International Journal of Forecasting 37 (2021) 205–223

Contents lists available at ScienceDirect

International Journal of Forecasting

journal homepage: www.elsevier.com/locate/ijforecast

Online distributed learning in wind power forecasting

Benedikt Sommer [a], Pierre Pinson [a,*], Jakob W. Messner [a], David Obst [b]

[a] Technical University of Denmark, Department of Electrical Engineering, Kgs. Lyngby, Denmark
[b] EDF R&D, Palaiseau, France

ARTICLE INFO

Keywords:
Energy forecasting
Distributed optimisation
Online learning
Multivariate time series
Wind energy

ABSTRACT

Forecasting wind power generation up to a few hours ahead is of the utmost importance for the efficient operation of power systems and for participation in electricity markets. Recent statistical learning approaches exploit spatiotemporal dependence patterns among neighbouring sites, but their requirement of sharing confidential data with third parties may limit their use in practice. This explains the recent interest in distributed, privacy preserving algorithms for high-dimensional statistical learning, e.g. with autoregressive models. The few approaches that have been proposed are based on batch learning. However, these approaches are potentially computationally expensive and do not allow for the accommodation of nonstationary characteristics of stochastic processes like wind power generation. This paper closes the gap between online and distributed optimisation by presenting two novel approaches that recursively update model parameters while limiting information exchange between wind farm operators and other potential data providers. A simulation study compared the convergence and tracking ability of both approaches. In addition, a case study using a large dataset from 311 wind farms in Denmark confirmed that online distributed approaches generally outperform existing batch approaches while preserving privacy such that agents do not have to actively share their private data.

1. Introduction

Following the sustained deployment of renewable energy generation capacities, especially in the case of wind energy, forecasting has received increasing interest. Accurate wind power forecasts enhance the profitability of wind farms when participating in electricity markets (Mazzi & Pinson, 2017). Power system operators rely on generation and load forecasts to optimally schedule conventional generation units and operating reserves (Matos, Bessa, Botterud, & Zhou, 2017). An overview of state-of-the-art methods for wind power forecasting is presented in Giebel and Kariniotakis (2017). While lead times ranging from hours to days have been the central focus of such forecasting methods, owing to wind power participation in electricity markets, other lead times ranging from a few minutes to a week ahead (or possibly more) are of relevance to a broad range of operational and decision-making problems. The demand for accurate wind power forecasts with very short lead times ranging from minutes to a few hours has supported the development of novel forecasting methods (Pinson, 2012; Pinson & Madsen, 2012). For very short lead times, statistical and machine-learning methods clearly dominate over methods based on numerical weather forecasts. The majority of forecasting methods only utilise local data that is recorded at the wind farm of interest. Numerous publications have shown that using high-dimensional learning methods in combination with data from surrounding sites, such as meteorological stations or wind farms, can improve forecast accuracy substantially (He, Yang, Zhang, & Vittal, 2014; Tastu, Pinson, Kotwa, Madsen, & Nielsen, 2011; Tastu, Pinson, Trombe, & Madsen, 2014).

* Corresponding author.
E-mail address: ppin@dtu.dk (P. Pinson).

https://doi.org/10.1016/j.ijforecast.2020.04.004

Generally, this modelling approach exploits spatiotemporal patterns in wind power generation. This is fairly intuitive since a propagating wind field causes lagged changes in the power production between two or more geographically dispersed wind farms. Naturally, an upwind location experiences changes in wind speed first, before a downwind location that is in the trajectory of the same wind field. Hence, using explanatory variables that are related to wind speed or the power production of an upwind location helps to forecast future changes to the power production of a downwind location. These dependencies may be very complex and conditional on prevailing weather conditions (Girard & Allard, 2013). Exploiting off-site information and spatiotemporal dynamics is something that is broadly considered in environmetrics, e.g. ozone forecasting (Paci, Gelfand, & Holland, 2013), for the prediction of weather variables, e.g. precipitation (Sigrist, Künsh, & Stahel, 2012), and in traffic forecasting (Min & Wynter, 2011). It also shares similarities with the problem of forecasting panel data with cross-sectional dependencies in econometrics (H., Baltagi, Fingleton, & Pirotte, 2014).

In practice, many algorithms that exploit spatiotemporal dependencies in wind power forecasting are based on batch learning. That is, they are based on the assumption that model coefficients are time-invariant. They are hence estimated once and for all on a training dataset (the so-called "batch" of data). The estimated coefficients are then used to issue predictions even though new data arrives sequentially. In applications where the true model coefficients are time-varying, such as in wind power forecasting due to seasonal variations in wind dynamics, as well as the environment of wind farms, using batch-learning algorithms impacts forecast accuracy negatively. An approach that is then often considered is to re-estimate the coefficients using a sliding or expanding training window whenever new data samples are available (Dowell & Pinson, 2016; Zhang & Wang, 2018). The training samples are usually weighted to control how fast the estimated coefficients adapt to changes in the dataset. However, this approach is unattractive in cases where the learning algorithm cannot efficiently re-estimate the model coefficients. For high-dimensional models in particular, the time required to estimate model coefficients can be prohibitive for many applications. Computationally efficient algorithms estimate time-varying model coefficients on the fly by using recursivity whenever a new data sample is available. We use the term "online learning" when referring to such learning algorithms. Analogously, the term "offline (batch) learning" refers to algorithms where the time-invariant model coefficients are estimated on a batch of training samples.

Most online learning algorithms for forecasting wind power are used in combination with models that rely on explanatory variables, which are measured exclusively at the wind farm of interest. Relevant methods are presented in Bessa, Miranda, Botterud, Zhou, and Wang (2012) and Møller, Nielsen, and Madsen (2008). Notable exceptions are the sparse online warped Gaussian model of Kou, Gao, and Guan (2013) and the proposal of Messner and Pinson (2019). In the latter case, the authors described an online algorithm for high-dimensional vector autoregressive models. A limitation is that, to employ the algorithm, all explanatory variables must be collected by a single agent. In what follows, we use the term "centralised learning" when referring to situations where it is necessary to have direct access to all explanatory variables centrally in order to estimate model coefficients. Considering that wind farms are operated by competing agents and that power production data and related measurements are often deemed confidential, the requirement to collect all explanatory variables centrally brings some limitations. The unwillingness of wind farm operators to share data with third parties has motivated recent interest in distributed learning (and privacy-preserving) algorithms in the field of wind power forecasting.

Distributed learning algorithms conceptually aim at relaxing the need to collect all explanatory variables centrally, by decomposing a learning problem into many subproblems and one master problem. When estimating the coefficients of a forecasting model for a given wind farm of interest, for which some explanatory variables are provided by other wind farms, the distributed algorithm assigns a subproblem to each wind farm where explanatory variables are available. The model coefficients are then estimated by alternating between solving the master problem and subproblems, taking advantage of algorithm-specific variables that link the subproblems to the master problem and vice versa. With such algorithms it is no longer necessary to collect all explanatory variables centrally since the explanatory variables that are provided by other wind farms are only used in their respective subproblem. The appropriate design of distributed learning algorithms protects the explanatory variables of wind farm operators by not exposing them to others. We refer to this condition when stating that the data privacy of a wind farm operator is protected.

To our knowledge, only a handful of papers have investigated distributed learning algorithms for wind power forecasting (and renewable energy forecasting, more generally). The most prominent papers all build upon the Alternating Direction Method of Multipliers (ADMM). In Cavalcante, Bessa, Reis, and Browell (2017) and Pinson (2016) algorithms are developed to estimate the coefficients of an AR-X model while regularising with the LASSO. Zhang and Wang (2018) extend prior work to probabilistic forecasts but replace the L_1-penalization of the LASSO with an L_2-penalization to obtain a computationally cheaper algorithm. Unfortunately, prior distributed algorithms do not allow online learning to be performed. Therefore, to estimate time-varying coefficients the algorithms must be applied on a sliding or expanding training window while weighting the data samples. Consequently, we here aim to close the gap between online and distributed learning methods. Our contribution is twofold: we propose an online ADMM version, and we additionally propose a mirror-descent-inspired algorithm for online distributed learning. Both proposals have advantages and caveats that we explored through simulation studies and a case study with a large real-world dataset consisting of hundreds of wind farms in Denmark.

The remainder of this paper is organised as follows. The general model and forecasting framework are introduced in Section 2. Section 3 describes an Online ADMM version (OADMM). Anticipating the non-negligible computational complexity of the OADMM, the computationally lighter Adaptive Distributed MIrror Descent Algorithm made Sparse (Adaptive D-MIDAS) is presented thereafter in Section 4. The inherent properties of these two approaches were analysed through a simulation study, as described in Section 5. The algorithms were benchmarked on a large real-world dataset consisting of 311 wind farms, as discussed in Section 6. Conclusions and perspectives for future work are given in Section 7.

2. Modelling and forecasting framework

2.1. From agents and their data to relevant models

Wind power generation is observed at regular time intervals at S sites. Let us write $y_{s,t}$ for the power measurement of site $s \in \Omega_S = \{s_1, s_2, \ldots, s_S\}$ and timestamp $t \in \{1, 2, \ldots, T\}$. Power measurements are commonly normalised by the nominal capacity of the site, such that eventually, $y_{s,t} \in [0, 1]$. We restrict ourselves to AR-X models using recent power measurements as explanatory variables, in a fashion similar to the models used by Cavalcante et al. (2017), Messner and Pinson (2019), Pinson (2016) and Zhang and Wang (2018). However, extending such AR-X models to accommodate additional explanatory variables (e.g. wind speed) is straightforward. Generalisation to nonlinear modelling approaches would be more complicated. Depending on the type of data collected, it may be sensible to centre the data. Other types of transformations may additionally be considered. For instance, for nonlinear and bounded processes like wind power generation, the generalised logit-Normal transformation of Pinson (2012) may render more Gaussian innovations and yield a stochastic process that is more homoskedastic. Without any loss of generality, we assume that $y_{s,t}$ denotes transformed power measurements.

The operator of site s_j, referred to as the *central agent*, contracts a set $\Omega_S^{(j)} \subset \Omega_S \setminus s_j$ of other sites to enter a *learning agreement*. Consequently, all sites $s_i \in \Omega_S^{(j)}$ are referred to as *contracted agents*. In practice, this means that the contracted agents will support s_j in improving wind power forecasts through a distributed learning framework without exposing their explanatory variables to the central agent. The cardinality of $\Omega_S^{(j)}$ will certainly be small in practice, since it may not be relevant for a central agent to contract a large number of sites, e.g. due to the limited scale of dependence structures in space and time and possible transaction costs. Here, for simplicity, we assume that the cardinality of $\Omega_S^{(j)}$ is $S - 1$, so as to overlook the selection problem. We further assume that all contracted agents are rational and act truthfully. Therefore, we overlook the potential of malicious behaviour by assuming that the learning network design incentivises all agents to be fully collaborative (e.g. through contracts).

An AR-X model is used to link the power measured at site s_j and time t with past measurements of site s_j and the sites of the contracted agents. This gives

$$y_{s_j,t} = \beta_{s_j,0,t} + \sum_{l=1}^{L} \left(\underbrace{\beta_{s_j,l,t}\, y_{s_j,t-l}}_{\text{on-site}} + \sum_{s \in \Omega_S^{(j)}} \underbrace{\beta_{s,l,t}\, y_{s,t-l}}_{\text{off-site}} \right) + \epsilon_{s_j,t} \tag{1}$$

as a linear combination of past power measurements for all sites plus an intercept term $\beta_{s_j,0,t}$ and an innovation term $\epsilon_{s_j,t}$ with zero mean and finite variance. The scalars $\beta_{s,l,t}$ are the model coefficients for lag $l = 1, \ldots, L$ and site $s \in \Omega_S$, and L denotes the order of the auto-regressive process. For simplicity, we consider that the maximum lag L is the same for the central and contracted agents, though it does not need to be. In addition, a time index t is used, as it is assumed that the model coefficients are time-varying. While it may be common in the econometrics literature to assume that those coefficients follow some process, e.g. autoregressive (Bekierman & Manner, 2018), we consider that these coefficients follow a random walk with varying means. They can hence be tracked with some simple form of Kalman filtering, where parameters are updated recursively. This approach is common in wind power forecasting, as in the examples of Pinson (2012), Pinson and Madsen (2012), among others.

The model in (1) has many coefficients, since a different coefficient is used for each combination of location and lagged value. This potentially leads to the need to estimate $L \times S + 1$ coefficients with $L \times S + 1$ being large. As an alternative, one may parameterise the spatiotemporal dynamics of wind power generation, as commonly done in environmetrics and statistical modelling of meteorological variables (Sigrist et al., 2012). Here, however, these dynamics are very complex and conditional on prevailing weather conditions (Girard & Allard, 2013). Consequently, with access to large datasets, as is common with wind power forecasting, it is possible to increase the number of coefficients to be estimated. In parallel, note that in practice many of the $\beta_{s,l,t}$ coefficients are expected to be 0, depending on the de-correlation range and prevailing wind direction. This is why we employ a fully data-driven approach to variable selection and coefficient estimation through L_1-regularisation. In addition, since we are working within an online learning framework, the resulting model coefficients are time-varying and are thus expected to capture the slow variations in wind power dynamics, e.g. induced by seasons and changes in the environment of the wind farms.

For convenience we rewrite (1) in a compact form:

$$y_{s_j,t} = \sum_{s \in \Omega_S} \mathbf{a}_{s,t-1} \boldsymbol{\beta}_{s,t} + \epsilon_{s_j,t} \tag{2}$$

where $\mathbf{a}_{s,t-1}$ is an horizontal vector gathering the values of explanatory variables, at time t and location s, and $\boldsymbol{\beta}_{s,t}$ is the corresponding vector of model coefficients. That is,

$$\mathbf{a}_{s,t-1} = \begin{cases} [1,\ y_{s,t-1},\ \ldots,\ y_{s,t-L}], & s = s_j \\ [y_{s,t-1},\ \ldots,\ y_{s,t-L}], & \text{otherwise} \end{cases} \tag{3}$$

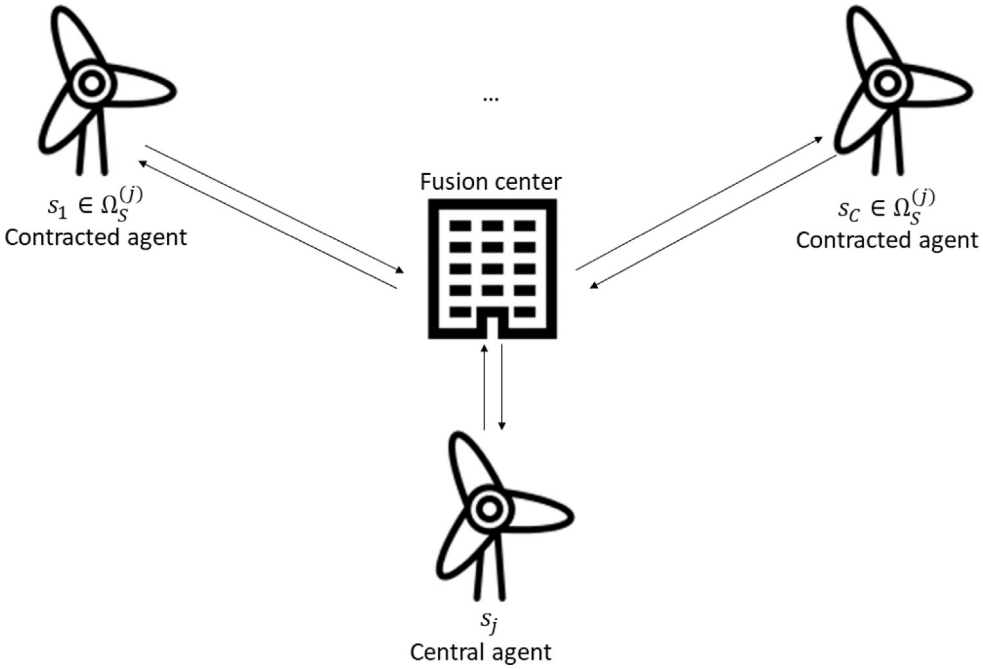


Fig. 1. Architecture of the distributed learning network.

and

$$\boldsymbol{\beta}_{s,t} = \begin{cases} [\beta_{s,0,t},\ \beta_{s,1,t},\ \ldots,\ \beta_{s,L,t}]^\top, & s = s_j \\ [\beta_{s,1,t},\ \ldots,\ \beta_{s,L,t}]^\top, & \text{otherwise} \end{cases} \tag{4}$$

Within this modelling framework, the largest contribution to explaining the dynamics of $y_{s_j,t}$ comes from local information given by the lagged values of this process. In comparison, offsite information provides a lower contribution, while still allowing a significant improvement in forecast accuracy for short lead times (Messner & Pinson, 2019). Since most of the $\beta_{s,l,t}$ coefficients are expected to be 0, this also implies that a central agent eventually does not need to make a learning agreement with many other wind farms, hence limiting communication needs and potential contracts if distributed learning were remunerated.

2.2. Framework for distributed and online learning

When estimating the model coefficients of such an ARX model in a centralised setup, an agent (most likely the operator of site s_j, i.e. the central agent or the contracted forecast vendor) is required to gather all explanatory variables. In a distributed learning network, however, the coefficient estimation problem is decomposed into many subproblems that are solved by the agents who entered the learning agreement. In our case the problem is conveniently decomposed across all S wind farm operators. The architecture of our distributed learning network is visualised in Fig. 1, where the arrows indicate information exchange.

Regularly applied in distributed networks, a fusion centre (supervisory node) oversees the communication among all agents. In practice, the central agent does not directly communicate with its contracted agents. That is, information is not directly exchanged via a peer-to-peer connection. The reason for designing the network like this is twofold. On the one hand, the communication becomes more structured for large-scale applications where each member of the learning agreement receives a forecast for its site. This requires estimating the coefficients of at least S models in parallel. On the other hand, it may address some of the privacy concerns of wind farm operators who do not wish to expose their private information to other agents.

In centralised learning the flow of information is unidirectional from the contracted agents to the central agent. Distributed learning algorithms instead require a bidirectional exchange of information, as the arrows show in Fig. 1. Our distributed learning algorithms require each agent to solve their assigned subproblem. This is fundamentally different from centralised learning, where only the central agent performs computations when estimating the model coefficients.

Numerous distributed and online algorithms have been proposed in the literature, though not for applications in renewable energy forecasting. We observed that all online versions of the ADMM address consensus problems. That is, they all require the design matrix to be horizontally partitioned across all agents (Matamoros, 2017; Suzuki, 2013; Wang & Banerjee, 2012). Fig. 2 illustrates the difference between a horizontal and vertical partitioning of the set of explanatory variables in a model like the one we use here as a basis for forecasting.

From (2), it can clearly be seen that in our forecasting problem the design matrices are naturally vertically partitionable across all S agents. That is, each agent observes a unique subset of the whole set of explanatory variables.

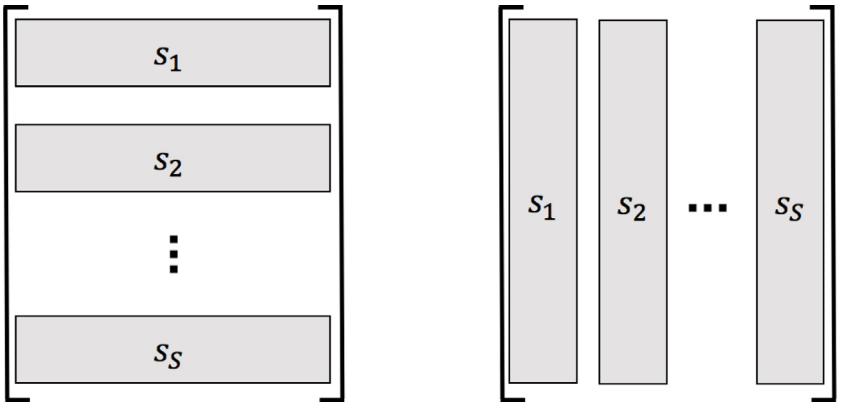


Fig. 2. Horizontal (left) and vertical (right) partitioning of a matrix across S agents. Both matrices have equal dimensions. Each column represents a unique feature, whereas a row is related to a time instance.

Horizontally partitionable datasets are found in applications where instances of the design matrix are recorded at different locations but with identical features (e.g. in clinical trials carried out across multiple hospitals). While online versions of the ADMM have been proposed for horizontally partitionable design matrices, this is not the case for vertically partitionable ones. This motivated our proposal, as described in the following section.

3. Online Alternating Direction Method of Multipliers (OADMM)

Pinson (2016) originally proposed using the ADMM (Boyd, Parikh, Chu, Peleato, & Eckstein, 2010) to estimate the AR-X model coefficients in (1) in a distributed fashion while applying L_1-regularisation with the LASSO. The applied ADMM estimates the model coefficients on a batch of training samples and does not allow for efficient coefficient re-estimates in applications where the true coefficients are expected to be time-varying. We thus extend this algorithm to an online version that minimises the cumulative loss over all observed data samples. Our online version efficiently re-estimates all model coefficients through recursions whenever a new data sample is available. A flowchart for our OADMM approach is presented in Fig. 3, and a detailed algorithm is available in Appendix.

3.1. Coefficient estimation through a time-varying optimisation problem

Considering one-step-ahead forecasting, the OADMM approach solves an unconstrained minimisation problem. For every timestamp t, it can be formulated as

$$\min_{\{\boldsymbol{\beta}_{s,t}\}_s} \frac{1}{2} \sum_{\tau=1+L}^{t} \left(\sum_{s\in\Omega_S} \boldsymbol{a}_{s,\tau-1}\boldsymbol{\beta}_{s,t} - y_{s_j,\tau} \right)^2 + \lambda \sum_{s\in\Omega_S} \|\boldsymbol{\beta}_{s,t}\|_1 \quad (5)$$

where $\boldsymbol{a}_{s,\tau-1}$ and $\boldsymbol{\beta}_{s,t}$ are as defined in (3) and (4). In parallel, $\lambda \geq 0$ is the L_1-regularisation parameter that controls sparsity. L_1-regularisation penalises the model coefficient absolute values and thereby shrinks coefficients deemed to be insignificant towards 0.

In order to solve the minimisation problem in (5) every time a new data sample is made available, it should be made computationally efficient. Additionally, we want to control the level of adaptivity with a forgetting factor, as in Messner and Pinson (2019), Møller et al. (2008) and Pinson and Madsen (2012). By giving less weight to older data, the model coefficient estimates better reflect the recent dynamics in the time-series data. Introducing an exponential forgetting factor ν into (5) results in

$$\min_{\{\boldsymbol{\beta}_{s,t}\}_s} \frac{1}{2} \sum_{\tau=1+L}^{t} \nu^{t-\tau} \left(\sum_{s\in\Omega_S} \boldsymbol{a}_{s,\tau-1}\boldsymbol{\beta}_{s,t} - y_{s_j,\tau} \right)^2 + \lambda \sum_{s\in\Omega_S} \|\boldsymbol{\beta}_{s,t}\|_1 \quad (6)$$

where $\nu \in]0, 1]$. A value of 1 results in no forgetting, while decreasing values increase the amount of forgetting. Values slightly less than 1 are generally preferred. ν may be optimised in practice through, e.g., cross-validation.

The standard ADMM builds on the dual-ascent method, which is used to solve optimisation problems where the objective function is separable, by splitting the complete model coefficient vector into sub-vectors. Our problem is naturally separable since each wind farm operator has unique explanatory variables $\boldsymbol{a}_{s,\tau-1}$ and related model coefficients $\boldsymbol{\beta}_{s,t}$ in (6). The optimisation problem is transformed into an appropriate ADMM sharing form by adding the auxiliary vector $\boldsymbol{z}_{s,t}$ to (6). The constrained optimisation problem then reads

$$\begin{aligned} \min_{\{\boldsymbol{\beta}_{s,t}\}_s} \quad & \frac{1}{2} \sum_{\tau=1+L}^{t} \nu^{t-\tau} \left(\sum_{s\in\Omega_S} \boldsymbol{a}_{s,\tau-1}\boldsymbol{z}_{s,t} - y_{s_j,\tau} \right)^2 \\ & + \lambda \sum_{s\in\Omega_S} \|\boldsymbol{\beta}_{s,t}\|_1 \\ \text{subject to} \quad & \boldsymbol{\beta}_{s,t} - \boldsymbol{z}_{s,t} = 0, \quad \forall s \in \Omega_S \end{aligned} \quad (7)$$

The ADMM uses the augmented Lagrangian to solve the constrained optimisation problem with respect to $\boldsymbol{\beta}_{s,t}$

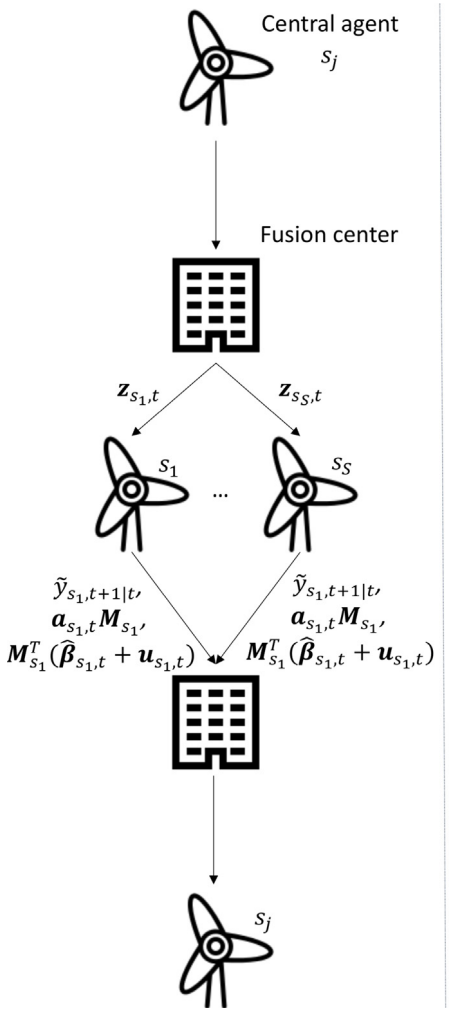


Fig. 3. Flowchart for the Online ADMM (OADMM) approach to online distributed learning applied to wind power forecasting.

and $z_{s,t}$, by updating the variables in an alternating fashion. For a detailed description of the ADMM and its application in distributed networks, the reader is referred to Boyd et al. (2010). When following the standard ADMM to solve (7), the central agent may be able to retrieve the explanatory variables of all contracted agents. This violates the data privacy of the contracted agents. In the offline ADMM for distributed learning of Pinson (2016), the explanatory variables $a_{s,\tau-1}$ of each agent are naturally protected in each step of the algorithm by being multiplied by the respective $\beta_{s,t}$. Taking this as inspiration, we introduce the encryption matrix $M_s \in \mathbf{R}^{L,L}$ and multiply it by $a_{s,\tau-1}$ whenever it appears. We achieve this by changing the affine constraint in (7) into

$$\beta_{s,t} - M_s z_{s,t} = 0, \quad \forall s \in \Omega_S \tag{8}$$

and additionally adjusting the objective function by replacing the term $a_{s,\tau-1}z_{s,t}$ with $a_{s,\tau-1}M_s z_{s,t}$. As a requirement, each encryption matrix must be non-singular and chosen by each agent privately. This eventually yields the encrypted version of the constrained optimisation problem (7). That is,

$$\min_{\{\beta_{s,t}\}_s} \frac{1}{2}\sum_{\tau=1+L}^{t}\nu^{t-\tau}\left(\sum_{s\in\Omega_S} a_{s,\tau-1}M_s z_{s,t} - y_{s_j,\tau}\right)^2 + \lambda\sum_{s\in\Omega_S}\|\beta_{s,t}\|_1 \tag{9}$$
$$\text{subject to} \quad \beta_{s,t} - M_s z_{s,t} = 0, \quad \forall s \in \Omega_S$$

The augmented Lagrangian of (9) in its scaled form is then written as

$$\mathcal{L}_\rho(\beta_t, z_t, u_t) = \frac{1}{2}\sum_{\tau=1+L}^{t}\nu^{t-\tau}\left(\sum_{s\in\Omega_S} a_{s,\tau-1}M_s z_{s,t} - y_{s_j,\tau}\right)^2 + \lambda\sum_{s\in\Omega_S}\|\beta_{s,t}\|_1 + \frac{\rho}{2}\sum_{s\in\Omega_S}\|\beta_{s,t} - M_s z_{s,t} + u_{s,t}\|^2 \tag{10}$$

where $\rho > 0$ is a penalty parameter and $\boldsymbol{u}_{s,t}$ denotes the dual variables for the constraints in (9). The OADMM then performs the minimisation of the augmented Lagrangian by sequentially optimising for $\boldsymbol{\beta}_{s,t}$ and $\boldsymbol{z}_{s,t}$ while updating the dual variables $\boldsymbol{u}_{s,t}$ as part of the dual ascent algorithm. By optimising for $\boldsymbol{\beta}_{s,t}$ and $\boldsymbol{z}_{s,t}$ individually, it is possible to take advantage of the separability of (10) with respect to all $\boldsymbol{\beta}_{s,t} \in \Omega_S$.

3.2. Recursive updates of parameters

3.2.1. Central agent updates

To perform an update at time t, let us first focus on the master problem of the central agent. The central agent first observes the true power production $y_{s_j,t}$ and subsequently derives the prediction error $y_{s_j,t} - \hat{y}_{s_j,t|t-1}$. The augmented Lagrangian is then minimised with respect to the auxiliary variable z. The minimisation is written as a parameter update, which is carried out exclusively by the central agent. To obtain the update equations we first define

$$\boldsymbol{\beta}_{t-1} = \left[\boldsymbol{\beta}_{s_1,t-1}^\top, \ldots, \boldsymbol{\beta}_{s_S,t-1}^\top\right]^\top, \tag{11a}$$

$$\boldsymbol{z}_{t-1} = \left[\boldsymbol{z}_{s_1,t-1}^\top, \ldots, \boldsymbol{z}_{s_S,t-1}^\top\right]^\top, \tag{11b}$$

$$\boldsymbol{u}_{t-1} = \left[\boldsymbol{u}_{s_1,t-1}^\top, \ldots, \boldsymbol{u}_{s_S,t-1}^\top\right]^\top, \tag{11c}$$

$$\boldsymbol{a}_{t-1} = \left[\boldsymbol{a}_{s_1,t-1}, \ldots, \boldsymbol{a}_{s_S,t-1}\right], \tag{11d}$$

the model coefficient estimates $\hat{\boldsymbol{\beta}}_t$ at time t, and the block-wise diagonal matrix

$$\boldsymbol{M} = \operatorname{diag}\left(\boldsymbol{M}_{s_1}, \ldots, \boldsymbol{M}_{s_S}\right) \tag{12}$$

Differentiating the augmented Lagrangian with respect to $\boldsymbol{z}_t$ yields

$$\frac{\partial \mathcal{L}_\rho(\hat{\boldsymbol{\beta}}_t, \boldsymbol{z}_t, \boldsymbol{u}_t)}{\partial \boldsymbol{z}_t} = \sum_{\tau=1+L}^{t} \nu^{t-\tau} \left(\boldsymbol{a}_{\tau-1}\boldsymbol{M}\right)^\top \left(\boldsymbol{a}_{\tau-1}\boldsymbol{M}\boldsymbol{z}_t - y_{s_j,t}\right) - \rho \boldsymbol{M}^\top \left(\hat{\boldsymbol{\beta}}_{t-1} - \boldsymbol{M}\boldsymbol{z}_t + \boldsymbol{u}_{t-1}\right) \tag{13}$$

By writing

$$\boldsymbol{H}_t = \sum_{\tau=1+L}^{t} \nu^{t-\tau} \left(\boldsymbol{a}_{\tau-1}\boldsymbol{M}\right)^\top \left(\boldsymbol{a}_{\tau-1}\boldsymbol{M}\right) \tag{14}$$

and

$$\boldsymbol{p}_t = \sum_{\tau=1+L}^{t} \nu^{t-\tau} \left(\boldsymbol{a}_{\tau-1}\boldsymbol{M}\right)^\top y_{s,t} \tag{15}$$

the auxiliary vectors are updated by equating (13) to 0 and then solving for $\boldsymbol{z}_t$. Hence, the OADMM requires

$$\left(\boldsymbol{H}_t + \rho \boldsymbol{M}^\top \boldsymbol{M}\right) \boldsymbol{z}_t = \boldsymbol{p}_t + \rho \boldsymbol{M}^\top \left(\hat{\boldsymbol{\beta}}_{t-1} + \boldsymbol{u}_{t-1}\right) \tag{16}$$

to be solved for $\boldsymbol{z}_t$. Before solving the equation system, the covariance structures $\boldsymbol{H}_t$ and $\boldsymbol{p}_t$ are efficiently updated via the recursions

$$\boldsymbol{H}_t = \nu \boldsymbol{H}_{t-1} + \left(\boldsymbol{a}_{\tau-1}\boldsymbol{M}\right)^\top \left(\boldsymbol{a}_{\tau-1}\boldsymbol{M}\right) \tag{17a}$$

$$\boldsymbol{p}_t = \nu \boldsymbol{p}_{t-1} + \left(\boldsymbol{a}_{\tau-1}\boldsymbol{M}\right)^\top y_{s,t} \tag{17b}$$

Both covariance structures comprise the memory of the recursive updating process, controlled by the forgetting factor ν.

After the central agent has updated the auxiliary vectors, it shares them with its contracted agents via the fusion centre. This is considered a broadcasting operation where the central agent distributes local variables within the network.

3.2.2. Contracted agent updates

After each contracted agent receives its respective auxiliary vector $\boldsymbol{z}_{s,t}$, all S agents update their dual variables in parallel with the recursion:

$$\boldsymbol{u}_{s,t} = \boldsymbol{u}_{s,t-1} + \hat{\boldsymbol{\beta}}_{s,t-1} - \boldsymbol{M}_s \boldsymbol{z}_{s,t} \tag{18}$$

where the update is part of the dual ascent method.

Next, we update $\hat{\boldsymbol{\beta}}_t$, where the augmented Lagrangian is separable across all $\hat{\boldsymbol{\beta}}_{s,t} \in \Omega_S$. Hence, this update is also carried out in parallel. Due to the L_1-norm of the LASSO, the Lagrangian is not differentiable with respect to $\hat{\boldsymbol{\beta}}_t$, though sub-differentiable. The final update then reads

$$\hat{\boldsymbol{\beta}}_{s,t} = \mathbb{S}_{\lambda/\rho}\left(\boldsymbol{M}_s \boldsymbol{z}_{s,t} - \boldsymbol{u}_{s,t}\right) \tag{19}$$

where $\mathbb{S}_\kappa(c)$ is a soft-thresholding operator

$$\mathbb{S}_\kappa(c) = \begin{cases} c - \kappa & \text{if } c > 0 \text{ and } \kappa < |c| \\ c + \kappa & \text{if } c < 0 \text{ and } \kappa < |c| \\ 0 & \text{if } \kappa > |c| \end{cases} \tag{20}$$

which is applied element-wise to the input $\boldsymbol{M}_s \boldsymbol{z}_{s,t} - \boldsymbol{u}_{s,t}$.

3.2.3. Back to the central agent

After all agents update their model coefficient estimates $\hat{\boldsymbol{\beta}}_{s,t}$, they compute the partial prediction $\boldsymbol{a}_{s,t}\hat{\boldsymbol{\beta}}_{s,t}$ with the latest explanatory variables. Besides sharing the partial prediction with the central agent, due to the z-update, the algorithm also requires each contracted agent to share $\boldsymbol{a}_{s,t}\boldsymbol{M}_s$ and $\boldsymbol{M}_s^\top \left(\hat{\boldsymbol{\beta}}_{s,t} + \boldsymbol{u}_{s,t}\right)$ with the central agent. The central agent also requires $\boldsymbol{M}_s^\top \boldsymbol{M}_s$, but this only needs to be shared once.

The last step before obtaining the next prediction requires the central agent to sum all partial predictions:

$$\hat{y}_{s_j,t+1|t} = \sum_{s \in \Omega_S} \boldsymbol{a}_{s,t}\hat{\boldsymbol{\beta}}_{s,t} \tag{21}$$

Because the OADMM requires each contracted agent to share the prior stated vectors and scalar with the central agent, the central agent has access to $L^2 + L + 1$ equations for each contracted agent and timestamp. The central agent cannot retrieve the elements of $\boldsymbol{a}_{s,t}$ because the obtained equations contain $2(L^2 + L)$ unknowns. Therefore, the data privacy of the contracted agents is protected. Besides protecting the data of each wind farm operator, the OADMM requires only a single bidirectional data exchange between the central agent and its contracted agents. Taking into consideration that only low-dimensional vectors and matrices are exchanged, the algorithm is efficient communication-wise. The pseudocode for the final version of the OADMM is presented in Appendix.

Considering all five required algorithm parameter updates, due to their low complexity, it is expected that the β-, u- and covariance structure updates can be performed efficiently and quickly. However, the z-update is more expensive because a linear system is solved that grows linearly with the number of agents S and the order of the AR process L. Therefore, for large-scale applications with hundreds or thousands of contracted agents, the z-update becomes time-intensive. This motivated us to develop a computationally lighter algorithm that can perform all parameter updates quickly, even in very large learning networks.

4. Adaptive Distributed MIrror Descent Algorithm made Sparse (Adaptive D-MIDAS)

In the following, we first present basic concepts for stochastic gradient descent algorithms, which are of relevance to our online distributed learning algorithm. A flowchart for the proposed Adaptive D-MIDAS is presented in Fig. 4, and a detailed algorithm is available in Appendix.

4.1. Basics of the SMIDAS

Stochastic gradient descent algorithms provide a great platform for designing computationally inexpensive online distributed learning methods. We derive in the following an algorithm that is heavily influenced by the work of Shalev-Shwartz and Tewari (2011). The authors proposed the Stochastic MIrror Descent Algorithm made Sparse (SMIDAS) for solving problems of the form

$$\min_{\boldsymbol{\beta}_{s_1},\ldots,\boldsymbol{\beta}_{s_S}} C\left(\boldsymbol{\beta}_{s_1},\ldots,\boldsymbol{\beta}_{s_S}\right) + \lambda \sum_{s\in\Omega_S} \|\boldsymbol{\beta}_s\|_1 \tag{22}$$

where, in regression problems, C is commonly the squared loss

$$C\left(\boldsymbol{\beta}_{s_1},\ldots,\boldsymbol{\beta}_{s_S}\right) = \sum_{\tau=1+L}^{T}\left(\sum_{s\in\Omega_S} \boldsymbol{a}_{s,\tau-1}\boldsymbol{\beta}_s - y_{s_j,\tau}\right)^2 \tag{23}$$

The proposal of Shalev-Shwartz and Tewari (2011) was motivated by previous work on stochastic optimisation for L_1-regularised problems. First, Duchi, Shalev-Shwartz, Singer, and Chandra (2008) described an algorithm that replaces the L_1-regularisation term in (22) with the constraint $\|\sum_{s\in\Omega_S}\boldsymbol{\beta}_s\|_1 \leq B$ and then uses a stochastic gradient projection procedure to estimate the model coefficients. Later, Langford, Lihong, and Zhang (2009) introduced a stochastic gradient descent algorithm where sparse solutions are obtained by truncating the model coefficients. That is, elements in the model coefficient vector that cross 0 during a gradient step are truncated to 0. The runtime of both algorithms might grow in some situations in a quadratic way with the dimension of the feature space even though the optimal coefficient vector is very sparse (Shalev-Shwartz & Tewari, 2011). Mirror descent algorithms instead achieve a runtime that is linear in the dimension of the feature space of the problem (Beck & Teboulle, 2003). This makes them particularly suitable for high-dimensional learning. However, they do not necessarily yield sparse solutions. In short, the SMIDAS uses mirror descent updates in combination with the truncation method of Langford et al. (2009). Hence, the SMIDAS achieves a superior runtime compared to the algorithms of Duchi et al. (2008) and Langford et al. (2009), while still yielding sparse solutions. Based on these properties we used the SMIDAS as a starting point for the proposal of an online distributed learning approach.

In the following, we first apply the SMIDAS to learn the time-invariant model coefficients of (22). This will help explain the subsequent derivation of our algorithm for learning time-varying model coefficients in a distributed setting.

4.2. Batch estimation with SMIDAS

Like most gradient-based optimisation methods, the algorithm in Langford et al. (2009) updates only one weight vector $\boldsymbol{\beta}$ every iteration. Mirror descent algorithms are conceptually different because they maintain two weight vectors: the primal vector $\boldsymbol{\beta}$ and the dual vector $\boldsymbol{\theta}$. The mirror descent algorithm was first derived in Nemirovski and Yudin (1983), while a new derivation is presented in Beck and Teboulle (2003). We recommend both works for a more detailed description of the algorithm.

The two weight vectors are linked via the transformation $\boldsymbol{\theta} = f(\boldsymbol{\beta})$, where f is a link function. From the derivation in Nemirovski and Yudin (1983) and under the right conditions, f is invertible. Hence, the inverse transformation $\boldsymbol{\beta} = f^{-1}(\boldsymbol{\theta})$ exists. In Shalev-Shwartz and Tewari (2011) a p-norm link function is used:

$$\beta_n = f_n^{-1}(\boldsymbol{\theta}) = \frac{\operatorname{sign}(\theta_n)|\theta_n|^{p-1}}{\|\boldsymbol{\theta}\|_p^{p-2}} \tag{24}$$

with

$$\|\boldsymbol{\theta}\|_p = \left(\sum |\theta_n|^p\right)^{\frac{1}{p}} \tag{25}$$

where θ_n is the nth element in $\boldsymbol{\theta}$.

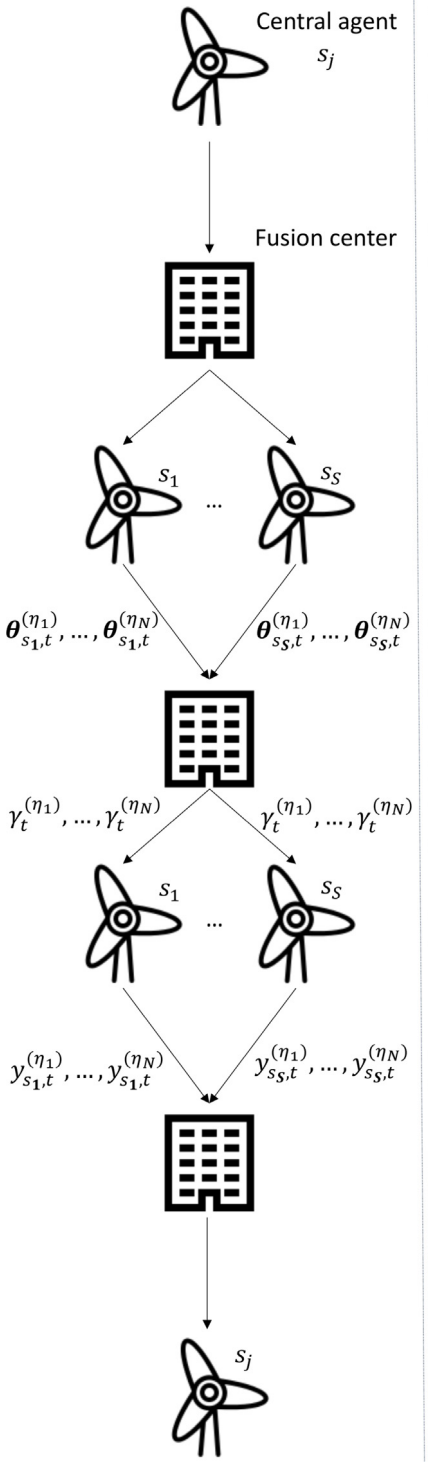
After this initial description of the mirror descent algorithm, let us apply the SMIDAS to learn the time-invariant model coefficients of (22) in a centralised setup where the central agent receives the explanatory variables from all its contracted agents.

At each iteration k, the algorithm uniformly samples a training example $i \in \{1+L,\ldots,T\}$. Then, the gradient of the squared loss function C is estimated with

$$\nabla C\left(\hat{\boldsymbol{\beta}}_1^{(k-1)},\ldots,\hat{\boldsymbol{\beta}}_S^{(k-1)}\right) = 2\left(\sum_{s\in\Omega_S}\boldsymbol{a}_{s,i-1}\right)^{\top}\left(\sum_{s\in\Omega_S}\boldsymbol{a}_{s,i-1}\hat{\boldsymbol{\beta}}_s^{(k-1)} - y_{s_j,i}\right) \tag{26}$$

where $\hat{\boldsymbol{\beta}}_s^{(k-1)}$ are the estimated model coefficients of the previous iteration and agent s. Next, the estimated gradient is used in

$$\widetilde{\boldsymbol{\theta}}_s^{(k)} = \boldsymbol{\theta}_s^{(k-1)} - \eta\nabla C\left(\boldsymbol{\beta}_s^{(k-1)}\right), \quad \forall s \in \Omega_S \tag{27}$$



The *central agent* receives y_t. Then the forecast error $r_{t|t-1}^{(\eta)}$ and the cumulative absolute error $CAE_t^{(\eta)}$ are computed for all $\eta \in \Omega_\eta$

The central agent uses the key $\mathbb{K}$ to encrypt all $r_{t|t-1}^{(\eta)}$ and shares them then with the fusion center operator

The *fusion center operator* forwards the encrypted forecast errors to all contracted agents

After decrypting the forecast errors with $\mathbb{K}^*$, *all agents* compute the dual variables $\boldsymbol{\theta}_{s,t}^{(\eta)}$ for all $\eta \in \Omega_\eta$

All agents share all dual variables $\boldsymbol{\theta}_{s,t}^{(\eta)}$ with the fusion center operator

The *fusion center operator* computes the denominator of the link function, $\gamma_t^{(\eta)}$, for all $\eta \in \Omega_\eta$

The *fusion center operator* shares the denominators with all agents

All agents compute $\widehat{\boldsymbol{\beta}}_{s,t}^{(\eta)}$ *and* $\tilde{y}_{s,t+1|t}^{(\eta)}$ for all $\eta \in \Omega_\eta$

All contracted agents share all $\tilde{y}_{s,t+1|t}^{(\eta)}$ with the fusion center operator

The *fusion center operator* forwards all partial predictions to the central agent

The central agent computes for all $\eta \in \Omega_\eta$ $\hat{y}_{s,t+1|t}^{(\eta)} = \sum_{s \epsilon \Omega_S} \tilde{y}_{s,t+1|t}^{(\eta)}$. The prediction $\hat{y}_{s_j,t+1|t}$ is obtained as $\min_\eta CAE_t^{(\eta)}$

Fig. 4. Flowchart for the Adaptive Distributed MIrror Descent Algorithm made Sparse (Adaptive D-MIDAS) approach for online distributed learning applied to wind power forecasting.

to update the dual variables, where $\eta > 0$ is a fixed learning rate. The SMIDAS then applies the truncation step:

$$\theta_{s,j}^{(k)} = \text{sign}\left(\widetilde{\theta}_{s,j}^{(k-1)}\right) \max\left(0, |\widetilde{\theta}_{s,j}^{(k-1)}| - \eta\lambda\right), \quad \forall j \in \{1, \ldots, L\}, \forall s \in \Omega_S \tag{28}$$

where the regularisation strength λ pulls the dual variables towards 0. As soon as a coefficient crosses 0, it is truncated to 0. This procedure is conceptually the same as in Langford et al. (2009), though applied to the dual variables of the mirror descent instead of to the primal variables of the stochastic gradient descent. The last step of the SMIDAS applies the p-norm link function

$$\hat{\boldsymbol{\beta}}_1^{(k)}, \ldots, \hat{\boldsymbol{\beta}}_S^{(k)} = f^{-1}\left(\boldsymbol{\theta}_1^{(k-1)}, \ldots, \boldsymbol{\theta}_S^{(k-1)}\right) \tag{29}$$

to update the model coefficient estimates.

Eqs.(26) to (29) are applied until a defined convergence criterion is reached. Depending on the dataset size and convergence criterion, the algorithm can sample a single training example multiple times.

4.3. Online distributed MIDAS

The motivation to use the SMIDAS as the basis for our distributed online algorithm comes from the separability of (26) in the explanatory variables and the possibility of obtaining sparse model coefficient vectors through the truncation step (28). By choosing the loss function C as the quadratic criterion, the term $\sum_{s\in\Omega_S} \boldsymbol{a}_{s,i-1}\hat{\boldsymbol{\beta}}_s^{(k-1)} - y_{s_j,i}$ in (26) is the one-step-ahead forecast error of iteration k. Hence, if the central agent shares the forecast error for its site with the contracted agents, each agent is able to estimate their share of the gradient locally. Given the remaining steps of the SMIDAS, this allows, with only a few modifications, the obtainment of a distributed online algorithm where the privacy of each agent is protected. However, one might argue that the forecast error is private information for the central agent, who will hence be unwilling to share it. In situations where the central agent is unwilling to share forecast errors with competitors, we propose the following distributed learning network design. When sharing the forecast error through a fusion centre with contracted agents, the forecast error is anonymised such that the contracted agents cannot infer the identity of the central agent, and hence cannot identify the location of the related site. The anonymisation of the central agent would further require that potential compensations for the participation in a learning agreement are handled by the fusion centre operator.

To obtain a learning algorithm that is able to track time-varying model coefficients, our algorithm does not randomly sample training examples. Instead, the algorithm estimates the gradient for a sample only once as observations arrive sequentially. Therefore, the index t replaces i in all previous SMIDAS formulations. We further change the name of the algorithm to MIDAS, because we remove the stochasticity by not using random samples. Finally, we remove the iteration counter k (t is the equivalent in online learning) from all formulations. Estimating the gradient of a sample only once is fundamentally different from the OADMM, where the cumulative loss is minimised over all past observations. This means that, when updating the model coefficients, all past information is implicitly considered. Consequently, when using the distributed MIDAS version, we expect greater variance to the estimated model coefficients.

Starting from the willingness of the central agent to share its forecast error $r_{t|t-1} = \hat{y}_{s_j,t|t-1} - y_{s_j,t}$ with the contracted agents at time t, we propose the following distributed MIDAS. Instead of sharing the forecast error directly with the contracted agents, it is shared through a fusion centre. This is considered the first broadcasting step. The agents then update their local dual variables by taking a step into the direction of the negative estimated gradient while controlling the step size with the learning rate η. The update is performed by all agents in parallel and is written as

$$\widetilde{\boldsymbol{\theta}}_{s,t} = \boldsymbol{\theta}_{s,t-1} - \eta \boldsymbol{a}_{s,t-1} r_{t|t-1} \tag{30}$$

The SMIDAS subsequently uses the element-wise truncation (28). A simulation study revealed that a single sample evaluation does not yield significant benefits in terms of forecast accuracy. Furthermore, we realised that the p-norm link function was sufficient to shrink unimportant model coefficients to 0, though the estimated model coefficients never became exactly 0. Therefore, we dismissed the option to obtain sparse coefficient vectors by neglecting the truncation step in our distributed version (i.e. $\widetilde{\boldsymbol{\theta}}_{s,t}$ is hereafter replaced by $\boldsymbol{\theta}_{s,t}$). We subsequently obtained an algorithm that has one fewer hyperparameter. However, the inability to shrink unimportant model coefficients to 0 is a setback if compared to the case of OADMM.

The next step of the algorithm utilises the fusion centre, with which all agents share their dual variables. This allows the computation of the denominator of the link function with

$$\gamma_t = \|\boldsymbol{\theta}_t\|_p^{p-2} \tag{31}$$

where $\boldsymbol{\theta}_t$ is the assembly of all local dual vectors $\boldsymbol{\theta}_{s,t} \in \Omega_S$ and p is a hyperparameter. This step marks the first gathering step, even though the local variables are not gathered by the central agent. The norm γ_t is consequently shared with all agents such that they can apply the link function to their respective dual variables. This is the second broadcasting step of the algorithm. The final update is the element-wise application of the link function:

$$\hat{\boldsymbol{\beta}}_{s,t} = \frac{\text{sign}\left(\boldsymbol{\theta}_{s,t}\right)|\boldsymbol{\theta}_{s,t}|^{p-1}}{\gamma_t} \tag{32}$$

The aforementioned shrinkage behaviour of the link function is controlled via the hyperparameter p, where a greater value in p applies a greater shrinkage to all $\hat{\boldsymbol{\beta}}_{s,t}$'s.

After obtaining re-estimated model coefficients, each agent calculates a new partial prediction and shares it through the fusion centre with the central agent, who eventually calculates the next prediction for its site. The final exchange of information accounts for the second gathering step. In total the algorithm requires two broadcasting and two gathering steps for each t.

4.4. Extending the distributed MIDAS

With the distributed MIDAS version, the fusion centre operator could retrieve information about local explanatory variables. This possibility exists since the access to all dual variables and the forecast errors results in an equal amount of equations and unknowns. Therefore, additional measures are required to protect the data of the wind farm operators. Due to the different structure of the algorithms, introducing an encryption matrix, as in the OADMM, was unsuccessful. Our proposal is therefore to encrypt the forecast errors using an encryption technique such as AES (Li, Chen, Liu, & Wan, 2009), and then share it through the fusion centre with the contracted agents. This requires the direct exchange of the decryption key with the contracted agents before all agents start performing online learning. Because the fusion centre operator no longer has access to the forecast error, it has more unknowns than equations to solve. Hence, it is not possible to retrieve the explanatory variables with sufficient accuracy. In a setting with anonymised forecast errors, the central agent still shares the decryption key with its contracted agents. However, the central agent does not reveal its identity and therefore the contracted agents cannot obtain information about the location of the central agent's site.

In the presented algorithm, named Distributed MIDAS (D-MIDAS), the learning rate η controls the general speed with which the algorithm approaches the global optimum of the minimisation problem within a given period. When η is large, the global optimum is approached faster. At the same time, however, the estimated model coefficients experience greater variance between consecutive timestamps. This statement is derived from (30), where a large forecast error translates directly to a significant change in the dual variables, and subsequently to a notable change in the model coefficients. Ideally, in stationary periods the algorithm requires smaller η values compared to non-stationary periods. Taking into account that all algorithm parameter updates are computationally cheap, our proposal is to learn multiple AR-X models in parallel while varying the learning rate η between the models.

Based on the past performance of each model, the algorithm adaptively chooses which model to use for the next prediction. This can considered adaptive learning, where in stationary periods a small η is used, and in non-stationary periods a larger η is applied instead. We use the cumulative absolute error (CAE) with decaying weights

$$CAE_t^{(\eta)} = \mu CAE_{t-1}^{(\eta)} + |\hat{y}_{s_j,t|t-1}^{(\eta)} - y_{s_j,t}| \tag{33}$$

to evaluate the performance of each model, where μ controls the level of decay. The superscript η indicates which model the prediction is coming from. We name this extension Adaptive D-MIDAS and its pseudocode is shown in Appendix.

The computational complexity and the amount of exchanged data increases linearly with the number of models that the Adaptive D-MIDAS learns in parallel. Concerning the amount of exchanged data, the Adaptive D-MIDAS exchanges almost the same amount of data as the OADMM when learning two models in parallel. However, the Adaptive D-MIDAS requires two bidirectional data exchange steps, whereas the OADMM requires only one. The additional data exchange step comes from the requirement to compute the denominator of the link function at the fusion centre. Based on this insight, we obtained a communication-reduced version of the algorithm by using the denominator and dual variables of the previous timestamp to update the model coefficients via the link function. Consequently, it is no longer necessary to send the dual variables to the fusion centre before updating the weight vector. The denominator of the link function can instead be calculated after the central agent has calculated the next prediction for its site. With this strategy to reduce the overall time between obtaining the newest observation and calculating a new prediction with re-estimated model coefficients, it is expected that a negative impact on forecast accuracy will be observed. However, as revealed by the following case study using real-world data, the reduction in forecast accuracy is small.

5. Simulation study

A study on simulated data investigated the ability of both algorithms to estimate time-varying model coefficients and the related computational costs. We only considered the standard Adaptive D-MIDAS, and not its communication-reduced version, since the lagged calculation of the denominator γ_t was verified to have only a small impact on the estimated model coefficients.

5.1. Tracking of time-varying coefficients

We first generated a multivariate time series with time-varying coefficients of the form

$$\boldsymbol{y}_t = \boldsymbol{A}\boldsymbol{y}_{t-1} + \boldsymbol{\epsilon}_t \tag{34}$$

where $\boldsymbol{\epsilon}_t$ is a vector of independent standard Gaussian noise with 0 mean and finite variance (set to 0.1 in our experiments). Each simulated time series had 25 000 time-steps. The coefficient matrix $\boldsymbol{A}$ is defined as given in Box I, where a_1 and a_2 are time-varying coefficients (as illustrated in Fig. 5). We performed a Monte-Carlo simulation with 1000 replicates to estimate the variance of $\hat{a}_1$ and $\hat{a}_2$. The hyperparameters p and μ of the Adaptive D-MIDAS were set to 2.5 and 0.996, respectively. In this experiment we used six evenly spaced (from 0.025 to 0.15) learning rates.

Fig. 5 shows the temporal evolution of the mean, as well as the $5th$ and $95th$ quantiles of the estimated coefficient distributions for a_1 and a_2. Both algorithms were able to track the time-varying coefficients in expectation (the mean of the estimated coefficient distributions follows the true values). However, some noticeable differences were observed between both algorithms. First, the Adaptive D-MIDAS had a greater "burn-in" period, i.e., the number of samples required to learn from before a fair approximation of the true coefficient value is reached. Second, the spread in the estimated model coefficients for the 1000 replicates (represented by the difference between both quantiles) increased for the Adaptive D-MIDAS when the true coefficients changed. Contrary to

$$\boldsymbol{A} = \begin{bmatrix} 0.9 & 0 & 0 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.8 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.85 & 0 & 0 & 0 & 0 & -0.15 & 0 & 0 \\ 0 & 0 & 0 & 0.75 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_1 & 0.2 & 0 & 0 & 0 & 0 \\ 0 & a_2 & 0 & 0 & 0 & 0.9 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.85 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.1 & 0 & 0 & 0 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.9 & 0 \\ 0 & 0 & 0 & 0 & 0.15 & 0 & 0 & 0 & 0 & 0.8 \end{bmatrix} \tag{35}$$

Box I.

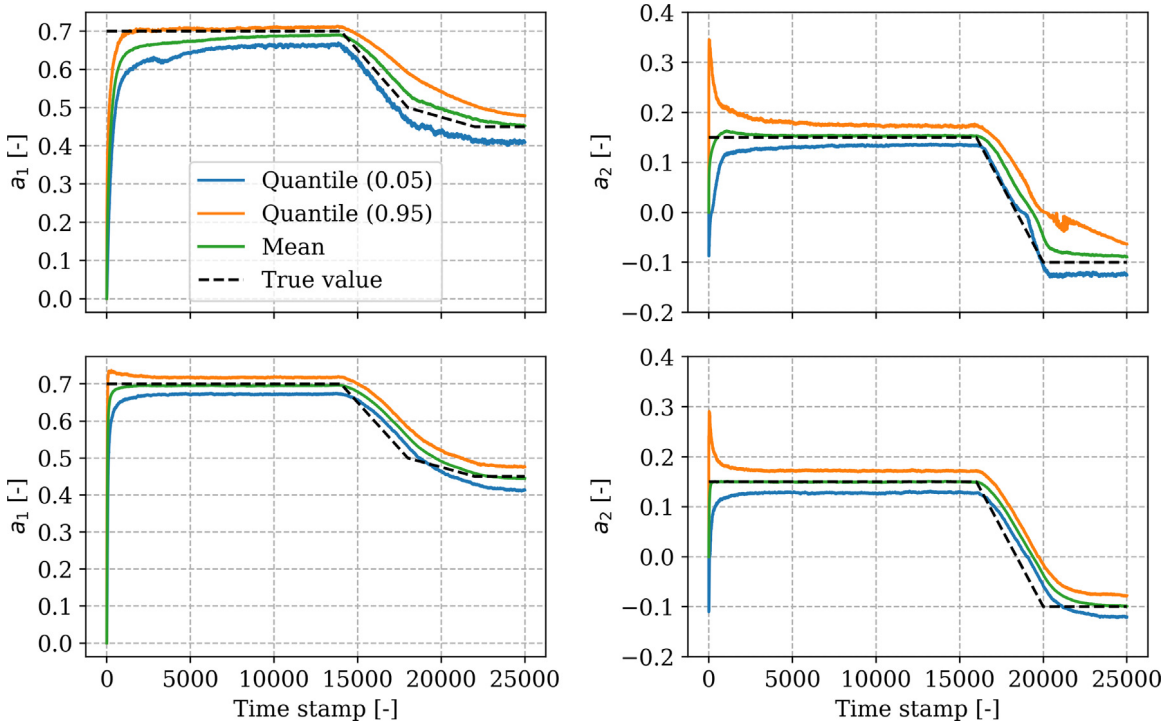


Fig. 5. Coefficient estimates obtained through the Monte-Carlo simulation. Top row: Adaptive D-MIDAS, bottom row: OADMM.

this, the spread in coefficient estimation for the OADMM was either constant or even decreased for changing true coefficient values. The increased spread for the Adaptive D-MIDAS coefficient estimates was a consequence of the faster learning rate, which is required in order to keep track of the decreasing true coefficient value.

This can also be observed from Fig. 6, which shows the average (over all 1000 Monte-Carlo replicates) learning rate of the Adaptive D-MIDAS. There is a clear relationship between the spread in the coefficient estimates and the best-performing learning rate (based on (33)).

Thus, there is a clear trade-off for the Adaptive D-MIDAS between the variation in the estimated model coefficients and the ability to track time-varying coefficients: when trying to achieve a high degree of adaptivity, one has to pay the price of higher variation in the estimated coefficients.

A similar trade-off was observed for the OADMM, where adaptivity is controlled by the forgetting factor ν. When applying a smaller ν value, the algorithm became more adaptive, but since the effective training data length decreased, the variance in the estimated coefficient increased. However, the OADMM provided a better trade-off between adaptivity and estimated coefficient variance due to the fact that it minimised the cumulative loss over all past observations. Thus, occasional outliers in the form of large forecast errors had a smaller impact on the estimated coefficients.

5.2. Computational costs

Besides assessing the ability of both algorithms to track time-varying model coefficients, a simulation study

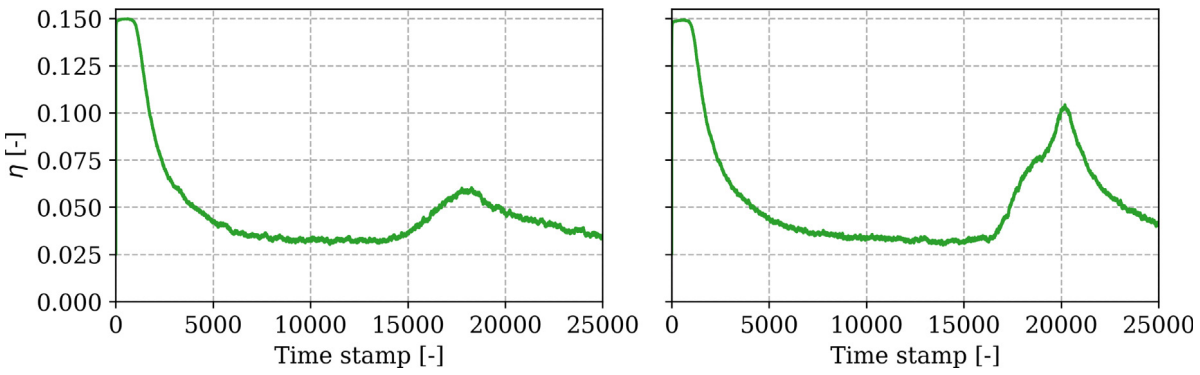


Fig. 6. Average learning rate of the Adaptive D-MIDAS across all 1000 replicates. Left a_1 right a_2.

was performed to compare the computational costs. The study estimated the time required by each agent and algorithm to calculate a new prediction after the central agent obtained a new data sample. To show the expected better scaling properties of the D-MIDAS, we estimated the computational time for an increasing learning network size of contracted agents. We again used simulated time series data that were generated with the previously introduced approach. However, instead of creating an AR(1)-, we used an AR(4)-process. In this study we performed online learning for 1000 simulated time-steps while recording the time it took to complete each operation. To achieve comparability between the Adaptive D-MIDAS and OADMM, the Adaptive D-MIDAS learned two models in parallel. For both algorithms, this resulted in an almost equal amount of data that was exchanged within the learning network. The simulation study was performed on a system with an i5-5200U CPU, 8 GB DDR3 RAM and a Windows 10 OS. Because both algorithms were used locally, we neglected the encryption and decryption steps of the Adaptive D-MIDAS. Hence, the observed performance gap in computational speed would decrease in case encryption was required to ensure data privacy.

Computational times are summarised in Fig. 7. They were obtained by averaging the computational time for each of the 1000 time-steps. The Adaptive D-MIDAS was faster than the OADMM overall. Furthermore, the algorithm showed better scaling behaviour with respect to the learning network size. This was expected since, at each and every time t, the OADMM needed to solve a linear system of equations. Depending on the solving technique, complexity grew at least quadratically with the number of equations. The Adaptive D-MIDAS scaled better because its updates were simple linear operations.

6. Case study

Our distributed and online learning algorithms were benchmarked on a real-world dataset of wind power generation for 311 sites. The dataset was a subset of the one used in Girard and Allard (2013). The temporal resolution was 15 min and our subset covered 40 000 time-steps, corresponding to 416 days. Fig. 8 shows the location of the sites in western Denmark. Many sites are located in close proximity to each other. This allows accounting for relevant spatiotemporal patterns when forecasting wind power generation for short lead times. To highlight the benefits of online learning, we benchmarked the forecasts from our online algorithms against those that would be obtained from L_1-regularised AR-X models with time-invariant coefficients. The model coefficients were then estimated on the training part of the dataset only.

The benefits of exploring spatiotemporal patterns in wind power generation data have been shown for a different subset in Messner and Pinson (2019). The authors showed that high-dimensional regularised AR-X models outperform univariate AR models that only use on-site power measurements. All model coefficients were estimated in a time-varying fashion. Based on these results, we allowed ourselves to disregard univariate AR models with time-varying coefficients in our case study.

6.1. Data preprocessing

First, the raw data was normalised by dividing the time series of each site by the respective nominal capacity. We write $x_{s,t}$ for the normalised wind power generation observed at time t and for site s. In addition, a logit-Normal transformation of the original time series was considered, as proposed by Lau and McSharry (2010). That is, at each and every time t and site s,

$$y_{s,t} = \ln\left(\frac{x_{s,t}}{1 - x_{s,t}}\right), \quad \forall s, t\ . \tag{36}$$

To account for the bound effects, a coarsening approach was used (Pinson, 2012), for which values of 0 and 1 were set to 0.01 and 0.99, respectively.

6.2. Case study setup

The data was split into two equal sub-periods of 20 000 time-steps. The first part was used for training and hyperparameter optimisation, and the second for genuine out-of-sample forecast verification. Over the first period, the hyperparameters of all algorithms were optimised with a grid search scheme. After identifying suitable hyperparameters for the Adaptive D-MIDAS, the same hyperparameters were then applied to its communication-reduced version. The LASSO's L_1-regularisation parameter

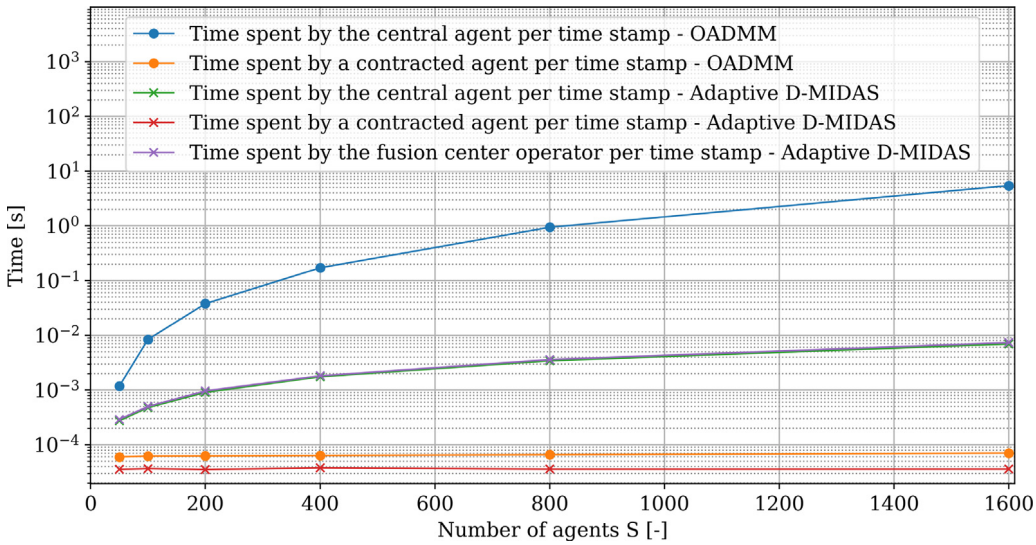


Fig. 7. Average time (over 1000 time-steps) required by each agent to complete its tasks at a given time-step, for both OADMM and Adaptive D-MIDAS approaches, as a function of total number of agents S.

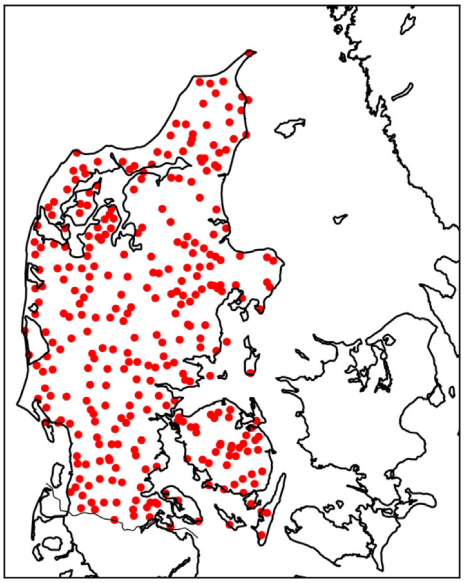


Fig. 8. Location of sites in Western Denmark.

λ for the batch AR-X models was determined through one-fold cross-validation. For both approaches, the forgetting factors are to be seen as variables that control how much of the past data is used for estimation. Hence, optimising these forgetting factors through cross-validation is to be seen as equivalent to determining an optimal training set size in the case of batch learning.

For a given site s, the Mean Absolute Error (MAE),

$$\text{MAE} = \frac{1}{T}\sum_{t=1}^{T}|y_{s,t} - \hat{y}_{s,t}| \tag{37}$$

and the Root Mean Squared Error (RMSE),

$$\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(y_{s,t} - \hat{y}_{s,t}\right)^2} \tag{38}$$

were used as performance metrics. To further quantify the performance of each model, the skill score I

$$I_S = 1 - \frac{S_{\text{Model}}}{S_{\text{Pers}}} \tag{39}$$

was used to assess the improvement over persistence forecasts (the latest observed power measurement is used

as the next prediction), where S could be the metrics MAE or RMSE.

Owing to the large number of wind farms in the dataset, hyperparameter optimisation was performed for all wind farms at once, instead of for each site individually. A set of hyper-parameters was evaluated by considering the skill score distribution that contained the scores of all 311 sites. The median, the lower quartile and the inter-quartile range were used here as decision criteria. It was further decided to perform multi-step-ahead forecasting with up to four steps ahead. In general, different strategies exist for multi-steps-ahead forecasting, where a good overview is presented in Ben Taieb, Bontempi, Atiya, and Sorjamaa (2012). The most common approaches are either the iterative calculations of one-step-ahead predictions or training separate models for each lead time. The iterative calculation of one-step-ahead predictions results in the accumulation of forecast errors. Therefore, we used the direct approach and trained models for each lead time.

The hyperparameters of the Adaptive D-MIDAS and the batch AR-X model were optimised for each of the four lead times. Based on the significantly longer simulation times, the hyperparameters of the OADMM were optimised for one-step-ahead predictions only. The selected hyperparameters were then applied for all other lead times. Due to the observed longer burn-in period of the Adaptive D-MIDAS, the performance metrics were calculated for both online algorithms only for the time-steps between $t = 10\,000$ and $t = 20\,000$.

To obtain predictions for all sites of the dataset, each site took the role of the central agent once, while acting as a contracted agent in the other 310 simulations (i.e., for all other sites). Therefore, to obtain predictions for all sites and a single lead time, in total 311 AR-X models were estimated.

6.3. Results

Focusing first on hyperparameter optimisation, Fig. 9 gives an example of the results obtained by optimizing the forgetting factor μ for the Adaptive D-MIDAS approach, when performing 1-step ahead forecasting. The boxplot shows the improvement over persistence forecasts for all 311 sites, as a function of the forgetting factor μ. Each box extends from the lower to the upper quartile (denoted Q_1 and Q_3, respectively), where the horizontal line indicates the median of the obtained RMSE skill score distributions. The maximum length of the whiskers was set to 1.5 times the interquartile range (Q_3–Q_1). The upper whisker then indicated the last sample which is found to be below or equal to the threshold of $Q_3 + 1.5(Q_3 - Q_1)$. If a data point of the distribution was found outside this range, it was classified as an outlier and marked with a circle. The same concept applied to the lower whisker, which marked the first sample that was found to be within the range of $Q_1 - 1.5(Q_3 - Q_1)$.

A μ value of 0.95 performed slightly better than the remaining selected values when considering the aforementioned decision criteria. Therefore, this value was subsequently selected when performing online learning to estimate the performance on unseen data. The same approach was followed when tuning the other hyperparameters.

After finding suitable hyperparameters for both online algorithms, online learning was performed on the complete dataset. In contrast, the batch AR-X model coefficients were estimated over the first 20 000 time-steps and then used to generate predictions over the remaining 20 000 time-steps without re-estimating the model coefficients. Results are collated in Fig. 10 for all approaches considered and for all sites, again with boxplots for skill score values (both in terms of RMSE and MAE).

Overall, all online distributed learning algorithms outperformed the batch learning one (LASSO estimation in AR-X models), with the advantage that no data from contracted agents was actually shared with the central agents. A paired t-test supported the statistical significance by rejecting the null hypothesis of equal means at the 0.05 significance level. The number of outliers for the batch LASSO additionally emphasised the strength of online learning because the poor performance of batch estimation could be explained by the non-stationarity of the wind power generation time series. Hence, there were significant differences between the time-varying coefficients throughout the value period, and the coefficients estimated over and fixed at the end of the training period. Furthermore, when computing the bias it was observed that all forecasting models exhibited negligible bias values (not shown here).

The results further showed that OADMM, despite being computationally more expensive, outperformed the Adaptive D-MIDAS for all lead times and skill scores. A paired t-test also supported the statistical significance of the results here. In addition, the performance gap increased for further lead times. This may be due to the structure and workings of both online algorithms. Indeed, the OADMM minimised the cumulative loss over all past observations where the covariance structures $\boldsymbol{H}_t$ and $\boldsymbol{p}_t$ carried the information of all previous samples. By varying the applied forgetting factor the number of past samples that were effectively used to update all algorithm parameters was controlled. As a result, even when a set of successive large forecast errors were observed, the estimated model coefficients were less subject to variation. The Adaptive D-MIDAS on the other hand does not utilise covariance structures to estimate model coefficients. Instead it uses the estimated gradient of the current squared loss between the observation and prediction to re-estimate the model coefficients. Therefore, large forecast errors directly translate to noticeable variations in the estimated model coefficients. A resulting shortcoming may be that the algorithm could be highly sensitive to outliers and structural breaks in the time series. While for one-step-ahead predictions the model coefficient update is performed right after, i.e. naturally one time-step after the prediction is made. For greater lead times there is a time lag because one must wait k steps to obtain the forecast error of a k-steps-ahead forecast. This, paired with the sensitivity with respect to large forecast errors, may explain the lower forecast accuracy of the Adaptive D-MIDAS for further lead times. As mentioned above with

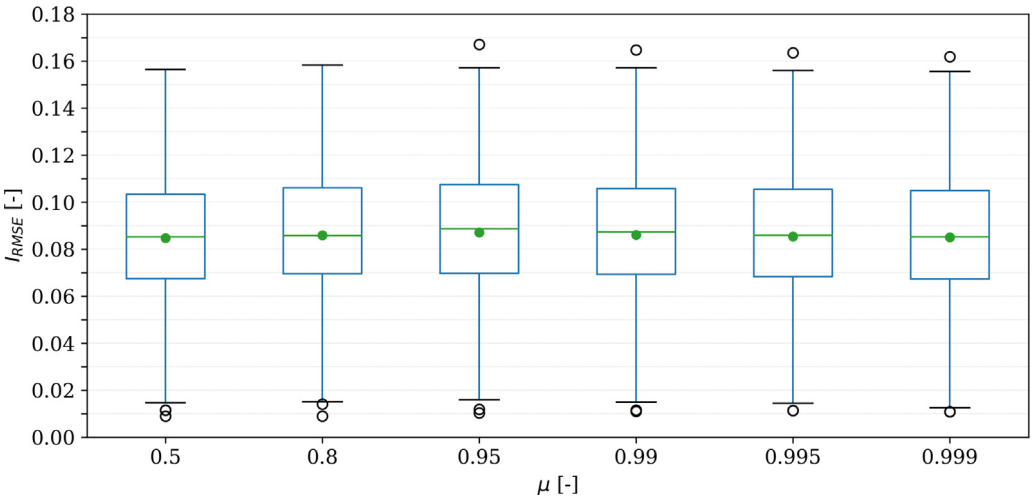


Fig. 9. RMSE skill score of the Adaptive D-MIDAS with reference to the persistence forecast for 1-step ahead forecasts, as a function of the forgetting factor μ. The skill score values were computed for the timestamps t =10 000 to t =20 000.

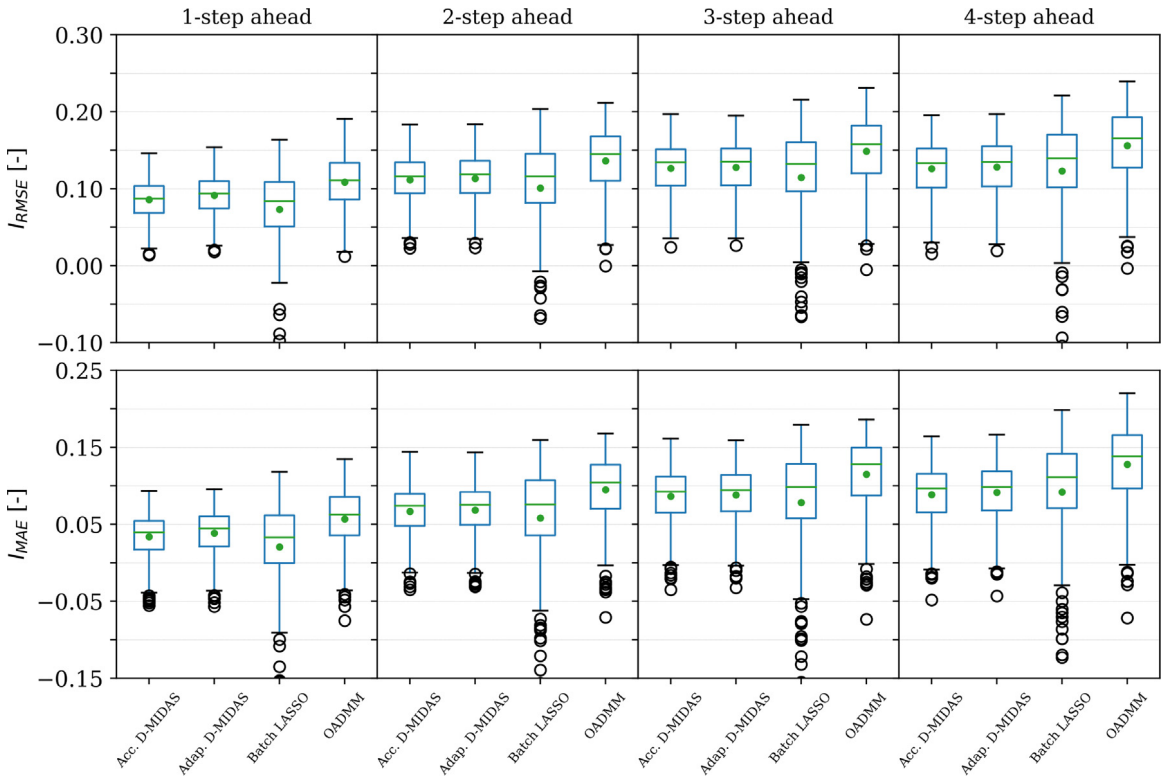


Fig. 10. RMSE (top row) and MAE (bottom row) skill scores for the online distributed and batch learning approaches for different lead times. The skill score values were computed over the evaluation period, from t =20 000 to t =40 000. The dots indicate the mean of the skill score distributions.

regard to the simulation, the OADMM is essentially better with this condition since the covariance structures carry inertia, whereby single or multiple large forecast errors do not affect the model coefficient estimates as much. Here it should be noted that this statement only holds for a sufficiently large forgetting factor.

Finally, the communication-reduced Adaptive D-MIDAS version (Acc. D-MIDAS) did not perform significantly worse than the standard version. Thus, this version is an viable alternative for applications where there is a high priority for re-estimating the model coefficients as quickly as possible.

7. Conclusions

Two novel online distributed learning algorithms, the OADMM and Adaptive D-MIDAS, were proposed for high-dimensional AR-X model coefficient estimation to be used in wind power forecasting. The distributed component of both algorithms enables the estimation of AR-X models without the necessity of sharing sensitive data, such as power measurements, directly with other agents or entities. This enables competing wind farm operators to cooperate and collectively improve the forecasts for their sites. Moreover, the online component allows the estimated model coefficients to follow the time-varying conditions of wind power generation time series. Our main focus has been on distributed learning and forecasting for a class of linear models. Obviously then, the quality of the forecasts obtained is linked to the relevance of such linear models in practice. In view of the literature on short-term wind power forecasting, AR-X models are highly relevant for the lead times and forecasting setups considered in the paper. Some relevant generalisation could readily be considered, e.g. to regime-switching models such as Self-Exciting Threshold Auto-Regressive (SETAR) models and Smooth Transition Auto-Regressive (STAR) models. As long as the models involved are linear and separable, the methods discussed in the paper could be used in a similar manner. More broadly though, generalisation to nonlinear and more complex models may be more involved.

The OADMM relies on a LASSO-type objective function to estimate the coefficients of regularised AR-X models, in combination with an exponential forgetting factor to control the level of adaptivity. The algorithm minimises at any time t the cumulative loss over all observed samples (up to t), which requires solving a linear system of equations to update the algorithm parameters. Due to the non-negligible time for solving large equation systems, the Adaptive D-MIDAS was subsequently introduced. Owing to its design, all parameter updates are computationally cheaper to obtain. The algorithm is based on a mirror descent method where the gradient of the current squared forecast error is used to update dual variables. These are then mapped via a link function to the AR-X model coefficients. In addition, an accelerated version of the Adaptive D-MIDAS was proposed, i.e. a communication-reduced version. The algorithm achieves faster model coefficient re-estimates by using the dual variables from the previous timestamp. We verified that the impact on forecast accuracy is small.

A study on simulated data verified the ability of both algorithms to track time-varying model coefficients. However, the OADMM approach offers a better trade-off between adaptivity and the limited variability of the estimated model coefficients than the Adaptive D-MIDAS approach. Owing to its design, by minimising the cumulative loss over all past samples, large forecast errors do not directly cause large variations in the model coefficient estimates. The better controllability between adaptivity and the estimated model coefficient variance was the reason why the OADMM achieved more accurate forecasts than the Adaptive D-MIDAS in a case study with a real-world dataset of 311 wind farms. The case study additionally confirmed that online learning is superior to offline learning, as already supported by previous work, although based on centralised learning algorithms.

Future work should address strategies to reduce the greater variability in the estimated model coefficients of the Adaptive D-MIDAS. Since we only considered deterministic forecasting models, future work should investigate extensions to the online distributed learning algorithms for the case of probabilistic forecasting. Then, alongside other proposals for distributed online learning, and to relax the assumption such that agents are willing to collaborate truthfully and rationally, it may be crucial to investigate federated learning and data markets. These new concepts may incentivise and support improvements in forecast quality when relevant data and features are distributed, both geographically and in terms of ownership.

Acknowledgment

The work was partly funded by the Danish Innovation Fund and the ForskEL programme through the project CITIES (DSF-1305-00027B) and The Energy Collective (Grant 2016-1-12530). The authors would like to acknowledge Energinet.dk for providing the wind power dataset used as input to the empirical investigation. The authors are finally grateful to two reviewers and one associate editor, who comments and suggestions contributed to improving the contents and presentation of the paper.

Appendix. Algorithms

Algorithm 1 Online ADMM

1: Central agent decides on λ, ρ, ν and L. Initialise $\hat{\boldsymbol{\beta}}_{s,0}$, $\boldsymbol{z}_{s,0}$ and $\boldsymbol{u}_{s,0}$ for $s \in \Omega_s$, $\boldsymbol{H}_0$, $\boldsymbol{P}_0$ and t to be 0. To build $\boldsymbol{M}^\top\boldsymbol{M}$, contracted agents share $\boldsymbol{M}_s^\top\boldsymbol{M}_s$ with the central agent j.
2: **while** agents want to perform distributed online learning **do**
3: $t := t + 1$
4: $y_{s_j,t}$ is revealed to the central agent
5: $\boldsymbol{H}_t := \nu\boldsymbol{H}_{t-1} + (\boldsymbol{a}_{t-1}\boldsymbol{M})^\top(\boldsymbol{a}_{t-1}\boldsymbol{M})$
6: $\boldsymbol{p}_t := \nu\boldsymbol{p}_{t-1} + y_{s_j,t}(\boldsymbol{a}_{t-1}\boldsymbol{M})$
7: Central agent updates $\boldsymbol{z}_{t-1}$ and distributes local values to contracted agents
8: $\left(\boldsymbol{H}_t + \rho\boldsymbol{M}^T\boldsymbol{M}\right)\boldsymbol{z}_t = \boldsymbol{p}_t + \rho\boldsymbol{M}^\top(\hat{\boldsymbol{\beta}}_{t-1} + \boldsymbol{u}_{t-1})$
9: $\left[\boldsymbol{z}_{s_1,t}, ..., \boldsymbol{z}_{s_S,t}\right] := \boldsymbol{z}_t$
10: **for** $s \in \Omega_s$ **do**
11: $\boldsymbol{u}_{s,t} := \boldsymbol{u}_{s,t-1} + \hat{\boldsymbol{\beta}}_{s,t-1} - \boldsymbol{M}_s\boldsymbol{z}_{s,t}$
12: $\hat{\boldsymbol{\beta}}_{s,t} := \mathbb{S}_{\lambda/\rho}\left(\boldsymbol{M}_s\boldsymbol{z}_{s,t} - \boldsymbol{u}_{s,t}\right)$
13: Agent s uses latest observation $y_{s,t}$ to form $\boldsymbol{a}_{s,t}$
14: $\widetilde{y}_{s,t+1|t} := \boldsymbol{a}_{s,t}\hat{\boldsymbol{\beta}}_{s,t}$
15: share $\widetilde{y}_{s,t+1|t}$, $\boldsymbol{a}_{s,t}\boldsymbol{M}_s$ and $\boldsymbol{M}_s^\top\left(\hat{\boldsymbol{\beta}}_{s,t} + \boldsymbol{u}_{s,t}\right)$ with central agent
16: **end for**
17: $\hat{y}_{s_j,t+1|t} := \sum_{s\in\Omega_s}\widetilde{y}_{s,t+1|t}$
18: Central agent j stacks local $\boldsymbol{a}_{s,t}\boldsymbol{M}_s$ and $\boldsymbol{M}_s^\top\left(\hat{\boldsymbol{\beta}}_{s,t} + \boldsymbol{u}_{s,t}\right)$
19: $\boldsymbol{a}_t\boldsymbol{M} := \left[\boldsymbol{a}_{s_1,t}\boldsymbol{M}_{s_1}, ..., \boldsymbol{a}_{s_S,t}\boldsymbol{M}_{s_S}\right]$
20: $\boldsymbol{M}^\top\left(\hat{\boldsymbol{\beta}}_t + \boldsymbol{u}_t\right) := \left[\boldsymbol{M}_{s_1}^\top\left(\hat{\boldsymbol{\beta}}_{s_1,t} + \boldsymbol{u}_{s_1,t}\right), ..., \boldsymbol{M}_{s_S}^\top\left(\hat{\boldsymbol{\beta}}_{s_S,t} + \boldsymbol{u}_{s_S,t}\right)\right]$
21: **end while**

Algorithm 2 Adaptive D-MIDAS

Central agent creates decryption key $\mathbb{K}^*$, selects a set of learning rates (collected in Ω_η) and shares both quantities with its contracted agents.
Initialise t and $\hat{\boldsymbol{\beta}}_{s,0}^{(\eta)}$, $\boldsymbol{\theta}_{s,0}^{(\eta)}$ for $s \in \Omega_s$ and $\eta \in \Omega_\eta$ to be 0. Additionally, initialise $CAE_{s_j}^{(\eta)}$ for $\eta \in \Omega_\eta$ to be 0. Central agent selects μ and p but only shares p with the fusion centre and contracted agents.
while agents want to perform distributed online learning **do**
  $t := t + 1$
  $y_{s_j,t}$ is revealed to the central agent
  for $\eta \in \Omega_\eta$ **do**
    $r_{t|t-1}^{(\eta)} := \hat{y}_{s_j,t|t-1}^{(\eta)} - y_{t,s_j}$
    $CAE_t^{(\eta)} = \mu CAE_{t-1}^{(\eta)} + |r_{t|t-1}^{(\eta)}|$
  end for
  Central agent encrypts forecast errors with $\mathbb{K}(\cdot)$ and shares them through the fusion centre with its contract agents
  for $s \in \Omega_s$ **do**
    Agent s decrypts forecast errors with $\mathbb{K}^*(\cdot)$
    for $\eta \in \Omega_\eta$ **do**
      $\boldsymbol{\theta}_{s,t}^{\eta} := \boldsymbol{\theta}_{s,t-1}^{(\eta)} - \eta \cdot \boldsymbol{a}_{s,t-1} r_{t|t-1}^{(\eta)}$
    end for
    Transmit list of dual vectors to fusion centre
  end for
  Fusion centre operator computes denominators of link function, $\gamma_t^{(\eta)}$
  for $\eta \in \Omega_\eta$ **do**
    $\boldsymbol{\theta}_t^{(\eta)} := \left[\boldsymbol{\theta}_{s_1,t}^{(\eta)}, ..., \boldsymbol{\theta}_{s_S,t}^{(\eta)}\right]$
    $\gamma_t^{(\eta)} := \|\boldsymbol{\theta}_t^{(\eta)}\|_p^{p-2}$
  end for
  Fusion centre operator shares $\gamma_t^{(\eta)}$ with all agents
  for $s \in \Omega_s$ **do**
    Agent s uses latest observation $y_{s,t}$ to form $\boldsymbol{a}_{s,t}$
    for $\eta \in \Omega_\eta$ **do**
      $\forall\ k,\ \hat{\beta}_{s,t,k}^{(\eta)} := \frac{\operatorname{sign}(\theta_{s,t,k}^{(\eta)})|\theta_{s,t,k}^{(\eta)}|^{p-1}}{\gamma_t^{(\eta)}}$
      $\tilde{y}_{s,t|t-1}^{(\eta)} := \boldsymbol{a}_{s,t} \hat{\boldsymbol{\beta}}_{s,t}^{(\eta)}$
      Each contracted agent shares partial predictions through fusion centre with central agent
    end for
  end for
  for $\eta \in \Omega_\eta$ **do**
    $\hat{y}_{s_j,t|t-1}^{(\eta)} := \sum_{s \in \Omega_s} \tilde{y}_{s,t|t-1}^{(\eta)}$
  end for
  Central agent selects final prediction according to $\min_\eta MAE_{s_j}^{(\eta)}$
end while

References

Beck, A., & Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimisation. *Operations Research Letters*, *31*, 167–175.

Bekierman, J., & Manner, H. (2018). Forecasting realized variance measures using time-varying coefficient models. *International Journal of Forecasting*, *34*, 276–287.

Ben Taieb, S., Bontempi, G., Atiya, A., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert Systems with Applications*, *39*, 7067–7083.

Bessa, R. J., Miranda, V., Botterud, A., Zhou, Z., & Wang, J. (2012). Time-adaptive quantile-copula for wind power probabilistic forecasting. *Renewable Energy*, *40*, 29–39.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2010). Distributed optimisation and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, *3*, 1–122.

Cavalcante, L., Bessa, R. J., Reis, M., & Browell, J. (2017). Lasso vector autoregression structures for very short-term wind power forecasting. *Wind Energy*, *20*, 657–675.

Dowell, J., & Pinson, P. (2016). Very-short-term probabilistic wind power forecasts by sparse vector autoregression. *IEEE Transactions on Smart Grid*, *7*, 763–770.

Duchi, J., Shalev-Shwartz, S., Singer, Y., & Chandra, T. (2008). Efficient projections onto the l_1-ball for learning in high dimensions. *International Coference on Machine Learning*, *27*, 2–279.

Giebel, G., & Kariniotakis, G. (2017). Wind power forecasting - a review of state of the art. In G. Kariniotakis (Ed.), *Renewable Energy Forecasting – From Models to Applications* (pp. 59–96).

Girard, R., & Allard, D. (2013). Spatio-temporal propagation of wind power prediction errors. *Wind Energy*, *16*(7), 999–1012.

H., Badi, Baltagi, B. H., Fingleton, B., & Pirotte, A. (2014). Estimating and forecasting with a dynamic spatial panel data model. *Oxford Bulletin of Economics and Statistics*, *76*, 112–138.

He, M., Yang, L., Zhang, J., & Vittal, V. (2014). A spatio-temporal analysis approach for short-term forecast of wind farm generation. *IEEE Transactions on Power Systems*, *29*, 1611–1622.

Kou, P., Gao, F., & Guan, X. (2013). Sparse online warped gaussian process for wind power probabilistic forecasting. *Applied Energy*, *108*, 410–428.

Langford, J., Lihong, L., & Zhang, T. (2009). Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, *10*, 777–801.

Lau, A., & McSharry, P. (2010). Approaches for multi-step density forecasts with application to aggregated wind power. *Annals of Applied Statistics*, *4*(3), 1311–1341.

Li, X., Chen, J., Liu, W., & Wan, W. (2009). An improved AES encryption algorithm. In IET International Communication Conference on Wireless Mobile and Computing (CCWMC 2009), (pp 694–698).

Matamoros, J. Asynchronous online ADMM for consensus problems. In Proc. of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, (pp 5875–5879).

Matos, M., Bessa, R., Botterud, A., & Zhou, Z. (2017). Forecasting and setting power system operating reserves. In G. Kariniotakis (Ed.), *Renewable Energy Forecasting – From Models to Applications* (pp. 279–305).

Mazzi, N., & Pinson, P. (2017). Wind power in electricity markets and the value of forecasting. In G. Kariniotakis (Ed.), *Renewable Energy Forecasting – From Models to Applications* (pp. 259–278).

Messner, J. W., & Pinson, P. (2019). Online adaptive lasso estimation in vector autoregressive models for high-dimensional wind power forecasting. *International Journal of Forecasting*, *35*, 1485–1498.

Min, W., & Wynter, L. (2011). Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research C*, *19*, 606–616.

Møller, J. K., Nielsen, H. A., & Madsen, H. (2008). Time-adaptive quantile regression. *Computational Statistics & Data Analysis*, *52*, 1292–1303.

Nemirovski, A. S., & Yudin, D. B. (1983). *Problem complexity and method efficiency in optimisation*. Wiley.

Paci, L., Gelfand, A. E., & Holland, D. M. (2013). Spatio-temporal modelling for real-time ozone forecasting. *Spatial Statistics*, *4*, 79–93.

Pinson, P. (2012). Very short-term probabilistic forecasting of wind power with generalized logit-normal distributions. *Journal of the Royal Statistical Society C*, *61*, 555–576.

Pinson, P. Introducing distributed learning approaches in wind power forecasting. In Proc. of the 2016 international conference on probabilistic methods applied to power systems.

Pinson, P., & Madsen, H. (2012). Adaptive modelling and forecasting of wind power fluctuations with markov-switching autoregressive models. *Journal of Forecasting*, *31*, 281–313.

Shalev-Shwartz, S., & Tewari, A. (2011). Stochastic methods for l1-regularized loss minimisation. *Journal of Machine Learning Research*, *12*, 1865–1892.

Sigrist, F., Künsh, H. R., & Stahel, W. A. (2012). A dynamic nonstationary spatio-temporal model for short term prediction of precipitation. *Annals of Applied Statistics*, *6*, 1452–1477.

Suzuki, T. Dual averaging and proximal gradient descent for online Alternating Direction Multiplier Method. In Proc. of the 30th international conference on machine learning, vol. 28, pp. 392–400.

Tastu, J., Pinson, P., Kotwa, E., Madsen, H., & Nielsen, H. A. (2011). Spatio-temporal analysis and modelling of short-term wind power forecast errors. *Wind Energy*, *14*, 43–60.

Tastu, J., Pinson, P., Trombe, P. J., & Madsen, H. (2014). Probabilistic forecasts of wind power generation accounting for geographically dispersed information. *IEEE Transaction on Smart Grid*, *5*, 480–489.

Wang, H., & Banerjee, A. Online alternating direction method. In Proc. of the 29th international conference on machine learning.

Zhang, Y., & Wang, J. (2018). A distributed approach for wind power probabilistic forecasting considering spatio-temporal correlation without direct access to off-site information. *IEEE Transactions on Power Systems*, *33*(5), 5714–5726.