Forecasting offshore wind energy

Online learning, bounds and missing data

Pierrot, Amandine

Link to article, DOI:
10.11581/DTU.00000281

Publication date:
2023

Document Version
Publisher's PDF, also known as Version of record

Division for Power and Energy Systems
Department of Wind and Energy Systems

Forecasting offshore wind energy: Online learning, bounds and missing data

Amandine Pierrot

PhD Thesis, July 2023, Kongens Lyngby, Denmark

DANMARKS TEKNISKE UNIVERSITET
Division for Power and Energy Systems (PES)
DTU Wind and Energy Systems

Forecasting offshore wind energy:

Online learning, bounds and missing data

Dissertation, by Amandine Pierrot

Supervisors:
Pierre Pinson, Imperial College London, Technical University of Denmark
Jalal Kazempour, Technical University of Denmark

DTU - Technical University of Denmark, Lyngby - July 2023

Forecasting offshore wind energy:

This thesis was prepared by:
Amandine Pierrot

Supervisors:
Pierre Pinson, Imperial College London, Technical University of Denmark
Jalal Kazempour, Technical University of Denmark

Dissertation Examination Committee:
Tuhfe Göçmen
Department of Wind and Energy Systems, Technical University of Denmark, Denmark

Jethro Browell
School of Mathematics and Statistics, University of Glasgow, United Kingdom

Benjamin Guedj
Department of Computer Science, University College London, United Kingdom
Inria Lille, France

Division for Power and Energy Systems (PES)
DTU Wind and Energy Systems
Elektrovej, Building 325
DK-2800 Kgs. Lyngby
Denmark

Tel: (+45) 4525 3500
Fax: (+45) 4588 6111
E-mail: cee@elektro.dtu.dk

Release date: July 2023

Edition: 1.0

Class: Internal

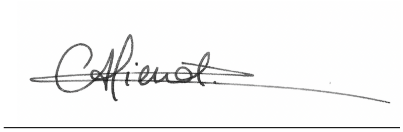
Remarks: The dissertation is presented to the Department of Wind and Energy Systems of the Technical University of Denmark in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

DOI: https://doi.org/10.11581/DTU.00000281

— Peppermint Patty, Schulz, 1973

To Monsieur Body

Preface



Amandine Pierrot
July 2023

Acknowledgements

I wanted to do a PhD for so long, and I was acknowledged so often in theses for my always-up-for-a-beer skill over the past fifteen years, that it feels unreal now that it is my turn.

This could not have happened without Pierre Pinson, my supervisor, his open mind, his enthusiasm and his wisdom. Pierre, thank you for giving me the opportunity I was longing for and for always supporting me, no matter where you were headed. Thank you for the science, the math, the funky ideas: I learned so much and - I never thought I would write this - I think I really enjoy optimization now! Jalal, thank you for your effort to keep the E(L)MA spirit up, and for always being Peyo's biggest fan. Linde and Liyang, sadly, no time for our collaboration plans. But it felt good already, just to think about it. Anubhav, the prince of feedback, thank you for spending so much holiday time reviewing this thesis. I hope it did not *consist* too much *of* a *curse*. Peter, thank you for correcting my Google-translate Danish resumé at the last minute.

I was lucky enough to be part of the ELMA group, where each and everyone was welcomed and supported in their diversity. In particular, I would like to thank Andreas, Anubhav, Eléa, Ilgiz, Jiawei, Jochen, Li, Linde, Liyang, Misha, Morten, Sam, Tiago, Yannick. I wish I had more time with the *new* E(L)MA, in particular the *new* E(L)MA members I had the chance to meet: Andrea, Ben, Enrica, Lesia, Peter, Thomas. Eva, Mette, Kamilla, thank you for always being so helpful and efficient.

Not everything was about work in Copenhagen. Eléa, thank you for our perfect, relaxing puzzle/white-Russian/Eléa's-playlist nights. You already know that, but I really missed those. Anubhav, thank you for our Black Diamond concerts, among many other things. Also, moving to Amager: great idea. Tiago, thank you for the football, and for always being so confident I was going to make it. Also, leaving Amager: bad bad move. Liyang, thank you for your amazing plays and house/brunch/barbecue parties, and for such a great time on the French Riviera. Tivoli, I am glad you exist. Also, I am indebted to my streaming platform providers: my good friend Lolo, and my favorite little sister, Aurore.

Jochen, thank you for taking the dog to places, here and there.

I would not have made it to Copenhagen without my EDF family. So, *par ordre d'apparition*, Nicolas, Yannig, Virginie, Gilles, Bérénice, Manel, thank you for your friendship during all those years. Ghislain, thank you for being so creatively supportive, for Béninoise deliveries and Fâ magic. I am looking forward to making my own vegetarian offering someday. Jairo, thank you for accepting me on my external stay, I wish it had happened! I hope this is only postponing working together again.

I would not have made it back here without the support of my dad and Martine. Thank you so much for that. It felt a bit strange, and at the same time it was very comforting, to finish this PhD journey in the neighborhood where I grew up, and which I had left more than thirty years

ago. Thank you Mom and the family for the nice memories I have here. Valérie, no idea if you will have the chance to read this one day. If you do, I cannot thank you enough for the life I have now.

Against all odds, someone else made it through this PhD. I suspect he did it for me... À mon Peyo.

Amandine

Aix-les-Bains, France, July 2023.

Table of Contents

List of Figures

Abstract

Forecasting is of the utmost importance to the integration of renewable energy into power systems and electricity markets. Indeed, to get electricity from conventional generators such as fuel-based or nuclear power plants, one is in charge of the production, whereas renewable energy sources are fundamentally variable and weather-dependent. Full benefits from their integration can only be reaped if one is given reliable, trustworthy forecasts and therefore the opportunity to accommodate the actual renewable power generation in an optimal way. In this thesis, we focus on offshore wind power short-term forecasting, as wind power fluctuations at horizons of a few minutes ahead particularly affect the system balance and are the most significant offshore. Those very short-term lead times are not only crucial but also the most difficult to improve the forecasts for, especially compared to the simple but very effective persistence benchmark.

Forecasts characterize but do not eliminate uncertainty. Therefore, they ought to be probabilistic, taking the form of distributions. Wind power generation is a stochastic process which is double-bounded by nature, by zero when there is no production and by the nominal power for high-enough wind speeds. It is non-linear and non-stationary. For short-term forecasting, statistical methods have proved to be more skilled and accurate. However, they often rely on stationary, Gaussian distributions, which cannot be appropriate for wind power generation. We start by extending previous works on generalized logit-normal distributions for wind power generation. First, we develop a rigorous statistical framework to estimate the full parameter vector of the distribution through maximum likelihood inference. Then, we derive the corresponding recursive maximum likelihood estimation and propose a recursive algorithm which can track the full parameter of the distribution in an online fashion.

From the observation that bounds are always assumed to be fixed when dealing with bounded distributions, which may not be appropriate for wind power generation as curtailment actions happen, we develop new statistical frameworks where the bounds of a distribution are allowed to vary without being observed. First, we address the bounds as additional parameters of the distribution and propose an online algorithm for quasiconvex functions which is able to track a new bound parameter over time along with the original parameters of the distribution. Alternatively, to account for the uncertainty in the bounds as well, we propose to introduce them in the statistical model as discrete latent variables. To deal with these additional, missing, variables, we suggest batch and online algorithms based on the expectation-maximization method.

The algorithms developed during this thesis were run on both synthetic and real power generation data. We question the k-nearest neighbors imputation method we implicitly used to deal with missing data in historical records. To address a common shortcoming in such non-parameteric methods, which is to overlook the distances between the neighbors themselves, we propose to explicitly acknowledge the structure of the wind farm by considering it as a graph. Then, we design an augmented imputation method which combines spectral graph theory and online learning to exploit information from both the wind farm layout and the data already collected.

Resumé

Forudsigelser og prognoser er af største betydning for integrationen af vedvarende energi i elsystemer og elmarkeder. For at få elektricitet fra konventionelle generatorer, såsom brændstofbaserede eller atomkraftværker, er man faktisk ansvarlig for produktionen, hvorimod vedvarende energikilder er fundamentalt variable og vejrafhængige. Det fulde udbytte af deres integration kan kun høstes, hvis man får pålidelige, troværdige prognoser og derfor mulighed for at imødekomme den faktiske vedvarende elproduktion på en optimal måde. I dette speciale fokuserer vi på korttidsprognoser for offshore vindkraft, da produktionsudsving i vindkraft i horisonter på få minutter frem især påvirker systembalancen og er især markante for offshore. Disse meget kortsigtede horisonter er ikke kun afgørende, men også de sværeste at forbedre prognoserne for, især sammenlignet med det enkle, men meget effektive benchmark.

Prognoser karakteriserer, men fjerner ikke usikkerhed. Derfor bør de være probabilistiske i form af fordelinger. Vindkraftproduktion er en stokastisk proces, som er afgrænset af naturen, når der ikke er produktion pga., at vinden ikke blæser, og af den nominelle effekt for høje nok vindhastigheder. Den er ikke-lineær og ikke-stationær. Til kortsigtede prognoser har statistiske metoder vist sig at være mere dygtige og nøjagtige. De er dog ofte afhængige af stationære, normalfordelte distributioner, som ikke altid er passende til vindkraftproduktion. Vi starter med at udvide tidligere arbejde med generaliserede logit-normalfordelinger til vindkraftproduktion. Først udvikler vi en stringent statistisk ramme for at estimere den fulde parametervektor for fordelingen gennem maksimal sandsynlighedsinferens. Derefter udleder vi den tilsvarende rekursive maksimale sandsynlighedsestimering og foreslår en rekursiv algoritme, som kan estimere parameterne for distributionen online.

Ud fra iagttagelsen af, at grænser altid antages at være faste, når man har at gøre med afgrænsede fordelinger, hvilket måske ikke er passende for vindkraftproduktion, da der sker afskæringshandlinger, udvikler vi nye statistiske rammer, hvor grænserne for en fordeling får lov til at variere uden at blive observeret. Først adresserer vi grænserne som yderligere parametre for fordelingen og foreslår en online-algoritme for kvasikonvekse funktioner, som er i stand til at spore en ny bundet parameter over tid sammen med de originale parametre for fordelingen. Alternativt, for også at tage højde for usikkerheden i grænserne, foreslår vi at introducere dem i den statistiske model som diskrete latente variable. For at håndtere disse yderligere variable foreslår vi batch- og onlinealgoritmer baseret på forventningsmaksimeringsmetoden.

Algoritmerne udviklet i løbet af dette speciale blev kørt på både syntetiske og reelle produktionsdata. Vi sætter spørgsmålstegn og undersøger k-nærmeste naboers imputationsmetode, som vi implicit brugte til at håndtere manglende data i historiske optegnelser. For at løse en almindelig mangel ved sådanne ikke-parameteriske metoder, som er at overse afstandene mellem naboerne selv, foreslår vi eksplicit at anerkende vindmølleparkens struktur ved at betragte den som en graf. Derefter designer vi en udvidet imputationsmetode, som kombinerer spektralgrafteori og online læring for at udnytte information fra både vindmølleparkens layout og de allerede indsamlede data.

CHAPTER 1

Introduction

1.1 Context and motivation

Forecasting is of the utmost importance to the integration of renewable energy into power systems and electricity markets. Indeed, to get electricity from conventional generators such as fuel-based or nuclear power plants, one is in charge of the production, whereas renewable energy sources are fundamentally variable and weather-dependent. Full benefits from their integration can only be reaped if one is given reliable, trustworthy forecasts and hence, the opportunity to accommodate the actual renewable power generation in an optimal way. Forecasts characterize but do not eliminate uncertainty. Therefore, they should be probabilistic, taking the form of predictive probability distributions [1]–[3]. In particular, renewable energy forecasting should be performed in a probabilistic framework, since point forecasts are not informative enough when it comes to quantify the inherent uncertainty of renewable power generation.

Attention to energy forecasting has increased to a great level over the years [4]. The choice of a forecasting method is driven by an important notion in energy forecasting, which is the forecast horizon. This refers to the length of time into the future for which forecasts are to be prepared, and typically runs from very short-term to long-term lead times. Short-term forecasts are issued a few minutes or hours before the event of interest. Long-term forecasts, on the other hand, can be issued decades ahead of time. For short-term energy forecasting, statistical methods have proved to be especially skilled and accurate, as the most recent records of the quantity or event to be forecast next carry a lot of information.

Wind power fluctuations at horizons of a few minutes ahead particularly affect the system balance, and are the most significant offshore. Experience from Horns Rev offshore wind farm showed that power fluctuations within 10-minute intervals can be remarkably high due to the concentration of wind power in a small area [5], [6]. Those very short-term lead times are not only crucial but also the most difficult to improve the forecasts for, especially compared to the simple yet effective persistence benchmark. Persistence point forecasting simply takes the last observed value as the next forecast. To get probabilistic forecasts out of them, persistence point forecasts are often dressed with point forecast past errors. Even though most efforts in wind power forecasting are placed on lead times ranging from hours to days, there is a growing interest in alternative approaches to improve the accuracy of very short-term forecasts, for instance leveraging detailed turbine-level data [7].

Wind power forecasting has been leading probabilistic forecasting in the energy forecasting field, thanks to the close collaboration between wind power forecasters and meteorologists. Wind power generation is a stochastic process which is double-bounded by nature: by zero when there is no production and by the nominal power for high-enough wind speeds. It is non-linear and non-stationary [8]. However, short-term statistical models often rely on stationary frameworks and Gaussian distributions, which cannot be appropriate for wind power generation. Accommodating

the non-linearity, the non-stationarity and the bounded support of wind power generation may pave the way to improving both point and probabilistic wind power forecasts.

1.2 Challenges and research directions

Throughout this thesis, focus is placed on very short-term probabilistic forecasting for offshore wind energy, since short-term fluctuations in power generation are most significant offshore. We are interested in parametric probability distributions that can handle the non-linearity and bounded support of wind power generation. Probability distributions suited for continuous bounded random variables include the beta distribution, truncated distributions and distributions of transformed normal variables [9]. We work with the generalized logit-normal (GLN) distribution [10], which belongs to the third category and has been shown to better account for the special characteristics of wind power generation compared to classical beta and normal assumptions [11]. Lastly, to accommodate the non-stationarity of wind power generation, we aim to develop online methods.

For forecasting purposes, to choose a parametric distribution is to make assumptions about the shape of the predictive densities. This choice being made, we still need to get estimates for the parameters of the chosen distribution, and parameter estimation is more or less challenging depending on the distribution and the parameter at hand. First, **we aim to provide an approach where all the parameters of a GLN distribution are estimated from data in a rigorous framework.** Easy problems can be solved using least squares methods, for instance in the case of linear models. A more general estimation method is maximum likelihood inference. It assumes that the most reasonable values for the parameter vector of a distribution, or more generally a model, are those for which the probability of the observations is the highest. Indeed, it can be shown that least squares for a linear model with a Gaussian assumption on its errors is equivalent to maximum likelihood estimation. So, although the additional assumption of normality seems more restrictive, the results are the same when estimating the parameters of the linear model. Adding assumptions about the errors makes it possible to move to probabilistic forecasting, and through maximum likelihood inference one is now able to estimate the variance as an additional parameter of the linear model. Maximizing a likelihood comes down to solving an optimization problem with respect to the parameters of the statistical model, the observations being fixed. Depending on the parameter at hand, simple closed-form solutions to this optimization problem might exist. This is the case when estimating the parameters of the linear model, including the variance of the normally distributed errors. The GLN distribution, on the other hand, relies on three parameters: a location and a scale parameters, usually denoted as μ and σ^2 by analogy with the normal distribution, and a shape parameter ν which makes the distribution *generalized*, as for $\nu = 1$ it is just called logit-normal. The difficulty of maximum likelihood inference for GLN distributions lies with the parameter ν, since there exists no closed-form expression of its maximum likelihood estimate, thereby requiring the use of iterative methods.

Next, to accommodate the non-stationarity of wind power generation, **we move from batch to recursive maximum likelihood estimation for the parameter vector of the GLN distribution to be able to evolve and adjust to new characteristics in the signal.** Because we are to issue probabilistic forecasts only a few minutes ahead of time, the proposed algorithms should run fast, along with being sparse and efficient, as progress in renewable energy forecasting should not be made at the cost of intensive computations. Recursive estimation allows for parametric time-variability and provides information not only on the existence of non-stationarity, but also on

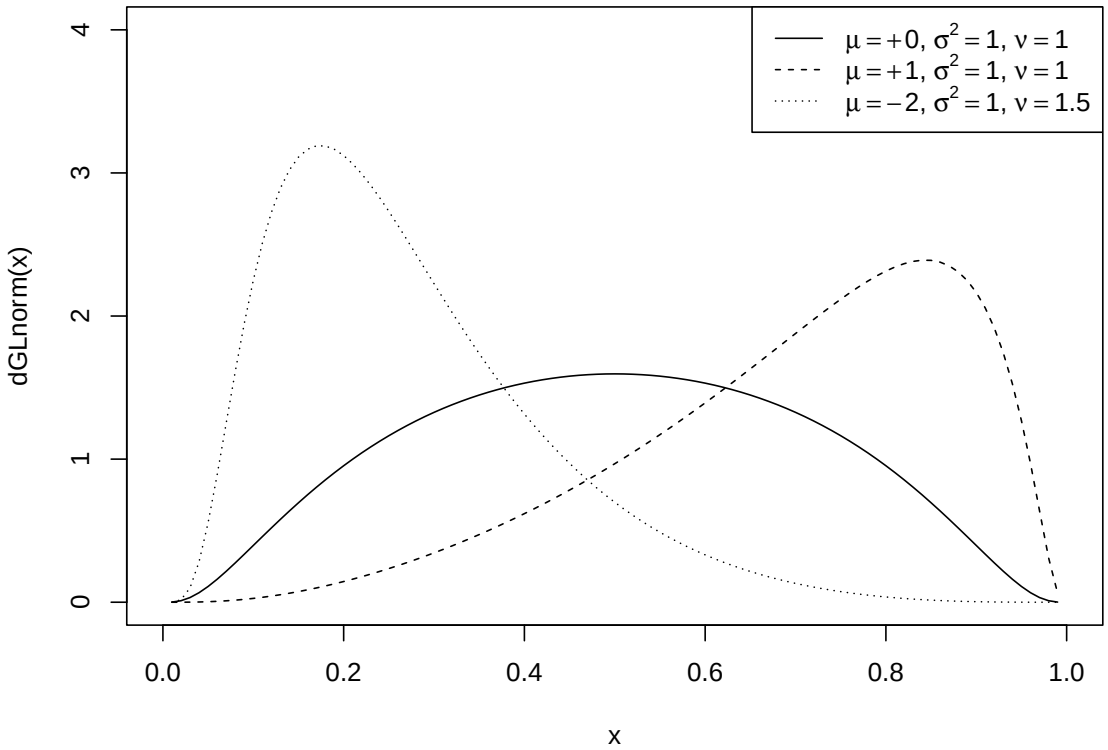


Figure 1.1: Probability density function of the GLN distribution for different values of its location parameter μ, its scale parameter σ^2, and its shape parameter ν.

the possible nature of the parametric variations [12]. As maximum likelihood inference relies on the likelihood function, recursive estimation procedures can be derived by introducing a time-dependent likelihood and getting a new parameter estimate as a function of the previous estimate [13], [14]. Ultimately, our objective is to move towards online learning approaches for wind power probabilistic forecasting, which make it possible to track a full parameter vector online [15]–[17].

We note that while dealing with bounded distributions, the bounds are always assumed to be fixed, i.e., the support of the probability distribution does not vary with time or exogenous variables. This may not be appropriate for wind power generation, as curtailment actions happen, and negatively impact inference and forecasting. We do not observe these varying bounds nor have reliable information about them. **Therefore, we look at the bounds of the support as additional parameters of the GLN distribution and aim to develop online algorithms capable of tracking them over time along with the original parameters**. The main challenges here concern both the inference and the forecasting tasks. Regarding inference, we need to solve a new, possibly not convex optimization problem, in order to estimate these additional bound parameters. Moreover, in an online framework we need to allow and handle observations falling outside of the support of the probability distribution. This needs to be addressed for enabling online inference, and when moving to forecasting, since it is still possible that the next observation falls outside of the support of our predictive density.

Alternatively, to account for the inherent uncertainty which comes with the inability to observe the bounds, **we investigate more general frameworks where the bounds are missing random variables with their own probability distribution.** This calls for a range of statistical methods

which are known as expectation-maximization (EM) algorithms [18]. The EM algorithm is a popular tool for simplifying difficult maximum likelihood problems. The maximum likelihood estimation may become difficult if it involves missing variables. Therefore, the EM algorithm is often associated with problems involving missing variables or data. Let $\mathbf{X}$ and $\mathbf{X}^m$ be the observed and missing data, respectively. The algorithm works with the *complete* log-likelihood, which is the log-likelihood of the complete data $(\mathbf{X}, \mathbf{X}^m)$. It relies on two steps: an expectation step, the E-step, and a maximization step, the M-step. The E-step is about the expectation of the complete log-likelihood conditional on the observed data $\mathbf{X}$ and the current estimate of the model parameter vector; at the M-step, a new estimate of the parameter vector is computed by maximizing the expectation function from the E-step. This is a rather general and flexible framework, where the uncertainty in the bounds is accounted for, since the parameters of the bounds' distribution are included in the model parameter vector and updated at the M-step. However, it brings its own challenges, which mostly concern the EM algorithm. Because there is no closed-form solution for the parameter ν, the M-step of the EM algorithm entails iterative methods as well and requires what is termed as a generalized EM algorithm [18]–[20]. Moreover, depending on the distribution which has been chosen for the bounds, the E-step could be straightforward, or intractable and require Monte Carlo [21] or stochastic versions of the EM algorithm [22], [23]. Last but not least, getting an online version of the EM algorithm might be particularly challenging as current works on online/stochastic EM algorithms largely focus on the exponential family [24]–[26]. Although Titterington's recursive algorithm can be applied to probability distributions outside of the exponential family [27], it requires the computation of the complete-data Fisher information matrix, which would be cumbersome for GLN distributions.

The algorithms discussed in this thesis are intended to be applied to both synthetic and real power generation data. When interested in the average production at a wind farm at time t, it is quite intuitive to work with the average of the wind turbine productions that are recorded in the dataset at time t. By doing so, one implicitly performs k-nearest neighbors (k-NN) imputation [28]. The k-NN algorithm is a seminal non-parametric method in machine learning, which uses the k points closest to a point of interest to make a decision about it [29]–[31]. A common shortcoming in current non-parametric methods is to only consider the distances between the decision point and its neighbors, and ignore the geometrical relation between those neighbors. Before we get any records from its sensors, **a wind farm is a graph with its own geometry and we aim to take advantage of this a priori information to improve the imputation of power generation missing values.** Since the world's first offshore wind farm, Vindeby in Denmark, which totalled 11 turbines in 1991, the size of offshore wind farms has increased to more than a hundred wind turbines, e.g., Hornsea 1 in the United Kingdom which totals 174 wind turbines. Because of the increasing number of turbines in offshore wind farms, the issue of missing data becomes even more critical than it used to be.

Pursuing the methods and algorithms developed during this thesis originate while considering wind power forecasting. However, they are of interest for a much broader range of statistical and forecasting applications, as soon as bounded variables are involved. Staying in the renewable energy field, one can think of solar power generation since it is also often expressed as a percentage of the nominal power of the power plant, which can shift over time for no documented reason, see, e.g., [32], [33]. Even if the nominal power remains constant, some phenomena may happen that limit the effective capacity, such as dust in desert areas.

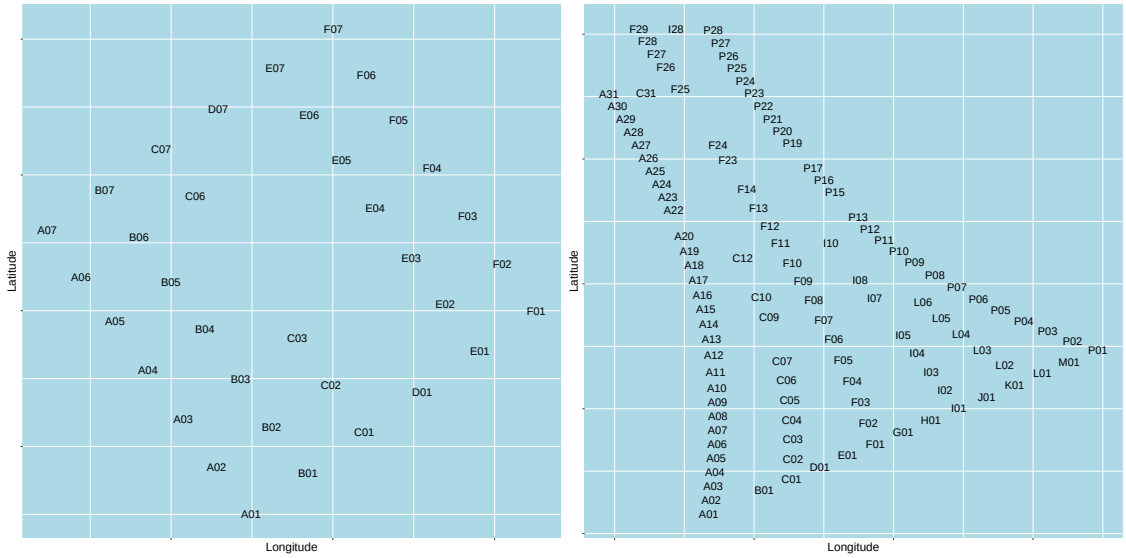


Figure 1.2: Position and name of the wind turbines at Westermost Rough offshore wind farm (left) and Anholt offshore wind farm (right).

1.3 Application framework

Many case studies could be considered for the methods and algorithms we aim to develop. In this thesis, we will make use of two newly available datasets from two offshore wind farms we were provided with, when evaluating the performance of our algorithms on real data. Both datasets include metadata about the wind turbines' location and technical characteristics, along with supervisory control and data acquisition (SCADA) files, i.e., signals recorded for each wind turbine such as active power, wind speed, yaw position or ambient temperature. These time series are available at a temporal resolution of every ten minutes. The two offshore wind farms are Westermost Rough, situated in the North Sea, England, and Anholt, situated between Djursland and the island Anholt in the Kattegat, Denmark. Westermost Rough totals 35 wind turbines placed according to a grid pattern, while Anholt totals 111 wind turbines in a rather non-conventional layout. Representations of the wind farms through the position and name of their wind turbines are shown in Figure 1.2.

Westermost Rough We have individual time series over two years, from January 1, 2016, to December 31, 2017. However, not all time series start on January 1. For wind power generation, the first complete record, i.e., with values for all wind turbines, is only available on February 2, 2016. About half the total 105,264 records are complete records, i.e., we have the power generation value for all wind turbines. Most of the incomplete records only miss one (60.80%) or two values (23.13%). The distribution of the number of missing entries by record is plotted in Figure A.1 of Appendix A. Sequences of consecutive missing data can be very long for a wind turbine, up to 8,728 time steps, i.e., about 61 days. Depending on the wind turbine, between 0.15 and 5.81% of individual power generation data are missing over the period.

Anholt We have individual time series over 2.5 years, from January 1, 2013, to June 30, 2015. Although some individual power generation records start from January 1, 2013, the first complete record of wind power generation is only available on June 22, 2013. After July 2013, many entries

are still missing for a few months: October 2014, February 2015 and March 2015. About 66% of the total 131,184 records are complete records. About 30% of the incomplete records only miss one value. However, 4.62% of the incomplete records miss 75 values and 4.02% are fully empty records. The distribution of the number of missing entries by record is plotted in Figure A.2 of Appendix A. Again, sequences of consecutive missing data can be very long for a wind turbine, up to 11,024 time steps, i.e., about 77 days. Depending on the wind turbine, between 4.10 and 22.45% of individual power generation data are missing over the period.

1.4 Scientific contributions

The main objective of this thesis is the design of new methods and algorithms to improve predictive densities for wind power short-term forecasting. In particular, we focus on accommodating the non-stationarity of wind power generation through online learning approaches, its non-linearity and bounds through more appropriate distribution assumptions.

Full recursive estimation of the generalized logit-normal distribution We start by extending previous work on GLN distributions for wind power generation [11], where not all the parameters of the distribution were estimated, as the shape parameter ν was selected upon cross-validation. We propose to revisit this work and to estimate all the parameters of the GLN distributions within a maximum likelihood framework. In [**Paper A**] we provide a batch algorithm which relies on conditional maximization of the model log-likelihood [34]. Conditional maximization enables us to benefit from the closed-form solutions which are available for the location and scale parameters of the GLN distribution, and to limit the iterative part of the algorithm to a one-dimensional Newton-Raphson step. Then, we apply ideas from [13] and [14] to our model and propose a recursive version of our algorithm. The recursive algorithm makes it possible to estimate and update the full parameter vector of the GLN distribution in an online fashion. It can be seen as a quasi-Newton approach, since it uses only first-order information and an approximation of the Hessian matrix which ensures positive definiteness. Both batch and recursive algorithms are applied to 10-minute-ahead point and probabilistic forecasting at the Anholt offshore wind farm. Point forecasts are evaluated through root mean square errors. Probabilistic forecasts are evaluated according to a strictly proper scoring rule, the continuous ranked probabilistic score (CRPS) [35], and by looking at reliability diagrams and marginal calibration plots [3], [36]. The recursive algorithm, in particular, provides significant improvement over batch and Gaussian methods when issuing predictive densities.

Online tracking of a varying upper bound We are then interested in challenging the assumption of fixed bounds for bounded variables. We are not aware of previous works on this matter. We start by considering the bounds as new, additional parameters of the distribution. Bounds as parameters only make sense in an online framework, as they need to vary with time. Because our application is wind power generation, we are mostly concerned with the upper bound. Therefore, we derive the method focusing on this upper bound, but we want to emphasize that it would be straightforward to apply it to a lower bound. So far, we have worked with an upper bound assumed to always be 1, as we normalize the wind power generation by the nominal power of the turbine. We still aim to estimate the full parameter vector of the GLN distribution, which now includes the upper bound, through maximum likelihood inference. The first challenge when dealing with the bound as a parameter in a non-stationary framework is to handle past observations which

are outside of the support of the bounded distribution, and make the log-likelihood to be infinite. In order to take into account these observations in a soft, finite way, in [**Paper B**] we introduce a new term into the log-likelihood which relies on the sigmoid function. We choose to call this new log-likelihood the *extended* log-likelihood. The second challenge we need to tackle is that when considering the bound as a parameter, we cannot be in a classical convex optimization framework anymore, since the negative log-likelihood appears not to be convex in the bound parameter. Therefore, we propose to move to quasiconvex optimization, and use recent theoretical results about local quasiconvexity and (stochastic) normalized gradient descent [37]. We design a batch algorithm out of normalized gradient descent (NGD), and an online algorithm out of stochastic normalized gradient descent (SNGD). In addition to these novel quasiconvex algorithms, we propose a more classical online convex algorithm, which is the recursive algorithm from [**Paper A**], augmented to include an upper bound parameter in the statistical model. The proposed algorithms are first run on synthetic data for checking their tracking abilities. They are also applied to 1-step-ahead probabilistic forecasting. Here a third challenge arises, as it can always happen that the next observation falls outside of the support of our predictive density, when the current estimate of the upper bound is too low. We show that the CRPS is nicely increased by an observation falling outside of the support of the predictive distribution, to a higher finite value, contrary to, e.g., the logarithmic score, another strictly proper scoring rule which becomes infinite. The proposed algorithms are then applied to 10-min-ahead probabilistic forecasting at the Anholt offshore wind farm and compared to the recursive algorithm proposed in [**Paper A**], i.e., for which the bound is assumed to be fixed to 1. Our *online normalized gradient descent* (ONGD) algorithm shows significant improvement over the recursive algorithm of [**Paper A**], contrary to the updated version of the recursive algorithm which shows equivalent performances. However, none of the proposed algorithms achieve probabilistic calibration [36], as prediction intervals are too narrow on average, i.e., the predictive densities are overconfident.

Online mixture of scaled generalized logit-normal distributions When considered as a time-varying parameter, a bound can take only one value at time t, and this value cannot be too far from the previous one. To account for uncertainty and allow sharp variations in the values successively taken by the bound, we introduce the latter in our model as a random variable with its own probability distribution. This implies we do not have a single value of the bound when issuing the predictive density of the bounded variable, but a full distribution. Therefore, if the distribution of the bound does not change over time, we will always forecast the same values with the same probabilities. Focusing again on an upper bound, and from the intuition that a grid of plausible values over the unit interval may matter more to modelling than the accuracy of each value, we propose to assume the bound to be a discrete random variable which can take K values over $(0, 1]$. The marginal probability density function of the bounded variable then becomes a finite mixture of K scaled GLN distributions, which is similar to a kernel density with scaled GLN kernels. Ideally, we wish the probabilities assigned to each value of the bound to be updated online depending on the most recent information, in order for the probabilistic forecasts to achieve better probabilistic calibration while still being informative. When we do not have information in favor of certain values over others, equal probabilities would be assigned to all values; otherwise, positive probabilities would go to a few values only. This framework is very flexible in the sense that one is free to decide on how many and which values they want for the bound, on covering the whole unit interval or focusing on a smaller interval with closer GLN kernels. Flexibility also comes with

more control over the values which can be taken by the bound. The main challenge lies in the design of online EM algorithms, as we need the distributions of both the bound and the bounded variable to vary with time for wind power forecasting applications. This work is presented in the technical report [**Paper C**] as of now. It has been inspired by wind power forecasting but still is at exploratory stage, focusing on methodological aspects and simulation studies. We propose a batch Monte Carlo expectation-conditional maximization (MCECM) algorithm and two online algorithms for stochastic processes with auto-regressive (AR) dependency. We also provide an expectation-conditional maximization (ECM) algorithm along with an online algorithm for more general frameworks where the bounded variables are independent and identically distributed. All algorithms were run on synthetic data to illustrate their performance for convergence and tracking purposes. The performance of the online EM algorithms for stochastic processes was also assessed for forecasting purposes. The batch algorithms and the online EM algorithm for independent data performed well on the simulation study. However, further work is required on an online EM algorithm for stochastic processes so it can be applied to real-world data.

Online imputation of missing values using spectral graph theory Having worked with the power generation averaged over the wind farm, we question the implicit k-NN imputation which is performed by averaging over available entries when a record is incomplete at time t. We address the common shortcoming of only considering distances between a point of interest and its neighbors by using the geometry of the wind farm in [**Paper D**]. Our method relies on learning Laplacian eigenmaps out of the graph of the wind farm through spectral graph theory [38], [39]. These learned representations can be based on the wind farm's layout only, or additionally account for information provided by observed data. The corresponding weighted graph is allowed to change with time and can be tracked in an online fashion. Application to the Westermost Rough offshore wind farm shows significant improvement over approaches that do not account for the wind farm's layout information. Our imputation method does not rely on any model assumptions. Therefore, it can be used for subsequent supervised learning tasks, such as forecasting, for any kind of learner/predictor. In [**Paper D**] we focus on what is known as single imputation, as we try to impute missing entries as accurately as possible, which gives us only one completed dataset. Multiple imputation on the other hand consists of predicting M different values for each missing data point, and provides M imputed datasets. Multiple imputation is usually preferred, especially for inference tasks, as it ensures the variance is properly accounted for [40]. Because a neighbor's weight can be seen as the probability of the missing point to take the value of its neighbor, weighted k-NN nicely enable to move to multiple imputation, or more generally, towards a probabilistic framework.

1.5 Thesis outline

This thesis gives an overview of the contributions made during this Ph.D. project, based on the papers written over the period. Chapter 2 sets the statistical framework of using GLN distributions for wind power probabilistic forecasting. It deals with maximum likelihood inference for both batch and recursive estimations. We also introduce the evaluation tools we will be using for assessing our predictive densities throughout the thesis. In Chapter 3, we address the issue of having a varying support for the GLN distribution without observing the varying bounds. The scientific contributions related to the bounds as parameters are first presented, followed by those related to the bounds as discrete random variables. Chapter 4 is dedicated to dealing with missing values in

general and at offshore wind farms, and in particular while considering the wind farm as a graph. Finally, Chapter 5 concludes and discusses possible directions for future work.

The corresponding scientific articles are attached at the end of the thesis.

1.6 List of publications

The relevant publications which are the core of this thesis are:

[**Paper A**] A. Pierrot and P. Pinson, "Adaptive generalized logit-normal distributions for wind power short-term forecasting", in *Proceedings 2021 IEEE Madrid PowerTech*, 2021.

[**Paper B**] A. Pierrot and P. Pinson, "On tracking varying bounds when forecasting bounded time series", submitted to *Technometrics*, 2023.

[**Paper C**] A. Pierrot and P. Pinson, "Mixtures of bounded distributions with different bounds: Estimation and forecasting", to be submitted to the *Annals of Applied Statistics*, technical report as of now, 2023.

[**Paper D**] A. Pierrot and P. Pinson, "Data is missing again – Reconstruction of power generation data using k-nearest neighbors and spectral graph theory", submitted to *Wind Energy*, 2023.

CHAPTER 2

Generalized logit-normal densities for wind power forecasting

This chapter presents the preliminary methods and algorithms developed in [**Paper A**] to fully estimate and issue (predictive) GLN densities for wind power forecasting, and are the base for subsequent more advanced methods. We introduce the GLN distribution in Section 2.1 and we show in Section 2.2 how to apply it to wind power generation. This involves accommodating the serial dependency in the wind power stochastic process and considering densities conditional on the past of the sequence. We then state the maximum likelihood inference for both stationary and non-stationary frameworks in Section 2.3, and explain the corresponding batch and recursive algorithms. Finally, the question of evaluating the forecasts which are to be issued by these models is addressed in Section 2.4, focusing on predictive densities. This chapter also sets the notations used throughout this thesis in order to ensure consistency. Note that they might differ from the ones used in the related articles, in particular from the notations of [**Paper A**].

2.1 The generalized logit-normal distribution

The practical use of any family of distributions depends on the possible variation in its shape, and on the ease with which the distribution can be fitted. The GLN distribution is very flexible thanks to three parameters: the location μ, the scale σ^2, and the shape ν. It relies on a generalization of the logit transform and comes down to the logit-normal distribution when $\nu = 1$ [10]. Let X be the original random variable, $X \in (0, 1)$. The generalized logit transform Y is given by

$$Y = \gamma(X; \nu) = \log \frac{X^\nu}{1 - X^\nu}, \tag{2.1}$$

where $\nu > 0$ is the shape parameter. When Y is distributed according to a normal distribution $\mathcal{N}(\mu, \sigma^2)$, the probability distribution of the original variable X is the GLN distribution $L_\nu(\mu, \sigma^2)$ [11], [41], [42]. Because the transformed variable is normally distributed, nice properties can be derived for the original random variable X. In particular, the probability density function p of X can be expressed as a function of the standard normal density,

$$p(x; \theta) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma^2}} \frac{\nu}{x(1-x^\nu)} \exp\left[-\frac{1}{2}\left(\frac{\gamma(x;\nu)-\mu}{\sigma}\right)^2\right] & \text{if } 0 < x < 1, \\ 0 & \text{otherwise,} \end{cases} \tag{2.2}$$

where $\theta = (\mu, \sigma^2, \nu)$ is the parameter vector of the distribution.

This distribution belongs to the so-called S_B family of distributions [41]. This minimally informative name comes from the fact that these distributions have a Bounded domain. In the case of the GLN distribution, the domain is $(0, 1)$. The first analytic characterisation of the logit-normal family was

done in terms of a *system of curves* constructed from transformations of normal variables [9]. There are no available analytical expressions for the expectation and the variance. In the Appendix to his original article [9], Johnson mentioned that "the analytical expressions for these moments must be very complicated". From the observation that their numerical expression is "straightforward though tedious", [41] provided a graphical display over the parameter space of the first two moment functions of $L(\mu, \sigma^2)$, i.e., the logit-normal distribution for which $\nu = 1$. These moment functions were conveniently specified in terms of the signal-to-noise ratio μ/σ and the standard deviation σ of the normal family from which the logit-normal distributions are derived. The corresponding systematic numerical integration can be done using any regular software.

2.2 Application to wind power generation

Wind power generation is a double-bounded random variable which naturally belongs to $[0, 1]$ when scaled by the nominal power of the wind turbine, calling for S_B distributions. Note that the logit transformation, where $\nu = 1$, was used for wind power forecasting in [43]. Nevertheless, when applied to wind power generation the transformation should allow for asymmetry, as power curves for low and high power values show different inflections. First inspired by [10], the generalized logit transformation was proposed for wind power generation in [11]. Moreover, from the observation that when forecasting wind power generation the standard deviation of the errors is directly linked to their conditional expectation [44], [45], it is hoped that applying a generalized logit transformation to wind power generation would lower the effect, and allow predictive densities for which the variance can be assumed to be independent from the mean.

Wind power generation is a discrete-time stochastic process with continuous state space, i.e., a sequence of continuous random variables whose realizations define a time series. Let X_t be the wind power generation measured at time t. The series (x_t) is a series of dependent observations. The simplest dependency structure for a stochastic process is the Markov dependency, where the distribution of the observation x_t conditional on the past values only depends on a fixed number of past observations. One of the most common (linear) time series models is the AR(p) model, where AR stands for auto-regressive and p is the lag dependency on the past values. An AR(p) process is defined by the conditional (given the past) representation

$$X_t = \sum_{r=1}^{p} \lambda_r X_{t-r} + \epsilon_t, \tag{2.3}$$

where (ϵ_t) is a white noise. Therefore, the model assumes X_t to be a noisy linear combination of the previous p observations. Often, the white noise is assumed to be normally distributed. Note that the general AR(p) model is Markovian because the distribution of X_{t+1} only depends on the last p values. Sometimes an intercept term is included to indicate drift, but we ignore this for simplicity. A stochastic process $(X_t)_{t\in\mathcal{T}}$ is strictly stationary if the joint distributions of $(X_1, \ldots, X_k)$ and $(X_{1+h}, \ldots, X_{k+h})$ are the same for all k and $k + h \in \mathcal{T}$. A weaker version of stationarity, called second-order stationarity, imposes invariance in time only on the first two moments of the process. As shown in [46], an AR process can be second-stationary only if the roots of the polynomial

$$\mathcal{P}(x) = \prod_{r=1}^{p} (1 - \lambda_r x) \tag{2.4}$$

are all outside the unit circle in the complex plane. Note that to verify that a given vector $\Lambda = (\lambda_1, \ldots, \lambda_p)$ satisfies this condition requires to find the roots of the p-th degree polynomial $\mathcal{P}$,

and check that they are all of modulus greater than 1, which makes it difficult to find appropriate values of Λ for simulation purposes when p becomes large. For low values of p, simulation of stationary AR processes is easier, since when $p = 1$, λ_1 must satisfy $-1 < \lambda_1 < 1$; when $p = 2$, λ_1 and λ_2 must lie in the triangular region

$$\begin{aligned} \lambda_2 + \lambda_1 &< 1 \\ \lambda_2 - \lambda_1 &< 1 \\ -1 &< \lambda_2 < 1 \end{aligned} \tag{2.5}$$

for the process to be stationary [47].

In order to handle the dependency structure between successive realizations of wind power generation, we replace the constant location parameter μ of the GLN distribution by an AR(p) model. We only consider AR dynamics as results on offshore data have suggested that such dynamics are appropriate when considering short lead times [14]. This comes down to assuming the expectation of the normal transform Y_t to be

$$\mu_t = \mathbb{E}(Y_t) = \sum_{r=1}^{p} \lambda_r Y_{t-r}. \tag{2.6}$$

Let $\mathcal{F}_{t-1}$ be the previous information set, i.e., the σ-algebra generated by $X_1, \ldots, X_{t-1}$, and $\mathbf{x}_{t:(t-p)} = (x_t, \ldots, x_{t-p})$. The probability density function of the random variable X_t conditional on $\mathcal{F}_{t-1}$ is

$$p(x_t|\mathcal{F}_{t-1};\theta) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma^2}} \frac{\nu}{x_t\left(1-x_t^{\nu}\right)} \exp\left[-\frac{1}{2}\left(\frac{\gamma(x_t;\nu)-\mu_t}{\sigma}\right)^2\right] & \text{if } 0 < \mathbf{x}_{t:(t-p)} < 1, \\ 0 & \text{otherwise,} \end{cases} \tag{2.7}$$

where $\theta = (\Lambda, \sigma^2, \nu)$, $\Lambda = (\lambda_1, \ldots, \lambda_p)$.

Finally, note that while the support of the GLN densities in (2.2) and (2.7) is $(0, 1)$, wind power generation can actually take values 0 and 1. For simplicity and because we focus on the GLN distribution, we choose to look at the observation $x_t \in [0, 1]$ as a coarsened version of X_t [48]. This coarsened data framework has been formalized by [49] and [50]. An alternative could be to use Dirac delta masses located at 0 and 1 [11].

2.3 Maximum likelihood inference

In [11] the shape parameter ν of the GLN distribution was assumed to be a meta-parameter and chosen upon cross-validation once and for all from available data. Because for wind power applications ν is likely to vary from site to site, it was suggested as a concluding remark to develop a more rigorous framework, such as maximum likelihood estimation, in order to estimate the shape parameter along with the other parameters of the distribution. We introduce maximum likelihood estimation in [**Paper A**] for both stationary and non-stationary frameworks, along with corresponding batch and recursive algorithms.

2.3.1 Stationary framework

Let $\mathbf{x}_{1:T} = (x_1, \ldots, x_T)$ be an observed sequence of the stochastic process from distribution (2.7). The associated likelihood is given by

$$L(\theta|\mathbf{x}_{1:T}) = \prod_{t=p+1}^{T} p(x_t|\mathcal{F}_{t-1};\theta), \tag{2.8}$$

which is the probability of the observed sequence under model p_θ and the assumption that the variables X_t are independent and identically distributed given $\mathcal{F}_{t-1}$. One must think of $L(\theta|\mathbf{x}_{1:T})$ as a function of θ, the data $\mathbf{x}_{1:T}$ being fixed. Note that we do not take into account the distribution of the first p observed values $(x_1, \dots, x_p)$ and consider instead the likelihood conditional on them. Maximum likelihood inference is about recovering the value of θ that maximizes (2.8). The logarithm of $L(\theta|\mathbf{x}_{1:T})$ being easier to maximize, especially when exponential families are involved, maximization of the log-likelihood $l(\theta|\mathbf{x}_{1:T}) = \log L(\theta|\mathbf{x}_{1:T})$ is often preferred.

When maximizing $l(\theta|\mathbf{x}_{1:T})$, or equivalently minimizing the negative log-likelihood $-l(\theta|\mathbf{x}_{1:T})$, we nicely get closed-form solutions for Λ and σ^2 from the transform Y_t being normally distributed. They are the regular closed-form solutions of the normal distribution parameters, applied to Y_t. Let $\mathbf{y} = (y_{p+1}, \dots, y_T)$ and $\mathbf{Y}$ denote the matrix of stacked vectors $B\mathbf{y}, \dots, B^p\mathbf{y}$ where B is the backshift operator, $\mathbf{Y} \in \mathbb{R}^{(T-p)\times p}$. Computing the first derivatives of the negative log-likelihood with respect to the parameters of the distribution we can retrieve stationary points. As for the parameters Λ and σ^2, solving

$$-\frac{\partial}{\partial \Lambda} l(\theta|\mathbf{x}_{1:T}) = 0, \tag{2.9}$$

$$-\frac{\partial}{\partial \sigma^2} l(\theta|\mathbf{x}_{1:T}) = 0, \tag{2.10}$$

we get the (usual) maximum likelihood estimators

$$\hat{\Lambda} = (\mathbf{Y}^\top \mathbf{Y})^{-1} \mathbf{Y}^\top \mathbf{y}, \tag{2.11}$$

$$\hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{Y}\hat{\Lambda})^\top (\mathbf{y} - \mathbf{Y}\hat{\Lambda})}{T - p}. \tag{2.12}$$

Taking the first derivative of the negative log-likelihood with respect to ν we need to solve

$$-\frac{T-p}{\nu} - \sum_{t=p+1}^{T} \frac{x_t^\nu \log x_t}{1 - x_t^\nu} + \frac{(\mathbf{u} - \mathbf{U}\Lambda)^\top (\mathbf{y} - \mathbf{Y}\Lambda)}{\sigma^2} = 0, \tag{2.13}$$

where $\mathbf{u} = \frac{\partial}{\partial \nu}\mathbf{y}$, $\mathbf{U} = \frac{\partial}{\partial \nu}\mathbf{Y}$, $u_t = (1 + \exp y_t) \log x_t$. There is no closed-form maximum likelihood estimator for $\hat{\nu}$ and we need an iterative algorithm to solve (2.13). Note that these stationary points are (global) minimizers only if the negative log-likelihood is convex in θ.

Convexity in Λ and σ^2 It can easily be shown that $-l(\theta|\mathbf{x}_{1:T})$ is convex in Λ, as we have

$$-\frac{\partial^2}{\partial \Lambda^2} l(\theta|\mathbf{x}_{1:T}) = \frac{1}{\sigma^2} \mathbf{Y}^\top \mathbf{Y} \succ 0. \tag{2.14}$$

As for σ^2, we have

$$-\frac{\partial^2}{\partial (\sigma^2)^2} l(\theta|\mathbf{x}_{1:T}) = -\frac{T-p}{2\sigma^4} + \frac{(\mathbf{y} - \mathbf{Y}\Lambda)^\top (\mathbf{y} - \mathbf{Y}\Lambda)}{\sigma^6}. \tag{2.15}$$

Plugging (2.12) into (2.15) we get

$$-\frac{\partial^2}{\partial (\sigma^2)^2} l(\theta|\mathbf{x}_{1:T})\bigg|_{\hat{\theta}} = -\frac{T-p}{2\hat{\sigma}^4} + \frac{(T-p)\hat{\sigma}^2}{\hat{\sigma}^6} = \frac{T-p}{2\hat{\sigma}^4} > 0. \tag{2.16}$$

Convexity in ν It is more complicated to show convexity in the shape parameter ν. However, simulation work adapted to our framework allows us to be rather confident in assuming convexity

in the shape parameter as well, at least for reasonable values. We use $\|.\|$ to denote the Euclidean norm. The second derivative with respect to ν is

$$-\frac{\partial^2}{\partial\nu^2}l(\theta|\mathbf{x}_{1:T}) = \frac{T-p}{\nu^2} - \sum_{t=p+1}^{T}\log(x_t)^2\frac{x_t^\nu}{(1-x_t^\nu)^2} + \frac{(\mathbf{v}-\mathbf{V}\Lambda)^\top(\mathbf{y}-\mathbf{Y}\Lambda)}{\sigma^2} + \frac{\|\,\mathbf{u}-\mathbf{U}\Lambda\,\|^2}{\sigma^2}, \quad (2.17)$$

where $\mathbf{v} = \frac{\partial}{\partial\nu}\mathbf{u}$, $\mathbf{V} = \frac{\partial}{\partial\nu}\mathbf{U}$, $v_t = u_t \log x_t \exp y_t$. We have $\frac{\|\mathbf{u}-\mathbf{U}\Lambda\|^2}{\sigma^2} \geq 0$ and $\frac{T-p}{\nu^2} > 0$. We can also show that $\sum_{t=p+1}^{T}\log(x_t)^2\frac{x_t^\nu}{(1-x_t^\nu)^2}$ is upper bounded by $\frac{T-p}{\nu^2}$, since $\lim_{\substack{x_t\to 1\\ x_t<1}}\log(x_t)^2\frac{x_t^\nu}{(1-x_t^\nu)^2} = \frac{1}{\nu^2}$. Our simulation studies showed the remaining term $\frac{(\mathbf{v}-\mathbf{V}\Lambda)^\top(\mathbf{y}-\mathbf{Y}\Lambda)}{\sigma^2}$ to be always positive, as most of the elements of $\mathbf{v}-\mathbf{V}\Lambda$ and $\mathbf{y}-\mathbf{Y}\Lambda$ shared the same sign and when they did not, they were close to zero. We did not observe simulations suggesting otherwise. However, we did not run extreme scenarios, e.g., with values of ν very close to zero or very large. Finally, note that convexity in each parameter would not ensure the Hessian matrix to be positive semi-definite.

A class of efficient algorithms for minimizing convex functions are Newton methods [51]. They are gradient descent methods which use the Newton step as a descent direction. Newton's method in particular, also known as the Newton-Raphson method, is a powerful algorithm with super-linear convergence properties. However, it requires second-order information, i.e, to compute a Hessian matrix for dimensions larger than one. Moreover, the method requires this matrix to be positive definite, in order to ensure the Newton step is actually a descent direction. Other methods exist in the optimization literature, in particular cyclic coordinate ascent methods [34], [52]. Although they are well known for their simplicity and stability, these methods have been less preferred in practice since they converge only linearly. Because they have shown to be more stable for our purpose, the algorithm we propose for stationary frameworks in [**Paper A**] is derived from these methods. Instead of using a $(p+2)$-dimensional Newton-Raphson algorithm, each iteration of our conditional optimization algorithm consists in first computing the closed-form maximum likelihood estimates of Λ and σ^2 given the current value of ν, and then applying a (simpler) one-dimensional Newton step to improve the current estimate of ν, until we reach convergence. This is Algorithm 1 of [**Paper A**], also referred to as "Batch MLE with diagonalization".

2.3.2 Non-stationary frameworks

Algorithm 1 relies on the assumption that the stochastic process is (weakly) stationary, i.e., the parameter vector θ of the distribution in (2.2) does not change with time. However, for wind power generation, several works have shown that the parameter vector, in particular the location-related parameters, should be allowed to vary at different timescales [14], [53]–[55], as it would be tedious to properly account for (complicated) relationships to various features in parametric models.

In order to design a recursive algorithm which would enable to recursively update the parameter vector of the GLN distribution for wind power generation, we express the negative log-likelihood of $\mathbf{x}_{1:t}$, i.e., the sequence of observations until time t, as a time-dependent objective function to be minimized at time t. For ease of notation and because the negative log-likelihood is to be minimized with respect to θ, let $p(x_j|\mathcal{F}_{j-1};\theta) = p_j(\theta)$. The time-dependent negative log-likelihood to be minimized at time t is

$$-l_t(\theta) = -\frac{1}{n_\alpha}\sum_{j=p+1}^{t}\alpha^{t-j}\log p_j(\theta), \quad (2.18)$$

where $\alpha \in (0,1)$ is an exponential forgetting factor which puts more weight onto the most recent observations, and $n_\alpha = \frac{1}{1-\alpha}$ is a constant normalizing the weighted negative log-likelihood. Let $\hat{\theta}_t$ be the estimate of the parameter vector at time t. A recursive maximum likelihood procedure relies on a Newton step for obtaining the new estimate $\hat{\theta}_t$ from the previous estimate $\hat{\theta}_{t-1}$ [13], [14]. Applying one Newton step at time t we have

$$\hat{\theta}_t = \hat{\theta}_{t-1} - \frac{\nabla_\theta l_t(\hat{\theta}_{t-1})}{\nabla^2_\theta l_t(\hat{\theta}_{t-1})}. \tag{2.19}$$

Let $\mathbf{h}_t = \nabla_\theta p_t(\hat{\theta}_{t-1})$ and $\hat{\mathbf{R}} = -\nabla^2_\theta l_t(\hat{\theta}_t)$. The recursive estimation relies on a few additional assumptions which are classic in the online learning framework. First, assuming $\hat{\theta}_{t-1}$ minimizes $-l_{t-1}(\theta)$ we get

$$\nabla_\theta l_t(\hat{\theta}_{t-1}) = (1-\alpha)\mathbf{h}_t. \tag{2.20}$$

Then, assuming p_t is (almost) linear in θ in the neighborhood of θ_{t-1} we get the approximation $\nabla^2_\theta \log p_t(\hat{\theta}_{t-1}) = -\mathbf{h}_t\mathbf{h}_t^\top$, which is the key for ensuring that the approximate $\hat{\mathbf{R}}_t$ of the Hessian matrix is always positive definite. Finally, we assume that the objective function $-l_t$ is smooth in the vicinity of $\hat{\theta}_t$ and the adaptation step small enough so that $\hat{\mathbf{R}} = -\nabla^2_\theta l_t(\hat{\theta}_t) \approx -\nabla^2_\theta l_t(\hat{\theta}_{t-1})$. This is a classic assumption for deriving recursive estimation methods for stochastic systems [56]. The two-step recursive scheme at time t is then

$$\hat{\mathbf{R}}_t = \alpha\hat{\mathbf{R}}_{t-1} + (1-\alpha)\mathbf{h}_t\mathbf{h}_t^\top, \tag{2.21}$$

$$\hat{\theta}_t = \hat{\theta}_{t-1} + (1-\alpha)\hat{\mathbf{R}}_t^{-1}\mathbf{h}_t. \tag{2.22}$$

This is Algorithm 2 of [**Paper A**], also referred to as "Recursive MLE". Note that an algorithm based upon such a recursive scheme might face computational issues as it requires inverting a matrix, the information matrix $\hat{\mathbf{R}}_t$, at each iteration. This can be prevented by working directly with the matrix inverse, the covariance matrix $\hat{\mathbf{P}}_t$, which can be computed by using the matrix inversion rule. Such a matrix $\hat{\mathbf{P}}_t$ is used in [**Paper B**]. Finally, note that this algorithm is a quasi-Newton approach in the sense that it approximates the Hessian matrix with a positive definite matrix, thanks to first-order information only, contrary to the Newton-Raphson method used for retrieving an estimate of ν in Algorithm 1.

2.4 Assessment of predictive densities

In [**Paper A**] Algorithms 1 and 2 are run on the average power generation at Anholt wind farm, in order to estimate either a static or dynamic parameter vector. Some hyper-parameters need to be decided prior to running the algorithms. They are the coarsening parameter, see Section 2.2, which we will denote by δ, the lag dependency p of the AR process, see (2.3) and (2.7), and the exponential forgetting factor α for Algorithm 2, see (2.18). We select them by performing cross-validation on a validation set. For such a task, we need to decide on one score we wish to minimize over the set of hyper-parameters, that we will also use to ultimately evaluate our forecasts. As we issue both point and probabilistic forecasts, we need to use one score for point forecasting and one score for probabilistic forecasting. The score we choose needs to be easy to compute and to look at, so we can decide on a set of hyper-parameters in a relatively automated way and in a reasonable computation time. It will be used when evaluating the forecasts as we select the best set of hyper-parameters in order to have the best forecasts according to a specific score. Last but not least, it needs to match the forecast we are to issue. Because our point forecasts

are expectation-based forecasts, a consistent score is the root mean square error. If they were to be median-based forecasts, an appropriate choice would rather be the mean absolute error. Since we are primarily concerned with probabilistic forecasting of wind power generation, we focus in the following on the main scores and tools for the assessment of predictive densities. Nonetheless, we want to emphasize the importance of carefully matching the scoring function and the forecasting task also for point forecasting, and we refer the reader to [57] for details.

2.4.1 Scoring rules

A set of personal probabilities as theorized by [58] and [59] is an important input to decision analysis. The elicitation of personal probabilities received considerable attention, and various methods were then developed to help an individual in assessing personal probabilities. Scoring rules belong to elicitation procedures and were designed in order to encourage a forecaster to honestly reveal its true beliefs. This is known as the ex ante role of scoring rules, or role in terms of elicitation. The ex post evaluation role takes place after one knows what actually happened and is to measuring the *goodness* of the forecasts, so that accurate forecasts are rewarded and inferior ones are penalized. The two roles of scoring rules are related, since the ex ante incentive comes from the anticipation of the ex post evaluation [60]. A scoring rule can be any function of the probability forecasts and the observations. However, because of the ex ante incentive, one should only be interested in what is known as (strictly) proper scoring rules, for which a forecaster can minimize their expected score, when negatively oriented, only by issuing honest probability forecasts.

The development of scoring rules was at first generally restricted to individual probabilities or discrete probability distributions. Scoring rules for continuous probability distributions were then designed from scoring rules for binary situations [61]. Let y be a continuous random variable, p the forecaster's predictive density for y and r the predictive density the forecaster actually reports. If a scoring rule gives a score $S_x(r)$ when $y = x$, the forecaster's expected score is

$$\mathbb{E}_p[S(r)] = \int_{-\infty}^{\infty} S_x(r)p(x)dx. \tag{2.23}$$

By analogy with point forecasting, we use scores from scoring rules in negative orientation, the lower the better. Note that both orientations can be encountered. The scoring rule S is *proper* if

$$\mathbb{E}_p[S(r)] \geq \mathbb{E}_p[S(p)] \quad \text{for all } r,\ p. \tag{2.24}$$

It is *strictly proper* if (2.24) holds with equality if and only if $r = p$. Strictly proper scoring rules for continuous random variables include the logarithmic score [62]

$$S_x(r) = -\log r(x), \tag{2.25}$$

which seems rather suited to frameworks where the predictive densities have been estimated through maximum likelihood inference. Such a score is a *local* score for it depends on the predictive density r only through its value at the event x. It is not sensitive to distance as no reward is granted for assigning high probabilities to values close to, but different from x. Moreover, often predictive distributions are not issued as predictive densities, but rather as samples for which scores like (2.25) cannot be used. Other scores have been proposed, that are sensitive to distance [63], [64]. One of the most popular scoring rules nowadays, which addresses distance sensitivity and can be defined in terms of predictive cumulative distribution functions, is the continuous ranked probability score,

i.e., the CRPS [35], [61]. Let $\mathcal{P}$ consist of the Borel probability measures on $\mathbb{R}$, and a probabilistic forecast in $\mathcal{P}$ be identified with its cumulative distribution function F. The CRPS is defined as

$$\mathrm{CRPS}_x(F) = \int_{-\infty}^{\infty} (F(z) - \mathbb{1}_{z \geq x})^2 dz. \tag{2.26}$$

The CRPS is proper relative to the class $\mathcal{P}$ and strictly proper relative to the subclass $\mathcal{P}_1$ of the Borel probability measures that have finite first moment [35]. The CRPS is the scoring rule we use to evaluate our predictive densities in [**Paper A**], [**Paper B**] and [**Paper C**]. Often it has a closed-form expression, when F belongs to a certain family of distributions [53], [65], [66]. This is not the case for GLN distributions, for the same reason the first moments are not available either, see Section 2.1. However, if the predictive distribution takes the form of a sample of size N, then the right side of (2.26) can be evaluated in $\mathcal{O}(N \log N)$ operations using order statistics [67]. This is the decomposition method we use for computing the CRPS, as it can be applied to any kind of predictive distributions we wish to compare our predictive densities to, as soon as they take the form of a sample.

2.4.2 Sharpness and calibration

Scoring rules are attractive measures of predictive performance for they evaluate *calibration* and *sharpness* simultaneously. Calibration is about the statistical consistency between the probabilistic predictions and the values which are actually observed. Therefore, it is a joint property of the forecasts and the observations. On the other hand, sharpness is a property of the forecasts only, for it refers to the concentration of the predictive distributions, the sharper the more informative.

Calibration Probabilistic calibration is the most widely considered notion of calibration. Let F_t be a predictive cumulative distribution function, and an outcome x_t be a random number with distribution G_t. The probability integral transform (PIT) value is the value that the predictive distribution F_t takes at the observation x_t,

$$p_t = F_t(x_t), \tag{2.27}$$

see, e.g., [1], [68], [69]. If the forecasts are ideal and F_t is continuous, then the PIT value p_t has a uniform distribution. Uniformity can be checked empirically by looking at PIT histograms [70]–[72], where 10 or 20 bins generally seem adequate. Asymptotic uniformity of the empirical sequence p_t for probabilistic forecasts of continuous variables was characterized by [36]. Because the uniformity of the PIT values is a necessary but not sufficient condition for a forecaster to be ideal [73], work [36] also proposed three major modes of calibration for probabilistic forecasts of continuous variables: probabilistic, exceedance and marginal calibration. Let

$$\bar{G}(x) = \lim_{T \to \infty} \left\{ \frac{1}{T} \sum_{t=1}^{T} G_t(x) \right\}, \tag{2.28}$$

and

$$\bar{F}(x) = \lim_{T \to \infty} \left\{ \frac{1}{T} \sum_{t=1}^{T} F_t(x) \right\}. \tag{2.29}$$

Marginal calibration requires that the limit distributions $\bar{G}$ and $\bar{F}$ exist and are equal to each other. To empirically assess marginal calibration, one can compare the average predictive cumulative distribution function

$$\bar{F}_T(x) = \frac{1}{T} \sum_{t=1}^{T} F_t(x), \quad x \in \mathbb{R}, \tag{2.30}$$

with the empirical cumulative distribution function of the observations

$$\hat{G}_T(x) = \frac{1}{T}\sum_{t=1}^{T} G_t(x), \quad x \in \mathbb{R}. \tag{2.31}$$

We use marginal calibration plots which show the difference

$$\bar{F}_T(x) - \hat{G}_T(x), \quad x \in \mathbb{R}, \tag{2.32}$$

for the forecasters in the simulation study and use cases of [**Paper A**] and [**Paper B**]. From a practical point of view, $\bar{F}_T(x)$ and $\hat{G}_T(x)$ are computed over a grid of values of x. In the case of wind power generation, we have $x \in [0, 1]$ and use a grid which is fine enough to capture the behaviour of the distributions near the bounds, typically $x = 0, 0.01, \ldots, 0.99, 1$. To assess probabilistic calibration, we use PIT histograms in [**Paper B**] and reliability diagrams in [**Paper A**], which are an equivalent cumulative version of PIT histograms [74].

Sharpness Sharpness is a characteristic of the predictive distributions only. Probabilistic forecasts should not only be calibrated, they should also be as informative/sharp as possible [3]. Sharpness of predictive distributions can be assessed by looking at the width of prediction intervals, on average or using, for instance, box plots [36], [75].

The paradigm of *maximizing the sharpness of the predictive distributions suject to calibration* was proposed in [36], under the conjecture that ideal forecasts and the maximization of sharpness subject to calibration are equivalent. Not only scoring rules are measures of both calibration and sharpness, when they are proper for the class of predictive distributions at hand, they also incentivize truthful (calibrated) and informative forecasts [76].

CHAPTER 3

Generalized logit-normal densities with a varying bound

In Chapter 2 and [**Paper A**] the support of the GLN distributions is assumed to be $(0, 1)$, according to the common definition. However, this support could be changed to any interval (a, b) by re-scaling the bounded random variable. In this chapter, we investigate a new assumption stating that having fixed values for the bounds which define the support of a bounded distribution is limiting, and that these bounds should be allowed to vary while not being observed. We focus here on the upper bound of the interval, as it is the one we are particularly interested in for wind power forecasting. First, we introduce two possible statistical frameworks when having a varying upper bound. Then, we address the first one for which the upper bound is a parameter, as in [**Paper B**], and the second one for which the upper bound is a discrete latent variable, as in [**Paper C**].

3.1 A varying upper bound

The support of any bounded distribution in the S_B family can be changed to (a, b), where $a, b \in \mathbb{R}$, by scaling a bounded variable $X_t \in (0, 1)$, i.e., through the transformation $X_t := (b - a)X_t + a$. However, when doing so, the bounds a and b of the interval are always assumed to be fixed and do not vary over time or depending on exogenous variables. We argue that this assumption is inappropriate for wind power generation, as for instance curtailment actions may impact the upper bound b to be lower than 1. In fact, an upper bound greater than 1 can also happen as some tests are sometimes run on wind turbines that allow them to produce beyond their nominal power. We are concerned by varying bounds which cannot be observed, or not reliably, and need to be estimated from the data as well.

So far we have always assumed that we were to observe the realizations $x_t \in [0, 1]$ of $X_t \in (0, 1)$. We challenge this implicit assumption by suggesting that we in fact may have been observing the realizations x_t/b_t of $X_t/b_t \in (0, 1)$, where b_t does not equal 1 anymore, and may even vary with t. We focus on an upper bound b_t but it would be straightforward to apply this new framework to a lower bound a_t. There are at least two ways of dealing with a varying b_t. In the specific context of stochastic processes, a first way is to consider the upper bound as a scaling parameter b of a parametric bounded distribution which belongs to S_B, in our case the GLN distribution, to include it in the parameter vector θ of the distribution, and to assume a non-stationary framework for the process. Then, we would consider online learning algorithms so that the parameter vector θ can evolve over time as in Section 2.3.2. The upper bound as a parameter is addressed in Section 3.2, and refers to [**Paper B**].

In a more general framework, an alternative is to consider the bounds as missing random variables which the distribution of the bounded variable X_t is conditional on. The main advantage of this

approach is its generality and flexibility, with a latent upper bound B_t being distributed according to a well-specified probability distribution, which might depend on exogenous variables. Because we do not have access to the realizations of the bound, the maximization of the likelihood function of the realizations x_t of X_t might involve complicated high-dimensional integration, possibly computationally infeasible, and would therefore call for EM-based algorithms [18]. Moreover, with such a method and for forecasting applications, one needs to first compute (good enough) forecasts of the bounds in order to be able to forecast the response variable. This framework makes it possible to specify any kind of model for an upper bound B_t. However, how simple or complicated the maximum likelihood inference through the EM algorithm will be depends on this specification. A simple model for B_t will make the inference easier but will likely not perform well when moving to forecasting. We propose to model B_t as a discrete random variable taking a finite number of values over $(0, 1]$. This proposition is presented in Section 3.3 and refers to [**Paper C**].

3.2 Bounds as parameters

Let us rescale $X_t \in (0, 1)$ to the interval $(0, b)$ by applying the transformation $X_t := bX_t$. The generalized logit transform $Y_t \in \mathbb{R}$ of $X_t \in (0, b)$ is given by

$$Y_t = \gamma(X_t/b; \nu) = \log \frac{(X_t/b)^\nu}{1 - (X_t/b)^\nu}. \tag{3.1}$$

The expectation of Y_t is now $\mu_t = \sum_{r=1}^{p} \lambda_r \gamma(x_{t-r}/b; \nu)$ and the probability density function of X_t conditional on the previous information set $\mathcal{F}_{t-1}$ becomes

$$p(x_t|\mathcal{F}_{t-1}; \theta) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma^2}} \frac{\nu}{x_t(1-(x_t/b)^\nu)} \exp\left[-\frac{1}{2}\left(\frac{\gamma(x_t/b;\nu)-\mu_t}{\sigma}\right)^2\right] & \text{if } 0 < \mathbf{x}_{t:(t-p)} < b, \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

where $\theta = (\Lambda, \sigma^2, \nu, b)$.

3.2.1 Quasiconvex optimization

The first challenge we face when considering the upper bound b as an additional (scaling) parameter of the GLN distribution is that the negative log-likelihood minimization problem at hand is not convex anymore, since the negative log-likelihood is not convex in b. Recall that when assuming stationarity, maximum likelihood estimation of the parameter vector θ consists of solving the optimization problem

$$\min_{\theta} \quad -\sum_{t=p+1}^{T} \log p(x_t|\mathcal{F}_{t-1}; \theta). \tag{3.3}$$

Simulation studies suggest that while not convex in b, the negative log-likelihood to be minimized in (3.3) might still have a global minimum in θ. A broader class of functions which include convex functions as a subclass are quasiconvex functions. For simplicity, let us assume that functions are differentiable. From [51], a definition of quasiconvexity is

Definition 3.2.1 (Quasiconvexity) *A function $f : \mathbb{R}^d \to \mathbb{R}$ is called quasiconvex (or unimodal) if its domain and all its sublevel sets*

$$S_\alpha = \{\mathbf{x} \in \mathbf{dom}\ f \mid f(\mathbf{x}) \leq \alpha\},$$

for $\alpha \in \mathbb{R}$, are convex.

Quasiconvexity is a considerable generalization of convexity. Still, many of the properties of convex functions hold or have analogs for quasiconvex functions. Since all convex functions are quasiconvex, the negative log-likelihood in (3.3) is quasiconvex in Λ and σ^2. We focus then on quasiconvexity in ν and b.

Quasiconvexity in ν Keeping only the terms that depend on ν we get

$$\begin{aligned} -\sum_{t=p+1}^{T} \log p(x_t|\mathcal{F}_{t-1};\theta) = &-(T-p)\log\nu + \sum_{t=p+1}^{T}\log(1-(x_t/b)^{\nu}) \\ &+ \frac{1}{2\sigma^2}\sum_{t=p+1}^{T}\left(\log\frac{(x_t/b)^{\nu}}{1-(x_t/b)^{\nu}} - \mu_t\right)^2 + C, \end{aligned} \tag{3.4}$$

where C is a constant. In Definition 3.2.1, that a sublevel set S_α is convex means if $\mathbf{x}_1, \mathbf{x}_2 \in S_\alpha$, then $w\mathbf{x}_1 + (1-w)\mathbf{x}_2 \in S_\alpha$ for all $w \in [0,1]$, or equivalently $\mathbf{x} \in [\mathbf{x}_1, \mathbf{x}_2]$ is also in S_α. Let $\alpha = \alpha_a + \alpha_b + \alpha_c$. It is straightforward to show that for all $\nu_1, \nu_2 \in S_\alpha$, $\nu \in [\nu_1, \nu_2]$,

$$-(T-p)\log\nu_2 \leq -(T-p)\log\nu \leq -(T-p)\log\nu_1 \leq \alpha_a, \tag{3.5}$$

and

$$\sum_{t=p+1}^{T}\log(1-(x_t/b)^{\nu_1}) \leq \sum_{t=p+1}^{T}\log(1-(x_t/b)^{\nu}) \leq \sum_{t=p+1}^{T}\log(1-(x_t/b)^{\nu_2}) \leq \alpha_b, \tag{3.6}$$

so that

$$-(T-p)\log\nu + \sum_{t=p+1}^{T}\log(1-(x_t/b)^{\nu}) \leq \alpha_a + \alpha_b. \tag{3.7}$$

When μ is constant, it is also straightforward to show

$$\begin{aligned} \frac{1}{2\sigma^2}\sum_{t=p+1}^{T}\left(\log\frac{(x_t/b)^{\nu_2}}{1-(x_t/b)^{\nu_2}} - \mu\right)^2 &\leq \frac{1}{2\sigma^2}\sum_{t=p+1}^{T}\left(\log\frac{(x_t/b)^{\nu}}{1-(x_t/b)^{\nu}} - \mu\right)^2 \\ &\leq \frac{1}{2\sigma^2}\sum_{t=p+1}^{T}\left(\log\frac{(x_t/b)^{\nu_1}}{1-(x_t/b)^{\nu_1}} - \mu\right)^2 \\ &\leq \alpha_c. \end{aligned} \tag{3.8}$$

This is not straightforward anymore when μ is replaced with $\mu_t = \sum_{r=1}^{p}\lambda_r \log\frac{(x_{t-r}/b)^{\nu}}{1-(x_{t-r}/b)^{\nu}}$. However, note that $\sum_{t=p+1}^{T}\left(\log\frac{(x_t/b)^{\nu}}{1-(x_t/b)^{\nu}} - \mu_t\right)^2$ is an estimate of $(T-p)\times\sigma^2$ at, or close enough to the minimum $\hat{\theta}$, so that the third term simplifies to $(T-p)/2$ and does not depend on ν anymore. It then seems reasonable to assume the negative log-likelihood to be quasiconvex in ν at least locally, i.e., not too far from the minimum.

Quasiconvexity in b The same reasoning can be applied to b as

$$\sum_{t=p+1}^{T}\log(1-(x_t/b_1)^{\nu}) \leq \sum_{t=p+1}^{T}\log(1-(x_t/b)^{\nu}) \leq \sum_{t=p+1}^{T}\log(1-(x_t/b_2)^{\nu}) \leq \alpha_b. \tag{3.9}$$

Even if the negative log-likelihood still has a (global) minimum, we cannot use a Newton step as in Chapter 2 for the Hessian matrix is not positive definite anymore. First-order methods such as ordinary gradient descent with fixed step sizes are known to perform poorly on quasiconvex

functions for which the gradient might be too small in a plateau area of the function, or explode in cliff areas. For stationary frameworks, a pioneering paper by [77] was the first to propose an efficient algorithm, NGD, and to prove that this algorithm converges to an ϵ-optimal solution within $\mathcal{O}(1/\epsilon^2)$ iterations given a differentiable quasiconvex objective function. There have been several attempts to tackle plateaus and cliffs among the deep learning community. However, those works do not provide a theoretical analysis showing better convergence guarantees than NGD.

NGD is similar to gradient descent, except the gradient is normalized. One can feel that to achieve robustness to plateaus and cliffs, we must ignore the size of the gradient; it is more surprising that the information in the direction of the gradient is enough to guarantee convergence. A more recent work extended quasiconvexity by introducing *local-quasiconvexity* to capture functions which are not quasiconvex in the former sense, and proved that NGD finds an ϵ-optimal minimum for such functions within $\mathcal{O}(1/\epsilon^2)$ iterations as well [37].

3.2.2 Learning and forecasting outside of a support

The negative log-likelihood we wish to minimize in (3.3) becomes infinite as soon as an observation x_t is outside of the support of p in (3.2), i.e., is greater or equal than b. This is an implicit constraint on b when estimating $\hat{\theta}$. Because we want b to be able to adapt to a non-stationary stochastic process, we do not need nor want b to be greater than all the observations x_t in the past. Recall the time-dependent negative log-likelihood (2.18) in Chapter 2 and let now the parameter vector θ include b, $\theta = (\Lambda, \sigma^2, \nu, b)$. Let $U_t = \{p+1, \dots, t\}$, $C_t(\theta) = \{j \in U_t \mid x_{j-r} < b, r = 0, \dots, p\}$ and $\overline{C_t}(\theta) = \{j \in U_t \mid j \notin C_t(\theta)\}$ be the complement of $C_t(\theta)$ in U_t. The log-likelihood takes finite values only for observations x_j such that $j \in C_t(\theta)$. Therefore, we can informally rewrite $\sum_{j=p+1}^{t} \alpha^{t-j} \log p_j(\theta)$ as $\sum_{j \in C_t(\theta)} \alpha^{t-j} \log p_j|_b(\theta) + \sum_{j \in \overline{C_t}(\theta)} \alpha^{t-j} \log 0$, where $p_j|_b(\theta)$ is the probability density function $p_j(\theta)$ restricted to its support $(0, b)$. When estimating the parameter vector θ_t over time, we need to take into account all the observations in the past, i.e., even the observations for which the log-likelihood does not take a finite value, i.e., the observations x_j such that $j \notin C_t(\theta)$. We propose to replace the value 0 in $\log 0$, which originally corresponds to the value of $p_j(\theta)$ outside of its support, with a sigmoid function of $b - x_j$,

$$s_j(b) = \frac{1}{1 + \exp(-b + x_j)}. \tag{3.10}$$

The function s_j is illustrated in Figure 3.1. It can be seen as the probability of x_j to be lower or equal than b: we have $s_j(b) \to 0^+$ when $x_j >> b$ and $s_j(b) \to 1^-$ when $x_j << b$. Moreover, $-\log s_j(b)$ is convex and differentiable in b. Hence, the *extended* time-dependent negative log-likelihood we propose to minimize is

$$-l_t^{\infty}(\theta) = -\frac{1}{n_\alpha} \left[\sum_{j \in C_t(\theta)} \alpha^{t-j} \log p_j(\theta) + \sum_{j \in \overline{C_t}(\theta)} \alpha^{t-j} \log s_j(b) \right]. \tag{3.11}$$

Note that $j \in \overline{C_t}(\theta)$ does not necessarily mean we have $x_j \geq b$, since this may happen because of a lagged observation $x_{j-r} \geq b$. In such a case, i.e., $j \in \overline{C_t}(\theta)$ and $x_j < b$, the observation x_j will still increase the value of the total log-likelihood compared to the event $\{x_j \geq b\}$, which is also a nice feature of choosing this specific function s_j.

In order to be able to track the bound parameter over time, we have introduced the extended time-dependent negative log-likelihood $-l_t^{\infty}$ and allowed b_t to vary on $\mathbb{R}$, which makes sense from

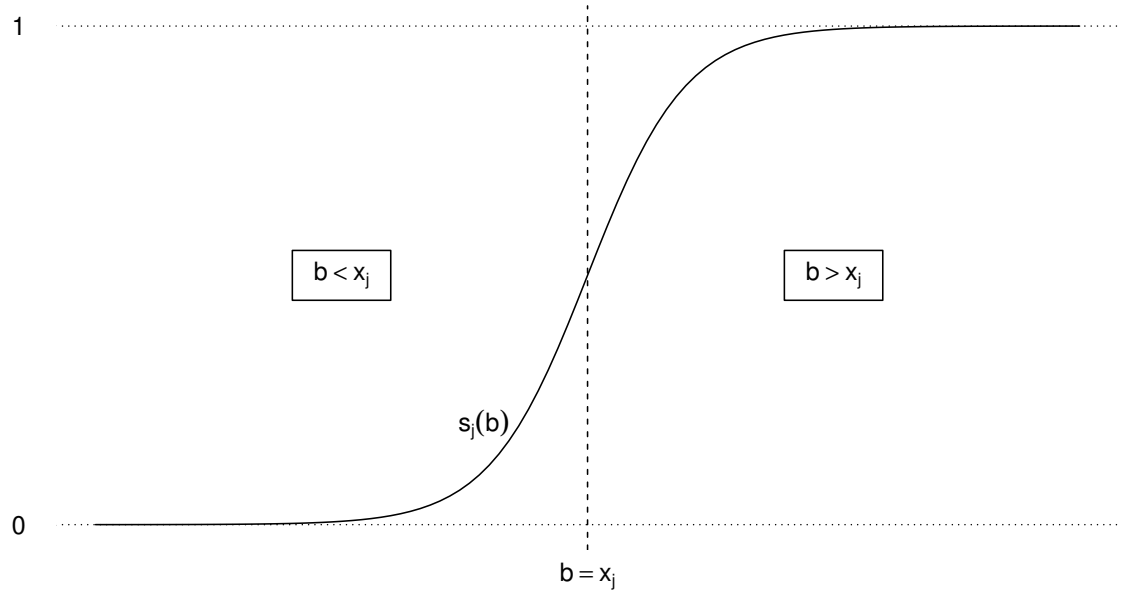


Figure 3.1: Sigmoid function $s_j(b)$ on the real line.

an inference point of view. Because we work with sequences of dependent observations, when moving to forecasting at time t the distribution of the bounded variable X_{t+1}, we need the current value b_t to be greater than all p former observed values $x_t, \dots, x_{t-p+1}$, for the expected value of X_{t+1} to exist. Therefore, we introduce a projection step, and project the current estimate $\hat{b}_t$ on the convex set $(\max(x_t, \dots, x_{t-p+1}), +\infty)$. We get the projected parameter

$$\tilde{b}_t = \Pi_{\mathcal{K}}(\hat{b}_t) = \begin{cases} \max(x_t, \dots, x_{t-p+1}) + \delta & \text{if } \max(x_t, \dots, x_{t-p+1}) \geq \hat{b}_t, \\ \hat{b}_t & \text{if } \max(x_t, \dots, x_{t-p+1}) < \hat{b}_t. \end{cases}$$

Note that we need to introduce a small $\delta > 0$ as we project $\hat{b}_t$ on an open convex set. When looking at the observation x_t as a coarsened version of X_t, δ can be seen as a coarsening parameter, see Section 2.2. Note that whereas being somehow optimized through cross-validation in [**Paper A**], we fix $\delta = 0.001$ in [**Paper B**]. Indeed, while it can be seen as a hyper-parameter worth tuning for real-world applications, this is a subsidiary matter in [**Paper B**] and we would rather choose a sensible value once and for all than carrying an extra hyper-parameter.

The predictive probability distributions obtained out of this methodology are to be evaluated and compared to classic benchmarks, in particular benchmarks that assume $b = 1$. As we work with predictive densities, one could think of using the logarithmic score, which is a strictly proper scoring rule relative to all measures that are absolutely continuous, at least to compare the predictive densities provided by Algorithm 2 of [**Paper A**] and algorithms minimizing (3.11). However, in our framework it can always happen that x_{t+1} falls outside of the support of the predictive density $\hat{p}_{t+1}$ we have issued at time t, when the current estimate of b is too low. In such a case, the logarithmic

score is $-\log \hat{p}_{t+1}(x_{t+1}) = -\log 0 = +\infty$, which is not suitable. In contrast, the CRPS gives

$$\mathrm{CRPS}_{x_{t+1}}(\hat{F}_{t+1}) = \int_{-\infty}^{b_t} \left(\hat{F}_{t+1}(z) - \mathbb{1}_{z \geq x_{t+1}}\right)^2 dz + \int_{b_t}^{\infty} \left(1 - \mathbb{1}_{z \geq x_{t+1}}\right)^2 dz, \tag{3.12a}$$

$$= \int_{-\infty}^{b_t} \hat{F}_{t+1}(z)^2 dz + \int_{b_t}^{x_{t+1}} 1 \; dz + \int_{x_{t+1}}^{\infty} 0 \; dz, \tag{3.12b}$$

$$= \int_{-\infty}^{b_t} \hat{F}_{t+1}(z)^2 dz + x_{t+1} - b_t, \tag{3.12c}$$

where $\hat{F}_{t+1}$ is the predictive cumulative distribution function we issue at time t and x_{t+1} is the value that realizes outside of the support of the predictive distribution. We see in (3.12c) that the CRPS is indeed increased by an observation falling outside of the support of the predictive distribution, but to a higher finite value, contrary to the logarithmic score which becomes infinite.

3.2.3 Online learning and stochastic approximation

NGD can be applied to minimize (3.11). However, it is a batch algorithm, designed for stationary frameworks. Therefore, it would require to run until convergence for each time t to estimate θ_t. It can still be used at successive intervals, if θ_t does not change too quickly. For instance, when applied to wind power generation at a temporal resolution of every ten minutes, if we run NGD every 500 time steps, we get a new $\hat{\theta}_t$ every 3.5 days. NGD can also be used as a warm-up for subsequent online learning algorithms, i.e., first run on a small batch of data points to provide better estimates for online learning algorithms to start with.

We present learning algorithms within the framework introduced by [78]. The goal of a learning system is to find the minimum of an *expected risk function*

$$J(\theta) = \mathbb{E}_x(Q(x, \theta)) = \int Q(x, \theta) dP(x), \tag{3.13}$$

where θ is the part of the learning system which must adapt to observing events x. The *loss function* $Q(x, \theta)$ measures the performance of the learning system with parameter θ under the circumstances described by event x. The occurrence of the events x is modelled as random independent observations drawn from an unknown probability distribution $dP(x)$. Using successive observations x_t, the learning system is to uncover a part of this unknown *truth* in the form of parameter values $\hat{\theta}_t$ that hopefully decrease $J(\hat{\theta}_t)$. The expected risk (3.13) cannot be minimized directly because the distribution is unknown. However, general theorems show that minimizing the *empirical risk*

$$\hat{J}_T(\theta) = \frac{1}{T} \sum_{t=1}^{T} Q(x_t, \theta) \tag{3.14}$$

can provide a good estimate of the minimum of $J(\theta)$, for a large enough T [79]. Our learning problem is a problem of maximum likelihood inference where we assume P to be a GLN distribution and wish to learn its parameter vector θ through minimizing the empirical risk

$$-\frac{1}{T} \sum_{t=p+1}^{T} \log p(x_t | \mathcal{F}_{t-1}; \theta), \tag{3.15}$$

since multiplying the negative log-likelihood of a sample $\mathbf{x}$ by $1/T$ does not change the optimization problem. Unlike standard, batch algorithms which use the whole sample $\mathbf{x}$ to return an hypothesis $\hat{\theta}$ from the minimization of (3.15), at time t online learning algorithms take in a former hypothesis

$\hat{\theta}_t$ and a single example x_t as inputs, to return a new hypothesis $\hat{\theta}_{t+1}$ [80]. They are closely related to *stochastic approximation* algorithms [81], which were first introduced as recursive adaptive algorithms [82]. Online learning typically addresses *successive* observations, with more and more applications due to the increasing number of continuous streams of data which need to be processed online. On the other hand, stochastic approximation is often referred to when dealing with (very) large datasets. There is no reference to *sequences* of data per se. It is the large number T of observations which make it prohibitive to minimize the expected risk function at once, and one will rather choose an observation x at random out of an available sample $\mathbf{x}$, and iteratively update θ using $Q(x, \theta)$ only. What is fundamentally common is that the observed events need to be independent. Therefore, as long as our framework is designed so that our observations can be considered to be the realizations of random (conditional) independent variables, both online learning and stochastic optimization methods can be seen as interchangeable [80], [81], [83]. In [81] the convergence of online learning algorithms is analyzed using stochastic approximation theory, and proved under very weak conditions, e.g., without requiring convexity. Note that when studying the convergence of stochastic algorithms, or regrets in the online learning paradigm, a key condition is often that the loss function needs be bounded [80], [81].

A seminal algorithm in both domains is online gradient descent (OGD) in the online learning paradigm and stochastic gradient descent (SGD) for stochastic optimization. Recall the iterative optimization methods discussed in Section 2.3, in particular gradient descent methods. They consist of updating the current value of the parameter θ by taking a step in a descent direction, according to the gradient of the loss function over the complete set of observations $\mathbf{x}$:

$$\hat{\theta}_{i+1} = \hat{\theta}_i - \eta_i \frac{1}{T} \sum_{i=1}^{T} \nabla_\theta Q(x_t, \hat{\theta}_i), \tag{3.16}$$

where i is the current iteration and η_i the *step size*, or *learning rate* at iteration i. Online/stochastic gradient descent is obtained by dropping the averaging operation in (3.16) and updating the parameter θ according to

$$\hat{\theta}_{i+1} = \hat{\theta}_i - \eta_i \nabla_\theta Q(x_i, \hat{\theta}_i), \tag{3.17}$$

where x_i has been chosen randomly for SGD, $i := t$ and x_t are successive observations for OGD. The simplification relies on the hope that the random noise introduced by this procedure will not compromise the average behavior of the algorithm. For convex and Lipschitz loss functions, SGD is guaranteed to find an ϵ-optimal solution within $\mathcal{O}(1/\epsilon^2)$ iterations and requires only an unbiased estimator for the gradient $\nabla_\theta J(\theta)$, which is obtained with only one, as in (3.17), or a few data samples, as in

$$\hat{\theta}_{i+1} = \hat{\theta}_i - \eta_i \frac{1}{m} \sum_{j=1}^{m} \nabla_\theta Q(x_j, \hat{\theta}_i), \tag{3.18}$$

also known as the updating scheme of *mini-batch* gradient descent (with a mini-batch of size m). In order to reach convergence, the step size in (3.17) and (3.18) is often set to decrease with i. When used for tracking a time-varying parameter θ_t, a standard approach consists of taking a fixed step size, so the algorithm can adapt to changes in the process. Choosing an appropriate step size is difficult: if η is too large the statistical fluctuations around θ_t may be important, while the algorithm will lose its tracking ability for too small values of η. This is a typical bias-variance trade-off [84], a tracking/accuracy compromise in the context of adaptive algorithms [85].

In [37] the authors not only extended theoretical results for NGD, they also introduced a new setup: stochastic optimization of locally quasiconvex functions. For this setup they propose a

stochastic version of NGD, i.e., SNGD, and show it converges to an ϵ-optimal minimum in $\mathcal{O}(1/\epsilon^2)$ iterations. Note that unlike SGD which is ensured to converge with $m = 1$ for convex functions, SNGD requires a minimal mini-batch size for locally quasiconvex functions. Using the similarity between online learning and stochastic optimization, we propose to use the equivalent *online normalized gradient descent* (ONGD) to track the parameter vector $\theta_t = (\Lambda_t, \sigma_t^2, \nu_t, b_t)$ of $J(\theta_t)$, where $Q(x_t, \theta_t) = \log p(x_t|\mathcal{F}_{t-1}; \theta_t)$. In order to get an alternative recursive algorithm, we also adapt Algorithm 2 of [**Paper A**] and Section 2.3.2 to the time-dependent negative log-likelihood in (3.11). This is straightforward and the reader is referred to [**Paper B**] for computational details. Algorithm 2 relies on a Newton step whereas (3.11) is not convex. However, as mentioned in Section 2.3.2, it resembles a quasi-Newton approach as it approximates the Hessian with a positive definite matrix using only the gradients of (3.11). Although suboptimal, we can hope for this classic approach to work in an online learning framework if the Hessian approximation is locally good enough. We call it rMLE.b, while Algorithm 2 is now referred to as rMLE.1, for $b = 1$ in Algorithm 2. In [**Paper B**] we apply NGD, ONGD and rMLE.b on both synthetic data and data from the real use case in [**Paper A**]. These new algorithms are compared to usual benchmarks and rMLE.1. The simulation study allows us to verify the behaviour of the algorithms we propose and to evaluate the improvement in making the upper varying bound assumption for probabilistic forecasting. The real case tests the validity of this new assumption for wind power data generation, based on Anholt dataset.

3.3 Bounds as random variables

3.3.1 Mixtures of scaled GLN distributions

To account for the uncertainty inherent to the case of a missing upper bound, in [**Paper C**] we propose to alternatively model it as a latent variable. Assuming it as a sequence of independent variables also allows for sharper changes in the successive values taken by the bound. The log-likelihood function of a sample of the bounded variable $\mathbf{x} = (x_t)_{t=1,\dots,T}$ is

$$l(\mathbf{x}; \theta) = \log \int_{\mathbf{b}} p(\mathbf{x}, \mathbf{b}; \theta) d\mathbf{b}. \tag{3.19}$$

A direct maximization of (3.19) would require complicated high-dimensional integration and is likely to be computationally infeasible. Moreover, for forecasting purposes, we need to be able to forecast values for the bound before forecasting the bounded variable. This calls for the use of EM-based algorithms in order to maximize the log-likelihood [18].

The choice of a probability distribution for the upper bound affects the E-step of the EM algorithm. In order to keep it as simple as possible, and from the observation that a grid of plausible values may matter more than accurate continuous values, we propose to model the upper bound as a discrete random variable taking K values $b[1], \dots, b[K] \in (0, 1]$ with probabilities $\mathbf{w} = (w[1], \dots, w[K])$. By doing so, we also give more control to the forecaster, who is free to feed the statistical model with a priori values for the bound. The values $k_t \in \mathcal{K} = \{1, \dots, K\}$ correspond to the unobserved states associated with the different values B_t can take. Let Y_t be the generalized logit transform, $Y_t = \gamma(X_t/b[k_t]; \nu)$. The expectation of Y_t is $\mathbb{E}(Y_t) = \mu_t = \sum_{r=1}^{p} \lambda_r y_{t-r}$. Let $\mathbf{b} = (b_t)_{t=1,\dots,T}$, $\mathbf{x}_{t:(t-p)} = (x_t, \dots, x_{t-p})$, $\mathbf{b}_{t:(t-p)} = (b_t, \dots, b_{t-p}) = (b[k_t], \dots, b[k_{t-p}])$, where k_t is the unobserved state at time t. Finally, let $\mathcal{F}_{t-1}$ be the previous information set for X_t, i.e., the σ-algebra generated

by $X_1, \ldots, X_{t-1}$. The conditional pdf of X_t is

$$p(x_t \mid \mathcal{F}_{t-1}, \mathbf{b}_{t:(t-p)}; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \frac{\nu}{x_t \left(1 - (x_t/b_t)^\nu\right)} \exp\left[-\frac{1}{2}\left(\frac{\gamma(x_t/b_t; \nu) - \mu_t}{\sigma}\right)^2\right], \tag{3.20}$$

if $0 \prec \mathbf{x}_{t:(t-p)} \prec \mathbf{b}_{t:(t-p)}$, $p(x_t \mid \mathcal{F}_{t-1}, \mathbf{b}_{t:(t-p)}; \theta) = 0$ otherwise. Therefore, the marginal pdf of X_t can be seen as a mixture of scaled GLN densities whose number of positive components depends on the possible values for $\mathbf{b}_{t:(t-p)}$. The log-likelihood of a sample $(\mathbf{x}, \mathbf{b})$ is

$$\sum_{t=p+1}^{T} (\log p(x_t \mid \mathcal{F}_{t-1}, \mathbf{b}_{t:(t-p)}; \theta) + \log w[k_t]). \tag{3.21}$$

3.3.2 Monte Carlo expectation-maximization algorithms

The complete log-likelihood of a sample $(\mathbf{x}, \mathbf{b})$ is much easier to maximize than the right-hand side of Equation (3.19). It is called complete for it refers to the sample we could have if we were to observe $\mathbf{b}$. Because B_t is missing, the EM algorithm aims to iteratively maximize the expected value of the complete log-likelihood given the observations $\mathbf{x}$ and the current value of the parameter $\psi = (\theta, \mathbf{w})$. This quantity is usually called the Q function and in our model is

$$Q(\psi|\psi^{(j)}) = \mathbb{E}\left[\sum_{t=p+1}^{T} (\log p(x_t \mid \mathcal{F}_{t-1}, \mathbf{b}_{t:(t-p)}; \theta) + \log w[k_t]) \;\middle|\; \mathbf{x}; \psi^{(j)}\right]. \tag{3.22}$$

It can be shown that increasing Q at each iteration of EM comes down to increasing the log-likelihood. Then, the EM algorithm is a strict ascent algorithm for the log-likelihood as long as it is possible, at iteration j, to find ψ such that

$$Q(\psi|\psi^{(j)}) > Q(\psi^{(j)}|\psi^{(j)}). \tag{3.23}$$

Since X_t needs to be considered given the former (missing) values of B_t, it is not straightforward to compute Q in (3.22). Therefore, we approximate it through Monte Carlo integration by

$$\hat{Q}_M(\psi|\psi^{(j)}) = \frac{1}{M} \sum_{m=1}^{M} \sum_{t=p+1}^{T} (\log p(x_t \mid \mathcal{F}_{t-1}, \mathbf{b}^m_{t:(t-p)}, ; \theta) + \log w[k^m_t]), \tag{3.24}$$

where M is the number of Monte Carlo samples and $\mathbf{b}^m = (b[k^m_1], \ldots, b[k^m_T])$ is distributed according to the posterior distribution $p(\mathbf{b} \mid \mathbf{x}; \psi^{(j)})$. This leads to the usual E-step of the EM algorithm being replaced with a Monte Carlo E-step [21].

This kind of algorithm comes with its own challenges, in particular regarding the choice of an appropriate number of samples M, as it is not efficient to start with a large value of M when the maximum likelihood estimate might be far from the true value of ψ. Also, a standard stopping rule for deterministic EM algorithms is to stop and claim convergence when the relative change in the parameter values from successive iterations is small. This is not possible in the case of MCEM algorithms, unless the MCEM steps can be resolved into the true EM steps and the Monte Carlo error [86]. Last, because we work with only an approximate and not with the true function Q, the ascent property of EM is also lost. An automated MCEM algorithm was proposed in [86], in that an appropriate value for M is chosen after each iteration and the algorithm is stopped when changes in the parameter estimates are small after taking the Monte Carlo error into account. A few years later, [87] built on the work of [86] and [88], [89] by studying a data-driven automated

MCEM algorithm, and proposed an ascent-based algorithm that recovers EM's ascent property with high probability. Both methods proposed in [86] and [87] rely on the central limit theorem, and quantities involving the derivative of $\hat{Q}_M$ and its limiting distribution when M goes to infinity. We do not propose innovative methods in automating an MCEM algorithm, as we choose to rather focus on an online version of the algorithm. Therefore, the batch algorithm we propose uses a fixed size M and a predetermined number of iterations J.

In order to generate a sample $(\mathbf{b}^1, \ldots, \mathbf{b}^M)$ for the Monte Carlo integration in (3.24), we consider Markov chain algorithms. There are many ways of constructing Markov chains to eventually approximate the desired expectation [90], [91], but all of them are special cases of the general framework of [92] and [93]. We use a single-component Metropolis-Hastings (MH) algorithm with an independence sampler. Let b_t^m denote the value of b_t at the end of iteration m. For step t of iteration $m+1$, b_t is updated using MH. The candidate B_t^* is generated from a proposal distribution $q_t(B_t^*|b_t^m, \mathbf{b}_{-t}^m)$, where $\mathbf{b}_{-t}^m$ denote the value of $\mathbf{b}_{-t}$ after completing step $t-1$ of iteration $m+1$,

$$\mathbf{b}_{-t}^m = \{b_1^{m+1}, \ldots, b_{t-1}^{m+1}, b_{t+1}^m, \ldots, b_T^m\}, \tag{3.25}$$

and components $1, \ldots, t-1$ have already been updated. This is the updating scheme of the single-component MH algorithm, which is in fact the framework originally proposed by [92]. To use an independence sampler means to use a proposal distribution $q_t(B_t^*|\mathbf{b}_{-t}^m)$ instead of $q_t(B_t^*|b_t^m, \mathbf{b}_{-t}^m)$. In general, the independence sampler can work very well or very badly. For it to work well, $q_t(.)$ should be a good approximation to the target distribution. It is rather easy to fulfill this condition in our framework as the random variables B_t are independent and identically distributed, hence $q_t(B_t^*|\mathbf{b}_{-t}^m) = q_t(B_t^*)$, and we can choose $q_t(.)$ to be the marginal distribution of b_t truncated according to the observed value x_t. A candidate B_t^* is then accepted with a probability which is straightforward to calculate in our model,

$$\rho(\mathbf{b}_{-t}^m, b_t^m, B_t^*) = \min\left(1, \frac{\prod_{l=t}^{t+p} p(x_l \mid \mathcal{F}_{l-1}, B_t^*, \mathbf{b}_{-t}^m; \theta^{(j)})}{\prod_{l=t}^{t+p} p(x_l \mid \mathcal{F}_{l-1}, b_t^m, \mathbf{b}_{-t}^m; \theta^{(j)})}\right). \tag{3.26}$$

The corresponding MH algorithm is described in the Appendix of [**Paper C**].

3.3.3 Expectation-conditional maximization algorithms

Once a sample $(\mathbf{b}^1, \ldots, \mathbf{b}^M)$ is available at iteration j, which depends on the current value $\psi^{(j)}$, we need to maximize the corresponding $\hat{Q}_M$ function (3.24) at the M-step. Again, because of the shape parameter of the GLN distribution, the direct application of EM to our problem would require iterative methods at the M-step, i.e., inner loop iterations. Moreover, our maximization problem is a constrained optimization problem, as the probabilities $w[k]$ of the values taken by the bound, i.e., of the states $k = 1, \ldots, K$, need to be non-negative and sum up to 1. To benefit from the closed-form solutions which exist for $\mathbf{w}$, Λ and σ^2 conditional on the other parameters, and following [19], we replace the M-step of the MCEM algorithm with several simpler conditional maximization (CM) steps. When no Monte Carlo integration is involved, this is the ECM algorithm, which shares all the appealing convergence properties of EM, such as always increasing the log-likelihood. As argued by [19], when used for the M-step of EM, simple and stable linear converging methods are often more suitable than super-linear converging but less stable algorithms such as Newton's method. Indeed, the super-linear convergence does not directly transfer to EM-based algorithms, for the latter converge linearly regardless of the maximization method employed within the M-step. Hence,

stability is to be preferred since the maximization method is used repeatedly. In each iteration, first, three CM-steps compute the closed-form solutions of $\mathbf{w}$, Λ and σ^2 given the current value of the other parameters. The fourth CM-step is a one-dimensional Newton-Raphson step which then updates the current value of ν.

To the best of our knowledge, there are no theoretical results for MCEM outside of exponential families, even more when replacing the M-step by CM-steps. Nevertheless, let us define the set of functions $G = \{g_s, s = 1, \ldots, S\}$, with $S = 4$, $g_1(\psi) = \mathbf{w}$, $g_2(\psi) = \Lambda$, $g_3(\psi) = \sigma^2$ and $g_4(\psi) = \nu$. The condition for the ECM to share the convergence properties with EM is that G is *space filling* so that we are guaranteed the resulting maximum is an unconstrained maximum of the log-likelihood (over the whole space of the parameters). This is verified if

$$J(\psi) = \cap_{s=1}^{S} J_s(\psi) = \{0\},$$

where $J_s(\psi) = \frac{\partial}{\partial \psi} g_s(\psi)$, which can be easily checked in our case (and in many applications). Note that in the last CM-step, which applies Newton's method, we do not exactly perform a maximization step, as we use backtracking line search and not exact line search [51]. However, performing backtracking line search is simpler and faster, and enough to ensure

$$Q(\psi^{(j+1)}|\psi^{(j)}) \geq Q(\psi^{(j+\frac{S-1}{S})}|\psi^{(j)}), \tag{3.27}$$

where the right-hand side of (3.27) is the value of the Q function before updating ν in iteration j, the left-hand side of (3.27) is the value of Q after updating ν in iteration j, i.e., at the end of iteration j.

3.3.4 Online and stochastic EM algorithms

The MCECM algorithm presented in Sections 3.3.2 and 3.3.3 is computationally expensive as a large M is needed to get unbiased maximum likelihood estimates. However, it can be useful as a warm-up to a subsequent online EM algorithm. Online versions of EM have been mostly designed for distributions which belong to curved exponential families [24]–[26], unlike the GLN distribution. In the field of EM, online learning and stochastic approximation are again closely related [24]. Originally, stochastic EM algorithms were alternatives to MCEM algorithms, replacing extensive Monte Carlo simulation at the E-step with simulation of a single value [22], [23], [94]. Since, SGD has also inspired EM-based algorithms [95], and stochasticity in EM comes in various forms.

Even if closed-form solutions are available for Λ and σ^2, which are the usual closed-form solutions for a normal distribution, we cannot use the corresponding, convenient recursive equations for updating them, because of the transformation γ which involves the shape parameter ν. Besides an algorithm which can handle a large amount of data, we still need the parameters of the GLN distribution to be allowed to vary with time. Also, when making predictions for the bound, we wish to put more weight on values that are more likely regarding a rather recent past if we have information, or put equal weights otherwise, i.e., we want the distribution of the bound to be allowed to vary with time. Therefore, staying in the online learning paradigm [15], [81], we propose an online version of our batch algorithm where each EM iteration is a round in a repeated game for which we take the best position we can given our past choices, without looking backwards. At each time t, we look at a Q-based function which only depends on the new observation x_t and the current value of the parameter vector ψ_t. We aim to update ψ_t by accounting only for the information

brought by time t and forgetting about the past missing values of B_t. Let $\mathcal{K}_t = \{k \in \mathcal{K} \mid b[k] > x_t\}$. In terms of the function Q, this translates to observing at time t

$$-Q_t(\psi|\psi_t) = -\sum_{k \in \mathcal{K}_t} \hat{w}_t[k](\log p(x_t \mid \mathcal{F}_{t-1}, b[k]; \theta) + \log w[k]), \tag{3.28}$$

where the posterior probabilities are

$$\hat{w}_t[k] = \frac{p(x_t \mid \mathcal{F}_{t-1}, b[k]; \theta_t)\ w_t[k]}{\sum_{l=1}^{K} p(x_t \mid \mathcal{F}_{t-1}, b[l]; \theta_t)\ w_t[l]}. \tag{3.29}$$

The key lies in $p(x_t \mid \mathcal{F}_{t-1}, b[k]; \theta_t)$, for we do not want the probability of observing x_t to still be impacted by the missing values $\mathbf{b}_{(t-1):(t-p)}$.

For the computation of the posterior probabilities in (3.29), integration over possible values of $\mathbf{b}_{(t-1):(t-p)}$ is easy and limited to $\mathcal{K}_{(t-1):(t-p)} = \mathcal{K}_{t-1} \times \cdots \times \mathcal{K}_{t-p}$, since the conditional probability density function of x_t is equal to zero for values of $\mathbf{b}_{(t-1):(t-p)}$ outside of $\mathcal{K}_{(t-1):(t-p)}$. Let k_{t-r} be the state corresponding to the value of B_{t-r} in combinations $\mathbf{b}_{(t-1):(t-p)}$, and $\hat{w}_{t-r}[k_{t-r}]$ be the corresponding posterior probability computed at time $t-r$ according to (3.29). We approximate $p(x_t \mid \mathcal{F}_{t-1}, b[k]; \theta_t)$ in (3.29) with

$$\sum_{\mathbf{b}_{(t-1):(t-p)} \in \mathcal{K}_{(t-1):(t-p)}} p(x_t \mid \mathcal{F}_{t-1}, b[k], \mathbf{b}_{(t-1):(t-p)}; \theta_t)\ \hat{w}_{t-1}[k_{t-1}]\ \ldots\ \hat{w}_{t-p}[k_{t-p}]. \tag{3.30}$$

The approximation lies in using $P(\mathbf{b}_{(t-1):(t-p)} \mid \mathcal{F}_{t-1}; \theta_t) \approx \hat{w}_{t-1}[k_{t-1}]\ \ldots\ \hat{w}_{t-p}[k_{t-p}]$. Therefore, we consider the successive posterior probabilities, which have been computed at previous iterations, of b_{t-r} given x_{t-r} rather than given the whole filtration $\mathcal{F}_{t-r}$. Moreover, we assume $\psi_t \approx \psi_{t-1} \approx \cdots \approx \psi_{t-p}$.

Integration cannot be considered in (3.28) because of the logarithm. We propose two ways of dealing with the past missing values of B_t. Let $\hat{\mathbf{w}}_t = (\hat{w}_t[1], \ldots, \hat{w}_t[K])$. As a first option, we replace the past missing values $\mathbf{b}_{(t-1):(t-p)}$ with filtered values $\hat{\mathbf{b}}_{(t-1):(t-p)}$, such as

$$\hat{b}_{t-r} = \sum_{k=1}^{K} \hat{\mathbf{w}}_{t-r}[k]\ b[k], \quad r = 1, \ldots, p, \tag{3.31}$$

i.e., each missing value b_{t-r} is replaced with its conditional expectation on the observation x_{t-r} and the current parameter ψ_{t-r}. As a second option, we instead simulate a single value of $\mathbf{b}_{(t-1):(t-p)}$ according to the posterior probabilities $\hat{\mathbf{w}}_{(t-1):(t-p)}$. This is closer in spirit to historical stochastic EM algorithms, see, e.g., [22], [23]. Then, at each time t we have

$$-\hat{Q}_t(\psi|\psi_t) = -\sum_{k \in \mathcal{K}_t} \hat{w}_t[k](\log p(x_t \mid \mathcal{F}_{t-1}, b[k], \hat{\mathbf{b}}_{(t-1):(t-p)}; \theta) + \log w[k]), \tag{3.32}$$

and we update the parameter ψ_t through a descent step using the gradient of the function $-\hat{Q}_t$.

The online EM we propose borrow ideas from stochastic optimization by using noisy estimates of the objective's gradient. The second online EM algorithm was also inspired by historical stochastic EM algorithms in using simulation at the E-step for the past values of the bound [22], [23]. These algorithms have connections with Titterington's generic method [27]. However, in our specific mixture case, we remain closer to the EM spirit and to [24] by having still an E-step through the computation of $\hat{w}_t[1], \ldots, \hat{w}_t[K]$, and an M-step through the descent step in the direction of the negative gradient of $-\hat{Q}_t$. Moreover, we do not need to compute and inverse the complete-data

Fisher information matrix as in [27], which would be prohibitive in our case. The online algorithms we propose show nice tracking abilities in changing environments. However, they fail to converge to unbiased estimates of the parameters of the GLN distribution. We believe this mostly has to do with the approximation which is performed at the E-step to replace the missing past values of the bound. We have performed simulation studies with $K = 3$ and $p = 1$ or $p = 2$. Additional tests are required to evaluate the impact on the bias of an increase of both K and p. Alternative or complementary approaches need to be considered to reduce this bias. Lastly, the assumptions which are made at each time t when computing the posterior probabilities of the bound with (3.30) should also be further investigated.

CHAPTER 4

Data is missing again

We applied the algorithms which were designed over the course of this thesis to real power generation at Anholt offshore wind farm in [**Paper A**] and [**Paper B**]. Because we worked with the average power generation X_t, when individual, wind turbine data records were missing, we averaged the records which were available at time t. In our last [**Paper D**], we question the way we dealt with missing values and propose a more sophisticated imputation method, which takes advantage of both the structure of the wind farm and collected data in an online fashion. This chapter first introduces the issue of having to deal with missing data points when learning from data in general, and at offshore wind farms in particular. It then summarises the methodology proposed in [**Paper D**], which relies on spectral graph theory.

4.1 Dealing with missing values

4.1.1 When learning from data

Whether they rely on statistical assumptions, probability distributions, do not make any assumptions at all, operate on a batch sample or on data streams, learning methods and algorithms have in common that they need data to learn. Losing data while recording, collecting or even preparing them has been an issue since the earliest attempts to build knowledge out of their exploitation, for it is intrinsically part of the process: sensors or machines fail, people forget or do not want to answer questions, etc. This becomes an issue when methods at hand assume and require complete samples, or the most recent records of a data stream. This is the case for most statistical methods and online algorithms.

The simplest way of dealing with missing values when learning from data is to delete incomplete records and assume the dataset to be complete. This seems rather appropriate if the number of complete records left is large enough for the learning method to perform well. As for variables, if we are missing most observations of a single variable, we would rather delete this variable than delete all records for which the variable is missing. However, one should not be oblivious of the *missingness mechanism*. Three seminal, yet controversial [96], mechanisms have been introduced by [97]: missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR). Let $\mathbf{X}$ and $\mathbf{X}^m$ be the observed and missing data of a dataset, respectively. $\mathbf{X}$ is only a subset of the complete data model $(\mathbf{X}, \mathbf{X}^m)$. Observations are said to be MCAR if the probability that an observation is missing does not depend on $(\mathbf{X}, \mathbf{X}^m)$. They are said to be MAR if the probability that an observation is missing only depends on $\mathbf{X}$. Finally, they are said to be MNAR in every other case. The first two mechanisms are *ignorable* mechanisms, in the sense that they do not make it necessary to explicitly model the missing values distribution. MNAR on the other hand leads to significant biases in the data $\mathbf{X}$, for the missingness depends on values which are not observed. For instance, let us focus on the case of maximum likelihood inference,

when one wishes to estimate the parameter θ of a probability density function p_θ based on the realizations of a random variable $(\mathbf{X}, \mathbf{X}^m)$. In order to do so, one needs to maximize the likelihood of the observations $p_\theta(\mathbf{X})$. To acknowledge the presence of missing values, let the missingness mechanism be an additional variable $\mathbf{M}$ with its own probability density function p_ϕ, $\mathbf{M}_i = 0$ if $\mathbf{X}_i \in \mathbf{X}$, $\mathbf{M}_i = 1$ if $\mathbf{X}_i \in \mathbf{X}^m$. We are to maximize the likelihood of the observations

$$p_\psi(\mathbf{X}, \mathbf{M}) = \int p_\psi(\mathbf{X}, \mathbf{X}^m, \mathbf{M}) d\mathbf{X}^m, \tag{4.1a}$$

$$= \int p_\phi(\mathbf{M}|\mathbf{X}, \mathbf{X}^m) p_\theta(\mathbf{X}, \mathbf{X}^m) d\mathbf{X}^m, \tag{4.1b}$$

where $\psi = (\theta, \phi)$. If $\mathbf{M}$ does not depend on $\mathbf{X}^m$ we get $p_\phi(\mathbf{M}|\mathbf{X}, \mathbf{X}^m) = p_\phi(\mathbf{M}|\mathbf{X})$ and

$$p_\psi(\mathbf{X}, \mathbf{M}) = p_\phi(\mathbf{M}|\mathbf{X}) \int p_\theta(\mathbf{X}, \mathbf{X}^m) d\mathbf{X}^m, \tag{4.2a}$$

$$= p_\phi(\mathbf{M}|\mathbf{X}) p_\theta(\mathbf{X}). \tag{4.2b}$$

We see that maximizing the likelihood to get an estimate of θ while acknowledging $\mathbf{M}$ comes down to maximizing $p_\theta(\mathbf{X})$, since $p_\phi(\mathbf{M}|\mathbf{X})$ does not depend on θ. However, this is appropriate only when $p_\phi(\mathbf{M}|\mathbf{X}, \mathbf{X}^m) = p_\phi(\mathbf{M}|\mathbf{X})$, i.e., only for MCAR and MAR mechanisms. The EM algorithm, see Section 3.3, can be used for maximizing the log-likelihood

$$\log p_\theta(\mathbf{X}) = \log \int p_\theta(\mathbf{X}, \mathbf{X}^m) d\mathbf{X}^m, \tag{4.3}$$

when this quantity cannot be computed explicitly.

A popular way of dealing with missing data is *imputation*, which consists of filling in the missing values with reasonable estimates to get a completed dataset. Imputation can be single, or multiple [40]. If one aims to replace missing records as accurately as possible, then they perform single imputation and get one completed dataset. Multiple imputation on the other hand consists of generating M possible values for each missing entry, in order to get M completed datasets, so that the variability across the datasets reflects the variance when predicting each missing entry. Therefore, the aim of multiple imputation is rather to take into account the uncertainty which comes from missing values. Note that the objective of estimating parameters of a distribution as well as possible despite missing values, while properly accounting for their variance, is shared by the EM approach, which also performs imputation in the E-step, even if imputation here is more a by-product than a goal per se. The design of methods which do not require the completeness assumption and are more robust to missing values constitutes another research area. Often, some filling-in is performed or implied as part of the process and these methods are not imputation-free [98].

Note that we focus on data which are missing during the learning process, as our probabilistic model does not require data other than the wind power generation itself, and only uses past recorded values. Many forecasting models use exogenous variables which were available during training, but might be missing when the models actually need them for forecasting. For such models it is worth designing algorithms more robust to missing data at test time [99].

4.1.2 At offshore wind farms

So far we have considered X_t, the average wind power generation at a wind farm at time t. To be exactly computed, X_t requires a complete record at time t, i.e., $(X_t^1, \ldots, X_t^N)$, where X_t^i is

the wind power generation of wind turbine i at time t, N is the number of wind turbines in the wind farm. Because the probability of getting complete records decreases with the number of wind turbines in a wind farm, deleting incomplete records is not an option. As a matter of fact, complete records are only half of the records in Westermost Rough dataset and 66% of the records in Anholt dataset, see Section 1.3. When dealing with time series, it is usually advised against deleting records, for it breaks the dynamics, especially when there are too many to delete: a short sequence might be relatively easy to fill in, but not a long one. Again, recall that some individual power generation time series are missing over periods that can be longer than a month in both Westermost Rough and Anholt datasets (and not only at the beginning of the dataset). Also, it does not sound sensible to give up on records which miss only one or a few entries. Lastly, missing entries may be MNAR [100].

The time series $(X_t^1, \dots, X_t^N)_{t\in\mathcal{T}}$ can be seen as a multivariate time series of N variables X_t^i, i.e., the individual power generations at time t. In the case of a wind farm, it is likely that these variables will be correlated, and a variable X_t^i can be a good estimator when it comes to providing plausible estimates for another variable whose observation is missing at time t. This is particularly interesting for extended sequences of missing values. The average power generation X_t at time t is

$$X_t = \frac{1}{N}\sum_{i=1}^{N} X_t^i. \tag{4.4}$$

When some entries are missing at time t, assume that, instead, we work with

$$\hat{X}_t = \frac{1}{n_t}\sum_{j=1}^{n_t} X_t^{(j)}, \tag{4.5}$$

where $X_t^{(j)}$ is the power generation at time t for the (j)-th available wind turbine record, n_t being the number of available entries, $n_t \leq N$. This is the choice we made when working with data from Anholt in [**Paper A**] and [**Paper B**]. Note that we mentioned averaging over 110 wind turbines when there are actually 111 wind turbines in Anholt wind farm. Indeed, we removed one wind turbine from the dataset, for a lot of its records were missing over the period which was the best fit for the other wind turbines. We show that this intuitive averaging choice is equivalent to n_t-NN imputation.

Imputation with k-NN consists of filling in a missing value using the values from its k-nearest neighbors. Unweighted k-NN give the same weight to each neighbor, when weighted k-NN assign a higher weight to a closer neighbor. Let us replace each missing value $X_t^{(l)}$ with its unweighted n_t-NN estimate $\hat{X}_t^{(l)} = \frac{1}{n_t}\sum_{j=1}^{n_t} X_t^{(j)}$, where $X_t^{(j)}$ are the n_t values available at time t. We have

$$\hat{X}_t = \frac{1}{N}\left(X_t^{(1)} + \cdots + X_t^{(n_t)} + \hat{X}_t^{(n_t+1)} + \cdots + \hat{X}_t^{(N)}\right), \tag{4.6a}$$

$$= \frac{1}{N}\sum_{j=1}^{n_t} X_t^{(j)} + \frac{1}{N}\sum_{l=n_t+1}^{N} \hat{X}_t^{(l)}, \tag{4.6b}$$

$$= \frac{1}{N}\sum_{j=1}^{n_t} X_t^{(j)} + \frac{1}{N}\sum_{l=n_t+1}^{N} \frac{1}{n_t}\sum_{j=1}^{n_t} X_t^{(j)}, \tag{4.6c}$$

$$= \frac{1}{n_t}\sum_{j=1}^{n_t} X_t^{(j)}. \tag{4.6d}$$

To work with $\hat{X}_t$ in (4.5) is equivalent to filling in the missing values $(X_t^{(l)})_{l=n_t+1,\dots,N}$ using unweighted n_t-NN estimates, i.e., where each neighbor is assigned the same weight. However, the

power generation from a wind turbine is likely to be more similar to the power generation from the real neighbors of this wind turbine, i.e., the wind turbines which are located nearby in the wind farm. Therefore, accounting for the geographical, structural neighborhoods in the wind farm could already improve this simple imputation.

Imputation with k-NN is appealing for the associated assumptions are very weak: we do not assume any model generating the data, observed or missing, and only assume similar groups of observations; moreover, the method applies for all missing data mechanisms. Staying in the k-NN framework, we can improve our estimates through the number of neighbors k, the weights we assign to neighbors, or both. Indeed, to fill in a missing individual power generation record, we can replace it with the averaged power generation records of $k \leq n_t$ neighboring wind turbines, and this average can be either weighted or unweighted. Theoretical results about k-NN mostly concern the asymptotic mode [101]–[103], when n_t tends to infinity, which cannot be assumed here as we are limited by the number N of wind turbines in the wind farm. It is rather critical to choose k in a finite regime, and it is usually advised to perform cross-validation. This would be cumbersome as we would need to run cross-validation for each combination of available wind turbine records and for each missing data point. It would also require quite an amount of complete records. Hence, we propose to keep $k = n_t$ at each time t and rather improve the weights of the n_t-NN imputation. A weighted n_t-NN imputation means that we wish to assign greater weights to neighbors which are closer to the missing data point, according to some distance. Learning the distance metric for k-NN has been extensively studied and it was found that metric learning may significantly affect the performance of the method in many applications. Because we perform imputation at each time step t considering power generation from similar sensors at the same time step t, the Euclidean distance seems a fair enough metric for our application. Instead, we focus on a common shortcoming in non-parametric methods, which is to only consider the distances between the point of interest and its neighbors, and ignore the geometrical relation between these neighbors.

4.2 Imputation at a wind farm using online graphs

4.2.1 Laplacian eigenmaps of a wind farm

In order to account for the geometry of the wind farm, we consider a wind farm as an undirected graph $G = (V, E)$, where the nodes $V = v_1, \ldots, v_N$ are the wind turbines. As for the edges E, we connect nodes depending on the layout of the wind farm only, i.e., depending on the position of the wind turbines with respect to each other. Let $\mathbf{A} = (a_{ij})_{i,j=1,\ldots,N}$ be the *adjacency* matrix of graph G. All edges are first assumed to have the same strength: $a_{ij} = 1$ if nodes v_i and v_j are connected by an edge, $a_{ij} = 0$ otherwise. Note that the diagonal of $\mathbf{A}$ is equal to zero, i.e., $a_{ii} = 0$, as we do not consider self-connections. Through the adjacency matrix $\mathbf{A}$, each wind turbine is represented by the vector of size N of its connections to the other wind turbines in the wind farm. Out of this representation, we wish to learn a low-dimensional embedding for each wind turbine that preserves the structure of the wind farm. We are interested in Laplacian eigenmaps, which optimally preserve local neighborhood information and produce coordinate maps that are smooth functions over the original graph [38]. By trying to preserve local information in the embedding, the algorithm implicitly emphasizes the natural clusters in the data. Close connections to spectral clustering then become very clear [39].

Let $\mathbf{D}$ be the diagonal matrix associated with graph G, whose entries are the degree of each node, i.e., $d_{ii} = \sum_j a_{ji}$. The matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is called the graph Laplacian and we get eigenmaps by

computing eigenvalues and eigenvectors for the generalized eigenvector problem

$$\mathbf{L}\mathbf{f} = \lambda \mathbf{D}\mathbf{f}. \tag{4.7}$$

Using r eigenvectors $\mathbf{f}$ we now have new representations, or embeddings, for each wind turbine, which account for the geometry of the wind farm. Let $\mathbf{z}^i$ be the embedding of wind turbine v_i, $\mathbf{z}^i \in \mathbb{R}^r$. Distances $\|\mathbf{z}^i - \mathbf{z}^j\|$ for every pair of wind turbines (v_i, v_j) can be computed once and for all and used for assigning weights to a neighbor v_j, when power generation has not been recorded for v_i (or conversely).

4.2.2 Accounting for observed data

The former representations embed the structure of the wind farm only, and can be used without having any other data but the map of the wind farm. So far we have assumed the same strength to all edges. To take into account both the structure of the wind farm and information from power generation data when performing imputation, we propose to weigh the edges depending on a measure of similarity between the power generations of two connected wind turbines. If v_i and v_j are connected by an edge, a simple and intuitive choice for the similarity between X_t^i and X_t^j is $s_t(i,j) = 1 - \|x_t^i - x_t^j\|$, $s_t(i,j) \in [0,1]$, where x_t^i and x_t^j are the recorded values for X_t^i and X_t^j. Note that if it happens at time t that two wind turbines which are not connected in graph G have very similar power generations, they remain unconnected. By doing so, we enforce an a priori on the relationships between the wind turbines upon the structure of the wind farm, but we also keep a sparse adjacency matrix $\mathbf{A}$. Indeed, $\mathbf{A}$ now depends on t and we have $\mathbf{A}_t = (a_{t,ij})_{i,j=1,\dots,N}$, where $a_{t,ij} = s_t(i,j)$ if nodes v_i and v_j are connected by an edge, $a_{t,ij} = 0$ otherwise. This implies to solve the generalized eigenproblem (4.7) at each time step t, which can be done rather easily when dealing with sparse matrices $\mathbf{A}_t$.

It may happen that two wind turbines which are a priori connected get disconnected because $s_t(i,j) = 0$ for some time t. In such a case, we can obtain a graph which is not *connected* anymore, i.e., we cannot travel through the whole graph from any point in the graph. To compute eigenmaps, the generalized eigenproblem (4.7) must be solved for a connected graph, which ensures $\text{rank}(\mathbf{D}) = N$ and there are N eigenvalues, or $\lambda(\mathbf{L}, \mathbf{D})$ may be finite, empty or infinite [104]. Therefore, when a graph has several components, the algorithm for computing eigenmaps consists of solving the generalized eigenproblem (4.7) for each connected component. Since we are online and in high dimension, we need to be able to automatically detect when the graph is not connected anymore, and what are the components. Let $\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$ be the *normalized* graph Laplacian, which is the graph Laplacian we use when computing eigenmaps. We recall a very useful property of this graph Laplacian [39], that makes it easy to derive the connected components of G at time t if the multiplicity of the eigenvalue 0 of $\mathbf{L}_{rw,t}$ becomes higher than 1:

Proposition 4.2.1 (Number of connected components and spectra of $\mathbf{L}_{rw}$) *Let G be an undirected graph with non-negative weights. Then the multiplicity d of the eigenvalue 0 of $\mathbf{L}_{rw}$ equals the number of connected components $A_1, \dots, A_d$ in the graph and the eigenspace of 0 is spanned by the indicator vectors $\mathbb{1}_{A_i}$ of those components.*

The similarities $s_t(i,j)$ are known at time t among the wind turbines for which data points are available, but they are not for the edges involving wind turbines for which the record is missing,

and we need to replace them with estimates. Although the correlations between wind turbine power generations are unlikely to be stationary, we infer they will be more stable than the individual time series themselves, and easier to fill in. To avoid specifying a model for the similarities between all individual time series, we again place ourselves in the online learning framework. Our goal is to guess the sequence of similarities as precisely as possible, while they are chosen by an adversary rather than generated stochastically [15], [16]. This turns to the following repeated game: in each round $t = 1, \dots, T$, for each similarity between two connected wind turbines $s_t(i, j)$, an adversary chooses a real number in $[0, 1]$ and keeps it secret; we try to guess the real number, choosing $\hat{s}_t(i, j)$; the adversary number is revealed and we pay the squared difference $(s_t(i, j) - \hat{s}_t(i, j))^2$. One should note that the last step of the repeated game when the adversary number is revealed does not happen if some of the data are missing. Therefore, to allow missing data, the game is slightly modified and we pay $(s_t(i, j) - \hat{s}_t(i, j))^2$ only if $s_t(i, j)$ is revealed [98]. Using the notation $\mathbb{1}_{\{s_t(i,j)\}}$ as the indicator of the event $\{s_t(i, j) \text{ is revealed}\}$, we pay now $(s_t(i, j) - \hat{s}_t(i, j))^2 \mathbb{1}_{\{s_t(i,j)\}}$. In such a missing data, convex framework, an online strategy with good theoretical guarantees is the lazy version of OGD [17]. We also consider the best constant strategy, i.e, the strategy that minimizes $\sum_{t=1}^{T} (s_t(i, j) - \hat{s}_t(i, j))^2 \mathbb{1}_{\{s_t(i,j)\}}$, which is just constantly choosing $\hat{s}_t(i, j)$ to be the average similarity $\sum_{t=1}^{T} s_t(i, j) \mathbb{1}_{\{s_t(i,j)\}} \big/ \sum_{t=1}^{T} \mathbb{1}_{\{s_t(i,j)\}}$ at each round.

Because we have a new graph at each time t, we also get new embeddings $\mathbf{z}_t^i$ for each wind turbine v_i, which account for the geometry of the wind farm and the online similarities between power generations. Distances $\|\mathbf{z}_t^i - \mathbf{z}_t^j\|$ for every pair of wind turbines (v_i, v_j) and corresponding time-varying n_t-NN weights need to be computed at each time t as well.

4.2.3 From distances to neighbors' weights

To convert the distances between two wind turbine embeddings into weights for the n_t-NN imputation, we propose to use Nadaraya-Watson estimators [105], [106]. Let K be a given non-negative measurable function on $\mathbb{R}$, the *kernel*, h be a positive number, the *bandwith*, depending on n_t only, and $\|\mathbf{z}^{(j)} - \mathbf{z}^{(l)}\|$ be the Euclidean distance between the representations of two wind turbines (j) and (l). In case (l)'s record is missing and (j)'s is available at time t, the weight we assign to $X_t^{(j)}$ when computing a weighted estimate $\hat{X}_t^{(l)} = \sum_{j=1}^{n_t} w^{(jl)} X_t^{(j)}$ of $X_t^{(l)}$ is

$$w^{(jl)} = \frac{K\left(\frac{\|\mathbf{z}^{(j)} - \mathbf{z}^{(l)}\|}{h}\right)}{\sum_{i=1}^{n_t} K\left(\frac{\|\mathbf{z}^{(i)} - \mathbf{z}^{(l)}\|}{h}\right)}. \tag{4.8}$$

Let us sort the neighbors of a wind turbine (l) by increasing distance,

$$\|\mathbf{z}^{(1)} - \mathbf{z}^{(l)}\| \leq \|\mathbf{z}^{(2)} - \mathbf{z}^{(l)}\| \leq \cdots \leq \|\mathbf{z}^{(n_t)} - \mathbf{z}^{(l)}\|.$$

We choose an adaptive bandwith $h_t = \|\mathbf{z}^{(n_t)} - \mathbf{z}^{(l)}\|$, so that the weights adjust depending on n_t, i.e., depending on the availability of other records at time t. These Nadaraya-Watson estimators could be replaced with an algorithm able to compute optimal weights efficiently and adaptively for each data point we wish to estimate, out of the distances between the wind turbine embeddings [107]. Of course the price to pay would be more computational effort.

In [**Paper D**] we apply weighted n_t-NN imputation to Westermost Rough offshore wind farm and compare it to averaging over available records, i.e., to naive unweighted n_t-NN imputation. The improvement on wind power generation data is already significant overall; it can be more than 20% for some wind turbines, depending on their position inside the wind farm. The method we propose is useful already in the early stage of a wind farm's life, as it can start with the edges of G being unweighted edges, and move to using online weighted graphs G_t as soon as there is one observed similarity for each pair of connected wind turbines. It can be applied to any task, model or distribution, while pre-processing batch data, or run on data streams to better deal with recent missing entries for forecasting purposes.

CHAPTER 5

Conclusion

In this thesis, we focused on probability distributions and statistical models more suited to the special characteristics of wind power generation than classical Gaussian or beta assumptions in stationary frameworks. We addressed the non-linearity and the bounded support of wind power generation by working with the GLN distribution, and extended the methods available for the estimation of this, in our opinion, underused distribution. In order to accommodate non-stationarity, we always intended to go towards online learning approaches. This led us to questioning the assumption of stationarity even for the support of the bounded distribution. The range of applications which could benefit from such a novel framework is broad and motivated us to generalize the idea. Finally, because we worked on real datasets from two offshore wind farms, we challenged the way we were dealing with missing data points, and suggested an augmented imputation method compatible with our statistical framework and with any learning algorithm. We conclude here with an overview of our contributions to statistical inference involving GLN distributions in particular, and to applications involving continuous bounded response variables in general. Lastly, we give a few directions for future research, inspired by wind power probabilistic forecasting, but with a larger impact in terms of statistical and forecasting applications.

5.1 Overview of contributions

First, we developed a rigorous statistical framework to estimate the full parameter vector of a GLN distribution through maximum likelihood inference. We proposed a batch algorithm which takes advantage of the usual closed-form solutions available for the location and the scale parameters of the distribution. The algorithm relies on the conditional maximization of each parameter given the other parameters and, by doing so, only requires a one-dimensional Newton-Raphson step for retrieving the estimator of the shape parameter. The GLN distribution can be easily applied to stochastic processes of dependent variables by replacing the constant location parameter μ with an AR model. Indeed, the coefficients of this AR model also have closed-form solutions, and any sort of linear regression could be considered for including exogenous variables in the statistical model. Because we chose to perform maximum likelihood estimation, it was straightforward to derive the corresponding recursive maximum likelihood estimation. Hence, we were able to propose a recursive algorithm which can track a time-varying parameter vector and adapt a GLN distribution to changing environments. Both algorithms were applied to real data from Anholt offshore wind farm for 10-minute-ahead forecasting, and compared to usual benchmarks such as the probabilistic persistence and classical Gaussian benchmarks. The recursive algorithm in particular outperformed the other approaches, improving the CRPS of probabilistic persistence by more than 20% for probabilistic forecasting. Marginal calibration was also much improved, at the bounds, by replacing the Gaussian distribution with the GLN distribution.

Next, we made the observation that when dealing with bounded variables, the bounds of the probability distribution were always assumed to be fixed in the literature. We came to doubt this

assumption by thinking of wind power generation, but figured that a broad range of statistical applications may be concerned by varying bounds which are not, or cannot be observed. We first focused on the statistical framework for wind power forecasting. Indeed, in the context of stochastic processes, an upper varying bound can be seen as an additional parameter of the GLN distribution. Considering the bound as a parameter asked for the framework to be non-stationary, or the bound would never vary. This involved some challenges, in particular because the negative log-likelihood we wished to minimize when proceeding to maximum likelihood estimation was not convex in this new bound parameter. The batch algorithm developed at the former stage, for the original GLN distribution, used a Newton-Raphson step, which was meant for convex optimization problems. Therefore, it could not be used, even for sequential re-training. As for the former recursive algorithm, it relies on an approximation of the Hessian which is guaranteed to be positive definite, whether the optimization problem is convex or not. Nevertheless, this can only be sub-optimal at the best. From the observation that, if not convex, the optimization problem involving the bound parameter was with high probability locally quasiconvex, we used recent theoretical results about SNGD for strictly locally quasiconvex functions to derive an equivalent ONGD for our problem. Because, in this specific framework, the observations needed to be allowed to fall outside of the support of the distribution, we proposed to use an extended log-likelihood, i.e, extended outside of its support through the addition of a sigmoid, convex, function. We also showed that the CRPS dealt nicely with an observation falling outside of the support of a predictive distribution. ONGD was applied to real data from Anholt offshore wind farm for 10-minute-ahead probabilistic forecasting and compared to the former recursive algorithm, i.e., to the assumption of an upper bound fixed to 1. It improved the CRPS of the recursive algorithm by almost 18%, and the CRPS of probabilistic persistence by more than 34%. To achieve this performance, the size of the mini-batch, i.e., the number of observations which are used for updating the parameter vector, required to be set to 1. This suggests that the model calls for fast changes in the value of the upper bound. However, the predictive densities are overconfident on average, and if marginal calibration is satisfying, no method has yet achieved probabilistic calibration on Ahnolt dataset.

When considering the bound as a parameter of the GLN distribution, we could learn and forecast only one value for the bound at a time. Hence, we did not account for the uncertainty of this value. For probabilistic forecasting purposes, when we are sure about the upper bound, for instance because observations have been very close to 1 for quite some time, this should be reflected in our probabilistic forecasts. On the other hand, when we do not have much knowledge about the current value of the upper bound, because the observations have drifted away from it, we should be allowed to include this uncertainty in our probabilistic forecasting. Therefore, a general statistical framework that naturally arose was to consider the bound as a latent variable. Moreover, this framework could be applied to bounded variables other than stochastic processes, contrary to the former approach. To directly maximize the log-likelihood of the observations becomes more challenging when latent variables are involved, and direct maximization is usually replaced with EM approaches. We proposed to model the bounds as discrete independent variables. Hence, EM procedures were eased, and the forecaster was free to feed the model with sensible values for the bound. We started by designing batch algorithms for the case where the bounded variables were independent and identically distributed, and the case where they were a stochastic process with AR dependency. For independent response variables, we proposed an ECM algorithm which, similarly to the algorithm for estimating the parameter vector of a regular GLN distribution, relies on iterative conditional maximization of the parameters given the others. For stochastic processes,

we proposed an MCECM algorithm, which needs simulation at the expectation step to handle the dependency structure. We proposed to perform this simulation by using an MCMC algorithm suited to our framework. Both batch algorithms performed well on various simulation studies. However, designing online algorithms was challenging, as we expected. Online EM algorithms were not only necessary to track the parameter of the GLN distribution in the case of stochastic processes, but also to allow the rather simple distribution of the bound to vary. Currently, most online - or stochastic - versions of the EM algorithm are dedicated to the curved exponential family and do not transfer to the GLN distribution, or distributions involving generalized link functions in general. Staying in the online learning paradigm, we proposed novel online EM-based algorithms relying on OGD. For both independent response variables and stochastic processes, they showed nice abilities in tracking changes in the distribution of the bound. The online EM algorithm for independent variables also proved to converge to the maximum likelihood estimates of the parameters, whereas the online EM for stochastic processes did converge, but to biased estimates. As of now, further work is required to reduce and remove this bias, but the results obtained so far are promising.

Finally, because we ran most of the algorithms designed during this thesis on real data from offshore wind farms, we questioned the imputation method we implicitly used when dealing with missing values in historical records. Indeed, working with the average wind power generation at a wind farm, we used the average of the available entries when the wind power generation had not been recorded for some of the wind turbines. By doing so, we performed k-NN imputation. A common shortcoming of non-parametric methods such as the k-NN algorithm is to consider the distances between a point of interest and its neighbors only, and to overlook the distances between the neighbors themselves. To address this shortcoming and improve the k-NN imputation, we proposed to explicitly acknowledge the structure of the wind farm by considering it as a graph. Then, we combined spectral graph theory and lazy gradient descent to propose an online imputation method using information from both the wind farm layout and the data already collected. The proposed approach was applied to wind power generation at Westermost Rough offshore wind farm, and showed significant improvement over approaches that did not account for the wind farm layout information.

5.2 Perspectives for future research

Technical directions for further work which apply to all the online algorithms developed in this thesis concern the issue of choosing a step size, in particular for tracking purposes, as focus has largely been placed on convergence in the literature. This translates to the choice of a forgetting factor when using algorithms of the recursive kind. Further work should not only address adaptive step sizes or forgetting factors over time, but also adaptive to the parameter at hand, as the parameters of a distribution are unlikely to ask for the same varying pace when adapting to changing environments. Both directions have been considered in the control and online learning communities. In the control literature, works have addressed *selective* and *directional* forgetting in adaptive procedures [108]–[110]. In the online learning paradigm, the notions of *adaptive* and *dynamic* regret have been introduced to extend the original *static* regret [17], [111]–[113]. However, these works mostly concern either online convex optimization or recursive procedures involving quasi-Newton approximations. Therefore, they can be considered for the initial GLN distribution, but not when including the upper bound in the parameter vector.

Following up on our remark about not having been able to achieve probabilistic calibration for 10-minute-ahead forecasting at Anholt offshore wind farm, a future research direction could be

to address the uncertainty when estimating the parameters of the distribution, whether a bound parameter is included in the parameter vector or not. Throughout this thesis, the estimation of the parametric GLN distribution was performed through maximum likelihood inference in what is known as the *frequentist* paradigm, with the objective function being the log-likelihood of the data. From a *Bayesian* perspective, this is equivalent to performing *maximum a posteriori* estimation with uniform prior distributions on the parameters [114]. A research direction would be to move the estimation to the Bayesian framework, while making different assumptions on the prior distributions of the parameters: different from the uniform priors, but also different depending on the parameter at hand, in particular when including a bound parameter in the parameter vector. Of course, this would need to be done in the online paradigm if the bound parameter is included, or at least in a recursive way otherwise.

In the context of stochastic processes with serial dependency, we can suggest another approach to specifically address the uncertainty of not observing the bound, which combines ideas from the bound as a parameter and the mixtures of GLN distributions with different bounds. We propose to consider different forecasters, with their own GLN distribution, where the bound is a parameter, but a fixed parameter, the original parameters of the GLN distribution being estimated. This is far from trivial as it implies the different forecasters would learn, adapt and issue their forecasts on sub-samples of the data only, where their bound parameter is compatible with the observations. Moreover, this research direction requires to be able to combine predictive densities, in an online fashion. Related research includes forecast combinations [115], [116], expert aggregation [15] or Bayesian and dynamic model averaging [117], [118].

Finally, we focused in this thesis on the aggregate wind power generation at a wind farm. Because we have access to the power generation recorded for each wind turbine in the wind farm, future research could be dedicated to benefiting from the information which is available at the wind turbine level when forecasting at the wind farm level. Hence, the statistical framework would move from considering univariate time series to considering multivariate time series, the time series of power generation at the turbine level. For wind power forecasting, approaches based on vector auto-regressive models were considered to deal with multivariate time series [119], [120]. Such works use the logit-normal distribution, i.e., they assume a shape parameter $\nu = 1$. They could be revisited in order for the model estimation to also include the shape parameter, and even the upper bound parameter. Note that, in such works, the variables of the multivariate time series were the wind power generation from different wind farms, stacked into a multivariate time series in order to benefit from the possible cross-correlations. Working at a turbine level may be more challenging because power generation is likely to be more noisy and highly non-stationary. Other multivariate approaches could be graph-based, as proposed in Chapter 4, in order to account for the layout of the wind farm, or by learning a graph directly out of the cross-correlations of the multivariate time series. Ultimately, both individual and aggregated levels could be combined to improve probabilistic forecasting for offshore wind energy, for instance using *hierarchical* probabilistic forecasting [121]. Indeed, a recent work confirmed that novel bottom-up probabilistic approaches can improve wind power forecasting at the wind farm level by leveraging data from individual wind turbines [7]. We could even imagine introducing intermediate levels of aggregation between the individual time series and the aggregated one, to improve forecasting over the whole wind farm. Future research could combine such ideas with the frameworks developed in this thesis.

APPENDIX A

Additional figures

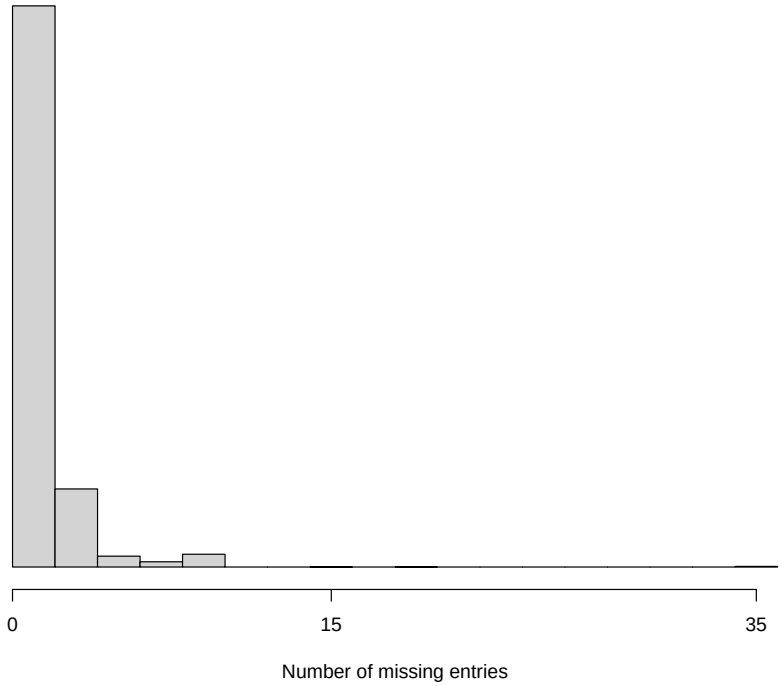


Figure A.1: Distribution of missing entries in incomplete records (Westermost Rough).

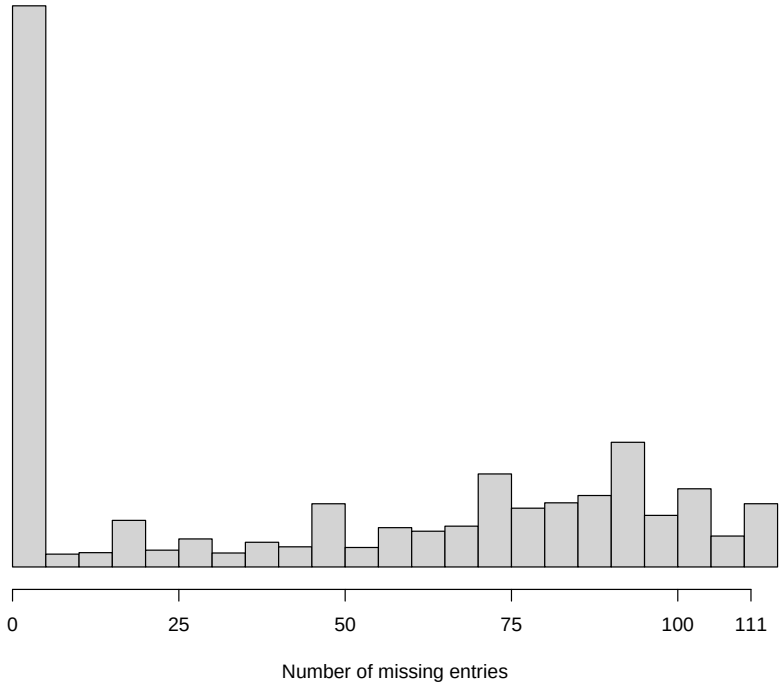


Figure A.2: Distribution of missing entries in incomplete records (Anholt).

Bibliography

[1] A. P. Dawid, 'Present position and potential developments: Some personal views: Statistical theory: The prequential approach,' *Journal of the Royal Statistical Society. Series A (General)*, vol. 147, no. 2, pp. 278–292, 1984.

[2] T. Gneiting, 'Editorial: Probabilistic forecasting,' *Journal of the Royal Statistical Society Series A: Statistics in Society*, vol. 171, no. 2, pp. 319–321, 2008.

[3] T. Gneiting and M. Katzfuss, 'Probabilistic forecasting,' *Annual Review of Statistics and Its Application*, vol. 1, no. 1, pp. 125–151, 2014.

[4] T. Hong, P. Pinson, Y. Wang, R. Weron, D. Yang and H. Zareipour, 'Energy forecasting: A review and outlook,' *IEEE Open Access Journal of Power and Energy*, vol. 7, pp. 376–388, 2020.

[5] P. Eriksen, A. Orths and V. Akhmatov, 'Integrating dispersed generation into the Danish power system - present situation and future prospects,' in *2006 IEEE Power Engineering Society General Meeting*, 2006.

[6] V. Akhmatov, 'Influence of wind direction on intense power fluctuations in large offshore windfarms in the North Sea,' *Wind Engineering*, vol. 31, no. 1, pp. 59–64, 2007.

[7] C. Gilbert, J. Browell and D. McMillan, 'Leveraging turbine-level data for improved probabilistic wind power forecasting,' *IEEE Transactions on Sustainable Energy*, vol. 11, no. 3, pp. 1152–1160, 2020.

[8] P. Pinson, 'Wind energy: Forecasting challenges for its operational management,' *Statistical Science*, vol. 28, no. 4, pp. 564–585, 2013.

[9] N. L. Johnson, 'Systems of frequency curves generated by methods of translation,' *Biometrika*, vol. 36, no. 1/2, pp. 149–176, 1949.

[10] R. Mead, 'A generalised logit-normal distribution,' *Biometrics*, vol. 21, no. 3, pp. 721–732, 1965.

[11] P. Pinson, 'Very-short-term probabilistic forecasting of wind power with generalized logit-normal distributions,' *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 61, no. 4, pp. 555–576, 2012.

[12] P. Young, *Recursive Estimation and Time-Series Analysis*. Springer Berlin, Heidelberg, 1984.

[13] H. Madsen, *Time Series Analysis*. Chapman and Hall/CRC, 2007.

[14] P. Pinson and H. Madsen, 'Adaptive modelling and forecasting of offshore wind power fluctuations with Markov-switching autoregressive models,' *Journal of Forecasting*, vol. 31, no. 4, pp. 281–313, 2012.

[15] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[16] F. Orabona, *A modern introduction to online learning*, 2022. arXiv: 1912.13213.

[17] E. Hazan, *Introduction to online convex optimization*, 2021. arXiv: 1909.05207.

[18] A. P. Dempster, N. M. Laird and D. B. Rubin, 'Maximum likelihood from incomplete data via the EM algorithm,' *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[19] X.-L. Meng and D. B. Rubin, 'Maximum likelihood estimation via the ECM algorithm: A general framework,' *Biometrika*, vol. 80, no. 2, pp. 267–278, 1993.

[20] K. Lange, 'A gradient algorithm locally equivalent to the EM algorithm,' *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 57, no. 2, pp. 425–437, 1995.

[21] G. C. G. Wei and M. A. Tanner, 'A Monte Carlo implementation of the EM algorithm and the Poor Man's data augmentation algorithms,' *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 699–704, 1990.

[22] G. Celeux, D. Chauveau and J. Diebolt, 'Stochastic versions of the EM algorithm: An experimental study in the mixture case,' *Journal of Statistical Computation and Simulation*, vol. 55, no. 4, pp. 287–314, 1996.

[23] I. C. Marschner, 'On stochastic versions of the EM algorithm,' *Biometrika*, vol. 88, no. 1, pp. 281–286, 2001.

[24] O. Cappé and E. Moulines, 'On-line expectation-maximization algorithm for latent data models,' *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 71, no. 3, pp. 593–613, 2009.

[25] O. Cappé, 'Online EM algorithm for hidden Markov models,' *Journal of Computational and Graphical Statistics*, vol. 20, no. 3, pp. 728–749, 2011.

[26] S. L. Corff and G. Fort, 'Online expectation maximization based algorithms for inference in hidden Markov models,' *Electronic Journal of Statistics*, vol. 7, no. none, pp. 763–792, 2013.

[27] D. M. Titterington, 'Recursive parameter estimation using incomplete data,' *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 46, no. 2, pp. 257–267, 1984.

[28] O. Troyanskaya, M. Cantor, G. Sherlock *et al.*, 'Missing value estimation methods for DNA microarrays,' *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[29] E. Fix and J. L. Hodges Jr., 'Discriminatory analysis - nonparametric discrimination: Consistency properties,' USAF School of Aviation Medicine, Tech. Rep., Feb. 1951.

[30] T. Cover and P. Hart, 'Nearest neighbor pattern classification,' *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[31] G. Biau and L. Devroye, *Lectures on the Nearest Neighbor Method*. Springer, 2015.

[32] M. Zamo, O. Mestre, P. Arbogast and O. Pannekoucke, 'A benchmark of statistical regression methods for short-term forecasting of photovoltaic electricity production, part i: Deterministic forecast of hourly production,' *Solar Energy*, vol. 105, pp. 792–803, 2014.

[33] M. Zamo, O. Mestre, P. Arbogast and O. Pannekoucke, 'A benchmark of statistical regression methods for short-term forecasting of photovoltaic electricity production. part ii: Probabilistic forecast of daily production,' *Solar Energy*, vol. 105, pp. 804–816, 2014.

[34] S. T. Jensen, S. Johansen and S. L. Lauritzen, 'Globally convergent algorithms for maximizing likelihood function,' *Biometrika*, vol. 78, no. 4, pp. 867–877, 1991.

[35] T. Gneiting and A. E. Raftery, 'Strictly proper scoring rules, prediction and estimation,' *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.

[36] T. Gneiting, F. Balabdaoui and A. E. Raftery, 'Probabilistic forecasts, calibration and sharpness,' *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 69, no. 2, pp. 243–268, 2007.

[37] E. Hazan, K. Y. Levy and S. Shalev-Shwartz, 'Beyond convexity: Stochastic quasi-convex optimization,' in *Advances in Neural Information Processing Systems*, vol. 28, 2015, pp. 1594–1602.

[38] M. Belkin and P. Niyogi, 'Laplacian eigenmaps for dimensionality reduction and data representation,' *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[39] U. von Luxburg, 'A tutorial on spectral clustering,' *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[40] R. J. Little and D. B. Rubin, *Statistical Analysis with Missing Data*, Third Edition. John Wiley & Sons, 2019.

[41] P. Frederic and F. Lad, 'Two moments of the logitnormal distribution,' *Communications in Statistics - Simulation and Computation*, vol. 37, no. 7, pp. 1263–1269, 2008.

[42] J. Aitchison and S. M. Shen, 'Logistic-normal distributions: Some properties and uses,' *Biometrika*, vol. 67, no. 2, pp. 261–272, 1980.

[43] A. Lau and P. McSharry, 'Approaches for multi-step density forecasts with application to aggregated wind power,' *The Annals of Applied Statistics*, vol. 4, no. 3, pp. 1311–1341, 2010.

[44] P. Pinson, 'Estimation of the uncertainty in wind power forecasting,' PhD Thesis, École Nationale Supérieure des Mines de Paris, 2006.

[45] M. Lange, 'On the uncertainty of wind power predictions - Analysis of the forecast accuracy and statistical distribution of errors,' *Journal of Solar Energy Engineering*, vol. 127, no. 2, pp. 177–184, 2005.

[46] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*. Springer, 1996.

[47] G. E. Box, G. M. Jenkins and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, Fourth Edition. John Wiley & Sons, 2008.

[48] E. Lesaffre, D. Rizopoulos and R. Tsonaka, 'The logistic transform for bounded outcome scores,' *Biostatistics*, vol. 8, no. 1, pp. 72–85, 2007.

[49] D. F. Heitjan and D. B. Rubin, 'Ignorability and coarse data,' *The Annals of Statistics*, vol. 19, no. 4, pp. 2244–2253, 1991.

[50] D. F. Heitjan, 'Ignorability and coarse data: Some biomedical examples,' *Biometrics*, vol. 49, no. 4, pp. 1099–1109, 1993.

[51] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[52] W. Zangwill, *Nonlinear Programming - A Unified Approach*. Prentice-Hall, 1969.

[53] T. Gneiting, K. Larson, K. Westrick, M. G. Genton and E. Aldrich, 'Calibrated probabilistic forecasting at the Stateline Wind Energy Center,' *Journal of the American Statistical Association*, vol. 101, no. 475, pp. 968–979, 2006.

[54] I. Sánchez, 'Short-term prediction of wind energy production,' *International Journal of Forecasting*, vol. 22, no. 1, pp. 43–56, 2006.

[55] C. Vincent, G. Giebel, P. Pinson and H. Madsen, 'Resolving nonstationary spectral information in wind speed time series using the Hilbert–Huang transform,' *Journal of Applied Meteorology and Climatology*, vol. 49, no. 2, pp. 253–267, 2010.

[56] L. Ljung and T. Södeström, *Theory and Practice of Recursive Identification*. The MIT Press, 1983.

[57] T. Gneiting, 'Making and evaluating point forecasts,' *Journal of the American Statistical Association*, vol. 106, no. 494, pp. 746–762, 2011.

[58] B. de Finetti, 'La prévision : Ses lois logiques, ses sources subjectives,' *Annales de l'institut Henri Poincaré*, vol. 7, no. 1, pp. 1–68, 1937.

[59] L. Savage, *The Foundations of Statistics*, Second Edition. Dover Publications, 1972.

[60] R. L. Winkler, J. Muñoz, J. Cervera *et al.*, 'Scoring rules and the evaluation of probabilities,' *Test*, vol. 5, no. 1, pp. 1–60, 1996.

[61] J. E. Matheson and R. L. Winkler, 'Scoring rules for continuous probability distributions,' *Management Science*, vol. 22, no. 10, pp. 1087–1096, 1976.

[62] I. J. Good, 'Rational decisions,' *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 14, no. 1, pp. 107–114, 1952.

[63] E. S. Epstein, 'A scoring system for probability forecasts of ranked categories,' *Journal of Applied Meteorology (1962-1982)*, vol. 8, no. 6, pp. 985–987, 1969.

[64] C.-A. S. Staël von Holstein, 'A family of strictly proper scoring rules which are sensitive to distance,' *Journal of Applied Meteorology (1962-1982)*, vol. 9, no. 3, pp. 360–364, 1970.

[65] T. L. Thorarinsdottir and T. Gneiting, 'Probabilistic forecasts of wind speed: Ensemble model output statistics by using heteroscedastic censored regression,' *Journal of the Royal Statistical Society Series A: Statistics in Society*, vol. 173, no. 2, pp. 371–388, 2010.

[66] S. Baran and S. Lerch, 'Log-normal distribution based ensemble model output statistics models for probabilistic wind-speed forecasting,' *Quarterly Journal of the Royal Meteorological Society*, vol. 141, no. 691, pp. 2289–2299, 2015.

[67] H. Hersbach, 'Decomposition of the continuous ranked probability score for ensemble prediction systems,' *Weather and Forecasting*, vol. 15, no. 5, 2000.

[68] K. Pearson, 'On a method of determining whether a sample of size n supposed to have been drawn from a parent population having a known probability integral has probably been drawn at random,' *Biometrika*, vol. 25, no. 3/4, pp. 379–410, 1933.

[69] M. Rosenblatt, 'Remarks on a multivariate transformation,' *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 470–472, 1952.

[70] F. X. Diebold, T. A. Gunther and A. S. Tay, 'Evaluating density forecasts with applications to financial risk management,' *International Economic Review*, vol. 39, no. 4, pp. 863–883, 1998.

[71] 'A comparison of financial duration models via density forecasts,' *International Journal of Forecasting*, vol. 20, no. 4, pp. 589–609, 2004.

[72] T. Gneiting, A. E. Raftery, A. H. Westveld and T. Goldman, 'Calibrated probabilistic forecasting using ensemble model output statistics and minimum CRPS estimation,' *Monthly Weather Review*, vol. 133, no. 5, pp. 1098–1118, 2005.

[73] T. M. Hamill, 'Interpretation of rank histograms for verifying ensemble forecasts,' *Monthly Weather Review*, vol. 129, no. 3, 2001.

[74] P. Pinson, P. McSharry and H. Madsen, 'Reliability diagrams for non-parametric density forecasts of continuous variables: Accounting for serial correlation,' *Quarterly Journal of the Royal Meteorological Society*, vol. 136, no. 646, pp. 77–90, 2010.

[75] J. B. Bremnes, 'Probabilistic forecasts of precipitation in terms of quantiles using NWP model output,' *Monthly Weather Review*, vol. 132, no. 1, 2004.

[76] H. Holzmann and M. Eulert, 'The role of the information set for forecasting—with applications to risk management,' *The Annals of Applied Statistics*, vol. 8, no. 1, pp. 595–621, 2014.

[77] Y. E. Nesterov, 'Minimization methods for nonsmooth convex and quasiconvex functions.,' *Matekon*, vol. 29, pp. 519–531, 1984.

[78] Y. Tsypkin, *Adaptation and Learning in Automatic Systems*. Academic Press, 1971.

[79] V. Vapnik, *Estimation of Dependences Based on Empirical Data*. Springer New York, NY, 1982.

[80] N. Cesa-Bianchi, A. Conconi and C. Gentile, 'On the generalization ability of on-line learning algorithms,' *IEEE Transactions on Information Theory*, vol. 50, no. 9, pp. 2050–2057, 2004.

[81] L. Bottou, 'On-line learning and stochastic approximations,' in *On-Line Learning in Neural Networks*. Cambridge University Press, 1999, pp. 9–42.

[82] H. Robbins and S. Monro, 'A stochastic approximation method,' *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[83] J. Duchi, E. Hazan and Y. Singer, 'Adaptive subgradient methods for online learning and stochastic optimization,' *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[84] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*, Second Edition. Springer New York, NY, 2009.

[85] M. M. Albert Benveniste and P. Priouret, *Adaptive Algorithms and Stochastic Approximations*. Springer Berlin, Heidelberg, 1990.

[86] J. G. Booth and J. P. Hobert, 'Maximizing generalized linear mixed model likelihoods with an automated Monte Carlo EM algorithm,' *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 61, no. 1, pp. 265–285, 1999.

[87] B. S. Caffo, W. Jank and G. L. Jones, 'Ascent-based Monte Carlo expectation-maximization,' *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 235–251, 2005.

[88] C. E. McCulloch, 'Maximum likelihood variance components estimation for binary data,' *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 330–335, 1994.

[89] C. E. McCulloch, 'Maximum likelihood algorithms for generalized linear mixed models,' *Journal of the American Statistical Association*, vol. 92, no. 437, pp. 162–170, 1997.

[90] W. R. Gilks, S. Richardson and D. J. Spiegelhalter, *Markov Chain Monte Carlo in Practice*. Springer New York, NY, 1995.

[91] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer New York, NY, 2004.

[92] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, 'Equation of state calculations by fast computing machines,' *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[93] W. K. Hastings, 'Monte carlo sampling methods using markov chains and their applications,' *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[94] S. F. Nielsen, 'The stochastic EM algorithm: Estimation and asymptotic results,' *Bernoulli*, vol. 6, no. 3, pp. 457–489, 2000.

[95] J. Chen, J. Zhu, Y. W. Teh and T. Zhang, 'Stochastic expectation maximization with variance reduction,' in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[96] S. Seaman, J. Galati, D. Jackson and J. Carlin, 'What is meant by "missing at random"?' *Statistical Science*, vol. 28, no. 2, pp. 257–268, 2013.

[97] D. B. Rubin, 'Inference and missing data,' *Biometrika*, vol. 63, no. 3, pp. 581–592, 1976.

[98] O. Anava, E. Hazan and A. Zeevi, 'Online time series prediction with missing data,' in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 37, 2015, pp. 2191–2199.

[99] A. Stratigakos, P. Andrianesis, A. Michiorri and G. Kariniotakis, 'Towards resilient energy forecasting: A robust optimization approach,' *IEEE Transactions on Smart Grid*, 2023.

[100] R. Tawn, J. Browell and I. Dinwoodie, 'Missing data in wind farm time series: Properties and effect on forecasts,' *Electric Power Systems Research*, vol. 189, p. 106 640, 2020.

[101] C. J. Stone, 'Consistent nonparametric regression,' *The Annals of Statistics*, vol. 5, no. 4, pp. 595–620, 1977.

[102] L. Devroye, L. Györfi and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. Springer New York, NY, 1996.

[103] R. J. Samworth, 'Optimal weighted nearest neighbour classifiers,' *The Annals of Statistics*, vol. 40, no. 5, pp. 2733–2763, 2012.

[104] G. H. Golub and C. F. V. Loan, *Matrix Computations, 4th edition*. Johns Hopkins University Press, 2013.

[105] A. Nadaraya, 'On estimating regression,' *Theory of Probability and Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[106] G. S. Watson, 'Smooth regression analysis,' *Sankhyā: the Indian Journal of Statistics, Series A*, vol. 26, no. 4, pp. 359–372, 1964.

[107] O. Anava and K. Levy, 'k^*-nearest neighbors: From global to local,' in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon and R. Garnett, Eds., vol. 29, 2016.

[108] S. Saelid and B. Foss, 'Adaptive controllers with a vector variable forgetting factor,' in *The 22nd IEEE Conference on Decision and Control*, 1983.

[109] J. Parkum, N. Poulsen and J. Holst, 'Selective forgetting in adaptive procedures,' *IFAC Proceedings Volumes*, vol. 23, no. 8, Part 2, pp. 137–142, 1990.

[110] L. Cao and H. Schwartz, 'A novel recursive algorithm for directional forgetting,' in *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, vol. 2, 1999, pp. 1334–1338.

[111] M. Zinkevich, 'Online convex programming and generalized infinitesimal gradient ascent,' in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ser. ICML'03, Washington, DC, USA, 2003, pp. 928–935.

[112] E. Hazan and C. Seshadhri, 'Efficient learning algorithms for changing environments,' in *Proceedings of the 26th International Conference on Machine Learning*, 2009, pp. 393–400.

[113] E. Hall and R. Willett, 'Dynamical models and tracking regret in online convex programming,' in *Proceedings of the 30th International Conference on Machine Learning*, vol. 28, 2013, pp. 579–587.

[114] C. P. Robert, *The Bayesian Choice*. Springer New York, NY, 2007.

[115] A. Timmermann, 'Chapter 4 forecast combinations,' in ser. Handbook of Economic Forecasting, G. Elliott, C. Granger and A. Timmermann, Eds., vol. 1, Elsevier, 2006, pp. 135–196.

[116] K. C. Lichtendahl, Y. Grushka-Cockayne and R. L. Winkler, 'Is it better to average probabilities or quantiles?' *Management Science*, vol. 59, no. 7, pp. 1594–1611, 2013.

[117] J. A. Hoeting, D. Madigan, A. E. Raftery and C. T. Volinsky, 'Bayesian model averaging: A tutorial,' *Statistical Science*, vol. 14, no. 4, pp. 382–401, 1999.

[118] A. E. Raftery, M. Kárný and P. Ettler, 'Online prediction under model uncertainty via dynamic model averaging: Application to a cold rolling mill,' *Technometrics*, vol. 52, no. 1, pp. 52–66, 2010.

[119] J. Dowell and P. Pinson, 'Very-short-term probabilistic wind power forecasts by sparse vector autoregression,' *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 763–770, 2016.

[120] J. W. Messner and P. Pinson, 'Online adaptive lasso estimation in vector autoregressive models for high dimensional wind power forecasting,' *International Journal of Forecasting*, vol. 35, no. 4, pp. 1485–1498, 2019.

[121] S. B. Taieb, J. W. Taylor and R. J. Hyndman, 'Hierarchical probabilistic forecasting of electricity demand with smart meter data,' *Journal of the American Statistical Association*, vol. 116, no. 533, pp. 27–43, 2021.

Collection of relevant publications

[**Paper A**] A. Pierrot and P. Pinson, "Adaptive generalized logit-normal distributions for wind power short-term forecasting", in *Proceedings 2021 IEEE Madrid PowerTech*, 2021.

[**Paper B**] A. Pierrot and P. Pinson, "On tracking varying bounds when forecasting bounded time series", submitted to *Technometrics*, 2023.

[**Paper C**] A. Pierrot and P. Pinson, "Mixtures of bounded distributions with different bounds: Estimation and forecasting", to be submitted to the *Annals of Applied Statistics*, technical report as of now, 2023.

[**Paper D**] A. Pierrot and P. Pinson, "Data is missing again – Reconstruction of power generation data using k-nearest neighbors and spectral graph theory", submitted to *Wind Energy*, 2023.

[Paper A] Adaptive generalized logit-normal distributions for wind power short-term forecasting

Authors:
Amandine Pierrot and Pierre Pinson

Published in:
Proceedings of IEEE Madrid PowerTech Conference 2021

DOI:
10.1109/PowerTech46648.2021.9494900

Adaptive Generalized Logit-Normal Distributions for Wind Power Short-Term Forecasting

Amandine Pierrot
Technical University of Denmark
Kgs Lyngby, Denmark
amapi@dtu.dk

Pierre Pinson
Technical University of Denmark
Kgs Lyngby, Denmark
ppin@dtu.dk

***Abstract*—There is increasing interest in very short-term and higher-resolution wind power forecasting (from mins to hours ahead), especially offshore. Statistical methods are of utmost relevance, since weather forecasts cannot be informative for those lead times. Those approaches ought to account for the fact wind power generation as a stochastic process is non-stationary, double-bounded (by zero and the nominal power of the turbine) and non-linear. Accommodating those aspects may lead to improving both point and probabilistic forecasts. We propose to focus on generalized logit-normal distributions, which are naturally suitable and flexible for double-bounded and non-linear processes. Relevant parameters are estimated via maximum likelihood inference. Both batch and online versions of the estimation approach are described – the online version permitting to additionally handle non-stationarity through parameter variation. The approach is applied and analysed on the test case of the Anholt offshore wind farm in Denmark, with emphasis placed on 10-min-ahead forecasting.**

***Index Terms*—Wind power, Probabilistic forecasting, Dynamic models, Bounded time-series**

I. Introduction

Forecasting is of utmost importance to the integration of renewable energy into power systems and electricity markets. The attention of energy forecasting has increased tremendously over the years [1]. For instance, thinking of short-term operational problems, transmission system operators (TSOs) have to operate reserves optimally to keep the system in balance at reasonable costs. Indeed, in Denmark, the TSO has some time argued the 10-min lead time as the most important since wind power fluctuations at this horizon particularly affect the system balance, see [2] for instance. Emphasis here is on offshore wind power forecasting, since those short-term fluctuations in power generation are most significant offshore. Even though most efforts in wind power forecasting are placed on lead times from hours to days, many are investing in alternative approaches to improve the accuracy of very short-term forecasts, for instance leveraging detailed turbine-level data [3]. Those very short-term lead times are not only crucial but also those it is the most difficult to improve the forecasts for, especially compared to the simple but very effective persistence benchmark. Forecasts characterize and reduce but do not eliminate uncertainty. Thus forecasts should be probabilistic in nature taking the form of probability distributions, following the argument of [4] among others.

Wind power generation is a stochastic process which is double-bounded by nature, both by zero when there is no production at all, and by the nominal power for high-enough wind speeds. For short-term forecasting, statistical methods have proved to be more skilled and accurate. However, those methods often rely on a Gaussian assumption – which cannot be appropriate for a double-bounded variable. In [5], it is proposed to move from the classical Gaussian assumption to a framework where the wind power variable follows a generalized logit-normal distribution. In this framework though, not all the parameters of the distribution are estimated and tracked, the shape parameter being selected upon cross-validation.

Consequently here, we propose to revisit this work and to estimate all the parameters of the generalized logit-normal distributions within a maximum likelihood framework. Such a framework is particularly suitable to obtain skilled probabilistic forecasts. In addition, emphasis is placed on describing both batch and recursive estimation approaches, in order to go towards an online learning approach as a basis for probabilistic forecasting. For a nice introduction to online learning, the reader is referred to [6]. Online learning (with exponential forgetting) makes it possible to accommodate the non-stationarity of wind power generation time-series. The models and estimation framework are first presented in Section II, and the resulting algorithms in Section III. They are then applied to 10-min-ahead point and probabilistic forecasting at the Anholt offshore wind farm in Section IV. Finally some concluding remarks and prospects are given in Section V.

The research leading to this work is being carried out as a part of the Smart4RES project (European Union's Horizon 2020, No. 864337). The sole responsibility of this publication lies with the authors. The European Union is not responsible for any use that may be made of the information contained therein. The authors additionally acknowledge Ørsted for providing the data for the Anholt offshore wind farm.

978-1-6654-3597-0/21/$31.00 ©2021 European Union

II. Model and Estimation Framework

A. Generalized Logit-Normal Distribution and its Parameters

For an original random variable $X \in (0, 1)$, the generalized logit transform Y is given by

$$Y = \gamma(X; \nu) = \ln\left(\frac{X^\nu}{1 - X^\nu}\right), \quad \nu > 0, \tag{1}$$

where ν is the shape parameter. When Y follows a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, the original variable X follows a generalized logit-normal distribution $L_\nu(\mu, \sigma^2)$, see [5]. The probability density function is given by

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \frac{\nu}{x(1-x^\nu)} \exp\left[-\frac{1}{2}\left(\frac{\gamma(x;\nu)-\mu}{\sigma}\right)^2\right]. \quad (2)$$

Let X the wind power random variable. We want ν such as the transform variable Y is as close as possible to a Gaussian variable which then can be forecast in a Gaussian framework. As we have access to some realizations (x_t) of X and to the analytical expression of its density, we can then maximize the probability of observing the data (x_t) depending on ν, μ and σ^2, that is estimate all the parameters of the distribution (2) using maximum likelihood inference.

In the case of wind power generation, the observations (x_t) are strongly correlated. We thus assume that $Y_t|Y_{t-1}, ..., Y_{t-p} \sim \mathcal{N}(\mu_t, \sigma^2)$ where $\mu_t = \sum_{k=1}^p \phi_k Y_{t-k}$, that is the distribution of $X_t|X_{t-1}, ..., X_{t-p}$ is a generalized logit-normal distribution of density

$$\frac{1}{\sqrt{2\pi\sigma^2}} \frac{\nu}{x_t(1-x_t^\nu)} \exp\left[-\frac{1}{2}\left(\frac{y_t - \sum_{k=1}^p \phi_k y_{t-k}}{\sigma}\right)^2\right], \quad (3)$$

where $y_t = \gamma(x_t; \nu)$. While the density in (3) is defined only for $x \in (0, 1)$, the wind power generation can take values 0 and 1. We thus choose to look at the observations $x_t \in [0, 1]$ as a coarsened version of X, see [7]. This coarsened data framework has been formalized by [8] and [9].

B. Maximum Likelihood Inference

Let $\Phi = (\phi_1, ..., \phi_p)^\top \in \mathbb{R}^p$. The maximum likelihood inference is based on the likelihood function, given by

$$L(\nu, \Phi, \sigma^2|\mathbf{x}) = \prod_{t=1}^N f(x_t|x_{t-1}, ..., x_{t-p}, \nu, \Phi, \sigma^2), \quad (4)$$

which is the probability of the observed data under the model f, assuming the realizations of $X_t|X_{t-1}, ..., X_{t-p}$ are independent and identically distributed. We think of $L(\nu, \Phi, \sigma^2|\mathbf{x})$ as a function of ν, Φ and σ^2, the data (x_t) being fixed. The method of maximum likelihood chooses the values $(\nu, \Phi, \sigma^2) = (\hat{\nu}, \hat{\Phi}, \hat{\sigma}^2)$ to maximize $L(\nu, \Phi, \sigma^2|\mathbf{x})$. The logarithm of L being easier to maximize, especially when exponential distributions are involved, it is used instead of the likekihood. For model f the negative log-likelihood function is

$$\begin{aligned}\tilde{l}(\nu, \Phi, \sigma^2|\mathbf{x}) = &\frac{N-p}{2}\ln(\sigma^2) - (N-p)\ln(\nu) \\ &+ \sum_{t=p+1}^N \ln(1-x_t^\nu) \\ &+ \frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{Y}\Phi)^\top(\mathbf{y}-\mathbf{Y}\Phi) + C,\end{aligned} \quad (5)$$

where $\mathbf{y} = (y_{p+1}, ..., y_N)^\top \in \mathbb{R}^{N-p}$, $\mathbf{Y}$ is a matrix with columns $B\mathbf{y}, B^2\mathbf{y}, ..., B^p\mathbf{y} \in \mathbb{R}^{(N-p)\times p}$, B being the backshift operator, C is a constant which does not depend on ν, Φ or σ^2. Computing the first derivatives of (5) w.r.t. the parameters of the distribution we can retrieve stationary points. It is worth noting that those points are minimizers only if the negative log-likelihood is convex. Taking the derivative of (5) w.r.t. Φ, resp. σ^2, and setting it equal to zero, leads to the usual maximum likelihood estimators

$$\hat{\Phi} = (\mathbf{Y}^\top\mathbf{Y})^{-1}\mathbf{Y}^\top\mathbf{y}, \qquad \hat{\sigma}^2 = \frac{(\mathbf{y}-\mathbf{Y}\hat{\Phi})^\top(\mathbf{y}-\mathbf{Y}\hat{\Phi})}{N-p}. \quad (6)$$

Taking the derivative of (5) w.r.t. ν, we thus need to solve

$$-\frac{N-p}{\nu} - \sum_{t=p+1}^N \frac{\ln(x_t){x_t}^\nu}{1-{x_t}^\nu} + \frac{(\mathbf{u}-\mathbf{U}\Phi)^\top(\mathbf{y}-\mathbf{Y}\Phi)}{\sigma^2} = 0, \quad (7)$$

where $\mathbf{u} = \frac{\partial \mathbf{y}}{\partial \nu}$ with $u_t = \ln(x_t)(1 + \frac{{x_t}^\nu}{1-{x_t}^\nu})$, $\mathbf{U} = \frac{\partial \mathbf{Y}}{\partial \nu}$ with columns $B\mathbf{u}, B^2\mathbf{u}, ..., B^p\mathbf{u}$. Unlike $\hat{\Phi}$ and $\hat{\sigma}^2$, $\hat{\nu}$ does not have a closed-form solution and a descent algorithm is then to be used to solve (7).

III. Batch and Recursive Algorithms

A. Batch Algorithm

We use both the closed-form solutions in (6) for $\hat{\Phi}$ and $\hat{\sigma}^2$, and a Newton-Raphson algorithm to solve (7) in order to estimate the shape parameter ν. The computation of the Newton-Raphson step requires the second derivative of (5) w.r.t. ν, i.e.

$$\begin{aligned}\frac{\partial^2 \tilde{l}}{\partial \nu^2} = &\frac{N-p}{\nu^2} - \sum_{t=p+1}^N \ln(x_t)^2 \frac{x_t^\nu}{(1-x_t^\nu)^2} \\ &+ \frac{(\mathbf{v}-\mathbf{V}\Phi)^\top(\mathbf{y}-\mathbf{Y}\Phi)}{\sigma^2} + \frac{\| \mathbf{u}-\mathbf{U}\Phi \|_2^2}{\sigma^2},\end{aligned} \quad (8)$$

where $\mathbf{v} = \frac{\partial \mathbf{u}}{\partial \nu}$ with $v_t = u_t \ln(x_t)\frac{{x_t}^\nu}{1-{x_t}^\nu}$, $\mathbf{V} = \frac{\partial \mathbf{U}}{\partial \nu}$ with columns $B\mathbf{v}, B^2\mathbf{v}, ..., B^p\mathbf{v}$.

The full algorithm is described in Algorithm 1 and has showed very fast convergence on numerous simulations of samples distributed according to the generalized logit-normal distribution with different values of Φ, σ^2 and ν.

Algorithm 1 Batch MLE with diagonalization

Set $i \leftarrow 1$ and let $\nu_1 = 1$, $\epsilon = 0.001$.
repeat
1. *Update.* $\Phi_i = (\mathbf{Y}^\top\mathbf{Y})^{-1}\mathbf{Y}^\top\mathbf{y}$; $\sigma_i^2 = \frac{(\mathbf{y}-\mathbf{Y}\Phi_i)^\top(\mathbf{y}-\mathbf{Y}\Phi_i)}{N-p}$.
2. *Compute the Newton step and decrement for* ν.
$\Delta\nu_{nt} = -\frac{\nabla_\nu \tilde{l}}{\nabla_\nu^2 \tilde{l}}$; $\lambda^2 = \frac{(\nabla_\nu \tilde{l})^2}{\nabla_\nu^2 \tilde{l}}$.
3. *Stopping criterion.* **quit** if $\lambda^2/2 \leq \epsilon$.
4. *Line search.* Choose step size t by backtracking line search.
5. *Update.* $\nu_{i+1} = \nu_i + t\Delta\nu_{nt}$.
until termination test satisfied.

B. Recursive Algorithm

The batch algorithm is well suited if the data are known to be stationary to second order, that is assuming the parameters of the distribution (2) do not change over the curse of time. But if, as we suspect in the case of wind power data, the time series is not stationary and the parameters are not constant, then the batch algorithm is not appropriate and alternative solutions are required. Recursive estimation allows for such a parametric time-variability and provides information not only on the existence of non-stationarity but also on the possible nature of the parametric variations (see e.g. [10]).

As the inference relies on the likelihood function, it is straightforward to derive a recursive algorithm which only requires the first derivatives of (5) w.r.t. to the parameters. Let introduce $\hat{\Theta}_t = (\hat{\Phi}_t, \hat{\sigma}_t^2, \hat{\nu}_t)$ the estimate of the parameters at time t. The recursive estimation procedure relies on a Newton-Raphson step for obtaining the estimate $\hat{\Theta}_t$ as a function of the previous estimate $\hat{\Theta}_{t-1}$, see e.g. [11] and [12]. Let introduce the time-dependent negative log-likelihood objective function to be minimized at time t

$$S_t(\Theta) = -\frac{1}{n_\alpha} \sum_{j=p+1}^{t} \alpha^{t-j} \ln(f_j(\Theta)), \tag{9}$$

where $f_j(\Theta) = f(x_j|x_{j-1}, ..., x_{j-p}; \Theta)$, α is a forgetting factor, $\alpha \in (0, 1)$, allowing for exponential forgetting of past observations, $n_\alpha = \frac{1}{1-\alpha}$ is the effective number of observations used for normalizing the weighted negative log-likelihood function. Applying one Newton-Raphson step we have

$$\hat{\Theta}_t = \hat{\Theta}_{t-1} - \frac{\nabla_\Theta S_t(\hat{\Theta}_{t-1})}{\nabla^2_\Theta S_t(\hat{\Theta}_{t-1})}. \tag{10}$$

As

$$\begin{aligned} \nabla_\Theta S_t(\hat{\Theta}_{t-1}) = \alpha \nabla_\Theta S_{t-1}(\hat{\Theta}_{t-1}) \\ - (1-\alpha)\nabla_\Theta \ln(f_t(\hat{\Theta}_{t-1})), \end{aligned} \tag{11}$$

assuming that $\hat{\Theta}_{t-1}$ minimizes $S_{t-1}(\Theta)$, we get

$$\nabla_\Theta S_t(\hat{\Theta}_{t-1}) = -(1-\alpha)\nabla_\Theta \ln(f_t(\hat{\Theta}_{t-1})). \tag{12}$$

From (11) we also get

$$\begin{aligned} \nabla^2_\Theta S_t(\hat{\Theta}_{t-1}) = \alpha \nabla^2_\Theta S_{t-1}(\hat{\Theta}_{t-1}) \\ - (1-\alpha)\nabla^2_\Theta \ln(f_t(\hat{\Theta}_{t-1})). \end{aligned} \tag{13}$$

As

$$\begin{aligned} \nabla^2_\Theta \ln(f_t(\hat{\Theta}_{t-1})) = \frac{\nabla^2_\Theta f_t(\hat{\Theta}_{t-1})}{f_t(\hat{\Theta}_{t-1})} \\ - \frac{\nabla_\Theta f_t(\hat{\Theta}_{t-1})(\nabla_\Theta f_t(\hat{\Theta}_{t-1}))^T}{f_t(\hat{\Theta}_{t-1})^2}, \end{aligned} \tag{14}$$

assuming f_t is (almost) linear in Θ in the neighborhood of $\hat{\Theta}_{t-1}$, the first term in (14) vanishes and we obtain the following approximation

$$\nabla^2_\Theta \ln(f_t(\hat{\Theta}_{t-1})) = -\mathbf{h}_t \mathbf{h}_t^\top, \tag{15}$$

where $\mathbf{h}_t = \frac{\nabla_\Theta f_t(\hat{\Theta}_{t-1})}{f_t(\hat{\Theta}_{t-1})} = \nabla_\Theta \ln(f_t(\hat{\Theta}_{t-1}))$.

Let $\hat{\mathbf{R}}_t = \nabla^2_\Theta S_t(\hat{\Theta}_t)$ and assume that the objective criterion S is smooth in the vicinity of $\hat{\Theta}_t$, and the adaptation step small enough so that

$$\hat{\mathbf{R}}_t = \nabla^2_\Theta S_t(\hat{\Theta}_t) \simeq \nabla^2_\Theta S_t(\hat{\Theta}_{t-1}). \tag{16}$$

This is a classic assumption for deriving recursive estimation methods for stochastic systems (see [13]). The two-step recursive scheme at time t is then

$$\hat{\mathbf{R}}_t = \alpha \hat{\mathbf{R}}_{t-1} + (1-\alpha)\mathbf{h}_t \mathbf{h}_t^\top, \tag{17}$$

$$\hat{\Theta}_t = \hat{\Theta}_{t-1} + (1-\alpha)\hat{\mathbf{R}}_t^{-1}\mathbf{h}_t. \tag{18}$$

Equation (17) derives from (13) and (15). Equation (18) derives from (10), (12) and (16). The final algorithm is available in Algorithm 2.

Algorithm 2 Recursive MLE

Let $\Phi_0 = \mathbf{0}, \sigma_0^2 = 1, \nu_0 = 1, \mathbf{h}_0 = \mathbf{0}, R_0 = 0_{(p+2,p+2)}$.
repeat
1. *Update.* $\hat{\mathbf{R}}_i = \alpha \hat{\mathbf{R}}_{i-1} + (1-\alpha)\mathbf{h}_i \mathbf{h}_i^\top$.
2. *Update.* $\hat{\Theta}_i = \hat{\Theta}_{i-1} + (1-\alpha)\hat{\mathbf{R}}_i^{-1}\mathbf{h}_i$ if $i > 100 + p$.

until t the forecasting time.

IV. Very-short-term Wind Power Forecasting Application

We apply the proposed models to a real dataset consisting of wind power generation from a large wind farm, Anholt in Denmark, from July 1, 2013 to August 31, 2014. Emphasis is placed on the maximum likelihood framework and its online learning derivation. For a comparison of the generalized logit-normal distribution to other distributions (e.g., Beta) for the purpose of wind power forecasting, see [5].

A. Data Description

Active power is available for 110 wind turbines at a temporal resolution of every 10 minute. The time series are scaled individually according to the nominal power of the wind turbines. The average generation over the wind farm is then computed depending on the number of wind turbines being available at each time step, in order to handle missing values. The resulting random variable is then $X_t \in [0, 1]$, the average active power generated in the wind farm at time t.

We are interested in forecasting X_{t+1} (point forecasting) and its distribution (probabilistic forecasting) knowing the realization of X_t; the lead time is therefore 10-minute-ahead. We split our data into two datasets:

- a training/cross-validation dataset from July 1, 2013 to March 31, 2014, resulting in 39,450 observations,
- a test dataset from April 1 to August 31, 2014, resulting in 22,029 observations.

The training set is used to fit all models, the cross-validation set to select hyper-parameters if needed and the test set to compare the proposed methodology to the benchmarks. It is

worth noting the training set is long enough for the Algorithm 2 to be recursive yet on the training period, after a short warm-up of 100 iterations.

B. Point Forecasting

In order to evaluate and compare the performance of the proposed methods for point forecasting we use the Root Mean Square Error (RMSE). When a model requires hyper-parameters to be selected before estimating the parameters, we use the following procedure:

- The candidate models are fitted over a grid of hyper-parameters' values from July 1 to October 31, 2013;
- they are then retrained in a time-series cross-validation scheme, from November 1, 2013 to March 31, 2014, for which the size of the training window increases as we evolve through the validation set (consistent with a leave-one-out setup);
- the hyper-parameters leading to the smallest RMSE on the cross-validation set are selected;
- finally the final model is fitted over the whole training/cross-validation set and used for forecasting on the test set.

a) Benchmarks: We compare our methods to three benchmarks: the persistence, a normal auto-regressive (NAR) model and its recursive version. The persistence consists in taking $\hat{x}_{t+1} = x_t$. The normal AR model assumes $X_t|X_{t-1},...,X_{t-p} \sim \mathcal{N}(\mu_t, \sigma^2)$ where $\mu_t = \sum_{k=1}^{p} \phi_k X_{t-k}$. In this Gaussian setup the forecasts are unbounded and happen to be greater than 1 or lower than 0. Thus we need to truncate *a posteriori* the out-of-range predictions so they lie in the interval $[0, 1]$. We test AR models up to lag $p = 5$ and observe that no significant improvement is provided beyond lag 2 for both batch and recursive approaches. We thus select $p = 2$. For the recursive AR model we also need to select the forgetting factor α, which exponentially weights data in the past. In a similar way, it is selected such as $\alpha = 0.995$.

b) Forecasting using generalized logit-normal distributions: Let $\delta > 0$ such as each value being lower than δ (resp. greater than $1 - \delta$) is set to δ (resp. $1 - \delta$) and consider those "corrected" observations as the realizations of $X \in (0, 1)$. In a symmetric way, forecasts being lower than δ (resp. greater than $1 - \delta$) will be set to 0 (resp. 1). δ is selected over cross-validation along with p. Algorithm 1 converges in 11 iterations towards the estimated values $\hat{\nu} = 1.39$, $\hat{\Phi} = (1.363, -0.370)^\top$ and $\hat{\sigma}^2 = 0.11$ for the selected combination of hyper-parameters $\delta = 0.005$ and $p = 2$. For Algorithm 2, we choose $\delta = 0.005$, $p = 2$ and $\alpha = 0.9994$ upon cross-validation. See in Fig. 1 the estimated parameters of the generalized logit-normal distributions over the test period.

c) Results: The point forecasting performance over the test set of the benchmarks and the (GLNAR) proposed algorithms are available in Table I. It is worth noting that the test set consists in 22,023 observations, which is a volume of data large enough to claim for significant results. The best point forecasts are obtained by the model using adaptive generalized

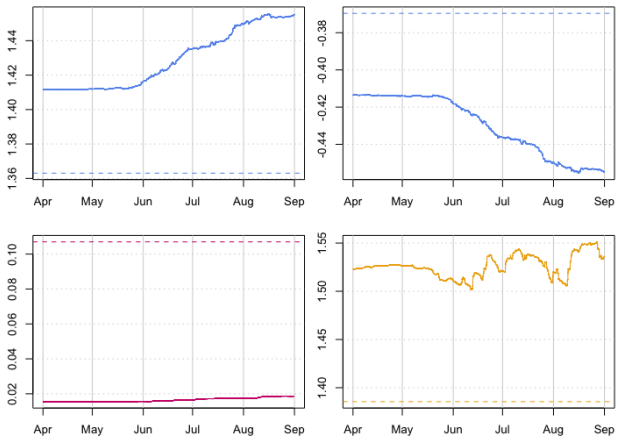


Fig. 1. Parameters of the generalized logit-normal distribution for $p = 2$ and $\alpha = 0.9994$: $\hat{\Phi}$ (top), $\hat{\sigma}^2$ (bottom left) and $\hat{\nu}$ (bottom right).

logit-normal distributions. One can note that the model which uses a constant generalized logit-normal distribution gets poorer performance than the recursive AR model. Therefore the assumption that seems to matter the most here is the time-varying parameters assumption. Moreover, the estimated value of the scale parameter is significantly larger in the batch setup than in the recursive one, while the shape parameter is significantly lower. It may confirm that the recursive setup is more appropriate to the characteristics of the time series and thus allows for a better discrimination between the scale and the shape parameters of the distribution.

TABLE I
10-MINUTE-AHEAD RMSE OVER THE TEST PERIOD, AND RESPECTIVE IMPROVEMENTS OVER PERSISTENCE

Model	RMSE	Imp. over persist.
persistence	3.27%	-
batch NAR	2.79%	14.68%
recursive NAR	2.72%	16.82%
batch GLNAR	2.74%	16.21%
recursive GLNAR	**2.70%**	**17.43%**

**Best forecast bolded.*

C. Probabilistic Forecasting

Let F_t a predictive cumulative distribution function at time t. The Continuous Ranking Probabilistic Score (CRPS) is defined by

$$\text{CRPS} = \frac{1}{T}\sum_{t=1}^{T} \text{crps}(F_t, x_t) = \int_{-\infty}^{\infty} \text{BS}(y)\text{d}y, \qquad (19)$$

where

$$\text{crps}(F_t, x_t) = \int_{-\infty}^{\infty} \{F_t(y) - \mathbf{1}(y \geq x_t)\}^2 \text{d}y, \qquad (20)$$

and BS is the Brier score

$$\text{BS}(y) = \frac{1}{T}\sum_{t=1}^{T} \{F_t(y) - \mathbf{1}(x_t \leq y)\}^2. \qquad (21)$$

See for example [14] and [15]. To evaluate the performance of the proposed models for probabilistic forecasting we use the

CRPS instead of the RMSE, following the scheme described at the beginning of section IV-B.

a) Benchmarks: We compare our method to four benchmarks: climatology, probabilistic persistence, and probabilistic versions of the batch and recursive AR models. Climatology consists in computing empirical quantiles on the training set. We test different grids and choose upon cross-validation to estimate the predictive cumulative distribution from the quantiles $\{0, 0.01, ..., 0.99, 1\}$. On the test set the quantiles are updated whenever a new observation is recorded. Probabilistic persistence consists in dressing the point persistence prediction with the most recent observed values of the persistence error. We choose the number of observed values upon cross-validation to be 20. For probabilistic AR forecasts, the least squares estimator of the variance of the residuals is used in both batch and recursive modes, and we assume those residuals to follow a Gaussian distribution $\mathcal{N}(0, \hat{\sigma}^2)$. The forecast distribution of x_t is then a Gaussian distribution $\mathcal{N}(\hat{x}_t, \hat{\sigma}^2)$ where $\hat{x}_t$ is the point forecast from the AR model. The hyper-parameters p and α for the recursive model are selected upon cross-validation with CRPS, which leads to $p = 2$ as for point forecasting, but to a different α which is now equal to 0.983 instead of 0.995.

b) Forecasting using generalized logit-normal distributions: The lag p selected upon cross-validation with CRPS remains equal to 2 in both batch and recursive algorithms, while δ and α change. For Algorithm 1, now $\delta = 0.006$ which leads to slightly different estimated parameters of the distribution: $\hat{\nu} = 1.37$ and $\hat{\Phi} = (1.358, -0.365)^\top$, while the variance $\hat{\sigma}^2 = 0.11$ remains the same. For Algorithm 2, now $\delta = 0.004$ and α decreases from 0.9994 for point forecasting to 0.9986 for probabilistic forecasting. See in Fig. 2 the estimated parameters of the generalized logit-normal distributions over the test period, which show higher time-variability because of the lower value of the forgetting factor.

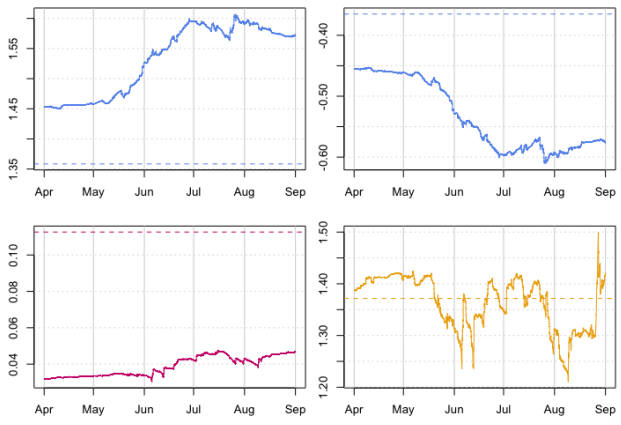


Fig. 2. Temporal evolution of the parameters of the generalized logit-normal distributions for $p = 2$ and $\alpha = 0.9994$: $\hat{\Phi}$ (top), $\hat{\sigma}^2$ (bottom left) and $\hat{\nu}$ (bottom right).

c) Results: The CRPS computed over the test set for all the benchmarks and the proposed models are available in Table II. The climatology's predictive cumulative distribution function F_{t+1} remains unchanged whatever the value of x_t, which explains the very poor global performance of this method. The performance of the predictive cumulative distributions assuming a Gaussian setup and that of the approach using a constant generalized logit-normal distribution are close as for point forecasting. However, for probabilistic forecasting, the approach using adaptive generalized logit-normal distributions outperforms the other methods. The Brier scores are plotted in Fig. 3. As expected the methods using the generalized logit transformation perform better close to the bounds of the interval $[0, 1]$.

TABLE II
10-MINUTE-AHEAD CRPS OVER THE TEST PERIOD, AND RESPECTIVE IMPROVEMENTS OVER CLIMATOLOGY AND PERSISTENCE

Model	CRPS	Imp. over clim.	Imp. over persist.
climatology	22.04%	-	-
prob. persistence	1.36%	93.85%	-
batch NAR	1.28%	94.17%	5.28%
recursive NAR	1.23%	94.34%	9.40%
batch GLNAR	1.21%	94.52%	10.90%
recursive GLNAR	**1.06%**	**95.17%**	**21.57%**

**Best forecast bolded.*

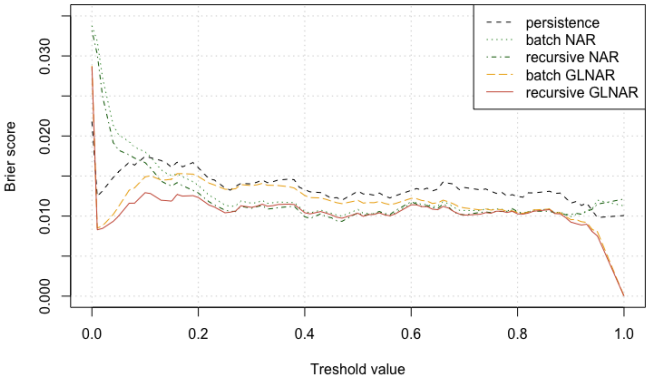


Fig. 3. Brier score computed over the test set for all methods but climatology, as a function of the chosen threshold.

The CRPS and the Brier score give indications about the sharpness of the distributions. In order to check the calibration we show the results of two tools: the reliability diagram in Fig. 4 and a marginal calibration plot which is the difference between the average predictive $\bar{F}$ on the test set and the empirical cumulative distribution function in Fig. 5. For the reliability diagram, the closer to the diagonal, the better the calibration, the empirical probabilities getting closer to the nominal ones. See [16] and [15] for more details about those calibration tools. One can see that for both indicators the approach using adaptive generalized logit-normal distributions outperforms the other probabilistic forecasting methods. In Fig. 5 the climatology difference is not presented for being far bigger than zero.

Example probabilistic forecasts obtained from the adaptive generalized logit-normal approach over a 36 hour period of time are depicted in Fig. 6 by using prediction intervals with nominal coverage rates of 95 and 75%.

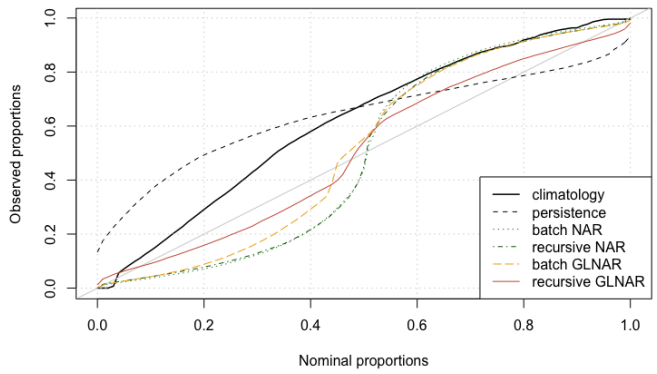


Fig. 4. Reliability diagram over the test set.

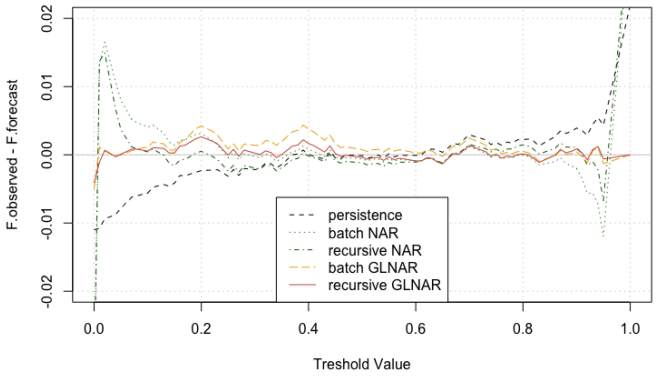


Fig. 5. Marginal calibration plot over the test set.

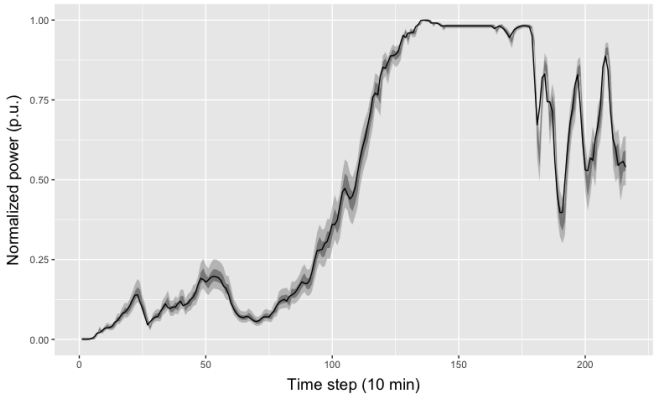


Fig. 6. Probabilistic forecasts from the recursive approach (Algorithm 2), based on prediction interval with nominal coverage rates of 95 and 75%, along with the power measurements (solid black line).

V. Conclusions

A generalized logit-normal distribution was considered for very short-term wind power forecasting, in order to adequately handle the double-bounded nature of the time series. All the parameters of the distribution were estimated from the data in a maximum likelihood framework, for both batch and online setups. The adaptive version of the distribution provides only a slight improvement in the accuracy of the point forecasts compared to approaches within a Gaussian framework, though it substantially outperforms the other benchmarks when focusing on probabilistic forecasting (intervals and full predictive densities). This confirms that such a choice of distribution may be most appropriate. While it achieves better calibration and sharpness, there is still room for improvement. In particular, we have emphasized the importance of the double-bounded nature of the process, but in practice the upper bound may also change in time. Indeed, wind power generation is not always bounded by the nominal capacity of the wind farm, e.g. in case of curtailment. It should then be taken into account within the modelling and forecasting framework, by additionally adaptively estimating this upper bound from data.

Furthermore, the proposed framework could be applied for multi-step ahead forecasting, and makes it easy to assume other models for the conditional expectation of the transformed variable. In particular it is straightforward to add exogenous variables to the auto-regressive model, or to generalize it with a non-linear one. This may be a way to account for the individual productions of the wind turbines in order to improve the prediction of power generation for the whole wind farm. Finally, the δ hyper-parameter which handles the coarsened version of the distribution was selected upon cross-validation. It could instead enter a Bayesian or a likelihood inference as a parameter to be properly estimated.

References

[1] T. Hong, P. Pinson, Y. Wang, R. Weron, D. Yang and H. Zareipour, "Energy forecasting: A review and outlook," IEEE Open Access Journal of Power and Energy, vol. 7, pp. 376-388, 2020.

[2] V. Akhmatov, "Influence of wind direction on intense power fluctuations in large offshore wind farms in the North Sea," Wind Energ., vol. 31(1), pp. 59–64, 2007.

[3] C. Gilbert, J. Browell and D McMillan, "Leveraging turbine-level data for improved probabilistic wind power forecasting," IEEE Trans. Sust. Energ., vol. 11, no. 3, pp. 1152–1160, 2019.

[4] A.P. Dawid, "Statistical theory: the prequential approach", J. R. Statist. Soc. A, vol. 157(2), pp. 278–292, 1984.

[5] P. Pinson, "Very-short-term probabilistic forecasting of wind power with generalized logit-normal distributions," J. R. Statist. Soc. C, vol. 61(4), pp. 555–576, 2012.

[6] F. Orabona. A Modern Introduction to Online Learning. Lecture notes, Boston University, 2020.

[7] E. Lesaffre, D. Rizopoulous and R. Tsonaka, "The logistic transform for bounded outcome scores," Biostatistics, vol. 8(1), pp. 72–85, 2007.

[8] D. Heijtan and D. Rubin, "Ignorability and coarse data," Ann. Stat., vol. 19(4), pp. 2244–2253, 1991.

[9] D. Heijtan, "Ignorability and coarse data: some biomedical examples," Biometrics, vol. 49(4), pp. 1099–1109, 1993.

[10] P. Young, Recursive estimation and time-series analysis: An introduction. Springer-Verlag, Berlin, Heidelberg, 1984.

[11] H. Madsen, Time Series Analysis. Chapman & Hall, Boca Raton, 2007.

[12] P. Pinson and H. Madsen, "Adaptive modelling and forecasting of offshore wind power fluctuations with Markov-switching autoregressive models," J. Forecast., vol. 31, pp. 281–313, 2012.

[13] L. Ljung and T. Söderström, Theory and Practice of Recursive Estimation, 1983.

[14] G.W. Brier, "Verification of forecasts expressed in terms of probability," Monthly Weather Review, vol. 78(1), pp. 1–3, 1950.

[15] T. Gneiting, F. Balabdaoui and A.E. Raftery, "Probabilistic forecasts, calibration and sharpness," J. R. Statist. Soc. B, vol. 69(2), pp. 243–268, 2007.

[16] P. Pinson, H.Aa. Nielsen, J.K. Møller and H. Madsen, "Non-parametric probabilistic forecasts of wind power: Required properties and evaluation," Wind Energ., vol. 10(6), pp. 497–516, 2007.

[Paper B] On tracking varying bounds when forecasting bounded time series

Authors:
Amandine Pierrot and Pierre Pinson

Submitted to:
Technometrics

On tracking varying bounds when forecasting bounded time series

Amandine Pierrot[†,*], Pierre Pinson[♯,‡]

[†] Technical University of Denmark, Department of Wind and Energy Systems

[♯] Imperial College London, Dyson School of Design Engineering

[‡] Technical University of Denmark, Department of Technology, Management and Economics

Abstract

We consider a new framework where a continuous, though bounded, random variable has unobserved bounds that vary over time. In the context of univariate time series, we look at the bounds as parameters of the distribution of the bounded random variable. We introduce an extended log-likelihood estimation and design algorithms to track the bound through online maximum likelihood estimation. Since the resulting optimization problem is not convex, we make use of recent theoretical results on Normalized Gradient Descent (NGD) for quasiconvex optimization, to eventually derive an Online Normalized Gradient Descent algorithm. We illustrate and discuss the workings of our approach based on both simulation studies and a real-world wind power forecasting problem.

Keywords: Generalized logit-normal distribution; Normalized Gradient Descent; Online quasiconvex optimization; Inventory problem; Wind power probabilistic forecasting.

[*]The authors gratefully acknowledge Ørsted for providing the data for the Anholt offshore wind farm. The research leading to this work was carried out as a part of the Smart4RES project (European Union's Horizon 2020, No. 864337). The sole responsibility of this publication lies with the authors. The European Union is not responsible for any use that may be made of the information contained therein.

1 Introduction

Many statistical applications involve response variables which are both continuous and bounded. This is especially the case when one has to deal with rates, percentages or proportions, for example when interested in the spread of an epidemic (Guolo and Varin, 2014), the unemployment rates in a given country (Wallis, 1987) or the proportion of time spent by animals in a certain activity (Cotgreave and Clayton, 1994). Indeed, proportional data are widely encountered within ecology-related statistical problems, see Warton and Hui (2011) among others. Similarly, when forecasting wind power generation, the response variable is also such a continuous bounded variable. Wind power generation is a stochastic process with continuous state space which is bounded from below by zero when there is no wind, and from above by the nominal capacity of the turbine (or wind farm) for high-enough wind speeds. More generally, renewable energy generation from both wind and solar energy are bounded stochastic processes, with the same lower bound (i.e., zero energy production) and different characteristics of their upper bound (since solar energy generation has a time-varying maximum depending on the time of day and time of year), see for example Pinson (2012) and Bacher et al. (2009).

These continuous bounded random variables call for probability distributions with a bounded support such as the beta distribution, truncated distributions or distributions of transformed normal variables as discussed for example in Johnson (1949). Very often the response variable is first assumed to lie in the unit interval $(0, 1)$ and is then rescaled to any interval (a, b) through the transformation $X = (b - a)\tilde{X} + a$, where $\tilde{X} \in (0, 1)$ and $X \in (a, b)$. For applications involving such response variables, these bounds (a, b) are

always assumed to be fixed to the same values over the sample or throughout the time series. While this assumption surely makes sense in some cases, we argue it can be misleading and negatively impacts inference when the bounds (a, b) actually vary over time or depending on exogenous variables whilst not being observed. In particular it is highly relevant for energy applications, such as wind power probabilistic forecasting, as in practice the upper bound b may change over time, while being unknown, for example in case of curtailment actions for which information is not available or not reliable. Another application could be the inventory problem of the retailer, see Laderman and Littauer (1953). Let X_t be the demand for a certain item at time t. Like wind power generation, X_t is double-bounded, from below by zero and from above by the stock available at time t, that is by a time-varying upper bound b_t. To prepare for demand X_{t+1}, the retailer needs to find the quantity they should order in the light of the knowledge they have of the past stocks and demands. Similarly to the problem of forecasting wind power generation, the inventory problem might then involve a double-bounded random variable, the demand for a certain item, which can be regarded as a continuous variable for large quantities being involved, and upper bounded by a bound which may vary over time whilst not being observed, for example in case of supply chain issues, information mismanagement, or just for very large retailers that could not track the evolution of the stocks for each item or could so but would rather benefit from an automatic data-driven tracking.

In both those applications, if the random variable happens to get very close to the upper bound, it might be the case that a higher upper bound would have resulted in a higher wind power generation or item demand. Therefore we do not observe the "true"

power generation nor item demand. In that sense one could arguably think of it as being related to censoring and truncation. However we make here a different assumption. While truncation assumes the value of the response variable to be never seen (or recorded) if above the upper bound, and censoring assumes one does not know the exact value but does know it lies above the upper bound, we assume here that an upper bound lower than the "true" response results in squeezing the observed value of the variable, and thus in reshaping the probability distribution of the variable.

There are at least two ways of looking at varying bounds which cannot be observed. One can think of them as latent random variables the distribution of the response variable is conditional on. The main advantage of this approach is its generality and flexibility, with the latent bounds A and B being distributed according to a well-specified probability distribution, which might depend on exogenous variables. Suppose we assume a parametric model with parameter vector θ for both the bounds and the response variable X. Because we do not have access to the realizations a and b of the bounds, the maximization of the likelihood function of the realizations x of X might involve complicated high-dimensional integration, possibly computationally infeasible, and would therefore call for algorithms of the Expectation-Maximization kind (Dempster et al., 1977). Moreover, with such a method and for forecasting applications, one needs to first compute (good enough) forecasts of the bounds in order to be able to forecast the response variable.

An alternative way of thinking of varying bounds which cannot be observed in the more specific context of time series is to consider them as scaling parameters a and b of the parametric distribution of the bounded response variable, to include them in the

parameter vector θ, and to assume the time series to be non-stationary, at least regarding a and b. This involves the use of online learning algorithms so that the parameter vector can evolve over time. We will focus in this paper on this setup with an upper varying bound b. It would be straightforward to carry the same analysis with a lower varying bound a. From now on we will refer to X as X_t, as our response variable is now indexed by time t. As for the bounded distribution of X_t, we use the generalized logit-normal distribution introduced by Mead (1965). The practical use of any family of distributions depends on the possible variation in its shape, and on the ease with which the distribution can be fitted. The generalized logit-normal distribution is very flexible thanks to three parameters: its location μ, its scale σ^2 and its shape ν. It relies on a generalization of the logit transform and comes down to the logit-normal distribution when $\nu = 1$. Because the transformed variable is normally distributed, nice properties can be derived for the original random variable X_t. In particular, the probability density function (pdf) of X_t can be expressed as a function of the standard normal density.

We aim to estimate the full parameter θ of the pdf of X_t which now includes the upper bound b through Maximum Likelihood Estimation (MLE). The first challenge we need to tackle when dealing with the bound as a parameter in a non-stationary setup is how to handle past observations which are out of the support $(0, b)$ of the bounded distribution of X_t and make the log-likelihood to be infinite. We introduce into the log-likelihood a new term which relies on the sigmoid function to take into account those observations in a "soft" finite way. We choose to call this new log-likelihood the *extended* log-likelihood. The second challenge we need to tackle is that when considering the bound as a parameter, we cannot

be in a classical convex optimization setup anymore as the negative log-likelihood appears not to be convex with respect to (w.r.t.) the bound parameter. Therefore we propose to move to the more general quasiconvex optimization setup and use recent results about local quasiconvexity and (Stochastic) Normalized Gradient Descent (Hazan et al., 2015) to design a batch algorithm out of Normalized Gradient Descent (NGD) and an online algorithm out of Stochastic Normalized Gradient Descent (SNGD). In addition to these novel quasiconvex algorithms we propose a more classical online convex algorithm which relies on a positive definite approximation of the Hessian. We present the statistical parametric model for the time series framework with a varying upper bound b in Section 2 and the corresponding MLE in Section 3. In Section 4 we perform simulations of synthetic data to run the three algorithms we introduced in Section 3. First we look at their performances when tracking the parameter vector θ over time, then at their performances when forecasting the probability distribution of the bounded variable. In Section 5 we apply these algorithms to real data in order to provide 10-min-ahead probabilistic forecasts of the wind power generation at Anholt offshore wind farm (Denmark). Finally we discuss the results, the limitations and some prospects of the methodology in Section 6.

2 Statistical model

2.1 Parametric distribution with upper bound b as a parameter

Let $\tilde{X}_t$ be a continuous bounded random variable, $\tilde{X}_t \in (0,1)$, and X_t be the corresponding variable rescaled to $(0,b)$ by applying the transformation $X_t = (b-a)\tilde{X}_t + a = b\tilde{X}_t$ where

$a = 0$. The generalized logit transform $Y_t \in \mathbb{R}$ of $X_t \in (0, b)$ is given by

$$Y_t = \gamma(X_t/b; \nu) = \log \frac{(X_t/b)^\nu}{1 - (X_t/b)^\nu}, \quad \nu > 0,$$

where ν is the shape parameter. When Y_t is distributed according to a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, the original variable X_t/b is then distributed according to a generalized logit-normal distribution $L_\nu(\mu, \sigma^2)$, see for example Frederic and Lad (2008) and Pinson (2012). By time series we also mean series of dependent observations, therefore we assume the expectation of the normal transform Y_t to be an auto-regressive process of order p, that is $\mu_t = \sum_{k=1}^{p} \lambda_k \gamma(x_{t-k}/b; \nu)$. The pdf of X_t conditional on the previous information set $\mathcal{F}_{t-1}$ (the σ-algebra generated by $X_1, \ldots, X_{t-1}$), with parameter vector $\theta = (\lambda_1, \ldots, \lambda_p, \sigma^2, \nu, b)$, is then

$$p_\theta(x_t|\mathcal{F}_{t-1}) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma^2}} \frac{\nu}{x_t(1-(x_t/b)^\nu)} \exp\left[-\frac{1}{2}\left(\frac{\gamma(x_t/b;\nu)-\mu_t}{\sigma}\right)^2\right] & \text{if } 0 < x_{t-k} < b, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

where $k = 0, \ldots, p$.

2.2 Time-dependent log-likelihood function

We wish to estimate the parameter vector θ of the pdf $p_\theta(x_t|\mathcal{F}_{t-1})$ in (1) through MLE. In the case of a stationary time series and constant parameter θ, this comes to minimizing the negative log-likelihood objective function

$$-l(\theta) = -\sum_{t=p+1}^{T} \log p_\theta(x_t|\mathcal{F}_{t-1}) \tag{2}$$

w.r.t. θ, the data sample $x_1, \ldots, x_T$ being fixed, assuming the random variables X_t are i.i.d conditionally on $\mathcal{F}_{t-1}$.

In a non-stationary setup, estimating the parameter vector θ comes to estimating a parameter vector θ_t which varies over time, that is to minimizing a time-dependent negative log-likelihood. For ease of notation and because the negative log-likelihood is to be minimized w.r.t. θ, let $p_\theta(x_j|\mathcal{F}_{j-1}) = p_j(\theta)$. The time-dependent negative log-likelihood to be minimized at time t is then

$$-l_t(\theta) = -\frac{1}{t - j_0 + 1} \sum_{j=j_0}^{t} \log p_j(\theta). \tag{3}$$

We choose to normalize the time-dependent log-likelihood by the number of observations $t - j_0 + 1$. This does not change the optimal value obtained when minimizing w.r.t. θ and leads to more consistent values of the objectives when the number of observations varies. The time-dependent negative log-likelihood in (3) is said to be computed over a moving rectangular window, as all $t - j_0 + 1$ observations are equally weighted. If we wish to give more weight to the most recent observations we can use instead an exponential forgetting factor $\alpha \in (0, 1)$. The time-dependent negative log-likelihood is now said to be computed over a moving exponential window and is

$$-l_t(\theta) = -\frac{1}{n_\alpha} \sum_{j=j_0}^{t} \alpha^{t-j} \log p_j(\theta), \tag{4}$$

where we use $n_\alpha = \frac{1}{1-\alpha}$ for normalizing the weighted negative log-likelihood.

From (1) we can see that the negative log-likelihood in (2) we wish to minimize takes the value $+\infty$ as soon as an observation x_t is greater or equal to b. This is an implicit constraint on b when estimating $\hat{\theta}$. However moving from the stationary setup to the non-stationary one we do not want b to be greater than all the observations x_t as b should be able to vary over time. Let $U_t = \{j_0, j_0 + 1, \ldots, t\}$, $C_t(\theta) = \{j \in U_t \mid x_{j-k} < b, k = 0, \ldots, p\}$ and

$\overline{C_t}(\theta) = \{j \in U_t \mid j \notin C_t(\theta)\}$ the complement of $C_t(\theta)$ in U_t. The log-likelihood takes finite values only for observations x_j such that $j \in C_t(\theta)$. Therefore we can - informally - rewrite $\sum_{j=j_0}^{t} \alpha^{t-j} \log p_j(\theta)$ as $\sum_{j \in C_t(\theta)} \alpha^{t-j} \log p_j|_b(\theta) + \sum_{j \in \overline{C_t}(\theta)} \alpha^{t-j} \log 0$, where $p_j|_b(\theta)$ is the pdf $p_j(\theta)$ restricted to its support $(0, b)$, $\alpha = 1$ in the case of a rectangular window. When estimating the parameter vector θ_t over time, we need to take into account all the observations in the past, i.e. even the observations for which the log-likelihood does not take a finite value, that is the observations x_j such that j does not belong to $C_t(\theta)$. We then propose to replace the value 0 in $\log 0$, which originally corresponds to the value of the pdf $p_j(\theta)$ outside of its support, with a sigmoid function of $b - x_j$

$$s_j(b) = \frac{1}{1 + \exp(-b + x_j)}. \tag{5}$$

The function s_j is illustrated in Figure 1. It can be seen as the probability of x_j to be

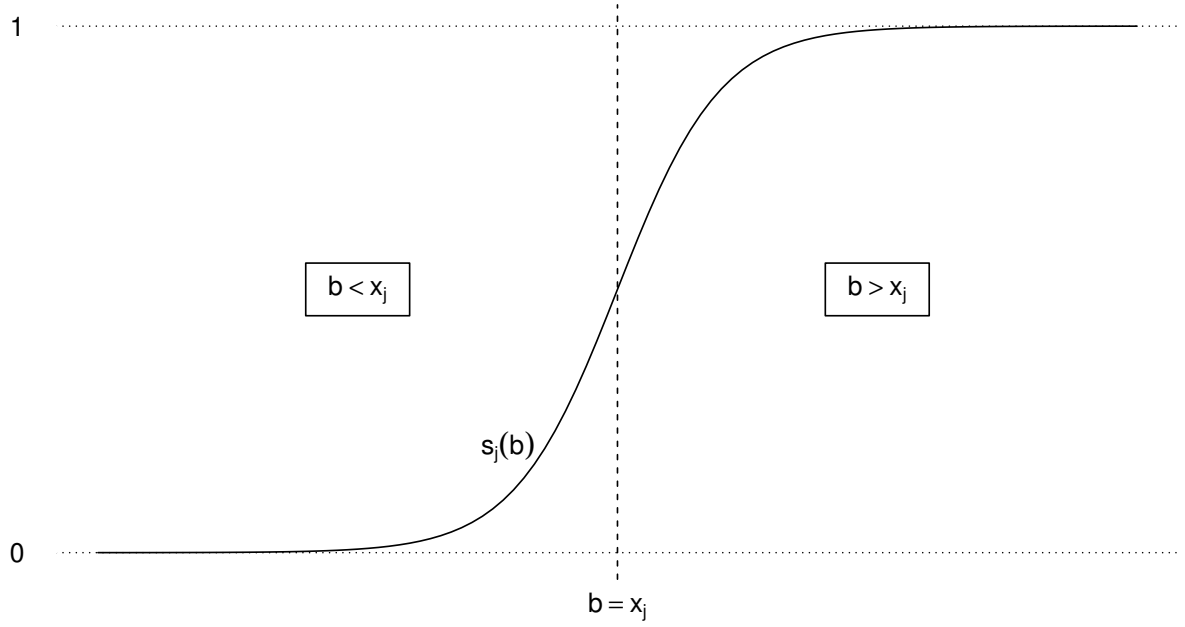


Figure 1: Sigmoid function $s_j(b)$ on the real line.

lower or equal than b: we have $s_j(b) \to 0^+$ when $b << x_j$ and $s_j(b) \to 1^-$ when $b >> x_j$. Moreover $-\log s_j(b)$ is convex and differentiable in b. Therefore the "extended" time-dependent negative log-likelihood we propose for a moving rectangular window is

$$-l_t^\infty(\theta) = -\frac{1}{t - j_0 + 1}\left[\sum_{j \in C_t(\theta)} \log p_j(\theta) + \sum_{j \in \overline{C_t}(\theta)} \log s_j(b)\right], \tag{6}$$

or equivalently for a moving exponential window

$$-l_t^\infty(\theta) = -\frac{1}{n_\alpha}\left[\sum_{j \in C_t(\theta)} \alpha^{t-j} \log p_j(\theta) + \sum_{j \in \overline{C_t}(\theta)} \alpha^{t-j} \log s_j(b)\right]. \tag{7}$$

One can note that $j \in \overline{C_t}(\theta)$ does not necessarily mean $x_j \geq b$ as it can happen because a lagged observation x_{j-k} is such that $x_{j-k} \geq b$. In such a case, that is $j \in \overline{C_t}(\theta)$ and $x_j < b$, the observation x_j will still increase the value of the total log-likelihood compared to the event $\{x_j \geq b\}$, which is also a nice feature of choosing this function s_j.

2.3 (Local-)Quasiconvexity

The first term of the extended time-dependent negative log-likelihoods $-l_t^\infty$ in (6) and (7) might not be convex in θ, in particular in b. However $-l_t^\infty$ can still have nice properties for it to be globally minimized w.r.t θ. In this section we want to recall a broader class of functions which include convex functions as a subclass: quasiconvex functions. For simplicity let assume functions are differentiable. We use $\|.\|$ to denote the Euclidean norm. From Boyd and Vandenberghe (2010), a definition of quasiconvexity is

Definition 2.1 (Quasiconvexity) *A function $f : \mathbb{R}^d \to \mathbb{R}$ is called quasiconvex (or unimodal) if its domain and all its sublevel sets*

$$S_\alpha = \{\mathbf{x} \in \mathbf{dom}\ f \mid f(\mathbf{x}) \leq \alpha\},$$

for $\alpha \in \mathbb{R}$, are convex.

As an illustrative example, Figure 2 shows the negative probability density function of a normal variable which is a quasiconvex function but not a convex function.

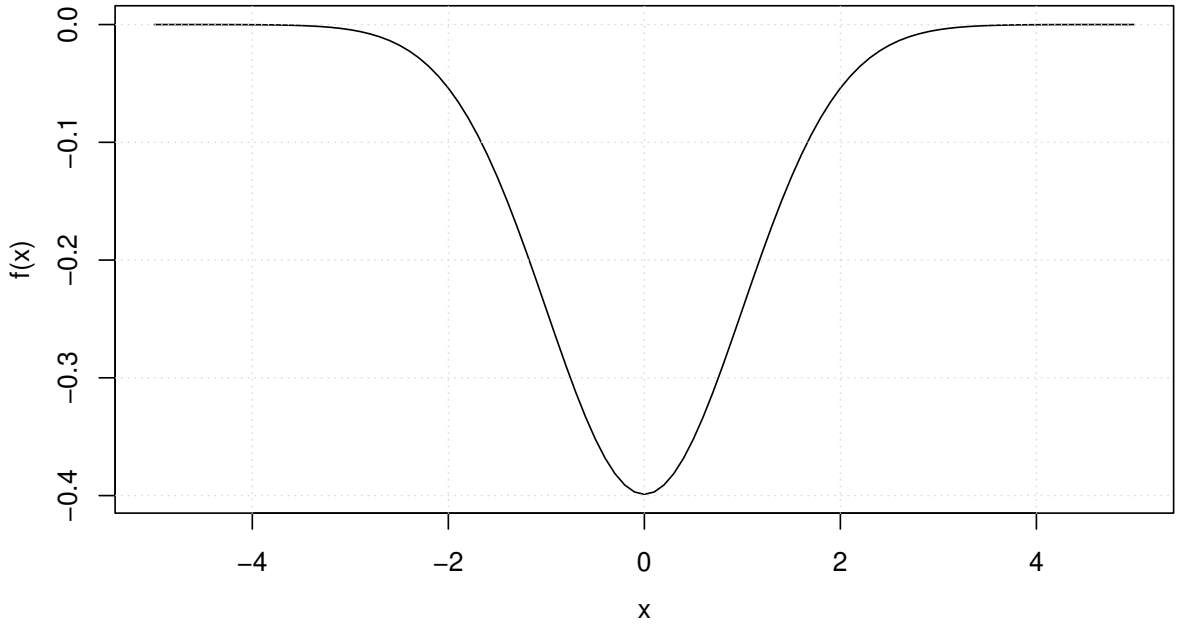


Figure 2: A quasiconvex differentiable function on $\mathbb{R}$, the negative density of a normal variable, with plateau areas when going away from the global minimum.

Quasiconvexity is a considerable generalization of convexity. Still, many of the properties of convex functions hold or have analogs for quasiconvex functions. Following is another definition of quasiconvexity which is equivalent to definition 2.1 and an analog to the first-order conditions which hold for convex functions:

Definition 2.2 (Quasiconvexity) *We say that $f : \mathbb{R}^d \to \mathbb{R}$ is quasiconvex if and only if* **dom** *f is convex and for all $\mathbf{x}, \mathbf{y} \in$* **dom** *$f$*

$$f(\mathbf{y}) \leq f(\mathbf{x}) \implies \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) \leq 0.$$

We further say that f is strictly-quasiconvex, if it is quasiconvex and its gradients vanish only at the global minima, i.e. $\forall\ \mathbf{y} : f(\mathbf{y}) > \min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \implies \|\nabla f(\mathbf{y})\| > 0$.

However, quasiconvexity broadens but does not fully capture the notion of unimodality in several dimensions. This is the argument of Hazan et al. (2015) who introduce *local-quasiconvexity*, a property that extends quasiconvexity and captures unimodal functions which are not quasiconvex. Let $\mathbb{B}_d(\mathbf{x}, r)$ denote the d dimensional Euclidean ball of radius r centered around $\mathbf{x}$, and $\mathbb{B}_d := \mathbb{B}_d(0, 1)$. The definition of local-quasiconvexity as introduced by Hazan et al. (2015) is the following:

Definition 2.3 (Local-quasiconvexity) *Let $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$, $\kappa, \epsilon > 0$.*

We say that $f : \mathbb{R}^d \mapsto \mathbb{R}$ is $(\epsilon, \kappa, \mathbf{z})$-Strictly-Locally-QuasiConvex (SLQC) in $\mathbf{x}$, if at least one of the following applies:

1. $f(\mathbf{x}) - f(\mathbf{z}) \leq \epsilon$.
2. $\|\nabla f(\mathbf{x})\| > 0$, *and for every* $\mathbf{y} \in \mathbb{B}_d(\mathbf{z}, \epsilon/\kappa)$ *it holds that* $\nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x}) \leq 0$.

When considering the Generalized Linear Models (GLM) regression as a fitting problem in which we want to minimize the error function

$$\widehat{\mathrm{err}}_m(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \phi\langle\mathbf{w}, \mathbf{x}_i\rangle)^2, \tag{8}$$

where $(\mathbf{x}_i, y_i)_{i=1,\dots,m} \in \mathbb{B}_d \times [0, 1]$ and $\phi : \mathbb{R} \mapsto \mathbb{R}$ is an activation function, Hazan et al. (2015) show that if ϕ is the sigmoid function and if we are guaranteed to have $\mathbf{w}^* \in \mathbb{R}^d$ such that $y_i = \phi\langle\mathbf{w}^*, \mathbf{x}_i\rangle$, $\forall i = 1, \dots, m$, then the error function in (8) is not generally quasiconvex but is indeed SLQC. This setup is said to be the idealized GLM setup. In

the more common noisy GLM setup (McCullagh and Nelder, 1989), where we assume now $(\mathbf{x}_i, y_i)_{i=1,\dots,m}$ are i.i.d. samples from an unknown distribution $\mathcal{D}$ and there exists a predictor $\mathbf{w}^*$ such that $\mathbb{E}[y|\mathbf{x}] = \phi\langle \mathbf{w}^*, \mathbf{x}\rangle$, $\mathbf{w}^*$ can be shown to be a global minima of the expected error

$$\mathcal{E}(\mathbf{w}) = \mathbb{E}(y - \phi\langle \mathbf{w}, \mathbf{x}\rangle)^2. \tag{9}$$

Given m samples from $\mathcal{D}$, their empirical error $\widehat{\mathrm{err}}_m(\mathbf{w})$ is defined as in (8) and Hazan et al. (2015) show that it is also SLQC, with high probability.

Simulations of $-l_t^\infty$ in (6) and (7) show our extended negative log-likelihood not to be convex but rather be, with high probability, a quasiconvex function with plateau areas when b is away from the optimal value b^* and steep concave cliffs in the neighborhood of b^*. Therefore it seems reasonable to assume $-l_t^\infty$ to be quasiconvex in θ, or at least locally-quasiconvex.

3 Time-dependent maximum likelihood estimation

3.1 Normalized Gradient Descent

Let f be the quasiconvex objective function we wish to minimize w.r.t parameter $\mathbf{x} \in \mathbb{R}^d$. It is well known that quasiconvex problems can be solved through a series of convex feasibility problems (Boyd and Vandenberghe, 2010). However solving such feasibility problems can be very costly and involves finding a family of convex functions $\phi_t : \mathbb{R}^d \to \mathbb{R}$, $t \in \mathbb{R}$, that satisfy

$$f(\mathbf{x}) \leq t \iff \phi_t(\mathbf{x}) \leq 0,$$

and $\phi_s(\mathbf{x}) \leq \phi_t(\mathbf{x})$ whenever $s \geq t$, which is far from straightforward in our case, i.e. with $f(\mathbf{x}) := -l_t^{\infty}(\theta)$. In the batch setup, a pioneering paper by Nesterov (1984) was the first to propose an efficient algorithm, the Normalized Gradient Descent (NGD), and to prove that this algorithm converges to an ϵ-optimal solution within $O(1/\epsilon^2)$ iterations given a differentiable quasiconvex objective function. Gradient descent with fixed step sizes is known to perform poorly when the gradients are too small in a plateau area of the function or explode in cliff areas. Among the deep learning community, there have been several attempts to tackle plateaus and cliffs. However those works do not provide a theoretical analysis showing better convergence guarantees than NGD.

NGD is presented in Algorithm 1. It is similar to Gradient Descent, except one normalizes the gradient. It is intuitively clear that to achieve robustness to plateaus (with vanishing gradients) and cliffs (with exploding gradients), one must ignore the size of the gradient. It is more surprising that the information in the direction of the gradient is enough to guarantee convergence. Having introduced SLQC functions, Hazan et al. (2015) prove that NGD also finds an ϵ-optimal minimum for such functions in $O(1/\epsilon^2)$ iterations. They even show faster convergence rates for quasiconvex objective functions which are

Algorithm 1 Normalized Gradient Descent (NGD)

Input: #Iterations $I, \mathbf{x_1} \in \mathbb{R}^d$, learning rate η

for $i = 1, \ldots, I$ **do**

Update: $\mathbf{x}_{i+1} = \mathbf{x}_i - \eta \hat{g}_i$ where $g_i = \nabla f(\mathbf{x}_i)$, $\hat{g}_i = \frac{g_i}{\|g_i\|}$

end for

Return: $\overline{\mathbf{x}}_I = \arg\min_{\mathbf{x}_1, \ldots, \mathbf{x}_I} f(\mathbf{x}_i)$

locally-smooth.

The adaptation of Algorithm 1 to our setup is straightforward taking $f(\mathbf{x}) := -l_t^\infty(\theta)$ and $I := t$. However one can note that when minimizing $-l_t^\infty$ w.r.t $\theta = (\Lambda, \sigma^2, \nu, b)$ we shall recover positive estimates of the scale parameter σ^2 and the shape parameter ν. This is constrained optimization which can be easily overcome by a change of variable such as replacing σ^2 with $\omega = \log \sigma^2$ and ν with $\tau = \log \nu$.

3.2 Recursive maximum likelihood estimation

In the above, we presented the NGD algorithm in order to find a global minimum to $-l_t^\infty$ w.r.t θ in a batch setting. This implies to run NGD every time we want to update the parameters θ to account for new observations which is computationally prohibitive. In order to move from the batch setting of the NGD algorithm to the online learning setting we first consider a (classic) recursive MLE procedure to recover time-dependent estimates $\hat{\theta}_t$ through the time-dependent negative log-likelihood $-l_t^\infty$. It is worth noting that this recursive procedure relies on a Newton step whereas our function $-l_t^\infty$ is with high probability not convex. However it is more a quasi-Newton approach as it approximates the Hessian with a positive definite matrix thanks to first-order information, i.e. gradients. Because we are in the online learning setting we can hope for this approximate to be a good enough very local approximation. Before introducing the recursive MLE procedure let finally note that the algorithm we derive from the latter is related to a quasi-Newton algorithm from online convex optimization (OCO), namely Online Newton Step (ONS), see for example Hazan (2022).

Consider the extended time-dependent log-likelihood with a moving exponential window in (7) and recall that $n_\alpha = \frac{1}{1-\alpha}$. We can rewrite (7) as

$$-l_t^\infty(\theta) = \begin{cases} -\alpha l_{t-1}^\infty(\theta) - (1-\alpha)\log p_t(\theta) & \text{if } t \in C_t(\theta), \\ -\alpha l_{t-1}^\infty(\theta) - (1-\alpha)\log s_t(b) & \text{if } t \in \overline{C_t}(\theta). \end{cases} \tag{10}$$

Let now $\hat{\theta}_t$ be the estimate of the parameter vector at time t. The recursive MLE procedure relies on a Newton step for obtaining the estimate $\hat{\theta}_t$ as a function of the previous estimate $\hat{\theta}_{t-1}$, see for example Madsen (2007) and Pinson and Madsen (2012). Applying one Newton step at time t we have

$$\hat{\theta}_t = \hat{\theta}_{t-1} - \frac{\nabla_\theta l_t^\infty(\hat{\theta}_{t-1})}{\nabla_\theta^2 l_t^\infty(\hat{\theta}_{t-1})}. \tag{11}$$

Let $\mathbf{h}_t = \nabla_\theta \log p_t(\hat{\theta}_{t-1})$ if $t \in C_t(\hat{\theta}_{t-1})$, $\mathbf{h}_t = \nabla_\theta \log s_t(\hat{b}_{t-1})$ if $t \in \overline{C_t}(\hat{\theta}_{t-1})$ and $\hat{\mathbf{R}}_t = -\nabla_\theta^2 l_t^\infty(\hat{\theta}_t)$. The recursive estimation relies on a few additional assumptions which are classic in the online learning framework. First assuming $\hat{\theta}_{t-1}$ minimizes $-l_{t-1}^\infty(\theta)$, from (10) we get

$$\nabla_\theta l_t^\infty(\hat{\theta}_{t-1}) = (1-\alpha)\mathbf{h}_t. \tag{12}$$

Then assuming p_t and s_t are (almost) linear in Γ in the neighborhood of $\hat{\Gamma}_{t-1}$, we get the approximation $\nabla_\theta^2 \log p_t(\hat{\theta}_{t-1}) = -\mathbf{h}_t\mathbf{h}_t^\top$ if $t \in C_t(\hat{\theta}_{t-1})$ or $\nabla_\theta^2 \log s_t(\hat{b}_{t-1}) = -\mathbf{h}_t\mathbf{h}_t^\top$ if $t \in \overline{C_t}(\hat{\theta}_{t-1})$. These approximations, which can be made because $\log p_t$ and $\log s_t$ are logarithms, are the key for ensuring that the approximate $\hat{\mathbf{R}}_t$ of the Hessian matrix is always positive definite. Finally we assume that the objective criterion $-l_t^\infty$ is smooth in the vicinity of $\hat{\theta}_t$, and the adaptation step small enough so that $\hat{\mathbf{R}}_t = -\nabla_\theta^2 l_t^\infty(\hat{\theta}_t) \simeq -\nabla_\theta^2 l_t^\infty(\hat{\theta}_{t-1})$. This is a classic assumption for deriving recursive estimation methods for stochastic systems (Ljung

and Södeström, 1983). Our two-step recursive scheme at time t is then

$$\hat{\mathbf{R}}_t = \alpha \hat{\mathbf{R}}_{t-1} + (1-\alpha)\mathbf{h}_t \mathbf{h}_t^\top,$$

$$\hat{\theta}_t = \hat{\theta}_{t-1} + (1-\alpha)\hat{\mathbf{R}}_t^{-1}\mathbf{h}_t.$$

An algorithm based upon such a recursive scheme might face computational issues as it requires inverting a matrix, the information matrix $\hat{\mathbf{R}}_t$, at each iteration. This can be prevented by working directly with the matrix inverse, the covariance matrix $\hat{\mathbf{P}}_t$, which can be computed by using the matrix inversion rule. The detailed computations are available in the supplementary material and the resulting algorithm is described in Algorithm 2.

Algorithm 2 Recursive Maximum Likelihood Estimation (rMLE)

Input: $T, \theta_p \in \mathbb{R}^{p+3}$, forgetting factor $\alpha \in (0,1)$, $\hat{\mathbf{P}}_p = 10^6 \mathbf{I}_{p+3}$

for $t = p+1, \ldots, T$ **do**

Set $\mathbf{h}_t = \nabla_\theta \log p_t(\hat{\theta}_{t-1})$ if $t \in C_t(\hat{\theta}_{t-1})$ or set $\mathbf{h}_t = \nabla_\theta \log s_t(\hat{\theta}_{t-1})$ if $t \in \overline{C_t}(\hat{\theta}_{t-1})$.

Update:

$$\hat{\mathbf{P}}_t = \frac{1}{\alpha}\left[\mathbf{I}_{p+3} - \frac{\hat{\mathbf{P}}_{t-1}\mathbf{h}_t\mathbf{h}_t^\top}{\frac{\alpha}{1-\alpha} + \mathbf{h}_t^\top \hat{\mathbf{P}}_{t-1}\mathbf{h}_t}\right]\hat{\mathbf{P}}_{t-1}$$

$$\hat{\theta}_t = \hat{\theta}_{t-1} + (1-\alpha)\hat{\mathbf{P}}_t\mathbf{h}_t$$

end for

3.3 Online Normalized Gradient Descent

The recursive MLE algorithm described in the former section is an OCO algorithm. While it can be used for solving our quasiconvex optimization problem in an online learning setting

it is likely that it would do it in a suboptimal way. Therefore we propose to alternatively use Online Normalized Gradient Descent (ONGD) as an Online QuasiConvex Optimization (OQCO) algorithm. From the observation that online learning and stochastic optimization are closely related and interchangeable (see for example Cesa-Bianchi et al. (2004) and Duchi et al. (2011)), we use the Stochastic Normalized Gradient Descent (SNGD) introduced by Hazan et al. (2015) to deriving the corresponding Online Normalized Gradient Descent for online learning.

Recall the definition 2.3 of local-quasiconvexity in section 2.3. Taking advantage of the SLQC assumption, Hazan et al. (2015) present SNGD, which is similar to Stochastic Gradient Descent (SGD) except they normalize the gradients, and prove the convergence of SNGD within $O(1/\epsilon^2)$ iterations to an ϵ-optimal minimum. This positive result requires that at each iteration of SNGD the gradient should be estimated using a minibatch of minimal size m. Indeed the authors provide a negative result showing that if the minibatch size is too small then the algorithm might diverge. This is where SNGD differs again from SGD as in the latter and for the case of convex functions even a minibatch of size 1 is enough for guaranteed convergence. The general ONGD derived from SNGD is presented in Algorithm 3. To the best of our knowledge, this is the first time SNGD is used in an online learning fashion for OQCO.

Following the framework introduced by Cesa-Bianchi and Lugosi (2006), ONGD is defined in terms of a repeated game played between the online player and the "environment" generating the outcome sequence. At each iteration t, we play a parameter vector $\mathbf{x}_t$. After we have committed to this choice, a (SLQC) cost function f_t is revealed and the cost we

Algorithm 3 Online Normalized Gradient Descent (ONGD)

Input: convex set $\mathcal{K}, T, \mathbf{x}_m \in \mathcal{K}$, step size η, minibatch size m

for $t = m, \ldots, T$ **do**

Play $\mathbf{x}_t$ and observe cost $f_t(\mathbf{x}_t) = \dfrac{1}{m} \displaystyle\sum_{j=t-m+1}^{t} f_j(\mathbf{x}_t)$.

Update and project:

$$\mathbf{y}_{t+1} = \mathbf{x}_t - \eta \hat{g}_t \text{ where } g_t = \nabla f_t(\mathbf{x}_t),\ \hat{g}_t = \frac{g_t}{\|g_t\|}$$

$$\mathbf{x}_{t+1} = \Pi_{\mathcal{K}}(\mathbf{y}_{t+1})$$

end for

incur is therefore $f_t(\mathbf{x}_t)$, the value of the cost function for the choice $\mathbf{x}_t$. Similarly to Online Gradient Descent (OGD), which is based on standard gradient descent from offline optimization and was introduced in its online form by Zinkevich (2003), we have included in ONGD a projection step $\Pi_{\mathcal{K}}(.)$. Indeed in each iteration, the algorithm takes a step from the previous point in the direction of the normalized gradient of the previous cost. This step may result in a point outside of the underlying convex set $\mathcal{K}$. In such cases the algorithm therefore projects the point back to the convex set $\mathcal{K}$, i.e. finds its closest point in $\mathcal{K}$.

The adaptation of Algorithm 3 to our setup is straightforward taking

$$f_j(\mathbf{x}_t) := \begin{cases} -\log p_j(\hat{\theta}_t) & \text{if } j \in C_t(\hat{\theta}_t), \\ -\log s_j(\hat{b}_t) & \text{if } j \in \overline{C_t}(\hat{\theta}_t). \end{cases}$$

Note that working with $\theta = (\Lambda, \omega, \tau, b)$, $\theta \in \mathbb{R}^{p+3}$, as in sections 3.1 and 3.2, we do not need the projection step in our setup. Finally we want to emphasize that moving from

Algorithms 1 and 2 to Algorithm 3 we do not need a moving window anymore. Instead we use a constant step size η, analog to the learning rate in NGD and SNGD, and a minibatch size m. The technical derivations required for all the algorithms of section 3 are available in the supplementary material.

4 Simulation study

4.1 Tracking the parameter vector

In order to test the convergence of the algorithms proposed in section 3 we perform an empirical study on synthetic data which fit our setup. We run 100 Monte Carlo (MC) simulations with $T = 12000$, $\lambda = 0.9$, $\sigma^2 = 1$, $\nu = 1.5$ and b varying in a sinusoidal way. For all algorithms and simulations the lag $p = 1$ of the auto-regressive process is assumed to be known and the initial values of the parameter vector $\theta = (\lambda, \sigma^2, \nu, b)$ are always set to $(0, 1, 1, 1)$. For NGD each batch algorithm is run every 500 data points after a burn-in period of 1,000 points. A new estimate of the parameter vector is therefore available every 500 time steps. For rMLE in order to fulfill the necessary condition to (12) in section 3.2, the recursive algorithm is run after a warm-up period of 1,000 data points: first a batch algorithm, i.e. NGD, is run on the first 1,000 data points with $\theta_0 = (0, 1, 1, 1)$; then the resulting estimates are used as initial values for rMLE. From the 1,000th time step on, a new estimate of the parameter vector is then available every time step. For ONGD the algorithm starts as soon as there are enough data points for a minibatch of size m. Afterwards a new estimate of the parameter vector is available every time step. The values

of the hyperparameters for each algorithm are summarized in Table 1. They were decided upon looking to the first MC simulation.

Table 1: Hyperparameter values for each algorithm: forgetting factor α, number of iterations I, learning rate/step size η and minibatch size m.

	α	I	η	m
NGD (Algorithm 1)	0.990	10000	0.003	-
rMLE (Algorithm 2)	0.975	-	-	-
ONGD (Algorithm 3)	-	-	0.001	100

The tracking of the parameters depending on the algorithm is presented in Figure 3. First note that while all three algorithms succeed in tracking the parameter vector, there are clear differences in their performances. ONGD is the algorithm which manages to closer track the parameters with the less variance. However for some parameters it needs a few time steps before converging to the true value at the time: about 1,000 for the auto-regressive parameter λ, 2,000 for the shape parameter ν and 500 for the upper bound b. The scale parameter σ^2 is already at its true value when the algorithms get started but serves as a reference that there is no divergence from the optimal value. As a matter of fact and as stated by Hazan et al. (2015) we have observed the parameter σ^2 to diverge when the minibatch size m was too small: for $m = 1$ the algorithm was surely diverging while it was converging for $m \geq 10$. NGD converges from the first update already, that is with 1,000 data points. This was to be expected as α was set to 0.99, which means most of the weight was placed on the last 100 points of each batch. However NGD shows more variance

than ONGD and is a bit slower in following a decreasing b. The latter drawback would likely be improved by increasing the frequence in the updates. Finally rMLE performs well in tracking a decreasing b, but fails to properly track an increasing bound. Moreover this comes with a cost in variance which is very high.

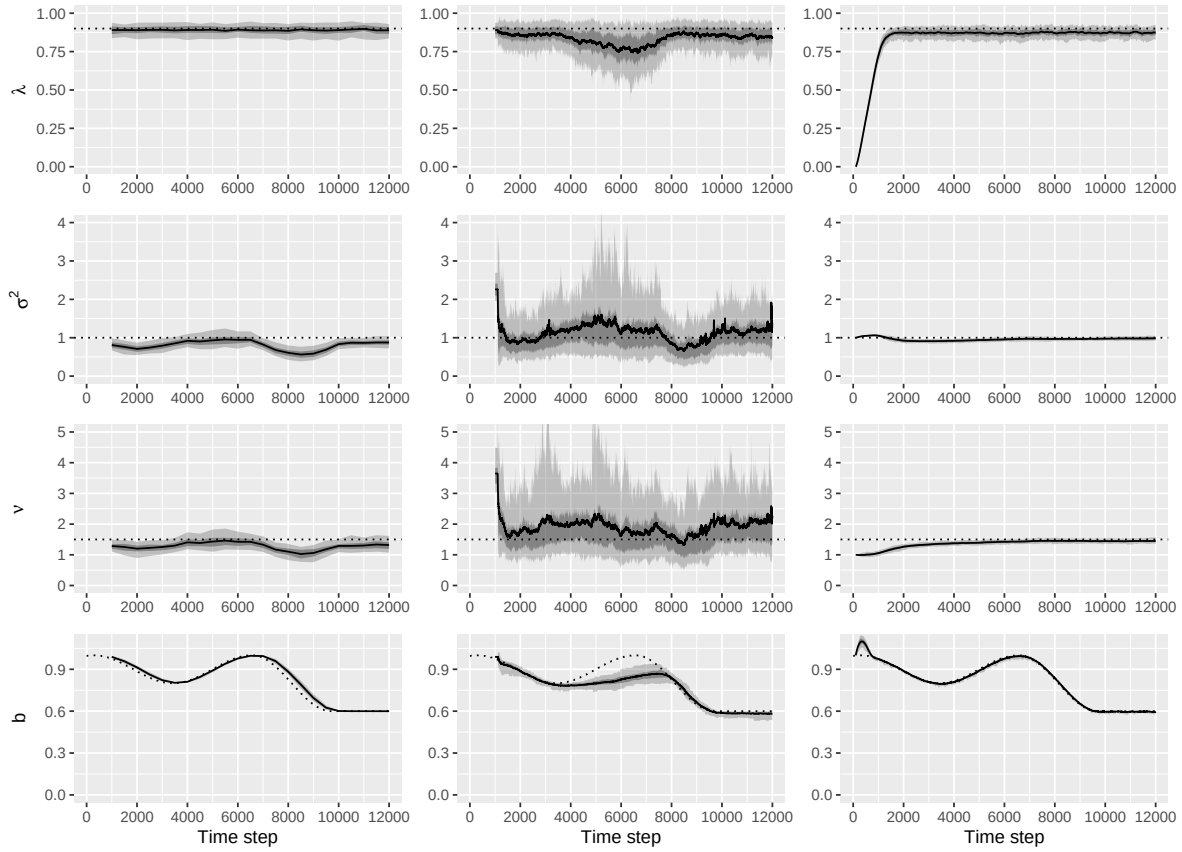


Figure 3: Confidence intervals of the tracked parameters for NGD (left), rMLE (center) and ONGD (right) with coverage probabilities 0.9 and 0.5, along with the average estimates (solid lines) and the true parameters (dotted lines).

4.2 Forecasting the distribution

Because many applications which might benefit from this framework involve forecasting, we are now interested in the performance of the algorithms when forecasting at time t the distribution of the bounded variable X_{t+1}. To be able to track the bound parameter over time, we have introduced in section 2 the extended time-dependent negative log-likelihood $-l_t^\infty$ and allowed b_t to vary on $\mathbb{R}$, which makes sense from an inference point of view. Because we are working with series of dependent observations, when moving to forecasting the distribution of X_{t+1} we need the current value b_t to be greater than all the p former observed values of $X_t, \ldots, X_{t-p+1}$ for the expected value of X_{t+1} to exist. Therefore we introduce a projection step as described in section 3 for ONGD: we project $\hat{\theta}_t$ on the convex set $\mathcal{K} = \mathbb{R}^{p+2} \times (\max(x_t, \ldots, x_{t-p+1}), +\infty)$ and we get the projected parameter $\tilde{\theta}_t = \Pi_{\mathcal{K}}(\hat{\theta}_t) = (\hat{\Lambda}_t, \hat{\omega}_t, \hat{\tau}_t, \tilde{b}_t)$ where $\tilde{b}_t = \max(x_t, \ldots, x_{t-p+1}) + \delta$ if $\max(x_t, \ldots, x_{t-p+1}) > \hat{b}_t$, $\tilde{b}_t = \hat{b}_t$ if $\max(x_t, \ldots, x_{t-p+1}) < \hat{b}_t$. Note that we need to introduce a small $\delta > 0$ as we project $\hat{b}_t$ on an open convex set. When looking at the observation x_t as a coarsened version of X_t, δ can be seen as a coarsening parameter. This coarsened data framework has been formalized by Heitjan and Rubin (1991) and Heitjan (1993). We use $\delta = 0.001$.

The predictive probability distributions obtained from Algorithms 1, 2, 3 are evaluated and compared to classic benchmarks such as climatology and probabilistic persistence. The climatology is based on all past data available at the time of forecasting and the probabilistic persistence is the most recent observed value at the time of forecasting which we dress with the most recent observed values of the persistence error. We also provide the predictive distributions obtained from the alike algorithm to Algorithm 2 when the

bound is assumed to be fixed and equal to 1 (Pierrot and Pinson, 2021). From now on we will refer to this algorithm as rMLE.1 and to Algorithm 2 as rMLE.b. We evaluate the predictive distributions through calibration and proper scoring rules, as probabilistic predictions should be calibrated and as informative/sharp as possible, see for example Gneiting et al. (2007), Gneiting and Raftery (2007) and Gneiting and Katzfuss (2014). To empirically check on probabilistic, respectively marginal calibration, we provide Probability Integral Transform (PIT) histograms, respectively marginal calibration plots. Scoring rules are attractive measures of predictive performance as they evaluate calibration and sharpness simultaneously. We use the Continuous Ranked Probability Score (CRPS) which is a proper scoring rule relative to the class $\mathcal{P}$ of the Borel probability measures on $\mathbb{R}$ and a strictly proper scoring rule relative to the subclass $\mathcal{P}_1$ of the Borel probability measures that have finite first moment (Gneiting and Raftery, 2007). Proper scoring rules are often used in negative orientation, e.g. the lower the better. As we work with predictive densities, one could think of using the logarithmic score which is strictly proper relative to all measures that are absolutely continuous, at least to compare the predictive densities provided by Algorithms 1, 2, 3 and the rMLE.1 benchmark algorithm. However in our framework it can always happen that x_{t+1} falls out of the support of the predictive density $\hat{p}_{t+1}$ we have produced at time t, when the current estimate of the upper bound is too low. In such a case the logarithmic score is equal to $-\log \hat{p}_{t+1}(x_{t+1}) = -\log 0 = +\infty$, which is not suitable. In contrast the CRPS is defined on $\mathbb{R}$ as

$$\mathrm{CRPS}(F, x) = \int_{-\infty}^{\infty} \left(F(y) - \mathbb{1}_{y \geq x}\right)^2 \mathrm{d}y, \tag{13}$$

where F is the cumulative distribution function (cdf) of the probabilistic forecast and y is

the evaluation point. In our setup $F := \hat{F}_{t+1}$ and $x := x_{t+1}$. If x_{t+1} happens to be greater than the upper bound b_t we get

$$\begin{aligned}
\mathrm{CRPS}(\hat{F}_{t+1}, x_{t+1}) &= \int_{-\infty}^{b_t} \left(\hat{F}_{t+1}(y) - \mathbb{1}_{y \geq x_{t+1}}\right)^2 \mathrm{d}y + \int_{b_t}^{\infty} \left(1 - \mathbb{1}_{y \geq x_{t+1}}\right)^2 \mathrm{d}y, \\
&= \int_{-\infty}^{b_t} \hat{F}_{t+1}(y)^2 \mathrm{d}y + \int_{b_t}^{x_{t+1}} 1 \ \mathrm{d}y + \int_{x_{t+1}}^{\infty} 0 \ \mathrm{d}y, \\
&= \int_{-\infty}^{b_t} \hat{F}_{t+1}(y)^2 \mathrm{d}y + x_{t+1} - b_t.
\end{aligned}$$

Therefore the CRPS is increased by an observation falling out of the support of the predictive distribution but to a higher finite value contrary to the logarithmic score which becomes infinite. Moreover the CRPS allows us to compare discrete and continuous distributions, that is to compare our density-based algorithms to climatology and probabilistic persistence, as if the predictive distribution takes the form of a sample of size N, then the right side of (13) can be evaluated in $\mathcal{O}(N \log N)$ operations (Hersbach, 2000).

We start computing predictive distributions after having seen 2,000 observations. Recall that the hyperparameters were chosen in section 4.1 upon looking at only the first MC simulation for each algorithm, but the whole simulated time series, that is looking at the data we are now computing probabilistic forecasts for. This may be optimistic, even if we only looked at the data from the first MC simulation. In order for our algorithms to not be more optimistic than the benchmarks, we then use the hyperparameters for probabilistic persistence and rMLE.1 that gave the best CRPS on the first MC simulation. We also provide the results for the ideal forecaster, which is the true distribution of the synthetic data, that is the GLN distribution with the true constant values Λ, σ^2 and ν and the true values of the upper bound $b_{t+1}, \ldots, b_{t-p+1}$. The CRPS.s are available in Table 2, PIT

histograms with 20 bins in Figure 4, and the marginal calibration plot in Figure 5. Note that they are all averages on the MC sample.

Table 2: 1-step-ahead CRPS and respective improvement over climatology and persistence. The CRPS is averaged over the MC sample, and the standard deviation is also provided.

	mean (sd)	Imp./clim.	Imp./persist.
ideal forecaster	5.78% (0.10)	-	-
climatology	15.26% (0.24)	-	-
probabilistic persistence	6.28% (0.10)	58.85%	-
rMLE.1	6.04% (0.09)	60.40%	3.77%
NGD (Algorithm 1)	5.87% (0.09)	61.53%	6.52%
rMLE.b (Algorithm 2)	6.04% (0.12)	60.42%	3.82%
ONGD (Algorithm 3)	5.81% (0.10)	61.94%	7.52%

The average CRPS obtained by the ideal forecaster over all simulations is 5.78%. Probabilistic calibration is reflected through a uniform histogram and marginal calibration through the proximity between the predictive and the empirical cdf.s. As expected the forecasts issued by the ideal forecaster are perfectly calibrated in terms of both probabilistic and marginal calibration. On the other hand climatology appears to be significantly not calibrated for all kinds of calibration. Probabilistic persistence performs very well on our synthetic dataset with a CRPS which is already much closer to the CRPS of the ideal forecaster. As one can see in Figure 4, forecasts from probabilistic persistence also essentially achieve probabilistic calibration. However marginal calibration is not quite satisfactory.

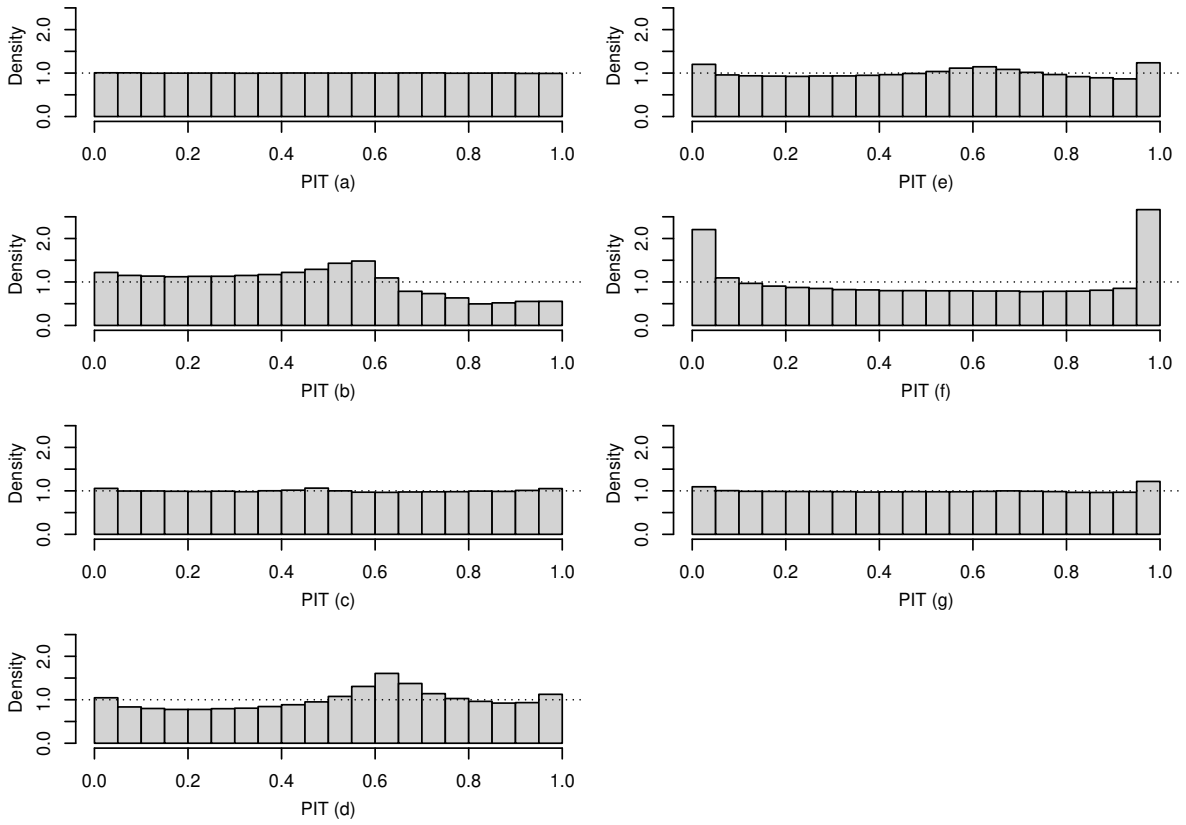


Figure 4: PIT histograms for all benchmarks (left) and Algorithms 1, 2, 3 (right): (a) ideal forecaster, (b) climatology, (c) probabilistic persistence, (d) rMLE.1, (e) NGD, (f) rMLE.b, (g) ONGD.

The last benchmark, that is the rMLE.1 algorithm, performs much better than climatology and better than probabilistic persistence when looking at the CRPS, even with a wrong assumption on the upper bound. However the rMLE.1 forecasts show more departures from probabilistic calibration than probabilistic persistence.

As for the proposed algorithms 1, 2 and 3, all of them show lower average CRPS.s than probabilistic persistence, the best performance being achieved by NGD and ONGD. Note that ONGD shows a CRPS which is very close to the CRPS of the ideal forecaster,

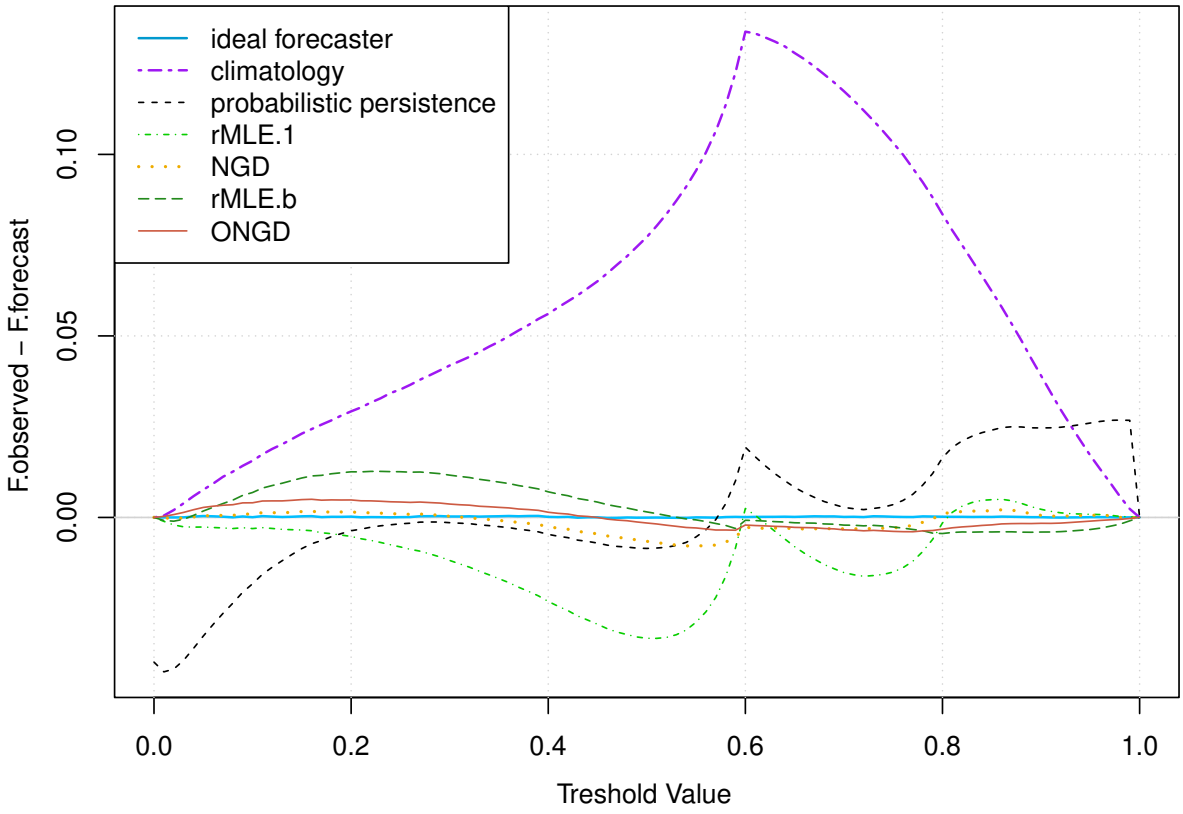


Figure 5: Marginal calibration plot for all benchmarks and Algorithms 1, 2, 3.

while rMLE.b achieves similar results as rMLE.1 in terms of CRPS. As for probabilistic calibration, ONGD is the algorithm whose histogram is the closest to uniformity, rMLE.b's being clearly not uniform. Only for marginal calibration do all the proposed algorithms show on average better calibration than probabilistic persistence and rMLE.1, especially NGD and ONGD. Overall ONGD is the algorithm which achieves the best results in terms of both sharpness and calibration. In particular it achieves better sharpness and marginal calibration when compared to the already very efficient probabilistic persistence.

5 Application to wind power forecasting

Accurately forecasting wind power generation is highly important for the integration of wind energy into power systems. We are interested here in very short-term forecasting, that is in lead times of a few minutes, which are not only crucial for transmission system operators to keep the system in balance but also very difficult to improve the forecasts for, especially compared to the simple but very efficient persistence.

5.1 Data description

We have historical data from a large offshore wind farm, Anholt in Denmark, from July 1, 2013 to August 31, 2014. The active power is available for 110 wind turbines at a temporal resolution of every 10 minute. We scale each time series individually according to the nominal power of the wind turbine and compute the average generation over the wind farm depending on the number of wind turbines which are available at each time step in order to handle missing values. The response random variable we wish to forecast at time t is $X_{t+1} \in (0, 1)$, the average active power generated by the wind farm at time $t + 1$. As stated in section 4.2 we choose to look at the observation x_t as a coarsened version of X_t with $\delta = 0.001$. Therefore an observation x_t is set to δ if $x_t < \delta$ and to $1 - \delta$ if $x_t > 1 - \delta$ and $x_t \in [\delta, 1 - \delta]$ whereas $X_t \in (0, 1)$.

5.2 Validation setup

We split our dataset into two datasets that we keep separate: a training/cross-validation dataset from July 1, 2013 to March 31, 2014, resulting in 39,450 observations; a test

dataset from April 1 to August 31, 2014, resulting in 22,029 observations. As in section 4.2 we compare Algorithms 1, 2, 3 to climatology, probabilistic persistence and to the rMLE.1 algorithm. We use the training set to run the online algorithms, that is the rMLE.1 algorithm and Algorithms 2 and 3, and to train NGD, i.e. Algorithm 1. For methods involving hyperparameters, that is all of them but climatology, we choose the hyperparameters upon cross-validation: we use part of the training set, from November 1, 2013 to March 31, 2014 and for each method select the hyperparameters which give the lowest CRPS on the cross-validation subset. The hyperparameter values selected for each algorithm are available in Table 3. Note that for Algorithm 1 the maximum number

Table 3: Hyperparameter values for each algorithm: order p of the AR process, forgetting factor α, number of iterations I, learning rate/step size η and minibatch size m.

	p	α	η	m
NGD (Algorithm 1)	3	0.9975	0.1	-
rMLE.b (Algorithm 2)	5	0.9982	-	-
ONGD (Algorithm 3)	4	-	0.03	1

of iterations I does not appear in Table 3 as it is not selected upon cross-validation but set to 5,000, which seems to be enough for the objective function to not be significantly decreasing anymore. The frequency in estimating a new model should also be considered as a hyperparameter, as it has an influence on the overall performance of the algorithm. However a serious drawback of Algorithm 1 is the associated computing time which is prohibitive. Even if the algorithm runs in a few seconds when $I = 5000$, to estimate a new

model every 500 data points as in Section 4 means to estimate 44 models on the cross-validation dataset for one set of hyperparameters. To increase the frequency of the updates for example to an update every 100 data points would mean five times more models for one set of hyperparameters. Therefore we tested only a few update frequencies (100, 250, 500, 750, 1000) and will present the results for 500, which was the frequency that performed the best on the cross-validation set according to this limited grid. This does not mean that a better set of hyperparameters were not to be found if we had infinite computing resources.

5.3 Assessment of the probabilistic forecasts

The algorithms are run on the test set with the hyperparameters we have selected and we evaluate the predictive distributions through calibration and the CRPS as a proper scoring rule, as we did in section 4.2. The CRPS are presented in Table 4 and the calibration plots in Figures 6 and 7. On those real data probabilistic persistence improves the CRPS of climatology by a very large percentage already. Nevertheless all algorithms but Algorithm 1 perform better than probabilistic persistence in terms of CRPS. One can note that both rMLE algorithms perform quite similarly and improve the persistence by roughly 20%, while ONGD is the one which achieves the most significant improvement compared to both the persistence and the rMLE algorithms. When looking at the calibration plots, it appears that no method is as well calibrated as in the simulation study, no matter which kind of calibration. Regarding probablistic calibration, all PIT histograms suggest departures from uniformity, in a similar way for probabilistic persistence and ONGD, and for rMLE.1 and rMLE.b. The PIT histograms of both persistence and ONGD show a

Table 4: 10-minute-ahead CRPS and respective improvements over climatology, persistence and the rMLE.1 algorithm.

	CRPS	Imp./clim.	Imp./persist.	Imp./rMLE.1
climatology	22.03%	-	-	-
probabilistic persistence	1.35%	93.87%	-	-
rMLE.1	1.08%	95.09%	19.89%	-
NGD (Algorithm 1)	1.43%	93.52%	-5.74%	-31.99%
rMLE.b (Algorithm 2)	1.06%	95.21%	21.83%	2.43%
ONGD (Algorithm 3)	**0.89**%	**95.97**%	**34.22**%	**17.89**%

**Best forecast bolded.*

too large number of very low (close to 0) and very high (close to 1) PIT values, which suggests the predictive distributions are underdispersed with too narrow prediction intervals in general. On the contrary the PIT histogram for NGD is hump shaped which indicates the predictive distributions are overdispersed with too large prediction intervals in general. The PIT histograms for the rMLE algorithms somehow show both patterns. Regarding marginal calibration rMLE.1 and rMLE.b are very close to one another. Overall the online algorithms, that is rMLE.1, rMLE.b and ONGD, show better marginal calibration than the other methods. Note that we even removed climatology from Figure 7 as the corresponding predictive cdf is way too far on average from the empirical one to be plotted on the same graph as the other methods.

The parameter vector estimates for the rMLE.1 benchmark and Algorithms 1, 2, 3 are

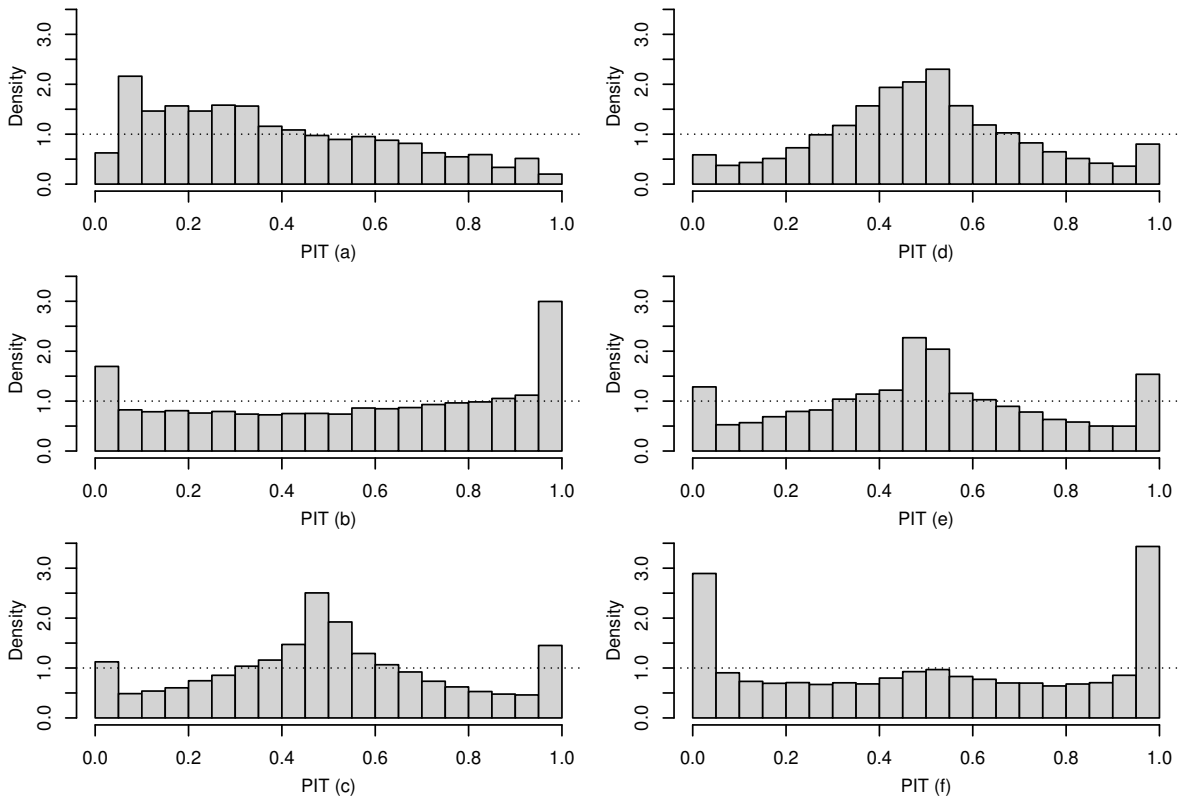


Figure 6: PIT histograms for all benchmarks (left) and Algorithms 1, 2, 3 (right): (a) climatology, (b) probabilistic persistence, (c) rMLE.1, (d) NGD, (e) rMLE.b, (f) ONGD.

plotted for some sub-sample of the test set in Figure 8. Regarding the bound parameter b we plot the projection $\tilde{b}_t$ of $\hat{b}_t$, see section 4.2, to display the bound which is actually used for prediction. One can first note that the estimates of the parameter vector Λ are consistent from one method to another, while being more noisy for Algorithms 1, 2, 3 which include a varying upper bound, especially for NGD and ONGD. The estimates from rMLE.1 and rMLE.b are in general very close to one another and show similar patterns with more noise for rMLE.b. Moreover in the latter the estimated upper bound does not vary significantly away from 1. Therefore it is hard to see what rMLE.b brings compared

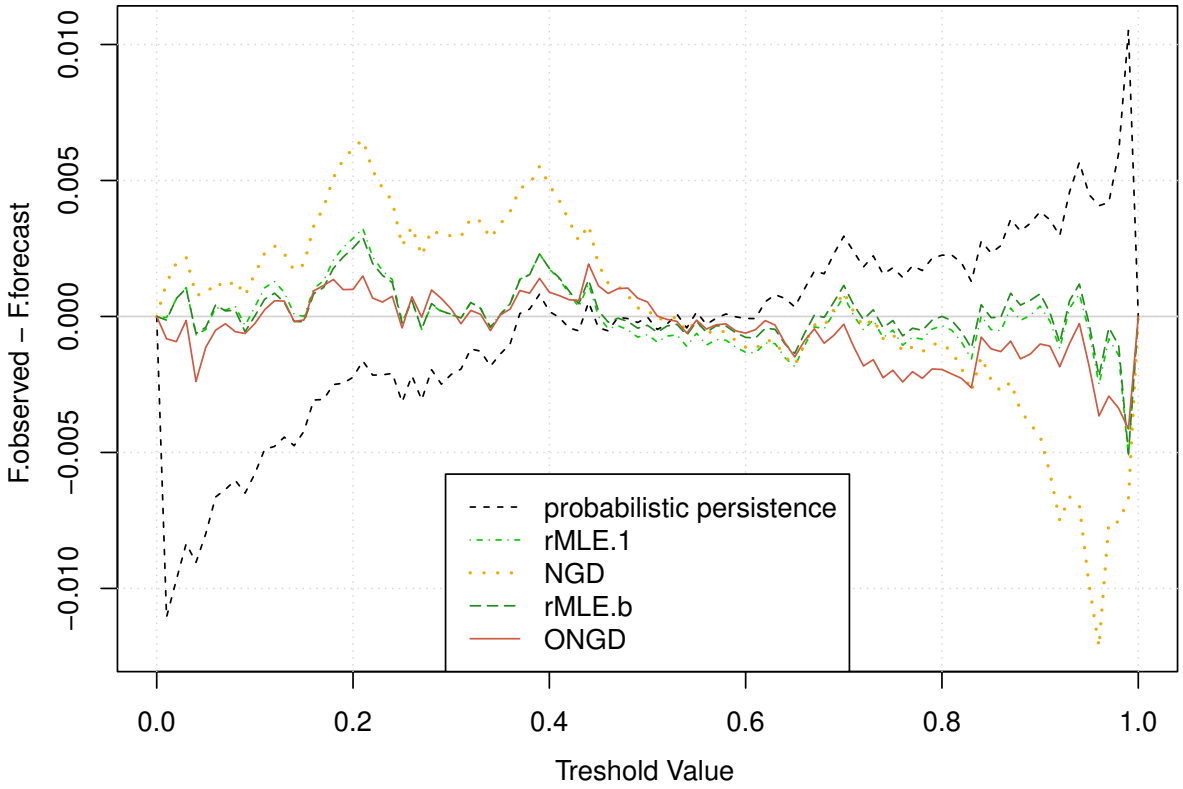


Figure 7: Marginal calibration plot for probabilistic persistence, rMLE.1 and Algorithms 1, 2, 3.

to rMLE.1 on this real dataset. As for Algorithm 1, the CRPS achieved by NGD on the test set was quite high, especially compared to the rMLE.1 benchmark. When looking at its parameter estimates, let first recall that one set of parameters is estimated by one batch model, independently of the other batch models. There is clearly instability from one batch to another, which can be a sign that the maximum number $I = 5000$ of iterations we used was not enough for Algorithm 1 to converge. Moreover the values estimated for the bound parameter are mostly above 1, which is not satisfactory. Nevertheless the associated computational time would make it very difficult to run the algorithm through even more

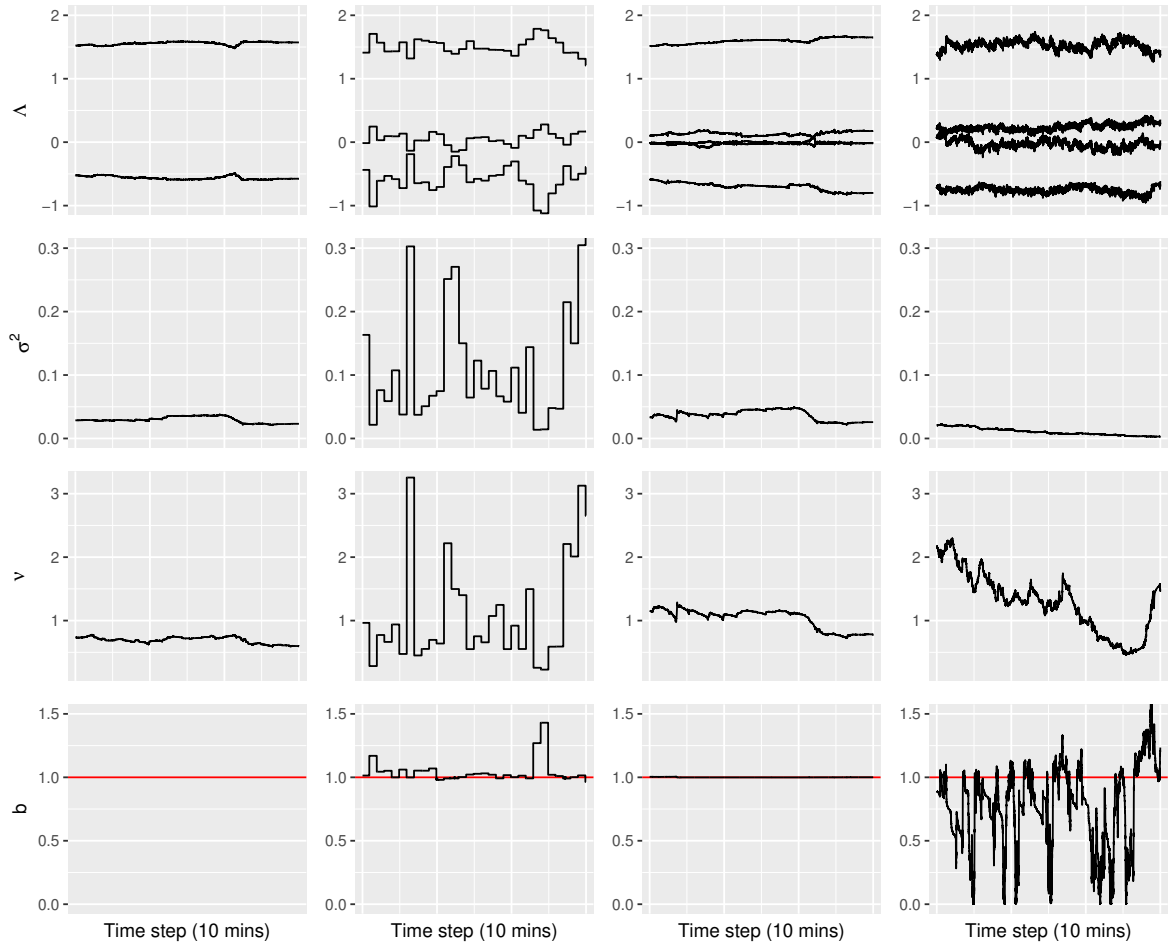


Figure 8: Estimates of Λ, σ^2, ν and projected estimate of b on a sub-sample of the test set for rMLE.1 (left), NGD (center left), rMLE.b (center right) and ONGD (right).

iterations, as already mentioned in Section 5.3. Finally ONGD is the only algorithm which captures some variations of the bound parameter below 1 and achieves a very significant improvement over probabilistic persistence (about 32% reduction of the CRPS) and a significant improvement over rMLE.1 (about 18% reduction of the CRPS). Although while choosing the hyperparameters for ONGD, we saw a significant improvement in the CRPS on the cross-validation set for a minibatch size $m = 1$ only (the values tested for m being

$m \in \{1, 5, 10, 20, 50, 100, 150\}$). By looking at the parameter estimates, we noticed that for $m = 5$ the bound estimate was also significantly varying below 1, but more slowly and closer to 1, and led to CRPS.s in the range of the other online algorithms. When looking at the power generation itself, the bound tracked by ONGD for $m = 1$ did make sense. Those results are confirmed by the test set, as the CRPS we get for it is very close to the one we got on the cross-validation set, confirming the generalization performance of the model. Therefore it seems the data call for a very aggressive choice of m so that the algorithm can track the bound. In return some noise is introduced in the other parameters, especially in the expectation parameter vector Λ.

We provide probabilistic forecasts over two different 36-hour periods of time on the test set for rMLE.1 and Algorithms 1, 2, 3. As for NGD (second from top) we observe the prediction intervals to be extremely wide on the first period of time (left). This shall correspond to a moment, that is to a batch model, when the scale and/or shape parameters were very large, confirming how problematic the instability from one batch model to another is. As for ONGD (bottom), the plots confirm how tight the prediction intervals are. However when the observation falls out of the prediction interval, which happens quite often as showed by the PIT histogram for ONGD in Figure 6, it does not fall far away, which explains the very good CRPS achieved by ONGD overall.

6 Discussion

We have introduced a new framework where we aim to track varying bounds for bounded time series as well as an extended negative log-likelihood to deal with this new framework.

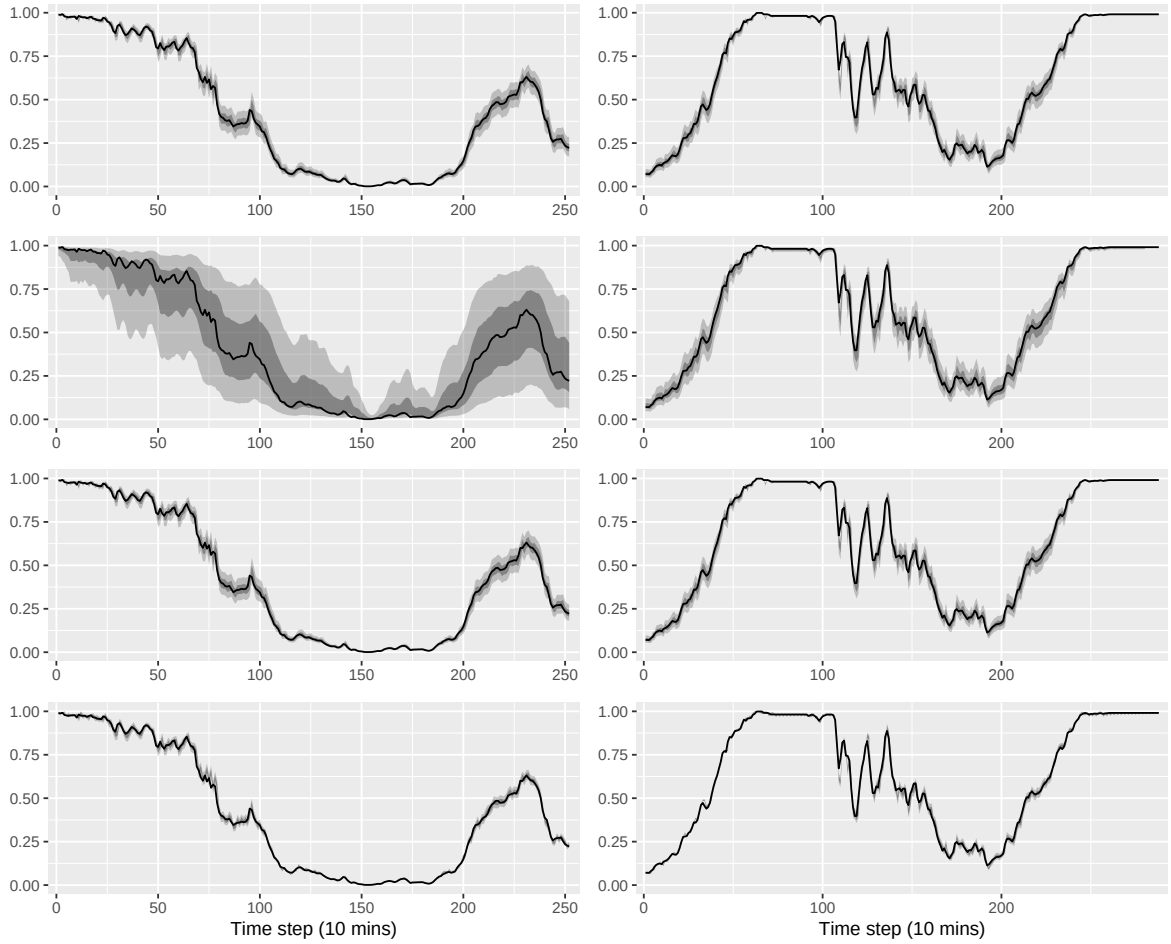


Figure 9: Probabilistic forecasts from rMLE.1 and Algorithms 1, 2, 3 (from top to bottom), on two different periods (left and right), based on prediction intervals with nominal coverage rates of 95 and 75%, along with the power measurements (solid black line).

As the objective functions now at hand are not convex anymore, we have proposed to make use of the broader quasiconvexity assumption through two algorithms, a batch algorithm and an online algorithm, which both rely on NGD. We have also proposed a more usual online algorithm out of quasi-Newton methods. On both a synthetic and a real dataset, we have run those algorithms for tracking the parameters of a time series distribution over

time, including the upper bound of the support of the distribution. Then we have presented how to use those tracked parameters for forecasting.

The first algorithm, which relies on a time-dependent negative log-likelihood, exponentially weighted through a forgetting factor, and on NGD for its optimization, is a batch algorithm which needs to be updated when new observations come in. It performed well on our (smooth) synthetic dataset but did not scale when moving to our application, that is to wind power forecasting. Extra work could be performed in order to improve it, for example by trying different kinds of initialization for the parameters when starting the optimization of a new batch model. Indeed one could take advantage of the past optimizations by initializing the algorithm with the estimates from the previous batch algorithm.

The second algorithm is an online algorithm which relies on the same time-dependent, exponentially weighted, negative log-likelihood, but makes use of classical online, local assumptions to recursively update the parameter estimates. It does not perform as well as the first algorithm on our simulated example as it struggles to track an increasing bound and comes with a high price in variance while doing so. It performed well though when tracking a decreasing bound. When moving to wind power forecasting, this second algorithm did not show a significant improvement when compared to its equivalent with a fixed bound. It could be improved by working on adaptive multiple forgetting factors, to adjust to different variation speeds in time and depending on the parameter. This could also benefit to the first algorithm, which performed very well when tracking an increasing bound but was a bit late in following a decreasing one, with some visible impact on both the scale and the shape parameters of the distribution.

The third algorithm is an online algorithm which is directly derived from SNGD and so, similarly to "ordinary" OGD, only relies on the negative log-likelihood we observe at time t for our current set of parameters. It does not longer require any kind of forgetting action and only asks for the usual step size when updating the parameter vector through the gradient at time t. A new hyperparameter which is related to a specificity of SNGD is the size m of the minibatch as SNGD is not guaranteed to converge for $m = 1$, unlike SGD. This third algorithm performed extremely well on our simulated examples, with performances in forecasting very close to the ideal forecaster. When moving to wind power forecasting, it improved the CRPS of probabilistic persistence by more than 30% on the test set. However the predictive distributions do not achieve probabilistic calibration, as the prediction intervals appear to be too narrow in general.

It is worth noting that ONGD required to set the minibatch size m to 1 on the wind power generation dataset in order to be able to track the bound. This is quite aggressive and suggests that this kind of data might call for methods which can handle big jumps in the bound values. Overall none of the proposed methods nor the benchmarks managed to achieve probabilistic calibration on those data, but looking at the CRPS and at marginal calibration it is quite clear wind power generation forecasting calls for online methods. This is also why we chose not to further investigate NGD on this use case.

SUPPLEMENTARY MATERIAL

Calculation details: Technical derivations required for Algorithms 1, 2, 3 and detailed computation of the matrix $\hat{\mathbf{P}}_t$ in Algorithm 2. (.pdf file)

R project: R project with the R scripts for the simulation study in section 4, along with the corresponding synthetic data. All outputs necessary for the study can be reproduced with the corresponding scripts and are also provided. The structure and content of the R project is described in a README file. (.zip file)

References

Bacher, P., Madsen, H., and Nielsen, H. A. (2009), "Online Short-Term Solar Power Forecasting," *Solar Energy*, 83, 1772–1783.

Boyd, S. P. and Vandenberghe, L. (2010), *Convex Optimization*, Cambridge University Press.

Cesa-Bianchi, N., Conconi, A., and Gentile, C. (2004), "On the Generalization Ability of On-Line Learning Algorithms," *IEEE Transactions on Information Theory*, 50, 2050–2057.

Cesa-Bianchi, N. and Lugosi, G. (2006), *Prediction, Learning, and Games*, Cambridge University Press.

Cotgreave, P. and Clayton, D. H. (1994), "Comparative Analysis of Time Spent Grooming by Birds in Relation to Parasite Load," *Behaviour*, 131, 171–187.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977), "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, 39, 1–38.

Duchi, J., Hazan, E., and Singer, Y. (2011), "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, 12, 2121–2159.

Frederic, P. and Lad, F. (2008), "Two Moments of the Logitnormal Distribution," *Communications in Statistics - Simulation and Computation®*, 37, 1263–1269.

Gneiting, T., Balabdaoui, F., and Raftery, A. E. (2007), "Probabilistic Forecasts, Calibration and Sharpness," *Journal of the Royal Statistical Society. Series B (Methodological)*, 69, 243–268.

Gneiting, T. and Katzfuss, M. (2014), "Probabilistic Forecasting," *Annual Review of Statistics and Its Application*, 1, 125–151.

Gneiting, T. and Raftery, A. E. (2007), "Strictly Proper Scoring Rules, Prediction and Estimation," *Journal of the American Statistical Association*, 102, 359–378.

Guolo, A. and Varin, C. (2014), "Beta Regression for Time Series Analysis of Bounded Data, with Application to Canada Google® Flu Trends," *The Annals of Applied Statistics*, 8, 74–88.

Hazan, E. (2022), *Introduction to Online Convex Optimization*, The MIT Press, 2nd edition.

Hazan, E., Levy, K. Y., and Shalev-Shwartz, S. (2015), "Beyond Convexity: Stochastic Quasi-Convex Optimization," in *Advances in Neural Information Processing Systems*, volume 28.

Heitjan, D. F. (1993), "Ignorability and Coarse Data: Some Biomedical Examples," *Biometrics*, 49, 1099–1109.

Heitjan, D. F. and Rubin, D. B. (1991), "Ignorability and Coarse Data," *The Annals of Statistics*, 19, 2244–2253.

Hersbach, H. (2000), "Decomposition of the Continuous Ranked Probability Score for Ensemble Prediction Systems," *Weather and Forecasting*, 15, 559–570.

Johnson, N. L. (1949), "Systems of Frequency Curves Generated by Methods of Translation," *Biometrika*, 36, 149–176.

Laderman, J. and Littauer, S. B. (1953), "The Inventory Problem," *Journal of the American Statistical Association*, 48, 717–732.

Ljung, L. and Södeström, T. (1983), *Theory and Practice of Recursive Identification*, The MIT Press.

Madsen, H. (2007), *Time Series Analysis*, Chapman & Hall.

McCullagh, P. and Nelder, J. A. (1989), *Generalized Linear Models*, Chapman & Hall, 2nd edition.

Mead, R. (1965), "A Generalised Logit-Normal Distribution," *Biometrics*, 21, 721–732.

Nesterov, Y. E. (1984), "Minimization Methods for Nonsmooth Convex and Quasiconvex Functions." *Matekon*, 29, 519–531.

Pierrot, A. and Pinson, P. (2021), "Adaptive Generalized Logit-Normal Distributions for Wind Power Short-Term Forecasting," in *2021 IEEE Madrid PowerTech*, Institute of Electrical and Electronics Engineers Inc.

Pinson, P. (2012), "Very-Short-Term Probabilistic Forecasting of Wind Power with Generalized Logit-Normal Distributions," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 61, 555–576.

Pinson, P. and Madsen, H. (2012), "Adaptive Modelling and Forecasting of Offshore Wind Power Fluctuations with Markov-Switching Autoregressive Models," *Journal of Forecasting*, 31, 281–313.

Wallis, K. F. (1987), "Time Series Analysis of Bounded Economic Variables," *Journal of Time Series Analysis*, 8, 115–123.

Warton, D. I. and Hui, F. K. C. (2011), "The arcsine is asinine: the analysis of proportions in ecology," *Ecology*, 92, 3–10.

Zinkevich, M. (2003), "Online Convex Programming and Generalized Infinitesimal Gradient Ascent," in *Proceedings of the 20th International Conference on Machine Learning*.

Calculation details

1 Algorithm 1: NGD

In this section,
$\theta = (\Lambda, \omega, \tau, b) = (\Lambda, \log\sigma^2, \log\nu, b) \in \mathbb{R}^{p+3}$,
$\mathbf{y} = (y_j) \in \mathbb{R}^{|C_t(\theta)|}$, where $y_j = \gamma(x_j/b; \nu)$ and $j \in C_t(\theta)$,
$\mathbf{Y}$ is a matrix with columns $B\mathbf{y}, B^2\mathbf{y}, \ldots, B^p\mathbf{y} \in \mathbb{R}^{|C_t(\theta)|\times p}$, where B is the backshift operator,
$\mathbf{L} = \mathbf{diag}\left(\alpha^{t-j}\right) \in \mathbb{R}^{|C_t(\theta)|\times|C_t(\theta)|}$ with $j \in C_t(\theta)$,
C is a constant which does not depend on θ.

1.1 Objective function

$$\begin{aligned}
-l_t^\infty(\theta) &= -\frac{1}{n_\alpha}\left[\sum_{j\in C_t(\theta)} \alpha^{t-j}\log p_j(\theta) + \sum_{j\in\overline{C_t}(\theta)} \alpha^{t-j}\log s_j(b)\right], \\
&= (1-\alpha)\left(\frac{\omega}{2} - \tau\right)\sum_{j\in C_t(\theta)} \alpha^{t-j} - (1-\alpha)\sum_{j\in C_t(\theta)} \alpha^{t-j}\log\left(1-(x_j/b)^{\exp\tau}\right) \\
&+ \frac{1-\alpha}{2\exp\omega}(\mathbf{y}-\mathbf{Y}\Lambda)^\top\mathbf{L}(\mathbf{y}-\mathbf{Y}\Lambda) + (1-\alpha)\sum_{j\in\overline{C_t}(\theta)} \alpha^{t-j}\log(1+\exp(-b+x_j)) \\
&+ C.
\end{aligned}$$

1.2 Gradient

First derivative w.r.t. Λ

$$\frac{-\partial l_t^\infty}{\partial\Lambda} = -\frac{1-\alpha}{\exp\omega}\,\mathbf{Y}^\top\mathbf{L}(\mathbf{y}-\mathbf{Y}\Lambda).$$

First derivative w.r.t. ω

$$\frac{-\partial l_t^\infty}{\partial\omega} = \frac{1-\alpha}{2}\sum_{j\in C_t(\theta)} \alpha^{t-j} - \frac{1-\alpha}{2\exp\omega}(\mathbf{y}-\mathbf{Y}\Lambda)^\top\mathbf{L}(\mathbf{y}-\mathbf{Y}\Lambda).$$

First derivative w.r.t. τ

$$\begin{aligned}
\frac{-\partial l_t^\infty}{\partial\tau} &= -(1-\alpha)\sum_{j\in C_t(\theta)} \alpha^{t-j} - (1-\alpha)\exp\tau\sum_{j\in C_t(\theta)} \alpha^{t-j}\exp y_j\log(x_j/b) \\
&+ \frac{1-\alpha}{\exp\omega}\,(\mathbf{u}-\mathbf{U}\Lambda)^\top\mathbf{L}(\mathbf{y}-\mathbf{Y}\Lambda),
\end{aligned}$$

where $\mathbf{u} = \dfrac{\partial\mathbf{y}}{\partial\tau}$, $u_j = \exp\tau\dfrac{\log(x_j/b)}{1-(x_j/b)^{\exp\tau}}$.

First derivative w.r.t. b

$$\frac{-\partial l_t^\infty}{\partial b} = (1-\alpha)\frac{\exp\tau}{b}\sum_{j\in C_t(\theta)}\alpha^{t-j}\exp y_j + \frac{1-\alpha}{\exp\omega}\,(\mathbf{z}-\mathbf{Z}\Lambda)^\top\mathbf{L}(\mathbf{y}-\mathbf{Y}\Lambda)$$
$$-(1-\alpha)\sum_{j\in\overline{C_t}(\theta)}\alpha^{t-j}\frac{\exp(-b+x_j)}{1-\exp(-b+x_j)},$$

where $\mathbf{z} = \dfrac{\partial\mathbf{y}}{\partial b}$, $z_j = -\dfrac{\exp\tau}{b\,(1-(x_j/b)^{\exp\tau})}$.

2 Algorithm 2: rMLE.b

In this section,
$\theta = (\Lambda, \omega, \tau, b) = (\Lambda, \log\sigma^2, \log\nu, b) \in \mathbb{R}^{p+3}$,
$\mathbf{y} = (y_{t-1}, \ldots, y_{t-p}) \in \mathbb{R}^p$, $y_j = \gamma(x_j/b;\nu)$.

2.1 Gradient

$$\mathbf{h}_t = \begin{cases}\nabla_\theta \log p_t(\hat\theta_{t-1}) & \text{if } t\in C_t(\theta),\\ \nabla_\theta \log s_t(\hat b_{t-1}) & \text{if } t\in\overline{C_t}(\theta).\end{cases}$$

First derivatives w.r.t. Λ

$$\frac{\partial\log p_t}{\partial\Lambda} = \frac{1}{\exp\omega}\,(y_t-\Lambda^\top\mathbf{y})\mathbf{y},$$
$$\frac{\partial\log s_t}{\partial\Lambda} = 0.$$

First derivatives w.r.t. ω

$$\frac{\partial\log p_t}{\partial\omega} = -\frac{1}{2}+\frac{1}{2\exp\omega}(y_t-\Lambda^\top\mathbf{y})^2,$$
$$\frac{\partial\log s_t}{\partial\omega} = 0.$$

First derivatives w.r.t. τ

$$\frac{\partial\log p_t}{\partial\tau} = 1+\exp\tau\exp y_t\log(x_t/b) - \frac{1}{\exp\omega}(u_t-\Lambda^\top\mathbf{u})(y_t-\Lambda^\top\mathbf{y}),$$
$$\frac{\partial\log s_t}{\partial\tau} = 0,$$

where $\mathbf{u} = \dfrac{\partial\mathbf{y}}{\partial\tau}$, $u_t = \exp\tau\dfrac{\log(x_t/b)}{1-(x_t/b)^{\exp\tau}}$.

First derivatives w.r.t. b

$$\frac{\partial\log p_t}{\partial b} = -\frac{\exp\tau}{b}\exp y_t + \frac{1}{\exp\omega}(z_t-\Lambda^\top\mathbf{z})(y_t-\Lambda^\top\mathbf{y}),$$
$$\frac{\partial\log s_t}{\partial b} = \frac{\exp(-b+x_t)}{1+\exp(-b+x_t)},$$

where $\mathbf{z} = \dfrac{\partial\mathbf{y}}{\partial b}$, $z_t = \dfrac{\exp\tau}{b\,(1-(x_j/b)^{\exp\tau})}$.

2.2 Detailed computation of the matrix $\hat{\mathbf{P}}_t$

We use the matrix inversion rule

$$[\mathbf{A}+\mathbf{BCD}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left[\mathbf{DA}^{-1}\mathbf{B}+\mathbf{C}^{-1}\right]^{-1}\mathbf{DA}^{-1},$$

with $\mathbf{A} = \alpha\hat{\mathbf{R}}_{t-1}$, $\mathbf{B} = \mathbf{D}^\top = \mathbf{h}_t$, $\mathbf{C} = (1-\alpha)\mathbf{I}$ and we get

$$\begin{aligned}
\hat{\mathbf{P}}_t &= \hat{\mathbf{R}}_t^{-1}, \\
&= \left[\alpha\hat{\mathbf{R}}_{t-1} + (1-\alpha)\mathbf{h}_t\mathbf{h}_t^\top\right]^{-1}, \\
&= \left(\alpha\hat{\mathbf{R}}_{t-1}\right)^{-1} - \left(\alpha\hat{\mathbf{R}}_{t-1}\right)^{-1}\mathbf{h}_t\left[\mathbf{h}_t^\top\left(\alpha\hat{\mathbf{R}}_{t-1}\right)^{-1}\mathbf{h}_t + ((1-\alpha)\mathbf{I})^{-1}\right]^{-1}\mathbf{h}_t^\top\left(\alpha\hat{\mathbf{R}}_{t-1}\right)^{-1}, \\
&= \frac{1}{\alpha}\hat{\mathbf{P}}_{t-1} - \frac{1}{\alpha^2}\hat{\mathbf{P}}_{t-1}\mathbf{h}_t\left[\frac{1}{\alpha}\mathbf{h}_t^\top\hat{\mathbf{P}}_{t-1}\mathbf{h}_t + \frac{1}{1-\alpha}\right]^{-1}\mathbf{h}_t^\top\hat{\mathbf{P}}_{t-1}, \\
&= \frac{1}{\alpha}\hat{\mathbf{P}}_{t-1} - \frac{\hat{\mathbf{P}}_{t-1}\mathbf{h}_t\mathbf{h}_t^\top\hat{\mathbf{P}}_{t-1}}{\alpha^2\left[\frac{1}{\alpha}\mathbf{h}_t^\top\hat{\mathbf{P}}_{t-1}\mathbf{h}_t + \frac{1}{1-\alpha}\right]}, \\
&= \frac{1}{\alpha}\left[\mathbf{I} - \frac{\hat{\mathbf{P}}_{t-1}\mathbf{h}_t\mathbf{h}_t^\top}{\frac{\alpha}{1-\alpha} + \mathbf{h}_t^\top\hat{\mathbf{P}}_{t-1}\mathbf{h}_t}\right]\hat{\mathbf{P}}_{t-1}.
\end{aligned}$$

3 Algorithm 3: ONGD

In this section,
$\theta = (\Lambda, \omega, \tau, b) = (\Lambda, \log\sigma^2, \log\nu, b) \in \mathbb{R}^{p+3}$,
$U_t^m = \{t-m+1, \ldots, t\}$,
$C_t^m(\theta) = \{j \in U_t^m \mid x_{j-k} < b, k = 0, \ldots, p\}$, $\overline{C_t^m}(\theta) = \{j \in U_t^m | j \notin C_t^m(\theta)\}$,
$\mathbf{y} = (y_j) \in \mathbb{R}^{|C_t^m(\theta)|}$, where $y_j = \gamma(x_j/b; \nu)$ and $j \in C_t^m(\theta)$,
$\mathbf{Y}$ is a matrix with columns $B\mathbf{y}, B^2\mathbf{y}, \ldots, B^p\mathbf{y} \in \mathbb{R}^{|C_t^m(\theta)|\times p}$, where B is the backshift operator,
C is a constant which does not depend on θ.

3.1 Objective function

$$\begin{aligned}
f_t(\theta) &= \frac{1}{m}\left[\sum_{j\in C_t^m(\theta)}\log p_j(\theta) + \sum_{j\in\overline{C_t^m}(\theta)}\log s_j(b)\right], \\
&= \frac{1}{m}\left(\frac{\omega}{2} - \tau\right)|C_t^m(\theta)| - \frac{1}{m}\sum_{j\in C_t^m(\theta)}\log\left(1-(x_j/b)^{\exp\tau}\right) \\
&+ \frac{1}{m}\frac{1}{2\exp\omega}(\mathbf{y}-\mathbf{Y}\Lambda)^\top(\mathbf{y}-\mathbf{Y}\Lambda) + \frac{1}{m}\sum_{j\in\overline{C_t^m}(\theta)}\log(1+\exp(-b+x_j)) \\
&+ C.
\end{aligned}$$

3.2 Gradient

First derivative w.r.t. Λ

$$\frac{\partial f_t}{\partial\Lambda} = -\frac{1}{m\exp\omega}\,\mathbf{Y}^\top(\mathbf{y}-\mathbf{Y}\Lambda).$$

First derivative w.r.t. ω

$$\frac{\partial f_t}{\partial \omega} = \frac{1}{2m}|C_t^m(\theta)| - \frac{1}{2m\exp\omega}(\mathbf{y} - \mathbf{Y}\Lambda)^\top(\mathbf{y} - \mathbf{Y}\Lambda).$$

First derivative w.r.t. τ

$$\frac{\partial f_t}{\partial \tau} = -\frac{1}{m}|C_t^m(\theta)| - \frac{\exp\tau}{m}\sum_{j\in C_t^m(\theta)} \exp y_j \log(x_j/b) + \frac{1}{m\exp\omega}\ (\mathbf{u} - \mathbf{U}\Lambda)^\top(\mathbf{y} - \mathbf{Y}\Lambda),$$

where $\mathbf{u} = \dfrac{\partial \mathbf{y}}{\partial \tau}$, $u_j = \exp\tau \dfrac{\log(x_j/b)}{1-(x_j/b)^{\exp\tau}}$.

First derivative w.r.t. b

$$\frac{\partial f_t}{\partial b} = \frac{1}{m}\frac{\exp\tau}{b}\sum_{j\in C_t^m(\theta)} \exp y_j + \frac{1}{m\exp\omega}\ (\mathbf{z} - \mathbf{Z}\Lambda)^\top(\mathbf{y} - \mathbf{Y}\Lambda) - \frac{1}{m}\sum_{j\in \overline{C_t^m}(\theta)} \frac{\exp(-b+x_j)}{1-\exp(-b+x_j)},$$

where $\mathbf{z} = \dfrac{\partial \mathbf{y}}{\partial b}$, $z_j = -\dfrac{\exp\tau}{b\,(1-(x_j/b)^{\exp\tau})}$.

[Paper C] Mixtures of bounded distributions with different bounds: Estimation and forecasting

Authors:
Amandine Pierrot and Pierre Pinson

To be submitted to:
Annals of Applied Statistics

Technical report as of now

Submitted to the Annals of Applied Statistics (technical report as of now)

MIXTURES OF BOUNDED DISTRIBUTIONS WITH DIFFERENT BOUNDS: ESTIMATION AND FORECASTING

BY AMANDINE PIERROT[1,a] AND PIERRE PINSON[2,b]

[1] *Technical University of Denmark, Department of Wind and Energy Systems,* [a] *amapi@dtu.dk*

[2] *Imperial College London, Dyson School of Design Engineering,* [b] *p.pinson@imperial.ac.uk*

We propose a new framework where the upper bound of a continuous bounded response variable is a discrete latent variable which can take a finite number of values over the unit interval. This is equivalent to modelling the marginal distribution of the bounded variable as a finite mixture of bounded distributions with different bounds. We focus on the generalized logit-normal distribution for the bounded variable, because of its flexibility and the ease with which it can be fitted. We propose batch EM-based algorithms for two frameworks: first, when the response variables are independent and identically distributed; second, when the response variables are a stochastic process with an auto-regressive dependency structure. We also suggest online EM-based algorithms which rely on online gradient descent. The performance of the proposed algorithms is evaluated through a simulation study for both frameworks. We address convergence in stationary settings and tracking in changing environments. For stochastic processes, we also consider the performance of the online algorithms for probabilistic forecasting. The batch algorithms and the online EM algorithm for the independent case show good performances. However, the online algorithm dealing with past dependent observations converges to biased estimates, which needs to be addressed in further work.

1. Introduction. When interested in response variables which are both continuous and bounded, the support of the variable distribution is usually assumed to be fixed to the unit interval. However, in many statistical applications the bounds of the support are by nature varying, and in particular the upper bound, for instance during the spread of an epidemic, see, e.g., Guolo and Varin (2014), for unemployment rates, see Wallis (1987), or any item of an inventory problem, see Laderman and Littauer (1953). For applications to renewable energy, both wind and solar power generation are impacted by an upper varying bound. Indeed, wind power generation is a random variable which is usually scaled by the nominal power of the turbine, but curtailment actions happen, for which information is not available or not reliable. As for solar power generation, it is also often expressed as a percentage of the nominal power of the power plant, which can shift over time for no documented reason, see, e.g., Zamo et al. (2014a,b). Even if the nominal power remains constant, some phenomena may occur that limit the effective capacity, such as dust in desert areas.

Bounds which vary while not being observed or documented may negatively impact statistical inference of the distribution of the bounded response variable. When the bounded variable is a stochastic process, a missing bound can be thought of as a an additional, scaling parameter of a parametric bounded distribution, and estimated along with the other parameters while assuming non-stationarity, see Pierrot and Pinson (2023). In a more general, still parametric framework, and to account for the inherent uncertainty of the value taken by the bound, we propose to address a missing bound as a latent random variable with its own

Keywords and phrases: EM algorithms, Monte Carlo EM algorithms, Stochastic/online EM algorithms, Generalized logit-normal distribution.

probability distribution. This calls for methods of the expectation-maximization (EM) kind, see Dempster, Laird and Rubin (1977), in particular if one is interested in forecasting the bounded response variable, for the bound will have to be forecast first.

Probability distributions for a continuous bounded variable include the beta distribution, truncated distributions or distributions of transformed normal variables, see, e.g., Johnson (1949). We choose to work with a distribution that belongs to the latter family, the generalized logit-normal (GLN) distribution introduced by Mead (1965), because of all the variations in its shape which are enabled by three - location, scale and shape - parameters, and the ease with which it can be fitted. The choice of a probability distribution for the bound is rather open but crucial to the difficulty of the subsequent EM algorithm. Hence, and from the intuition that a grid of plausible values over the unit interval may matter more to inference than the accuracy of each value, we propose to assume the bound to be a discrete variable which can take K values over $(0, 1]$. The marginal pdf of the bounded variable then becomes a finite mixture of K scaled GLN distributions which is similar to a kernel density with scaled GLN kernels.

To allow for a broad range of applications, we address two assumptions for the bounded random variables: first, the case when they are independent and identically (iid) distributed, and second, the case when they are a stochastic process with an auto-regressive (AR) dependency structure. For both cases we propose batch and online EM algorithms. We focus on an upper bound, as most applications which come to mind involve such a varying bound. Nonetheless, it would be straightforward to apply the method to a lower bound. In the independent case, we propose a generalized EM algorithm, the expectation-conditional maximization algorithm, see Meng and Rubin (1993), and an online version which relies on gradient descent at the M-step. In the case of stochastic processes with serial correlations, we propose a Monte Carlo EM algorithm (MCEM), see Wei and Tanner (1990). We also suggest two online versions: both apply gradient descent at the M-step, as in the independent case; at the E-step, in order to deal with the past missing values of the bound, the first online version uses filtering while the second version uses a stochastic E-step. Although they can be of practical interest for stationary frameworks, in particular in the case of stochastic processes as the MCEM algorithm can be computationally heavy, our call for online EM algorithms lies mostly in their tracking abilities, so that the distribution of the bound, which is rather simple, can at least adapt to changing environments.

In Section 2, we develop the statistical models and introduce the corresponding maximum likelihood inference to be performed through EM-based algorithms. Then, algorithms are proposed in Section 3 for the independent case and in Section 4 for stochastic processes with AR dependency. The convergence and tracking performance of these algorithms is illustrated in Section 5 through a simulation study for both frameworks. In addition, the forecasting performance of online EM algorithms is evaluated on the synthetic stochastic processes. Finally, we give some conclusions and perspectives for future work according to the results we obtain during the simulation study.

We use capital letters to denote random variables, e.g., random variable X. We use bold capital letters to denote matrices, e.g., matrix $\mathbf{Y}$. We use small bold letters to denote vectors, e.g., vector $\mathbf{w}$. We use parentheses to construct column vectors from comma separated lists. For instance, if $a, b, c \in \mathbb{R}$, (a, b, c) is the column vector of elements a, b and c. We use square brackets to refer to the i-th element $\mathbf{w}[i]$ of a vector $\mathbf{w}$. We use parentheses to denote a sample, which can also be seen as a vector of observations, e.g., a sample/vector $\mathbf{x} = (x_1, \ldots, x_n) = (x_i)_{i=1,\ldots,n}$.

2. A mixture of scaled generalized logit-normal distributions. Let X be a continuous bounded random variable which lies in the unit interval, $X \in (0, 1)$. We argue that while thinking we observe the realizations x of X, we in fact only have access to the value

$x/b \in (0,1)$ and implicitly assume $b = 1$. The value b is the value we assume for the upper bound of the bounded support $(0, b)$ of the distribution of X. To account for the possibility it might not always take the value 1, while not being observed, and acknowledge its randomness, we propose to consider b as the realization of a latent random variable B, i.e., the variable X is upper bounded by a latent variable B. We focus on parametric distributions. The general framework where B is a latent variable makes it possible to specify any kind of parametric probability distribution for B. However, because B is missing, and because we are also interested in forecasting applications, we need to use methods based on the EM algorithm in order to estimate the parametric distributions of both the bounded variable X and its bound B, see Dempster, Laird and Rubin (1977). How simple or complicated the inference through the EM algorithm will be depends on this specification. A simple model for B eases the inference but will likely not perform well when moving to forecasting. We propose to assume B to be a discrete random variable which takes a finite number of values over $(0, 1]$. For real application purposes, our idea stands from choosing K values for B in such a way that the unit interval is covered by a fine grid of plausible values. This grid can be decided upon a priori knowledge on the real data, or refined over the course of the modelling and learning procedure. In Section 2.1, we introduce the corresponding statistical model under the assumption that the bounded variables X are iid conditional on B. In Section 2.2, we extend the model to discrete-time stochastic processes $(X_t)_{t\in\mathcal{T}}$, assuming a simple Markov dependency structure through an AR model.

2.1. *Statistical model under the independence assumption.* Let B take K values $b[k] \in (0, 1]$ with probabilities $\mathbf{w} = (w[1], \ldots, w[K])$, where $k \in \mathcal{K} = \{1, \ldots, K\}$ is an unobserved state associated with $b[k]$. We choose this notation in order to emphasize the bijection between a value of b and a state k. Given $B = b[k]$, we assume X to be distributed according to a GLN distribution scaled to $(0, b[k])$ with pdf

$$p(x \mid b[k]; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \frac{\nu}{x\,(1 - (x/b[k])^{\nu})} \exp\left[-\frac{1}{2}\left(\frac{\gamma(x/b[k]; \nu) - \mu}{\sigma}\right)^2\right] \tag{1}$$

if $0 < x < b[k]$, $p(x \mid b[k]; \theta) = 0$ otherwise, with $\theta = (\mu, \sigma^2, \nu)$ and $\gamma(x/b[k]; \nu) = \log \frac{(x/b[k])^{\nu}}{1-(x/b[k])^{\nu}}$, $\nu > 0$. We get the following joint probability distribution of (X, B)

$$p(x \mid b[k]; \theta) w[k], \tag{2}$$

and the marginal pdf of X

$$\sum_{k=1}^{K} p(x \mid b[k]; \theta) w[k], \tag{3}$$

which is indeed a K-component mixture of scaled GLN densities. Let $\mathbf{x} = (x_i)_{i=1,\ldots,n}$ be an iid sample from (1), $\mathcal{K}_i = \{k \in \mathcal{K} \mid b[k] > x_i\}$. The complete-data log-likelihood associated with our model is

$$\sum_{i=1}^{n} \sum_{k\in\mathcal{K}_i} \mathbb{1}_{\{B_i = b[k]\}} \left(\log p(x_i \mid b[k]; \theta) + \log w[k]\right). \tag{4}$$

In order to estimate the full parameter ψ through maximum likelihood inference, we wish to apply an EM-based algorithm to (4). The Q function computed at the E-step of iteration j is

$$Q(\psi|\psi^{(j)}) = \mathbb{E}\left[\sum_{i=1}^{n} \sum_{k\in\mathcal{K}_i} \mathbb{1}_{\{B_i = b[k]\}} \left(\log p(x_i \mid b[k]; \theta) + \log w[k]\right) \,\middle|\, \mathbf{x}; \psi^{(j)}\right]$$

$$= \sum_{i=1}^{n} \sum_{k \in \mathcal{K}_i} \hat{w}_i[k] \left(\log p(x_i \mid b[k]; \theta) + \log w[k] \right), \tag{5}$$

where, using Bayes formula,

$$\hat{w}_i[k] = \frac{p(x_i \mid b[k]; \theta^{(j)})\ w^{(j)}[k]}{\sum_{l=1}^{K} p(x_i \mid b[k]; \theta^{(j)})\ w^{(j)}[l]}. \tag{6}$$

Then, the M-step of iteration j aims to maximize $Q(\psi|\psi^{(j)})$ with respect to ψ, subject to $\sum_{k=1}^{K} w[k] = 1$. In fact, the M-step does not need to be a maximization for sufficiently increasing $Q(\psi|\psi^{(j)})$ at iteration j is enough to give an ascent algorithm, which is usually termed as a generalized EM algorithm, see Dempster, Laird and Rubin (1977). For simplicity, we have assumed a common θ over the different states $k \in \mathcal{K}$. It would be straightforward to make θ depend on k. Also, note that the location parameter μ corresponds to the expectation of the normally distributed transform $Y = \gamma(X/b[k]; \nu)$. Other assumptions could be made for the expectation of Y, e.g., it could be conditional upon exogenous variables through a linear regression.

In Section 3, we design EM-based algorithms to iteratively increase (5). In particular, we design an ECM algorithm, see Meng and Rubin (1993). We also propose an online EM algorithm to increase (5) until convergence in stationary frameworks or to track the parameter vector ψ in changing environments.

2.2. *Statistical model for stochastic processes with serial correlations.* We adapt the former statistical model to serial correlations when (X_t) is a stochastic process. We denote by Y_t the generalized logit transform, $Y_t = \gamma(X_t/b[k]; \nu)$. We model the expectation of Y_t as an AR model of order p, i.e, $\mathbb{E}(Y_t) = \mu_t = \sum_{r=1}^{p} \lambda_r y_{t-r}$. Let $\mathbf{x} = (x_t)_{t=1,\dots,T}$, $\mathbf{b} = (b_t)_{t=1,\dots,T}$, $\mathbf{x}_{t:(t-p)} = (x_t, \dots, x_{t-p})$, $\mathbf{b}_{t:(t-p)} = (b_t, \dots, b_{t-p}) = (b[k_t], \dots, b[k_{t-p}])$, where k_t is the unobserved state at time t. Finally, let $\mathcal{F}_{t-1}$ be the previous information set for X_t, i.e., the σ-algebra generated by $X_1, \dots, X_{t-1}$. The conditional pdf of X_t is

$$p(x_t \mid \mathcal{F}_{t-1}, \mathbf{b}_{t:(t-p)}; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \frac{\nu}{x_t \left(1 - (x_t/b_t)^{\nu}\right)} \exp\left[-\frac{1}{2} \left(\frac{\gamma(x_t/b_t; \nu) - \mu_t}{\sigma} \right)^2 \right] \tag{7}$$

if $0 \prec \mathbf{x}_{t:(t-p)} \prec \mathbf{b}_{t:(t-p)}$, $p(x_t \mid \mathcal{F}_{t-1}, \mathbf{b}_{t:(t-p)}; \theta) = 0$ otherwise. Note that for stochastic processes, only a common θ over the different unobserved states $k \in \mathcal{K}$ is possible. The log-likelihood of a sample $(\mathbf{x}, \mathbf{b})$ is

$$\sum_{t=p+1}^{T} \left(\log p(x_t \mid \mathcal{F}_{t-1}, \mathbf{b}_{t:(t-p)}; \theta) + \log w[k_t] \right), \tag{8}$$

as we do not take into account the distribution of the first p observed values $\mathbf{x}_{p:1}$ and consider instead the likelihood conditional on them. The associated Q function is

$$Q(\psi|\psi^{(j)}) = \mathbb{E}\left[\sum_{t=p+1}^{T} \left(\log p(x_t \mid \mathcal{F}_{t-1}, \mathbf{b}_{t:(t-p)}; \theta) + \log w[k_t] \right) \,\middle|\, \mathbf{x}; \psi^{(j)} \right]. \tag{9}$$

Because X_t needs to be considered given the former (missing) values of B_t, it is not straightforward anymore to compute Q, as it was in (5). Therefore, we propose to approximate it through Monte Carlo integration by

$$\hat{Q}_M(\psi|\psi^{(j)}) = \frac{1}{M} \sum_{m=1}^{M} \sum_{t=p+1}^{T} \left(\log p(x_t \mid \mathcal{F}_{t-1}, \mathbf{b}^m_{t:(t-p)}, ; \theta) + \log w[k_t^m] \right), \tag{10}$$

where M is the number of Monte Carlo samples and $\mathbf{b}^m = (b[k_1^m], \ldots, b[k_T^m])$ is distributed according to the posterior distribution $p(\mathbf{b} \mid \mathbf{x}; \psi^{(j)})$. This leads to the EM algorithm being replaced with an MCEM algorithm, see Wei and Tanner (1990). This kind of algorithm comes with its own challenges, in particular regarding the appropriate number of samples M, as it is not efficient to start with a large value of M when the maximum likelihood estimate (MLE) might be far from the true value of ψ.

To iteratively increase (10), we design an MCECM algorithm which combines Markov chain Monte Carlo (MCMC) in the E-step, see Metropolis et al. (1953); Hastings (1970), and conditional maximization in the M-step, see Meng and Rubin (1993). This algorithm is presented in Section 4, along with two suggestions of online algorithms.

3. EM-based algorithms for inference under the independence assumption.

3.1. *A batch ECM algorithm.* The E-step of the EM algorithm is straightforward and consists of computing the posterior probabilities in (6) for all $i = 1, \ldots, n$ and $k = 1, \ldots, K$. Regarding the M-step, because of the shape parameter $\nu > 0$ of the GLN distribution, the direct application of EM to our problem would require applying a $(K + 3)$-dimensional Newton-Raphson algorithm, which is typically less stable than one-dimensional Newton-Raphson, as there is no closed-form solution when maximizing the Q function in (5) with respect to ν. Moreover, our maximization problem is a constrained optimization problem, as the probabilities of the values taken by the bound, i.e., of the states $k = 1, \ldots, K$, need to be non-negative and sum up to 1. Finally, note that conditional on the other parameters, $\mathbf{w}$, μ and σ^2 all have closed-form solutions. Therefore, following Meng and Rubin (1993), we recommend to replace the M-step of the EM algorithm with several simpler conditional maximization (CM) steps, taking advantage of the closed-form solutions available for all parameters but ν. This is the ECM algorithm, which shares all the appealing convergence properties of EM, such as always increasing the log-likelihood. In the absence of missing data, ECM is a special case of cyclic coordinate ascent methods, see, e.g., Zangwill (1969); Jensen, Johansen and Lauritzen (1991). As argued by Meng and Rubin, when used at the M-step of EM, simple and stable linear converging methods are often more suitable than super-linear converging, but less stable algorithms, such as Newton's method. Indeed, the super-linear convergence does not directly transfer to EM-based algorithms, for the latter converge linearly regardless of the maximization method employed within the M-step. Therefore, stability is to be preferred since the maximization method is used repeatedly in all iterations. The ECM algorithm we propose is given in Algorithm 1. In each iteration, first, three CM-steps compute the closed-form solutions of $\mathbf{w}$, μ and σ^2 given the current value of the other parameters. The fourth CM-step is a one-dimensional Newton-Raphson step which then updates the current value of ν. Let us define the set of functions $G = \{g_s, s = 1, \ldots, S\}$, with $S = 4$, $g_1(\psi) = \mathbf{w}$, $g_2(\psi) = \mu$, $g_3(\psi) = \sigma^2$ and $g_4(\psi) = \nu$. The condition for the ECM to share the convergence properties with EM is that G is *space filling* so that we are guaranteed the resulting maximum is an unconstrained maximum of the likelihood (over the whole space of the parameters). This is verified if

$$J(\psi) = \cap_{s=1}^{S} J_s(\psi) = \{0\},$$

where $J_s(\psi) = \frac{\partial}{\partial \psi} g_s(\psi)$, which can be easily checked in our case (and in many applications). Note that the last CM-step that applies Newton's method is not exactly a maximization step, as we perform backtracking line search, not exact line search, see, e.g., Boyd and Vandenberghe (2004). Performing backtracking line search is simpler and faster, and enough to ensure

$$Q(\psi^{(j+1)}|\psi^{(j)}) \geq Q(\psi^{(j+\frac{S-1}{S})}|\psi^{(j)}), \tag{14}$$

Algorithm 1 ECM algorithm (independence assumption)

Input: #Iterations J, initial guess ψ_0, $\epsilon > 0$
for $j = 0, \ldots, J$ **do**
1. **E-Step**
Compute:

$$\hat{w}_i[k] = \frac{p(x_i \mid b[k]; \theta^{(j)})\, w^{(j)}[k]}{\sum_{l=1}^{K} p(x_i \mid b[l]; \theta^{(j)})\, w^{(j)}[l]} \quad \forall\, i = 1, \ldots, n,\ k = 1, \ldots, K.$$

2. **CM-Steps**
Update:

$$w^{(j+1)}[k] = \frac{1}{n} \sum_{i=1}^{n} \hat{w}_i[k] \quad \forall\, k = 1, \ldots, K, \tag{11}$$

$$\mu^{(j+1)} = \frac{1}{n} \sum_{i=1}^{n} \sum_{k \in \mathcal{K}_i} \hat{w}_i[k] \gamma(x_i / b[k]; \nu^{(j)}), \tag{12}$$

$$\sigma^{2(j+1)} = \frac{1}{n} \sum_{i=1}^{n} \sum_{k \in \mathcal{K}_i} \hat{w}_i[k] (\gamma(x_i / b[k]; \nu^{(j)}) - \mu^{(j+1)})^2 \tag{13}$$

Compute: Newton step $\Delta\nu_{nt}$ and decrement λ^2.
if $\lambda^2/2 \leq \epsilon$ **then** $\nu^{(j+1)} = \nu^{(j)}$
else Choose step size η by backtracking line search and Update: $\nu^{(j+1)} = \nu^{(j)} + \eta \Delta\nu_{nt}$.
end if
end for

where the right-hand side of (14) is the value of the Q function before updating ν in iteration j; the left-hand side of (14) is the value of Q after updating ν in iteration j, i.e., at the end of iteration j.

3.2. *Online EM algorithm for inference under the independence assumption.* Online versions of the EM algorithm have been mostly designed for distributions which belong to curved exponential families, see, e.g. Cappé and Moulines (2009); Cappé (2011); Corff and Fort (2013). A notable exception is Titterington's algorithm, which does not make any distribution assumption, but requires the computation of the complete-data Fisher information. Because GLN distributions do not belong to curved exponential families, and there are no available analytical expression for their first moments, those algorithms cannot be applied to a mixture of GLN distributions. Even if closed-form solutions are available for μ and σ^2, which are the usual closed-form solutions for a normal distribution, we cannot make use of the corresponding, convenient recursive equations for updating them, because of the transformation γ which involves the shape parameter ν. Therefore, looking into the class of online learning algorithms, see, e.g., Cesa-Bianchi and Lugosi (2006), and in particular at online gradient descent (OGD), see, e.g., Bottou (1999), we propose to consider each EM iteration as a round in a repeated game, where we take the best position we can given our past choices, without looking backwards: at each iteration i, we look at an instance of $-Q$ which only depends on the new observation x_i and the current value of the parameter vector ψ_i, and update the latter through a descent step using the gradient of this instance of $-Q$. Note that because we place ourselves in the online setting, the former superscripts (j) are replaced with the subscripts i of the observations, as one iteration corresponds to one observation. We treat the observations sequentially, so that the algorithm can be transposed to tracking the parameter vector in a changing environment. However, they could also be picked randomly, as in both cases what eventually matters is that they are independent observations. The corresponding

Algorithm 2 Online gradient-based EM (independence assumption)

Input: convex set $\mathcal{K}, \psi_1 \in \mathcal{K}$, step size η

for $i = 1, \ldots, n$ **do**

Observe $-Q_i(\psi|\psi_i) = -\sum_{k \in \mathcal{K}_i} \hat{w}_i[k] \left(\log p(x_i \mid b[k]; \theta) + \log w[k]\right)$,

where $\hat{w}_i[k] \propto p(x_i \mid b[k]; \theta_i)\ w_i[k]$.

Update and project:

$$\xi_{i+1} = \psi_i + \eta \nabla_\psi Q_i(\psi|\psi_i), \tag{15}$$

$$\psi_{i+1} = \Pi_{\mathcal{K}}(\xi_{i+1}). \tag{16}$$

end for

algorithm is described in Algorithm 2. Note that the gradient descent step at iteration i in (15) is followed by a projection step back to the convex set $\mathcal{K}$ in (16), as it might result in values ξ_{i+1} outside of $\mathcal{K}$. In our case, this projection step is necessary for the parameter $\mathbf{w}$, which is constrained to belong to the probability simplex $\mathbf{P} = \{\mathbf{w} \in \mathbb{R}_+^K \mid \sum_{k=1}^K w[k] = 1\}$. The algorithm for projecting the updated $\mathbf{w}$ back to the simplex is given in Appendix A.

Online learning and stochastic optimization are closely related, see, e.g., Bottou (1999); Cesa-Bianchi, Conconi and Gentile (2004). The online EM version we propose borrows ideas from stochastic optimization by using noisy estimates of the objective's gradient. Stochasticity in EM algorithms can come in different forms, see, e.g, Celeux, Chauveau and Diebolt (1996); Cappé and Moulines (2009). Here, stochasticity comes from the observation x_i which is picked randomly, or sequentially, out of the batch sample $\mathbf{x}$, and addressed at iteration i. Then, the gradient which is used at the M-step is the gradient of $-Q_i$ only, instead of the gradient of $-Q$. First, note that minimizing $-Q(\psi|\psi_i)$ is equivalent to minimizing $-\frac{1}{n}Q(\psi|\psi_i)$. Our update term at iteration i, $-\nabla_\psi Q_i(\psi|\psi_i)$, fulfills the condition

$$\mathbb{E}_X[-\nabla_\psi Q_i(\psi|\psi_i)] = -\frac{1}{n}\nabla_\psi Q(\psi|\psi_i), \tag{17}$$

i.e, the gradient of $-Q_i(\psi|\psi_i)$ is a noisy but unbiased estimate of the gradient of $-\frac{1}{n}Q(\psi|\psi_i)$, since the x_i are independent realizations of X. Because the gradient estimate is unbiased, we can hope that using only one observation at each iteration will not compromise the average behaviour of the EM algorithm, i.e., we will recover $Q(\psi_{i+1}|\psi_i) > Q(\psi_i|\psi_i)$ on average. The convergence of Algorithm 2 to stationary points could be further analyzed using general results from Bottou (1999).

A way to understand how OGD works is to think that we update ψ using the minimizer of a linear approximation of $-Q_i(\psi|\psi_i)$, the minimization being constrained in a neighborhood of ψ_i, where we have good reason to believe the approximation is more precise, see Orabona (2022). Going back to the objective function we wish to minimize at iteration i, i.e, $-\log p(x_i; \psi)$, and using Jensen's inequality, we have

$$-\log p(x_i;\psi) = -\log \sum_{k=1}^K q_k \frac{p(x_i \mid b[k];\theta)\ w[k]}{q_k} \leq -\sum_{k=1}^K q_k \log \frac{p(x_i \mid b[k];\theta)\ w[k]}{q_k}, \tag{18}$$

see, e.g., Neal and Hinton (1998). In Algorithm 2, observing $-Q_i(\psi|\psi_i)$ with $q_k = \hat{w}_i[k]$ comes down to replacing the expectation step with a stochastic approximation step that makes the upper bound in (18) tight by minimizing it w.r.t. the distribution $\mathbf{q} = (q_1, \ldots, q_K)$. The gradient descent step then minimizes a linear approximation of this local upper bound w.r.t. ψ in the neighborhood of ψ_i. This gives an intuition about the behaviour of Algorithm 2

as a tracking algorithm in changing environments, by going back to looking at EM as a maximization-maximization algorithm from an online, local, perspective.

The online EM algorithm we propose has connections with Titterington's method, which consists of using a stochastic approximation algorithm, where the parameters are updated after each new observation by using the gradient of the incomplete-data likelihood weighted by the complete-data Fisher information matrix. However, in our specific mixture case, we remain closer to the spirit of EM by having still an E-step through the computation of $\hat{w}_i[1], \ldots, \hat{w}_i[K]$ in $-Q_i(\psi|\psi_i)$, and an M-step through the descent step in the direction of the negative gradient of $-Q_i(\psi|\psi_i)$. Moreover, we do not need to compute and inverse the complete-data Fisher information matrix, which would be prohibitive in our case.

4. EM-based algorithms for inference with serial correlations.

4.1. *A batch MCECM algorithm.* The challenging part of EM when moving to stochastic processes with serial correlations is the E-step which requires to properly simulate M samples $\mathbf{b}^m$ according to $p(\mathbf{b} \mid \mathbf{x}; \psi^{(j)})$, see Section 2.2. In order to perform the Monte Carlo integration required in (10), and to deal with the dependencies introduced into our observations by the new hypothesis on the expectation of X_t, we consider Markov chain algorithms. There are many ways of constructing Markov chains to eventually approximate the desired expectation, see, e.g., Gilks, Richardson and Spiegelhalter (1995); Robert and Casella (2004), but all of them are special cases of the general framework of Metropolis et al. (1953) and Hastings (1970). We use a single-component Metropolis-Hastings (MH) algorithm with an independence sampler. Let b_t^m denote the value of b_t at the end of iteration m. For step t of iteration $m+1$, b_t is updated using MH. The candidate B_t^* is generated from a proposal distribution $q_t(B_t^*|b_t^m, \mathbf{b}_{-t}^m)$, where $\mathbf{b}_{-t}^m$ denote the value of $\mathbf{b}_{-t}$ after completing step $t-1$ of iteration $m+1$,

$$\mathbf{b}_{-t}^m = \{b_1^{m+1}, \ldots, b_{t-1}^{m+1}, b_{t+1}^m, \ldots, b_T^m\}, \tag{19}$$

and components $1, \ldots, t-1$ have already been updated. This is the updating scheme of the single-component MH algorithm, which is in fact the framework originally proposed by Metropolis et al. (1953). To use an independence sampler means to use a proposal distribution $q_t(B_t^*|\mathbf{b}_{-t}^m)$ instead of $q_t(B_t^*|b_t^m, \mathbf{b}_{-t}^m)$. In general, the independence sampler can work very well or very badly. For it to work well, $q_t(.)$ should be a good approximation to the target distribution. It is rather easy to fulfill this condition here, as $q_t(B_t^*|\mathbf{b}_{-t}^m) = q_t(B_t^*)$, since the random variables B_t are iid and we can choose $q_t(.)$ to be the marginal distribution of b_t truncated according to the observed value x_t. The candidate B_t^* is then accepted with probability

$$\rho(\mathbf{b}_{-t}^m, b_t^m, B_t^*) = \min\left(1, \frac{p(B_t^* \mid \mathbf{b}_{-t}^m, \mathbf{x}; \psi^{(j)})\, q_t(b_t^m; \psi^{(j)})}{p(b_t^m \mid \mathbf{b}_{-t}^m, \mathbf{x}; \psi^{(j)})\, q_t(B_t^*; \psi^{(j)})}\right), \tag{20}$$

and straightforward calculation gives

$$\frac{p(B_t^* \mid \mathbf{b}_{-t}^m, \mathbf{x}; \psi^{(j)})\, q_t(b_t^m; \psi^{(j)})}{p(b_t^m \mid \mathbf{b}_{-t}^m, \mathbf{x}; \psi^{(j)})\, q_t(B_t^*; \psi^{(j)})} = \frac{p(\mathbf{x} \mid B_t^*, \mathbf{b}_{-t}^m; \theta^{(j)})\, w^{(j)}[k_t^*]\, q_t(b_t^m; \mathbf{w}^{(j)})}{p(\mathbf{x} \mid b_t^m, \mathbf{b}_{-t}^m; \theta^{(j)})\, w^{(j)}[k_t^m]\, q_t(B_t^*; \mathbf{w}^{(j)})}, \tag{21}$$

where k_t^* and k_t^m are the states associated with B_t^* and b_t^m, respectively. Note that we use

$$q_t(b_t; \mathbf{w}^{(j)}) = \begin{cases} c_t^{-1}\mathbf{w}^{(j)} & \text{if } b_t > x_t, \\ 0 & \text{if } b_t \leq x_t, \end{cases} \tag{22}$$

where c_t is the normalization constant, and therefore (21) simplifies to

$$\frac{\prod_{l=t}^{t+p} p(x_l \mid \mathcal{F}_{l-1}, B_t^*, \mathbf{b}_{-t}^m; \theta^{(j)})}{\prod_{l=t}^{t+p} p(x_l \mid \mathcal{F}_{l-1}, b_t^m, \mathbf{b}_{-t}^m; \theta^{(j)})}. \tag{23}$$

The algorithm to get M samples $\mathbf{b}^m$ distributed according to $p(\mathbf{b}|\mathbf{x}; \psi^{(j)})$ is given in Appendix A. It includes m_0 burn-in iterations, which give samples that are discarded so that the chain has time to forget its initial state $\mathbf{b}^0 = (1, \ldots, 1)$ before we start to keep samples. As suggested by Geyer (1992), we set m_0 to between 1% and 2% of M.

To the best of our knowledge, there are no theoretical results for MCEM outside of exponential families, moreover when replacing the M-step by CM-steps. Nevertheless, we point out that the condition for the ECM to share convergence properties with EM is again fulfilled, replacing μ with $\Lambda = (\lambda_1, \ldots, \lambda_p)$. We do not propose innovative methods in automating the algorithm, see, e.g., Booth and Hobert (1999); McCulloch (1994, 1997); Caffo, Jank and Jones (2005), regarding the number M of samples and the design of stopping rules, as we choose to rather focus on an online version of the algorithm. Therefore, the MCECM algorithm we propose uses a fixed size M and a predetermined number of iterations J. Let $\mathbf{y}^m = (y_t^m)_{t=p+1,\ldots,(T-p)}$ where $y_t^m = \gamma(x_t/b_t^m; \nu)$ and $\mathbf{Y}^m$ be the matrix of stacked vectors $[B\mathbf{y}^m, \ldots, B^p\mathbf{y}^m]$, where B is the backshift operator. The corresponding MCECM algorithm is described in Algorithm 3.

Algorithm 3 MCECM algorithm (stochastic processes with AR dependency)

Input: #Iterations J, Monte Carlo size M, initial guess ψ_0, $\epsilon > 0$

for $j = 0, \ldots, J$ **do**

1. **MCE-Step**

Sample: $\mathbf{b}^1, \ldots, \mathbf{b}^M \sim p(\mathbf{b} \mid \mathbf{x}; \psi^{(j)})$ using Algorithm 7.

2. **CM-Steps**

Update:

$$w^{(j+1)}[k] = \frac{1}{M(T-2p)} \sum_{m=1}^{M} \sum_{t=p+1}^{T-p} \mathbb{1}_{\{k_t^m = k\}} \quad \forall\, k = 1, \ldots, K, \tag{24}$$

$$\Lambda^{(j+1)} = \left(\sum_{m=1}^{M} (\mathbf{Y}^m)^\top \mathbf{Y}^m \right)^\top \left(\sum_{m=1}^{M} (\mathbf{Y}^m)^\top \mathbf{y}^m \right), \tag{25}$$

$$\sigma^{2(j+1)} = \frac{1}{M(T-2p)} \sum_{m=1}^{M} \left(\mathbf{y}^m - \mathbf{Y}^m \Lambda^{(j+1)} \right)^\top \left(\mathbf{y}^m - \mathbf{Y}^m \Lambda^{(j+1)} \right). \tag{26}$$

Compute: Newton step $\Delta\nu_{nt}$ and decrement λ^2.

if $\lambda^2/2 \leq \epsilon$ **then** $\nu^{(j+1)} = \nu^{(j)}$

else Choose step size η by backtracking line search and Update: $\nu^{(j+1)} = \nu^{(j)} + \eta\Delta\nu_{nt}$.

end if

end for

4.2. *Online EM algorithm for inference on stochastic processes with serial correlations.* An MCEM algorithm is computationally expensive as a rather large M might be required to get unbiased MLE. Moreover, when applying the method to stochastic processes we are especially interested in non-stationary frameworks, and would therefore wish to be able to update the parameters of the distribution without needing to rerun a heavy batch algorithm.

As in Section 3, we expect to update the current value of the parameter vector ψ_t, where the subscript now refers to time, or iteration t, by accounting for the information brought by time t only. In particular, we wish to forget the past missing values of B_t. Let $\mathcal{K}_t = \{k \in \mathcal{K} \mid b[k] > x_t\}$. In terms of the function Q, this translates to observing at time t

$$-Q_t(\psi|\psi_t) = -\sum_{k\in\mathcal{K}_t} \hat{w}_t[k](\log p(x_t \mid \mathcal{F}_{t-1}, b[k]; \theta) + \log w[k]), \tag{27}$$

where the posterior probabilities are

$$\hat{w}_t[k] = \frac{p(x_t \mid \mathcal{F}_{t-1}, b[k]; \theta_t)\ w_t[k]}{\sum_{l=1}^{K} p(x_t \mid \mathcal{F}_{t-1}, b[l]; \theta_t)\ w_t[l]}. \tag{28}$$

The key here lies in $p(x_t \mid \mathcal{F}_{t-1}, b[k]; \theta_t)$, for we do not want the probability of observing x_t to still be impacted by the missing values $\mathbf{b}_{(t-1):(t-p)}$.

For the computation of the posterior probabilities, integration over possible values of $\mathbf{b}_{(t-1):(t-p)}$ is easy and limited to $\mathcal{K}_{(t-1):(t-p)} = \mathcal{K}_{t-1} \times \cdots \times \mathcal{K}_{t-p}$, since the conditional pdf of x_t is equal to zero for values of $\mathbf{b}_{(t-1):(t-p)}$ outside of $\mathcal{K}_{(t-1):(t-p)}$. Let k_{t-r} be the state corresponding to the value of B_{t-r} in combinations $\mathbf{b}_{(t-1):(t-p)}$, and $\hat{w}_{t-r}[k_{t-r}]$ be the corresponding posterior probability computed at time $t-r$ according to (28). We approximate $p(x_t \mid \mathcal{F}_{t-1}, b[k_t]; \theta_t)$ in (28) with

$$\sum_{\mathbf{b}_{(t-1):(t-p)} \in \mathcal{K}_{(t-1):(t-p)}} p(x_t \mid \mathcal{F}_{t-1}, b[k], \mathbf{b}_{(t-1):(t-p)}; \theta_t)\ \hat{w}_{t-1}[k_{t-1}]\ \ldots\ \hat{w}_{t-p}[k_{t-p}]. \tag{29}$$

The approximation lies in using $P(\mathbf{b}_{(t-1):(t-p)} \mid \mathcal{F}_{t-1}; \theta_t) \approx \hat{w}_{t-1}[k_{t-1}]\ \ldots\ \hat{w}_{t-p}[k_{t-p}]$. Therefore, we consider the successive posterior probabilities of b_{t-r} given x_{t-r} only, rather than given the whole filtration $\mathcal{F}_{t-r}$. Moreover, we assume $\psi_t \approx \psi_{t-1} \approx \cdots \approx \psi_{t-p}$.

Integration cannot be considered in (27) because of the logarithm. We propose two ways of dealing with the past missing values of B_t. Let $\hat{\mathbf{w}}_t = (\hat{w}_t[1], \ldots, \hat{w}_t[K])$. As a first option, we replace the past missing values $\mathbf{b}_{(t-1):(t-p)}$ with filtered values $\hat{\mathbf{b}}_{(t-1):(t-p)}$, such as

$$\hat{b}_{t-r} = \sum_{k=1}^{K} \hat{\mathbf{w}}_{t-r}[k]\ b[k], \quad r = 1, \ldots, p, \tag{30}$$

i.e., each missing value b_{t-r} is replaced with its conditional expectation on the observation x_{t-r} and the current parameter ψ_{t-r}. The corresponding online EM algorithm is described in Algorithm 4.

As a second option, we instead simulate a single value of $\mathbf{b}_{(t-1):(t-p)}$ according to the posterior probabilities $\hat{\mathbf{w}}_{(t-1):(t-p)}$. This is closer in spirit to historical stochastic EM algorithms, see, e.g., Celeux, Chauveau and Diebolt (1996); Marschner (2001). The corresponding online EM algorithm is described in Algorithm 5.

The same ideas as in Section 3.2 apply for intuitively explaining the convergence and tracking properties of both versions of the online algorithm for stochastic processes. However, because we only approximate or simulate past missing values of B_t, the online EM algorithm can only rely on partial information from the past in the context of stochastic processes with AR dependency.

5. Simulation study.

5.1. *Convergence.* In order to illustrate the performance of batch and online algorithms on stationary data, we run 100 experiments with $K = 3$, $b[1] = 0.5$, $b[2] = 0.75$ and $b[3] = 1$. The parameters are set to $\mathbf{w} = (0.2, 0.3, 0.5)$, $\mu = 1$ under the independence assumption, $\Lambda = 0.9$ for stochastic processes, $\sigma^2 = 2$ and $\nu = 1.5$. Each synthetic dataset is of size $15,000$. The values $b[1], \ldots, b[K]$ and the lag p of the AR dependency for stochastic processes are assumed to be known.

Algorithm 4 Online EM for stochastic processes with AR dependency (filtering)

Input: convex set $\mathcal{K}, \psi_1 \in \mathcal{K}$, mini-batch size $m \geq 1$, step size η

for $t = 1, \ldots, p$ **do**

Compute $\hat{\mathbf{w}}_t = q_t(b_t; \mathbf{w}_t)$;

Compute $\hat{b}_t = (b[1], \ldots, b[K])\hat{\mathbf{w}}_t$;

$\psi_{t+1} = \psi_t$.

end for

for $t = p+1, \ldots, T$ **do**

Compute $\hat{w}_t[k] \propto p(x_t \mid \mathcal{F}_{t-1}, b[k]; \theta_t)\ w_t[k]$;

For next iteration, compute $\hat{b}_t = (b[1], \ldots, b[K])\hat{\mathbf{w}}_t$;

Observe

$$-Q_t(\psi|\psi_t) = -\sum_{k \in \mathcal{K}_t} \hat{w}_t[k](\log p(x_t \mid \mathcal{F}_{t-1}, b[k], \hat{\mathbf{b}}_{(t-1):(t-p)}; \theta) + \log w[k]) \tag{31}$$

Update and project:

$$\xi_{t+1} = \psi_t + \eta \nabla_\psi Q_t(\psi|\psi_t), \tag{32}$$

$$\psi_{t+1} = \Pi_{\mathcal{K}}(\xi_{t+1}). \tag{33}$$

end for

Algorithm 5 Online EM for stochastic processes with AR dependency (sampling)

Input: convex set $\mathcal{K}, \psi_1 \in \mathcal{K}$, mini-batch size $m \geq 1$, step size η

for $t = 1, \ldots, p$ **do**

Compute $\hat{\mathbf{w}}_t = q_t(b_t; \mathbf{w}_t)$;

Compute $\hat{b}_t = (b[1], \ldots, b[K])\hat{\mathbf{w}}_t$;

$\psi_{t+1} = \psi_t$.

end for

for $t = p+1, \ldots, T$ **do**

Compute $\hat{w}_t[k] \propto p(x_t \mid \mathcal{F}_{t-1}, b[k]; \theta_t)\ w_t[k]$;

Sample $\mathbf{b}_{(t-1):(t-p)} \sim \hat{\mathbf{w}}_{(t-1):(t-p)}$;

Observe

$$-Q_t(\psi|\psi_t) = -\sum_{k \in \mathcal{K}_t} \hat{w}_t[k](\log p(x_t \mid \mathcal{F}_{t-1}, b[k], \mathbf{b}_{(t-1):(t-p)}; \theta) + \log w[k]). \tag{34}$$

Update and project:

$$\xi_{t+1} = \psi_t + \eta \nabla_\psi Q_t(\psi|\psi_t), \tag{35}$$

$$\psi_{t+1} = \Pi_{\mathcal{K}}(\xi_{t+1}). \tag{36}$$

end for

5.1.1. *Under the independence assumption.* Algorithm 1, i.e., ECM for independent data, is run with a stopping rule on the variation of the parameters: convergence is claimed when the relative change in the parameter values from successive iterations is small. The initial values are set to $\psi_0 = (1/3, 1/3, 1/3, 0, 1, 1)$. Boxplots of the MLE we obtain for the parameter vector ψ are available in Figure 1. On average, the algorithm required about 150 iterations.

Online Algorithm 2 requires to tune the hyper-parameter η, i.e., the step size of the gradient descent step in (15). We set it to 0.01 by looking at the first simulation only. The initial values of the parameters are the same ones as for ECM. Boxplots of the MLE we obtain for the parameter vector ψ are available in Figure 2. The estimates reported are the last estimated values over the sample, i.e., the estimates after seeing the last observation. The online

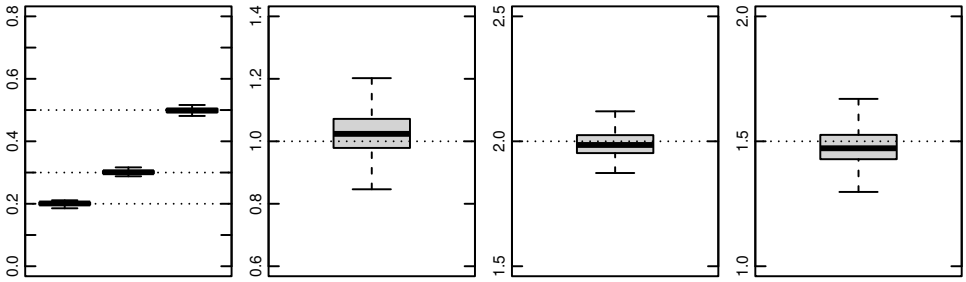


FIG 1. *Maximum likelihood estimates of* $\mathbf{w}$, μ, σ^2 *and* ν, *from left to right, for Algorithm 1.*

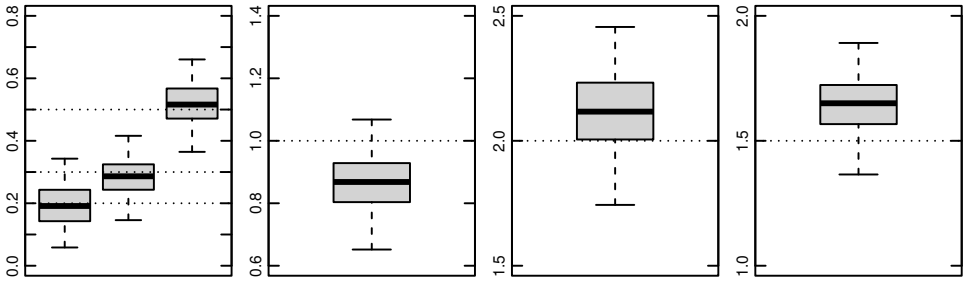


FIG 2. *Maximum likelihood estimates of* $\mathbf{w}$, μ, σ^2 *and* ν, *from left to right, for Algorithm 2 with* $\eta = 0.01$.

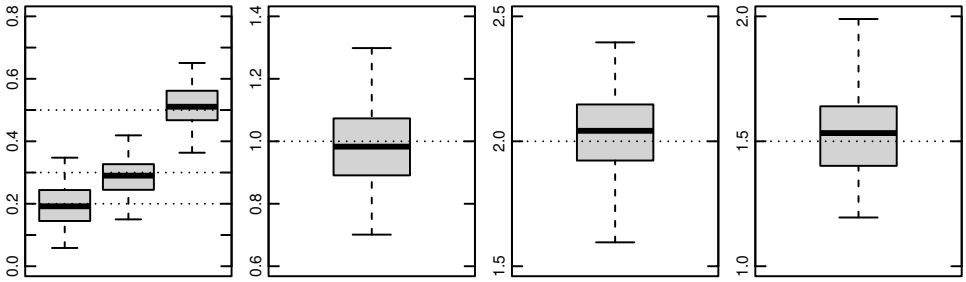


FIG 3. *Maximum likelihood estimates of* $\mathbf{w}$, μ, σ^2 *and* ν, *from left to right, for Algorithm 2 initialized with ECM estimates and* $\eta = 0.01$.

algorithm converges to slightly biased values on average, in particular for the parameters of the GLN distribution. Confidence intervals of the tracked parameters along with the average estimates are available in Figure 11 of Appendix B. For convergence purposes, the step size of an online algorithm is often set to decrease. Because our primary interest lies in tracking, we do not focus here on improving convergence, but Algorithm 2 could benefit from methods which aim to better tune the step size. This might involve Polyak-Ruppert averaging technique as a post-processing step, see, e.g., Ruppert (1988); Polyak and Juditsky (1992); Cappé and Moulines (2009); Bach and Moulines (2011). The bias can also be reduced by first applying Algorithm 1 on a small batch of data, say 500 observations, and then running the online algorithm 2 starting from the ECM estimates, as shown in Figure 3.

5.1.2. *For stochastic processes.* Algorithm 3, i.e., MCECM for stochastic processes with AR dependency, is run for a maximum number of 20 iterations. The size of the Monte Carlo

sample M is set to 100 for the first 10 iterations and 200 for the next 10 iterations, upon looking at the convergence behaviour of the algorithm on the first simulation of the study only. The initial values are set to $\psi_0 = (1/3, 1/3, 1/3, 0, 1, 1)$. Boxplots of the MLE we obtain for the parameter vector ψ are available in Figure 4. The estimates of the scale and shape parameters of the GLN distribution are slightly biased and can be improved by increasing M, although it might be computationally expensive.

The step size of both online versions of the EM algorithm are set to 0.003 by looking at the first simulation again. The initial values of the parameters are the same ones as for MCECM. Boxplots of the MLE we obtain for the parameter vector ψ are available in Figure 5 for the filtering version, and in Figure 6 for the stochastic version. Both online algorithms converge to biased estimates of the GLN parameters. On these synthetic data, no version seems to outperform the other. Confidence intervals of the tracked parameters along with the average estimates are available in Figures 12 and 13 of Appendix B. The filtering version converges to a slightly less biased estimate of σ^2, but a slightly more biased estimate of ν. We believe the

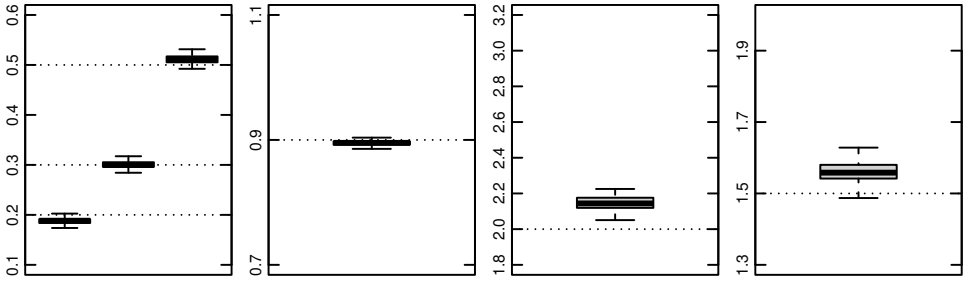


FIG 4. *Maximum likelihood estimates of* $\mathbf{w}$, Λ, σ^2 *and* ν, *from left to right, for Algorithm 3.*

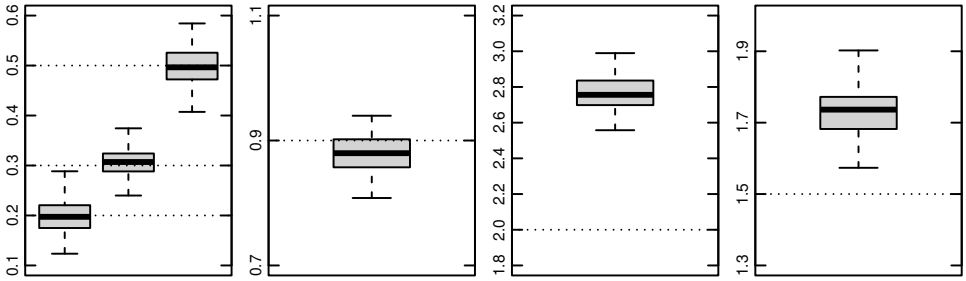


FIG 5. *Maximum likelihood estimates of* $\mathbf{w}$, Λ, σ^2 *and* ν, *from left to right, for Algorithm 4 with* $\eta = 0.003$.

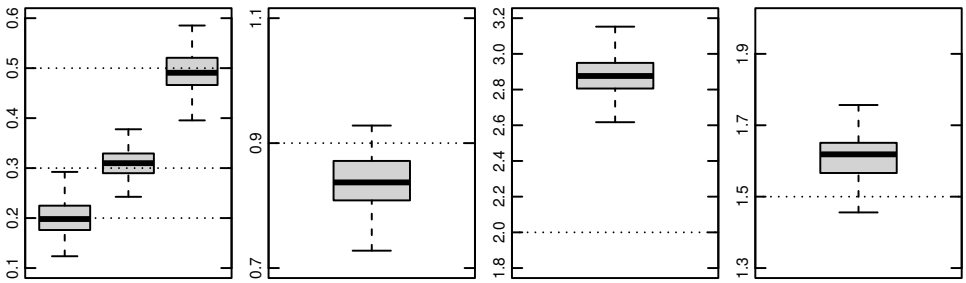


FIG 6. *Maximum likelihood estimates of* $\mathbf{w}$, Λ, σ^2 *and* ν, *from left to right, for Algorithm 5 with* $\eta = 0.003$.

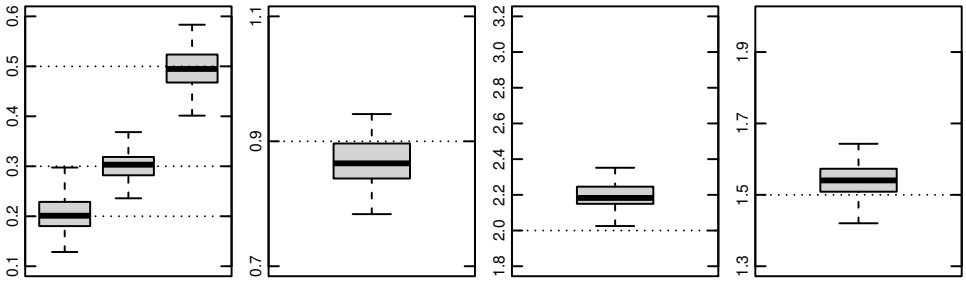


FIG 7. *Maximum likelihood estimates of* $\mathbf{w}$, Λ, σ^2 *and* ν, *from left to right, for an online EM algorithm with access to the past and* $\eta = 0.003$.

bias mostly comes from approximating the past values of B_t through filtering or simulation. For comparison purposes, we provide boxplots of an equivalent online algorithm which can access the past values of B_t in Figure 7.

5.2. *Tracking.* We apply the online algorithms to non-stationary data, in particular when the distribution of the bound slowly or suddenly changes, to illustrate their tracking abilities. In the independent case, ECM, i.e. Algorithm 1, is first run on the first 500 observations. Then, Algorithm 2 is run with $\eta = 0.01$ for both smooth and sharp simulations, starting from the estimates obtained by Algorithm 1. Confidence intervals of the tracked parameters are available in Figure 8 for $\mathbf{w}$ and in Figure 14 of Appendix B for the GLN distribution.

In the case of stochastic processes, both Algorithms 4 and 5 are run with $\eta = 0.003$. Confidence intervals of the tracked parameters are available in Figures 9 and 10 for $\mathbf{w}$. Confidence intervals for the GLN parameters are not shown as very similar to Figures 12 and 13 of Appendix B.

All online algorithms perform well in tracking the changing distribution of the bound. As in Section 5.1, the decisions about the hyper-parameters were made upon looking at the first simulation of each setting. Choosing an appropriate step size for tracking is also difficult: if η is too large, the statistical fluctuations around ψ_t may be important, while the algorithm will lose its tracking ability for too small values of η. This is a typical bias-variance trade-off, a tracking/accuracy compromise in the context of adaptive algorithms, see, e.g., Benveniste, Métivier and Priouret (1990).

5.3. *Forecasting.* As a first look into forecasting, we issue probabilistic forecasts from the parameters tracked with Algorithms 4 and 5, in the case of stochastic processes with a sharp change in the distribution of the bound. To obtain the forecasts at each time t for time $t+1$, we first sample the past missing value of the bound according to the posterior distribution $\hat{\mathbf{w}}_t$. Then, samples are drawn from the normal distribution $\mathcal{N}(\mu_{t+1}, \sigma_t^2)$ and transformed back to the interval $(0, 1)$ using the inverse transformation γ^{-1} with ν_t. Finally, the next value of the bound is sampled according to the marginal distribution $\mathbf{w}_t$. The first step of the forecasting procedure prevents us from issuing predictive densities and we issue probabilistic forecasts for X_{t+1} in terms of a sample.

We evaluate these probabilistic forecasts on the last 10,000 observations by computing the continuous ranked probability score (CRPS), which is a strictly proper scoring rule relative to the class of the Borel probability measures that have finite first moment, see Gneiting and Raftery (2007). As a comparison, we consider the probabilistic forecasts issued by the following forecasters:

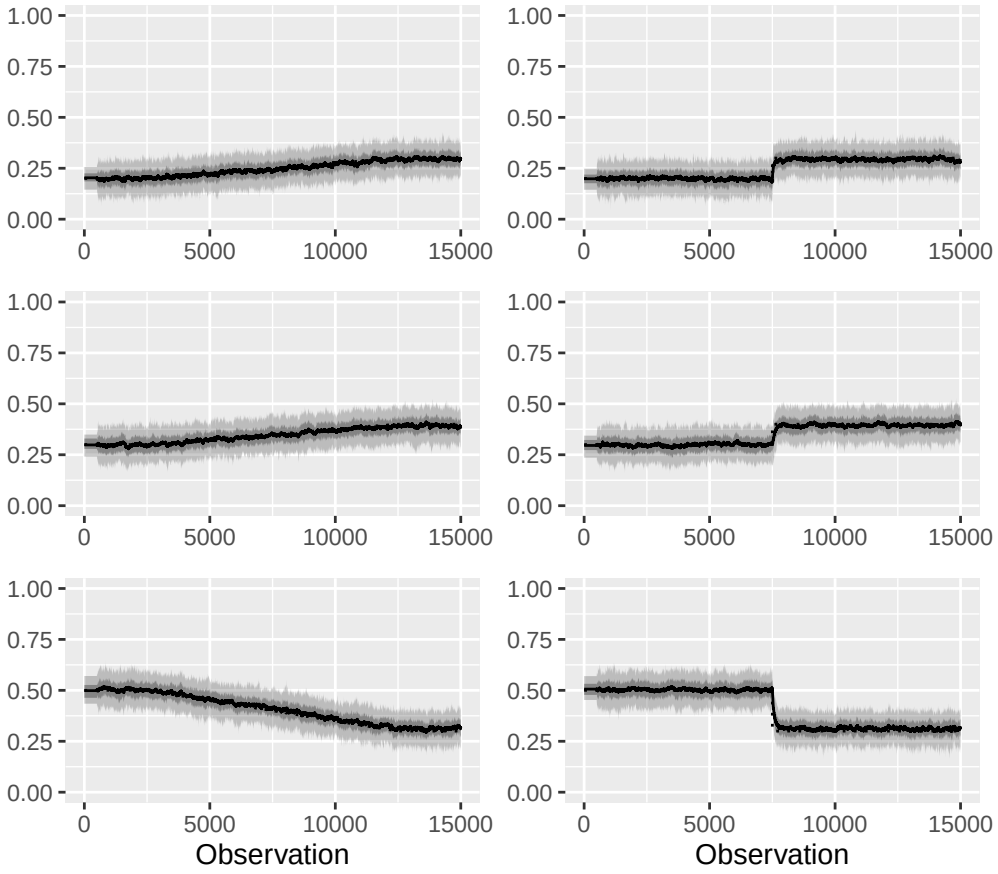


FIG 8. *Confidence intervals of the tracked parameter* $\mathbf{w}$ *for Algorithm 2 (independent data), in the case of a smooth change (left) and a sharp change (right), with coverage probabilities 0.9 and 0.5, along with the average estimates (solid lines) and the true parameters (dotted lines).*

- Oracle: the oracle observes the true realizations $b_1, \ldots, b_t$ and knows the future b_{t+1} at each forecasting time t; they also know the true parameter θ_t at each time t.
- Omniscient: the omniscient forecaster observes the true realizations $b_1, \ldots, b_t$ but does not know the future b_{t+1}; however, they know the true parameter ψ_t at each time t.
- Climatology: at each forecasting time t, climatology forecasts the empirical distribution of the observations until time t.
- Probabilistic persistence: at each forecasting time t, probabilistic persistence forecasts x_t dressed with the past observed errors.
- GLN: at each forecasting time t, the GLN forecaster issues a GLN predictive density as in Equation (7), where $b_1 = \cdots = b_t = 1$, and whose parameter vector θ has been estimated on the first 5,000 observations.

The average CRPS over the 100 synthetic datasets are presented in Table 1 for each forecaster, along with the standard deviations and three quantiles. The omniscient forecaster is the one which predicts the true mixture distribution at each time t. It is far from the oracle and shows the impact of having to forecast the bound on the overall forecasting performance. The CRPS achieved when using the parameters tracked with Algorithms 4 and 5 are relatively close to the CRPS of the omniscient forecaster, in particular compared to the gap between the

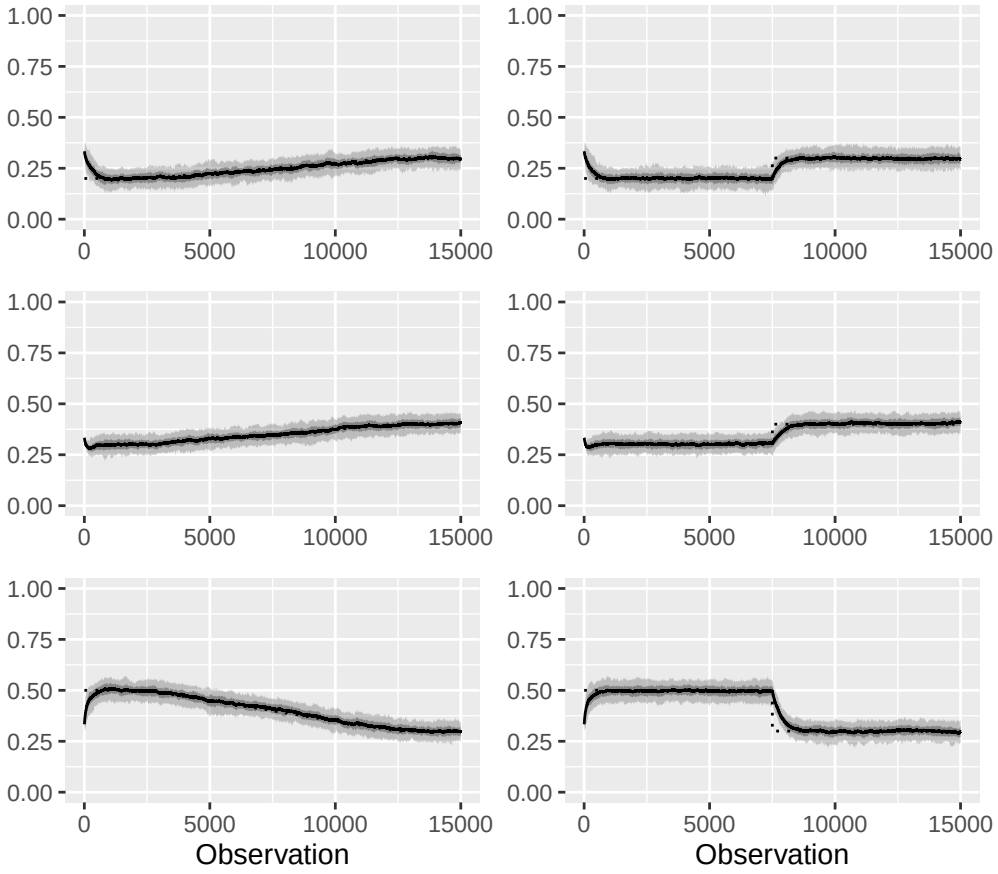


FIG 9. *Confidence intervals of the tracked parameter* $\mathbf{w}$ *for Algorithm 4 (stochastic processes), in the case of a smooth change (left) and a sharp change (right), with coverage probabilities 0.9 and 0.5, along with the average estimates (solid lines) and the true parameters (dotted lines).*

TABLE 1
CRPS for each forecaster

			Quantile (%)		
Forecaster	Mean (%)	Std. dev.	2.5%	50%	97.5%
Oracle	6.37	0.12	6.16	6.36	6.66
Omniscient	9.84	0.15	9.54	9.84	10.07
Climatology	17.43	0.15	17.11	17.43	17.72
Probabilistic persistence	12.24	0.18	11.87	12.24	12.59
GLN	11.73	0.18	11.36	11.74	12.04
Algorithm 4	10.26	0.15	9.95	10.26	10.50
Algorithm 5	10.30	0.15	10.00	10.31	10.54

performance of the oracle and the performance of the omniscient forecaster. Hence, it seems we do not loose too much when moving to forecasting by not observing the past values of the bound. Note that on these synthetic processes, the online algorithm which uses past expected values of the bound performs slightly better than the one which relies on the simulation of a single value. As for other forecasters, to use a fixed single GLN distribution still provides

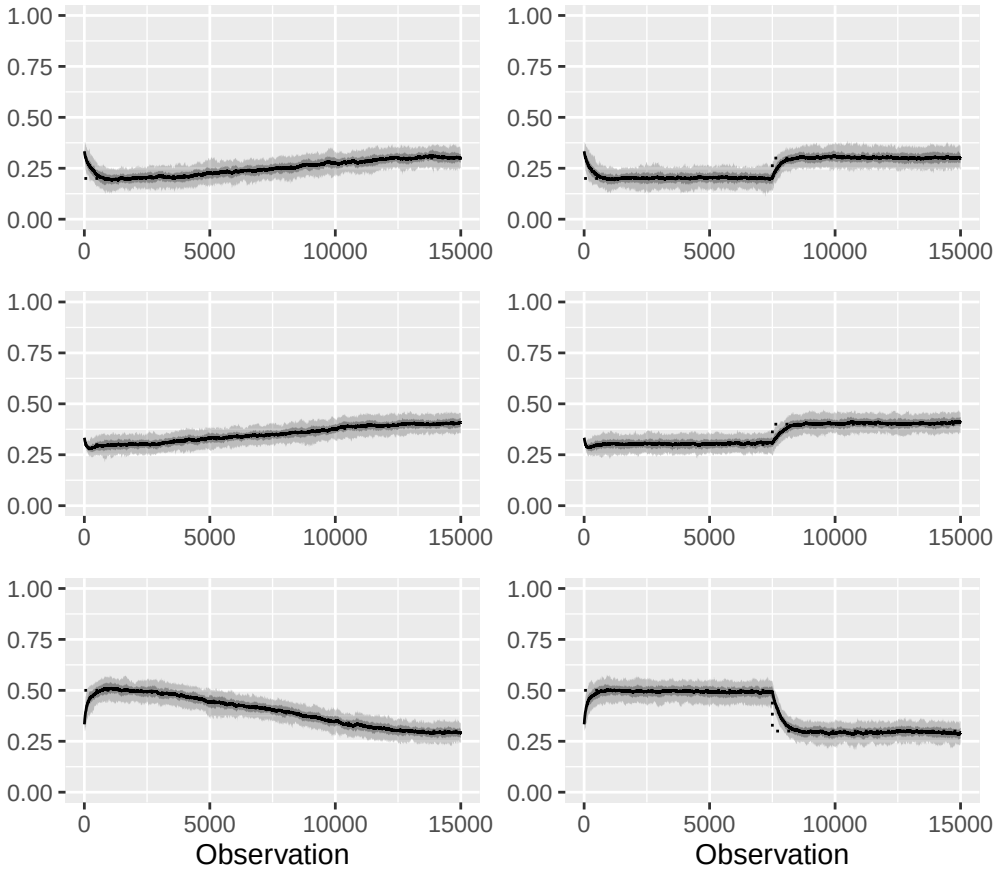


FIG 10. *Confidence intervals of the tracked parameter* $\mathbf{w}$ *for Algorithm 5 (stochastic processes), in the case of a smooth change (left) and a sharp change (right), with coverage probabilities 0.9 and 0.5, along with the average estimates (solid lines) and the true parameters (dotted lines).*

improvement compared to probabilistic persistence, while climatology shows a much larger CRPS on average.

Lastly, note that we have chosen this framework where $K = 3$ and all $w[k]$ are relatively far from 1 and 0 to illustrate the performance of the method and the designed algorithms above all, and because simulating more complicated synthetic data is rather challenging. This implies the results achieved here in terms of forecasting are mostly sanity check results. Our wish is that for real datasets, more differentiated weights will be put on more likely values of B_t, for a finer grid $b[1], \ldots, b[K]$, while these probabilities will change over time, making the method especially interesting for forecasting purposes.

6. Conclusion. We have introduced a new framework where the upper bound of a continuous response variable is a discrete latent variable. The corresponding statistical model is derived in the case of independent and identically distributed bounded variables, and in the case of bounded stochastic processes with AR dependency. We propose to perform maximum likelihood inference using EM-based algorithms: an expectation-conditional maximization (ECM) algorithm for independent data; a Monte Carlo expectation-conditional maximization (MCECM) algorithm for stochastic processes with AR dependency. We ran a simulation study which demonstrated the convergence of both batch algorithms. However, the MCECM

algorithm would benefit from automating the choice of an appropriate size for the Monte Carlo sample.

We wish to design online algorithms which enable the tracking of the model parameters in changing environments, partly because of the simplicity of the distribution we have assumed for the bound. While all online algorithms show good tracking abilities, the online EM algorithms for stochastic processes failed to recover unbiased estimates of the GLN distribution parameters. We believe this largely has to do with the approximation that is performed at the E-step to replace the missing past values of the bound. Additional simulation studies are required to evaluate the impact on the bias of an increase in both the number of components in the mixture and the lag of the AR dependency. Alternative or complementary approaches need to be considered to reduce this bias. Lastly, the assumptions made when computing the posterior probabilities of the bound at each time t should be further investigated as well. Nevertheless, the performance achieved by the online algorithms when moving to forecasting are not so far from the ideal forecaster and rather encouraging.

APPENDIX A: ADDITIONAL ALGORITHMS

Algorithm 6 Projection on the probability simplex $\mathbf{P}$

Input: $\tilde{\mathbf{w}} \in \mathbb{R}^K$
  if $\tilde{\mathbf{w}} \in \mathbf{P}$ **then** $\mathbf{w} = \tilde{\mathbf{w}}$
  else
    Sort $\tilde{w}_{(1)} \geq \tilde{w}_{(2)} \geq \cdots \geq \tilde{w}_{(K)}$
    Find $K_0 = \max \left\{ 1 \leq i \leq K; \tilde{w}_{(i)} - \frac{\sum_{j=1}^{i} \tilde{w}_{(j)} - 1}{i} > 0 \right\}$
    Define $\lambda^* = \dfrac{\sum_{j=1}^{K_0} \tilde{w}_{(j)} - 1}{K_0}$ and **Set** $w[k] = \max(0, \tilde{w}[k] - \lambda^*)$
  end if
Return: $\mathbf{w} \in \mathbf{P}$

Algorithm 7 Single-component Metropolis-Hastings with independence sampler

Input: #Monte Carlo samples M, burn-in parameter m_0, $\mathbf{b}^0 = (1, \ldots, 1)$
  for $m = 1, \ldots, M + m_0$ **do**
    Set $\mathbf{b}^m \leftarrow \mathbf{b}^{m-1}$
    for $t = p + 1, \ldots, T - p$ **do**
      Sample $B_t^* \sim q_t(b; \mathbf{w}^{(j)})$;
      Sample a uniform random variable $U \sim U(0, 1)$;
      if $U < \min \left(1, \dfrac{\prod_{l=t}^{t+p} p(x_l \mid \mathcal{F}_{l-1}, B_t^*, \mathbf{b}_{-t}^m; \theta^{(j)})}{\prod_{l=t}^{t+p} p(x_l \mid \mathcal{F}_{l-1}, b_t^m, \mathbf{b}_{-t}^m; \theta^{(j)})} \right)$ **then** $b_t^m \leftarrow B_t^*$
      else b_t^m remains unchanged
      end if
    end for
  end for
Return: $(\mathbf{b}^{m_0+1}, \ldots, \mathbf{b}^{m_0+M})$

APPENDIX B: ADDITIONAL FIGURES OF THE SIMULATION STUDY

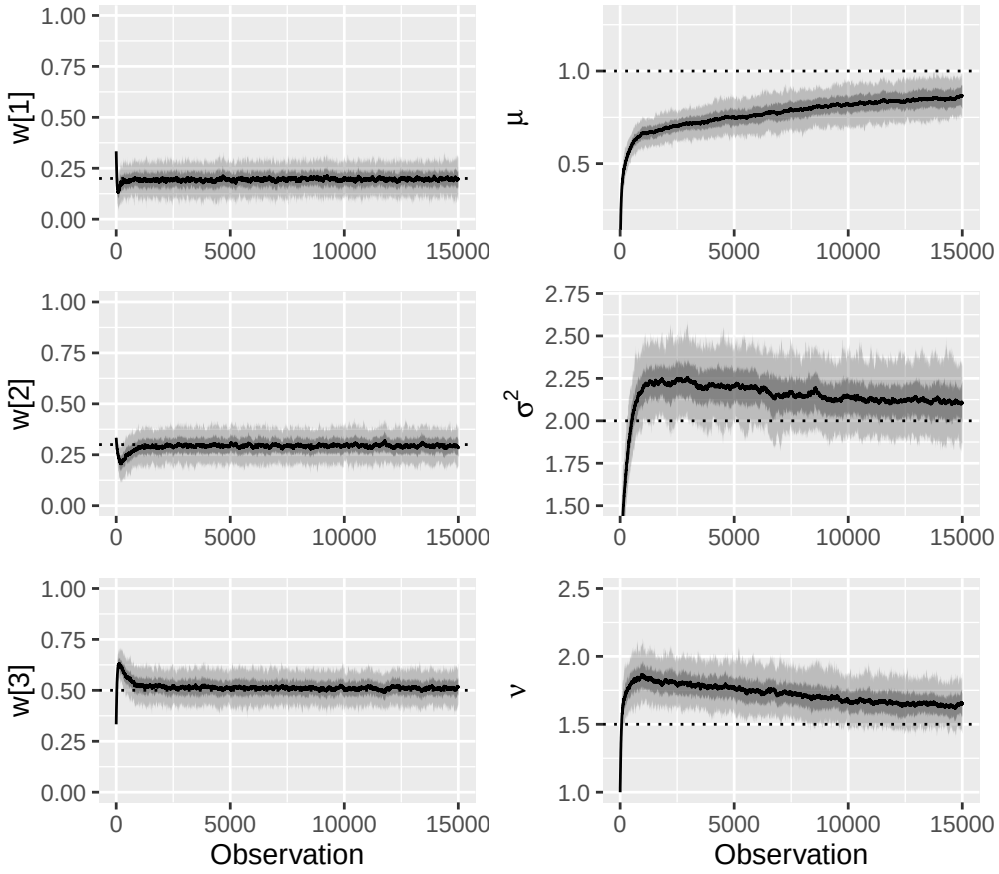


FIG 11. *Confidence intervals of the tracked parameters with coverage probabilities 0.9 and 0.5, along with the average estimates (solid lines) and the true parameters (dotted lines), for Algorithm 2 with* $\eta = 0.01$.

REFERENCES

BACH, F. and MOULINES, E. (2011). Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems* **24**.

BENVENISTE, A., MÉTIVIER, M. and PRIOURET, P. (1990). *Adaptive Algorithms and Stochastic Approximations*. Springer Berlin, Heidelberg.

BOOTH, J. G. and HOBERT, J. P. (1999). Maximizing generalized linear mixed model Likelihoods with an automated Monte Carlo EM algorithm. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* **61** 265–285.

BOTTOU, L. (1999). *On-line Learning and Stochastic Approximations* In *On-Line Learning in Neural Networks* 9–42. Cambridge University Press.

BOYD, S. and VANDENBERGHE, L. (2004). *Convex Optimization*. Cambridge University Press.

CAFFO, B. S., JANK, W. and JONES, G. L. (2005). Ascent-based Monte Carlo expectation-maximization. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* **67** 235–251.

CAPPÉ, O. (2011). Online EM algorithm for hidden Markov models. *Journal of Computational and Graphical Statistics* **20** 728-749.

CAPPÉ, O. and MOULINES, E. (2009). On-line expectation-maximization algorithm for latent data models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* **71** 593–613.

CELEUX, G., CHAUVEAU, D. and DIEBOLT, J. (1996). Stochastic versions of the EM algorithm: An experimental study in the mixture case. *Journal of Statistical Computation and Simulation* **55** 287-314.

CESA-BIANCHI, N., CONCONI, A. and GENTILE, C. (2004). On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory* **50** 2050-2057.

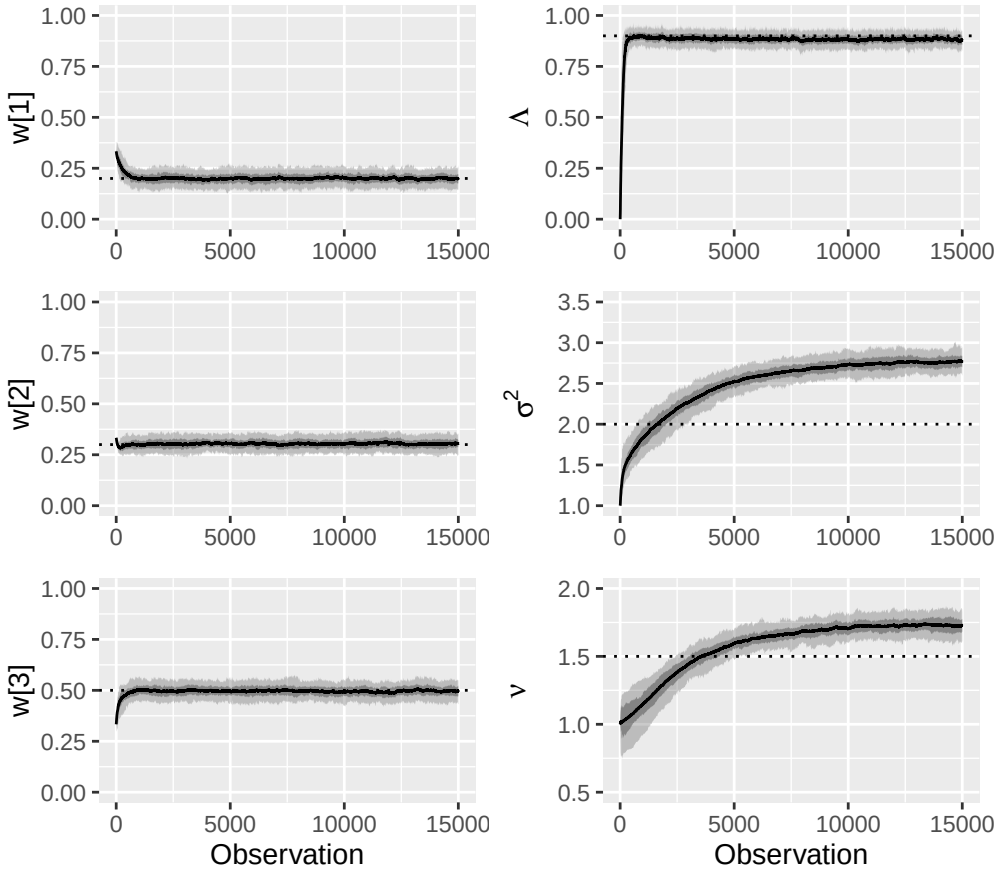


FIG 12. *Confidence intervals of the tracked parameters with coverage probabilities 0.9 and 0.5, along with the average estimates (solid lines) and the true parameters (dotted lines), for Algorithm 4 with* $\eta = 0.003$.

CESA-BIANCHI, N. and LUGOSI, G. (2006). *Prediction, Learning, and Games*. Cambridge University Press.

CORFF, S. L. and FORT, G. (2013). Online expectation maximization based algorithms for inference in hidden Markov models. *Electronic Journal of Statistics* **7** 763 – 792.

DEMPSTER, A. P., LAIRD, N. M. and RUBIN, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* **39** 1–38.

GEYER, C. J. (1992). Practical Markov chain Monte Carlo. *Statistical Science* **7** 473–483.

GILKS, W. R., RICHARDSON, S. and SPIEGELHALTER, D. J. (1995). *Markov Chain Monte Carlo in Practice*. Springer New York, NY.

GNEITING, T. and RAFTERY, A. E. (2007). Strictly proper scoring rules, prediction and estimation. *Journal of the American Statistical Association* **102** 359–378.

GUOLO, A. and VARIN, C. (2014). Beta regression for time series analysis of bounded Data, with application to Canada Google® flu trends. *The Annals of Applied Statistics* **8** 74–88.

HASTINGS, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57** 97–109.

JENSEN, S. T., JOHANSEN, S. and LAURITZEN, S. L. (1991). Globally convergent algorithms for maximizing likelihood function. *Biometrika* **78** 867–877.

JOHNSON, N. L. (1949). Systems of frequency curves generated by methods of translation. *Biometrika* **36** 149–176.

LADERMAN, J. and LITTAUER, S. B. (1953). The inventoryProblem. *Journal of the American Statistical Association* **48** 717–732.

MARSCHNER, I. C. (2001). On stochastic versions of the EM algorithm. *Biometrika* **88** 281–286.

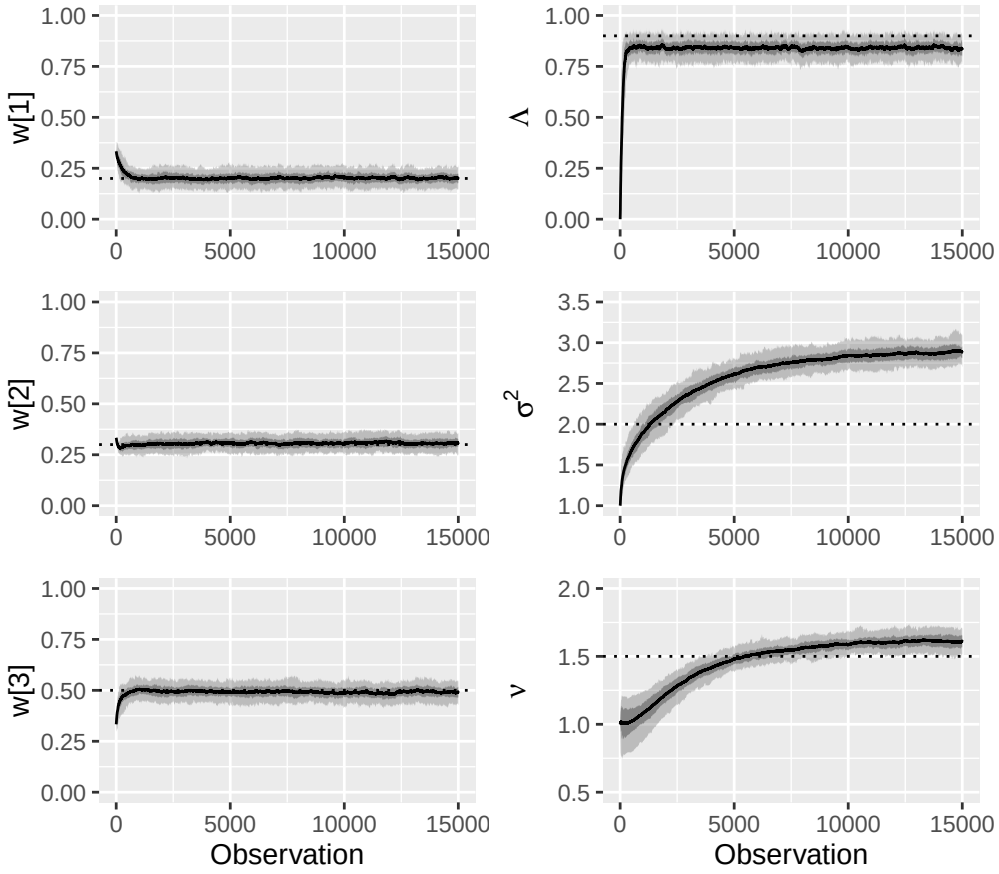


FIG 13. *Confidence intervals of the tracked parameters with coverage probabilities 0.9 and 0.5, along with the average estimates (solid lines) and the true parameters (dotted lines), for Algorithm 5 with* $\eta = 0.003$.

MCCULLOCH, C. E. (1994). Maximum likelihood variance components estimation for binary data. *Journal of the American Statistical Association* **89** 330–335.

MCCULLOCH, C. E. (1997). Maximum likelihood algorithms for generalized linear mixed models. *Journal of the American Statistical Association* **92** 162–170.

MEAD, R. (1965). A Generalised Logit-Normal Distribution. *Biometrics* **21** 721–732.

MENG, X.-L. and RUBIN, D. B. (1993). Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika* **80** 267–278.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H. and TELLER, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics* **21** 1087-1092.

NEAL, R. M. and HINTON, G. E. (1998). *A view of the EM algorithm that justifies incremental, sparse, and other variants* In *Learning in Graphical Models* 355–368. Springer Netherlands.

ORABONA, F. (2022). A modern introduction to online learning.

PIERROT, A. and PINSON, P. (2023). On tracking varying bounds when forecasting bounded time series.

POLYAK, B. T. and JUDITSKY, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization* **30** 838–855.

ROBERT, C. P. and CASELLA, G. (2004). *Monte Carlo Statistical Methods*. Springer New York, NY.

RUPPERT, D. (1988). Efficient estimations from a slowly convergent Robbins-Monro process. Technical Report, Cornell University Operations Research and Industrial Engineering.

TITTERINGTON, D. M. (1984). Recursive parameter estimation using incomplete data. *Journal of the Royal Statistical Society. Series B (Methodological)* **46** 257–267.

WALLIS, K. F. (1987). Time series analysis of bounded economic variables. *Journal of Time Series Analysis* **8** 115–123.

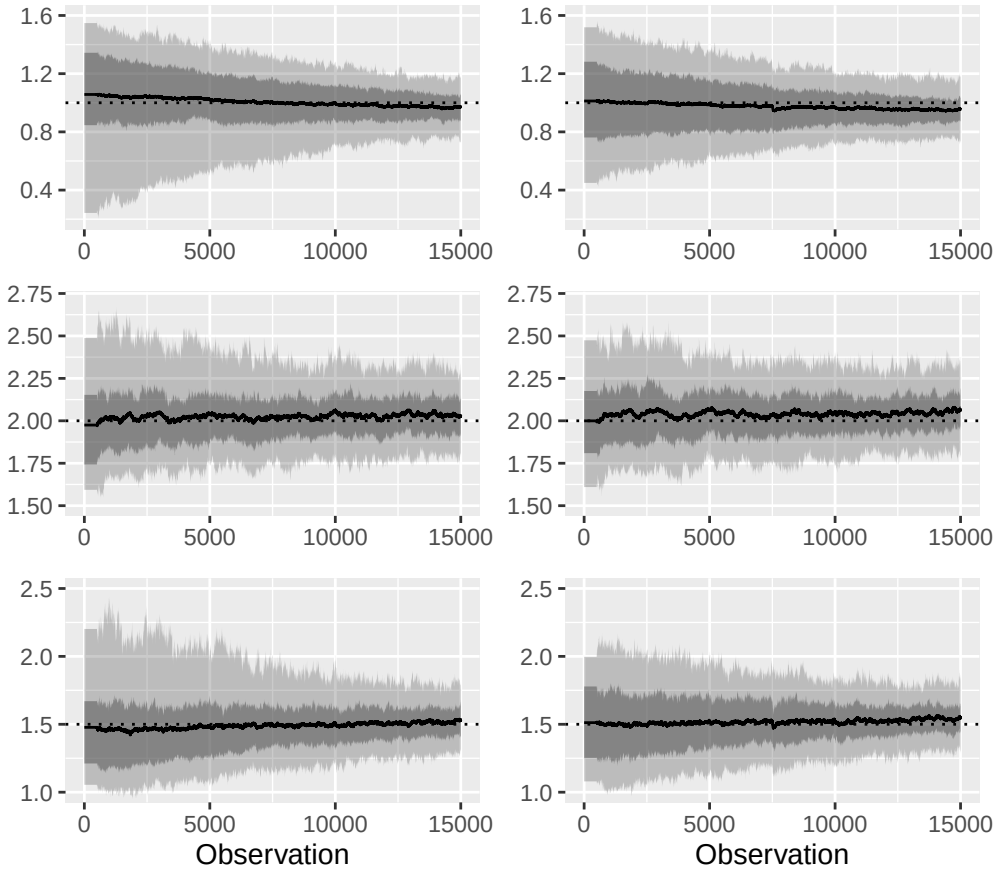


FIG 14. *Confidence intervals of the tracked parameters μ, σ^2 and ν, from top to bottom, for Algorithm 2 (independent data), in the case of a smooth change (left) and a sharp change (right) with coverage probabilities 0.9 and 0.5, along with the average estimates (solid lines) and the true parameters (dotted lines).*

WEI, G. C. G. and TANNER, M. A. (1990). A Monte Carlo implementation of the EM algorithm and the Poor Man's data augmentation algorithms. *Journal of the American Statistical Association* **85** 699–704.

ZAMO, M., MESTRE, O., ARBOGAST, P. and PANNEKOUCKE, O. (2014a). A benchmark of statistical regression methods for short-term forecasting of photovoltaic electricity production, part I: Deterministic forecast of hourly production. *Solar Energy* **105** 792-803.

ZAMO, M., MESTRE, O., ARBOGAST, P. and PANNEKOUCKE, O. (2014b). A benchmark of statistical regression methods for short-term forecasting of photovoltaic electricity production. Part II: Probabilistic forecast of daily production. *Solar Energy* **105** 804-816.

ZANGWILL, W. I. (1969). *Nonlinear Programming - A Unified Approach*. Prentice-Hall.

[Paper D] Data is missing again – Reconstruction of power generation data using k-nearest neighbors and spectral graph theory

Authors:
Amandine Pierrot and Pierre Pinson

Submitted to:
Wind Energy

Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

Data is missing again – Reconstruction of power generation data using k-Nearest Neighbors and spectral graph theory

Amandine Pierrot[1] | **Pierre Pinson**[2,3]

[1]Department of Wind and Energy Systems, Technical University of Denmark, Denmark

[2]Dyson School of Design Engineering, Imperial College London, United Kingdom

[3]Department of Technology, Management and Economics, Technical University of Denmark, Denmark

Correspondence
Pierre Pinson, Dyson School of Design Engineering, Imperial College London, UK.
Email: p.pinson@imperial.ac.uk

Abstract

The risk of missing data and subsequent incomplete data records at wind farms increases with the number of turbines and sensors. We propose here an imputation method that blends data-driven concepts with expert knowledge, by using the geometry of the wind farm in order to provide better estimates when performing nearest neighbors imputation. Our method relies on learning Laplacian eigenmaps out of the graph of the wind farm through spectral graph theory. These learned representations can be based on the wind farm layout only, or additionally account for information provided by collected data. The related weighted graph is allowed to change with time and can be tracked in an online fashion. Application to the Westermost Rough offshore wind farm shows significant improvement over approaches that do not account for the wind farm layout information.

KEYWORDS
missing data, time series, Laplacian eigenmaps, Nadaraya-Watson estimators, wind power forecasting

1 | INTRODUCTION

According to the International Energy Agency, overall wind power generation increased by a record 17% in 2021. Of the total 830 GW installed 93% were still onshore systems, as onshore wind is a developed technology while offshore wind is still at the early stage of expansion. However offshore reach is expected to increase in the coming years as more countries are developing or planning to develop their first offshore wind farms. From the world's first offshore wind farm, Vindeby in Denmark, which totalled 11 turbines in 1991, the size of offshore wind farms has increased up to more than a hundred wind turbines nowadays, e.g., Hornsea 1 in the United Kingdom which totals 174 wind turbines. While data recorded by wind turbines are of great value for wind farm and system operators, they are subject to information loss from, e.g., power and communication failures, instrumentation issues or human error. Missing data in wind farm time series can impact revenue[1], wind energy resource assessment[2], wind farm control[3] or the estimation of power curves[4]. In particular they negatively impact forecasting models which for short-term lead times (from a few minutes to a few hours ahead) are better be statistical models trained on historical data and/or online learning methods which require the most recent observed data[5]. For offshore wind power forecasting, short-term fluctuations in power generation are most significant, calling for very-short-term forecasting models and methods. Because of the increasing number of turbines in offshore wind farms, the issue of missing data gets even more critical. Let T be the total number of records over a wind farm, measured at successive time steps $t = 1, \dots, T$ (usually spaced at uniform intervals). Now assume a data point is missing for a wind turbine at time t with probability 0.01, independently from other wind turbines. With a number of wind turbines $N = 11$, this would result in about 90% of the T records being complete, i.e., data points are available for all N wind turbines. With $N = 174$ wind turbines, the proportion of complete records drops to 17% and the workaround which consists in assuming data completeness and deleting records with missing entries is not sensible anymore[6].

Alternatives remain for dealing with increasing missing data. One is to develop methods where the assumption of data completeness is not needed anymore. In the context of time series, works exist that make assumptions about the missing data

Abbreviations: NN, nearest neighbors; AR, autoregressive; EM, expectation-maximization; MCAR, missing completely at random; MAR, missing at random; MNAR, missing not at random; OGD, online gradient descent; RMSE, root mean square error.

patterns[7], or need not even make any assumptions[8], and estimate AR models. Other works develop models which are robust to missing data[9]. Another alternative is to provide imputations for missing values, i.e., to replace the data points which are missing with plausible values. Classical statistical imputation methods use maximum likelihood estimators which correspond to a specific underlying model. A very popular approach for dealing with missing data in time series is the EM algorithm[10] which relies on two steps: at the E-step missing values are filled in with their conditional expectation given the observed data and the current estimate of the model parameter vector, and at the M-step a new estimate of the parameter vector is computed from the current version of the completed data. This procedure requires assumptions on the distributions of both observed and missing data. A widely used[11], yet controversial[12], nomenclature for missing value mechanisms distinguishes between three cases: MCAR, MAR and MNAR. In MCAR the probability of a data point being missing is completely independent of any variables in the dataset while in MAR the probability of being missing depends only on observed values. These first two mechanisms are considered the simple ones in the sense that they do not make it necessary to model explicitly the distribution of the values which are missing when maximizing the likelihood of the observations. The third mechanism is the harder yet prevalent one, as the probability of a point being missing is dependent on the value it would have taken. This leads to important biases in the remaining data whose distribution is not the true distribution anymore.

Missing value imputation is appealing as it makes it possible to first get a completed dataset and then to apply any statistical learning algorithm which relies on the completeness assumption. However it covers a wide range of situations where it might be more or less legitimate to handle the problem of missing values that way. This has to do with the missing value mechanism, which one can hardly be sure of, but also with the task to be performed on the completed dataset. For supervised learning tasks such as forecasting, theoretical and empirical results outline simple practical recommendations[13]. In particular, that to train and test on data with missing values, the same imputation model should be used. Empirically, that good imputation methods reduce the number of samples required to reach good prediction. Also, that when missingness is related to the prediction target, imputation is not sufficient and it is useful to add indicator variables of missing entries as features. Last but not least it is shown that a predictor suited for complete observations can predict optimally on incomplete data through multiple imputation, which consists in generating multiple plausible values for each missing value. When the supervised learning algorithm is of the regression kind, with a regression function which can be nonlinear, almost all imputations lead asymptotically to the optimal prediction with a powerful learner, no matter the missing value mechanism. This result gives theoretical grounding to all impute-then-regress procedures. However a good choice of imputation can reduce the complexity of the regression function to be learned and therefore it is suggested that learning imputation and regression jointly is easier[14,15].

In the context of offshore wind farms, we deal with multivariate time series, as we record N data points, one for each wind turbine, at each time step t. This opens a new range of methods for missing data imputation, as one can exploit information from another (potentially correlated) sensor, in our case another wind turbine, when one is missing. Recently, several deep learning approaches have been proposed for multivariate time series imputation[16,17,18]. When interested in the average production of a wind farm, it is quite intuitive to work with the average of the individual production values from the wind turbines which are available at time t. By doing so one implicitly performs k-NN imputation. The k-Nearest Neighbors algorithm is a seminal nonparametric method in machine learning[19,20]. In a nutshell, it uses the k points which are the closest to a point of interest to make a decision about it. In k-NN imputation, we consider the k nearest neighbours of a missing point to provide an estimate of its value[21]. The assumptions associated with this imputation method are very weak: we do not assume any model generating the data, observed or missing, and only assume similar groups of observations. Moreover the method applies for all missing data mechanisms. In section 2 we explicit how working with a quantity of interest averaged over n_t available records at each time t comes down to (unweighted) n_t-NN imputation. We propose to improve this implicit imputation by moving from unweighted to weighted n_t-NN imputation through Nadaraya-Waston estimators. Each neighbor will now enter the k-NN algorithm with a different weight, hopefully the closer the higher. A higher weight for a closer neighbor means we are able to measure how close with an appropriate distance. We show how to use graph spectral theory to compute Laplacian eigenmaps, i.e., new representations of the wind farm as a graph which take into account local and global geometries. We consider the case when we only use the structure of the wind farm for learning its representation and the case when we also use values of the quantity we wish to perform imputation for. The method is illustrated on the Westermost Rough offshore wind farm and results for the imputation of power generation missing values are presented in section 3. Finally we provide some conclusions and perspectives in section 4.

2 | IMPUTATION USING NEAREST NEIGHBORS AND GRAPHS

2.1 | Weighted nearest neighbors imputation

Let X_t be an average quantity of interest over a wind farm at time t. We have

$$X_t = \frac{1}{N}\sum_{i=1}^{N} X_t^i, \tag{1}$$

where X_t^i is the quantity of interest for the i-th wind turbine at time t, N is the total number of wind turbines in the wind farm. When some X_t^is are missing, assume instead we work with

$$\hat{X}_t = \frac{1}{n_t}\sum_{j=1}^{n_t} X_t^{(j)}, \tag{2}$$

where $X_t^{(j)}$ is the quantity of interest for the (j)-th available wind turbine record, n_t being the number of available wind turbine records, at time t. Imputation with k-NN consists in filling in a missing value using the values from its k-nearest neighbors. Unweighted k-NN give the same weight to each neighbor, when weighted k-NN assign a higher weight to a closer neighbor. Replace each missing value $X_t^{(l)}$ with its unweighted n_t-NN estimate $\hat{X}_t^{(l)} = \frac{1}{n_t}\sum_{j=1}^{n_t} X_t^{(j)}$, where $X_t^{(j)}$ are the n_t values available at time t. We have

$$\hat{X}_t = \frac{1}{N}\left(X_t^{(1)} + \cdots + X_t^{(n_t)} + \hat{X}_t^{(n_t+1)} + \cdots + \hat{X}_t^{(N)}\right), \tag{3a}$$

$$= \frac{1}{N}\sum_{j=1}^{n_t} X_t^{(j)} + \frac{1}{N}\sum_{l=n_t+1}^{N} \hat{X}_t^{(l)}, \tag{3b}$$

$$= \frac{1}{N}\sum_{j=1}^{n_t} X_t^{(j)} + \frac{1}{N}\sum_{l=n_t+1}^{N} \frac{1}{n_t}\sum_{j=1}^{n_t} X_t^{(j)}, \tag{3c}$$

$$= \frac{1}{n_t}\sum_{j=1}^{n_t} X_t^{(j)}. \tag{3d}$$

We see that to work with $\hat{X}_t$ in (2) is equivalent to filling in the missing values $(X_t^{(l)})_{l=n_{t+1},\ldots,N}$ using unweighted n_t-NN estimates, i.e., where each neighbor is assigned the same weight. However a quantity of interest at a wind turbine level is likely to be more similar to the same quantity from the actual neighbors of this wind turbine, i.e., the wind turbines which are located nearby in the wind farm. Staying in the k-NN framework, we can improve our estimates through the number of neighbors k, the weights put on neighbors, or both. Theoretical results about k-NN mostly concern the asymptotic mode, when n_t tends to infinity, which cannot be assumed here as we are limited by the number of wind turbines in the wind farm. It is rather critical to choose k in a finite regime, and it is usually advised to perform cross-validation. This would be cumbersome here as cross-validation would need to be run for each combination of available data points, for each missing data point, and would require quite an amount of complete data. Therefore we suggest to keep $k = n_t$ at each time t and rather improve the weights of the n_t-NN imputation.

Learning the distance metric for k-NN has been extensively studied and it was found that metric learning may significantly affect the performance of the method in many applications. Because we perform imputation at each time step t considering values from similar sensors at the same time step t, the Euclidean distance seems a fair enough metric in our framework. Instead we focus on a common shortcoming in current nonparametric methods, which is to only consider the distances between the decision point and its neighbors and ignore the geometrical relation between those neighbors. Indeed, before we even get any records from its sensors, a wind farm is a graph with its own geometry which provides a priori information not only on the distance between a wind turbine and its neighbors, but also between these neighbors.

Moving to weighted n_t-NN imputation we wish to provide each wind turbine (l) which misses a record with a better estimate by weighting the available records (j) according to their proximity to the wind turbine, while acknowledging the whole structure of the wind farm. In order to do so we need to be able to assign weights depending on the distance between the wind turbine for which a record is missing and the wind turbines with available records. We choose to use Nadaraya-Watson estimators[22,23], which assign weights that are proportional to some given similarity kernel K. More optimal methods could be used[24], which we

will discuss later. Let K be a given nonnegative measurable function on $\mathbb{R}$ (the kernel), h be a positive number (the bandwith) depending upon n_t only and $\|\mathbf{z}^{(j)} - \mathbf{z}^{(l)}\|$ be the Euclidean distance between the representations of two wind turbines (j) and (l). In case (l) is missing and (j) is available at time t, the weight we give to $X_t^{(j)}$ when computing a weighted estimate $\hat{X}_t^{(l)} = \sum_{j=1}^{n_t} w^{(jl)} X_t^{(j)}$ of $X_t^{(l)}$ is

$$w^{(jl)} = \frac{K\left(\frac{\|\mathbf{z}^{(j)} - \mathbf{z}^{(l)}\|}{h}\right)}{\sum_{i=1}^{n_t} K\left(\frac{\|\mathbf{z}^{(i)} - \mathbf{z}^{(l)}\|}{h}\right)}. \tag{4}$$

Let us sort the neighbors of a wind turbine (l) by increasing distance, $\|\mathbf{z}^{(1)} - \mathbf{z}^{(l)}\| \leq \|\mathbf{z}^{(2)} - \mathbf{z}^{(l)}\| \leq \cdots \leq \|\mathbf{z}^{(n_t)} - \mathbf{z}^{(l)}\|$. We choose an adaptive bandwith $h_t = \|\mathbf{z}^{(n_t)} - \mathbf{z}^{(l)}\|$, so that the weights adjust depending on n_t, i.e., depending on the availability of other records at time t. Note that if all distances at play at time t are very similar, the estimator will be very close to the unweighted one, which seems legit. Consider the so-called naive kernel $K(\mathbf{u}) = \mathbb{1}_{\{\|\mathbf{u}\| \leq 1\}}$. With such a choice for h_t, to use a naive kernel is to use our former estimate $\frac{1}{n_t} \sum_{j=1}^{n_t} X_t^{(j)}$. Therefore from now on we will refer to this estimate as the Nadaraya-Watson estimator with a naive kernel. For a more general kernel the weight $w_t^{(jl)}$ depends on the distance $\|\mathbf{z}^{(j)} - \mathbf{z}^{(l)}\|$ through the kernel shape. We will consider the following ones:

- Gaussian kernel $K(\mathbf{u}) = e^{-\|\mathbf{u}\|^2}$,
- Epanechnikov kernel $K(\mathbf{u}) = (1 - \|\mathbf{u}\|^2)\, \mathbb{1}_{\{\|\mathbf{u}\| \leq 1\}}$,
- triangular kernel $K(\mathbf{u}) = (1 - \|\mathbf{u}\|)\, \mathbb{1}_{\{\|\mathbf{u}\| \leq 1\}}$,
- quartic kernel $K(\mathbf{u}) = (1 - \|\mathbf{u}\|^2)^2\, \mathbb{1}_{\{\|\mathbf{u}\| \leq 1\}}$,
- triweight kernel $K(\mathbf{u}) = (1 - \|\mathbf{u}\|^2)^3\, \mathbb{1}_{\{\|\mathbf{u}\| \leq 1\}}$,
- tricube kernel $K(\mathbf{u}) = (1 - \|\mathbf{u}\|^3)^3\, \mathbb{1}_{\{\|\mathbf{u}\| \leq 1\}}$.

Note that only the Gaussian kernel gives a positive weight to the furthest neighbor (or neighbors) (n_t) of (l), while all other kernels put a zero weight.

2.2 | Wind farms as unweighted graphs

2.2.1 | Graphs

A graph G is defined by a set of nodes - or vertices - $V = v_1, \ldots, v_N$ and a set of edges E between nodes. It is said to be undirected if there is no direction implied by an edge. Often when a vertex v_i represents a data point x_i, two vertices v_i and v_j are connected if x_i and x_j are close. Let the wind farm be a graph $G = (V, E)$ where the set of nodes V are the wind turbines, $|V| = N$, and the set of edges E connecting two nodes are to be decided upon. We base our graph on the layout of the wind farm, without considering any data points x_i.

We start with an unweighted graph, i.e., each edge between two nodes is unweighted. Let $\mathbf{A} = (a_{ij})_{i,j=1,\ldots,N}$ be the *adjacency* matrix of the graph G. Unweighted edges (or simple-minded weights) means that $a_{ij} = 1$ if vertices v_i and v_j are connected by an edge, $a_{ij} = 0$ otherwise. Therefore all the edges are assumed to have the same strength. Note that the diagonal of $\mathbf{A}$ is equal to zero, i.e. $a_{ii} = 0$, as we do not consider self-connections. Through the adjacency matrix $\mathbf{A}$ each wind turbine is represented by the vector of size N of its connections to the other wind turbines in the wind farm. Out of this representation we wish to learn a low-dimensional embedding for each wind turbine that preserves the structure of the wind farm.

2.2.2 | Laplacian eigenmaps

We are interested in spectral graph embeddings, and in particular in Laplacian eigenmaps, which optimally preserve local neighborhood information and produce coordinate maps that are smooth functions over the original graph[25]. By trying to preserve local information in the embedding, the algorithm implicitly emphasizes the natural clusters in the data. Close connections to spectral clustering then become very clear[26]. We hope for the Laplacian eigenmaps to provide a smooth clustering of the wind turbines over the wind farm. Let $\mathbf{D}$ be the diagonal matrix associated with the graph G whose entries are the degree of each node, i.e., $d_{ii} = \sum_j a_{ji}$. The matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is called the Laplacian matrix of the graph and one gets eigenmaps by computing

eigenvalues and eigenvectors for the generalized eigenvector problem

$$\mathbf{Lf} = \lambda \mathbf{Df}. \tag{5}$$

Let $\mathbf{f}_0, \ldots, \mathbf{f}_{N-1}$ be the solutions of Equation (5), ordered according to their eigenvalues

$$\mathbf{Lf}_0 = \lambda_0 \mathbf{Df}_0 \tag{6a}$$
$$\mathbf{Lf}_1 = \lambda_1 \mathbf{Df}_1 \tag{6b}$$
$$\ldots \tag{6c}$$
$$\mathbf{Lf}_{N-1} = \lambda_{N-1} \mathbf{Df}_{N-1} \tag{6d}$$
$$0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{N-1}. \tag{6e}$$

We leave out the constant eigenvector $\mathbf{f}_0$ corresponding to eigenvalue 0 and use the next r eigenvectors for embedding each wind turbine v_i in a r-dimensional Euclidean space:

$$\mathbf{z}^i = (\mathbf{f}_1(v_i), \ldots, \mathbf{f}_r(v_i)). \tag{7}$$

The embedding $\mathbf{z}^i$ is a new representation of the wind turbine v_i. Distances $\|\mathbf{z}^i - \mathbf{z}^j\|$ for every pair of wind turbines (v_i, v_j) can now be computed once and for all but the number of components we keep in $\mathbf{z}^i$ needs to be decided upon. For each missing data point the kernel function will then adjust the weights at each time t depending on the set of n_t available data points from other wind turbines.

2.3 | Wind farms as weighted graphs

2.3.1 | Weighting of the original graph

The representations we get out of the unweighted graph embed the structure of the wind farm only and can be used without having any other data but the map of the wind farm. The unweighted graph can be seen as a stationary a priori part of the relationships between the wind turbines, which comes from the location of the wind turbines inside the wind farm and could be completed with an online part coming from the time series we are interested in. Therefore we propose to keep the structure of the unweighted graph G and move from unweighted to weighted edges. Say we are interested in X_t^i the power generation of wind turbine v_i at time t normalized by the nominal capacity of the wind turbine. Therefore we have $X_t^i \in [0, 1]$ for $t = 1, \ldots, T$, $i = 1, \ldots, N$. An edge between two wind turbines shall be weighted according to the similarity of these wind turbines. Working with power generation this translates to the similarity between the power generations of these wind turbines at time t, as we are interested in an online similarity. Let x_t^i, resp. x_t^j, be the observed power generation of wind turbine v_i, resp. v_j, at time t. A simple and intuitive choice for the similarity between x_t^i and x_t^j is $s_t(i,j) = 1 - \|x_t^i - x_t^j\|$, $s_t(i,j) \in [0, 1]$. Note that if it happens at time t that two wind turbines which are not connected in graph G have very similar power generation values, they remain unconnected. By doing so we enforce an a priori on the relationship between the wind turbines upon the structure of the wind farm, but we also keep a sparse adjacency matrix $\mathbf{A}$. Indeed the adjacency matrix $\mathbf{A}$ now depends on t and we have $\mathbf{A}_t = (a_{t,ij})_{i,j=1,\ldots,N}$ where $a_{t,ij} = s_t(i,j)$ if vertices v_i and v_j are connected by an edge, $a_{t,ij} = 0$ otherwise. This implies to solve the generalized eigenproblem (5) at each time step t, which can be done rather easily when dealing with sparse matrices $\mathbf{A}_t$.

2.3.2 | Online, changing graphs

Because of the way G is now weighted, it can happen that two wind turbines which are a priori connected get disconnected because $s_t(i,j) = 0$ for some time t. In such a case we can get a graph which is not *connected* anymore, i.e., we cannot travel through the whole graph from any point in the graph. To compute eigenmaps the generalized eigenproblem (5) must be solved for a connected graph, which ensures $\text{rank}(\mathbf{D}) = N$ and there are N eigenvalues, or $\lambda(\mathbf{L}, \mathbf{D})$ may be finite, empty or infinite[27]. Therefore when a graph has several components the algorithm for computing eigenmaps consists in solving the generalized eigenproblem (5) for each connected component, which we will do for components with at least three wind turbines. When a wind turbine splits from the graph on its own, it is straightforward to derive its production value from its neighbors as typically

a wind turbine v_i gets disconnected at time t because $x_t^i = 0$ and $x_t^j = 1$ for all its neighbors v_j. When two wind turbines get disconnected together, the similarity between the two is usually high enough to replace the record which is missing with the one which is available.

Since we are online and in high dimension, we need to be able to automatically detect when the graph is not connected anymore, and what are the components. Let $\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$ be the so-called normalized graph Laplacian, which is the graph Laplacian we use to compute eigenmaps. We recall a basic yet very useful property of this graph Laplacian[26], that makes it easy to derive the connected components of G at time t if the multiplicity of the eigenvalue 0 of $\mathbf{L}_{rw,t}$ becomes higher than 1. Let G be an undirected graph with non-negative weights. The multiplicity d of the eigenvalue 0 of the graph Laplacian $\mathbf{L}_{rw}$ equals to the number of connected components $A_1, \dots, A_d$ in the graph and the eigenspace of 0 is spanned by the indicator vectors $\mathbb{1}_{A_i}$ of those components.

Last but not least, so far we have assumed the similarities $s_t(i,j)$ to be known. Because our application is the imputation of missing values, the true distances are actually known at time t among the wind turbines for which data points are available, but they are not for the edges involving wind turbines for which the record is missing and we need to replace them with estimates. To avoid having to specify a model for the similarities between all individual time series, we place ourselves in the online learning framework. The goal in this learning paradigm is to guess a sequence of numbers as precisely as possible, when the data are chosen by an adversary rather than generated stochastically[28,29]. In our framework this translates as the following repeated game: in each round $t = 1, \dots, T$, for each similarity between two connected wind turbines $s_t(i,j)$, an adversary chooses a real number in $[0, 1]$ and keeps it secret; we try to guess the real number, choosing $\hat{s}_t(i,j)$; the adversary number is revealed and we pay the squared difference $(s_t(i,j) - \hat{s}_t(i,j))^2$. Online learning is appealing from both a theoretical and practical point of view because a lot of problems can be described as such a repeated game, which does not require strong assumptions to offer nice theoretical guarantees. One shall note that the last step of the repeated game when the adversary number is revealed does not happen if some of the data are missing. Therefore to account for the possibility of missing data the game is slightly modified and we pay $(s_t(i,j) - \hat{s}_t(i,j))^2$ only if $s_t(i,j)$ is revealed[8]. Using the notation $\mathbb{1}_{\{s_t(i,j)\}}$ as the indicator of the event $\{s_t(i,j) \text{ is revealed}\}$, we pay now $(s_t(i,j) - \hat{s}_t(i,j))^2 \mathbb{1}_{\{s_t(i,j)\}}$. In such a missing data, convex, framework, an online strategy with good theoretical guarantees is the lazy version of OGD[30] which is applied to our problem in Algorithm 1, where $\Pi_{[0,1]}$ is the projection back to $[0, 1]$. We also consider the best constant strategy, i.e, the strategy that minimizes $\sum_{t=1}^{T} (s_t(i,j) - \hat{s}_t(i,j))^2 \mathbb{1}_{\{s_t(i,j)\}}$, which is just constantly choosing $\hat{s}_t(i,j)$ to be the average similarity $\sum_{t=1}^{T} s_t(i,j)\mathbb{1}_{\{s_t(i,j)\}} \Big/ \sum_{t=1}^{T} \mathbb{1}_{\{s_t(i,j)\}}$ at each round.

Algorithm 1

Input: learning rate η

Set $\hat{s}_1(i,j) = y_1 = 1$.

for $t = 1, \dots, T$ **do**

Play $\hat{s}_t(i,j)$ and occur cost $f_t(\hat{s}_t) = (s_t(i,j) - \hat{s}_t(i,j))^2 \mathbb{1}_{\{s_t(i,j)\}}$.

Update and project:

$$y_{t+1} = y_t - \eta \nabla f_t(\hat{s}_t)$$
$$\hat{s}_{t+1}(i,j) = \Pi_{[0,1]}(y_{t+1})$$

end for

3 | APPLICATION AND CASE-STUDY: WESTERMOST ROUGH

We apply the method presented in section 2 to a real use case, the Westermost Rough offshore wind farm. Westermost Rough is located close to the Eastern coast of the United Kingdom and totals 35 wind turbines which are placed according to a grid pattern. A representation of the wind farm through the position and name of its wind turbines is available in Figure 1. The pattern is rather usual and the number of wind turbines high enough to support our method, but not too high for us to deliver a detailed and visual analysis. Through this example we wish to deliver good practices and to emphasize challenges which generalize to bigger and/or more complicated wind farms. Along with the exact position of the wind turbines, we have data records over two

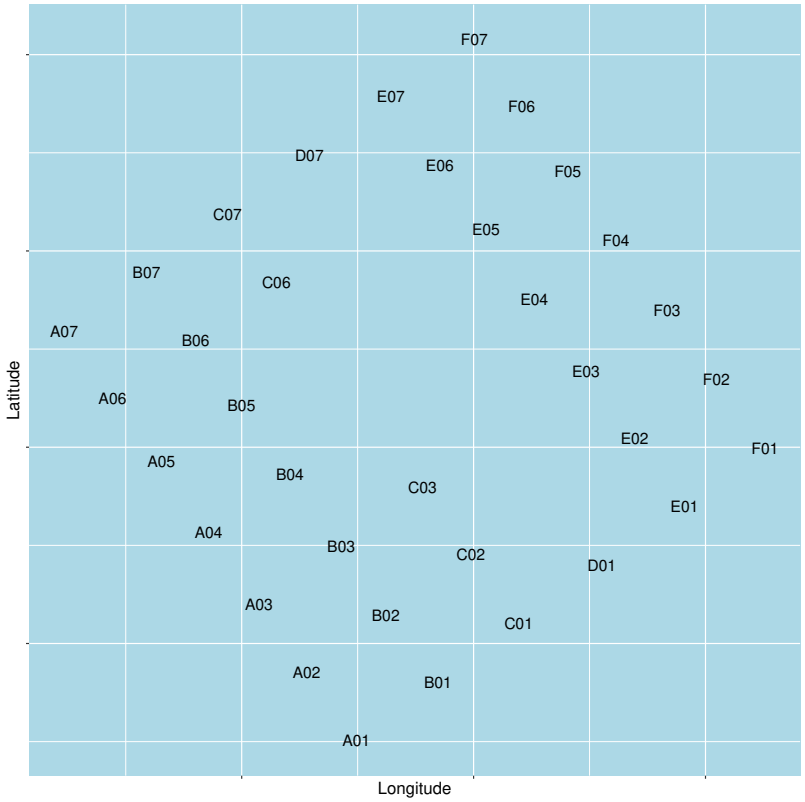


FIGURE 1 Position and name of the 35 wind turbines in Westermost Rough offshore wind farm (UK).

years, from January 1, 2016 to December 31, 2017, at a temporal resolution of every 10 minute. We will focus on the graph representations in section 3.1 and apply the method to power generation imputation in section 3.2.

3.1 | The Westermost Rough graph representations

When constructing a graph one's objective is to model the local neighborhood relationships and we choose to connect wind turbines that are actual neighbors, i.e., there is no other wind turbine nor empty space between them. The corresponding graph is presented in Figure 2. This is an important step for which there is no absolutely right choice and one should keep in mind that choosing an appropriate a priori graph matters to the results. In the case of Westermost Rough, one could choose not to connect wind turbines that are neighbors through a diagonal. How did we choose this graph? When looking at the exact locations, it appears the grid pattern is not exact and in reality two wind turbines on what looks like a diagonal might be closer than two wind turbines on an horizontal line. Also the direction of the wind might matter more than the real distances, and last but not least we feel that for a wind turbine the more neighbors the better.

Now that we have decided upon our reference graph $G = (V, E)$, we start with an unweighted version of G, i.e., the edges E have 0/1 weights only. We compute the corresponding Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ and solve the generalized eigenvector problem in (5). We leave out the (first) constant eigenvector and get an embedding $\mathbf{z}^i$ for every wind turbine v_i. These embeddings are of maximal size 34, i.e., N minus the first constant eigenvector $\mathbf{f}_0$. In Figure 3 we show the corresponding representations of the wind turbines according to eigenvectors $\mathbf{f}_1$ and $\mathbf{f}_2$ (left) and eigenvectors $\mathbf{f}_2$ and $\mathbf{f}_3$ (right). On these eigenmaps the wind turbines are smoothly clustered together over the wind farm according to their position, where both the local neighborhoods and the whole structure of the wind farm are accounted for. These plots provide nice insight about the representations learned from running the eigenmap algorithm over the unweighted graph, and how they embed the local and global geometry of the wind farm layout in a lower dimension.

We focus here on imputing missing data for wind power generation. Therefore we can weight the edges E of G using the methodology and the similarity described in section 2.3, where $s_t(i,j) = 1 - \|x_t^i - x_t^j\|$ is the similarity at time t between the power

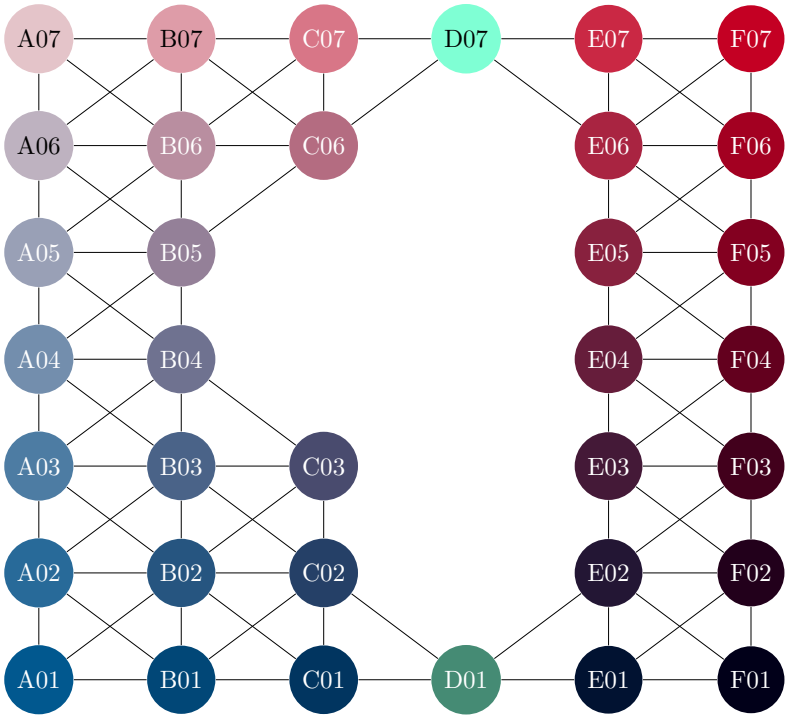


FIGURE 2 Graph of the Westermost Rough offshore wind farm with color code.

generations x_t^i and x_t^j of two connected wind turbines v_i and v_j. Working with weighted graphs whose weights vary over time, the eigenmap algorithm needs to be run, and we get different embeddings, at every time t. As mentioned in section 2.3.2, the graph itself can change, if an edge's weight becomes 0 at time t. As for the representations learned by the eigenmap algorithm, the clusters are still dependent upon the original graph G, i.e. upon the geography of the wind farm, but the distances between the embeddings may now change depending on an edge's weight $s_t(i,j)$.

3.2 | Imputation of power generation values

3.2.1 | Evaluation setup

In the methodology we propose, we do not estimate any parameters and have tried to minimize the number of choices that need to be made. Nonetheless when moving to imputation we need to decide on a few hyperparameters, which are:

- the number of dimensions r we keep for the embeddings $\mathbf{z}^i$ we get out of the unweighted graph or the embeddings $\mathbf{z}_t^i$ we get out of the weighted graphs,
- the kernel function which turns the distances between these embeddings into weights for the weighted k-NN imputation,
- the learning rate η in Algorithm 1 if we use weighted graphs and lazy OGD.

Recall that we have two years of data records over the Westermost Rough wind farm, 2016 and 2017. We split this dataset into two sets: a validation set, the first year of data, 2016, for deciding upon the hyperparameters; a test set, the second year, 2017, for evaluating our method compared to the naive Nadaraya-Watson estimator which is our current reference. Another simple benchmark is to consider the geographical locations of the wind turbines and just test Nadaraya-Watson estimators out of the actual distances between the wind turbines. We will refer to this benchmark as the "location" Nadaraya-Watson estimator. We

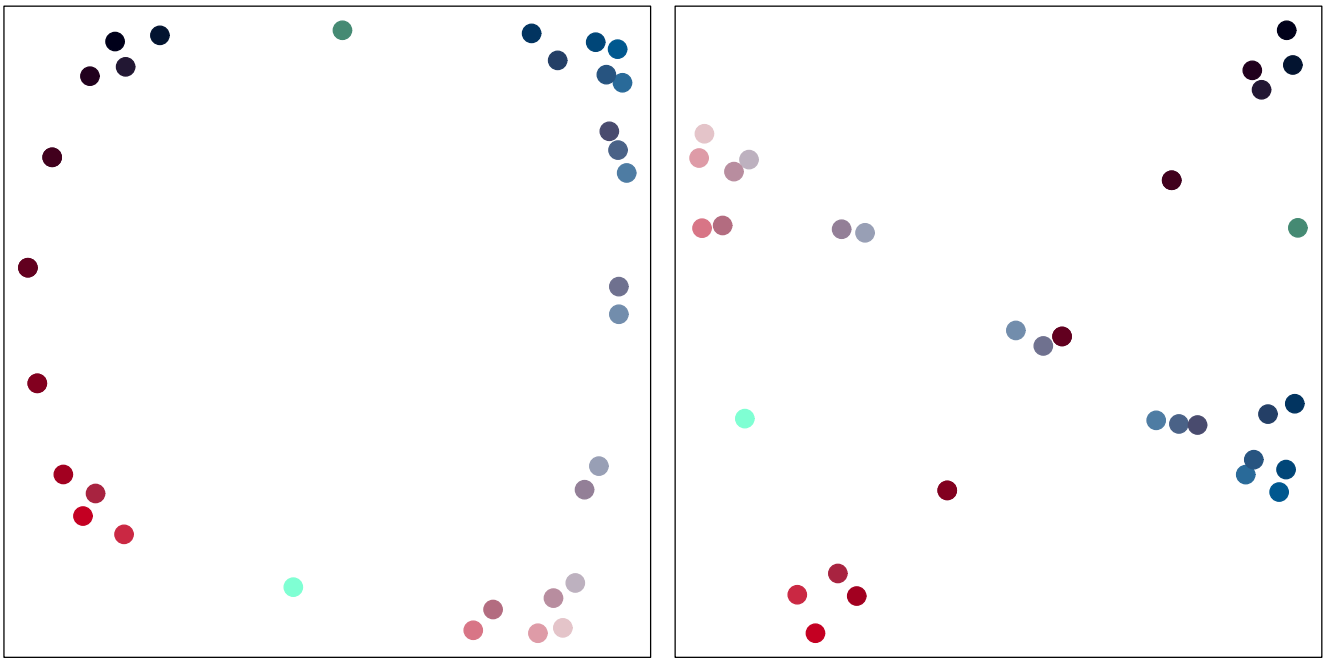


FIGURE 3 Representation of the wind turbines according to dimensions 1 and 2 (left) and dimensions 2 and 3 (right).

evaluate the different estimators through the RMSE

$$\sqrt{\frac{1}{T_i}\sum_{t=1}^{T_i}\left(x_t^i - \sum_{j=1}^{n_t} w_t^{(ji)} x_t^{(j)}\right)^2}, \tag{8}$$

which we will compute for each wind turbine v_i, where T_i is the number of records for which x_t^i is available. We compute the RMSE in equation (8) for two different setups:

- a setup we call *incomplete*, for which we compute (8) for all data records which include x_t^i, no matter which other data records are available at time t;
- a setup we call *complete*, for which we compute (8) on complete data records only, i.e., such that we have x_t for all N wind turbines.

The validation set consists in 52,669 records, 23,499 records being complete records. The test set consists in 52,549 records, 29,560 records being complete records. The *incomplete* setup enables us to evaluate the quality of the estimates for each wind turbine taking into account the reality of the availability of other records, while the *complete* setup measures an ideal quality of the estimate, in the sense that we assume all the other records to be available. Note that to simulate our own missing values is not really an option on this dataset, as we only have nearly half of the records which are complete records, and they are unlikely to be successive records, breaking down the dynamics of the time series.

3.2.2 | Hyperparameter selection

The estimators we are evaluating are

- the naive estimator: Nadaraya-Watson estimator with a naive kernel, which comes down to equal weights for all available data records;
- the location estimator: Nadaraya-Watson estimator which assigns weights depending on the real geographical distances between the wind turbines;

- the unweighted-graph estimator: Nadaraya-Watson estimator which assigns weights depending on the distances between embeddings obtained from the unweighted graph G;
- the weighted-graph estimator: Nadaraya-Watson estimator which assigns weights depending on the distances between embeddings obtained from online weighted graphs G_t.

On the validation set, the location estimator gives best results with a triweight kernel. The unweighted-graph estimator is always better using a dimension $r = 2$ for the embeddings. In the complete setup, the triweight kernel is by far the best one. It is still the first one in the incomplete setup but much closer to the second best one, which is the quartic kernel. This is the best kernel overall, but let us note that on the validation set the power generation of wind turbines C03 and E04 would be (slightly) better approximated using this estimator with a Gaussian kernel.

For the weighted-graph estimator, we need to use estimated similarities on the edges between the wind turbine we are testing and its neighbors in the complete setup, and on all edges involving missing data records in the incomplete setup. We can choose which estimated similarity to use without running the eigenmap algorithm by looking at the loss we pay on the validation set:

$$\sum_{t=1}^{T_{\text{val}}} l_t(i,j) = \sum_{t=1}^{T_{\text{val}}} (s_t(i,j) - \hat{s}_t(i,j))^2 \mathbb{1}_{\{s_t(i,j)\}}. \tag{9}$$

In the online learning paradigm, the loss in (9) is a reference when computed for the best constant strategy

$$\hat{s}(i,j) = \sum_{t=1}^{T_{\text{val}}} s_t(i,j) \mathbb{1}_{\{s_t(i,j)\}} \Big/ \sum_{t=1}^{T_{\text{val}}} \mathbb{1}_{\{s_t(i,j)\}}. \tag{10}$$

The difference between the loss of any other strategy and the loss of the best constant strategy is known as the *regret* (of not sticking to the best choice in hindsight) and is what theoretical results for online learning mostly focus on, i.e., ensuring the *regret* is (nicely) bounded when using a specific strategy. When computing the regret of lazy OGD on the validation set for different values of the learning rate η, it becomes very clear that lazy OGD provides far better results than the best constant strategy. This is a confirmation that the similarities between wind power time series are nonstationary. An optimal learning rate can be computed from theoretical results on lazy OGD. Let D be the diameter of the support $\mathcal{K}$ of the loss function l_t we pay at each round t, G an upper bound to the norm of the gradients of l_t and T the number of rounds. The regret of lazy OGD is best bounded by taking $\eta = D(G\sqrt{T})^{-1}$ [30]. We have $\mathcal{K} = [0,1]$, $D = \sqrt{\max_{s_t, \hat{s}_t \in \mathcal{K}} (s_t(i,j) - \hat{s}_t(i,j))} = 1$ and $G = 2$ as

$$\|\nabla l_t(i,j)\| = \|-2(s_t(i,j) - \hat{s}_t(i,j)) \mathbb{1}_{\{s_t(i,j)\}}\| \leq 2. \tag{11}$$

Since we deal with missing data, we get $\eta = \left(2\sqrt{\sum_{t=1}^{T_{\text{val}}} \mathbb{1}_{\{s_t(i,j)\}}}\right)^{-1}$ [8]. The learning rate is set to decrease over time when one is interested in converging to an optimal solution. And, as we want to track a time-varying quantity a standard approach is to rather choose the learning rate to be constant. Therefore we shall take $\eta = 0.5$. This is empirically verified on our validation set as better regrets are obtained for constant learning rates in $[0.3, 0.5]$. In the case where $\eta = 0.5$, lazy OGD comes down to what is known as persistence in time series forecasting without missing values, which is simply to use the last observed value. For the weighted-graph estimator, using lazy OGD with $\eta = 0.5$, the best kernel is again the triweight kernel in the complete setup but the best results are now obtained using a dimension $r = 4$ for the embeddings. Let us note that the wind power generation of C03 is way better approximated by this estimator, with an improvement of about 11%. In the incomplete setup, the best estimator is also the one with a triweight kernel and a dimension $r = 4$ for the embeddings. However the improvement we get on C03 is now of about 4% only.

3.2.3 | Results on the test set

From the results on the validation set, we compute all the estimators over the test set using a triweight kernel. For the unweighted-graph estimator, we use embeddings of dimension 2. For the weighted-graph estimator, we use embeddings of dimension 4 and lazy OGD with $\eta = 0.5$. The results on the test set in the complete, resp. incomplete, setup are given in Table 1, resp. in Table 2. They are averaged over the wind turbines. The improvement over the naive estimator is plotted by wind turbine in Figure 4, resp. in Figure 5, for each estimator. Accounting for the geographical distance between a wind turbine and its neighbors already

improves the estimates for every wind turbine compared to a naive estimator which assigns the same weight to all neighbors. With additional information about the distances between the neighbors themselves, through the structure of the wind farm, the estimators perform better on average over the wind farm, but there is more variability depending on the wind turbine. Especially the power generation of wind turbine C03 is better approximated by a naive estimator. Wind turbine C03 has a very central position which supports the choice of an estimator with equal weights. It might also be that C03's closer neighbors show a rather different behaviour and do not help much in estimating its power generation. This degradation is limited by including information about the power generations through a weighted graph.

The improvement we get from using the weighted-graph estimator is significant overall, for both complete and incomplete setups, and more stable over wind turbines. In particular the weighted-graph estimator performs better imputation for wind turbines which are on the outer parts of the wind farm, up to more that 20% for some of them. As a reference, if we use the real similarities for the edges in the complete setup instead of the estimates from lazy OGD, the overall improvement over the naive estimator is 11.21%, compared to 10.34% with the estimates. There is still room for improving the estimated similarities, for example by taking into account exogenous information such as the wind direction or speed, but the lazy OGD already performs pretty well.

Finally, let us note that on the test set 43.76% of the records are not complete. Most of these incomplete records miss wind power generation data for one (69.57%), two (23.22%) or three wind turbines (6.79%). The case when only one record is missing is well described by our *complete* setup, where we remove the record of only one wind turbine and look into how well the missing data point is imputed by the estimator, depending on which turbine it has been removed for.

3.3 | Discussion

First, let us acknowledge that when working at an aggregated level, over large wind farms, if in most cases only one record is missing at time t out of N, averaging over the remaining $N-1$ records seems fair enough, at least for Westermost Rough. This might not be the case though for wind farms with less conventional layouts. Accounting for the geographical distances when computing the weights of a weighted average can already lead to significant improvements. The additional information of the structure of the wind farm gave better estimators on average, but at the cost of more instability, by degrading the imputation for some wind turbines - thinking of wind turbine C03. Note that we have restrained ourselves to the choice of a common kernel

TABLE 1 Results on the test set in the *complete* setup.

Estimator	avg RMSE (sd)	avg impr./naive	best impr./naive (v_i)	worst impr./naive (v_i)
naive	12.32% (2.86)	-	-	-
location	11.55% (3.02)	6.83%	17.69% (F06)	0.65% (C06)
unweighted-graph	11.50% (3.06)	7.25%	21.36% (F06)	-4.10% (C03)
weighted-graph	11.11% (2.95)	10.34%	20.50% (C07)	-2.11% (C03)

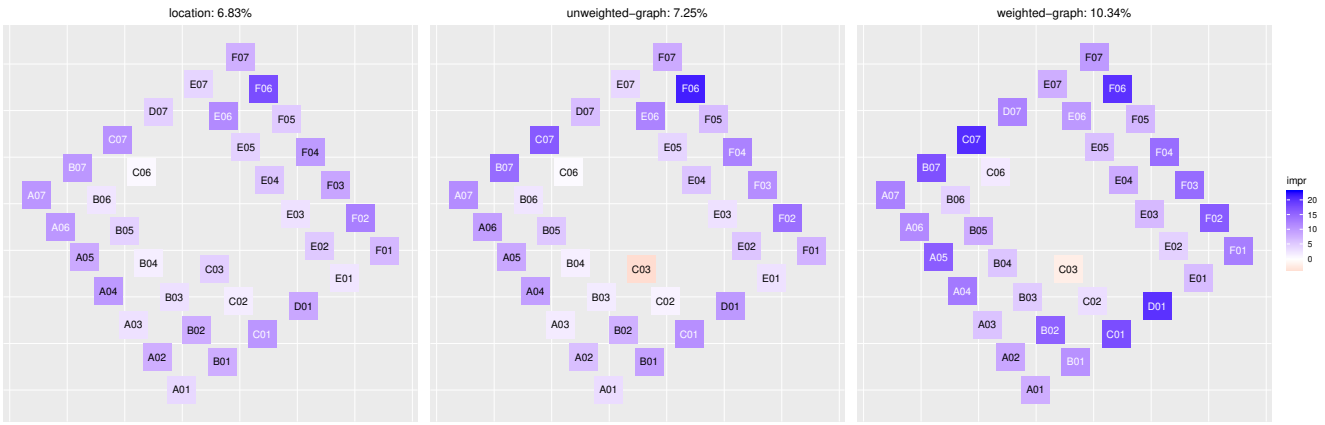


FIGURE 4 Improvements over the naive estimator by wind turbine in the *complete* setup.

TABLE 2 Results on the test set in the *incomplete* setup.

Estimator	avg RMSE (sd)	avg impr./naive	best impr./naive (v_i)	worst impr./naive (v_i)
naive	12.14% (2.35)	-	-	-
location	11.39% (2.44)	6.49%	14.45% (F06)	0.94% (C06)
unweighted-graph	11.34% (2.47)	6.93%	17.31% (F06)	-4.01% (C03)
weighted-graph	10.92% (2.42)	10.35%	23.20% (D01)	-2.59% (C03)

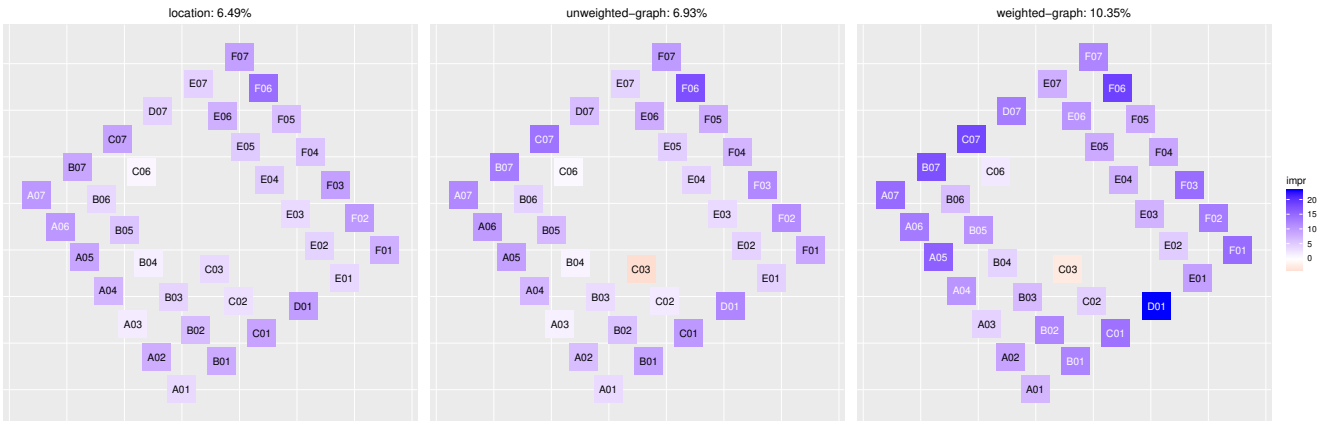


FIGURE 5 Improvements over the naive estimator by wind turbine in the *incomplete* setup.

over the wind farm. If choosing a different dimension r for the embeddings would not make much sense, to choose a different kernel for different wind turbines seems rather appropriate depending on the position of the wind turbine in the wind farm. However, moving into that direction, we would rather opt for replacing the Nadaraya-Watson estimators with an algorithm which can compute optimal weights efficiently and adaptively for each data point we wish to estimate, out of the distances between the wind turbines[24] - actual distances between locations or distances between embeddings.

If one is interested in individual signals, the methodology we propose can make much more difference, as for some wind turbines the graph-based estimators improve the power generation estimates by more than 20%. We have focused here on what is known as single imputation as we have tried to impute missing entries as accurately as possible, which gives us only one completed dataset. Multiple imputation on the other hand consists in predicting M different values for each missing data point and provides M imputed datasets. Multiple imputation is usually preferred, especially for inference tasks, as it ensures the variance is properly accounted for. We want to emphasize here that k-NN nicely enable to move to a probabilistic framework, when one is interested in the distributions of the time series, as a weight $w^{(jl)}$ can be seen as the probability of the missing data point $x^{(l)}$ to take the value of the available data point $x^{(j)}$. Let $\delta_{x^{(j)}}(x)$ denote the Dirac delta mass located at $x^{(j)}$. Instead of using a weighted average as a point estimate of $x^{(l)}$, we can assume $x^{(l)}$ to be distributed according to the empirical measure

$$\hat{\pi}(x) = \sum_{j=1}^{n} w^{(jl)} \delta_{x^{(j)}}(x), \tag{12}$$

and simply sample from (12), that is select $x^{(j)}$ with probability $w^{(jl)}$.

The location and unweighted-graph estimators can be very useful early on in the life of a wind farm, as they do not require any data points. The weighted-graph estimator does use data points, but can start as soon as there are data points for all wind turbines, as it does not require any model assumption nor estimation. The a priori we base our method on, by building the graph upon the structure of the wind farm, can be seen both as a pro and a con of the method, as it enforces proximity between wind turbines depending on their position inside the wind farm. This can bring robustness if it is appropriate, or instability when it is not, as we have seen with wind turbine C03. We have already mentioned this could be mitigated by moving from Nadaraya-Watson estimators to optimal weights thanks to an efficient algorithm[24]. When using data points, that is weighted graphs, an alternative would be to remove this assumption and start from what is known as a *complete* graph, in which every pair of wind turbines are connected by an edge. Then the complete graph would evolve online depending on the similarity on the edges as in section 2.3.2.

Nonetheless this would require to monitor $N(N-1)/2$ similarities and to perform eigendecomposition on matrices which might not be sparse anymore.

4 | CONCLUSION AND FUTURE DIRECTIONS

From the intuitive practice of averaging over a wind farm in order to deal with missing data points, we have focused on weighted k-NN imputation and dealt not only with distances between a wind turbine and its neighbors but also with distances between the neighbors themselves by learning graph representations. Weighting the graph edges with the similarity at time t between two data points from two time series and using spectral graph theory enabled us to compute online representations which could adapt to changes in the relationship between a wind turbine and its neighbors, typically when a wind turbine is not producing.

When using Nadaraya-Watson estimators, we chose and applied the kernel which gave the best results on average over the wind farm, and over the data records. We believe our method could benefit from replacing these estimators with an algorithm that would rather compute optimal weights efficiently and adaptively for each data point when performing the weighted k-NN imputation. By each data point we mean by wind turbine, and at each time step t in the case of online weighted graphs. The unweighted-graph imputation might be more robust, as the method would adapt to any wind turbine without having to make any kernel assumption. The same applies to weighted-graph imputation, although some of the uncertainty is already handled through the addition of data information. Of course the price to pay would be more computational effort.

A better imputation of missing values makes it easier to learn better forecasters but imputation methods often require assumptions on distributions and some might be difficult to apply to any sort of predictor. Nearest neighbors imputation can be used with any predictor and do not ask for any other assumption than similar neighbors. In a highly nonstationary setup, such as offshore wind energy, the data point from the neighboring time series might just be one of the best estimates we can get for the data point we are missing.

ACKNOWLEDGMENTS

The authors gratefully acknowledge Ørsted for providing the data for the Westermost Rough offshore wind farm. The research leading to this work was carried out as a part of the Smart4RES project (European Union's Horizon 2020, No. 864337). The sole responsibility of this publication lies with the authors. The European Union is not responsible for any use that may be made of the information contained therein.

CONFLICT OF INTEREST STATEMENT

The authors report no conflict of interests.

REFERENCES

1. Coville A, Siddiqui A, Vogstad KO. The effect of missing data on wind resource estimation. *Energy.* 2011;36(7):4505-4517. doi: https://doi.org/10.1016/j.energy.2011.03.067
2. Salmon J, Taylor P. Errors and uncertainties associated with missing wind data and short records. *Wind Energy.* 2014;17(7):1111-1118. doi: https://doi.org/10.1002/we.1613
3. Hosseini SH, Tang CY, Jiang JN. Calibration of a wind farm wind speed model with incomplete wind data. *IEEE Transactions on Sustainable Energy.* 2014;5(1):343-350. doi: 10.1109/TSTE.2013.2284490
4. Hu Y, Qiao Y, Liu J, Zhu H. Adaptive confidence boundary modeling of wind turbine power curve using SCADA data and its application. *IEEE Transactions on Sustainable Energy.* 2019;10(3):1330-1341. doi: 10.1109/TSTE.2018.2866543
5. Tawn R, Browell J, Dinwoodie I. Missing data in wind farm time series: properties and effect on forecasts. *Electric Power Systems Research.* 2020;189:106640. doi: https://doi.org/10.1016/j.epsr.2020.106640
6. Zhu Z, Wang T, Samworth RJ. High-dimensional principal component analysis with heterogeneous missingness. *Journal of the Royal Statistical Society Series B: Statistical Methodology.* 2022;84(5):2000-2031. doi: 10.1111/rssb.12550
7. Dunsmuir W, Robinson PM. Estimation of time series models in the presence of missing data. *Journal of the American Statistical Association.* 1981;76(375):560–568.
8. Anava O, Hazan E, Zeevi A. Online time series prediction with missing data. In: Bach F, Blei D., eds. *Proceedings of the 32nd International Conference on Machine Learning*. 37 of *Proceedings of Machine Learning Research*. PMLR 2015; Lille, France:2191–2199.
9. Stratigakos A, Andrianesis P, Michiorri A, Kariniotakis G. Towards resilient energy forecasting: a robust optimization approach. *IEEE Transactions on Smart Grid.* 2023:1-1. doi: 10.1109/TSG.2023.3272379
10. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological).* 1977;39(1):1–38.
11. Rubin DB. Inference and missing data. *Biometrika.* 1976;63(3):581–592.
12. Seaman S, Galati J, Jackson D, Carlin J. What is meant by "missing at random"?. *Statistical Science.* 2013;28(2):257–268.
13. Josse J, Prost N, Scornet E, Varoquaux G. On the consistency of supervised learning with missing values.; 2020.

14. Le Morvan M, Josse J, Scornet E, Varoquaux G. What's a good imputation to predict with missing values?. In: Ranzato M, Beygelzimer A, Dauphin Y, Liang P, Vaughan JW., eds. *Advances in Neural Information Processing Systems*. 34. Curran Associates, Inc. 2021:11530–11540.
15. Wen H, Pinson P, Gu J, Jin Z. Wind energy forecasting with missing values within a fully conditional specification framework. To be published in the International Journal of Forecasting; 2022.
16. Cini A, Marisca I, Alippi C. Filling the gaps: multivariate time series imputation by graph neural networks. Published as a conference paper at ICLR 2022; 2022.
17. Luo Y, Cai X, Zhang Y, Xu J, xiaojie Y. Multivariate time series imputation with generative adversarial networks. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R., eds. *Advances in Neural Information Processing Systems*. 31. Curran Associates, Inc. 2018.
18. Cao W, Wang D, Li J, Zhou H, Li L, Li Y. BRITS: bidirectional recurrent imputation for time series. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R., eds. *Advances in Neural Information Processing Systems*. 31. Curran Associates, Inc. 2018.
19. Hastie T, Tibshirani R, Friedman J. *The Elements of Statistical Learning*. Springer New York, NY, 2009.
20. Biau G, Devroye L. *Lectures on the Nearest Neighbor Method.* Springer, 2015.
21. Troyanskaya O, Cantor M, Sherlock G, et al. Missing value estimation methods for DNA microarrays . *Bioinformatics.* 2001;17(6):520-525. doi: 10.1093/bioinformatics/17.6.520
22. Nadaraya A. On estimating regression. *Theory of Probability and Its Applications.* 1964;9(1):141-142. doi: https://doi.org/10.1137/110902
23. Watson GS. Smooth regression analysis. *Sankhyā: the Indian Journal of Statistics, Series A.* 1964;26(4):359-372.
24. Anava O, Levy K. k^*-nearest neighbors: from global to local. In: Lee D, Sugiyama M, Luxburg U, Guyon I, Garnett R., eds. *Advances in Neural Information Processing Systems*. 29. Curran Associates, Inc. 2016.
25. Belkin M, Niyogi P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation.* 2003;15(6):1373-1396. doi: 10.1162/089976603321780317
26. Von Luxburg U. A tutorial on spectral clustering. *Statistics and Computing.* 2007;17(4):395–416. doi: 10.1007/s11222-007-9033-z
27. Golub GH, Loan CFV. *Matrix Computations, 4th edition*. Johns Hopkins University Press, 2013.
28. Cesa-Bianchi N, Lugosi G. *Prediction, Learning, and Games*. Cambridge University Press, 2006
29. Orabona F. A modern introduction to online learning.; 2022.
30. Hazan E. Introduction to online convex optimization.; 2021.

How to cite this article: Pierrot A., and Pinson P. Data is missing again – Reconstruction of power generation data using k-Nearest Neighbors and spectral graph theory .

APPENDIX

Nomenclature

η	Learning rate of lazy OGD
λ	Eigenvalue
$\mathbf{A}$	Adjacency matrix
$\mathbf{D}$	Degree matrix
$\mathbf{f}$	Eigenvector
$\mathbf{L}$	Laplacian matrix
$\mathbf{z}^i$	Embedding of wind turbine i
E	Set of edges
G	Graph
K	Kernel
$l_t(i,j)$	Loss function we pay at time t when guessing the similarity between records of wind turbine i and wind turbine j
N	Total number of wind turbines
n_t	Number of records available at time t
r	Dimension of the embeddings
$s_t(i,j)$	Similarity between records of wind turbine i and wind turbine j at time t
T	Total number of records
t	Time step
V	Set of nodes (or vertices)
v_i	Node i
$w^{(ji)}$	Weight of the j-th available record when imputing the i-th missing record with k-NN
$x^{(i)}$	x (e.g. power generation) of the i-th wind turbine among a varying set
x^i	x of wind turbine i among the total set $i = 1, \dots, N$

x_t^i x of wind turbine i at time t

Department of Wind and Energy Systems
Division for Power and Energy Systems (PES)
Technical University of Denmark
Elektrovej, Building 325
DK-2800 Kgs. Lyngby
Denmark