Large-scale Retrieval of Bayesian Machine Learning Models for Time Series Data via Gaussian Processes

Fabian Berns[1] and Christian Beecks[1,2]
[1]*Department of Computer Science, University of Münster, Germany*
[2]*Fraunhofer Institute for Applied Information Technology FIT, Sankt Augustin, Germany*

Keywords: Bayesian Machine Learning, Gaussian Process, Data Modeling, Knowledge Discovery.

Abstract: Gaussian Process Models (GPMs) are widely regarded as a prominent tool for learning statistical data models that enable timeseries interpolation, regression, and classification. These models are frequently instantiated by a Gaussian Process with a zero-mean function and a radial basis covariance function. While these default instantiations yield acceptable analytical quality in terms of model accuracy, GPM retrieval algorithms automatically search for an application-specific model fitting a particular dataset. State-of-the-art methods for automatic retrieval of GPMs are searching the space of possible models in a rather intricate way and thus result in super-quadratic computation time complexity for model selection and evaluation. Since these properties only enable processing small datasets with low statistical versatility, we propose the Timeseries Automatic GPM Retrieval (TAGR) algorithm for efficient retrieval of large-scale GPMs. The resulting model is composed of independent statistical representations for non-overlapping segments of the given data and reduces computation time by orders of magnitude. Our performance analysis indicates that our proposal is able to outperform state-of-the-art algorithms for automatic GPM retrieval with respect to the qualities of efficiency, scalability, and accuracy.

1 INTRODUCTION

Applying Gaussian Process Models (GPMs) for interpolation (Roberts et al., 2013; Li and Marlin, 2016), regression (Titsias, 2009; Duvenaud et al., 2013), and classification (Li and Marlin, 2016; Hensman et al., 2013) of timeseries necessitates to instantiate the underlying Gaussian Process by a covariance function and a mean function. While the latter is typically instantiated by a constant, zero-mean function, the covariance function is modelled either by (i) a *general-purpose kernel* (Wilson and Adams, 2013), (ii) a *domain-specific kernel* that is individually tailored to a specific application by a domain expert (Wilson and Adams, 2013; Abrahamsen and Petter, 1997; Rasmussen and Williams, 2006), or (iii) a *composite kernel* that is computed automatically by means of a GPM retrieval process (Duvenaud et al., 2013; Lloyd et al., 2014) in order to decompose the statistical characteristics underlying the data into multiple submodels. Although the first option (i) produces GPMs of sufficiently high model accuracy, both (ii) and (iii) encapsulate data's peculiarities and versatilities in a more detailed manner. Making use of domain-specific kernels requires laborious fine-tuning and extensive expert knowledge. Employing an automatic, domain-agnostic GPM retrieval process on the other hand produces expressive composite kernels without requiring any human interference.

State-of-the-art automatic GPM retrieval algorithms, such as Compositional Kernel Search (CKS) (Duvenaud et al., 2013), Automatic Bayesian Covariance Discovery (ABCD) (Lloyd et al., 2014; Steinruecken et al., 2019) and Scalable Kernel Composition (SKC) (Kim and Teh, 2018), apply an open-ended greedy search in the space of all feasible composite covariance functions in order to determine an optimal, e.g. in terms of maximum likelihood, instantiation of the underlying Gaussian Process. The resulting inextricable GPM needs to account for all different kinds of statistical peculiarities of the underlying data and thus lacks expressiveness especially with regards to local phenomena. Additionally, evaluating a GPM by usual measures such as the likelihood (Malkomes et al., 2016) function entails calculations of cubic computation time complexity, which limits

Berns, F. and Beecks, C.
Large-scale Retrieval of Bayesian Machine Learning Models for Time Series Data via Gaussian Processes.
DOI: 10.5220/0010109700710080
In *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2020) - Volume 1: KDIR*, pages 71-80
ISBN: 978-989-758-474-9

the application of GPM retrieval algorithms to small-to-moderate data collections (Kim and Teh, 2018).

In this paper, we aim to overcome the performance limitations of state-of-the-art GPM retrieval algorithms and propose the Timeseries Automatic GPM Retrieval (TAGR) algorithm for large-scale statistical data modeling based on Gaussian Processes. We aim to limit the algorithm's overall complexity by concurrently initiating multiple kernel searches on dynamically *partitioned* data (cf. Berns and Beecks, 2020b,a; Berns et al., 2019). In this way, TAGR describes large-scale datasets by means of a concatenation of independent sub-models. Our approach is able to outperform existing kernel search methods in terms of expressiveness, while reducing execution time by a factor of up to 500.

The paper is structured as follows. Section 2 presents related work and background information. Section 3 introduces large-scale GPMs, while Section 4 proposes the corresponding TAGR algorithm. Its performance and explanatory capacity are evaluated in Section 5. We conclude our paper with an outlook on future work in Section 6.

2 BACKGROUND AND RELATED WORK

2.1 Gaussian Process

A Gaussian Process (Rasmussen and Williams, 2006) is a stochastic process over random variables $\{f(x) \mid x \in \mathcal{X}\}$, indexed by a set $\mathcal{X}$, where every subset of random variables follows a multivariate normal distribution. The distribution of a Gaussian Process is the joint distribution of all of these random variables and it is thus a probability distribution over the space of functions $\{f \mid f : \mathcal{X} \to \mathbb{R}\}$. A Gaussian Process is formalized as follows:

$$f(\cdot) \sim \mathcal{GP}\big(m(\cdot), k(\cdot,\cdot)\big) \tag{1}$$

where the mean function $m : \mathcal{X} \to \mathbb{R}$ and the covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ are defined $\forall x \in \mathcal{X}$ as follows:

$$m(x) = \mathbb{E}[f(x)] \tag{2}$$

$$k(x,x') = \mathbb{E}\left[\big(f(x) - m(x)\big) \cdot \big(f(x') - m(x')\big)\right] \tag{3}$$

Given a finite dataset $D = \{X,Y\}$ with $X = \{x_i \mid x_i \in \mathcal{X} \wedge 1 \leq i \leq n\}$ representing the underlying input data (such as timestamps) and $Y = \{f(x_i) \mid x_i \in \mathcal{X}\}$ representing the target data values, such as sensor values or other complex measurements, the mean and covariance functions are determined by maximizing the log marginal likelihood (Rasmussen and Williams, 2006) of the Gaussian Process, which is defined as follows:

$$\mathcal{L}(m,k,\theta \mid D) = -\frac{1}{2} \cdot \big[(y-\mu)^T K^{-1}(y-\mu) + \log\det(K) + n\log(2\pi)\big] \tag{4}$$

As can be seen in Equation 4, the marginalization of a Gaussian Process for a given dataset D of n records results in a finite data vector $y \in \mathbb{R}^n$, a mean vector $\mu \in \mathbb{R}^n$, and a covariance matrix $K \in \mathbb{R}^{n \times n}$ which are defined as $y[i] = f(x_i)$, $\mu[i] = m(x_i)$, and $K[i,j] = k(x_i,x_j)$ for $1 \leq i,j \leq n$, respectively. We use the short-hand notation $\mathcal{GP}_\theta$ to denote a GPM whose underlying mean function m and covariance function k are parameterized via hyperparameters θ (cf. Cheng and Boots, 2017; Csató and Opper, 2000).

While the covariance function is frequently used as major data modeling entity it often lacks in describing individual statistical behaviors in a *structured* way. To this end, Duvenaud et al. (2013) propose to approximate and structure the covariance function via multiple compositional kernel expressions in order to obtain the resulting compositional kernel-based covariance function. The corresponding algorithms for automatically retrieving a GPM given an arbitrary dataset D are summarized in the following section.

2.2 Retrieval Algorithms

GPM retrieval algorithms, such as CKS (Duvenaud et al., 2013), ABCD (Lloyd et al., 2014; Steinruecken et al., 2019), and SKC (Kim and Teh, 2018), aim to discover the statistical structure of a dataset D by determining an optimal covariance function k[1]. For this purpose, the mean function of the Gaussian Process is commonly instantiated by a constant zero function (Duvenaud et al., 2013; Rasmussen and Williams, 2006), so as to correspond to an additional data normalization step. The covariance function is algorithmically composed via operators implementing addition and multiplication among different (composed) base kernels $b \in \mathcal{B}$. Prominent base kernels include the linear kernel, radial basis function kernel, and periodic kernel, which are able to capture for instance smooth, jagged, and periodic behavior (Duvenaud et al., 2013).

The algorithms mentioned above apply an open-ended, greedy search in the space of all feasible kernel combinations in order to progressively compute a GPM fitting the entire dataset D, respectively $Y \in D$.

[1] In (Berns and Beecks, 2020a), we detail the problem and definition of automatic GPM retrieval.

The CKS algorithm (Duvenaud et al., 2013) follows a simple iterative procedure, called *Basic Kernel Expansion Strategy* (BES), which aims to improve the best kernel expressions k retrieved in one iteration over the course of the next one. Therefore, this strategy produces a set of candidate kernel expressions $\mathcal{C}_k$ based on k (Duvenaud et al., 2013):

$$\mathcal{C}_k = \{k \oslash b \mid b \in \mathcal{B} \wedge \oslash \in \{+, \times\}\} \tag{5}$$

Starting with a set of base kernels $\mathcal{B}$, the candidates of every following iteration are generated via expanding upon the best candidate of the previous iteration. Moreover, a new candidate is also generated for every *replacement* of a base kernel b with another one b'. ABCD extends that grammar with regards to a sigmoid change point operator $|$, in order to locally restrict the effect of kernel expressions (Lloyd et al., 2014; Steinruecken et al., 2019):

$$\mathcal{C}_k = \{k \oslash b \mid b \in \mathcal{B} \wedge \oslash \in \{+, \times, |\}\} \tag{6}$$

The performance of those given automatic GPM retrieval algorithms is impeded by two major bottlenecks. Evaluating *every* expansion of *every* subexpression poses one of these bottlenecks. Thus, reducing the set of candidates per iteration in order to cover just the most promising candidates improves on the performance of the respective algorithms, presumably without affecting model quality. Moreover, assessing model quality in terms of log marginal likelihood epitomizes another bottleneck of current algorithms, which is intrinsic to the framework of Gaussian Processes itself (Hensman et al., 2013). The cubic runtime complexity of that basic measure inhibits analysis of large-scale datasets (cf. Kim and Teh, 2018).

Kim and Teh (2018) present SKC as a way to overcome at least the latter one of those bottlenecks. The algorithm improves the efficiency of the kernel search process by accelerating CKS using the Nyström method (cf. Williams and Seeger, 2000) for global approximations (Kim and Teh, 2018). This low-rank matrix approximations constructs an approximate covariance matrix by means of a small subset of the given data (cf. Gittens and Mahoney, 2016) and allows to retrieve GPMs for datasets of up to 100,000 records in a reasonable amount of time according to Kim and Teh (2018).

To summarize, state-of-the-art GPM retrieval algorithms are not suited for the analysis of large-scale datasets beyond 100,000 records due to their cubic computation time complexity as well as due to their large candidate sets, which need to be evaluated during the search procedure (cf. Kim and Teh, 2018).

3 LARGE-SCALE GAUSSIAN PROCESS MODELS

In this section, we propose the structural design of large-scale GPMs. It is designed to reduce computational complexity of GPM evaluations as well as to mitigate the amount of kernel expression candidates, which need to be evaluated as part of the respective retrieval algorithm. Since these two issues are the main bottlenecks of current GPM retrieval algorithms, respective solution strategies are separately covered in the following two subsections.

3.1 Reduction of Computational Complexity

Liu et al. (2020) as well as Rivera and Burnaev (2017) highlight *local approximations* as a key possibility to reduce complexity of common GPM evaluations based on likelihood measures. Instead of infering an approximate GPM based on a data subset (cf. *global* approximations), they construct a holistic GPM from locally-specialized models trained on non-overlapping segments of the data. Thus, the covariance matrix of the holistic GPM is composed of the respective matrices of its local sub-models (Liu et al., 2020). This divide-&-conquer approach accelerates calculation of log marginal likelihood, since the resulting covariance matrix is a block diagonal matrix (Rivera and Burnaev, 2017), whose inverse and determinant can be computed efficiently (Park and Apley, 2018). Rivera and Burnaev (2017) emphasize that *local approximations* allow to "model rapidly-varying functions with small correlations" in contrast to low-rank matrix approximations.

Embedding the concept of *local approximations* into GPM retrieval algorithms requires a globally partitioning covariance function. While in principle change point operators facilitate a global data partitioning, their nature of fading kernel expressions into one another (Lloyd et al., 2014; Steinruecken et al., 2019) does neither produce clear boundaries between sub-models nor enables independence of these models. Therefore, we adapt the given notion of a change point operator to utilize indicator functions (Iliev et al., 2017) instead of sigmoid functions (cf. Steinruecken et al., 2019) to separate kernel expressions. This leads us to the definition of a partitioning covariance function $\mathcal{K}$ for large-scale GPMs $\mathcal{GP}_\theta(0, \mathcal{K})$:

$$\mathcal{K}(x, x' \mid \{k_b\}_{b=1}^m, \{\tau_b\}_{b=0}^m) = \sum_{b=1}^{m} k_b(x, x') \cdot \mathbb{1}_{\{\tau_{b-1} < x \leq \tau_b\}}(x) \cdot \mathbb{1}_{\{\tau_{b-1} < x' \leq \tau_b\}}(x') \tag{7}$$

The parameter $m \in \mathbb{N}$ defines the number of sub-models $k_b : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, where each sub-model k_b can be thought of as a local covariance function modeling the restricted domain $[\tau_{b-1}, \tau_b] \subseteq \mathcal{X}$. In this way, each sub-model k_b is only responsible for a certain coherent fraction of the data, which is delimited by the change points τ_{b-1} and τ_b. The specific change points $\tau_0 = x_1$ and $\tau_m = x_n$ denote the start and end of the data set D. The usage of indicator functions (thus having disjoint data segments) produces a block diagonal covariance matrix (cf. Low et al., 2015), which facilitates independently searching the most likely local kernel expression (i.e. sub-model) per data segment.

In order to mathematically show the implication of aforementioned independence, we have to investigate the internal structure of the corresponding log marginal likelihood function $\mathcal{L}(m,k,\theta \mid D)$. To this end, we assume the inherent covariance matrix $K \in \mathbb{R}^{n \times n}$ (cf. Equation 4) to be a block diagonal matrix. That is, it holds that $K = diag(K_1, ..., K_m)$, where each matrix $K_b \in \mathbb{R}^{n_b \times n_b}$ for $1 \leq b \leq m$ is of individual arbitrary size $n_b \in \mathbb{N}$ such that $\sum_b n_b = n$. Furthermore, let $(I_1, ..., I_m)$ denote the partitioning of index set $\{1, ..., n\} \subseteq \mathbb{N}$ according to the given blocks, i.e. for a submatrix of K over index set I_b the following holds: $K_{I_b,I_b} = K_b \forall b \in [1,m]$. Given this notation, one can show (i) that the quadratic form $(y-\mu)^T K^{-1}(y-\mu)$ of the log marginal likelihood function $\mathcal{L}(m,k,\theta \mid D)$ is equivalent to $\sum_{b=1}^{m} (\tilde{y}_b - \tilde{\mu}_b)^T K_b^{-1} (\tilde{y}_b - \tilde{\mu}_b)$, where $\tilde{y}_b = (y_i)_{i \in I_b} \in \mathbb{R}^{n_b}$ and $\tilde{\mu}_b = (\mu_i)_{i \in I_b} \in \mathbb{R}^{n_b}$ are defined according to the block diagonal structure and (ii) that the logarithmic determinant can be decomposed as follows: $\log\det(K) = \log\prod_{b=1}^{m} \det(K_b) = \sum_{b=1}^{m} \log\det(K_b)$. By making use of the aforementioned algebraic simplifications, we can simplify the calculation of the log marginal likelihood function as shown below:

$$
\begin{aligned}
&\mathcal{L}(m,k,\theta \mid D) \\
&= -\frac{1}{2} \cdot \left[(y-\mu)^T K^{-1} (y-\mu) + \log\det(K) + n\log(2\pi) \right] \\
&= -\frac{1}{2} \cdot \left[\sum_{b=1}^{m} (\tilde{y}_b - \tilde{\mu}_b)^T K_b^{-1} (\tilde{y}_b - \tilde{\mu}_b) \right. \\
&\qquad \left. + \sum_{b=1}^{n} \log\det(K_b) + \sum_{b=1}^{m} n_b \cdot \log(2\pi) \right] \\
&= \sum_{b=1}^{m} -\frac{1}{2} \cdot \left[(\tilde{y}_b - \tilde{\mu}_b)^T K_b^{-1} (\tilde{y}_b - \tilde{\mu}_b) \right. \\
&\qquad \left. + \log\det(K_b) + n_b \log(2\pi) \right] \\
&= \sum_{b=1}^{m} \mathcal{L}(m,k_b,\theta \mid D_b)
\end{aligned}
\tag{8}
$$

As shown above, the equivalence $\mathcal{L}(m,k,\theta \mid D) = \sum_{b=1}^{m} \mathcal{L}(m,k_b,\theta \mid D_b)$ affects the computation of a large-scale GPM by means of $\mathcal{K}$ for a given data set D and its non-overlapping partitions D_b. Instead of jointly optimizing a holistic statistical model fitting the entire data set D, we are able to independently optimize individual sub-models k_b. This will not only increase the expressiveness of the overall GPM, but also reduce the computation time required for calculating such a model. Moreover, by splitting a GPM into smaller parts, we enable a large-scale GPM to be computed by means of the divide-&-conquer paradigm. Before we investigate the proposed algorithmic solution, we continue with describing strategies for efficiently reducing the size of candidate sets for sub-models k_b.

3.2 Expansion Strategies

In contrast to the previous section, where we have shown how to reduce the complexity of evaluating a single GPM via a divide-&-conquer-based approach, we investigate measures to reduce the amount of to-be-evaluated candidates per sub-model covariance function k_b in the remainder of this section. Equation 5 and 6 illustrate the state-of-the-art strategy used by CKS (Duvenaud et al., 2013), SKC (Kim and Teh, 2018) and ABCD (Lloyd et al., 2014). While this strategy allows to exhaustively search the space of possible kernel expressions for the most appropriate one, it entails to evaluate a lot of inferior candidates as a by-product. Since the sub-models k_b of $\mathcal{K}$ (cf. Equation 7) are designed to represent independent and inseparable behavioral patterns, introducing further change points on that layer of the covariance function would contradict the atomicity assumption of every sub-model k_b. Furthermore, by focusing on composite covariance functions which are only derived by additions and multiplications of base kernels, the number of candidate models can be further reduced.

Although state-of-the-art algorithms consider every possible kernel expansion regardless of their structure up to a certain depth (cf. Subsection 2.2), they *retroactively* enforce a hierarchy, i.e., the so-called Sum-of-Products (SoP) hierarchy (Duvenaud et al., 2013) (cf. Equation 9), among additive operators and multiplicative operators. Thus, these algorithms structure the resulting kernel expressions in order to foster comprehensibility *in hindsight*. We incorporate that SoP hierarchy into our kernel expansion strategy to prohibit functionally redundant composite kernel expressions, which result from different expressions producing the same SoP form, and define

the set of SoP kernel expressions $\mathcal{S}$ as follows:

$$\mathcal{S} = \left\{ \sum_{i=1}^{s} \prod_{j=1}^{p} b_{ij} \,|\, b_{ij} \in \mathcal{B} \right\} \tag{9}$$

Based on the mathematically formalization of the SoP form given in the equation above, we propose the new SoP Kernel Expansion Strategy (SES) by means of the following candidate set:

$$\mathcal{C}_k = \{k \oslash b \,|\, b \in \mathcal{B} \wedge \oslash \in \{+, \times\} \wedge (k \oslash b) \in \mathcal{S}\} \tag{10}$$

This candidate set ensures, that the resulting kernel expression complies with the SoP form. As with state-of-the-art algorithms, base kernels are considered as initial candidates of this strategy.

Orthogonal to the described improvement on kernel expansion strategies, we assess the tactic of base kernel *replacement* as ineffectual from a theoretical perspective, as it contradicts the *greedy* search (Resende and Ribeiro, 2016) propagated by authors of state-of-the-art methods (cf. Steinruecken et al., 2019). Therefore, we will separately analyze the effects of this *replacement tactic* on GPM retrieval in Section 5.

To sum up, our structural design for large-scale GPMs encompasses two solutions for the major bottlenecks of GPM structures used by current state-of-the-art algorithms. On the one hand, local approximations by means of an indicator change point operator reduce computational complexity by partitioning a GPM into several locally-specialized sub-models. On the other hand, reducing the amount of candidate covariance functions per sub-model k_b by employing SES mitigates evaluative calculations for inferior models.

4 AUTOMATIC GPM RETRIEVAL

In this section, we propose the Timeseries Automatic GPM Retrieval (TAGR) algorithm for efficient retrieval of large-scale GPMs. To this end, we make use of the efficiency improvements, i.e. locally approximated covariance function and SoP Kernel Expansion Strategy (SES), described in the previous section.

In order to limit the algorithm's overall complexity and to increase model locality, i.e. to capture and describe local evolving patterns underlying the data, the TAGR algorithm initiates multiple local kernel searches on dynamically partitioned data. Subsequently, TAGR concatenates the resulting sub-models, i.e. each independent composite kernel-based covariance function k_b, into a joint large-scale GPM by means of $\mathcal{K}$ (cf. Equation 7) and a zero-mean function.

Prior to the computation of these independent sub-models, a global partitioning needs to be determined. As the field of change point detection mechanisms for partitioning a dataset via their target values $y \in Y$ (Truong et al., 2020; Aminikhanghahi and Cook, 2017) is a well researched domain, we assume the set of change points $\mathcal{T}$ to be one of the parameters of TAGR. Besides $\mathcal{T}$, the given dataset $D = \{X, Y\}$ (cf. Subsection 2.1) and complexity parameter c_{max}, restricting sub-model size, are given as parameters.

Algorithm 1 illustrates the TAGR algorithm. Without loss of generality we initiate the sub-models $k_b \in K$ for every data partition D_b bounded by change points τ_{b-1} and τ_b via a white noise kernel k_{WN}. The set I encompasses the indices of all non-final sub-models $k_b \in K$. The following while-loop is executed until this set of indices is empty. During every iteration the segment K_i with the lowest model quality by means of log marginal likelihood $\mathcal{L}$ is selected, candidate kernel expressions $\mathcal{C}_{K_i}$ for that segment are found using the predefined kernel expansion strategy and the best performing kernel expression $k^\star$ is determined. If that kernel expression $k^\star$ reaches maximum complexity c_{max} in terms of number of involved base kernels or is equal to the previous best expression K_i, it is considered final and thus removed from I. Finally, a GPM utilizing a $\mathcal{K}$-based concatenation of sub-models $k_b \in K$ and zero-mean is returned.

Algorithm 1: Timeseries Automatic GPM Retrieval.

function (D, c_{max}, $\mathcal{T}$)
  $K = \{k_{WN}\}_{i=1}^{|\mathcal{T}|}$
  $I = \{i\}_{i=1}^{|\mathcal{T}|}$
  while $|I| > 0$ **do**
    $i = \arg\min_{i \in I} \mathcal{L}(0, K_i, \theta \,|\, D_i)$
    $k^\star = \arg\max_{k \in \mathcal{C}_{K_i}} \mathcal{L}(0, k, \theta \,|\, D_i)$
    if $k^\star = K_i \vee c(k^\star) \geq c_{max}$ **then**
      $I = I \setminus i$
    $K_i = k^\star$
  return $\mathcal{GP}_\theta(0, \mathcal{K}(\cdot, \cdot | K, \mathcal{T}))$

Since the given algorithm is designed to enable large-scale applications of GPM retrieval, it may be accelerated by optimizing multiple segments in parallel. The practical capabilities of the TAGR algorithm, both in an unparallelized as well as in a parallelized manner, are evaluated in the following chapter. Although we do not explicitly name that opportunity in Algorithm 1, a preliminary GPM $\mathcal{K}$ may be returned after every iteration of the while-loop making the algorithm anytime capable.

Table 1: Used Benchmark Datasets.

	Dataset Name	Size	Dim.
1	Airline	144	2
2	Solar Irradiance	391	2
3	Mauna Loa	702	2
4	SML (Zamora-Martínez et al., 2014)	4,137	24[2]
5	Power Plant (Tüfekci, 2014)	9,568	5[3]
6	GEFCOM (Hong et al., 2014)	38,064	21[4]
7	Jena Weather (Max Planck Institute for Biogeochemistry, 2019)	420,551	15[5]
8	Household Energy (Hebrail and Berard, 2012)	2,075,259	9[6]

5 EVALUATION

In this section, we investigate the performance of the proposed TAGR algorithm in terms of efficiency and accuracy. At first we are solely evaluating TAGR for different configurations of complexity parameter c_{max} and different kernel expansion strategies (cf. Section 3.2) to investigate, how they affect model quality and runtime (cf. Section 5.2). Based on the empirically proven best parameter configuration, we compare the TAGR algorithm against the state-of-the-art algorithms CKS, SKC and ABCD (cf. Section 5.3). They serve as a baseline to assess efficiency and accuracy of our proposed algorithm. Finally, TAGR's scalability and runtime performance on large-scale datasets is demonstrated in Section 5.4, where we omit the evaluation of ABCD, SKC and CKS due to their computation time complexity (Rasmussen and Williams, 2006), which impedes analyzing larger datasets (Kim and Teh, 2018) (cf. Table 2).

5.1 Setup of Experiments

In order to ensure comparability with existing and prospective algorithms, we base our experiments on publicly available datasets, which were also used by Duvenaud et al. (2013) and Lloyd et al. (2014) in their papers proposing CKS and ABCD (datasets 1, 2, 3 in Table 1). Kim and Teh (2018) note that datasets larger than a few thousand points are *"far beyond the scope of [...] GP optimisation in CKS"* and consequently beyond the scope of ABCD as well. Therefore, datasets 1 - 3 are rather small. Especially for scalability analysis, we include five additional datasets (datasets 4-8 in Table 1).

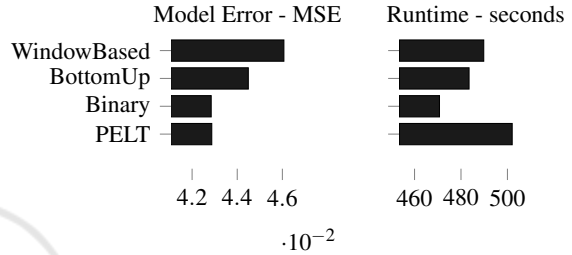

Figure 1: Impact of partitioning methods on average runtime and average model error of the TAGR algorithm.

As we focus on time series data in this paper and both algorithms CKS and ABCD have been evaluated for strictly one-dimensional input $X \subseteq \mathbb{R}$ and output data $Y \subseteq \mathbb{R}$ (Duvenaud et al., 2013; Lloyd et al., 2014), we will design our evaluation procedure accordingly.

In order to reflect practical circumstances, all experiments are executed on a 2019 Dell Laptop having a 1.9 GHz CPU (4 cores / 8 threads) with 16 GB of RAM. All algorithms were implemented using Python 3 and Tensorflow 2 (no GPU acceleration enabled). Since Duvenaud et al. (2013) and Lloyd et al. (2014) used $\mathcal{L}$ just for optimizing models and Mean Squared Error (MSE) for assessing model quality, we will adopt that procedure as well. For every evaluation, MSE reflects the five-fold cross-validated model error.

[2]Dimension "Carbon dioxide in ppm (room)" was used as target Y.

[3]Dimension "electrical power output (P_E)" was used as target Y.

[4]We chose one of the 20 utility zones (i.e. the first one) as the informational content among zones may be considered equivalent Hong et al. (2014). Our thorough empirical analysis indicated, that selecting this attribute over the other zones hardly affected the results.

[5]Dimension "Air_Temperature" was used as target Y. We are using the recognized intervall of this continuously gathered dataset from 2009-01-01 to 2016-12-31 (cf. Chollet, 2018)

[6]Dimension "Global_active_power" was used as target Y.

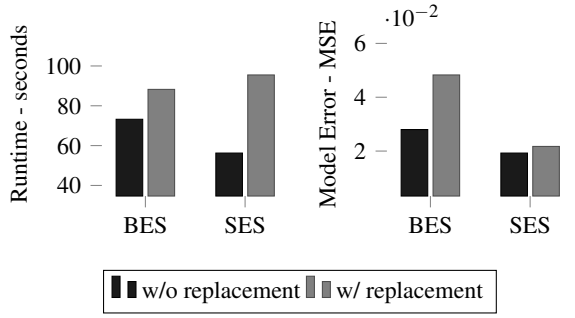

Figure 2: Impact of kernel expansion strategy on average runtime and average model error of the TAGR algorithm.

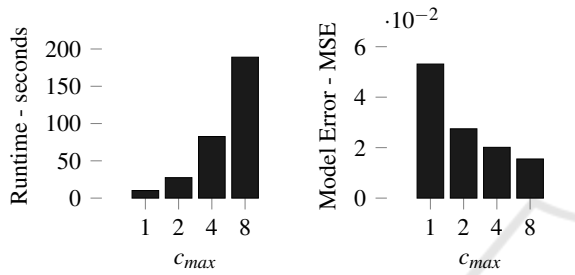

Figure 3: Impact of complexity parameter c_{max} on average runtime and average model error of the TAGR algorithm.

5.2 TAGR Configurations

In this first series of experiments, we evaluate the influence of various change point detection mechanisms to determine $\mathcal{T}$, model complexity parameter $c_{max} \in [1,2,3,4]$ and different kernel expansion strategies (i.e. BES and SES) on both efficiency of the computation, measured by runtime in seconds, and quality of the statistical model by MSE. We executed the TAGR algorithm with all possible parameter configurations on the datasets mentioned above (cf. Table 1). All experiments of the first series execute the TAGR algorithm in a non-parallel way. The results are summarized in Figures 2 and 3.

At first, we evaluate, which standard change point detection mechanism (cf. Truong et al., 2020; Aminikhanghahi and Cook, 2017) to use in order to determine the set of change points $\mathcal{T}$ as one of the parameters of TAGR. To do so, we compare the four mechanisms put forward by Truong et al. (2020) considering runtime performance and model error for executing TAGR utilizing the resulting change points. Figure 1 illustrates the given performance indicators averaged over all benchmark datasets and given parameter configurations of kernel expansion strategy and complexity c_{max}. Since *Binary Segmentation* offers one of the lowest average model error and benefits runtime the most, we choose this mean of change point detection for further use of TAGR.

Figure 2 shows, how different kernel expansion strategies impact average model error and average runtime performance of the TAGR algorithm. In general, BES is outperformed by SES and not using the *replacement tactic* further improves on both of these strategies. For most of the retrieved GPMs per benchmark dataset, the replacement tactic does not change the resulting model and only increases runtime by increasing the amount of candidate models. Still, some GPM retrievals are negatively affected by employing that tactic.

In addition to the different kernel expansion strategies, we evaluated the complexity parameter c_{max}. While model error assessed by MSE is positively influenced by increasing complexity parameter c_{max}, the opposite holds true for runtime performance (cf. Figure 3). Due to these results, we choose SES and a model complexity of $c_{max} = 4$ as default parameter configuration for the comparison of the proposed TAGR algorithm with existing state-of-the-art approaches. The chosen TAGR configuration does not include the *replacement tactic*.

5.3 Comparison with State of the Art

In the second series of experiments, we compare the performance in terms of model quality and runtime of the TAGR algorithm to state-of-the-art algorithms. In order to prevent interferences between parallelizability features and bare algorithmic qualities among the evaluated competitors, all experiments of the second series are executed in a non-parallel way. The runtimes are summarized in Figure 4, while the corresponding detailed results are listed in Table 2.

As can be seen in Figure 4, the TAGR algorithm's divisive approach leads to a lower growth of runtime in relation to the data size. This property comes especially handy for larger datasets. Considering for example the dataset *PowerPlant*, the SKC algorithm has a runtime of 9:19h ($\sim$ 33,533 seconds), while the TAGR algorithm only needs 0:01h ($\sim$ 68 seconds) to finish the corresponding computations. Hence, our proposal is approximately up to 500 times as fast as existing state-of-the-art solutions (SKC). This huge improvement can be traced back to TAGR's design, which not aims at retrieving one inextricable large model, but one consisting of small, independent partitions representing coherent fractions of the data. Subsequently, model evaluations by means of log marginal likelihood $\mathcal{L}$ or MSE are substantially cheaper.

Besides ensuring better runtime properties of our approach in comparison to CKS, ABCD and SKC, we also show that TAGR produces statistical models by

Table 2: Comparative evaluation results for state-of-the-art algorithms.

	Runtime				MSE			
Dataset	CKS	ABCD	SKC	TAGR	CKS	ABCD	SKC	TAGR
Airline	**00:00:01**	00:00:02	00:00:04	00:00:04	0.1361	0.0981	0.1175	**0.0269**
Solar	**00:00:03**	00:00:06	00:00:08	00:00:05	0.0856	0.0897	0.0845	**0.0769**
Mauna	00:00:15	00:00:26	00:00:24	**00:00:07**	0.1525	0.1519	0.1724	**0.0156**
SML	00:39:05	00:45:52	00:29:18	**00:00:25**	0.1103	0.1103	0.2259	**0.0112**
Power	12:27:51	14:37:15	09:18:53	**00:01:08**	0.0726	0.0726	0.1772	**0.0046**

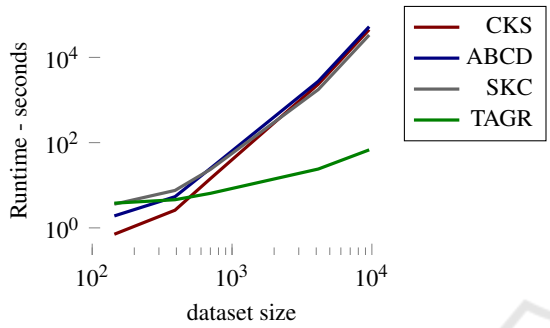

Figure 4: Runtime for different state-of-the-art algorithms as a function of dataset size.

Table 3: Performance and Model Quality.

	Runtime		
Dataset	**Parallel**	**Non-Parallel**	**MSE**
GEFCOM	0:03:44	0:13:09	0.0111
Jena	0:46:58	4:18:03	0.0109
Household	5:21:26	21:34:42	0.0024

means of kernel expressions of better quality. Table 2 illustrates the resulting model qualities per algorithm. In general, TAGR delivers better results regarding every dataset analyzed according to MSE measure.

5.4 Scalability of TAGR

The third and final series of experiments is only focused on evaluation of the TAGR algorithm. TAGR clearly outperforms state-of-the-art algorithms in terms of runtime, while still maintaining solid model quality. Nevertheless, the largest dataset considered comprises only approximately 10,000 records, which is by no definition large-scale. Thus, we aim to show the scalability features of our proposal in the remainder of this section, for which we employ the largest three of the selected datasets (cf. Table 1), containing up to two million records.

Table 3 presents the results of non-parallel and parallel execution of the TAGR algorithm for the given large-scale datasets. We executed it in a parallel as well as in a non-parallel fashion, to maintain comparability with given publications (cf. Lloyd et al., 2014; Duvenaud et al., 2013; Kim and Teh, 2018; Steinruecken et al., 2019), while still showing TAGR's scalability features. As parallelization of TAGR does not affect the resulting models, only one qualitative measure concerning model accuracy is given per analyzed dataset.

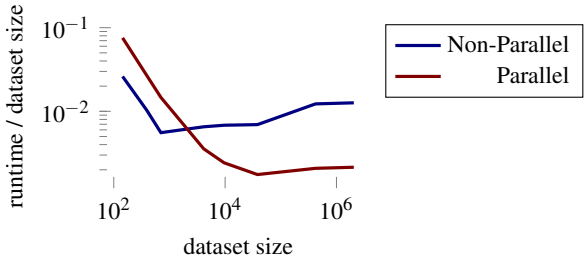

Figure 5: Runtime per Data Record.

Figure 5 illustrates how the ratio between runtime and dataset size evolves along absolute dataset size. It shows that TAGR becomes more efficient facing increasing dataset sizes. Beyond that, concurrent execution of our proposed algorithms building blocks reduces runtime compared to non-parallel execution. Those mentioned advances do not obstruct the accuracy of the derived GPM as shown in Tables 2 and 3.

To sum up, we have shown which kind of parameter configuration of our algorithmic approach produces the most promising results regarding model quality and performance. Based on that, we demonstrated how our approach produces statistical data models by means of Gaussian Processes of superior quality with regards to state-of-the-art algorithms, while outperforming these algorithms concerning runtime performance in general.

6 CONCLUSION

In this paper, we have investigated a new approach for efficient, automatic GPM retrieval for large-scale time series datasets. While state-of-the-art GPM retrieval algorithms lack scalability (cf. Berns and Beecks, 2020b), we propose a new structural design for large-scale GPMs. This design allows to get around the

two major bottlenecks of current methods by using a divisive approach to reduce effects of Gaussian Processes' cubic runtime complexity as well as employing a purposive strategy to generate fewer candidate models.

Our performance evaluation has revealed that GPMs resulting from the proposed TAGR algorithm deliver similar model quality in comparison to those models produced by state-of-the-art algorithms. In addition, runtime of the retrieval process is reduced significantly especially for larger time series, where we achieve a speed-up factor of approximately 500 with regards to existing methods such as CKS, ABCD and SKC.

As for future work, we consider global approximations an opportunity for further optimizing our approach. We therefore plan to address the opportunities of low-rank approximations such as the Nyström method (Hensman et al., 2013) in our future work. In addition, we plan to develop GPM retrieval algorithms for big data processing frameworks in order to scale GPM retrieval to very large and even multidimensional datasets.

REFERENCES

Abrahamsen and Petter (1997). A review of gaussian random fields and correlation functions: Technical report 917.

Aminikhanghahi, S. and Cook, D. J. (2017). A survey of methods for time series change point detection. *Knowl. Inf. Syst.*, 51(2):339–367.

Berns, F. and Beecks, C. (2020a). Automatic gaussian process model retrieval for big data. In *CIKM*. ACM.

Berns, F. and Beecks, C. (2020b). Towards large-scale gaussian process models for efficient bayesian machine learning. In *DATA*, pages 275–282. SciTePress.

Berns, F., Schmidt, K. W., Grass, A., and Beecks, C. (2019). A new approach for efficient structure discovery in iot. In *BigData*, pages 4152–4156. IEEE.

Cheng, C. and Boots, B. (2017). Variational inference for gaussian process models with linear complexity. In *NIPS*, pages 5184–5194.

Chollet, F. (2018). *Deep learning with Python*. Manning Publications Co, Shelter Island New York.

Csató, L. and Opper, M. (2000). Sparse representation for gaussian process models. In *NIPS*, pages 444–450. MIT Press.

Duvenaud, D., Lloyd, J. R., Grosse, R. B., Tenenbaum, J. B., and Ghahramani, Z. (2013). Structure discovery in nonparametric regression through compositional kernel search. In *ICML (3)*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1166–1174. JMLR.org.

Gittens, A. and Mahoney, M. W. (2016). Revisiting the nystrom method for improved large-scale machine learning. *J. Mach. Learn. Res.*, 17:117:1–117:65.

Hebrail, G. and Berard, A. (2012). Individual household electric power consumption data set. https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption. Accessed: 09/01/2020.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *UAI*. AUAI Press.

Hong, T., Pinson, P., and Fan, S. (2014). Global energy forecasting competition 2012. *International Journal of Forecasting*, 30(2):357–363.

Iliev, A. I., Kyurkchiev, N., and Markov, S. (2017). On the approximation of the step function by some sigmoid functions. *Math. Comput. Simul.*, 133:223–234.

Kim, H. and Teh, Y. W. (2018). Scaling up the automatic statistician: Scalable structure discovery using gaussian processes. In *AISTATS*, volume 84 of *Proceedings of Machine Learning Research*, pages 575–584. PMLR.

Li, S. C. and Marlin, B. M. (2016). A scalable end-to-end gaussian process adapter for irregularly sampled time series classification. In *NIPS*, pages 1804–1812.

Liu, H., Ong, Y., Shen, X., and Cai, J. (2020). When gaussian process meets big data: A review of scalable gps. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–19.

Lloyd, J. R., Duvenaud, D., Grosse, R. B., Tenenbaum, J. B., and Ghahramani, Z. (2014). Automatic construction and natural-language description of nonparametric regression models. In *AAAI*, pages 1242–1250. AAAI Press.

Low, K. H., Yu, J., Chen, J., and Jaillet, P. (2015). Parallel gaussian process regression for big data: Low-rank representation meets markov approximation. In *AAAI*, pages 2821–2827. AAAI Press.

Malkomes, G., Schaff, C., and Garnett, R. (2016). Bayesian optimization for automated model selection. In *NIPS*, pages 2892–2900.

Max Planck Institute for Biogeochemistry (2019). Weather Station Beutenberg / Weather Station Saaleaue: Jena Weather Data Analysis. https://www.bgc-jena.mpg.de/wetter/. Accessed: 09/01/2020.

Park, C. and Apley, D. W. (2018). Patchwork kriging for large-scale gaussian process regression. *J. Mach. Learn. Res.*, 19:7:1–7:43.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press.

Resende, M. G. and Ribeiro, C. C. (2016). *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures*. Springer New York, New York, NY.

Rivera, R. and Burnaev, E. (2017). Forecasting of commercial sales with large scale gaussian processes. In *ICDM Workshops*, pages 625–634. IEEE Computer Society.

Roberts, S., Osborne, M., Ebden, M., Reece, S., Gibson, N., and Aigrain, S. (2013). Gaussian processes for time-series modelling. *Philosophical transactions. Series*

A, Mathematical, physical, and engineering sciences, 371(1984):20110550.

Steinruecken, C., Smith, E., Janz, D., Lloyd, J. R., and Ghahramani, Z. (2019). The automatic statistician. In *Automated Machine Learning*, The Springer Series on Challenges in Machine Learning, pages 161–173. Springer.

Titsias, M. K. (2009). Variational learning of inducing variables in sparse gaussian processes. In *AISTATS*, volume 5 of *JMLR Proceedings*, pages 567–574. JMLR.org.

Truong, C., Oudre, L., and Vayatis, N. (2020). Selective review of offline change point detection methods. *Signal Process.*, 167.

Tüfekci, P. (2014). Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems*, 60:126–140.

Williams, C. K. I. and Seeger, M. W. (2000). Using the nyström method to speed up kernel machines. In *NIPS*, pages 682–688. MIT Press.

Wilson, A. G. and Adams, R. P. (2013). Gaussian process kernels for pattern discovery and extrapolation. In *ICML (3)*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1067–1075. JMLR.org.

Zamora-Martínez, F., Romeu, P., Botella-Rocamora, P., and Pardo, J. (2014). On-line learning of indoor temperature forecasting models towards energy efficiency. *Energy and Buildings*, 83:162–172.