SCALABLE TIME SERIES CHANGE DETECTION FOR BIOMASS MONITORING USING GAUSSIAN PROCESS

VARUN CHANDOLA* AND RANGA RAJU VATSAVAI*

Abstract. Biomass monitoring, specifically, detecting changes in the biomass or vegetation of a geographical region, is vital for studying the carbon cycle of the system and has significant implications in the context of understanding climate change and its impacts. Recently, several time series change detection methods have been proposed to identify land cover changes in temporal profiles (time series) of vegetation collected using remote sensing instruments. In this paper, we adapt Gaussian process regression to detect changes in such time series in an online fashion. While Gaussian process (GP) has been widely used as a kernel based learning method for regression and classification, their applicability to massive spatio-temporal data sets, such as remote sensing data, has been limited owing to the high computational costs involved. In our previous work we proposed an efficient Toeplitz matrix based solution for scalable GP parameter estimation. In this paper we apply these solutions to a GP based change detection algorithm. The proposed change detection algorithm requires a memory footprint which is linear in the length of the input time series and runs in time which is quadratic to the length of the input time series. Experimental results show that both serial and parallel implementations of our proposed method achieve significant speedups over the serial implementation. Finally, we demonstrate the effectiveness of the proposed change detection method in identifying changes in *Normalized Difference Vegetation Index* (NDVI) data.

1. Introduction

Increasing availability of high resolution remote sensing data has encouraged researchers to extract knowledge from these massive spatio-temporal data sets in order to solve different problems pertaining to our ecosystem. Land use land cover (LULC) monitoring, specifically identifying changes in land cover, is one such problem that has significant applications in detecting deforestation, crop rotation, urbanization, forest fires, and other such phenomenon. The knowledge about the land cover changes can then be used by policy makers to take important decisions regarding urban planning, natural resource management, water source management, etc.

In this paper we focus on the problem of identifying changes in the biomass or vegetation in a geographical region. *Biomass* is defined as the mass of living biological organisms in a unit area. In the context of this study, we restrict our monitoring to plant (specifically crop) biomass over large geographic regions. In recent years biomass monitoring is increasingly becoming important, as biomass is a great source of renewable energy. Moreover, biomass monitoring is also important from the changing climate perspective, as changes in climate are reflected in the change in biomass, and vice versa. The knowledge about biomass changes over time across a geographical region can be used estimate quantitative biophysical parameters which can be incorporated into global climate models.

The launch of NASA's Terra satellite in December of 1999, with the Moderate Resolution Imaging Spectroradiometer (MODIS) instrument aboard, introduced a new opportunity for terrestrial remote sensing. MODIS data sets represent a new and improved capability for terrestrial satellite remote sensing aimed at meeting the needs of global change research. With thirty-six spectral bands, seven designed for use in terrestrial application, MODIS provides daily coverage, of moderate spatial resolution, of most areas on the earth. Land cover products are available in 250m, 500m, or 1000m resolutions [17]. MODIS land products are generally available within weeks or even days

*Oak Ridge National Laboratory, chandolav@ornl.gov, vatsavairr@ornl.gov.

of acquisition and distributed through the EROS Data Center[1] and are currently available free of charge. MODIS land products allow users to identify vegetation changes over time across a region and estimate quantitative biophysical parameters which can be incorporated into global climate models. A NDVI *temporal profile* is a graphical plot of sequential NDVI observations against time. These profiles quantify the remotely sensed vegetation's seasonality and dynamics. These profiles can be described with simple parameters, like the amplitude, mean, and standard deviation. We can understand the onset and peak of greenness and the length of growing season from analyzing these profiles. By monitoring NDVI profiles as time series, we can understand the changes in the biomass in a continuous manner. MODIS data has been extensively used to study vegetation and crop phenological characteristics [19, 33], and *monitoring* [39]. However, online monitoring of biomass over large geographic regions is relatively unexplored area.

There is an imminent need for algorithms that can be applied to the problem of identifying land cover change in spatio-temporal data sets in an online fashion. Some of the challenges faced by the researchers in this domain include adapting to online setting, accounting for missing data and outliers, handling non-linear dependencies, seasonality and non-stationarity in the time series, incorporating spatial dependencies, and scaling to the massive data sizes. Recently, land cover change detection techniques have been proposed that identify changes in *normalized difference vegetation index* (NDVI) time series data collected by applying time series change detection techniques [3, 10, 25, 21], but do not address most of the challenges associated with the land cover change detection problem.

1.1. **Gaussian Process Based Time Series Analysis.** While change detection for time series data has been a widely researched topic in statistics and signal processing community, algorithms that can detect changes in periodic time series data are limited [14, 3], and even these techniques are not well-suited for online change detection. We propose a non-parametric statistical algorithm that can detect changes in noisy time series data in an online fashion. We use *Gaussian Process* [28, 31] as the basis for a Bayesian non-parametric predictive model for time series data and use the difference between the predicted and observed values to monitor change in a continuous manner, meaning that change detection map is continuously revised as soon as new data collected by the remote sensing satellites is available.

Gaussian process (GP) [28, 31] [2] based approaches are increasingly being used as a kernel machine learning tool for non-parametric regression and classification. If the time indices are used as the inputs, one can use a GP as a forecasting or prediction model for time series data [4, 12, 5]. Besides prediction, GP based models can also be used for other time series analysis tasks such as change detection, anomaly detection, missing data imputation, noise removal, etc. In this paper we use the predictive capabilities of GP for online change detection in time series data.

While GP has emerged as a popular kernel machine learning tool, its application to large scale data sets has been limited owing to the inherent $O(t^3)$ computational complexity as well as $O(t^2)$ memory storage requirements, where t is the input data size. The key bottleneck is the handling of a large $t \times t$ covariance matrix and solving a large system of equations. The standard approach [31] is to use Cholesky decomposition of the covariance matrix. When dealing with time series, t is the length of the time series, which can be large (and growing) for applications such as remote sensing, astronomy, electro-cardiograph (ECG) analysis, etc. For example, MODIS collects data for entire globe daily and hence the length of the NDVI time series is continuous growing.

The computational bottleneck for GP based analysis is further compounded by the fact that often one needs to simultaneously handle multiple time series. For NDVI time series, for example, multiple time series from a spatial region need to be analyzed simultaneously. As the spatial resolution of the remote sensing instruments grows, the number of time series to be simultaneously analyzed will grow accordingly. The standard GP analysis methods have a $O(pt^3)$ computational complexity and

[1] `http://eros.usgs.gov/`
[2] Henceforth, referred to as GP.

a $O(t^2 + p)$ memory footprint for handling p time series simultaneously. While multi-threaded or parallel programming can alleviate the issue of handling p time series simultaneously, the quadratic memory requirements are a significant bottleneck, especially in emerging heterogeneous computing architectures, hybrid of multi-core and Graphical Processing Units (GPUs), in which movement of data is expected to be the biggest computational bottleneck.

In our previous work [6], we proposed a hyper-parameter estimation algorithm for GP that exploits the special structure of the covariance matrix associated with the GP analysis to make the algorithm scale to massive data sizes. In this paper, we apply the fast algorithm for change detection using GP. The computational complexity of the GP based change detection is $O(t^2)$ and requires $O(t)$ memory. We also present a parallel version of the algorithm to simultaneously process multiple time series. We present results on artificial data to demonstrate the speedups achieved the proposed algorithm running in serial as well as in parallel (multi-threaded) mode on a multi-core system.

For biomass monitoring, we demonstrate the effectiveness of the proposed method in identifying changes in NDVI time series collected for the Iowa region. We also demonstrate the scalability of the proposed methods in handling six years of NDVI data for the Iowa region.

1.2. **Related Work.**

1.2.1. *Time Series Change Detection.* Change detection for time series data is a widely researched are in different research communities such as statistics [16], signal processing [2], and process control [22]. Most of the existing change techniques can be grouped into three categories, viz., *parameter change based techniques* [29, 16], *segmentation based techniques* [15, 27, 34] and *forecasting based techniques* [10, 23]. Parameter change based techniques assume that the time series follows a parametric distribution and focus on identifying when the parameters change using a hypothesis test procedure. For periodic time series, a parametric assumption is typically unrealistic, unless the seasonality from the time series is removed, which can result in loss of useful information. Segmentation based techniques are non-parametric but are usually not suitable for online setting. Forecasting based techniques [10, 23] use a forecasting model to predict at a given time instance and the combine the predicted and observed values to identify changes. Existing forecasting based techniques have been applied to time series.

Change detection has been applied to remote sensing data to identify events such as land use change, forest fires, and natural disasters. While some of these techniques directly handle the satellite images [26, 36, 30, 32], recently, several techniques have been proposed that identify changes in NDVI time series data by applying time series change detection techniques [3, 25, 10, 21].

GP have not been explicitly used for change detection in time series though similar online Bayesian algorithm has been proposed by Adams and Mackay [1] for time series data. Several papers have used GP for time series modeling and prediction [4, 5, 12].

1.2.2. *Addressing Computational Complexity of Gaussian Process Analysis.* As noted earlier, GP based methods typically scale as $O(t^3)$ with the size of the input data with a memory requirement of $O(t^2)$. This makes them impractical in domains that encounter massive data sizes such as remote sensing, ECG analysis, etc. Several approximation based methods have been proposed in the literature [40, 8, 11] to scale GP to such large datasets (See [31, Chapter 8] for a detailed overview). These methods fall under the general purview of sparse and approximate kernel methods. All of these methods use matrix approximation techniques to efficiently manipulate the covariance matrix (inverse computation, Cholesky factorization, solving system of equations). Several papers [40, 8] approximate the covariance matrix using lower rank approximation techniques, such as the Nyström approximation, for faster but approximate results. Several papers have used a "subset of regressors" approach [35, 11] that uses only m out of t regressors and hence entail $O(m^2t)$ complexity. In this paper we focus on scaling the GP analysis such that we obtain the exact solution and hence we do not compare our approach to the existing approximate methods.

While scalability has been a key issue for data mining applications, only a few existing techniques make use of the available concurrency from high performance computing hardware and software to address this issue in the context of GP analysis. Keane et al [20] proposed a data parallel approach for likelihood estimation in GP regression, but their method estimate the log likelihood locally, and hence the final outcome is not guaranteed to be the same as a sequential algorithm.

In this paper we make use of the fact that the covariance matrix encountered with GP for time series is a symmetric Toeplitz matrix and hence several solutions that have been proposed in literature [13, 18] can be utilized to make the hyper-parameter estimation algorithm computationally as well as memory efficient. Specifically, there have been many $O(t^2)$ algorithms developed to invert a Toeplitz matrix as well as solve a Toeplitz system of equations [37, 24, 9]. In our earlier work [6], we presented the adaptations of the algorithms by Trench [37, 41, 42] for the problem of scalable hyper-parameter estimation for GP.

2. Gaussian Process

A GP is a generalization of a Gaussian distribution and is defined as a collection of random variables, any finite number of which have a joint Gaussian distribution [31, Chapter 2]. A GP describes a distribution over a (potentially infinite) set of functions and is completely specified by its mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$[3]:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{1a}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \tag{1b}$$

where $\mathbf{x}$ is an input or index belonging to an input or index set $\mathcal{X}$. Typically the mean function is taken to be zero and the GP is written as:

$$f(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, k(\mathbf{x}, \mathbf{x}')) \tag{2}$$

Thus the above GP is a collection of random variables, where each random variable is the value of function $f(\mathbf{x})$ at location $\mathbf{x}$. When dealing with time series, the index set $\mathcal{X}$ is the set of time indices $\{1, 2, \ldots, T\}$, though a GP can be defined for more general forms of inputs such as $\Re^D$. For this paper, since we are dealing with time series, we will replace $\mathbf{x}$ with t to denote time. The covariance function k defines the covariance between the function values at two different time points:

$$cov(f(t_1), f(t_2)) = k(t_t, t_2) \tag{3}$$

Typically, a covariance function is specified using a set of parameters Θ, these are considered as the *hyper-parameters* for the GP. For example, a widely used covariance function, called *squared exponential* (se), can be written as:

$$k(t_1, t_2) = \sigma_f^2 exp(-\frac{\Delta t^2}{2l^2}) \text{ where } \Delta t = t_1 - t_2 \tag{4}$$

If the time series is periodic, such as the NDVI temporal profiles, an alternate covariance function, known as *Exponential Periodic (ep)*, can be used:

$$k(t_1, t_2) = \sigma_f^2 exp(-\frac{\Delta t^2}{2l^2\omega^2}) exp(-\frac{(1 - \cos\frac{2\pi\Delta t}{\omega})}{a}) \tag{5}$$

where ω is the length of a single cycle of the periodic time series.

[3]In this paper we will denote matrices with capital letters (K), vectors with small bold letters ($\mathbf{x}$, $\mathbf{s_i}$), and scalars with small letters (t,x_i).

2.1. **Time Series Prediction Using GP.** For GP based regression, it is assumed that the actual observations y_t are noisy versions of the function values $f(t)$ and the two quantities are related as:

$$y_t = f(t) + \varepsilon_t \tag{6}$$

where ε_t is a noise term that accounts for the noisy component of the observations. Traditionally, ε_t is assumed to be a Gaussian noise term $\sim \mathcal{N}(0, \sigma_n^2), \forall t$.

Given a noisy set of observations $\mathbf{y_{t-1}}$, the GP prior on $\{f(1), f(2), \ldots, f(t)\}$ (see (2)) and the relationship between y_t and $f(t)$ (see (6)), can be used to make a prediction at time t. The advantage of GP is that the prediction is not a value but a normal distribution $\sim \mathcal{N}(\hat{y}_t, \hat{v}_t)$, where the predictive mean $\hat{y}_t$ and predictive variance $\hat{v}_t$ are given by:

$$\hat{y}_t = K_{t\mathbf{t-1}}^{\mathsf{T}}(K_{\mathbf{(t-1)(t-1)}} + \sigma_n^2 I)^{-1}\mathbf{y_{t-1}} \tag{7}$$

$$\hat{v}_t = k(t,t) - K_{t\mathbf{t-1}}^{\mathsf{T}}(K_{\mathbf{(t-1)(t-1)}} + \sigma_n^2 I)^{-1}K_{t\mathbf{t-1}} \tag{8}$$

where $K_{\mathbf{(t-1)(t-1)}}$ is a $|\mathbf{t-1}| \times |\mathbf{t-1}|$ kernel matrix such that $K_{\mathbf{(t-1)(t-1)}}[i][j] = k(i,j)$. Similarly, $K_{t\mathbf{t-1}}$ is a $(t-1)$ length vector such that $K_{t\mathbf{t-1}}[i] = k(t,j)$.

Equations (7) and (8) allow one to use GP for time series prediction as well as other analysis tasks such as outlier/anomaly detection, noise removal, and change detection. But as can be observed in (7) and (8) handling the large covariance matrix, $(K_{\mathbf{(t-1)(t-1)}} + \sigma_n^2 I)$ is the key bottleneck for computing as well as memory resources.

For notational simplicity, we will drop the suffix t when referring to different quantities, wherever not necessary, and refer to the covariance matrix $(K_{\mathbf{(t-1)(t-1)}} + \sigma_n^2 I)$ as K and the observational time series as $\mathbf{y}$.

2.2. **Hyper-parameter Estimation Using Gradient Descent.** The hyper-parameters Θ associated with the covariance function can be calculated by minimizing the marginal log likelihood (l) for a training time series, which can be calculated as:

$$\begin{aligned} l = \log p(\mathbf{y}|\Theta) = & -\frac{1}{2}\mathbf{y}^{\mathsf{T}}K^{-1}\mathbf{y} \\ & -\frac{1}{2}\log|K| \\ & -\frac{n}{2}\log 2\pi \end{aligned} \tag{9}$$

The optimal hyper-parameters for the covariance function can be estimated by minimizing the function in (9) using a gradient based optimizing algorithm. The derivative of l_t with respect to a given hyper-parameter $\theta \in \Theta$ can be computed as ([31, Chapter 5]):

$$\frac{\partial l}{\partial \theta} = -\frac{1}{2}\mathbf{y}^{\mathsf{T}}K^{-1}\frac{\partial K}{\partial \theta}K^{-1}\mathbf{y} - \frac{1}{2}tr(K^{-1}\frac{\partial K}{\partial \theta}) \tag{10}$$

The computational complexity of the gradient based hyper-parameter estimation algorithm, referred to as *GPLearn*, requires computation of log-likelihood as well as the derivative of log-likelihood, which is $O(t^3)$, where t is the length of the time series $\mathbf{y}$, if standard inversion or Cholesky decomposition based methods are used. Moreover, the calculations require keeping the $O(t^2)$ matrix in the memory.

3. Gaussian Process Based Change Detection

We adapt the predictive capability of GP for time series to identify changes in an online fashion. The steps of the *GPChange* algorithm are shown in Algorithm 1.

The *GPChange* algorithm monitors the input time series from $(n+1)^{th}$ observation onwards. It uses GP to estimate the predictive distribution at time t, using observations available till time $(t-1)$ and then computes the p-value of for the actual observation y_t under the reference distribution, $\mathcal{N}(\hat{y}_t, \hat{\sigma}_t^2)$. A threshold $\alpha \in (0,1)$ is used to determine when the actual observation does not follow the predictive distribution, which is indicative of potential change. A running counter, a, is maintained

Input: $\mathbf{y_T}, n, \alpha, M, \Theta$
$a = 0$
foreach $t = n + 1$ *to* T **do**
  Compute $\hat{y}_t$ and $\hat{\sigma}_t^2$ (See (7) and (8))
  $p_t \leftarrow p$-value for y_t under $\mathcal{N}(\hat{y}_t, \hat{\sigma}_t^2)$
  if $p_t > \alpha$ **then**
    $a \leftarrow a + 1$ (Potential Alarm)
  else
    $a \leftarrow 0$ (Reset)
  end
  if $a \geq M$ **then**
    Raise Alarm
  end
end

Algorithm 1: Algorithm *GPChange*

to count the number of successive potential changes. An alarm for change is raised if the counter exceeds a threshold M.

The algorithm *GPChange* requires estimation of $\hat{y}_t$ and $\hat{\sigma}_t^2$ using (7) and (8). At time t, each of these steps are $O(t^3)$, since they involve solving two linear systems of equations. The calculations require keeping a t^2 sized covariance matrix in the memory at time t.

4. Efficient GP Analysis Using Toeplitz Matrices

In this section we present scalable methods for the algorithms *GPLearn* and *GPChange*. These methods were originally presented in our previous work [6], and are presented here for the sake of completeness. We assume that the covariance function used in the GP is *stationary* and only depends on the absolute difference between the inputs, i.e, $k(t_1, t_2) = k(|\Delta_t|)$. Many widely used covariance functions, such as the *squared exponential* function in (4) and the *exponetial periodic* function in (5) as well as the general *Matern* class of functions [31, Chapter 4] fall under this category of covariance functions.

4.1. **Characteristics of Covariance Matrix.** We first note that the covariance function which only depends on $|\Delta_t|$ will result in a symmetric *Toeplitz* matrix, K, as shown below:

$$K = \begin{pmatrix} k_0 & k_1 & k_2 & \dots & k_{t-1} \\ k_1 & k_0 & k_1 & \dots & k_{t-2} \\ k_2 & k_1 & k_0 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{t-1} & k_{t-2} & \dots & \dots & k_0 \end{pmatrix} \tag{11}$$

Moreover it can be shown that such functions result in a positive semi-definite covariance matrix while adding a σ_n^2 noise to the diagonal results in a positive definite covariance matrix. One can straightaway note that K in (11) can be represented using just the first row (or column) of K, henceforth denoted as κ. This characteristic straightaway provides a way to reduce the memory requirements of the algorithms involving K.

4.2. **Using Toeplitz Matrix Operations.** Several $O(t^2)$ algorithms have been proposed for Toeplitz matrix inversion which make use of the special matrix structure to compute the inverse [37, 24, 9]. But one can observe that a direct inversion of the covariance matrix K is not required to calculate the predictive distribution as well as the log-likelihood and its derivatives in (7)–(10). Instead, one only needs to calculate the following quantities:

(1) $k^\intercal K^{-1}\mathbf{y}$ (for (7))

Input: $(\kappa, \mathbf{y})$
Output: $\mathbf{z_t}$
if k_1 *!=* 1 **then**
  $\mathbf{k} \leftarrow \mathbf{k}/k_1$, $\mathbf{y} \leftarrow \mathbf{y}/k_1$
end
$\mathbf{z_1} \leftarrow y_1$, $\mathbf{g_1} \leftarrow -k_2$, $\lambda_1 \leftarrow 1 - k_2^2$
foreach $i = 1$ *to* $t-2$ **do**
  $\theta_i \leftarrow y_{i+1} - \mathbf{z_i}^\mathsf{T} \hat{\mathbf{k}}_{2:i+1}$
  $\gamma_i \leftarrow -k_{i+2} - \mathbf{g_i}^\mathsf{T} \hat{\mathbf{k}}_{2:i+1}$
  $\mathbf{z_{i+1}} \rightarrow \begin{bmatrix} \mathbf{z_i} + \frac{\theta_i}{\lambda_i}\hat{\mathbf{g}}_\mathbf{i} \\ \frac{\theta_i}{\lambda_i} \end{bmatrix}$
  $\mathbf{g_{i+1}} \rightarrow \begin{bmatrix} \mathbf{g_i} + \frac{\gamma_i}{\lambda_i}\hat{\mathbf{g}}_\mathbf{i} \\ \frac{\gamma_i}{\lambda_i} \end{bmatrix}$
  $\lambda_{i+1} \rightarrow \lambda_i - \frac{\gamma_i^2}{\lambda_i}$
end
$\theta_{t-1} \rightarrow y_t - z_{t-1}^\mathsf{T} \hat{\mathbf{k}}_{2:t}$
$\mathbf{z_t} \rightarrow \begin{bmatrix} \mathbf{z_{t-1}} + \frac{\theta_{t-1}}{\lambda_{t-1}}\hat{\mathbf{g}}_{t-1} \\ \frac{\theta_{t-1}}{\lambda_{t-1}} \end{bmatrix}$
return $\mathbf{z_t}$

Algorithm 2: *ToeplitzInverseSolve*

(2) $k^\mathsf{T} K^{-1} k$ (for (8))
(3) $\mathbf{y}^\mathsf{T} K^{-1} \mathbf{y}$ (for (9))
(4) $\log |K|$ (for (10))
(5) $\mathbf{y}^\mathsf{T} K^{-1} \frac{\partial K}{\partial \theta} K^{-1} \mathbf{y}$ (for (10))
(6) $tr(K^{-1} \frac{\partial K}{\partial \theta})$ (for (10))

One can use Cholesky decomposition and solve a system of equations using the Cholesky decomposition to compute each of these quantities but that will have $O(t^3)$ complexity to compute the decomposition and $O(t^2)$ memory requirement for the covariance matrix K.

We will show that each of these four quantities can be computed in a computational as well as memory efficient manner:

4.2.1. *Computing* $\mathbf{y}^\mathsf{T} K^{-1} \mathbf{y}$, $k^\mathsf{T} K^{-1} \mathbf{y}$, $k^\mathsf{T} K^{-1} k$. Algorithm 2 shows how one can compute $K^{-1}\mathbf{y}$ (or $K^{-1}\mathbf{y}$), i.e., solving a Toeplitz system of equations. This algorithm was originally proposed by Trench [38, 42] for Toeplitz matrices and we simplify it for the symmetric case. This algorithm takes the first row of the covariance matrix,$\kappa = \{k_1, k_2, \ldots, k_t\}$, as input and returns the solution vector. In the algorithm $\hat{\mathbf{x}}$ denotes a vector obtained by reversing the vector $\mathbf{x}$. A portion of a vector is denoted as $\mathbf{x}_{i:j}$.

Note that this algorithm is $O(t^2)$ and has $O(t)$ memory requirement.

4.2.2. *Computing* $\log |K|$. It has been shown that the determinant of the matrix K can be computed as a by-product of the Algorithm 2 by simply taking the product of λ_is [41], i.e.,

$$\log |K| = t \log k_1 \sum_{i=1}^{t-1} \log \lambda_i \tag{12}$$

Thus $\log |K|$ can be computed in linear time without requiring any additional memory.

4.2.3. *Computing* $\mathbf{y}^\mathsf{T} K^{-1} \frac{\partial K}{\partial \theta} K^{-1} \mathbf{y}$. Algorithm 2 computes $K^{-1}\mathbf{y}$. The vector $K^{-1}\mathbf{y}$ can be multiplied with $\frac{\partial K}{\partial \theta}$ in $O(n^2)$ time and the resulting vector can be multiplied with $\mathbf{y}^\mathsf{T}$ in linear time. Note

Input: κ
Output: $\mathbf{s}$
alpha $\leftarrow ToeplitzInverseSolve(\mathbf{k_{1:t-1}}, \mathbf{k_{2:t}})$
$\gamma \leftarrow \frac{1}{k_1 + \mathbf{k_{2:t}}\alpha}$
nu $\leftarrow [\gamma \hat{\alpha} \gamma]^{\mathsf{T}}$
foreach $k = 0$ *to* $t-1$ **do**
    $s_k \leftarrow \frac{1}{\gamma} \sum_{j=1}^{t-k} (2i + k - n + 1)\nu_i \nu_{i+k}$
    s_k is the sum of the k^{th} diagonal starting from main diagonal ($k = 0$).
end
return s

Algorithm 3: *ToeplitzDiagonalSums*

that since $\frac{\partial K}{\partial \theta}$ is Toeplitz, it can be multiplied using only one representative row of the matrix, i.e., with $O(n)$ space requirements.

4.2.4. *Computing* $tr(K^{-1}\frac{\partial K}{\partial \theta})$. Let $L = K^{-1}$ and $P = \frac{\partial K}{\partial \theta}$. We are interested in computing $tr(LP) = tr(PL)$ where P is a symmetric Toeplitz matrix and L is the inverse of a symmetric Toeplitz matrix. We can write:

$$\begin{aligned} tr(PL) &= \sum_{i=1}^{t}\sum_{j=1}^{t} p_{ij} l_{ij} \\ &= \sum_{i=1}^{t}\sum_{j=1}^{t} p_{|i-j(=k)|} l_{ij} \\ &= \sum_{k=-t+1}^{t=1} p_{|k|} \sum_{j=k+1}^{t} l_{j-k,j} \\ &= p_0 \sum_{j=1}^{t} l_{jj} + 2\sum_{k=1}^{t-1} p_k \sum_{j=k+1}^{t} l_{j-k,j} \end{aligned}$$

Note that each summation $\sum_{j=k+1}^{t} l_{j-k,j}, \forall k = 0 \ldots t-1$ is nothing but the sum of the k^{th} diagonal of L. Given the diagonal sums for $L(= K^{-1})$ we can compute $tr(K^{-1}\frac{\partial K}{\partial \theta})$ in linear time. An $O(n^2)$ algorithm for the computation of the diagonal sums is shown in Algorithm 3. The proof of correctness of the algorithm was given by Dias and Leitao [7].

The computational complexity involved with computing $tr(K^{-1}\frac{\partial K}{\partial \theta})$ using Algorithm 3 is $O(n^2)$ with $O(n)$ memory required.

5. Efficient Change Detection and Hyper-parameter Estimation

In Section 4 we have provided fast and memory efficient methods to compute various quantities required for the GP based change detection and hyper-parameter estimation. These methods can be used instead of traditional matrix operations, we refer to the change detection and hyper-parameter estimation methods which use these Toeplitz matrix based methods, as *GPChangeFast* and *GPLearnFast*, respectively.

5.1. **Handling Multiple Time Series for Prediction and Hyper-parameter Estimation.** In many scenarios one needs to estimate the GP hyper-parameters with respect to multiple time series. Let $\mathbf{Y} = [\mathbf{y_1 y_2} \ldots \mathbf{y_p}]$ be a set of input time series. The total marginal log likelihood for all time series will be equal to the sum of marginal log likelihoods for individual time series using (9),

i.e.,

$$\log p(\mathbf{Y}|\Theta) = \sum_{i=1}^{p} \log p(\mathbf{y_i}|\Theta) \tag{13}$$

Same holds for the computation of the derivative of the total marginal log likelihood with respect to a hyper-parameter. One can compute these two quantities in a loop using the results in Section 4 resulting in a $O(pt^2)$ complexity.

Similarly, often one needs to run the *GPChange* algorithm on multiple time series using the same set of hyper-parameters. Instead of repeatedly invoking *GPChange* for each time series in $\mathbf{Y}$, one can modify Algorithm 1 such that $\hat{y}_t$ is computed for each of the time series in $\mathbf{Y}$, while $\hat{\sigma}_t^2$ is only required to be computed once.

5.2. **Parallel Version of *GPChangeFast* and *GPLearnFast*.** For the parallel version, we assign the task of handling each time series $\mathbf{y} \in \mathbf{Y}$ to a different processing unit. We refer to the parallel versions of the change detection and hyper-parameter estimation algorithms as *GPChangeFastP* and *GPLearnFastP*, respectively.

For our experiments, we used a POSIX thread based implementation and an MPI based parallel implementation. The same algorithm can be implemented using *Map-Reduce* for cloud based computing architectures or for GPU based architectures using CUDA. The linear data size required by each child node is especially attractive for the latter, where the amount of data transferred between nodes can be a significant bottleneck.

6. Experimental Results

In this section we present results from two sets of experiments. First set of experiments show how well the proposed Toeplitz matrix based methods scale in comparison to the traditional methods. The second set of experiments demonstrate the effectiveness of the *GPChange* algorithm in identifying changes in six year NDVI time series data.

6.1. **Performance Results.** In this section we compare the computational performance of the proposed algorithm *GPLearnFast* against the standard algorithm (*GPLearnSlow*) for computing log-likelihoods and derivatives. We also investigate the performance of the multi-threaded and MPI based versions of *GPLearnFast*, referred to as *GPLearnFastThread* and *GPLearnFastMPI*. All experiments are done on time series with varying lengths. All algorithms were implemented in C using low level CBLAS routines[4]. The *GPLearnLow* algorithm used cholesky decomposition from the LAPACK library[5] to solve the system of equations and compute the inverse, etc.

All experiments were run on an SGI Altix ICE 8200 cluster called Frost[6]. Frost is currently configured with 128 compute nodes each having 16 virtual cores (2048 way concurrency) and 24GB of memory, infiniband interconnects, and a gigabit ethernet network. Each node is capable of supporting 16 threads.

6.1.1. *Performance of* GPLearnFast *vs.* GPLearnSlow. We first compare the performance of the computational and memory efficient *GPLearnFast* algorithm against the standard *GPLearnSlow* algorithm. Figure 1 shows the performance of the two algorithms on single time series with varying lengths of the time series. Note that the *GPLearnSlow* algorithm requires a $O(n^2)$ space in the memory and hence could not run for time series more than 100000 length, while the *GPLearnFast* algorithm has no memory related issue in dealing with time series as long as 1000000. Figure 1 shows that *GPLearnFast* is significantly faster than *GPLearnSlow*, with a maximum speedup of 137 achieved for time series of length 100000.

[4] `http://www.netlib.org/blas/index.html`
[5] `http://www.netlib.org/lapack/`
[6] `http://www.nccs.gov/computing-resources/frost/`

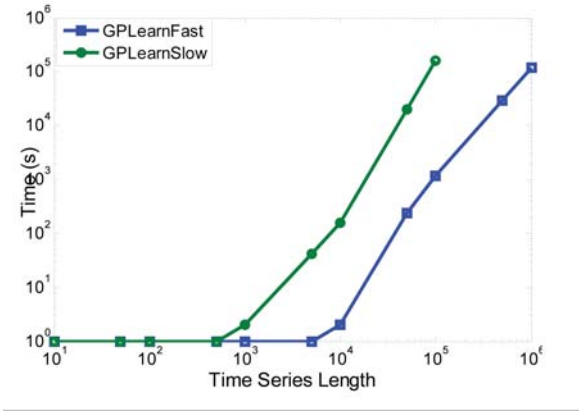
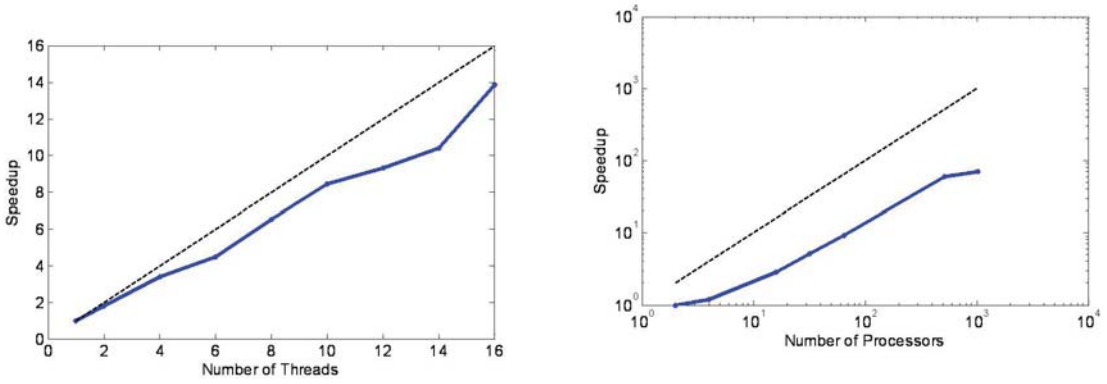


FIGURE 1. Performance Comparison of GPLearnFast and GPLearnSlow. Both axes are in logscale.

6.1.2. *Performance of Parallel Version.* To study the performance of the thread based and MPI based parallel versions, we used the NDVI data collected from the MODIS instrument. Global MODIS data is organized into non-overlapping tiles, where each image or tile is roughly 4800 rows × 4800 columns at 250 meters pixel resolution. We collected MODIS imagery from 2001 to 2006 for Iowa region, preprocessed and generated 16 day NDVI images (23 composite images per year). Final preprocessed Iowa image size contains 4,276,383 locations, where each location is a time series of length 138.

The speedup results (over the serial *GPChangeFast*) for the multi-threaded implementation, *GPLearnFastThread*, are shown in Figure 2a, and speedup results for MPI implementation, *GPLearnFastMPI*, are shown in Figure 2b. Speedup results in Figure 2 demonstrate that the GP based



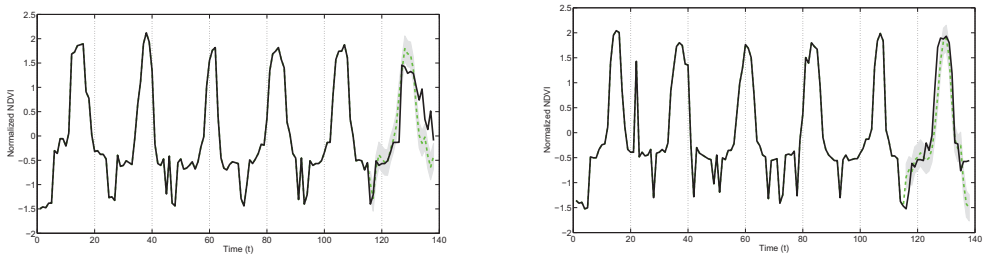
(A) Threads (B) MPI

FIGURE 2. Speedups for *GPLearnFastThread* and *GPLearnFastMPI* over serial *GPLearnFast*. Both axes for right figure are in logscale.

learning algorithm can be parallelized to achieve significant speedups. For the muli-threaded version (Figure 2a), the speedup is close to linear with the number of threads, but for the MPI based version (Figure 2b), the speedup is sub-linear, for 1024 nodes, the speedup achieved is 70. One reason for

this is the high communication cost entailed in sending the time series data from the master to the slave nodes, which results in high overhead costs, thereby offsetting the parallelization speedups. In future, we will develop methods that can further minimize the communication overheads, and achieve better speedups.

6.2. **Detecting Changes in NDVI Data.** In this section we show the effectiveness of the proposed *GPChange* algorithm in identifying changes in the NDVI data for Iowa state. The task is to use the first 5 years of data for training and identifying changes in the final year.



(A) Change (B) No Change

FIGURE 3. Results of *GPChange* on two NDVI time series.

(A) Change

Time (t)	P-value (p_t)	Possible?	Alarm?
116	0.00		
117	0.30	√	
118	0.13	√	
119	0.05		
120	0.23	√	
121	0.42	√	
122	0.39	√	
123	0.32	√	
124	0.35	√	√
125	0.01		
126	0.00		
127	0.17	√	
128	0.00		
129	0.00		
130	0.00		
131	0.30	√	
132	0.00		
133	0.00		
134	0.00		
135	0.00		
136	0.00		
137	0.00		
138	0.01		

(B) No Change

Time (t)	P-value (p_t)	Possible?	Alarm?
116	0.00		
117	0.00		
118	0.09		
119	0.09		
120	0.36	√	
121	0.15	√	
122	0.29	√	
123	0.00		
124	0.09		
125	0.02		
126	0.00		
127	0.00		
128	0.00		
129	0.06		
130	0.07		
131	0.01		
132	0.00		
133	0.28	√	
134	0.24	√	
135	0.00		
136	0.00		
137	0.00		
138	0.00		

TABLE 1. Labels assigned by *GPChange* to testing portion of two NDVI time series. Thresholds $\alpha = 0.1$ and $M = 5$.

Figure 3a shows results on a NDVI time series containing a permanent change in the sixth year, possibly a damaged crop. The same plot also shows the GP based prediction ($\hat{y}_t$) as dashed green

line as well as the bounds specified by the predictive variance ($\hat{\sigma}_t^2$) as grayed region. The locations where the p-value exceeds the threshold α (i.e. potential changes) are specified in Table 1(a). For these experiments we chose α threshold to be 0.1 and M threshold to be 5. Figure 3a and Table 1(a) show that *GPChange* is able to identify the true change after identifying 5 consecutive possible change points. For comparison, Figure 3b shows another NDVI time series which does not contain a change in the sixth year. The plots indicate that the GP based prediction follows the observed data well, and even though it identifies isolated possible changes (Table 1(b)), due to the presence of inherent noise in the data, the number of consecutive possible change points are not sufficient to raise an alarm for actual change.

7. Conclusions

GP based methods typically scale as $O(t^3)$ with the size of the input data with a memory requirement of $O(t^2)$. This makes them impractical in domains that encounter massive time series data sizes such as remote sensing, ECG analysis, etc. In this paper we have shown how Gaussian process analysis can be scaled to handle massive time series data sets. We have proposed an online change detection algorithm that has been shown to effectively identify changes in NDVI time series, making highly suitable for biomass monitoring at regional as well as global scale.

The parallelization demonstrated using the thread based and MPI implementations, indicate that GP analysis is naturally suited for parallelization and hence can be further scaled by utilizing the available as well as emerging computing architectures such as heterogeneous processing units and cloud computing.

While the proposed algorithms utilize the special structure of the underlying covariance matrix to produce an exact solution, in future this algorithm can be combined with the existing work in the area of approximate GP methods to achieve further speedups while staying close to the exact solution.

8. Acknowledgments

Prepared by Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, Tennessee 37831-6285, managed by UT-Battelle, LLC for the U. S. Department of Energy under contract no. DEAC05-00OR22725. This research is funded through the LDRD program at ORNL.

References

[1] R. P. Adams and D. J. MacKay. Bayesian online changepoint detection. Technical report, University of Cambridge, Cambridge, UK, 2007. arXiv:0710.3742v1.

[2] M. Basseville. Detecting changes in signals and systems–a survey. *Automatica*, 24(3):309 – 326, 1988.

[3] S. Boriah, V. Kumar, M. Steinbach, C. Potter, and S. Klooster. Land cover change detection: a case study. In *Proceeding of the 14th KDD*, pages 857–865, 2008.

[4] S. Brahim-Belhouari and J. Vesin. Bayesian learning using gaussian process for time series prediction. In *Statistical Signal Processing, 2001*, pages 433–436, 2001.

[5] B. J. Brewer and D. Stello. Gaussian process modelling of asteroseismic data. *Monthly Notices of the Royal Astronomical Society*, 395(4):2226–2233, June 2009.

[6] V. Chandola and R. R. Vatsavai. Scalable hyper-parameter estimation for gaussian process based time series analysis. In *Proceedings of The 2nd KDD Workshop on Large-scale Data Mining: Theory and Applications*, 2010.

[7] J. Dias and J. Leitao. Efficient computation of trTR-1 for toeplitz matrices. *Signal Processing Letters, IEEE*, 9(2):54 –56, feb 2002.

[8] P. Drineas and M. W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

[9] J. Durbin. The fitting of time-series models. *Revue de l'Institut International de Statistique / Review of the International Statistical Institute*, 28(3):233–244, 1960.

[10] Y. Fang, A. R. Ganguly, N. Singh, V. Vijayaraj, N. Feierabend, and D. T. Potere. Online change detection: Monitoring land cover from remotely sensed data. In *ICDMW '06*, pages 626–631, 2006.

[11] L. Foster, A. Waagen, N. Aijaz, M. Hurley, A. Luis, J. Rinsky, C. Satyavolu, M. J. Way, P. Gazis, and A. Srivastava. Stable and efficient gaussian process calculations. *J. Mach. Learn. Res.*, 10:857–882, 2009.

[12] A. Girard, C. E. Rasmussen, J. Quinonero-Candela, J. Q. N. Candela, M. Modelling, and R. Murray-smith. Gaussian process priors with uncertain inputs - application to multiple-step ahead time series forecasting. In *Advances in Neural Information Processing Systems*, pages 529–536. MIT Press, 2003.
[13] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins Press, Baltimore, MD, USA, second edition, 1989.
[14] C. M. Gruner and D. H. Johnson. Detection of change in periodic, nonstationary data. In *ICASSP '96*, pages 2471–2474, 1996.
[15] V. Guralnik and J. Srivastava. Event detection from time series data. In *KDD '99*, pages 33–42, 1999.
[16] C. Inclán and G. C. Tiao. Use of cumulative sums of squares for retrospective detection of changes of variance. *Journal of the American Statistical Association*, 89(427):913–923, 1994.
[17] C. O. Justice, E. Vermote, J. R. Townshend, R. Defries, D. P. Roy, D. K. Hall, V. V. Salomonson, J. L. Privette, G. Riggs, A. Strahler, W. Lucht, R. B. Myneni, Y. Knyazikhin, S. W. Running, S. W. Steve W. Nemani, Z. Wan, A. R. Huete, W. van Leeuwen, R. E. Wolfe, L. Giglio, J.-P. Muller, P. Lewis, and M. J. Barnsley. The moderate resolution imagin spectrradiometer (MODIS): Land remote sensing for global chang research. *IEEE Transactions on Geosciences and Remote Sensing*, 36:1228–1249, 1998.
[18] T. Kailath and A. H. Sayed, editors. *Fast reliable algorithms for matrices with structure*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.
[19] S. R. Karlsen, A. Tolvanen, E. Kubin, J. Poikolainen, K. A. Hgda, B. Johansen, F. S. Danks, P. Aspholm, F. E. Wielgolaski, and O. Makarova. Modis-ndvi-based mapping of the length of the growing season in northern fennoscandia. *International Journal of Applied Earth Observation and Geoinformation*, 10(3):253 – 266, 2008.
[20] A. J. Keane, A. Choudhury, P. B. Nair, P. B. Nair, A. J. K. F. Choudhury, and P. B. Nair. A data parallel approach for large-scale gaussian process modeling. In *in Proc. the Second SIAM International Conference on Data Mining*, 2002.
[21] J. Kucera, P. Barbosa, and P. Strobl. Cumulative sum charts - a novel technique for processing daily time series of modis data for burnt area mapping in portugal. In *MultiTemp 2007*, pages 1–6, July 2007.
[22] T. L. Lai. Sequential changepoint detection in quality control and dynamical systems. *Journal of the Royal Statistical Society. Series B*, 57(4):613–658, 1995.
[23] D. Lambert and C. Liu. Adaptive thresholds monitoring streams of network counts online. *Journal of the American Statistical Association*, 101(473):78–88, March 2006.
[24] N. Levinson. The Wiener RMS error criterion in filter design and prediction. *Journal of Mathematics and Physics*, 25(4):261–278, 1947.
[25] R. S. Lunetta, J. F. Knight, J. Ediriwickrema, J. G. Lyon, and L. D. Worthy. Land-cover change detection using multi-temporal MODIS NDVI data. *Remote Sensing of Environment*, 105(2):142–154, 2006.
[26] J. F. Mas. Monitoring land-cover changes: a comparison of change detection techniques. *International Journal of Remote Sensing*, 20(1):139–152, January 1999.
[27] T. Ogden and E. Parzen. Change-point approach to data analytic wavelet thresholding. *Statistics and Computing*, 6(2):93–99, 1996.
[28] A. O'Hagan and J. F. C. Kingman. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society. Series B*, 40(1):1–42, 1978.
[29] E. S. Page. On problems in which a change can occur at an unknown time. *Biometrika*, 44(1-2):248–252, 1957.
[30] S. Patra, S. Ghosh, and A. Ghosh. Unsupervised change detection in remote-sensing images using one-dimensional self-organizing feature map neural network. *ICIT '06*, pages 141–142, 2006.
[31] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2005.
[32] M. K. Ridd and J. Liu. A comparison of four algorithms for change detection in an urban environment - a remote sensing perspective. *Remote Sensing of Environment*, 63(2):95–100, 1998.
[33] T. Sakamoto, M. Yokozawa, H. Toritani, M. Shibayama, N. Ishitsuka, and H. Ohno. A crop phenology detection method using time-series modis data. *Remote Sensing of Environment*, 96(3-4):366 – 374, 2005.
[34] M. Sharifzadeh, F. Azmoodeh, and C. Shahabi. Change detection in time series data using wavelet footprints. *Advances in Spatial and Temporal Databases*, pages 127–144, 2005.
[35] B. W. Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 47(1):1–52, 1985.
[36] C. Tarantino, P. Blonda, and G. Pasquariello. Application of change detection techniques for monitoring man-induced landslide causal factors. In *IGARSS '04*, volume 2, pages 1103–1106, Sept. 2004.
[37] W. F. Trench. An algorithm for the inversion of finite toeplitz matrices. *SIAM Journal on Applied Mathematics*, 12(3):515–522, 1964.
[38] W. F. Trench. Weighting coefficients for the prediction of stationary time series from the finite past. *SIAM Journal on Applied Mathematics*, 15(6):1502–1510, 1967.
[39] M. A. White and R. R. Nemani. Real-time monitoring and short-term forecasting of land surface phenology. *Remote Sensing of Environment*, 104(1):43 – 49, 2006.

[40] C. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.
[41] S. Zohar. Toeplitz matrix inversion: The algorithm of W. F. Trench. *J. ACM*, 16(4):592–601, 1969.
[42] S. Zohar. The solution of a toeplitz set of linear equations. *J. ACM*, 21(2):272–276, 1974.