Time series forecasting with Gaussian Processes needs priors

Giorgio Corani[1]✉[0000−0002−1541−8384], Alessio Benavoli[2][0000−0002−2522−7178], and Marco Zaffalon[1][0000−0001−8908−1502]

[1] Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
USI - SUPSI
Lugano, Switzerland
giorgio.corani{marco.zaffalon}@idsia.ch
[2] School of Computer Science and Statistics
Trinity College Dublin
Ireland
alessio.benavoli@tcd.ie

Abstract. Automatic forecasting is the task of receiving a time series and returning a forecast for the next time steps without any human intervention. Gaussian Processes (GPs) are a powerful tool for modeling time series, but so far there are no competitive approaches for automatic forecasting based on GPs. We propose practical solutions to two problems: automatic selection of the optimal kernel and reliable estimation of the hyperparameters. We propose a fixed composition of kernels, which contains the components needed to model most time series: linear trend, periodic patterns, and other flexible kernel for modeling the non-linear trend. Not all components are necessary to model each time series; during training the unnecessary components are automatically made irrelevant via automatic relevance determination (ARD). We moreover assign priors to the hyperparameters, in order to keep the inference within a plausible range; we design such priors through an empirical Bayes approach. We present results on many time series of different types; our GP model is more accurate than state-of-the-art time series models. Thanks to the priors, a single restart is enough the estimate the hyperparameters; hence the model is also fast to train.

1 Introduction

Automatic forecasting [14] is the task of receiving a time series and returning a probabilistic forecast for the next time steps without any human intervention. The algorithm should be both accurate and fast, in order to scale on a large number of time series,

Time series models such as exponential smoothing (*ets*, [12]) and automated arima procedures (*auto.arima* [14]) are strong baselines on monthly and quarterly time series, which contain limited number of samples. In these cases they generally outperform recurrent neural networks [11], which are also much more time-consuming to train.

Time series which are sampled at higher frequency generally contain multiple seasonal patterns. For instance, a time series of hourly data typically contains a daily and a weekly seasonal pattern. This type of time series can be forecast with models such as *tbats* [5] and *Prophet* [27].

Gaussian Processes (GPs) [21] are a powerful tool for modeling correlated observations, including time series. The GP provides a prior over functions, which captures prior beliefs about the function behavior, such as smoothness or periodicity. Given some observations, the prior is updated to form the posterior distribution over the functions. Dealing with Gaussian noise, this posterior distribution is again a GP. The posterior GP is used to predict the value of the function in points which have yet to be sampled; this prediction is accompanied by a principled quantification of the uncertainty. GPs have been used for the analysis of astronomical time series (see [7] and the references therein), forecasting of electric load [17] and analysis of correlated and irregularly-sampled time series [22].

Within a GP model, the kernel determines which functions are used for curve fitting. Complex functions can be obtained by summing or multiplying basic kernels; this is called *kernel composition.* In some cases the composition can be based on physical considerations [7] or personal expertise [17]. However algorithms which automatically optimize the kernel composition [6, 19, 15] do not scale, given the need for training a large number of competing GP models, each with cubic complexity. Moreover, there is no result showing that this strategy can forecast as accurately as the best time series models.

Summing up, there are currently no competitive approaches for automatic forecasting based on Gaussian Processes. In this paper we fill this gap, proposing a GP model which is accurate, fast to train and suitable for different types of time series.

We propose a kernel composition which contains useful components for modeling time series: linear trend, periodic patterns, and other flexible kernel for modeling the non-linear trend. We keep this composition fixed, thus avoiding kernel search. When dealing with a specific time series, some components might be unnecessary; during training they are made automatically irrelevant by *automatic relevance determination* (ARD) [18]. Indeed, ARD yields automatic feature selection for GPs.

We then consider how to reliably estimate the hyperparameters even on short time series. We keep their inference within a reasonable range by assigning priors to them. We define the parameters of such priors by means of a Bayesian hierarchical model trained on a separate subset of time series.

Extensive results show that our model is very accurate and versatile. It generally outperforms the state-of-the-art competitors on monthly and quarterly time series; moreover, it can be easily extended to model time series with double seasonality. Also in this case, it compares favorably to specialized time series models. A single restart is enough to sensibly estimate the hyperparameters; hence the model is also fast to train.

The paper is organized as follows: in Sec.2 we introduce GPs; in Sec.2.2 we present our kernel composition and the definition of the priors; in Sec. 3 we present the experiments.

2 Gaussian processes

We cast time-series modeling as a regression problem:

$$y = f(\mathbf{x}) + v, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^p$, $f : \mathbb{R}^p \to \mathbb{R}$ and $v \sim N(0, s_v^2)$ is the noise. We assume a Gaussian Process (GP) as a prior distribution about function f:

$$f \sim GP(0, k_{\boldsymbol{\theta}}),$$

where $k_{\boldsymbol{\theta}}$ denotes the kernel with hyperparameters $\boldsymbol{\theta}$. It is common to adopt the zero function as a mean function, since a priori we do not know whether at any point the trend will be below or above the average [22].

The kernel defines the covariance between the value of the function in different locations: $Cov(f(\mathbf{x}), f(\mathbf{x}^*)) = k_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}^*)$, $k_{\boldsymbol{\theta}} : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}^+$ and thus it determines which functions are likely under the GP prior.

The most common kernel is the *squared exponential*, also referred to as radial basis funcion (RBF):

$$\text{RBF}: \; k_{\boldsymbol{\theta}}(x_1, x_2) = s_r^2 \exp\left(-\frac{(x_1 - x_2)^2}{2\ell_r^2}\right),$$

whose hyperparameters are the variance s_r^2 and the lengthscale ℓ_r. Longer lengthscales yields smoother functions and shorter lengthscales yields wigglier functions. A limit of the RBF kernel is that, once conditioned on the training data, it does not extrapolate more than ℓ units away from the observations.

The periodic (PER) kernels yields periodic functions which repeat themselves exactly. Such function correspond to the sum of infinite Fourier terms [26, 4] and hence the PER kernel can represent any periodic function. It is defined as:

$$\text{PER: } k_{\boldsymbol{\theta}}(x_1, x_2) = s_p^2 \exp\left(-\frac{(2\sin^2(\pi|x_1 - x_2|/p_e)}{\ell_p^2}\right),$$

where ℓ_p^2 controls the wiggliness of the functions, p_e denotes the period and s_p^2 the variance.

Notice that in general, when the lengthscale of a kernel tends to infinity, or its variance tends to zero, the kernel yields functions that vary less and less as a function of x.

The linear kernel, which yields linear functions, is:

$$\text{LIN}: k_{\boldsymbol{\theta}}(x_1, x_2) = s_b^2 + s_l^2 x_1 x_2,$$

A GP with LIN kernel is equivalent [21] to a Bayesian linear regression.

The white noise (WN) kernel, which is used to represent the noise of the regression, is:

$$\text{WN} : k_{\boldsymbol{\theta}}(x_1, x_2) = s_v^2 \delta_{x_1,x_2}.$$

The above expressions are valid for $p = 1$, which is the case of a univariate time series; see [21] for the case $p>1$ and further kernels.

2.1 Kernel compositions

Positive definite kernels (i.e., those which define valid covariance functions) are closed under addition and multiplication [21]. Hence, complex functions can be modeled by adding or multiplying simpler kernels; this is called composition.

There are algorithms which iteratively train and compare GPs equipped with different kernel compositions [6, 19], but they are characterized by large computational complexity. Even if recent works have made the procedures more scalable [15, 28], they are still not comparable to lighting-fast time series model.

The *spectral mixture* kernel [30] allows the GP to fit complex functions without kernel search. It is defined as the sum of Q components, where the i-th component is:

$$\text{SM}_i : k_{\boldsymbol{\theta}}(x_1, x_2) = s_{m_i}^2 \exp\left(-\frac{(x_1 - x_2)^2}{2\ell_{m_i}^2}\right) \cos\left(\frac{x_1 - x_2}{\tau_{m_i}}\right),$$

with hyperparameters are s_{m_i}, ℓ_{m_i} and τ_{m_i}. It also corresponds to the product of a RBF kernel and another kernel called cosine kernel. Estimating the hyperparameters of the SM kernel is however challenging: the marginal likelihood is highly multimodal and it is unclear how to initialize the optimization. In [31] Bayesian optimization is used for deciding the initialization at each restart. This is effective but requires quite a few restarts.

2.2 The composition

We propose the following kernel composition:

$$\text{K} = \text{PER} + \text{LIN} + \text{RBF} + \text{SM}_1 + \text{SM}_2, \tag{2}$$

which arguably contains the most important components for forecasting.

The periodic kernel (PER) models the seasonal pattern; for monthly and quarterly time series, we assume a period of one year and we set p_e=1. Time series with a double seasonality can be modeled by adding a second periodic kernel, as we do in Sec. 4.

The LIN kernel provides the linear trend. This is an important component: for instance, auto.arima [14] adds a linear trend (by applying first differences) to about 40% of the monthly time series of the M3 competition. The RBF and the two SM kernels are intended to model non-linear trends which might characterize the time series.

Automatic Relevance Determination Some components of the composition might be unnecessary when fitting a certain time series: for instance, a time series might show no seasonal pattern or no linear trend. This is automatically managed via *automatic relevance determination* (ARD) [18]. When fitting the hyperparameters, the unnecessary components are given long lengthscale and/or small variance; in this way they are made irrelevant within the curve being fitted.

2.3 Training strategy

Reliably estimating the hyperparameters of the GP can be challenging (see e.g. [31]), especially when dealing with small data sets such as monthly and quarterly time.

We keep the inference of the hyperparameters within a plausible range by assigning priors to them. Variances and lengthscales are non-negative parameters, to which we assign log-normal priors:

$$s_l^2, s_r^2, s_p^2, s_{m_1}^2, s_{m_2}^2, s_v^2 \sim \text{LogN}(\nu_s, \lambda_s) \tag{3}$$
$$\ell_r \sim \text{LogN}(\nu_r, \lambda_\ell) \tag{4}$$
$$\ell_p \sim \text{LogN}(\nu_p, \lambda_\ell) \tag{5}$$
$$\ell_{m_1} \sim \text{LogN}(\nu_{m_1}, \lambda_\ell) \tag{6}$$
$$\ell_{m_2} \sim \text{LogN}(\nu_{m_2}, \lambda_\ell) \tag{7}$$
$$\tau_{m_1} \sim \text{LogN}(\nu_{t_1}, \lambda_\ell) \tag{8}$$
$$\tau_{m_2} \sim \text{LogN}(\nu_{t_2}, \lambda_\ell), \tag{9}$$

where $\text{LogN}(\nu, \lambda)$ denotes the distribution with mean ν and variance λ.

According to Eq.(3), all components share the same prior on the variance. This assign to every component the same prior probability of being irrelevant, as a component can be made irrelevant by pushing its variance to zero. We assign moreover a shared variance λ_ℓ to all lengthscales, in order to simplify the numerical fitting of the hierarchical model described in the next section.

We manage time such that time increases of one unit when one year has passed. The lengthscales can be readily interpreted; for instance an RBF kernel with lengthscale of 1.5 years is able to forecast about 1.5 years in the future before reverting to the prior mean.

Hierarchical GP model To numerically define the priors (3)–(9), we adopt an empirical Bayes approach. We select a set of B time series and we fit a *hierarchical* GP model to extract distributional information about the hyperparameters. The hierarchical Bayes model allows learning different models from different related data sets [8, Chap. 5]. Example of hierarchical GP models, not related to time series, are given in [16] and [25].

We assume the hyperparameters of the different time series to be drawn from higher-level priors (*hyperprior*). For instance the lengthscales of the RBF kernel $(\ell_r^{(1)}, \ell_r^{(2)}, ..., \ell_r^{(B)})$ are all drawn from the same hyperprior.

The generative model for the j-th time series is hence:

$$
\begin{aligned}
& s_l^{2(j)}, s_r^{2(j)}, s_p^{2(j)}, s_{m_1}^{2(j)}, s_{m_2}^{2(j)}, s_v^{2(j)} \sim \mathrm{LogN}(\nu_s, \lambda_s) \\
& \ell_r^{(j)} \sim \mathrm{LogN}(\nu_r, \lambda_\ell) \\
& \ell_p^{(j)} \sim \mathrm{LogN}(\nu_p, \lambda_l), \\
& \ell_{m_1}^{(j)} \sim \mathrm{LogN}(\nu_{m_1}, \lambda_\ell) \\
& \ell_{m_2}^{(j)} \sim \mathrm{LogN}(\nu_{m_2}, \lambda_\ell), \\
& \tau_{m_1}^{(j)} \sim \mathrm{LogN}(\nu_{\tau_1}, \lambda_\ell) \\
& \tau_{m_2}^{(j)} \sim \mathrm{LogN}(\nu_{\tau_2}, \lambda_\ell), \\
& \mathbf{y}^{(j)} \sim N(0, K_{\boldsymbol{\theta}}^{(j)}(X^{(j)}, X^{(j)})),
\end{aligned}
$$

where K denotes our kernel composition, instantiated with the hyper-parameters of the j-th time series; $\boldsymbol{\theta}^{(j)}$ denotes the hyper-parameters of the j-th time series.

We assign weakly-informative priors to the ν, λ parameters:

$$\nu_s, \nu_p, \nu_r, \nu_{m_2} \sim N(0, 5) \tag{10}$$

$$\nu_{m_1} \sim N(-1.5, 5) \tag{11}$$

$$\lambda_s, \lambda_l \sim \mathrm{Gamma}(1, 1). \tag{12}$$

The lower prior mean for ν_{m_1} is helpful for differentiating the estimation of SM_1 and SM_2 towards shorter-term and longer-term trends respectively.

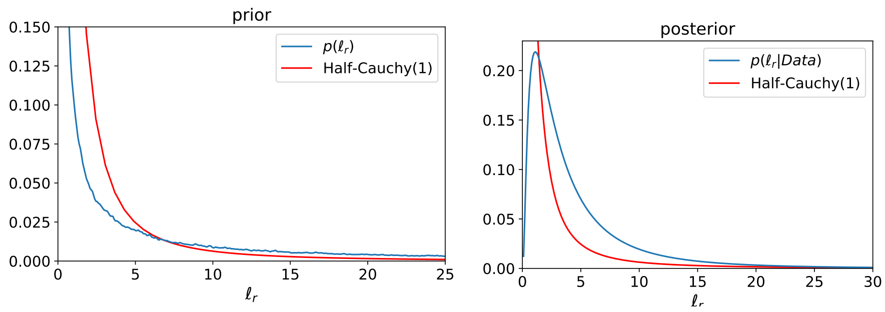

Fig. 1: Left: prior on ℓ_r induced by the hierarchical model. Right: posterior on ℓ_r estimated by the hierarchical model using 350 time-series. The Half-Cauchy distribution (with scale=1) is shown for comparison. In this paper we represent time such that, when one year has passed, x increases of one unit.

We implemented the model in PyMC3 [24]. We use automatic differentiation variational inference to approximate the posterior distribution of the ν's and λ's. We fit the hierarchical model on 350 monthly time series from the M3 competition. Before fitting the hierarchical model, we standardize each time series to have mean 0 and variance 1. Moreover, we manage time such that time increases of one unit when one year has passed.

The priors induced by the hierarchical model have fat tails. Consider for instance the prior induced on ℓ_r, which according to Eq.(10) – (12) is: $p(\ell_r) = \iint \text{LogN}(\ell_r; \nu_r, \lambda_\ell) N(\nu_r; 0, 5) \text{Gamma}(\lambda_l; 1, 1) d\nu_r d\lambda_l$. It is shown in the left plot of Fig.1, and its tails are actually fatter than those of the Half-Cauchy distribution.

Figure 1(right) shows instead the distribution on ℓ_r obtained using the posterior means of ν_r and λ_l, estimated by the hierarchical model. This yields a distribution on ℓ_r which we use as prior when fitting the GP. This prior has fat tails too, see the comparison with the half-Cauchy; nevertheless, it does inform the optimizer about the order of magnitude of ℓ_r. The median and the 95-th percentile of the prior of each hyperparameter are given in Tab.1.

parameter	median	95th	ν	λ
variance	0.2	1.2	-1.5	1.0
std_periodic	1.2	6.3	0.2	1.0
rbf	3.0	15.4	1.1	1.0
SM_1 (rbf)	0.5	2.5	-0.7	1.0
SM_1 (cos)	1.7	8.6	0.5	1.0
SM_2 (rbf)	3.0	15.4	1.1	1.0
SM_2 (cos)	5.0	25.8	1.6	1.0

Table 1: Quantiles on the hyperparameters implied by the lognormal priors and parameters (ν, λ) of the lognormal priors. By design, the λs are equal for all lengthscales.

The prior on the variance is coherent with the fact that we work with standardized time series, whose variance is one.

The priors over the lengthscales also yield plausible ranges, every component having a median lengthscale comprised between 0.5 and 3 years, with long tails arriving up to 25 years.

All the experiments of this paper are thus computed using the priors of Tab.1. To remove any danger of overfitting, we remove the 350 time series used to fit the hierarchical model from our experiments.

Further considerations In the jargon of time series, models which are fitted to a set of time series are referred to as *global* models, see for instance [20, 23]. The hierarchical model is a global model, as it jointly analyzes different time series. Global models can be more accurate than univariate models, if the time series are characterized by some common patterns. Yet, they are also more complicated to fit. In this paper we do not consider global models. We use the hierarchical model only for defining the priors on the hyperparameters of the GP.

2.4 MAP estimation

We estimate the hyperparameters by computing the maximum a-posteriori (MAP) estimate of $\boldsymbol{\theta}$, thus approximating the marginal of $\mathbf{f}^*$ with (14). We thus maximize w.r.t. $\boldsymbol{\theta}$ the joint marginal probability of $\mathbf{y}, \boldsymbol{\theta}$, which is the product of the prior $p(\boldsymbol{\theta})$ and the marginal likelihood [21, Ch.2]:

$$p(\mathbf{y}|X, \boldsymbol{\theta}) = N(\mathbf{y}; 0, K_{\boldsymbol{\theta}}(X, X)). \tag{13}$$

Using a single restart, MAP estimation is generally accomplished in less than a second (on a standard computer) on monthly and quarterly time series, yielding thus quick training times.

2.5 Forecasting

Based on the training data $X^T = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, $\mathbf{y} = [y_1, \ldots, y_n]^T$, and given m test inputs $(X^*)^T = [\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*]$, we wish to find the posterior distribution of $\mathbf{f}^* = [f(\mathbf{x}_1^*), \ldots, f(\mathbf{x}_m^*)]^T$.

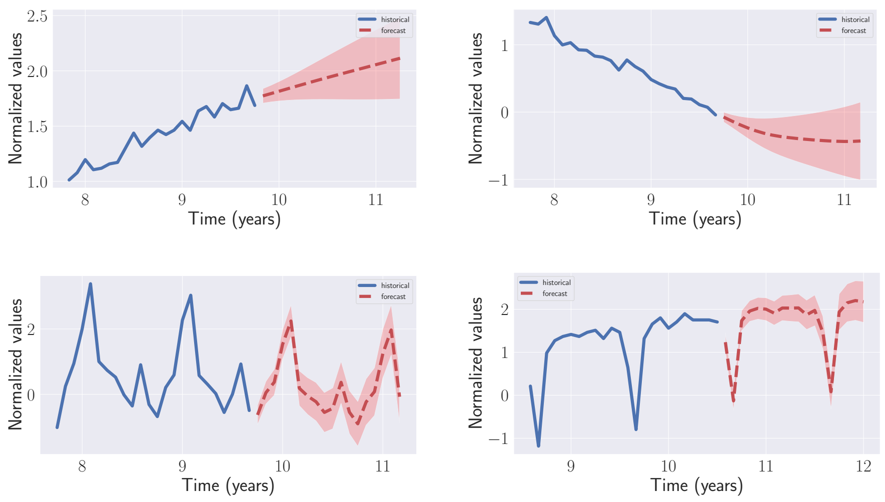

Fig. 2: Examples of GP forecasts on monthly time series, computed up to 18 months ahead.

From (1) and the properties of the Gaussian distribution, [3] the posterior distribution of $\mathbf{f}^*$ is [21, Sec. 2.2]:

$$p(\mathbf{f}^*|X^*, X, \mathbf{y}, \boldsymbol{\theta}) = N(\mathbf{f}^*; \hat{\boldsymbol{\mu}}_{\boldsymbol{\theta}}(X^*|X, \mathbf{y}), \hat{K}_{\boldsymbol{\theta}}(X^*, X^*|X)), \tag{14}$$

[3] In the paper, we incorporate the additive noise v into the kernel by adding a White noise kernel term.

with mean and covariance given by:

$$\begin{aligned}&\hat{\boldsymbol{\mu}}_{\boldsymbol{\theta}}(\mathbf{f}^*|X,\mathbf{y}) = K_{\boldsymbol{\theta}}(X^*,X)(K_{\boldsymbol{\theta}}(X,X))^{-1}\mathbf{y},\\ &\hat{K}_{\boldsymbol{\theta}}(X^*,X^*|X) = K_{\boldsymbol{\theta}}(X^*,X^*)\\ &-K_{\boldsymbol{\theta}}(X^*,X)(K_{\boldsymbol{\theta}}(X,X))^{-1}K_{\boldsymbol{\theta}}(X,X^*).\end{aligned} \tag{15}$$

Our kernel composition, trained using the proposed priors, yields sensible forecasts in very different contexts, as in Fig.2.

3 Experiments

We run experiments on the monthly and quarterly time series of the M1 and M3 competitions, available from the package Mcomp [13] for R. The original 1428 time series of the M3 competition drop to 1078 once we remove the 350 time series used to fit the hierarchical model. Overall we consider about 959 quarterly time series (203 from M1, 756 from M3) and 1695 monthly time series (617 from M1 and 1078 from M3). The test set of monthly time series contains 18 months; the test set of quarterly time series contains 8 quarters. We standardize each time series to have mean 0 and variance 1 on the training set.

	quarterly		monthly	
	M1	M3	M1	M3
number of time series	203	756	617	1078
median training length	40	44	66	115
Test set length	8	8	18	18

Table 2: Main characteristics of the M1 and M3 data sets.

We denote by GP our model trained *with* priors and by GP_0 our model trained *without* priors, i.e., by maximizing the marginal likelihood. We use a single restart when training both GP and GP_0; on these time series, which contain around 100 observations, the average training is generally less than one second on a standard laptop.

As competitors we consider *auto.arima* and *ets*, both available from the forecast package [12] for R. We tried also Prophet [27], but its accuracy was not competitive. We thus dropped it; we will consider it later in experiments with different types of time series.

Indicators

Let us denote by y_t and $\hat{y}_t$ the actual and the expected value of the time series at time t; by σ_t^2 the variance of the forecast at time t; by T the length of the

competition	freq	score	GP	*ets*	*arima*	GP$_0$
M1	monthly	MAE	**0.58**	0.59	0.62*	0.72*
M1	monthly	CRPS	**0.41**	0.45*	0.45*	0.53*
M1	monthly	LL	**-1.13**	-1.27*	-1.28*	-1.67*
M1	quarterly	MAE	**0.57**	0.63*	0.62*	0.75*
M1	quarterly	CRPS	**0.39**	0.47*	0.44*	0.59*
M1	quarterly	LL	**-1.07**	-1.41*	-1.44*	-2.66*
M3	monthly	MAE	**0.48**	0.51*	0.51*	0.59*
M3	monthly	CRPS	**0.35**	0.38*	0.37*	0.42*
M3	monthly	LL	**-1.01**	-1.05*	-1.06*	-1.23*
M3	quarterly	MAE	0.42	**0.41**	**0.41**	0.54*
M3	quarterly	CRPS	**0.30**	0.31	0.31	0.40*
M3	quarterly	LL	**-0.85**	-0.90*	-0.94*	-1.61*

Table 3: Median results on M1 and M3 time series.The best-performing model is boldfaced. Starred results correspond to the GP yielding a significant improvement over the competitor (95%, Bayesian signed-rank test).

test set. The mean absolute error (MAE) on the test set is:

$$\text{MAE} = \sum_{t=1}^{T} |y_t - \hat{y}_t|$$

The continuous-ranked probability score (CRPS) [9] is a proper scoring rule which generalizes MAE to probabilistic forecasts. Let us denote by F_t the cumulative predictive distribution at time t and and by z the variable over which we integrate. The CRPS is:

$$\text{CRPS}(F_t, y_t) = -\int_{-\infty}^{\infty} (F_t(z) - \mathbb{1}\{z \geq y_t\})^2 \mathrm{d}z.$$

The log-likelihood of the test set (LL) is defined as:

$$\text{LL} = \frac{1}{T}\left(-\frac{1}{2}\sum_{t=1}^{T}\log(2\pi\sigma_t^2) - \frac{1}{2\sigma_t^2}\sum_{t=1}^{T}(\, y_t - \hat{y}_t \,)^2\right)$$

MAE and CRPS are loss functions, hence the lower the better; instead for LL, the higher the better.

In Tab. 3 we report the median results for each indicator and each data set. In each setting the GP yields the best median on almost all indicators. However GP_0 is instead clearly outperformed by both ets and auto.arima. Hence, our GP model needs priors on the hyperparameters to produce highly accurate forecasts.

We then check the significance of the differences on the medians via the Bayesian signed-rank test [3], which is a Bayesian counterpart of the Wilcoxon signed-rank test. It returns posterior probabilities instead of the p-value. An advantage of this test over the frequentist one is that we can set a region of practical equivalence (rope) between the two algorithms being compared. When comparing algorithms A and B, the test returns three posterior probabilities: the probability of the two algorithms being practically equivalent, i.e, the probability of the median difference belonging to the rope; the probability of A being significantly better than B, and vice versa. As already pointed out, better means lower MAE, lower CRPS, higher LL. We considered a rope of ± 0.01 on each indicator, similarly to [2]. We consider as significant the differences in which the probability of an algorithm being better than another is at least 95%. The improvement yielded by the GP over the competitors are significant in most cases; see the starred entries in Tab.3. When the median of some competitor was better than that of the GP, the difference was not statistically significant.

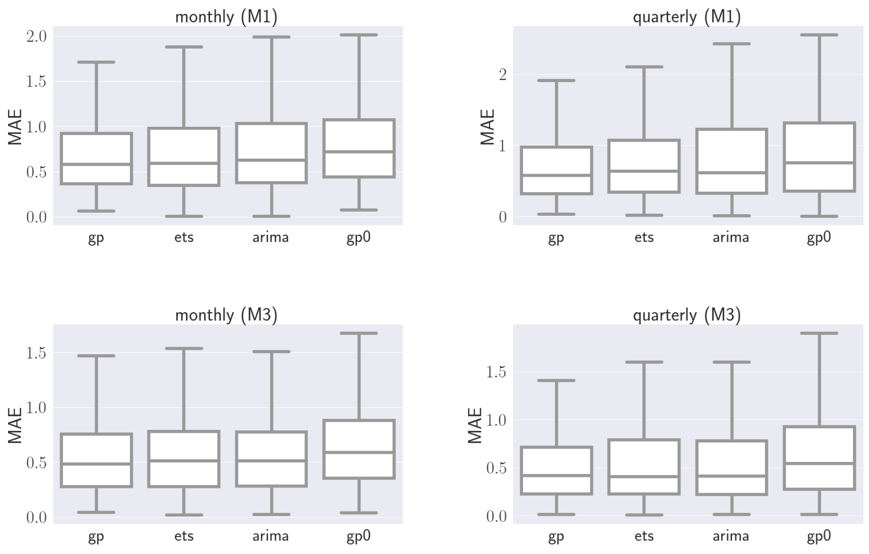

Fig. 3: Distribution of MAE on the monthly and quarterly time series of the M1 and M3 competition.

The improvement is not only on the medians, but it also involve the distribution across time series, as shown by the boxplots of MAE (Fig 3). Similar results hold also for the distribution of the other indicators, which we do not show for reasons of space.

4 Dealing with multiple seasonalities

We then test the versatility of our GP model, by considering time series with multiple seasonalities. We consider the electricity data set[4], which contains 370 time series regarding electricity demand for different Portuguese households.

Each time series covers the period January 2011 - September 2015 with a sampling frequency of 15 mins, totaling 140k points. To have a more manageable data set we aggregate the data to 6-hours steps. We consider a training set containing of 250 days (1000 points) and a test set of 10.5 days (42 steps).

Such time series have a daily and a weekly seasonal pattern. This can be addressed by approaches which model seasonality using Fourier terms. For instance TBATS [5] introduces Fourier terms within an exponential smoothing state space model. Prophet [27] is a decomposable Bayesian time series model, whose final forecast is the sum of different functions, which account for different effects. The seasonality function is is modeled by Fourier terms. Prophet however is not very effective on time series with simpler seasonality, such as monthly and quarterly time series, as we have already seen.

We adapt the kernel composition by adding a second periodic kernel:

$$\mathrm{K} = \mathrm{PER}_w + \mathrm{PER}_d + \mathrm{LIN} + \mathrm{RBF} + \mathrm{SM}_1 + \mathrm{SM}_2,$$

where PER_w and PER_d represents respectively the weekly and the daily pattern. We thus set the period of PER_w to $\frac{1}{52.18}$ and the period of PER_d to $\frac{1}{(365.25)}$ As in previous experiments, we standardize time series and one year corresponds to time increasing of one unit. We can keep unchanged the priors.

	GP	Tbats	Prophet
MAE	**0.26**	0.30^*	0.29^*
CRPS	**0.19**	0.23^*	0.21^*
LL	**-0.37**	-0.60^*	-0.49^*

Table 4: Median results on the electricy data sets (370 time series). Starred results imply statistical significance (Bayesian signed rank test).

In table 4 we report the median results across the 370 time series; the GP delivers the best performance on all indicators. The GP model compares favorably to the competitors also as for the distribution of the MAE (Fig. 4) across time series.

[4] `https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014`

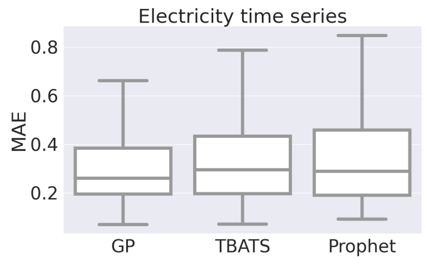

Fig. 4: Boxplot of MAE on the 370 electricity time series.

5 Code and replicability

We make available our code and the data of M1 and M3 time series at the link: `https://github.com/IDSIA/gpforecasting/`. Our implementation is based on the GPy library [10].

6 Conclusions

As far as we know, these are the best results obtained so far in automatic forecasting with Gaussian processes. Our model is competitive with the best time series models on different types of time series: monthly, quarterly and time series with multiple seasonalities.

The model is fast to train, at least on time series containing less than 500 data points. Recent computational advances with GPs in time series [26, 1] could allow the application of our methodology also to time series thousands of observations.

Our GP model yields both good point forecast and a reliable quantification of the uncertainty, as shown by the CRPS and LL indicators. It is thus an interesting candidate for problems of hierarchical forecasting [29], which require forecasts with a sound quantification of the uncertainty.

Due to the general properties of the GP, the model can be learned also from irregularly sampled or incomplete time series.

7 Acknowledgments

Work for this paper has been partially supported by the Swiss NSF grant n. 167199 of the funding scheme *NRP 75 Big Data*.

We thank David Huber, for polishing our initial implementation and helping with the experiments.

References

1. Ambikasaran, S., Foreman-Mackey, D., Greengard, L., Hogg, D.W., O'Neil, M.: Fast direct methods for gaussian processes. IEEE transactions on pattern analysis and machine intelligence **38**(2), 252–265 (2015)

2. Benavoli, A., Corani, G., Demšar, J., Zaffalon, M.: Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis. The Journal of Machine Learning Research **18**(1), 2653–2688 (2017)
3. Benavoli, A., Corani, G., Mangili, F., Zaffalon, M., Ruggeri, F.: A Bayesian Wilcoxon signed-rank test based on the Dirichlet process. In: Proc. International Conference on Machine Learning. pp. 1026–1034 (2014)
4. Benavoli, A., Zaffalon, M.: State Space representation of non-stationary Gaussian processes. arXiv preprint arXiv:1601.01544 (2016)
5. De Livera, A.M., Hyndman, R.J., Snyder, R.D.: Forecasting time series with complex seasonal patterns using exponential smoothing. Journal of the American Statistical Association **106**(496), 1513–1527 (2011)
6. Duvenaud, D., Lloyd, J., Grosse, R., Tenenbaum, J., Zoubin, G.: Structure discovery in nonparametric regression through compositional kernel search. In: Proc. International Conference on Machine Learning. pp. 1166–1174 (2013)
7. Foreman-Mackey, D., Agol, E., Ambikasaran, S., Angus, R.: Fast and scalable Gaussian process modeling with applications to astronomical time series. The Astronomical Journal **154**(6), 220 (2017)
8. Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A., Rubin, D.: Bayesian Data Analysis (3rd ed.). Chapman and Hall/CRC (2013)
9. Gneiting, T., Raftery, A.E.: Strictly proper scoring rules, prediction, and estimation. Journal of the American statistical Association **102**(477), 359–378 (2007)
10. GPy: GPy: A Gaussian process framework in Python. `http://github.com/SheffieldML/GPy` (since 2012)
11. Hewamalage, H., Bergmeir, C., Bandara, K.: Recurrent neural networks for time series forecasting: Current status and future directions. International Journal of Forecasting **37**(1), 388–427 (2021)
12. Hyndman, R.J. & Athanasopoulos, G.: Forecasting: principles and practice, 2nd edition,. OTexts: Melbourne, Australia (2018), `OTexts.com/fpp2`
13. Hyndman, R.: Mcomp: Data from the M-Competitions (2018), `https://CRAN.R-project.org/package=Mcomp`, r package version 2.8
14. Hyndman, R.J., Khandakar, Y.: Automatic time series forecasting: the forecast package for R. Journal of Statistical Software **26**(3), 1–22 (2008), `http://www.jstatsoft.org/article/view/v027i03`
15. Kim, H., Teh, Y.W.: Scaling up the Automatic Statistician: Scalable structure discovery using Gaussian processes. In: International Conference on Artificial Intelligence and Statistics. pp. 575–584. PMLR (2018)
16. Lawrence, N.D., Platt, J.C.: Learning to learn with the informative vector machine. In: Proceedings of the twenty-first international conference on Machine learning. p. 65 (2004)
17. Lloyd, J.R.: GEFCom2012 hierarchical load forecasting: Gradient boosting machines and Gaussian processes. International Journal of Forecasting **30**(2), 369–374 (2014)
18. MacKay, D.J.: Introduction to Gaussian processes. NATO ASI Series F Computer and Systems Sciences **168**, 133–166 (1998)
19. Malkomes, G., Schaff, C., Garnett, R.: Bayesian optimization for automated model selection. In: Hutter, F., Kotthoff, L., Vanschoren, J. (eds.) Proceedings of the Workshop on Automatic Machine Learning. vol. 64, pp. 41–47 (2016)
20. Montero-Manso, P., Athanasopoulos, G., Hyndman, R.J., Talagala, T.S.: Fforma: Feature-based forecast model averaging. International Journal of Forecasting **36**(1), 86–92 (2020)

21. Rasmussen, C., Williams, C.: Gaussian processes for machine learning. Gaussian Processes for Machine Learning (2006)
22. Roberts, S., Osborne, M., Ebden, M., Reece, S., Gibson, N., Aigrain, S.: Gaussian processes for time-series modelling. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **371**(1984), 20110550 (2013)
23. Salinas, D., Flunkert, V., Gasthaus, J., Januschowski, T.: Deepar: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting **36**(3), 1181–1191 (2020)
24. Salvatier, J., Wiecki, T.V., Fonnesbeck, C.: Probabilistic programming in Python using PyMC3. PeerJ Computer Science **2**, e55 (2016)
25. Schwaighofer, A., Tresp, V., Yu, K.: Learning gaussian process kernels via hierarchical bayes. In: Advances in neural information processing systems. pp. 1209–1216 (2005)
26. Solin, A., Särkkä, S.: Explicit link between periodic covariance functions and state space models. In: Artificial Intelligence and Statistics. pp. 904–912. PMLR (2014)
27. Taylor, S.J., Letham, B.: Forecasting at scale. The American Statistician **72**(1), 37–45 (2018)
28. Teng, T., Chen, J., Zhang, Y., Low, B.K.H.: Scalable variational bayesian kernel selection for sparse gaussian process regression. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 5997–6004 (2020)
29. Wickramasuriya, S.L., Athanasopoulos, G., Hyndman, R.J.: Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. Journal of the American Statistical Association **114**(526), 804–819 (2019)
30. Wilson, A., Adams, R.: Gaussian process kernels for pattern discovery and extrapolation. In: Proc. International Conference on Machine Learning. pp. 1067–1075 (2013)
31. Wu, J., Poloczek, M., Wilson, A.G., Frazier, P.: Bayesian optimization with gradients. In: Advances in Neural Information Processing Systems. pp. 5267–5278 (2017)