots, X_p$ are known without perturbations and therefore the problem comes down to

$$\max_{A_1, A_2, \ldots, A_p, \vartheta} \log \mathcal{L}(X_1, \ldots, X_p, A_1, \ldots, A_p, \vartheta) + \sum_{t=p+1}^{N} \log f(X_t - A_1 X_{t-1} - \ldots - A_p X_{t-p}, \vartheta)$$

Without prior knowledge on $(X_1, \ldots, X_p)$, $(A_1, \ldots, A_p)$ or Θ, the maximum likelihood problem can be rewritten as

$$\max_{A_1, A_2, \ldots, A_p, \vartheta} \sum_{t=p+1}^{N} \log f(X_t - A_1 X_{t-1} - \ldots - A_p X_{t-p}, \vartheta).$$

The objective is strongly concave with respect ϑ and $(A_1, \ldots, A_p)$ separately. An alternate maximization procedure therefore yields an argument-wise maximum (but not necessarily a global optimum).

5.1.2 Iterative un-conditional likelihood based estimation

In this setting we consider $(X_1, \ldots, X_p)$ as unknown. We also restrict to the case in which f is a Gaussian distribution with variance Σ_ε. In such a context, one can rewrite each X_t as an infinite linear combination of $(\varepsilon_s)_{s \leq t}$. The variables $(\varepsilon_s)_{s \leq t}$ are independent Gaussian vectors and therefore the infinite linear combination is also a Gaussian variable. Let Γ_p^0 the variance of $\begin{pmatrix} X_1 \\ \vdots \\ X_p \end{pmatrix}$. The problem becomes that of finding

$$
\max_{X_1, X_2, \ldots, X_p, A_1, A_2, \ldots, A_p, \vartheta} \begin{array}{c} \frac{1}{2} \log \left(\det \left(\left(\Gamma_p^0 \right)^{-1} \right) \right) - \left(X_1^T, \ldots, X_p^T \right) \left(\Gamma_p^0 \right)^{-1} \begin{pmatrix} X_1 \\ \vdots \\ X_p \end{pmatrix} \\ + \frac{N-p}{2} \log \left(\det \left(\left(\Sigma_\varepsilon^{-1} \right) \right) \right) \\ - \sum_{t=p+1}^{N} \left(X_t - A_1 X_{t-1} - \ldots - A_p X_{t-p} \right)^T \left(\Sigma_\varepsilon \right)^{-1} \left(X_t - A_1 X_{t-1} - \ldots - A_p X_{t-p} \right) \end{array}.
$$

It is somewhat similar yet somewhat more complex than the conditional likelihood based estimator.

5.1.3 First order methods based resolution

For strongly concave and Lipschitz gradient optimization problems, gradient ascent yields an exponential rate of convergence. The gradient of the conditional likelihood problem decomposes as

$$
\sum_{t=p+1}^{N} \nabla \log f \left(\vartheta, X_t - A_1 X_{t-1} - \ldots - A_p X_{t-p} \right).
$$

What is remarkable there is that only local data (namely $(X_t, \ldots X_{t-p})$ is needed to compute the term corresponding to datum X_t.

For strongly concave and Lipschitz gradient optimization problems consisting of a large sum, stochastic gradient descent has a squared L_2 error that converges in $\frac{1}{n \text{ iterations}}$ to 0 with an appropriate hyperbolically decreasing step size. In a big data context, if N is so high that a holistically computation of the gradient is too computationally expensive then one should use a stochastic gradient method in order to estimate their model.

One should definitively avoid second order methods here if $d \sim 10^4$ and choose a deterministic or stochastic first order optimization method. Practically this requires to sample out a certain value of t in $\{p+1 \ldots N\}$ and compute

$$
\nabla_{A_1, \ldots, A_p} \log f \left(\vartheta, X_t - A_1 X_{t-1} - \ldots - A_p X_{t-p} \right)
$$

and $\nabla_\vartheta \log f \left(\vartheta, X_t - A_1 X_{t-1} - \ldots - A_p X_{t-p} \right)$. There again, only local data is needed to compute the gradient contribution corresponding to X_t.
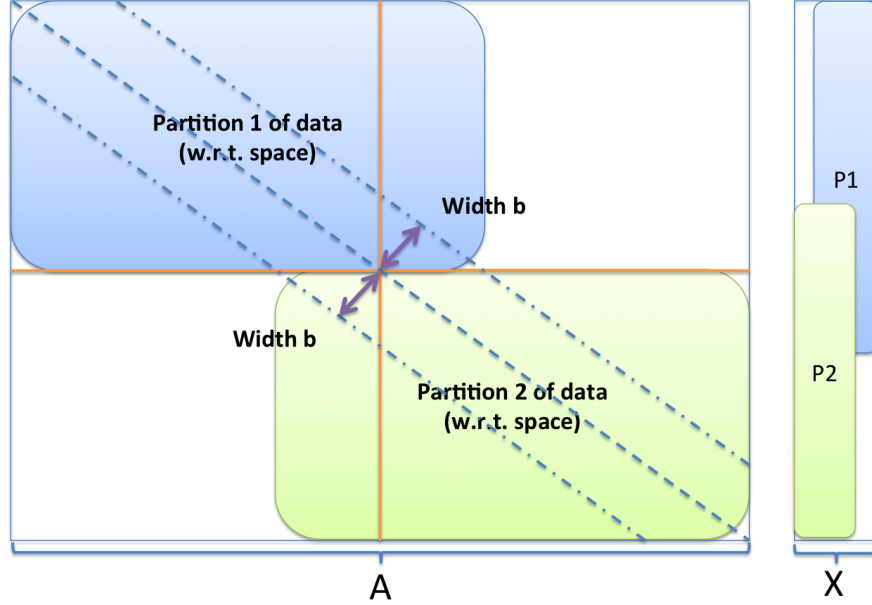
Figure 1: Banded A matrix.

6 When models become very highly dimensional

Let us come back to an order p auto-regressive model with d spatial dimensions where $d \gg p$. The Yule-Walker equations only yield estimates that will require the inversion of a size d square matrix, this is not a scalable solution with respect to d.

Let us assume the order of the model is 1 (one can always rewrite an order p model in this manner):

$$X_{t+1} = AX_t + \varepsilon_t$$

where A is sparse and rearranged so as to be a width b banded matrix with $b \ll d$. Such a sparsity pattern is quite frequent and arises for instance in numerical differentiation schemes. It is illustrated in Figure **??**.

6.1 Efficient one-step-ahead prediction with spatially sparse models

If one observes X_t at a given timestamp t, then the best linear predictor of X_{t+1} is

$$\widehat{X}_{t+1} = AX_t.$$

Let us assume we split A into a row partitioned matrix as follows:

$$
A = \begin{bmatrix}
A_{1,0} & A_{1,1} & & & \\
A_{2,-1} & A_{2,0} & A_{2,1} & & \\
& A_{3,-1} & A_{3,0} & A_{3,1} & \\
& & \ddots & \ddots & A_{P-1,1} \\
& & & A_{P,-1} & A_{P,0}
\end{bmatrix}.
$$

For any $i \in \{1 \ldots P\}$ let us denote P_i the set of indices spanned by the rows of $A_{i,0}$ and P_i^+ the set of indices spanned by $A_{i,-1}$, $A_{i,0}$, $A_{i,1}$.

The class of sparse models considered above is a linear specific case belonging to the family

$$
X_{t+1} = \begin{bmatrix}
A_\vartheta^1 \left(X_t^{P_1^+} \right) \\
A_\vartheta^2 \left(X_t^{P_2^+} \right) \\
\vdots \\
A_\vartheta^{P-1} \left(X_t^{P_{P-1}^+} \right) \\
A_\vartheta^P \left(X_t^{P_P^+} \right)
\end{bmatrix} + \varepsilon_t
$$

where $\left(A_\vartheta^i \right)_{i \in \{1 \ldots P\}}$ is a family of functions from $\mathbb{R}^{|P_i^+|}$ onto $\mathbb{R}^{|P_i|}$ parametric by ϑ.

An efficient unbiased estimator of the one-step-ahead predicted value can be written in a row partitioned manner:

$$
\widehat{X}_{t+1} = \begin{bmatrix}
\widehat{X_t^{P_1}} \\
\widehat{X_t^{P_2}} \\
\vdots \\
\widehat{X_t^{P_{P-1}}} \\
\widehat{X_t^{P_P}}
\end{bmatrix} = \begin{bmatrix}
A_\vartheta^1 \left(X_t^{P_1^+} \right) \\
A_\vartheta^2 \left(X_t^{P_2^+} \right) \\
\vdots \\
A_\vartheta^{P-1} \left(X_t^{P_{P-1}^+} \right) \\
A_\vartheta^P \left(X_t^{P_P^+} \right)
\end{bmatrix}.
$$

In the linear case above, the time complexity of the operation is $O\left(d \times (2b+1) \right) \ll O\left(d^2 \right)$.

6.2 Efficient sparsity leveraging Bayesian estimation

We assume that (ε_t) is a Gaussian white noise whose precision matrix, $\Pi_\varepsilon = \Sigma_\varepsilon^{-1}$ is block diagonal with blocks corresponding to the sets of rows and columns $P_1, P_2, \ldots, P_P$.

$$
\Pi_\varepsilon = \Sigma_\varepsilon^{-1} = \begin{bmatrix}
\pi_1 & 0 & & & \\
0 & \pi_2 & 0 & & \\
& & \ddots & & \\
& & 0 & \pi_{P-1} & 0 \\
& & & 0 & \pi_P
\end{bmatrix}.
$$

15

The conditional log-likelihood of an observed process is therefore

$$\mathcal{L}\left(\vartheta, \Pi_{\varepsilon}\right) = \frac{N-1}{2}\log\left(\det\left(\Pi_{\varepsilon}\right)\right) - \sum_{t=1}^{N-1}\left(X_{t+1} - A_{\vartheta}\left(X_{t}\right)\right)^{T}\Pi_{\varepsilon}\left(X_{t+1} - A_{\vartheta}\left(X_{t}\right)\right).$$

Consider

$$\mathcal{L}_{t}\left(\vartheta, \Pi_{\varepsilon}\right) = \left(X_{t+1} - A_{\vartheta}\left(X_{t}\right)\right)^{T}\Pi_{\varepsilon}\left(X_{t+1} - A_{\vartheta}\left(X_{t}\right)\right) = \sum_{i,j=1}^{P}\left(X_{t+1}^{P_{i}} - A_{\vartheta}^{i}\left(X_{t}^{P_{i}^{+}}\right)\right)^{T}\Pi_{\varepsilon}^{P_{i},P_{j}}\left(X_{t+1}^{P_{j}} - A_{\vartheta}^{j}\left(X_{t}^{P_{j}^{+}}\right)\right).$$

We have assumed $\forall i \neq j$, $\Pi_{\varepsilon}^{P_{i},P_{j}} = 0$, therefore

$$\mathcal{L}_{t}\left(\vartheta, \Pi_{\varepsilon}\right) = \sum_{i=1}^{P}\left(X_{t+1}^{P_{i}} - A_{\vartheta}^{i}\left(X_{t}^{P_{i}^{+}}\right)\right)^{T}\pi_{i}\left(X_{t+1}^{P_{i}} - A_{\vartheta}^{i}\left(X_{t}^{P_{i}^{+}}\right)\right)$$

and furthermore

$$\mathcal{L}\left(\vartheta, \Pi_{\varepsilon}\right) = \frac{N-1}{2}\log\left(\det\left(\Pi_{\varepsilon}\right)\right) - \sum_{i=1}^{P}\sum_{t=1}^{N-1}\left(X_{t+1}^{P_{i}} - A_{\vartheta}^{i}\left(X_{t}^{P_{i}^{+}}\right)\right)^{T}\pi_{i}\left(X_{t+1}^{P_{i}} - A_{\vartheta}^{i}\left(X_{t}^{P_{i}^{+}}\right)\right).$$

Let us consider we want to maximize $\mathcal{L}\left(\vartheta, \Pi_{\varepsilon}\right)$ with respect to ϑ, we have

$$\nabla_{\vartheta}\mathcal{L}\left(\vartheta, \Pi_{\varepsilon}\right) = -2\sum_{i=1}^{P}\sum_{t=1}^{N-1}\left(D_{\vartheta}A_{\vartheta}^{i}\left(X_{t}^{P_{i}^{+}}\right)\right)^{T}\pi_{i}\left(X_{t+1}^{i} - A_{\vartheta}\left(X_{t}^{P_{i}^{+}}\right)\right)$$

where, letting $|\vartheta|$ denote the number of parameters in the model,

$$D_{\vartheta}A_{\vartheta}^{i}\left(X_{t}^{P_{i}^{+}}\right) = \left[\begin{array}{ccc}\partial_{\vartheta_{1}}A_{\vartheta}^{i}\left(X_{t}^{P_{i}^{+}}\right) & \cdots & \partial_{\Theta_{|\vartheta|}}A_{\vartheta}^{i}\left(X_{t}^{P_{i}^{+}}\right)\end{array}\right].$$

What is remarkable with these expressions is that they enable embarrassingly parallel computations provided one uses P different nodes for any $i \in \{1 \ldots P\}$, node i holds the data corresponding to $\left(X_{t}^{P_{i}^{+}}\right)_{t\in\{1\ldots N\}}$. This is also true if one plans on using a second order method.

With a first order method, the time complexity of computing the gradient for each node is $O\left(N \times |\vartheta| \times \left|P_{i}^{+}\right|^{2}\right)$ where $\left|P_{i}^{+}\right|$ the cardinal of the overlapping partition which can be as low as $2 \times b + 1$. This implies that this solution is scalable with respect to the size of the model as it leverages the prior knowledge of the model's sparsity. To the best of our knowledge, there is no equivalent computational result with the matrix inversion and Yule-Walker equation based methods above.

6.3 Gradient descent, step size and rate of convergence

This section focuses on the maximization of the conditional likelihood of a Gaussian AR process. In order to simplify notations we only consider the AR1 case. The conclusions below are trivially extended to the general case. Letting π_{ε} the precision matrix of the process' noise, maximizing the conditional likelihood of the process is equivalent to maximizing

$$\mathcal{L}\left(A_{1}, \pi_{\varepsilon}\right) = \frac{1}{N}\sum_{t=1}^{N}\sum_{t=1}^{d}\sum_{t=1}^{d}\left(X_{t}^{i} - \sum_{k=1}^{d}A_{1}^{ik}X_{t-1}^{k}\right)\pi_{\varepsilon}^{i,j}\left(X_{t}^{j} - \sum_{l=1}^{d}A_{1}^{jl}X_{t-1}^{l}\right)$$

and

$$\frac{\partial \mathcal{L}}{\partial A_1^{i_0,j_0}} (A_1, \pi_\varepsilon) = -\frac{2}{N} \sum_{t=1}^{N} \sum_{j=1}^{d} X_{t-1}^{j_0} \pi_\varepsilon^{i_0,j} \left(X_t^j - \sum_{l=1}^{d} A_1^{jl} X_{t-1}^l \right).$$

We consider π_ε has already been estimated by $\widehat{\pi}_\varepsilon$. Maximizing $\mathcal{L}$ with respect to A_1 then comes down to a gradient ascent update by $\left(\frac{\partial \mathcal{L}}{\partial A_1^{i_0,j_0}} (A_1, \widehat{\pi}_\varepsilon) \right)_{i_0,j_0}$. In several particular cases this update matrix takes a remarkable form:

- If $\widehat{\pi}_\varepsilon = I$, $\frac{\partial \mathcal{L}}{\partial A_1^{i_0,j_0}} (A_1, \widehat{\pi}_\varepsilon) = -\frac{2}{N} \sum_{t=1}^{N} X_{t-1}^{j_0} \left(X_t^{i_0} - \sum_{l=1}^{d} A_1^{i_0 l} X_{t-1}^l \right) = -2 \widehat{\mathrm{Cov}} \left(\overrightarrow{X_t}, X_t \right)$.
  In this case the Hessian of $A_1 \to \mathcal{L}(A_1, \widehat{\pi}_\varepsilon)$ is a block diagonal matrix whose blocks are $\widehat{\mathrm{Cov}}(X_t, X_t)$. Therefore finding the smallest m and largest L eigenvalues of $\widehat{\mathrm{Cov}}(X_t, X_t)$ is sufficient to compute $\frac{2}{m+L}$. This step size is of importance as it provably achieves an exponential rate of convergence to the optimum in a gradient ascent.

- If $\widehat{\pi}_\varepsilon$ is diagonal, $\frac{\partial \mathcal{L}}{\partial A_1^{i_0,j_0}} (A_1, \widehat{\pi}_\varepsilon) = -\frac{2}{N} \sum_{t=1}^{N} X_{t-1}^{j_0} \widehat{\pi_\varepsilon^{i_0,j_0}} \left(X_t^{i_0} - \sum_{l=1}^{d} A_1^{i_0 l} X_{t-1}^l \right) = -2 \widehat{\pi_\varepsilon^{i_0,j_0}} \widehat{\mathrm{Cov}} \left(\overrightarrow{X_t}, X_t \right)$.
  The Hessian is then the Kronecker product of $\widehat{\pi}_\varepsilon$ and $\widehat{\mathrm{Cov}}(X_t, X_t)$ which means it is sufficient to compute the smallest and largest eigenvalues of $\widehat{\pi}_\varepsilon$ and $\widehat{\mathrm{Cov}}(X_t, X_t)$ in order to find a converging step size to the gradient ascent. The rate of convergence in that case will be linear.

- In any case, a line search method also enables convergence to the maximum.

For the detail of these convergence rates we refer the reader to

7 Common access patterns, fragmentation of data, distribution of computations

The M (frequentist, average based) and Z (bayesian, maximum likelihood based) estimators above rely on similar computational needs. Namely, one computes the output of a kernel function evaluated on neighboring data and then averages the results.

7.1 Map reduce for M kernel based estimators

There is a very common computation pattern to the estimators. Frequentist estimators rely on the estimation of a covariance function. In the centered case, this can be computed by considering a finite covariance matrix which can be estimated by computing quantities

$$\widehat{\gamma_h^X} \left((X_t)_{t \in \{1 \ldots N\}} \right) = \frac{1}{N-h-1} \sum_{k=1}^{N-h} X_k X_{k+h}^T$$

for $h \in \{-H \ldots H\}$. This means that the vector of square matrices

$$\left(\widehat{\gamma_h^X} \left((X_t)_{t \in \{1 \ldots N\}} \right) \right)_{h \in \{-H \ldots H\}}$$
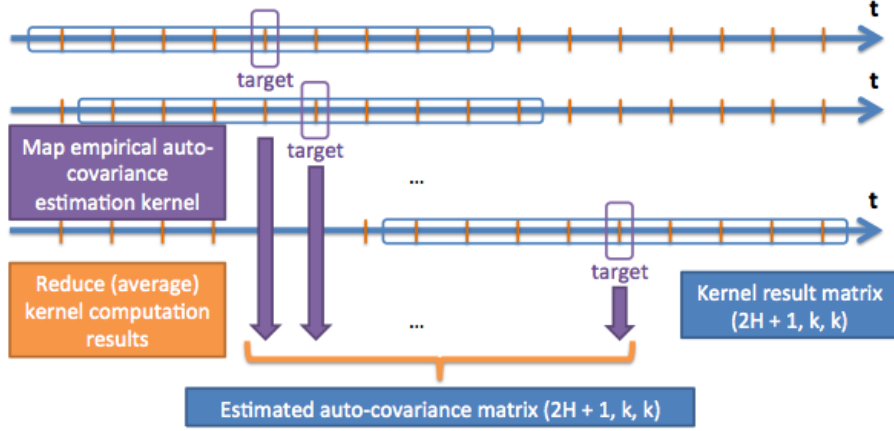
17

Figure 2: Example of a mapped kernel based estimation in the case of the estimation of order H auto-covariance

can be approximately estimated as

$$\frac{1}{N-(2h+1)} \sum_{k=h+1}^{N-h} \left(X_k X_{k-h}^T, \ X_k X_{k-(h-1)}^T, \dots, \ X_k X_k^T, \ X_k X_{k+1}^T, \dots, X_k X_{k+h}^T \right).$$

This estimator is biased but asymptotically unbiased. Computing this estimator of the auto-covariance matrix is quite straightforward with map-reduce operators. One maps the computation of the local kernel

$$(X_{k-h}, \dots, X_{k-1}, X_k, X_{k+1}, \dots, X_{k+h}) \xrightarrow{\kappa} \left(X_k X_{k-h}^T, \ X_k X_{k-(h-1)}^T, \dots, \ X_k X_k^T, \ X_k X_{k+1}^T, \dots, X_k X_{k+h}^T \right)$$

and then reduces with a sum operator prior to normalizing by $\frac{1}{N-(2h+1)}$. Such a computational schema is represented in Figure ??. This estimator is asymptotically normal with a $\frac{1}{N}$ convergence rate.

7.2 Map reduce for Z kernel based estimators

Computing the gradient of parameters in an auto-regressive model of finite order p can also be formulated in terms of a similar computational schema. For instance, when one tries to solve the conditional maximum likelihood problem for an AR model

$$\max_{A_1, A_2, \dots, A_p, \Theta} \sum_{t=p+1}^{N} \log f \left(\Theta, X_t - A_1 X_{t-1} - \dots - A_p X_{t-p} \right)$$

by a first order method, the gradient can also be computed as a sum over a large number of terms of locally computable quantities. Indeed, let F be the function

$$(\Theta, A_1, \dots, A_p) \xrightarrow{F} \sum_{t=p+1}^{N} \log f \left(\Theta, X_t - A_1 X_{t-1} - \dots - A_p X_{t-p} \right) = \sum_{t=p+1}^{N} F_t \left(\Theta, X_t - A_1 X_{t-1} - \dots - A_p X_{t-p} \right).$$

18

Then, by linearity of the gradient,

$$\nabla F\left(\Theta, A_1, \ldots, A_p\right) = \sum_{t=p=1}^{N} \nabla F_t\left(\Theta, X_t - A_1 X_{t-1} - \ldots - A_p X_{t-p}\right).$$

For any $t \in \{p+1 \ldots N\}$, computing

$$\nabla F_t\left(\Theta, X_t - A_1 X_{t-1} - \ldots - A_p X_{t-p}\right)$$

only relies on data $X_t, \ldots, X_{t-p}$.

7.3 Overlapping data

These quantities are straightforward to compute in a serial manner on a single CPU. However, when it comes to using a distributed system of computation such as Apache Spark, one has to take into account the requirement for data to be partitioned. In order to speed up the computation process and avoid the inter-node communication bottleneck one will want to mostly use a data partitioning scheme that, once loaded in the RAM of a cluster of machines, will enable embarrassingly parallel computations.

For GPU units to speed further up the process, it is also interesting to consider data partitioning as global memory units are usually slower to access than shared one. Furthermore, one will want to be able to run the computations above in an embarrassingly parallel way once appropriate parallelization steps have been undertaken.

As we explain in the following, this preparation step consists of replicating the data so as to create overlapping partitioning. The next part will explain the theoretical foundations of the scheme and how to choose its padding horizon parameter appropriately with respect to the calibration of a certain model. In particular, we introduce the fundamental data structure that has been designed to enable large scale analysis of time series: distributed overlapping blocks.

Part II

A general programming paradigm for weak memory estimation

In this paragraph we will give formal definitions of the notions of weak memory from a computational standpoint. The following enables general purpose time series analysis calculus to run on partitioned data architecture in an embarrassingly parallel manner. In particular, the distributed overlapping block data structure enables partitioning with respect to both the different dimensions of a time series (different signals, sources of data) and with respect to time, which is new to the best of our knowledge. The paradigm is simple and therefore can easily be adapted to other systems. This non system specific principle of overlapping blocks is very powerful though as it enables in-RAM computations on time series with an EC2 cluster in an embarrassingly parallel way as well as GPU parallelization of time series calculus thanks to CUDA.

The package created for Apache Spark, SparkGeoTS, that follows this scheme enables time series analysis at an unprecedented scale, both in terms of density of time stamps along the time axis and in terms of very highly dimensional time series analysis. It supports data analytics for both regularly time stamped data and irregularly spaced time series. Its novelty stems from leveraging the informational properties of weak memory in ergodic time series. It is not adapted to the non-ergodic context which is not an actual shortcoming as in this case most usual estimators for time series are not even convergent.

8   Weak memory computation for estimators

The first part of this document reviewed a vast class of time series models that feature weak memory computational needs. In the following we formalize that notion. Let us consider data in the form of $(X_t)_{t \in \{0...N\}}$.

Definition: Order H weak memory estimator   An estimator $\text{Est}\left((X_t)_{t \in \{0...N\}}\right)$ features order H weak memory if there exists an integer H (horizon in number of steps) such that sufficient statistics for that estimator can be computed by reducing kernel computation results feeding on a data point and the neighbors of that point that are less than H time steps away on the time index.

Example: Second order stationary time series   We have shown above that usual estimators for $\text{AR}(p)$ models are p weak memory estimators, similarly with $\text{MA}(q)$ (q weak memory) and $\text{ARMA}(p, q)$ models ($p + q$ weak memory).

9   Weak memory in time series graphs

If one considers a system in which sensors are included in a relational graph then the time series of readings produced by these sensors are naturally embedded in a graph. An example of the resulting data lattice is represented in Figure **??**.

A regularly indexed time series graph is a sequence $\left((X_t^v)_{v \in V}\right)_{t \in \{1...N\}}$ where the set V of vertices of the graph is tied together by a set E of edges with uniform weight 1.

9.1   Data with regularly indexed timestamps

Definition: Order (H, K) weak memory estimator   An estimator $\text{Est}\left[(X_t^v)_{v \in V}\right)_{t \in \{1...N\}}\right]$ is said to feature order (H, K) weak memory if sufficient statistics can be computed by reducing kernel computations that, for each vertex state X_t^v, only feed on the states of neighbors that are at most K hops away in the graph on a time window that does not go further than H time steps away from t_0.
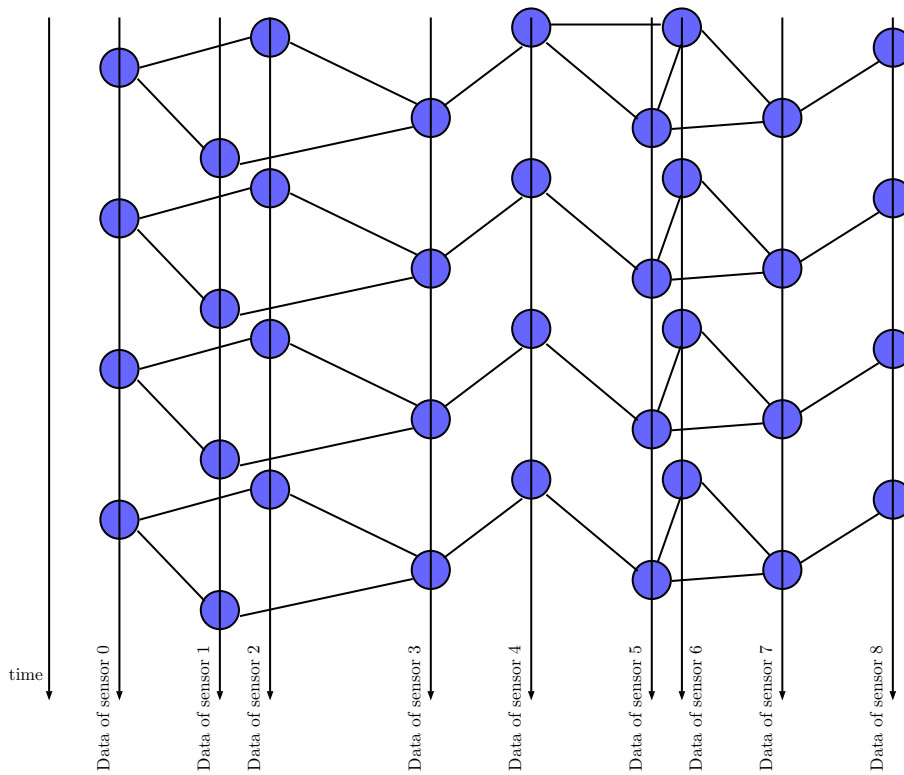
Figure 3: Sensor graph lattice

Example: queuing model for arterial traffic A traffic network can be mapped to a graph where vertices represent intersections and edges correspond to roads. Such a mapping correspond to a primal whose dual graph consists of vertices modeling roads and edges accounting for the binding intersections represent. One considers a discrete model in which the state of congestion at time t on vertex v only depends on the state of congestion of its upstream and downstream neighbors. This assumption is very usual in traffic as soon as the discretization resolution is such that $\frac{\Delta x}{\Delta t} \leq v$ where Δx is the length of the shortest road, Δt the time discretization resolution and v the maximum speed of vehicles in the network.

10 Embarrassingly parallel weak memory time series distributed representations

In this section we define a programming paradigm to leverage the informational structure of short memory time series in order to make their analysis distributed and embarrassingly parallel. In particular we focus on the case where both the sampling rate and the number of dimensions of the time series under consideration are high enough so that the data cannot fit in a reasonable amount of RAM on a single machine.

10.1 Very high dimensional short memory time series

Strategies have been developed to parallelize the analysis of time series on distributed clusters or high performance computers. For Apache Spark, libraries such as SparkTS and Thunder divide the time series into smaller data sets by splitting it along dimensions. This paradigm is efficient whenever each dimension has few enough timestamp for its entire data to fit a node's RAM. Also, for multivariate analysis, it is more suitable to preserve data locality across dimensions. The statistical estimation code that should be used then needs not be aware of the partitioning scheme. What is more, there is a gain in terms of efficiency as joins on timestamps are not necessary.

In the following we devise data structures that leverage the informational properties of short memory time series in order to make their analysis embarrassingly parallel. From a design standpoint, a map-reduce programming scheme is adopted so that pre-existing estimation tools can be used with that distributed container.

10.2 The overlapping data model

Going back to the definition of short memory in time series, it is obvious now that any local operation only needs to be aware of its H neighborhood. Neighborhoods are defined in terms of steps for regularly spaced data and units of time for irregularly spaced data.

Considering the data of a time series $(t_i, X_{t_i})_{i \in \{1...kN\}}$ and an estimation kernel $k\left(t_0, \left(t_j, X_{t_j}\right)_{|t_j - t_0| \leq H}\right)$ (with window width $2 \times H$), any estimation based on a reduced quantity of the results of this kernel can become embarrassingly parallel provided data is partitioned, partially replicated and contained in an overlapping partitioning with overlap at least H.
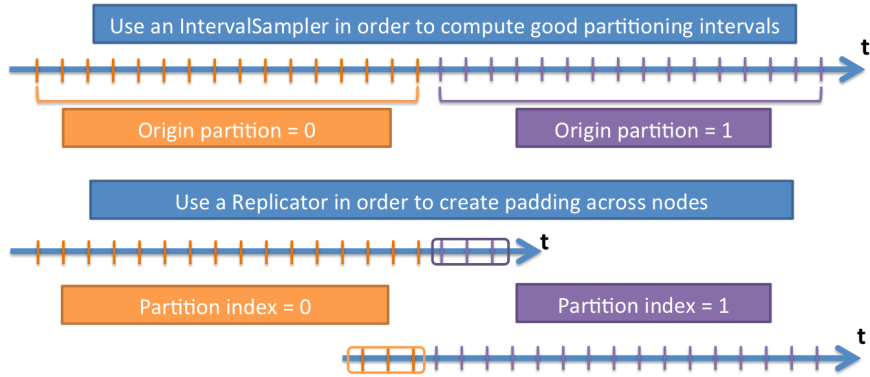
Figure 4: Overlapping distributed dataset.

10.2.1   Map-reduce based estimation with a kernel of width $2 \times H$

Let an estimator

$$\sum_{i=1}^{N} k\left(t_0, \left(t_j, X_{t_j}\right)_{|t_j - t_0| \leq H}\right)$$

where $\sum$ stands for any commutative and associative operation.

Let us assume that the data has been partitioned in k partitions with an overlap of width H between partitions. The computation flow is illustrated in the case of the estimation of an auto-covariance function in Figure **??**. The presence of an overlap directly enables one to make that computational embarrassingly parallel with no communication needed between computational nodes holding different partitions of the data. Such a representation of data is illustrated in Figure **??**.

10.2.2   Overlapping data set with respect to space

Time is not the only dimension along which overlapping partitioning can be computed. Obviously the scheme can be adapted to spatial meshes with regular indexing such as images. The case of data points irregularly arranged on a map falls under the same paradigm. More interesting is the case of data where no Euclidian distance is available in a straightforward way. Graphs correspond to that kind of data.

Local operations on graphs are straightforwardly defined in terms of reduced kernels provided they are only interested in the parenthood degree of the neighbors of the target. Therefore the same scheme can be adapted as illustrated in Figure **??**.

10.3   Long memory case

Time series such as stock valuations on the stock market or volatility are known to feature long memory. In other words, the informational footprint of an event never completely fades away in the system. After having computed its consecutive differences, any integrated process comes down
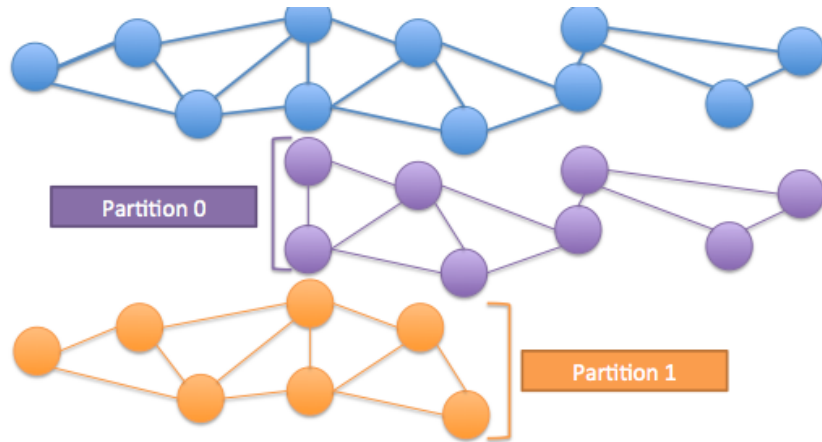
Figure 5: Overlapping graph structure

to a short memory time series. This means that the data representation paradigm above is valid in a vast range of cases. It can be extended to partially integrated processes thanks to partial differentiation provided the partial differentiation kernel is approximated by a finite support kernel.

11 Embarrassingly parallel representation of time series embedded in graphs

This section will be dedicated to finding applications of the graph overlapping paradigm above in vast systems where information flows within a sparse Dynamic Bayesian Network.

11.1 Sparse spatial dynamic bayesian network

Studying arterial network dynamics in traffic often comes down to considering the series of states of vertices (road links) bound together by intersections. In order to compute the current number of vehicles on a link, one takes into account the number of vehicles that are effectively leaving, baring the constraints of downstream occupancy capacity, to the current occupancy and adds up the number of vehicles flowing from upstream.

11.1.1 Order $(1, 1)$ time and space memory Directed Acyclic Bayesian network

Such a setting is a particular instantiation of a short time and space memory time series graph. It is illustrated in Figure **??**.

In such a framework, the next state of a given vertex can be computed as the result of the convolution of a kernel from its direct parents. Therefore, the overlapping data structure can be used in order to make the corresponding study embarrassingly parallel. This can be used for retrospective data analysis and simulation as well provided the same random number series are provided to edge representing identical forward state computations.
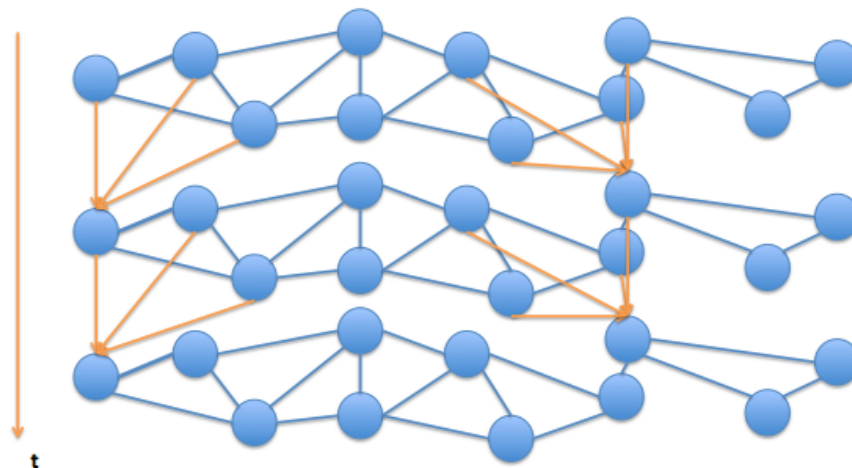
Figure 6: Order $(1, 1)$ time/space memory Bayesian network (for a discretized arterial traffic model)

11.2   Cross-product of overlapping representations

In order to maintain reasonable size partitions, that is to say, partitions that can fit in the RAM of a non HPC machine, one can combine overlapping partitioning with respect to time and the corresponding time series graph in a cross-product fashion. This creates more redundant data but still provides a representation of data that enables embarrassingly parallel computations if short memory is leveraged.

If the informational structure of the data set is as presented in Figure **??**, then the cross product partitioning illustrated in Figure **??** enables its embarrassingly parallel analysis.

12   Speeding up computations on a GPU

Here we prove that the overlapping block scheme is not system specific and can be adapted to another kind of hardware: Graphical Processing Unit (SIMD systems).

12.1   Memory hierarchy

Here we focus on the memory hierarchy highlighted by Nvidia's CUDA GPGPU language. It is possible to write data from the RAM to both the global device memory and shared memory.

The shared memory is only accessible by threads of the same block and only lives as long as a kernel execution. However, it enables one to leverage the computational power of the GPU as its bandwidth is often 100 times that of the global memory.

Time series analysis relies on computation of local kernels in which data is mostly accessed redundantly by several threads of the same block. Therefore, it seems reasonable to try and leverage this shared memory. Each block of this memory is only accessible to the corresponding threads and
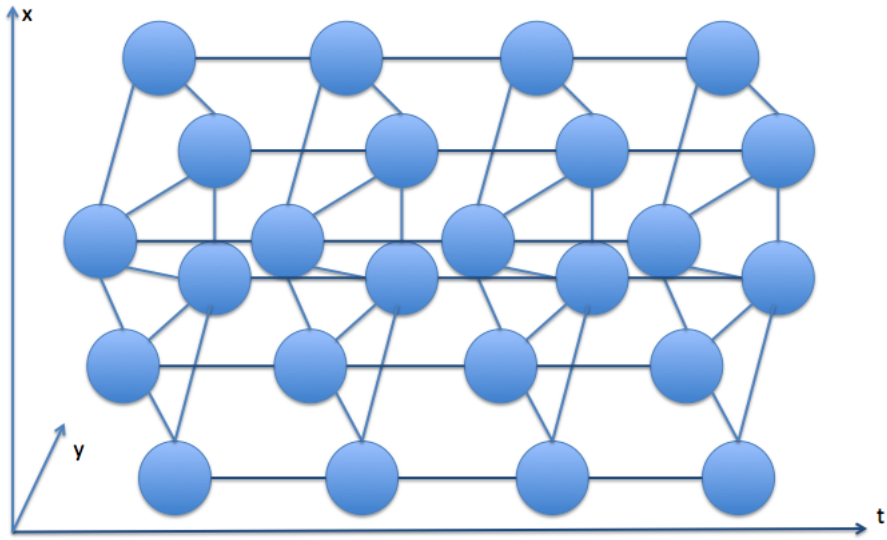
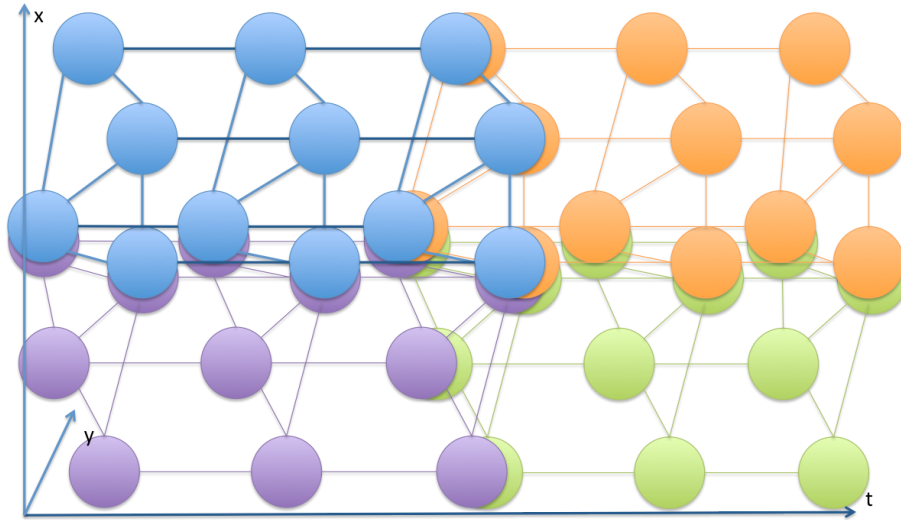Figure 7: Data embedded in a $(1,1)$ memory time series graph

Figure 8: Cross product of overlapping partitioning (with respect to time and space)
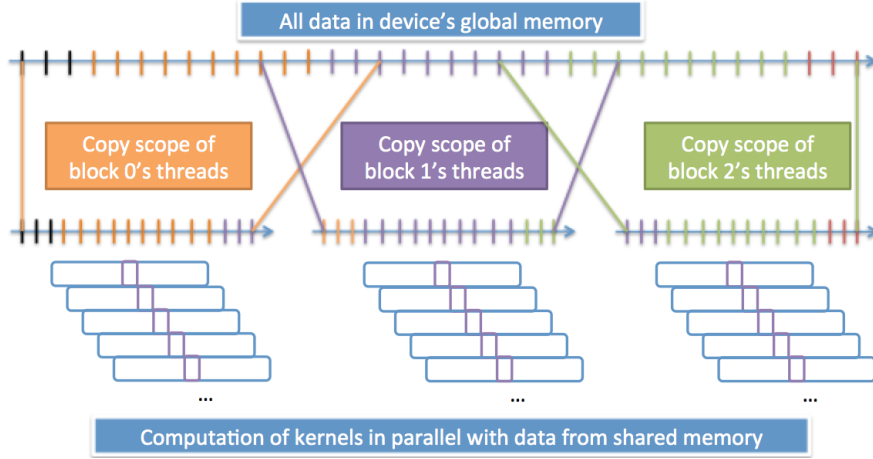
Figure 9: Overlapping block enabling shared memory used on a GPU

the size is much less than that of the global memory (by a factor of 1000).

We show here that overlapping blocks enable us to conduct time series analysis in a weak memory context in an embarrassingly parallel manner while leveraging the high bandwidth of shared GPU memory.

12.2 Parallel computation and overlapping blocks in shared memory for regularly spaced time series

In the following we illustrate the use of overlapping blocks on GPUs with regularly spaced time series.

We assume the data (X_t) has been copied in the global GPU memory and one wants to execute a kernel of width $2H$ prior to conducting a reduction on the results.

Let $T_{i,j}$ the jth thread of the ith thread block. We assume blocks are of size $N_B + 2H$ where N_B is small enough so the shared memory of each block is not saturated. $T_{i,j}$ will copy $X_{i(N_B)+j}$ from the global memory (address $iN_B + j$) to the shared memory (local address j).

Then each thread with index $j \in \{H, N_B + H\}$ will compute a kernel based on data contained in local addresses corresponding to the appropriate data, namely that contained in the shared memory addresses ranging from $j - H$ to $j + H$.

Provided the kernel width outweighs the cost of the copy from the global device memory to the block's shared memory, this provides a speed up and optimized embarrassingly parallel data analysis for weak memory systems. The flow of computations is represented in Figure **??**.

12.3 Parallel computation and overlapping blocks in shared memory for irregularly spaced time series

For irregularly spaced data, computations are slightly more challenging as the data that needs to be taken into account by the computation of a local kernel does not straightforwardly correspond to a range of memory addresses. However, we make a weak memory assumption that no data further away than H needs to taken into account in order to compute a kernel around a datum.

One creates an overlapping partitioning of the data with H overlap and baring the memory size constraint of the shared memory blocks. If the kernel considers all data points with timestamps in $[t - H, t + H]$ in order to compute its output about a datum at t, then letting each thread assigned with a kernel computation look backward and forward from t for the first indices whose timestamps are out of range and then run the kernel convolution on the data within range solves the problem.

When kernel centers do not necessarily to data points within the data set under consideration, things get more complicated but can be made computationally efficient thanks to binary search. One may also create a skip list in the global memory of the device for each thread to access and figure out the range of the data it needs to consider. This skip list of timestamps may sit in the global memory as it will be accessed only once by each thread. Copying it to shared memory blocks would cost too much overhead.

Conclusion

In this document we have shown how weak memory models in time series analysis can be estimated and used in the context of big distributed data sets. Identifying how many lagged values are necessary to the calibration of the model the user wants to implement is a necessary preliminary step. It paves the way to building up a distributed set of overlapping partitions. This overlapping partition scheme corresponds to partitioning with respect to time which is the new contribution the present document presents. We show how this can also be extended to spatial partitioning when banded causal relationship models are being considered.

The new overlapping distributed data set presented here enables a new any-scale any-dimensional analysis of data without the need for shuffling observations between computation nodes once the appropriate data representation has been created. This provides the user with the opportunity to calibrate linear weak memory time series at scale in a reactive manner and the possibility to quickly assess how much a given model is appropriate in terms of goodness of fit and complexity. This paradigm can also be extended to the calibration of conditionally heteroscedastic models [?] as well as long memory models [?, ?]. This is the subject of ongoing work.

References

[1] D. R. Brillinger, *Time series: data analysis and theory*, vol. 36. Siam, 1981.

[2] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods*. New York, NY, USA: Springer-Verlag New York, Inc., 1986.

[3] J. D. Hamilton, "Time series analysis princeton university press," *Princeton, NJ*, 1994.

[4] A. C. Harvey and A. Harvey, *Time series models*, vol. 2. Harvester Wheatsheaf New York, 1993.

[5] H. Ltkepohl, *New Introduction to Multiple Time Series Analysis*. Springer Publishing Company, Incorporated, 2007.

[6] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[7] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pp. 1–10, IEEE, 2010.

[8] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy, "Hive-a petabyte scale data warehouse using hadoop," in *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pp. 996–1005, IEEE, 2010.

[9] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pp. 2–2, USENIX Association, 2012.

[10] R. S. Tsay, *Analysis of financial time series*, vol. 543. John Wiley & Sons, 2005.

[11] M. Mudelsee, "Climate time series analysis," *Atmospheric and*, vol. 397, 2010.

[12] S. Basu, A. Mukherjee, and S. Klivansky, "Time series models for internet traffic," in *INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, vol. 2, pp. 611–620, IEEE, 1996.

[13] K. W. Hipel and A. I. McLeod, *Time series modelling of water resources and environmental systems*. Elsevier, 1994.

[14] T. Hunter, T. Moldovan, M. Zaharia, S. Merzgui, J. Ma, M. J. Franklin, P. Abbeel, and A. M. Bayen, "Scaling the mobile millennium system in the cloud," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, p. 28, ACM, 2011.

[15] M. Franklin *et al.*, "Mllib: A distributed machine learning library," *NIPS Machine Learning Open Source Software*, 2013.

[16] J. Durbin and S. J. Koopman, *Time series analysis by state space methods*. No. 38, Oxford University Press, 2012.

[17] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[18] D. P. Bertsekas, "Nonlinear programming," 1999.

[19] F. M. Callier and C. A. Desoer, *Linear system theory*. Springer Science & Business Media, 2012.

[20] H. Kantz and T. Schreiber, *Nonlinear time series analysis*, vol. 7. Cambridge university press, 2004.

[21] J. Fan and Q. Yao, *Nonlinear time series: nonparametric and parametric methods.* Springer Science & Business Media, 2003.

[22] Straumann and T. Mikosch, "Estimation in conditionally heteroscedastic time series models," tech. rep., Lecture Notes in Statist. 181, 2005.

[23] H. Akaike, "Block toeplitz matrix inversion," *SIAM Journal on Applied Mathematics*, vol. 24, no. 2, pp. 234–241, 1973.

[24] R. Lund, "Estimation in conditionally heteroscedastic time series models," *Journal of the American Statistical Association*, vol. 101, no. 475, p. 1319, 2006.

[25] P. Doukhan, G. Oppenheim, and M. S. Taqqu, *Theory and applications of long-range dependence.* Springer Science & Business Media, 2003.

[26] B. B. Mandelbrot, *Fractals and Scaling in Finance: Discontinuity, Concentration, Risk. Selecta Volume E.* Springer Science & Business Media, 2013.