Gaussian Processes for Hierarchical Time Series Forecasting

Luis Roque
luis_roque@live.com
Faculty of Engineering, University of Porto and LIACC
Porto, Portugal

Luis Torgo
ltorgo@dal.ca
Faculty of Computer Science, Dalhousie University
Halifax, Canada
Faculty of Computer Science, University of Porto
Porto, Portugal

Carlos Soares
csoares@fe.up.pt
Faculty of Engineering, University of Porto, Fraunhofer Portugal, AICOS and LIACC
Porto, Portugal

ABSTRACT

Recent time series forecasting tasks often involve large sets of related time series that can be naturally organized by various attributes of interest, e.g. a retailer that sells products for a variety of markets. In that case, forecasts for the most disaggregated series are required to add-up to the forecasts of the aggregated series. Gaussian Processes are a powerful tool for modeling time series behaviors, but recent work showed that prior knowledge was required in order for them to be competitive in automatic time series forecasting. The usage of prior knowledge can be considered an unfair advantage and not very useful in most real-world scenarios. We introduce a new hierarchical time series forecasting algorithm, Gaussian Process Hierarchical Forecaster (GPHF), that uses Gaussian Processes to jointly learn and forecast hierarchical time series. We propose a covariance mixture that, on one hand, allows the covariance matrices of the Gaussian Processes to share parameters and learn dependencies between series and, on the other hand, reduces the overall number of parameters required. Additionally, our approach uses a combination of a composition of kernels and a mean function that is able to fit time series patterns, while not requiring any complex kernel selection task. We show that Gaussian Processes are suited for automatic time series forecasting when working with hierarchical time series datasets and do not require any prior knowledge. Moreover, they can compete with methods that have an additional step of reconciliation. Finally, the introduction of the covariance mixture shows very significant improvements across hierarchical levels consistently for all datasets.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**.

KEYWORDS

gaussian processes, forecasting, hierarchical time series

MiLeTS '22, August 15th 2022, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Luis Roque, Luis Torgo, and Carlos Soares. 2022. Gaussian Processes for Hierarchical Time Series Forecasting. In *8th SIGKDD Workshop on Mining and Learning from Time Series' (MiLeTS '22).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Recent time series forecasting tasks often involve large sets of related time series. Thus, besides the traditional problem of learning the temporal dependencies, there is interest in modelling the cross-series information to improve the univariate models. Additionally, in many real-world situations, the time series have inherent hierarchical relations and structures. Examples include time series forecasting of the Gross Domestic Product [3], epidemics [9], sales demand data [15] and others. Hierarchical time series forecasting focuses on effectively modeling multiple time series that are consistent with given hierarchical relations. When the generated forecasts satisfy these relations, they are called coherent and this is usually required in many applications [16]. In this setting, the combination of leveraging cross-series information and the hierarchical relations can further increase the performance of algorithms ([13], [28]). Additionally, in several businesses settings, forecasts need be generated automatically, i.e. without any human intervention [14], and there is no prior information to rely on such as in [5].

Classical methods [11] for this type of problem typically use a Bottom-Up (BU) or a Top-Down (TD) approach where all time series for a single level of hierarchy (usually top-most or bottom-most) are modeled independently. The values of time series of other levels are derived using the respective aggregation function governing the hierarchy. Another approach relies on producing forecasts for all aggregation levels and then *reconcile* the forecasts using linear or non-linear models (e.g. [16], [27] or [21]). There are also approaches that address the dependencies between all the time series in the dataset without specifically addressing the hierarchical structure ([26], [7]).

Gaussian Processes are very flexible, expressive, interpretable and inherently probabilistic. A Gaussian Process is a generalization of the Gaussian probability distribution in the sense that it is governing the properties of functions instead of scalar or vector values. In simple terms, it works by defining prior probabilities over functions, before actually seeing any data. After observing new data points, we update our belief and increase the probability of the functions that we consider to be more likely to explain our data. Notice that it is not a parametric model, so we do not have to be worried whether it is possible for the model to fit the data. We can

sample functions at any point and not only compute the mean value but also the variance (i.e. the variability of the sample functions). This variance is often referred as uncertainty and the way Gaussian Processes are built ensures that uncertainty reduces around observations as we are more informed about the real value of the process at that instant. This is a nice property and very helpful in a time series forecasting setting, since when we extend the forecasting horizon we should be less confidence about our predictions.

Gaussian Processes are powerful when modeling correlated observations, which is the reality of a time series. Nevertheless, there are no competitive approaches for automatic forecasting based on Gaussian Processes [5]. In the same article, the authors presented the idea that prior knowledge is required for a Gaussian Process to be competitive against classical statistic forecasting methods such as Autoregressive Integrated Moving Average (ARIMA) and Exponential Smoothing (ETS). Furthermore, the fitting of a Gaussian Process depends on the choice of kernel functions and these have a strong impact on model performance. Moreover, algorithms which automatically optimize the kernel composition [20] do not scale, given the need for training a large number of competing GP models. Finally, there is no clear way to present a hierarchical structure to a Gaussian Process. While there are Multi-Output Gaussian Processes (MOGP) stragies [17], these tend to be extremely expensive computationally. The reason is that besides learning multiple covariances, one for each output, it usually requires learning cross covariances across pairs of outputs. Additionally, in a hierarchical setting, there would be the need to use several MOGP strategies (e.g. for each group) and then combine them using some other strategy. An example of a MOGP strategy applied in a financial time series analysis can be found here [6].

We introduce a new algorithm - Gaussian Process Hierarchical Forecaster (GPHF) - to forecast datasets of hierarchically organized time series using Gaussian Processes. Our algorithm is able to learn the dependencies between the related time series that belong to the same groups. Note that we do not use any reconciliation process, the dependencies are learned by the Gaussian Processes themselves. By making use of this information the model improves the forecast of individual time series. Since a Gaussian Process is completely specified by its mean and covariance functions, we used both functions to capture different components of the time series. In fact, we can think of the way we defined the hyperparameters of both functions as being at different levels, the first are estimated locally, i.e. for each time series, while the second are estimated by group element in the dataset. On one hand, this means that we are learning non-stationary patterns, such as the trend, per individual time series. On the other hand, we are sharing the knowledge for each group when learning the stationary and non-stationary components, namely irregularities or seasonality. The strategy of exploiting global patterns and coupling them with local calibration has been followed recently in high-dimensional time series forecasting [25]. Another motivation to fit the trend separately and locally was that in time series generated by stochastic processes, the separation between noise and changes in the trend is hard to capture. Using combinations of kernel functions to model these patterns would require some degree of prior information on the type of data to achieve competitiveness [5]. This is due to the flexible nature of Gaussian Processes, which can work against us in certain settings. The interpretability also improves by using a specific function as the mean function. In our case, we used a piece-wise linear function, to allow the model to capture non-linear trend patterns.

To the best of our knowledge, this is the first approach that uses a formulation of Gaussian Processes to solve a hierarchical time series forecasting problem.

In summary, our contributions are:

- A new strategy to extend Gaussian Processes to a hierarchical time series setting by introducing a covariance mixture that models the hierarchy. This allows learning the dependencies between series in the same group, while reducing the number of parameters of the model;
- A flexible composition of kernels that learn global stationary and non-stationary patterns in the data, namely irregularities and seasonality. The trend is learned by the mean function of the Gaussian Process applied at the local level, using a piece-wise linear function;
- An algorithm that fits the data automatically and without any prior knowledge due to the flexible composition of kernels used together with non-linear mean function and the covariance mixture;
- An empirical validation of the proposed model in four different real-world datasets indicating that it is competitive with the state-of-the-art when working with small and large datasets with typical trend and seasonal patterns varying between groups. Moreover, our algorithm can compete with methods that have an additional step of reconciliation. We also showed that the introduction of the covariance mixture shows very significant improvements across hierarchical levels consistently for all datasets.

2 RELATED WORK

2.1 Time Series Forecasting Methods

When we consider the forecasting process of a single time series and include constraints such as integer or single seasonality the state-space exponential smoothing (ETS) [11] and automated ARIMA [14] procedures are still considered state of the art approaches. When we extend to non-integer or multiple seasonalities, there are other methods which become relevant, including TBATS [18] or Prophet [29]. On the other hand, Recurrent Neural Networks (RNN) are gaining popularity as an alternative to statistical methods. Nevertheless, the settings where they can achieve competitiveness are still very narrow and require user adaptation (see [10] for an extensive study on the topic).

Traditional Gaussian Processes do not excel in automatic forecasting. Currently, to obtain competitive results on single univariate time series, it is necessary to include prior knowledge in GP methods [5]. There are several challenges when using Gaussian Processes to do automatic forecasting, namely selecting the right kernels and the long time required for training different competing kernels.

2.2 Hierarchical and Grouped Time Series

When working with sets of related time series, the focus is to model cross-series information to improve univariate models. There are

cases that address these dependencies in general, while others assume that the dependencies are of a specific kind, e.g. hierarchical. In the former, the only assumption is that time series are related by belonging to the same domain ([26], [7]) and these implicit relationships can be learned to improve the forecasts of each time series. In the case of [7], an autoregressive recurrent neural network learns a global model from the historical data of all time series in the dataset, at the same time. It is a probabilistic approach that produces forecasts in the form of Monte Carlo samples that allow the computation of quantile estimates. The model is both autoregressive, since it consumes the observation at the previous time step as input and recurrent, in the sense that the previous output of the network is fed back as an input at the next time step. The likelihood should match the statistical properties of the data and its parameters are given by a function of the network output. In the case of hierarchical time series settings, the hierarchical structure can be passed as multiple static categorical features to the model.

There are other cases where the data can be aggregated in groups or a hierarchy. While the models presented above can be used to produce forecasts for hierarchical datasets, the explicit information that encodes the hierarchy of the time series is not leveraged. There are different methods designed to model the hierarchical nature of these datasets explicitly. The method initially proposed by [12] and improved in [2] and [16] consists of optimally combining and reconciling all forecasts at all levels of the hierarchy. A linear regression combines the independent forecasts, guaranteeing that the revised forecasts are as close as possible to the independent forecasts but maintaining coherence. These approaches were further extended to allow a non-linear combination of the base forecasts [27]. They were adapted to a Bayesian perspective, that considers the uncertainty across all levels of the hierarchy to obtain the revised forecasts [21].

3 PROPOSED APPROACH

3.1 Problem Definition and Notation

We are working with a collection of S related univariate time series $\mathcal{Z}$, $\mathcal{Z} = \{\mathbf{z}_t^i, t \in \mathbb{N}, i = 1 \dots S\}$. The training values can be written as $\mathbf{z}_{1:T}^i = [z_1^i, z_2^i, \cdots, z_T^i]$ where $z_t^i \in \mathbb{R}$ denotes the value of time series i at time t and T represents the last training point. When the interpretation is unambiguous and in order to simplify the notation in specific sections, namely when introducing Gaussian Processes, we simply refer to time series $\mathbf{z}^i$ as the observed time series. The training range is denoted by $\mathbf{x} = \{1, 2, ..., T\}$, while $\mathbf{x}_* = \{T+1, T+2, \dots, T+\tau\}$ is the prediction range and τ is the forecast horizon. We are interested in getting point predictions $\hat{\mathbf{z}}_{T+1:\tau}^i$ from the posterior distributions. The posterior distributions for time series i are denoted as $\mathbf{z}_*^i = [f(x_{*1}), \dots, f(x_{*\tau})]^T$. Notice that we use the star notation in order to ensure consistency with [23]. The point forecast error for time series i and time t is denoted by $e_{1:\tau}^i = \hat{z}_{1:\tau}^i - z_{T+1:\tau}^i$.

Forecast accuracy is usually measured by summarizing the forecast errors using a scaled metric – e.g. the Mean Absolute Scaled Error (MASE) (see [11] for an extended overview on forecast error metrics). For seasonal time series, a scaled error can be computed by:

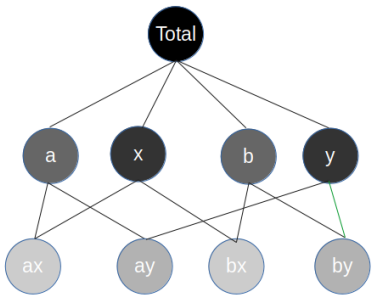

Figure 1: Example of time series aggregated by group.

$$MASE = \frac{\frac{1}{\tau}\sum_{t=1}^{\tau}|e_t|}{\frac{1}{T-m}\sum_{t=m+1}^{T}|z_t - z_{t-m}|}, \quad (1)$$

where e_t is the forecast error for a set of forecasts τ. The denominator is the mean absolute error of the one-step naive forecast on the training set.

The time series are organized into different groups; in fact, each time series $\mathbf{z}^i$ is associated with an element l of every group G present in the dataset. We sometimes write $L_i = \{l : \mathbf{z}^i \in G_l\}$ to refer to the subset of elements to which time series $\mathbf{z}^i$ belongs to. To give a simple example, consider a dataset with two groups (g_1 and g_2), each one with 2 different elements (a and b, x and y, respectively). We can think of g_1 as being different locations where a retailer has stores and g_2 categories of the products sold at those stores. We start with the top level of the data $\mathbf{z}_t$. We can aggregate the individual series $\mathbf{z}_t^i$ by group g_1, forming the series $\mathbf{z}_{g_1,t}$, and by its elements, forming $\mathbf{z}_{a,t}$ and $\mathbf{z}_{b,t}$. We can do the same for the second group g_2, forming series $\mathbf{z}_{g_2,t}$, or by its elements, $\mathbf{z}_{x,t}$ and $\mathbf{z}_{y,t}$. At the bottom level, this would generate four different series (i.e. $\mathbf{z}_{ax,t}$, $\mathbf{z}_{ay,t}$, etc). Figure 1 illustrates this particular example of dataset.

Our goal is to forecast accurately all levels of the hierarchy, not just the bottom or top levels. Using the previous example, we compute the error metric for all the bottom time series, e.g. MASE($\mathbf{z}_{ax,t}$). Since we also want to evaluate the performance for the aggregated levels, we also compute the error metric for every group element l, which is the result of the addition of all the bottom series that belong to the specific group element $\mathbf{z}_{a,t} = \mathbf{z}_{ax,t} + \mathbf{z}_{ay,t}$. Finally, to compute a metric for every group G we average MASE across all the elements l that belong to the group, e.g. $\mathbf{z}_{a,t}$ and $\mathbf{z}_{b,t}$ for group g_1. For the most aggregated level of the data the exercise is similar, but this time we add the values of all the bottom time series together and evaluate the predictions on those values.

3.2 Gaussian Processes

In a Gaussian Process we directly infer a distribution over functions. Each function can be seen as a random variable assigned to a finite number of discrete training points $\mathbf{x}$. Any finite number of these variables have a joint Gaussian distribution. More formally, a

Gaussian Process is completely specified by its mean and covariance functions:

$$f(x) \sim GP\big(m(x), k_\theta(x, x')\big) \tag{2}$$

where $m(x)$ denotes the mean function and k_θ denotes the kernel with hyperparameters θ.

The kernels define the types of functions that we are likely to sample from the distribution of functions [23]. We can then draw samples from the distribution of functions evaluated at any number of points, i.e. $\text{cov}(f(x), f(x')) = k(x, x')$. They can be separated into stationary kernels, such as the squared exponential kernel (RBF) and the periodic kernel (PER) and non-stationary ones, such as the linear kernel (LIN). The stationary kernels can be written as

$$\text{RBF}: \qquad k_r(x, x') = \eta_r^2 \exp\big(-\frac{1}{2l_r^2}(x - x')^T(x - x')\big) \tag{3}$$

$$\text{PER}: \qquad k_p(x, x') = \eta_p^2 \exp\big(-\frac{(2sin^2(\pi|x - x'|/p)}{l_p^2})\big) \tag{4}$$

where η_r, η_p, η_l represent the variances, l_r, l_p are the length-scale parameters which control the smoothness, c defines the offset and p is the period.

When we are predicting using Gaussian Processes, we want the training outputs $\mathbf{z}^i$ and the test outputs $\mathbf{f}_*^i$ to be correlated, so we can not have them as two independent Gaussian distributions. We are interested in the joint distribution of $\mathbf{z}^i$ and $\mathbf{f}_*^i$. For most of the applications we are faced with approximate function values, since there is noise to be considered in the form $\mathbf{z} = f(x) + \epsilon$. Assuming additive independent and identically distributed Gaussian noise with variance σ_n^2, the joint distribution of the observations and the function values at the test positions can be written as

$$\begin{bmatrix} \mathbf{z} \\ \mathbf{z}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_{xx} + \sigma_n^2\mathbf{I} & \mathbf{K}_{x*} \\ \mathbf{K}_{*x} & \mathbf{K}_{**} \end{bmatrix}\right) \tag{5}$$

where $\mathbf{K}_{x*}$ denotes the matrix of the covariances evaluated at all pairs of training $\mathbf{x}$ and test points $\mathbf{x}_*$, which means that it is a matrix of size $T \times \tau$. For the other entries, $\mathbf{K}_{xx}$, $\mathbf{K}_{*x}$, $\mathbf{K}_{**}$ the same idea applies. Finally, we can derive the conditional distribution following [23],

$$\mathbf{z}_*|\mathbf{x}, \mathbf{z}, \mathbf{x}_* \sim \mathcal{N}(\bar{\mathbf{z}}_*, \text{cov}(\mathbf{z}_*)) \tag{6}$$

$$\bar{\mathbf{z}}_* \triangleq \mathbb{E}[\mathbf{z}_*|\mathbf{x}, \mathbf{z}, \mathbf{x}_*] = \mathbf{K}_{*x}[\mathbf{K}_{xx} + \sigma_n^2\mathbf{I}]^{-1}\mathbf{z} \tag{7}$$

$$\text{cov}(\mathbf{z}_*) = \mathbf{K}_{**} - \mathbf{K}_{*x}[\mathbf{K}_{xx} + \sigma_n^2\mathbf{I}]^{-1}\mathbf{K}_{x*} \tag{8}$$

Exact inference for standard Gaussian Processes can be intractable for large datasets. Most of the implementations address the matrix inversion $\mathbf{K}_{xx}^{-1}\mathbf{z}$ using the Cholesky factorization [23], which requires $O(n^3)$ time and $O(n^2)$ memory. Alternative approaches have been developed based on matrix-vector multiplications, namely using conjugate gradients. They define an optimization problem that iteratively approximates $\mathbf{K}_{xx}^{-1}\mathbf{z}$. We are interested in the approach introduced by [8] and called Blackbox Matrix-Matrix multiplication. It is a modified batched version of the conjugate gradients algorithm that derives all terms for training and inference in a single call. The modifications also allow the algorithm to run on parallel compute hardware. It reduces the asymptotic complexity of exact Gaussian Processes inference to $O(n^2)$. The complexity can be further reduced with structured data or sparse GP approximations [30].

3.3 Hierarchical Structure

As we saw in Section 2.2, our work is focused on datasets that have a hierarchical or grouping structure, and thus, there are often dependencies between the time series nested in the hierarchy. We defined a set of Gaussian Processes, of which the covariance functions are used to capture seasonality, irregularities and noise, while the mean functions model the non-linear trend using a piece-wise linear model.

$$f_{\phi^i\theta^i}(x) \sim GP\big(m_{\phi^i}(x), k_{\theta^i}(xx')\big) \tag{9}$$

$$\epsilon^i \sim \mathcal{N}(0, \Sigma) \tag{10}$$

Before fitting our hierarchical model, we have standardized each time series to have mean 0 and variance 1. If we consider $\mathbf{z}^i$ to be generated by a series of independent Gaussian Processes, we can say that the we have different functions parameters and noise for each time series i. We could evaluate the functions considering two arbitrary points x and x':

$$f_{\phi^i\theta^i}(x) \sim GP\big(m_{\phi^i}(x), k_{\theta^i}(xx')\big) \tag{11}$$

$$\epsilon^i \sim \mathcal{N}(0, \Sigma) \tag{12}$$

$$\tag{13}$$

Hence, our observations are simply the Gaussian Process function values plus the independent and identically distributed noise ϵ evaluated at the input points $\mathbf{x} = \{1, 2, \ldots, T\}$.

$$\mathbf{z}^i = f_{\phi^i\theta^i}(\mathbf{x}) + \epsilon^i \tag{14}$$

For the mean function of the Gaussian Processes we defined a piece-wise linear function that allows learning non-linear trend patterns in the data. The parameters are all learned locally, i.e. each time series has its own set of parameters. We followed the proposal from [29], which we cover in more detail in section 3.5. The authors defined an initial growth parameter $\mathbf{k}^i$ and subsequent changes on the growth rate were denoted by the rate adjustment parameter δ_c^i. The changes can happen for every change points $\mathbf{c}$ that we set. The expression $\mathbf{m}^s + A(-\mathbf{c}\delta_c^s)$ ensures the function continuity across the different change points and growth rate changes. Thus, it can be written as:

$$\mathbf{p}_t^i = (\mathbf{k}^i + \mathbf{A}\delta_c^i)\mathbf{x} + (\mathbf{m}^i + \mathbf{A}(-\mathbf{c}\delta_c^i)) \tag{15}$$

where $\mathbf{p}^i$ is the piece-wise linear function used as the mean function of the Gaussian Process that models the time series i.

Following the motivation to model the dependencies nested in the hierarchy of the data, we want each element l of a group G to be explicitly expressed in the covariance matrix of each Gaussian Process. We applied this mixture by summing covariance functions $\mathbf{k}_{\theta^{G_l}}$ with hyperparameters defined for each group element. These covariance functions are summed for all group elements L that time series i belongs to, such that $L = \{l : \mathbf{z}^i \in G_l\}$. Note that the way we

defined our model reduces significantly the number of hyperparameters required when compared to a standard approach that would define a Gaussian Process per time series. In fact, the number of sets of hyperparameters required for each kernel function θ is reduced from the number of time series in the dataset i to the number of group elements l for every group G. The Gaussian Processes can then be defined as:

$$\gamma_t^i \sim \mathcal{N}(\mathbf{p}_t^i, \sum_{l:\mathbf{z}^i \in G_l} \mathbf{K}_{\theta^{G_l}}) \tag{16}$$

Our parameters are approximated using the Blackbox Matrix-Matrix multiplication algorithm, introduced in Section 3.2.

3.4 Covariance Mixture

Gaussian Processes were already defined as latent processes in a time series forecasting context [24]. The problem with this approach is that we cannot take advantage of the nice properties of the marginal likelihood of a Gaussian Process, namely the fact that it has a closed form. In this case, the observed data needs to be the sum of a Gaussian Process $f(x) \sim GP\big(m(x), k(x,x')\big)$ and random noise ϵ in the following form $\mathbf{z} = f(x) + \epsilon$. The unknown latent function can be analytically integrated out, which gives us the marginal likelihood:

$$p(\mathbf{z}|\mathbf{x}) \quad = \quad \int p(\mathbf{z}|\mathbf{f},\mathbf{x})p(\mathbf{f}|\mathbf{x})d\mathbf{f} \tag{17}$$

The log of the marginal likelihood $p(\mathbf{z}|\mathbf{x})$ can, thus, be written as:

$$\begin{aligned} \log p(\mathbf{z}|\mathbf{x}) \quad = \quad & -\frac{1}{2}(\mathbf{z}-\mathbf{m}_x)^T(\mathbf{K}_{xx}+\sigma)^{-1}(\mathbf{z}-\mathbf{m}_x) \\ & -\frac{1}{2}\log(\mathbf{K}_{xx}+\Sigma) - \frac{n}{2}\log(2\pi) \end{aligned} \tag{18}$$

where Σ is the covariance matrix of the Gaussian noise.

In order to move from the latent approach to one that we could assume that the Gaussian Process was actually observed, we re-expressed our model in a different way, using the addition property of Gaussian Processes:

$$GP(\mu_1+\mu_2, \mathbf{K}_1+\mathbf{K}_2) = GP(\mu_1,\mathbf{K}_1) + GP(\mu_2,\mathbf{K}_2) \tag{19}$$

Instead of applying the mixtures directly to the Gaussian Processes and then adding them together, as it is shown in the right side of the equation above, we applied the mixture to the covariance matrices and added these, as it is shown in the left side of the equation. This way, we can assume that each of our Gaussian Processes is observed as a single time series. The trick is in the way that we define the covariance matrix for each of the Gaussian Processes, since they are the ones encoding the hierarchical structure of the data.

$$\gamma_t^i \sim \mathcal{N}(0, \sum_{l:\mathbf{z}^i \in G_l} \mathbf{K}_{\theta^{G_l}}) \tag{20}$$

Finally, we can write our covariance functions as:

$$\mathbf{K}_{\theta^{G_l}} = \mathbf{K}_{\theta^{G_l},RBF} + \mathbf{K}_{\theta^{G_l},PER} = \mathbf{K}_{l_r^{G_l},\eta_r^{G_l}} + \mathbf{K}_{p^{G_l},l_p^{G_l},\eta_p^{G_l}} \tag{21}$$

where l_r, l_p, η_t, η_p and p are the hyperparameters to learn.

As a final note regarding kernel design, our algorithm can be trivially extended to have multiple seasonalities. This is useful when, for example, there is a weakly seasonality pattern in the data aside from the main pattern. It can be done by adding a new component to our covariance function. A second periodic kernel PER can be added on and the prior for its period p can be defined for the specific seasonality.

3.5 Mean Function

Preliminary experiments with different kernel compositions that included a kernel to model trend patterns in the data lead to unconvincing results. Therefore, we have decided to not use kernel functions to model that particular pattern very common in time series data. We decided to model the trend of the data using a piece-wise linear model, following the definition in [29]. The authors proposed two different formulations, one for saturated growth and one for linear growth. We are interested in estimating linear growth but, in a specific use case where saturated growth could be useful, our model could be trivially extended. We start by defining a parameter k that accounts for the initial growth. To allow differences in the trend, we set evenly distributed change points $\mathbf{c}$ at times $c_j, j = 1, ..., C$. Note that the location of the change points is fixed, what we learn is the vector of trend adjustments $\delta \in \mathbb{R}^c$. At each change point c_j our trend is allowed to change with a certain δ_j. The way to express this idea is by using a vector $\mathbf{a}_j(t)$ to indicate, for every time point, if the change point had already happened:

$$\mathbf{a}_j(t) = \begin{cases} 1, & \text{if } t \geq c_j, \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

To simplify the notation we define a matrix $\mathbf{A} = \mathbf{a}_j(t)^T$ of size n x c, number of time points per number of change points. The growth changes are then defined by $\mathbf{A}\delta$. An offset that it is not time dependent but only change point dependent is added to avoid having non-continuous function. The piece-wise linear function can thus be written as

$$\mathbf{p} \quad = \quad (\mathbf{k} + \mathbf{A}\delta)\mathbf{x} + (\mathbf{m} + \mathbf{A}(-\mathbf{c}\delta)) \tag{23}$$

where k is the initial growth.

We will not be focusing on the effect change of using a different number of change points, nor on the automatic selection of the number of change points. Nevertheless, it is possible to address this problem with the current formulation. If one chooses a large number of potential points and uses a sparse prior on the δ parameter, it is the equivalent of performing a L1 regularization.

4 EMPIRICAL VALIDATION

4.1 Datasets

The empirical evaluation uses four public datasets for our evaluations (see Table 1). We chose datasets with different characteristics to asses whether our algorithm is able to capture different types of trend and seasonal patterns, that vary across groups, while working with either small or large number of series. Notice that the third and fourth datasets are a subset of the original ones ([19], [1]. The former was downsampled to a weekly basis, and, for both datasets,

Dataset	Source	Characteristics	Groups
Prison	Australian Bureau of Statistics	Quarterly values, 32 time series with 48 time points	8 states, 2 genders, 2 legal status
Tourism	Australian Bureau of Statistics	Monthly values, 304 time series with 228 time points	8 states, 27 zones, 76 regions, 4 travel purposes
M5	Wallmart sales	Weekly values, 500 time series with 275 time points	5 departments, 3 categories, 10 stores, 3 states, 160 items
Police	Houston police	Daily values, 500 time series with 334 time points	19 types of crimes, 79 beats, 10 streets, 67 zip codes

Table 1: Datasets and respective summary. The respective sources are: [11], [4], [19] and [1].

we have selected the 500 bottom series with higher count levels. We took this decision due to the fact that the bottom level time series were very sparse and consistently showed intermittent patterns. These are in themselves two active subsets of the time series forecasting research, which we would rather avoid for now. Also, it allowed us to increase the speed of the experimentation process. Given that all methods were tested on the same datasets, this does not affect the validity of the results.

4.2 Experimental Setup and Baselines

The experiments assess how the performance of the proposed method compares to the performance of the state-of-the-art methods when forecasting a set of hierarchical time series. We were interested in understanding if any of the methods showed a trade-off between fitting the bottom level time series and the top level ones. Also, as we are forecasting several time series per group of the dataset, we wanted to evaluate if any of the models showed higher variability within a group.

In designing our GPs, the goal was to define a set of kernels that is flexible enough to be used in different settings. The combination of kernels that we used included the squared exponential kernel (RBF) and the period kernel (PER). The equations for each kernel were presented in Section 3.2. The RBF kernel was selected to model medium term non-linear irregularities in the data. We explored the selection of prior distributions and parameters for the kernel hyperparameters, which can help the model convergence, nevertheless, it affects negatively the model scalability. After preliminary experimentation, we decided to only supply the mean values proposed in our preliminary work to each of the hyperparameters. For the length-scales l_r, l_p, we defined 1 and 0.5 respectively. For the scale parameters, we supplied η_t with 0.5 and η_p with 1.5. To model the main seasonal pattern of the data we selected p based on the seasonality of the dataset.

For every dataset, we compare our model with the following approaches (see Figure 2 and Table 2):

- Standard GP: the equivalent of applying one Gaussian Process to each time series individually. Note that the Gaussian Processes have exactly the same kernel selection and mean function defined for our proposed algorithm - GPHF;
- DeepAR: a model that produces probabilistic forecasts based on training an auto regressive recurrent network model on related time series, introduced in Section 2. The hierarchy of the dataset was passed as multiple static categorical features to the model;
- ETS + BU: the ETS method was applied to the bottom time series and then the BU strategy was followed to aggregate the forecasts to the upper hierarchical levels;
- ETS + MinT: once again the ETS method was used as the base forecaster but this time for all the time series (including the upper levels). To ensure the coherency of the forecasts, MinT, the reconciliation method proposed by [16], was used.

4.3 Results and Discussion

Table 2 presents the MASE for the combination between all the different methods: our proposed method GPHF, the standard Gaussian Processes, DeepAR, ETS using the BU strategy and ETS using the MinT reconciliation method. It also includes all the datasets and all the hierarchical levels. The results indicate that the proposed method obtains competitive results, outperforming in several occasions the state-of-the-art methods. This is somewhat surprising, given previous work results using Gaussian Processes for time series forecasting, and taking into account that no prior information was used. Even for the bottom level series, that Gaussian Processes usually struggle against classical methods, GPHF is able to be consistently at the benchmark across all datasets. We can also see that this competitiveness comes from the introduced covariance mixture, since there are significant differences between the results of GPHF and the ones from the standard Gaussian Processes. The differences are significant not only for the bottom levels but also for the upper levels in the hierarchy. It shows that there is valuable information that the method is capturing from the hierarchical structure of the data. In fact, the magnitude of the increased performance surpasses largely the improvement that MinT had over a simple BU strategy. And recall that our method does not require any additional reconciliation step. Furthermore, our method is reliable at establishing a compromise between the bottom and top level prediction errors. The same can be said regarding the dataset size, our method is able to keep a good balance there also. DeepAR, for example, struggles more often in both dimensions.

Figure 2 helps to get a more detailed perspective on the distribution of the forecast error within groups. Once again, no method shows a variance significantly higher than the rest, which means that inside each group the behaviour is also similar. These results confirm that the proposed method is competitive with the state-of-the-art. Even in the cases where we see outliers, e.g. in the item or bottom groups in the M5 dataset or in the beat, crime, zip or bottom groups in the police dataset, no method shows significant capacity to overcome those more challenging time series.

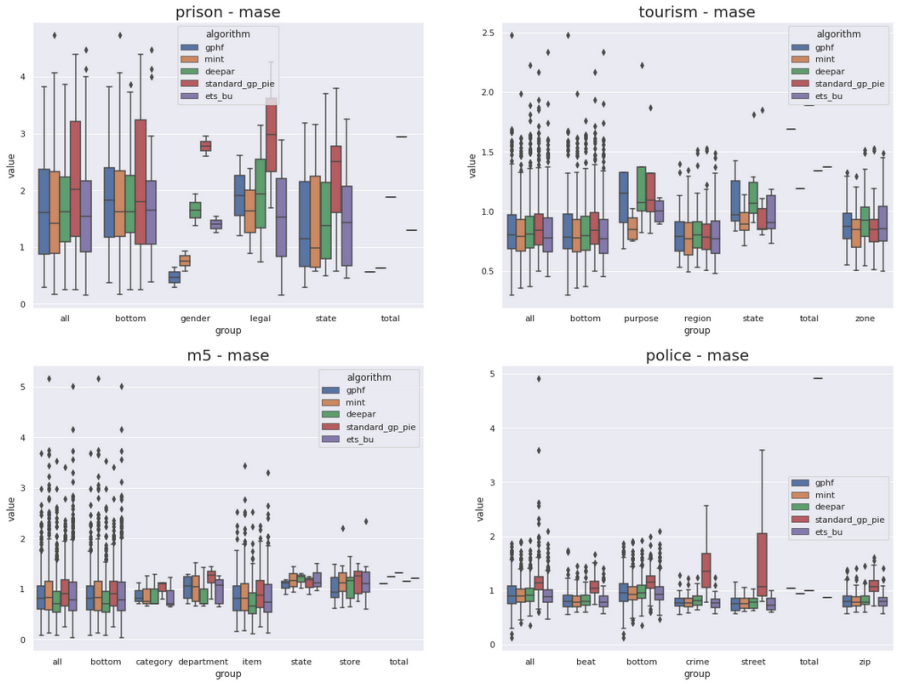

Figure 2: Visual representation of the results (MASE) for the different datasets considered in the experiments. We used box plots to show the distribution of the error inside each group. In light of these results, our proposed method shows similar capacity to forecast hierarchical time series as the state-of-the-art approaches.

Dataset	Algorithm	bottom	total	state	gender	legal	-	-	all
Prison	GPHF	1.92	**0.56**	**1.48**	**0.46**	1.91	-	-	1.74
	Standard GP	2.12	2.94	2.31	2.78	2.98	-	-	2.24
	DeepAR	1.83	1.88	1.60	1.65	1.93	-	-	1.79
	ETS + BU	1.78	1.30	1.59	1.40	1.52	-	-	1.70
	ETS + MinT	**1.77**	0.63	1.50	0.76	**1.63**	-	-	**1.64**
Dataset	Algorithm	bottom	total	state	zone	region	purpose	-	all
Tourism	GPHF	0.84	1.69	1.08	0.89	0.80	1.07	-	0.85
	Standard GP	0.89	1.34	1.04	0.89	0.81	1.22	-	0.88
	DeepAR	0.84	1.89	1.18	0.94	0.83	1.3	-	0.85
	ETS + BU	**0.82**	1.37	0.97	0.91	0.80	1.00	-	0.83
	ETS + MinT	**0.82**	**1.25**	**0.92**	**0.86**	**0.78**	**0.88**	-	**0.82**
Dataset	Algorithm	bottom	total	department	category	store	state	item	all
M5	GPHF	0.87	**1.10**	1.02	**0.87**	**1.02**	**1.06**	0.86	0.87
	Standard GP	0.96	1.15	1.20	1.01	1.16	1.11	0.94	0.96
	DeepAR	**0.79**	1.32	**0.90**	0.90	1.08	1.19	**0.74**	**0.79**
	ETS + BU	0.92	1.20	0.97	**0.87**	1.18	1.20	0.87	0.92
	ETS + MinT	0.93	1.20	0.99	**0.87**	1.15	1.17	0.88	0.92
Dataset	Algorithm	bottom	total	crime	beat	street	zip	-	all
Police	GPHF	**0.96**	1.04	**0.80**	**0.83**	**0.77**	**0.81**	-	**0.92**
	Standard GP	1.16	4.92	1.41	1.08	1.53	1.09	-	1.16
	DeepAR	0.98	1	0.83	0.84	0.79	0.82	-	0.94
	ETS + BU	**0.96**	**0.86**	**0.80**	**0.83**	**0.77**	**0.81**	-	**0.92**
	ETS + MinT	**0.96**	0.94	**0.80**	**0.83**	**0.77**	**0.81**	-	**0.92**

Table 2: Results (MASE) for the different datasets considered in the experiments. *Bold* values represent the smallest error across algorithms. The errors are calculated for each group and for the bottom and top level series.

5 CONCLUSIONS AND FUTURE WORK

We proposed a new algorithm to perform time series forecasting on datasets of related time series with a hierarchical structure. We propose a covariance mixture that allows the covariance matrices of the Gaussian Processes to share parameters and learn dependencies between series. At the same time, it reduces the overall number of parameters required. Additionally, our approach uses a combination of a composition of kernels and a mean function that is able to fit time series patterns, while not requiring any complex kernel selection task.

Our method was able to effectively capture the behaviours of the top level series, while not losing accuracy on the individual ones. We tested our hypothesis on four real world datasets of different sizes and frequencies: daily, weekly, monthly and quarterly. Interestingly enough, the results show that Gaussian Processes can compete with state-of-the-art methods in time series forecastings tasks.

As future work, an interesting research question to answer is the quantification of the uncertainty by our approach when compared to the state-of-the-art methods. Gaussian Processes, besides being inherently Bayesian, provide out of the box mean and variance estimates (which is not the case for most machine learning models). This variance is often referred as uncertainty and the way Gaussian Processes are built ensures that uncertainty reduces around observations, which is a nice property and very helpful in a time series forecasting setting. Also, the scalability of the model can be developed further if we consider approximation methods, such as sparse approximations (e.g. [22], [30]).

In the interest of reproducible science, our proposed algorithm and datasets are publicly available[1].

REFERENCES

[1] 2022. Houston dataset. https://www.houstontx.gov/police/public_information.htm. Accessed: 2022-03-01.

[2] George Athanasopoulos, Roman A. Ahmed, and Rob J. Hyndman. 2009. Hierarchical forecasts for Australian domestic tourism. *International Journal of Forecasting* 25, 1 (2009), 146–166. https://doi.org/10.1016/j.ijforecast.2008.07.004

[3] George Athanasopoulos, Puwasala Gamakumara, Anastasios Panagiotelis, Rob J. Hyndman, and Mohamed Affan. 2020. *Hierarchical Forecasting.* Springer International Publishing, Cham, 689–719. https://doi.org/10.1007/978-3-030-31150-6_21

[4] George Athanasopoulos and Rob Hyndman. 2006. Modeling and forecasting Australian domestic tourism. *Monash University, Department of Econometrics and Business Statistics, Monash Econometrics and Business Statistics Working Papers* 29 (01 2006). https://doi.org/10.1016/j.tourman.2007.04.009

[5] Giorgio Corani, Alessio Benavoli, and Marco Zaffalon. 2021. Time Series Forecasting with Gaussian Processes Needs Priors. In *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track*, Yuxiao Dong, Nicolas Kourtellis, Barbara Hammer, and Jose A. Lozano (Eds.). Springer International Publishing, Cham, 103–117.

[6] Taco de Wolff, Alejandro Cuevas, and Felipe A. Tobar. 2020. Gaussian Process Imputation of Multiple Financial Series. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020), 8444–8448.

[7] Valentin Flunkert, David Salinas, and Jan Gasthaus. 2017. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. *CoRR* abs/1704.04110 (2017). arXiv:1704.04110 http://arxiv.org/abs/1704.04110

[8] Jacob R. Gardner, Geoff Pleiss, David Bindel, Kilian Q. Weinberger, and Andrew Gordon Wilson. 2018. GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 7587–7597.

[9] Simone Gitto, Carmela Di Mauro, Alessandro Ancarani, and Paolo Mancuso. 2021. Forecasting national and regional level intensive care unit bed demand during COVID-19: The case of Italy. *Plos one* 16, 2 (2021), e0247726.

[10] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. 2021. Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. *International Journal of Forecasting* 37, 1 (2021), 388–427. https://doi.org/10.1016/j.ijforecast.2020.06.008

[11] Robin John Hyndman and George Athanasopoulos. 2021. *Forecasting: Principles and Practice* (3rd ed.). OTexts, Australia.

[12] Rob J Hyndman, Roman A Ahmed, George Athanasopoulos, and Han Lin Shang. 2011. Optimal combination forecasts for hierarchical time series. *Computational Statistics & Data Analysis* 55, 9 (2011), 2579–2589.

[13] Rob J Hyndman and George Athanasopoulos. 2018. *Forecasting: principles and practice.* OTexts.

[14] Rob J Hyndman and Yeasmin Khandakar. 2008. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software* 26, 3 (2008), 1–22.

[15] Juan Pablo Karmy and Sebastián Maldonado. 2019. Hierarchical time series forecasting via Support Vector Regression in the European Travel Retail Industry. *Expert Systems with Applications* 137 (2019), 59–73. https://doi.org/10.1016/j.eswa.2019.06.060

[16] Shanika L Wickramasuriya, George Athanasopoulos, and Rob Hyndman. 2018. Optimal Forecast Reconciliation for Hierarchical and Grouped Time Series Through Trace Minimization. *J. Amer. Statist. Assoc.* (03 2018), 1–45. https://doi.org/10.1080/01621459.2018.1448825

[17] Haitao Liu, Jianfei Cai, and Yew-Soon Ong. 2018. Remarks on multi-output Gaussian process regression. *Knowledge-Based Systems* 144 (2018), 102–121. https://doi.org/10.1016/j.knosys.2017.12.034

[18] Alysha M. De Livera, Rob J. Hyndman, and Ralph D. Snyder. 2011. Forecasting Time Series With Complex Seasonal Patterns Using Exponential Smoothing. *J. Amer. Statist. Assoc.* 106, 496 (2011), 1513–1527. https://doi.org/10.1198/jasa.2011.tm09771 arXiv:https://doi.org/10.1198/jasa.2011.tm09771

[19] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2021. The M5 competition: Background, organization, and implementation. *International Journal of Forecasting* (2021). https://doi.org/10.1016/j.ijforecast.2021.07.007

[20] Gustavo Malkomes, Charles Schaff, and Roman Garnett. 2016. Bayesian optimization for automated model selection. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/3bbfdde8842a5c44a0323518eec97cbe-Paper.pdf

[21] Julie Novak, Scott McGarvie, and Beatriz Etchegaray Garcia. 2017. A Bayesian Model for Forecasting Hierarchically Structured Time Series. *International Symposium on Forecasting* (2017).

[22] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. 2005. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research* 6, 65 (2005), 1939–1959. http://jmlr.org/papers/v6/quinonero-candela05a.html

[23] Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning).* The MIT Press.

[24] Luis Roque, Luis Torgo, and Carlos Soares. 2021. Automatic Hierarchical Time-Series Forecasting Using Gaussian Processes. *Engineering Proceedings* 5, 1 (2021). https://doi.org/10.3390/engproc2021005049

[25] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. 2019. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in neural information processing systems* 32 (2019).

[26] Slawek Smyl. 2019. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting* 36 (07 2019). https://doi.org/10.1016/j.ijforecast.2019.03.017

[27] Evangelos Spiliotis, Mahdi Abolghasemi, Rob J Hyndman, Fotios Petropoulos, and Vassilios Assimakopoulos. 2020. Hierarchical forecast reconciliation with machine learning. *arXiv preprint arXiv:2006.02043* (2020).

[28] Souhaib Ben Taieb, James W. Taylor, and Rob J. Hyndman. 2021. Hierarchical Probabilistic Forecasting of Electricity Demand With Smart Meter Data. *J. Amer. Statist. Assoc.* 116, 533 (2021), 27–43. https://doi.org/10.1080/01621459.2020.1736081 arXiv:https://doi.org/10.1080/01621459.2020.1736081

[29] Sean J. Taylor and Benjamin Letham. 2018. Forecasting at Scale. *The American Statistician* 72, 1 (2018), 37–45. https://doi.org/10.1080/00031305.2017.1380080 arXiv:https://doi.org/10.1080/00031305.2017.1380080

[30] Andrew Wilson and Hannes Nickisch. 2015. Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP). In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis Bach and David Blei (Eds.). PMLR, Lille, France, 1775–1784. https://proceedings.mlr.press/v37/wilson15.html

[1] https://github.com/luisroque/hierarchical_gp_forecaster