Predicting Electricity Consumption with Random Walks on Gaussian Processes

Anonymous Author(s)
Affiliation
Address
email

Abstract

We consider time-series forecasting problems where data is scarce, difficult to gather, or induces a prohibitive computational cost. As a first attempt, we focus on short-term electricity consumption in France, which is of strategic importance for energy suppliers and public stakeholders. The complexity of this problem and the many levels of geospatial granularity motivate the use of an ensemble of Gaussian Processes (GPs). Whilst GPs are remarkable predictors, they are computationally expensive to train, which calls for a frugal few-shot learning approach. By taking into account performance on GPs trained on a dataset and designing a random walk on these, we mitigate the training cost of our entire Bayesian decision-making procedure. We introduce our algorithm called DOMINO (ranDOM walk on gaussIaN prOcesses) and present numerical experiments to support its merits.

1 Introduction

Forecasting time series is at the centre of machine learning (ML). We focus on problem settings where we might have sparse data, limited compute capacity or unseen scenarios. We instantiate this problem in the setting of short-term electricity consumption prediction, and focus on doing this at the scale of France. For energy suppliers and public stakeholders, the necessity is to be able to predict consumption even in extreme events, such as a heat or cold wave, which can lead to large variations in consumption, possibly at a fairly high granularity.

These scenarios are also characterised by the fact that they do not have as much data as more classical time series forecasting scenarios such as stock price modelling, and therefore deep learning methods which have been a huge part of the recent artificial intelligence (AI) boom might not be as appropriate as they are unable to forecast as efficiently when there is little training data.

For time series forecasting, GPs are well-adapted as they natively quantify uncertainty. However, their performance is indexed on the size of the training data and they are computationally expensive to train. Recently, Few-Shot Learning (FSL) for time series prediction has gained attention both from theoretical [Iwata and Kumagai, 2020] and applied perspectives [Xu et al., 2024], to help mitigate the costs of training. This work is the start of a series of analyses on French regional short-term electricity consumption. We take a FSL approach to training GPs, using a set of GPs trained on synthetic data in a first instance, with the natural next step being with actual electricity consumption data, available at `https://www.rte-france.com/en/eco2mix/download-indicators`.

We describe our algorithm, called DOMINO, in Section 2 and in Section 3, we illustrate its performance on a synthetic dataset. We briefly discuss preliminary results and lay down ideas for future works, with the aim to use this present work as a stepping stone towards broader time series forecasting problems where data is scarce, difficult to gather or induces a prohibitive cost.

Submitted to Workshop on Bayesian Decision-making and Uncertainty, 38th Conference on Neural Information Processing Systems (BDU at NeurIPS 2024). Do not distribute.

2 Methodology

Due to the stochastic nature of the underlying phenomenon and its periodicity, we use GPs which quantify uncertainty and handle unseen scenarios. We refer to Rasmussen and Williams [2006] for a complete reference on GPs.

Notation. We adopt the following conventional notations to define our problem statement. We model I *time series*, where $i \in \{1, 2, \dots, I\}$ is an indicator of the time series used as a subscript. The time series all have $N \in \mathbb{N}$ entries. The inputs are vectors $\mathbf{t_i} = [t_{i1}, t_{i2}, \dots, t_{iN}] \in \mathbb{R}^N, \forall i \in \{i\}_{i=1}^I$. The outputs are $\mathbf{y_i} = [y_{i1}, y_{i2}, \dots, y_{iN}] = y_i(\mathbf{t}) = [y_i(t_1), y_i(t_2), \dots, y_i(t_N)] \in \mathbb{R}^N$.

For each time series $i \in \{i\}_{i=1}^I$, we look for the function $f_i : \mathbf{t} \mapsto \mathbf{y_i} + \epsilon_i(\mathbf{t})$, where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ with $\sigma \in \mathbb{R}^N$. To share knowledge between time series, we leverage an algorithm whose tasks share a common mean.

2.1 Existing work: the MAGMA algorithm

The Multi-tAsk GPs with common MeAn (MAGMA) algorithm [introduced by Leroy et al., 2022] shares knowledge between time series which are modelled by the same GP - this predicts unseen time series by using the common mean, which saves training resources and enables a more accurate prediction. The model is trained with an EM algorithm. Starting from an initialisation, in turn the hyperparameters given a distribution (E-step) and the distribution given the hyperparameters (M-step) are optimised. The optimisation can be made from the values learned at the previous step.

We refer to Leroy et al. [2022] for full explanations of the algorithm and its proof. The MAGMA algorithm is implemented in the `MAGMAClustR` package (`https://arthurleroy.github.io/MagmaClustR/`). Whilst MAGMA is a powerful predictor, it is computationally expensive. In sparse computational resource settings, this limits its applications. We look to sample the GPs output by the algorithm to transfer knowledge to unseen time series with a more frugal approach, *i.e.*, by leveraging much less data.

2.2 Our Algorithm: a Random Walk on Gaussian Processes (DOMINO)

The DOMINO (standing for ranDOM walk on GaussIaN PrOcess) algorithm takes the output of the MAGMA model and samples these GPs. A random walk switches between the sampled time series at each time point, following a probability for each time series. After each walk, the random walk's performance with respect to each sampled time series is evaluated. Given this, and how often each time series has been sampled during the random walk, the performance and weights of the time series are updated. Until a maximum number of epochs is reached, until the random walk and sampled time series have a Kullback-Leibler (KL) divergence which is lower than a chosen δ, or until a maximum number of samples over the δ threshold have a difference to the threshold with a lower standard deviation than the standard deviation of the in lying time series points and δ, the walk is repeated with the updated performances serving as a new probability at each epoch.

Notation. We superscript w the current epoch to identify the information which is specific to it. The performance of the sampled time series at the current epoch is p^w. The random walk for the current epoch is $\mathbf{y}^w \in \mathbb{R}^N$. The time series sampled at each stage of the random walk for the current epoch is $\mathbf{z}^w \in \mathbb{R}^N$. Let $P(f_i)$ be the performance of the function f_i.

We establish the following condition to end the random walk.

Condition 1 *Let δ denote a tolerance parameter (the largest divergence between any sample and the* DOMINO *at any point), and let W be the largest number of epochs possible such that at least one of the three following statement holds true.*

1. *$\forall i \in I$,* $\mathrm{KL}(\text{DOMINO} || \mathcal{GP}_i) \leq \delta$*: all time series' KL divergence from the* DOMINO *is under δ;*
2. *For $a \in \mathbb{N}$ where $a \ll N$, there are a points of all the time series whose values difference to the δ threshold have a standard deviation lower than that of the in-lying points and their difference to the δ threshold;*

3. $w \geq W \in \mathbb{N}$: *the maximum number of epochs set is reached.*

Initialisation. Given a GP, sample the I samples to walk on, set the maximum number of epochs W, choose the KL divergence threshold δ, the maximal number of outliers a and $\lambda \in \mathbb{R}_0$ the regularisation constant. Without prior knowledge, the starting performance of each sample is $p^0 = \{p_i^0\}_{i=1}^I = \frac{1}{I}$.

Algorithm 1 DOMINO.

while Condition 1 is False **do**:
  Initialise g with a random draw from $\mathcal{I} = \{1, 2, \ldots, I\}$ whose hyperparameters follow a categorical distribution with hyperparameters given by: $\left\{\frac{e^{\lambda * i}}{\sum_{k=1}^I e^{\lambda * k}}\right\}_{i=1}^I$
  Set $n = g$ and use the sample from the time series drawn above.
  Set $\mathbf{y}^w(t_1) = \mathbf{f_g}(t_1)$ the first time step using the drawn sample and $\mathbf{z^w}(t_1) = g = n$ the randomly drawn sample time series for the first step of the random walk.
  for $t_n \in t_2, \ldots, t_N$, with a probability of $p(f_i(t_n)) = \frac{e^{\lambda * i}}{\sum_{k=1}^I e^{\lambda * k}} \forall i \in \mathcal{I}$: **do**
    Set $\mathbf{y}^w(t_n) = \mathbf{f_i}(t_n)$ the next step in the random walk;
    Set $\mathbf{z}^w(t_n) = i$ from $\mathbf{y_i}$ the time series for the step.
  end for
  for each time step $\mathbf{y}^w = y^w(\mathbf{t_n}) = \{y^w(\mathbf{t_1}), y^w(\mathbf{t_2}), \ldots, y^w(\mathbf{t_N})\}$ of the random walk, evaluate it against the I time series: **do**
    Let $\mathbf{M}^w = \{P(f_1, y_1), \ldots, \{P(f_I, y_I)\}\} = \{P(f_i, y_i)\}_{i=1}^I$ be the performances of the time series.
    Let $m^w = \frac{1}{I} = \sum_{i=1}^I M_i^w$ be the average of all performances across time series.
    Update $\mathbf{p}^w = \{p_i^w\}_{i=1}^w$ with $\mathbf{z_i}$: set

$$\mathbf{p_i}^w = \frac{\prod_{a=1}^{w-1} \exp\left(\frac{1}{2} - \frac{|i \in z_a(\mathbf{t_n})|}{I}\right) * \mathbf{M}_i^w}{\sum_{i=1}^I \left(\prod_{a=1}^{w-1} \exp\left(\frac{1}{2} - \frac{|i \in z_a(\mathbf{t_n})|}{I}\right) * \mathbf{M}_i^w\right)}.$$

    Store the $\mathbf{p_w}$ performance values, $\mathbf{m^w}$ average performance across all time series, $\mathbf{M^w}$ time series performances, $\mathbf{z^w}$ steps from the random walk and $\mathbf{y}^w$ values from the random walk. for the w^{th} epoch.
  end for
  if Condition 1 is not False **then**
    $w = w + 1$
  end if
end while

3 Experiments

Datasets. With 10 years of regional half-hourly electricity consumption data and the current consumption levels for the regions, we can aim to predict the short-term electricity needs of France over the next three hours. From a set of time series with similar characteristics, the goal is to predict a hold-out time series from GPs trained on the rest of the set. As a proof of concept, in the present paper we experiment on artificially generated periodic data. The synthetic data is generated with trend, periodic and noise components, using the `mockseries` Python package `https://mockseries.catheu.tech/` which allows us to create time series indexed to a chosen time frame, and with constraints.

Evaluation. We evaluate using the Median Absolute Error (MAE) as it is robust to outliers whilst giving an error in the same unit as the output. Given y_i the i^{th} sample and $\hat{y}_i$ its predicted value, the MAE is calculated by: $\mathrm{MAE}(y, \hat{y}) = \mathrm{median}\left(|y_i - \hat{y}_i|_{i=1}^I\right)$. We use MAGMA to train GPs on the data. We evaluate the model, as well as the DOMINO trained on the model's samples. The test data is evaluated following the protocol in Appendix A.

Ablation studies. We study the impact of hyperparameter adjustment in DOMINO. We experiment with time series length to find the optimal number of points per time series for MAGMA and DOMINO,

and we establish the relative performance of both algorithms. Our code and data are available online at `https://anonymous.4open.science/r/domino-effect-155D/`.

Results. We gather in Table 1 the results of performance when varying the time series length for MAGMA and DOMINO; the cross-validation results are in Table 2. DOMINO consistently outperforms MAGMA by a significant margin.

Table 1: Average (std) MAGMA and DOMINO MAE on 10 runs.

Length N	**MAGMA**	**DOMINO**
50	8.089 (0.015)	4.405 (1.006)
100	6.059 (0.085)	4.526 (0.932)
150	33.454 (0.108)	3.618 (0.559)
200	48.108 (0.113)	**3.524** (0.386)
250	5.991 (0.029)	4.511 (0.336)

Discussion and limitations. We have used a uniform probability distribution on the sampled time series but can extend the work to a scenario with prior knowledge and therefore a known probability distribution across the samples at initialisation.

DOMINO dramatically improves on the MAGMA algorithm. A natural next step is to conduct a similar study on MAGMAClust [Leroy et al., 2023], a generalisation of MAGMA which learns cluster-specific means and infers clusters whilst learning the common means.

MAGMA can handle covariates. This is a natural next step for the DOMINO algorithm. We have worked with inputs on a regular grid, as there is a very regular data stream for electricity consumption and we worked with similar data. This approach will be limited where there is an irregular input and calls for an adaptation of DOMINO.

Table 2: Average (std) MAGMA and DOMINO MAE at cross-validation on 10 runs.

Length N	**MAGMA**	**DOMINO**
50	8.91 (0.319)	4.114 (0.419)
100	6.624 (0.363)	5.058 (0.333)
150	33.973 (0.466)	4.708 (0.334)
200	48.412 (0.326)	9.679 (0.079)
250	56.215 (0.342)	4.608 (0.226)

Hyperparameters. DOMINO is controlled by hyperparameters which determine the optimal maximal number of training epochs (30), the percentage of the minimum-maximum range of the output which is an acceptable KL divergence δ between the training time series and the random walk learned (5%), the maximal number of points (all outputs, all time series combined) which can be over the δ value (3%) and the λ regularisation parameter which smooths the weights when calculating the probability of each time series for the next sample (0.5). The full set of ablation studies are detailed in Appendix B.

Conclusion. In this work, we have used the MAGMA algorithm to predict short-term electricity usage based on GPs with common means. We then performed a random walk on samples of these GPs, which is iterated until there is a low divergence between the sampled time series and the points of the random walk. Our experiments show that this approach, called DOMINO, yields superior predictive results on a synthetic dataset. This is very promising to tackle similar problems in sparse data settings, with less computational resources, or heavy data settings, paving the way to more frugal probabilistic settings.

References

Tomoharu Iwata and Atsutoshi Kumagai. Few-shot learning for time-series forecasting. *arXiv preprint arXiv:2009.14379*, 2020.

Jiangjiao Xu, Ke Li, and Dongdong Li. An automated few-shot learning for time series forecasting in smart grid under data scarcity. *IEEE Transactions on Artificial Intelligence*, 2024.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 2006. ISBN 978-0-262-18253-9. OCLC: ocm61285753.

Arthur Leroy, Pierre Latouche, Benjamin Guedj, and Servane Gey. MAGMA: inference and prediction using multi-task Gaussian processes with common mean. *Machine Learning*, 111(5):1821–1849, May 2022. ISSN 1573-0565. doi: 10.1007/s10994-022-06172-1. URL `https://doi.org/10.1007/s10994-022-06172-1`.

Arthur Leroy, Pierre Latouche, Benjamin Guedj, and Servane Gey. Cluster-Specific Predictions with Multi-Task Gaussian Processes. *Journal of Machine Learning Research*, 24(5):1–49, 2023. ISSN 1533-7928. URL `http://jmlr.org/papers/v24/20-1321.html`.

A Evaluating the DOMINO algorithm

With the time series on which DOMINO has been trained as well as their weights, the model is queried by inputting a set of M time points such that $M < N$. The time points contain the same information as the training data, that is either $\mathbf{t}$ as an input; and $\mathbf{y}$ is the output. Made up of M time-steps, these have dimension $\mathbf{y}, \mathbf{t} \in \mathbb{R}^M$.

The DOMINO algorithm is also given a set of input time points over which an output must be returned - these will be made up of $\mathbf{x} \in \mathbb{R}^N$ and will consist of the first M points from the query input, plus $N - M$ extra points which will be used to predict the next points.

The random walk, starting at the $M + 1^{th}$ point, uses the training time-series and their probabilities to predict the rest of the time series.

B Hyperparameter tuning

The DOMINO model has multiple hyperparameters, which control the learning of probabilities for underlying individuals. We run ablation studies for each hyperparameter: the maximal number of epochs for the learning (Table 3), the maximum percentage δ of the data range which is a possible divergence threshold between the DOMINO and the underlying training individuals (Table 4), the maximum percentage of points in all the time series which can be over the δ hyperparameter (Table 5), and the regularisation parameter λ (Table 6). The best results for the hyperparameter are given in bold.

Table 3: Hyperparameter tuning: average (std) MAE for maximal number of epochs on 10 runs.

Max epochs	**Result**	**CV**
5	5.750 (0.753)	5.302 (0.313)
10	5.648 (0.792)	5.244 (0.225)
15	5.082 (0.792)	5.314 (0.308)
20	5.460 (0.668)	5.174 (0.331)
25	5.206 (0.544)	4.964 (0.215)
30	**5.049** (0.409)	5.054 (0.251)

Table 4: Hyperparameter tuning: average (std) MAE for δ threshold for divergence on 10 runs.

δ	**Result**	**CV**
1%	5.413 (0.597)	5.247 (0.341)
2%	5.218 (0.502)	5.034 (0.218)
3%	5.187 (0.537)	5.211 (0.307)
5%	**5.185** (0.583)	5.096 (0.283)
10%	5.640 (0.594)	5.331 (0.453)

Table 5: Hyperparameter tuning: average (std) MAE for maximal number of values over δ on 10 runs.

Maximum percentage of values over δ	**Result**	**CV**
1%	5.130 (0.732)	4.67 (0.360)
2%	5.249 (0.434)	5.102 (0.297)
3%	**5.130** (0.637)	5.156 (0.241)
5%	5.344 (0.521)	5.110 (0.375)
10%	5.275 (0.499)	5.054 (0.190)

Table 6: Hyperparameter tuning: average (std) MAE for λ the regularisation parameter on 10 runs.

λ	**Result**	**CV**
0.5	**4.698** (0.473)	5.450 (0.407)
1	5.308 (0.506)	5.231 (0.566)
1.5	5.315 (0.360)	5.129 (0.540)