A Gaussian Process-based Streaming Algorithm for Prediction of Time Series With Regimes and Outliers

Daniel Waxman
Department of Electrical and Computer Engineering
Stony Brook University
Stony Brook, New York, USA
Email: daniel.waxman@stonybrook.edu

Petar M. Djurić
Department of Electrical and Computer Engineering
Stony Brook University
Stony Brook, New York, USA
Email: petar.djuric@stonybrook.edu

***Abstract*—Online prediction of time series under regime switching is a widely studied problem in the literature, with many celebrated approaches. Using the non-parametric flexibility of Gaussian processes, the recently proposed INTEL algorithm provides a product of experts approach to online prediction of time series under possible regime switching, including the special case of outliers. This is achieved by adaptively combining several candidate models, each reporting their predictive distribution at time t. However, the INTEL algorithm uses a finite context window approximation to the predictive distribution, the computation of which scales cubically with the maximum lag, or otherwise scales quartically with exact predictive distributions. We introduce LINTEL, which uses the *exact* filtering distribution at time t with *constant-time* updates, making the time complexity of the streaming algorithm optimal. We additionally note that the weighting mechanism of INTEL is better suited to a mixture of experts approach, and propose a fusion policy based on arithmetic averaging for LINTEL. We show experimentally that our proposed approach is over five times faster than INTEL under reasonable settings with better quality predictions.**

I. INTRODUCTION

Online prediction for time series is an important problem in many machine learning and signal processing applications. It has accordingly received extensive attention from the perspectives of kernel learning [1], recurrent neural networks [2], and Gaussian processes (GPs) [3], among others. The problem is especially interesting when outliers or regime switches are present in the data, as outliers can contaminate a model's predictions if care is not taken to avoid this.

This paper focuses on improving the INstant TEmporal structure Learning (INTEL) algorithm [3], a recently proposed method that models time series as Gaussian processes, and accounts for outliers and regime switches by dynamically adjusting the current data used for inference. This is accomplished by prespecifying several "candidate models" and using a (generalized) product of experts approach for fusing estimates. The weights for each model are also dynamically adjusted in a modified version of Bayesian model averaging (BMA) – an approach that has recently proved successful in several other Bayesian online learning problems [4], [5].

One drawback to the GP approach of INTEL is the need to recompute the predictive distributions of each candidate GP at every time step. Using the typical kernel approach, the computation of the predictive distribution at time t scales with $\mathcal{O}(t^3)$ and quickly becomes prohibitively expensive. As a result, INTEL approximates the predictive distribution, only using the previous τ data points instead. If τ is small, the algorithm will be tractable, but as τ grows smaller, the approximation becomes more severe.

Instead, we propose to use Markovian GPs and their corresponding state-space representation [6]. Markovian GPs admit a representation as a linear time-invariant stochastic differential equation, which allows for online, exact predictive distributions via Kalman filtering. Accordingly, our algorithm LINTEL (short for Linear INTEL) provides constant time-complexity updates using the exact predictive distribution. Using the state-space representation of GPs, we also draw connections between the INTEL algorithm and the 3σ-rejection Kalman filter [7, Ch. 7]. We additionally make our code available open-source at https://www.github.com/DanWaxman/Lintel.

The rest of this paper is structured as follows: in Section II, we provide background in GPs, which is used in Section III, where we discuss the INTEL algorithm. In Section IV, we extend our discussion on GPs to filtering for inference with linear time complexity. This is used in Section V, where the LINTEL algorithm is introduced. We perform a series of experiments on synthetic and real data in Section VI, followed by brief discussion and conclusions in Section VII.

II. GAUSSIAN PROCESS REGRESSION

GPs are a powerful tool in Bayesian machine learning, being non-parametric, flexible, and theoretically tractable [8]. In this section, we provide a brief overview of their use in GP regression using kernels (Section II-A), and some computational drawbacks associated with the kernel approach (Section II-B).

A. Gaussian Process Regression with Kernels

GPs are stochastic processes $\{f(t)\}_{t\in\mathcal{T}}$ defined by the property that for any finite index set $\{t_1, \ldots, t_{N_{\text{data}}}\}$, the random variables $\{f(t_1), \ldots, f(t_{N_{\text{data}}})\}$ are jointly Gaussian distributed. In the signal processing and machine learning communities, GPs are typically presented in a functional space with a kernel formulation, whereby the covariance between two input points

This work was supported by the National Science Foundation under Award 2212506.

$\mathbf{x}, \mathbf{x}'$ is specified by a kernel function $\kappa(\mathbf{x}, \mathbf{x}')$. By additionally specifying a mean function $\mu(t)$, the GP is fully determined, and written as $f \sim \mathcal{GP}(\mu, \kappa)$.

In GP regression, we place a GP prior on an unknown function f and specify a likelihood for $y|t, f$ to obtain a posterior GP. The most common likelihood is also Gaussian, in which case the data-generating process is described by

$$f \sim \mathcal{GP}(\mu, \kappa),$$
$$y = f(t) + \varepsilon,$$

where the observation noise ε is i.i.d. Gaussian with variance σ_n^2. When Gaussian likelihoods are used, the model is conjugate, allowing for analytic posterior computations. In particular, given training data $\mathcal{D} = \{(t_n, y_n)\}_{n=1}^{N_{\text{data}}}$, the predictive distribution of the posterior GP at input t_* is given by

$$f(t_*)|\mathcal{D} \sim \mathcal{N}(m_*, s_*^2), \tag{1}$$
$$m_* = \mu(t) + \mathbf{k}_{*\odot} \left(\mathbf{K}_{\odot\odot} + \sigma_n^2 \mathbb{1}\right)^{-1} \boldsymbol{\delta}, \tag{2}$$
$$s_*^2 = K_{**} - \mathbf{k}_{*\odot} \left(\mathbf{K}_{\odot\odot} + \sigma_n^2 \mathbb{1}\right)^{-1} \mathbf{k}_{\odot *}, \tag{3}$$

where $[\mathbf{K}_{\odot\odot}]_{ij} \triangleq \kappa(t_i, t_j)$ is the cross-covariance matrix of the training data, $\mathbf{k}_{*\odot} = \mathbf{k}_{\odot *}^\top$ is a row vector whose entries are $\kappa(t_i, t_*)$, $k_{**} = \kappa(t_*, t_*)$ is the variance at the test point, and the mean function is used for the row vector $\boldsymbol{\delta}$ whose entries are $y_n - \mu(t_n)$.

A typical example of kernel function which is used in this work is the Matérn-$5/2$ kernel, which is given by

$$k_{\text{Mat-5/2}}(t, t') = \sigma_f^2 \left(1 + \frac{\sqrt{5}}{3\ell} + \frac{5}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}r}{\ell}\right), \tag{4}$$

where $r = |t - t'|$ and the process scale σ_f and length scale ℓ are treated as hyperparameters. GPs with kernels which only depend on r are known as stationary. The hyperparameters, which we will denote as $\boldsymbol{\psi}$, are often trained by maximizing the evidence.

B. Drawbacks of the Kernel Approach

While the predictive quantities given in Eqs. (1) to (3) are analytically tractable, their expressions involve the inversion of an $N_{\text{data}} \times N_{\text{data}}$ matrix. In practice, the memory cost (and some of the computational cost) of a full matrix inversion can be avoided by instead solving the corresponding linear systems using Cholesky decompositions (see, e.g., [8, App. A]). However, the computational complexity still scales cubically with N_{data}, making exact GP inference prohibitively costly for large datasets. This can be particularly problematic in the context of long time series, where N_{data} can be several thousands of points long.

Another drawback of using kernels for GP regression is difficulties with updating the training set. In particular, in an online setting, we are often interested in the predictive distribution $p(y_n|y_{1:n-1})$; using kernels, it is generally difficult to directly use $p(y_n|y_{1:n-1})$ to calculate $p(y_{n+1}|y_{1:n})$ without recomputing some part of the Cholesky decomposition of $\mathbf{K}_{\odot\odot}$, and one must resort to approximations for efficient (i.e., bounded) online updates [9], [10]. Note that in the remainder of the paper, the index n corresponds to the time instant t_n.

III. The Intel Algorithm

The Intel algorithm [3] uses several candidate GPs $f_1, \ldots, f_K$ that are fused to perform online prediction, allowing for the presence of outliers or the possibility of regime switches. We first describe the weighting and fusion mechanism of Intel (Section III-A), and subsequently its outlier and change point detection mechanism (Section III-B). We conclude with some notes about initialization and various approximations used in the algorithm (Section III-C).

A. Weighting and Fusing

The Intel algorithm fuses the predictions of candidate GPs $f_1, \ldots, f_K$ with predictive densities $p_1(y_{n+1}|\mathcal{D}_n), \ldots, p_K(y_{n+1}|\mathcal{D}_n)$ using a weighted product of experts framework [11], where $\mathcal{D}_n$ includes all data at the current time used for prediction — we will elaborate more on its contents in Section III-B. Each GP is assumed to have its own hyperparameters $\boldsymbol{\psi}_k$, but share a common (constant) mean function $\mu(\cdot) = C$. We first describe the weights, and then the fusion rule, since the former is required for the latter.

The weighting in Intel is based on a modification to BMA where a forgetting factor $0 \leq \alpha \leq 1$ is used, leading to a recursive update to the filtered weight

$$\tilde{w}_{k,n+1} \propto w_{k,n}^\alpha, \tag{5}$$

which are normalized to sum to unity and used for prediction. After y_{t+1} is observed, the weights are updated using the likelihood $p_k(y_{n+1}|\mathcal{D}_n)$ according to the rule

$$w_{k,n+1} \propto \tilde{w}_{k,n+1} p_k(y_{n+1}|\mathcal{D}_n), \tag{6}$$

which are once again normalized to sum to unity.

Once the weights are determined, the predictive densities of each candidate model are geometrically averaged. In particular, a weighted product of experts is proposed, where

$$p(y_{n+1}|\mathcal{D}_n) \propto \prod_{k=1}^K p_k(y_{n+1}|\mathcal{D}_n)^{\tilde{w}_k}.$$

In the GP literature, such a model is known as a generalized product of experts, and was introduced in [12]. Since each $p_k(\cdot|\cdot)$ is Gaussian, $p(y_{n+1}|\mathcal{D}_n)$ is also Gaussian, with mean and variance

$$m_{n+1} = \frac{\sum_{k=1}^K m_{k,n+1}/\tilde{w}_{k,n+1}\sigma_{k,n+1}^{-2}}{\sum_{k=1}^K \tilde{w}_{k,n+1}\sigma_{k,t+1}^{-2}}, \tag{7}$$
$$s_{t+1}^2 = \left(\sum_{k=1}^K \tilde{w}_{k,n+1}\sigma_{k,t+1}^{-2}\right)^{-1}. \tag{8}$$

For a fixed set of weights that sum to unity, arithmetic and geometric averaging can be understood to minimize a weighted sum of Kullback-Liebler divergences in opposite directions [13]. Typical methods to determine weights include relative entropy with respect to the prior [12], optimizing functionals of the

covariance matrix [14], and Bayesian learning [15], which can also be input-dependent [16]. The selection of weights is important, as the relative predictive performance of arithmetic and geometric fusion depends on the choice of weights [13].

B. Outlier and Change Point Detection

Using the fused estimate, a 3σ credible interval is constructed. If the data point y_{n+1} is within the 3σ credible interval $[m_{n+1}-3s_{n+1}, m_{n+1}+3s_{n+1}]$, it is then added to the data $\mathcal{D}_n$. Otherwise, it is declared an outlier and added to the potential changepoint bucket (PCB), denoted by $\mathcal{D}'_n$. If the PCB reaches a predetermined length N_{PCB}, a regime-switch is announced; in this case, the data is reset with $\mathcal{D}_n \leftarrow \mathcal{D}'_n$, and $\mu(\cdot)$ is set to the constant function with mean given by $\frac{1}{N_{\text{PCB}}}\sum_{y\in\mathcal{D}'_n} y$. After a datum that is not an outlier is observed, the PCB is reset, $\mathcal{D}'_n \leftarrow \{\}$.

Note that this detection mechanism for outliers and regime switches is not perfect; for example, a regime switch might *reduce* the variance of the observation noise, in which case all points would lay well within 3σ. However, in this case, the weighting and fusion mechanism will rapidly adapt to the "correct" model. In this sense, the PCB is not the only mechanism for regime switches, but serves as a rapid detection mechanism when extreme and sudden changes occur.

C. Initialization and Approximations

As previously mentioned in Section II-B, the computation of $p(y_{n+1}|\mathcal{D}_n)$ requires $\mathcal{O}(n^3)$ time, which is prohibitive as n grows large. As a result, the authors of INTEL propose using a finite context window of length τ, using $p(y_{n+1}|\tilde{\mathcal{D}}_n)$ instead of $p(y_{n+1}|\mathcal{D}_n)$ where $\tilde{\mathcal{D}}_n$ is the intersection of $\mathcal{D}_n$ and the finite context window of length τ, i.e.,

$$\tilde{\mathcal{D}}_n = \mathcal{D}_n \cap \{(t_{n'}, y_{n'})\}_{n'=n-\tau+1}^{n}.$$

The GP is then recomputed at every timestep using the last τ observations, resulting in $\mathcal{O}(\tau^3)$ updates.

To create a list of candidate models, a small subset of data $\mathcal{D}_0$ is set aside, and the hyperparameters are selected by evidence maximization. The authors of INTEL then advocate for creating several candidate models using prior knowledge about the system; for example, in a CPU utilization data, they choose a candidate model such that the process scale is $1/5$th that of the evidence maximization estimate, based on prior knowledge that volatility might shrink.

Two general remarks are in order:

Remark 1: Note that the Cholesky decomposition $L = \text{chol}(\mathbf{K}_{\odot\odot} + \sigma_\varepsilon^2 \mathbb{1})$ must be recomputed at each time step because we do not assume that the data $t_1, \ldots, t_{N_{\text{data}}}$ are equally-spaced. Even so, an improvement can be made to the INTEL algorithm by performing rank-1 updates to L, which can be performed in $\mathcal{O}(\tau^2)$ time. While these updates are worthwhile in practice, our proposed method allows for constant-time updates without windowing.

Algorithm 1 INTEL [3]

Input: Window Size τ, Maximum PCB Size N_{PCB}, Prior Mean C, Forgetting Factor α, Mean Update Period L, Initial Weights $\mathbf{w}_0$, Kernel Hyperparameters $\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_K$
Output: Output mean and variance m_n, s_n^2 and outlier flag $\texttt{is\_outlier}_n$ for $n = 1, 2, \ldots$

$\tilde{\mathcal{D}} \leftarrow \{\}$
$\mathcal{D}' \leftarrow \{\}$
$t_{\text{last\_mean\_update}} \leftarrow 0$
for $n = 1, 2, \ldots$ **do**
  Receive input t_n
  for $k = 1, \ldots, K$ **do**
    Calculate $m_{k,n+1}, s^2_{k,n+1}$ from Eqs. (2) and (3)
  end for
  Calculate $\tilde{\mathbf{w}}_{n+1}$ from Eq. (5)
  Receive output y_n
  Calculate m_{n+1}, s^2_{n+1} from Eqs. (7) and (8)
  $\texttt{is\_outlier} \leftarrow y_n \in [m_{n+1} - 3s_{n+1}, m_{n+1} + 3s_{n+1}]$
  if $\neg\texttt{is\_outlier}$ **then**
    $\mathcal{D}' \leftarrow \{\}$
    Add (t_n, y_n) to $\tilde{\mathcal{D}}$
    if $t_{\text{last\_mean\_update}} \geq L$ **then**
      Update C with the average of $\tilde{\mathcal{D}}$
      $t_{\text{last\_mean\_update}} \leftarrow 0$
    else
      $t_{\text{last\_mean\_update}} \leftarrow t_{\text{last\_mean\_update}} + 1$
    end if
  else
    Add (t_n, y_n) to $\mathcal{D}'$
    if $|\mathcal{D}'| \geq N_{\text{PCB}}$ **then** ▷ Declare Changepoint
      $\mathcal{D} \leftarrow \mathcal{D}'$
      Update C with the average of $\mathcal{D}$
      $t_{\text{last\_mean\_update}} \leftarrow 0$
    end if
  end if
end for

Remark 2: The typical choice of fusion rule when using BMA-style weighting is an arithmetic average, rather than a geometric average, which comes with several desirable properties in prediction [17]. While a mixture of experts approach is not intrinsically superior to a product of experts approach for any fixed weight (see, e.g., the discussion in [13]), there are no theoretical guarantees that the BMA-style weights are optimal for geometric pooling with respect to, for example, predictive log-likelihood. Our approach will therefore adopt the mixture of experts (i.e., arithmetic fusion) approach instead.

Pseudocode for implementing INTEL is available in Algorithm 1.

IV. LINEAR-TIME GAUSSIAN PROCESS REGRESSION

In this section, we introduce the linear-time inference of GP regression, which relies on the representation of GP as a stochastic differential equation and performs inference via

Kalman filtering and smoothing [6]. These will form the basis of our proposed method. GPs that fit into this framework are often called *Markovian GPs* or *state-space GPs*. Any GP with with a stationary kernel (i.e., $\kappa(x, x')$ expressed solely in terms of the difference $r = |x - x'|$) can be approximated arbitrarily well through Markovian GPs [6]. We provide an overview of the state-space representation of Markovian GPs (Section IV-A) and their inference using Kalman filtering (Section IV-B).

A. Markovian Gaussian Processes

The $\mathcal{O}(N_{\text{data}}^3)$ complexity of inference in GP regression has been a major practical issue, leading to many approximate inference methods over the last 20 years. However, for special choices of the covariance function in scalar GPs, *exact* inference can be performed in $\mathcal{O}(N_{\text{data}})$ time by viewing GPs through the lens of stochastic differential equations (SDEs) [6].

Exact details of SDEs are beyond the scope of this work, but with several technical caveats, we may view these as ordinary differential equations driven by some noise process. The most common noise process is a *Brownian motion* $\boldsymbol{\beta}(t) \in \mathbb{R}^k$ with diffusion matrix $\mathbf{Q}$, which is defined by the following three properties [18, Def. 4.1]: (1) $\boldsymbol{\beta}(0)$ is zero, (2) increments are independent for independent time intervals, and (3) $\boldsymbol{\beta}(t_1) - \boldsymbol{\beta}(t_2) \sim \mathcal{N}(\mathbf{0}, (t_2 - t_1)\mathbf{Q})$. For further details on SDEs, the reader is directed to the excellent texts of Särkkä & Solin [18] and Øksendal [19].

In this setting, the GP prior is represented as a continuous-discrete linear Gaussian state space:

$$d\boldsymbol{\theta} = \mathbf{F}\boldsymbol{\theta} + \mathbf{L}\, d\boldsymbol{\beta},$$
$$y_n = \mathbf{h}^\top \boldsymbol{\theta}(t_n) + \mu(t_n) + \varepsilon,$$

where $\boldsymbol{\beta}$ is a Brownian motion with diffusion matrix $\mathbf{Q}$ and ε is i.i.d. Gaussian noise with variance σ_n^2. This is discretized to a discrete-time state space model [18, Ch. 10.6]:

$$\boldsymbol{\theta}(t_{n+1}) = \mathbf{A}_n \boldsymbol{\theta}(t_n) + \mathbf{q}_n, \tag{9}$$
$$y_n = \mathbf{h}^\top \boldsymbol{\theta}(t_n) + \mu(t_n) + \varepsilon_n, \tag{10}$$

where $\mathbf{A}_n$ and the covariance $\boldsymbol{\Sigma}$ of $\mathbf{q}_n$ can be derived from the SDE. In particular, $\mathbf{A}_n$ is related to the continuous transition matrix $\mathbf{F}$ and the time between successive datapoints $\Delta t_n = t_n - t_{n-1}$ by $\exp(\mathbf{F}\Delta t_n)$. The notation $\boldsymbol{\theta}(t_n)$ (as opposed to $\boldsymbol{\theta}_n$) reinforces that $\boldsymbol{\theta}(t_n)$ arises from a continuously-indexed stochastic process. Using the discrete-time state space model, a solution with linear time complexity in N_{data} can then be computed using standard Kalman filtering and smoothing. GPs with many different covariance matrices (notably, Matérn covariance matrices) can be represented this way, and even more (e.g., squared exponential kernels) can be approximated similarly — see [18, Ch. 11] for several examples.

B. Inference With Kalman Filtering

In the case of online inference, we are principally concerned with filtering solutions. The filtering equations for Markovian GPs are the standard Kalman filter ones, which track the current mean and covariance of $\boldsymbol{\theta}(t_n)$, denoted as $\mathbf{m}_n$ and $\mathbf{P}_n$, respectively. The prediction density $p(\boldsymbol{\theta}(t_n)|y_{1:n-1})$ is a normal with mean and covariance

$$\mathbf{m}_n^- = \mathbf{A}_{n-1}\mathbf{m}_{n-1}, \tag{11}$$
$$\mathbf{P}_n^- = \mathbf{A}_{n-1}\mathbf{P}_{n-1}\mathbf{A}_{n-1}^\top + \boldsymbol{\Sigma}. \tag{12}$$

The predictive distribution $p(y_n|y_{1:n-1})$ is also normal with mean and covariance

$$m_n = \mathbf{h}^\top \mathbf{m}_n^- + \mu(t_n), \tag{13}$$
$$s_n^2 = \mathbf{h}^\top \mathbf{P}_n^{-1}\mathbf{h} + \sigma_n^2. \tag{14}$$

The filtering distribution $p(\boldsymbol{\theta}(t_n)|y_{1:n})$ is then once again normal with mean and covariance

$$\mathbf{m}_n = \mathbf{m}_n^- + \mathbf{k}_n(y_n - m_n), \tag{15}$$
$$\mathbf{P}_n = \mathbf{P}_n^- - \mathbf{k}_n \mathbf{S}_n \mathbf{k}_n^\top, \tag{16}$$

where $\mathbf{k}_n$ is the *Kalman gain* given by

$$\mathbf{k}_n = \mathbf{P}_n^- \mathbf{h}/s_n^2. \tag{17}$$

V. The Lintel algorithm

The form of the Intel algorithm is particularly well-suited for linear time GPR. In fact, using linear GPs with exact filtering, we can even avoid using finite context window approximations to the predictive distribution while maintaining constant-time updates. Because of its strictly linear time complexity with respect to the length of the dataset, we call our algorithm Lintel. In this section, we introduce the Lintel algorithm (Section V-A). We then discuss how updates to the mean function should be interpreted (Section V-B), and relate the algorithm to previous work in the GP and robust filtering communities (Section V-C).

A. The Lintel Algorithm

The Lintel algorithm falls very quickly from the adoption of Markovian GPs in the Intel algorithm, and the basic idea is simple: instead of keeping track of $\mathcal{D}_n$ and recomputing GP posteriors at each time step, we instead keep track of the filtered state $\boldsymbol{\theta}(t_n)$, which provides constant-time Bayesian updates to the GP predictive. In particular, the weighting mechanism of Section III-A stays the same, as does the basic mechanics of outlier and changepoint detection. We propose only two simple changes:

1) Use the filtering equations Eqs. (15) to (17) and predictive equations Eqs. (13) and (14) to perform online GP inference. Accordingly, we use the entire (correct) predictive distribution of the current regime, $p(y_n|\mathcal{D}_n)$, at no additional cost.
2) Use arithmetic averaging as the fusion rule, rather than geometric averaging.

For (1), implementation is trivial after the corresponding state-space model is obtained. The only catch is that now we have to "backtrack" over the previous N_{PCB} points when a changepoint is declared. However, this operation remains linear in N_{PCB}, unlike the corresponding operation in the Intel algorithm, which is cubic in N_{PCB}.

Algorithm 2 LINTEL

Input: Maximum PCB Size N_{PCB}, Prior Mean C, Forgetting Factor α, Mean Update Period L, Initial Weights $\mathbf{w}_0$, Kernel Hyperparameters $\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_M$

Output: Output mean and variance m_n, s_n^2 and outlier flag $\texttt{is\_outlier}_n$ for $n = 1, 2, \ldots$

$\mathcal{D} \leftarrow \{\}$
$t_{\text{last}} = t_1$
$t_{\text{last\_mean\_update}} \leftarrow 0$
for $n = 1, 2, \ldots$ **do**
  Receive input t_n
  for $k = 1, \ldots, K$ **do**
    Set $\Delta t = t_n - t_{\text{last}}$ and calculate $\mathbf{A}_n$
    Calculate $m_{k,n+1}, s_{k,n+1}^2$ from Eqs. (13) and (14)
  end for
  Calculate $\tilde{\mathbf{w}}_{n+1}$ from Eq. (5)
  Receive output y_n
  Calculate m_{n+1}, s_{n+1}^2 from Eqs. (18) and (19)
  $\texttt{is\_outlier} \leftarrow y_n \in [m_{n+1} - 3s_{n+1}, m_{n+1} + 3s_{n+1}]$
  if $\neg\texttt{is\_outlier}$ **then**
    Calculate filtering distributions from Eqs. (15) to (17)
    $t_{\text{last}} \leftarrow t_n$
    if $t_{\text{last\_mean\_update}} \geq L$ **then**
      Update C with the average of $\mathcal{D}$
      Update $\mathbf{m}_n$ with Eq. (20)
      $t_{\text{last\_mean\_update}} \leftarrow 0$
    else
      $t_{\text{last\_mean\_update}} \leftarrow t_{\text{last\_mean\_update}} + 1$
    end if
  else
    Add (t_n, y_n) to $\mathcal{D}'$
    if $|\mathcal{D}'| \geq N_{\text{PCB}}$ **then** ▷ Declare Changepoint
      $\mathcal{D} \leftarrow \mathcal{D}'$
      $t_{\text{last}} \leftarrow t_n$
      Update C with the average of $\mathcal{D}$
      $t_{\text{last\_mean\_update}} \leftarrow 0$
      Perform Kalman filtering on $\mathcal{D}$ to update $\boldsymbol{\theta}(t_n)$
    end if
  end if
end for

For (2), we in principle obtain a predictive distribution that is a mixture of Gaussians. For simplification, we follow recent work with ensembles of GPs (e.g., [4], [5]), in recording the minimum mean square error Gaussian estimate m_{n+1} and its variance, s_{t+1}^2. They are given by

$$m_{n+1} = \sum_{k=1}^{K} \tilde{w}_{k,n+1} m_{k,n+1}, \tag{18}$$

$$s_{n+1}^2 = \sum_{k=1}^{K} \left(\sigma_{k,n+1}^2 + (m_{n+1} - m_{k,n+1})^2\right) \tilde{w}_{k,n+1}. \tag{19}$$

B. Updating the Mean Function

Care must be taken to understand how periodically updating the mean is to be interpreted. A first approach would be to directly update $\mu(t_n)$ and use Eq. (10) as the likelihood. However, unlike in INTEL, changes in the mean function now correspond to steps in a piecewise-constant mean function in the GP prior. Accordingly, changing the mean from C to C' at time t_n corresponds to a discontinuous "jump" of magnitude $\Delta C \triangleq C' - C$ in the prior mean.

However, in our case, we likely want $p(y_{n+1}|\mathcal{D}_n)$ to remain the same after updating the mean. Fortunately, this is convenient and efficient to implement by considering the augmented state space Eqs. (9) and (10). In particular, $p(y_{n+1}|\mathcal{D}_n)$ will be the same after updating the mean if the change in mean and a change in $\mathbf{h}^\top \boldsymbol{\theta}(t_n)$ cancel each other. Then, we must change the current filtering mean $\mathbf{m}_n$ accordingly.

There are an infinite number of choices when performing the corresponding update; assuming without loss of generality that $\mathbf{h}$ is a vector of zeros and ones, the simplest such update corresponds to reducing all latent function values by an average of ΔC, i.e.,

$$\mathbf{m}_n \to \mathbf{m}_n - \frac{\mathbf{h}\Delta C}{\|\mathbf{h}\|_{\ell_1}}.$$

However, it seems intuitive to instead update proportional to the current function values. For example, suppose a GP is comprised of a quasiperiodic component and a trend component. In that case, if the current trend is near zero and the quasiperiodic component is very large, the latent function values corresponding to the quasiperiodic function should bear most of the update. The corresponding update, which we use in LINTEL, is given by

$$\mathbf{m}_n \to \mathbf{m}_n - \frac{(\mathbf{h} \odot \mathbf{m}_n)\,\Delta C}{\mathbf{h}^\top \mathbf{m}_n}, \tag{20}$$

where $\mathbf{a} \odot \mathbf{b}$ is the Hadamard (elementwise) product. One may interpret this update as placing a new (informative) GP prior on the space using the updated mean function. From a filtering perspective, it is convenient to think of this update as creating a new model with a well-motivated initial distribution $p(\boldsymbol{\theta})$.

Pseudocode for implementing LINTEL is available in Algorithm 2. Initial hyperparameters may be determined in the same manner as for INTEL, i.e., by maximizing the evidence over some "pretraining" points.

C. Related Work

For outlier detection, using a credible interval of the predictive distribution is well-studied. Indeed, the 3σ-rejection Kalman filter is established as a simple way to robustify the Kalman filter [7, Ch. 7], and its applications and limitations have been studied in the applied filtering literature [20].

In fact, an outlier-rejection Kalman filter was similarly used with Markovian GPs in [21]. There, GP factor analysis is combined with Markovian GPs to provide outlier-robust GP inference, where outliers are determined online by a threshold to the log-likelihood. Their work, however, does not incorporate the possibility of regime switching, the fusion of several

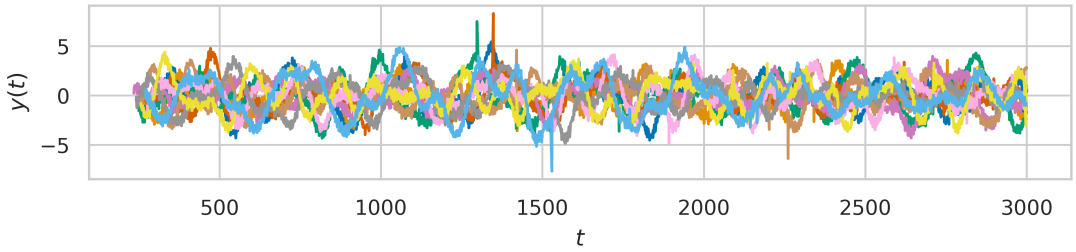

Fig. 1: Data used in the synthetic data with outliers experiment, with each color representing a different random seed.

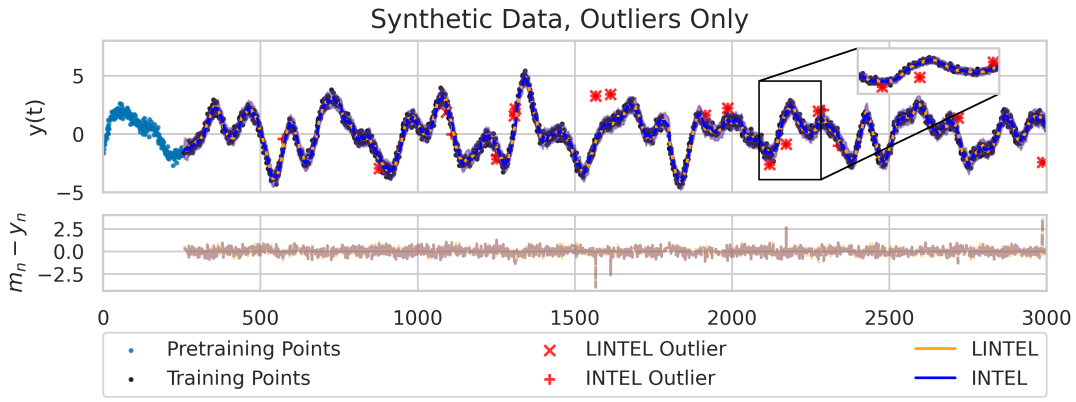

Fig. 2: An example of the output for the synthetic data with outliers experiment. **Top**: The outputs of INTEL and LINTEL, with reported outliers marked. Shaded regions denote two standard deviations. **Bottom:** The difference in predictive mean m_n and the data point y_n.

TABLE I: Results for Synthetic Data, Outliers Only Experiment. Higher is better for mean predictive-log-likelihood, and lower is better for normalized mean square error and running time. The best result (significant at the $p = 0.01$ level according to a Wilcoxon signed rank-sum test) is bolded.

	INTEL		LINTEL	
	Arithmetic	Geometric	Arithmetic	Geometric
Mean Predictive Log-Likelihood	-0.375 ± 0.049	-0.377 ± 0.049	$\mathbf{-0.365 \pm 0.038}$	-0.368 ± 0.039
Normalized Mean Square Error	0.055 ± 0.014	0.055 ± 0.014	$\mathbf{0.054 \pm 0.013}$	0.054 ± 0.013
Running Time (s)	32.39 ± 0.35	32.29 ± 0.28	$\mathbf{5.11 \pm 0.07}$	$\mathbf{5.09 \pm 0.06}$

candidate models, or adaptive means. Notably, in LINTEL, the credible region is constructed based on the fused estimate and not the individual estimates.

VI. EXPERIMENTS & DISCUSSION

To assess the performance of LINTEL, we first test INTEL and LINTEL on a synthetic dataset where only outliers are present (Section VI-A). This is followed by a similar example where regime switching occurs (Section VI-B). Finally, we experiment on a real-world datasets (Section VI-C). All code to reproduce experiments is available at https://www.github.com/DanWaxman/Lintel.

A. Synethetic Data With Outliers

We first experiment with synthetic datasets to verify two of our main assertions: (1) that the use of linear time GPs can increase performance and (2) that the use of arithmetic averaging can increase predictive performance, in terms of predictive likelihoods and squared error of the mean. The first experiment is with outliers only, where $N_{\text{data}} = 3000$ data points are generated as follows: first, values of t are sampled from a uniform distribution $\mathcal{U}(0, 3000)$ and sorted. Next, y values are sampled from a GP with a rather expressive kernel. Our focus is that the resulting functions are complex rather than the specific choice of kernel, but for completeness, a mixture Hida-Matérn-3/2 kernel of order 3 [22] is used. All y values are then given white Gaussian observation noise with variance 0.3^2, except for 10 random "true outliers," where the variance of the observation noise was instead $2.0^2 + 0.3^2$. We use 10 realizations of the GP corresponding to different random seeds, pictured in Fig. 1.

The synthetic data provide a convenient ground truth for comparison. Namely, we measure the mean predictive log-likelihood

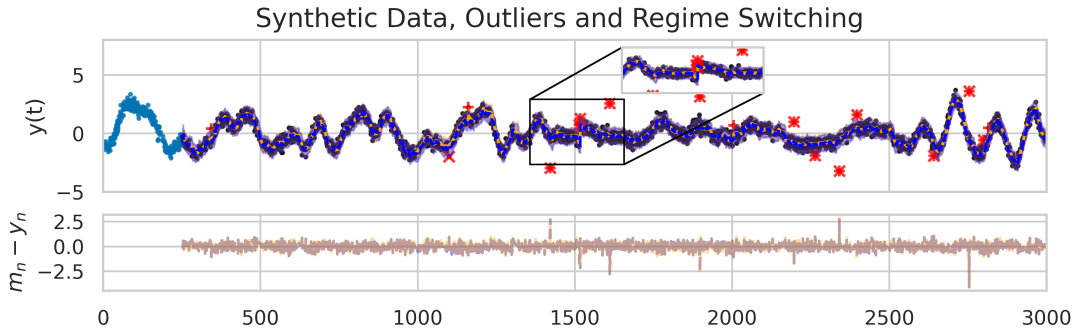

Fig. 3: An example of the output for the synthetic data with outliers and regime switching experiment. **Top**: The outputs of INTEL and LINTEL, with reported outliers marked. Shaded regions denote two standard deviations. **Bottom:** The difference in predictive mean m_n and the data point y_n. The legend is the same as Fig. 2 and is therefore omitted.

MPLL $= \frac{1}{N_{\text{data}}} \sum_n p(y_n | m_n, s_n^2)$, and the normalized mean square error nMSE $= \frac{1}{N_{\text{data}}} \sum_n (m_n - y_n)^2 / s_y^2$, where s_y^2 is the variance of y. Since the outliers are known, we exclude them in these calculations. We use two kernels for both INTEL and LINTEL: one is a simple Matérn-5/2 kernel, which provides reasonable but suboptimal performance alone, and the other is an evidence-maximized kernel from the mixture of Hida-Matérn-3/2 family. Once again, our focus is not on these specific choices, but rather, that the "true" kernel is available alongside a "reasonable," but incorrect, kernel. Evidence maximization is performed on the first 250 samples, and the rest are used for online inference. An example of the resulting predictions is pictured in Fig. 2. In this experiment, we set L to be larger than the length of the time series (i.e., the mean will never periodically update), choose a maximum PCB size of $N_{\text{PCB}} = 3$, and set $\tau = 20$.

The results of this experiment can be found in Table I, and support both of our claims — namely, that arithmetic fusion and the linear GP formulation outperform INTEL, and that LINTEL is dramatically faster than INTEL.

B. Synthetic Data With Outliers and Regime Switches

We next experiment with a synthetic dataset that contains both outliers and regime switches. The dataset is quite similar to that of Section VI-A, except $y_{1500}, \ldots, y_{2000}$ are drawn from a GP with a simple Matérn-5/2 kernel, rather than the more complex Hida-Matérn kernel. Since a regime switch is expected, we set $L = 250$ in this experiment.

An example plot of results for one realization is pictured in Fig. 3. The numerical results are extremely similar to the experiment of the previous section, so we do not report them here — the key takeaways are the same: LINTEL achieves quantifiably higher MPLL and lower nMSE (significant at the $p = 0.01$ level according to a Wilcoxon signed rank-sum test) in approximately $1/7$th the time.

We make the following remark regarding the performance of both INTEL and LINTEL:

Remark 1: In this experiment, regime switches are only occasionally detected, as the dynamic weighting mechanism often adjusts before three consecutive outliers are observed. This may be desirable if predictive performance is the paramount metric, but undesirable if the identity of change points is of importance.

C. CPU Utilization Dataset

We additionally test INTEL and LINTEL on the CPU utilization dataset, mirroring the analysis in the INTEL paper. The CPU utilization dataset, found in the Numenta Anomaly Benchmark (NAB) dataset[1] [23], is a real-world dataset with a simple structure but several regime switches. For this dataset, we use candidate models similar to those in the INTEL paper. In particular, we use a set of eight Matérn-3/2 kernels, where the process variance and variance of the observation noise are set to twice or half the value of the evidence-maximized kernel. Here, we set $L = 50$ and $N_{\text{PCB}} = 3$, and use arithmetic averaging for LINTEL and geometric averaging for INTEL.

The results can be found in Fig. 4. Overall, the results are extremely similar, with the notable exception of speed: LINTEL takes **27 seconds** while INTEL takes **195 seconds**. This speedup is consistent with that of the synthetic data experiments and shows significant promise in higher-throughput online inference.

Of particular note is that the change points detected by LINTEL and INTEL are the same, and that predictions are largely the same. This is intuitive given the nature of the function: such low-variance functions do not suffer harshly from finite context windowing.

VII. CONCLUSION & FUTURE DIRECTIONS

In this paper, we introduced LINTEL, a modified version of INTEL with constant-time updates using Markovian GPs. We showed how we can increase computational efficiency while also circumventing approximations to the predictive distributions. The state-space formulation of LINTEL in terms of Markovian GPs also provides connections to the robust filtering literature.

We evaluated the proposed method on two synthetic datasets, showing superior performance when the ground truth is known,

[1]Specifically, we use `ec2_cpu_utilization_ac20cd`

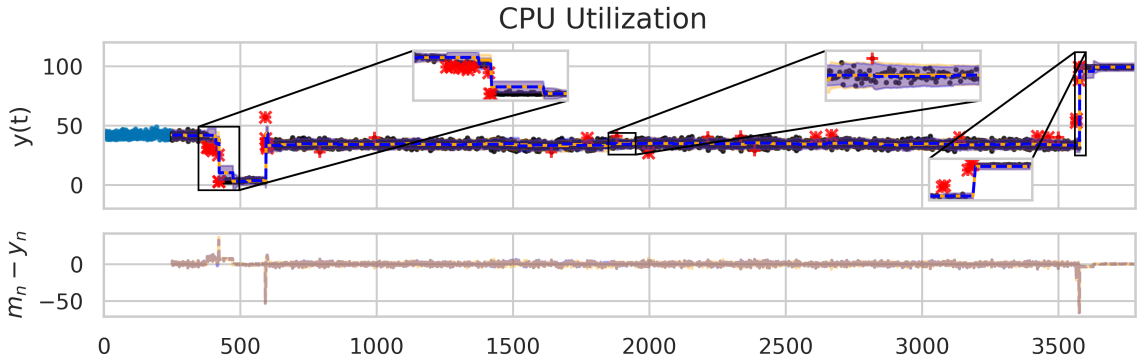

Fig. 4: Output of INTEL and LINTEL on the CPU utilization dataset. **Top**: The outputs of INTEL and LINTEL, with reported outliers marked. Shaded regions denote two standard deviations. **Bottom:** The difference in predictive mean m_n and the data point y_n. The legend is the same as Fig. 2 and is therefore omitted.

and on a real-world dataset, showing nearly identical prediction at 7× higher throughput. Future work may consider further benchmarking, for example, on the entire NAB dataset.

During our experiments, we occasionally noted the numerical "collapse" of the weights vector $\mathbf{w}$, in the sense that a weight would numerically underflow to zero, in which case it cannot be revived, even when working with 64-bit floating point numbers. As a fix in the code, we simply add a small perturbation to the weight matrix at each time step. However, this problem was also noticed and considered in similar online ensembling problems [5], and future research may incorporate the solutions proposed there.

In this work, we showed that using BMA-derived weights was more conducive to arithmetic averaging than geometric averaging when evaluating the predictive log-likelihood. However, instead of abandoning geometric averaging, it is also interesting to consider deriving weights as an online optimization problem. While naïvely one would consider weights that sum to unity, it has recently been noted in the literature that the generalized product of experts for GPs may benefit from allowing $\mathbf{w}$ to not sum to unity, as this allows finer control of the predictive variance [16]. It is then interesting to consider a version of LINTEL where the product of experts fusion rule is maintained.

REFERENCES

[1] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1058–1067, 2009.

[2] T. Guo, Z. Xu, X. Yao, H. Chen, K. Aberer, and K. Funaya, "Robust online time series prediction with recurrent neural networks," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2016, pp. 816–825.

[3] B. Liu, Y. Qi, and K.-J. Chen, "Sequential online prediction in the presence of outliers and change points: an instant temporal structure learning approach," *Neurocomputing*, vol. 413, pp. 240–258, 2020.

[4] Q. Lu, G. V. Karanikolas, and G. B. Giannakis, "Incremental ensemble Gaussian processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2, pp. 1876–1893, 2022.

[5] D. Waxman and P. M. Djurić, "Dynamic online ensembles of basis expansions," *Transactions on Machine Learning Research*, 2024. [Online]. Available: https://openreview.net/forum?id=aVOzWH1Nc5

[6] S. Särkkä, A. Solin, and J. Hartikainen, "Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at Gaussian process regression through Kalman filtering," *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 51–61, 2013.

[7] A. Zoubir, V. Koivunen, E. Ollila, and M. Muma, *Robust Statistics for Signal Processing*. Cambridge University Press, 2018.

[8] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005.

[9] A. Ranganathan, M.-H. Yang, and J. Ho, "Online sparse Gaussian process regression and its applications," *IEEE Transactions on Image Processing*, vol. 20, no. 2, pp. 391–404, 2010.

[10] M. F. Huber, "Recursive Gaussian process: On-line regression and learning," *Pattern Recognition Letters*, vol. 45, pp. 85–91, 2014.

[11] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[12] Y. Cao and D. J. Fleet, "Generalized product of experts for automatic and principled fusion of Gaussian process predictions," *arXiv preprint arXiv:1410.7827*, 2014.

[13] T. Li, H. Fan, J. García, and J. M. Corchado, "Second-order statistics analysis and comparison between arithmetic and geometric average fusion: Application to multi-sensor target tracking," *Information Fusion*, vol. 51, pp. 233–243, 2019.

[14] M. B. Hurley, "An information theoretic justification for covariance intersection and its generalization," in *Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002.*, vol. 1. IEEE, 2002, pp. 505–511.

[15] L. M. Carvalho, D. A. Villela, F. C. Coelho, and L. S. Bastos, "Bayesian inference for the weights in logarithmic pooling," *Bayesian Analysis*, vol. 18, no. 1, pp. 223–251, 2023.

[16] M. Ajirak, D. Waxman, F. Llorente, and P. M. Djurić, "Fusion of Gaussian process predictions with Monte Carlo," in *2023 Asilomar Conference on Signals, Systems, and Computers*, 2023, pp. 1367–1371.

[17] B. Liu, "Robust sequential online prediction with dynamic ensemble of multiple models: A review," *Neurocomputing*, p. 126553, 2023.

[18] S. Särkkä and A. Solin, *Applied Stochastic Differential Equations*. Cambridge University Press, 2019, vol. 10.

[19] B. Øksendal, *Stochastic Differential Equations: An Introduction with Applications*, ser. Universitext. Springer Berlin Heidelberg, 2013.

[20] Z. Berman, "Outliers rejection in Kalman filtering — some new observations," in *2014 IEEE/ION Position, Location and Navigation Symposium-PLANS 2014*. IEEE, 2014, pp. 1008–1013.

[21] C. Bock, F.-X. Aubet, J. Gasthaus, A. Kan, M. Chen, and L. Callot, "Online time series anomaly detection with state space Gaussian processes," *arXiv preprint arXiv:2201.06763*, 2022.

[22] M. Dowling, P. Sokół, and I. M. Park, "Hida-Matérn kernel," *arXiv preprint arXiv:2107.07098*, 2021.

[23] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.