IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 1, 2021, accepted July 19, 2021, date of publication July 26, 2021, date of current version August 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3099999

A Self-Adaptive Multikernel Machine Based on Recursive Least-Squares Applied to Very Short-Term Wind Power Forecasting

ERICK C. BEZERRA[1,2], PIERRE PINSON[1], (Fellow, IEEE), RUTH P. S. LEÃO[2], AND ARTHUR P. S. BRAGA[2]

[1]Electrical Engineering Department, Technical University of Denmark, 2800 Lyngby, Denmark
[2]Electrical Engineering Department, Federal University of Ceará, Fortaleza, Ceará 60455-760, Brazil

Corresponding author: Erick C. Bezerra (erick.bezerra@gmail.com)

This work was supported by the Brazilian Program Science without Borders through the Framework Program under Grant 204899/2014-1 and Grant 8499660428694715.

ABSTRACT Wind power has contributed significantly to the increase in electricity generation, but a decision-making tool capable of dealing with its variability and limited predictability is necessary. For this purpose, a novel self-adaptive approach for kernel recursive least-squares machines named multiple challengers is introduced in this work, which is successfully used to produce very short-term wind power forecasts at eight wind farms in Australia. The proposed method is based on a competitive tracking method, and the algorithm deals with some common difficulties of kernel methods, e.g., the increasing kernel matrix size associated with time and memory complexities and the overfitting problem. The proposed method always considers the new information received by the model, thus identifying changes in the time series, avoiding abrupt loss of information and maintaining a controlled number of examples since there is an adaptive selection of the active kernel. It works with the smallest dictionary possible, reducing the probability of overfitting. Five minute-ahead wind power forecasts are produced and evaluated in terms of point forecast skill scores and calibration. The results of the proposed method are compared with those provided by other kernel-based versions of the recursive least-squares algorithm, an online version of the extreme learning machine method, and the persistence time series model. An increase in the number of kernels used in the ensemble system can lead to better results when compared with those of single-kernel models.

INDEX TERMS Multiple kernel learning, online training, renewable energy, wind power forecasting.

I. INTRODUCTION

Wind power continues to receive significant attention throughout the world [1]. In this context, the variability of power production and the restricted control of wind turbines justify the development of wind power forecasting models [2]. This is a key factor in ensuring the successful integration of wind farms into the AC power grid [3]. Thus, the reliable and economical operation of power systems with high wind power penetration demands the utilization of accurate models for this purpose [4].

Wind power forecasting has different time horizons according to its application [5]. The time-scale classification is not strictly defined, but it can be classified into very short-term (few seconds to 30 minutes ahead), short-term (30 minutes to 24 hours ahead), medium-term (24 hours to one week ahead), and long-term (one week to years ahead) [5]–[8].

Wind speed and/or power forecasting methods are broadly classified into three categories: (i) physical models use descriptions of the lower atmosphere, geographical features, and obstacles to predict the flow of wind. Physical models are usually based on numerical weather prediction (NWP) models which predict meteorological variables like wind speed, wind direction, pressure, and other variables, using 3-D spatial and temporal information based on computational fluid dynamic (CFD) models. Wind power forecasts can be obtained based on the performance of a wind turbine or a wind farm using NWP results [9], [10] but require extensive calculations and considerable time [11]. Physical models are

The associate editor coordinating the review of this manuscript and approving it for publication was Mouloud Denai.

of limited practical use in very short-term forecasting due to latency issues, and need accurate initial conditions of wind farms that cannot be always guaranteed [12]. However, physical models can represent weather phenomena, such as forward edge of an advancing mass of air (front) and thunderstorm. (ii) Statistical and machine learning models represent the behavior of wind speed and/or power time series based on the wind or power historical data. NWP input is optional for these models. They are faster during the development period and processing results than physical models, and many have been studied for this purpose, such as autoregressive moving average with exogenous input (ARMAX) [13], autoregressive integrated moving average (ARIMA) [14], neural networks (NNs) [15], support vector machines (SVMs) [16], fuzzy logic [17], [18], and extreme learning machines (ELM) [19], [20]. (iii) Hybrid models, in order to improve forecasting performance, combine different methodologies to take advantage of each method [21], such as weighting-based models, hybrid models with data preprocessing techniques, hybrid models with parameter selection, and optimization techniques and hybrid models with error processing techniques [22]–[26]. Among them, decomposition-based approaches taking advantage of time series decomposition methods have been frequently reported, and the original time series can be decomposed into different subseries and modeled more effectively than the original time series [27].

Reviews related to wind power forecasting are available in [21], [28]–[30]. Giebel *et al.* [28] concluded that for forecast horizons of less than approximately six hours, statistical methods using local information are superior to physical models.

Statistical methods used for wind speed and power prediction are usually linear despite the nonlinear nature of the wind and are typically employed in single sites. Considering this key aspect, the present study aims to investigate a class of learning algorithms named kernel methods, which can provide linear processing of nonlinear features for one and multiple sites. This technique retains the properties of linear processing, such as fast learning algorithms and a unique optimal solution, while making it possible to capture some nonlinearities. Kernel machines combine statistical learning theory to optimize generalization [31], with mathematical programming to find solutions efficiently as well as to improve the similarity measure between points to handle nonlinearity issues [32]. This work addresses a regression problem, where kernel machines consider the fact that observational data can be represented by a linear combination of kernel functions [33]. Kernel methods have been successfully applied in time series prediction [34], wind speed forecasting [35], [36], wind power forecasting [37], electric load forecasting [38], [39], and many other applications.

The main goal of this study is to present a multikernel learning machine model that can deal with regression problems with a smaller and nonstatic dictionary, and achieve better or similar results when compared with their respective single-kernel counterparts. The model fitting procedure is fully data driven, making it ideal for smart grid applications where many generators share a highly interconnected power system and the use of spatial dependence is desirable. The results obtained with different kernel machines are compared in two scenarios, first considering only the temporal aspect of the dataset, while the spatial dependence is analyzed later. The contributions of the proposed study can be regarded as follows: (i) the proposal of a novel multiple kernel learning scheme based on multiple challengers (MC) with adaptive control of the number of kernels used by the predictor; (ii) evaluation of the lifespan controller proposed in [40] is adapted and applied to the proposed multiple kernel learning scheme; and (iii) investigation of the search behavior of the proposed algorithm as a method for solving regression problems.

The remainder of this work is organized as follows. Section II describes the problem followed by a theoretical background. Section III provides a detailed description of the proposed technique. Section IV presents some case studies to validate the proposed model in terms of a thorough discussion of the results and analysis of the MC-based kernel recursive least-squares algorithms. Finally, conclusions are presented in Section V.

II. KERNEL MACHINES AND NONLINEAR REGRESSION

High-dimensional feature spaces have drawn significant attention to the estimation of nonlinear functions. A direct application of this approach lies in regression, where some nonlinear mapping is followed by linear processing in a high-dimensional feature space.

A. PROBLEM DESCRIPTION

This is a problem regarding online multiple kernel-based learning that deals with nonlinear regression. By considering a wind farm, let (x_i, y_i) be an input-output stream of data pairs, where x_i is the historical data of wind power at time t, and y_i is the target power value at time $t+$horizon (Δ). The task consists of estimating a nonlinear function $f(\cdot)$ that describes the relationship between the input and output, denoted as $\hat{f}(\cdot)$. It computes the prediction of a wind farm $(\hat{y})$, or even a vector with the prediction of n wind farms $(\hat{Y})$. The unknown function $f(\cdot)$ may change over time, as opposed to standard regression settings that assume static models. The aim is to find an estimate $\hat{f}(\cdot)$ of $f(\cdot)$ in a prediction context to minimize the mean squared error (MSE) given by:

$$MSE = \frac{1}{N}\sum_{i=1}^{N} e_i^2, \tag{1}$$

where $e_i = y_i - \hat{y}_i$ and N is the number of observed data. The linear approximation of this problem is given by $\hat{f}(x_i) = Ax_i$, where $A \in \mathbb{R}^{n\times m}$ is a coefficient matrix whose entries are to be determined. Many estimation schemes based on this approximation have been studied, as reported in [41]–[43].

An approximation can be obtained in terms of the mapping $\phi(\cdot)$ to place it in a nonlinear setting simply writing $\hat{f}(x_i) = A\phi(x_i)$ with $A \in \mathbb{R}^{n\times d}$. Based on the properties of the Mercer kernel [32], it is possible to derive estimation schemes for $f(\cdot)$ in a high d-dimensional feature space without performing calculations in such space. This combines the simple implementation of linear methods with the advantageous properties of working with nonlinear mapping.

Kernel methods are based on a nonlinear transformation (kernel trick) of the input data into a high-dimensional Hilbert space ($\mathcal{H}$). Inner products can then be calculated by using a positive-definite kernel function satisfying Mercer's condition [32] to produce nonlinear versions of conventional linear learning algorithms.

The Mercer kernel is a continuous, symmetric, and positive-definite function $k(x_i, x_j) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $\mathcal{X} \in \mathbb{R}^n$ or $\mathbb{C}^n$, where $\mathcal{X}$ is a nonempty set. Mercer's theorem states that any Mercer kernel $k(\cdot, \cdot)$ can be expressed as the inner product of some fixed nonlinear function $\phi(x) : \mathcal{X} \to \mathcal{H}_1, x \in \mathcal{X}$,

$$k(x_i, x_j) = \left\langle \phi(x_i), \phi(x_j) \right\rangle_{\mathcal{H}_1}, \tag{2}$$

where $\mathcal{H}_1$ is a real- or complex-valued reproducing kernel in Hilbert space, for which $k(\cdot, \cdot)$ is a reproducing kernel and $\langle \cdot, \cdot \rangle_{\mathcal{H}_1}$ is the corresponding inner product in $\mathcal{H}_1$.

Equation (2) represents a Mercer kernel and states that if x_i and x_j are mapped onto $\mathcal{H}_1$ by $\phi(x_i)$ and $\phi(x_j)$, respectively, then the inner product of these functions can be calculated by evaluating the kernel $k(x_i, x_j)$ even if the mapping $\phi(\cdot)$ is unknown. This result is known as the kernel trick.

Many kernel functions exist, but the most common function is the Gaussian kernel. It is frequently used in real-world applications with particular success in time series prediction problems. It consists of the expansion function for an infinite-dimensional feature space given by

$$k(x_i, x_j) = exp(-||x_i - x_j||^2/2\sigma^2) \tag{3}$$

This kernel has been adopted in this study. Even though there are other possible options, the Gaussian kernel has a physical interpretation as a measure of similarity that perfectly fits this particular application. It is also worth mentioning that it has outperformed other kernel candidates, e.g., triangular and polynomial, as reported in other similar works [44], [45]. In addition, the choice or construction of kernels is very much an open problem, being the subject of ongoing research.

Three kernel recursive least-squares (KRLS) machines are used as benchmarks: the approximate linear dependency KRLS (ALD-KRLS), the sliding-window KRLS (SW-KRLS), and the KRLS tracker (KRLS-T). They were chosen because the ALD-KRLS machine was the first proposed and most popular kernel machine. Even though ALD-KRLS has no tracking system, the remaining algorithm aggregates such characteristics, albeit with different degrees of complexity. The complete derivation of such benchmark machines can be found in [44], [46], [47].

Kernel methods have proven to be successful in applications where data are entirely considered in an instance, i.e., batch applications. However, the extension of kernel methods to online settings where data arrive sequentially provides some minimized but unsolved challenges. The first is the overfitting risk when using a Hilbert space method because of the high dimension of the weight vectors. This has been handled by the use of regularization. Another problem is that the increasing complexity of the estimator representation becomes higher as the number of observations increases.

KRLS algorithms compute coefficients α_i, which consist of a minimizer used to compute the optimal weight vector, by solving a least-squares problem involving the inversion of a kernel matrix (K) whose dimension depends on the number of stored examples (M). The second challenge is that the amount of processed data M increases over time in online scenarios. Thus, practical algorithms must restrict the amount of data that will be stored. As a result, the third challenge is the training time of batch and/or the incremental update of algorithms, which typically increase linearly with the number of observations.

B. BACKGROUND REVIEW

Reviews of multiple kernel learning (MKL) algorithms are available in [48], [49]. Gonen and Alpaydın [48] concluded that overall, using multiple kernels instead of a single kernel achieves better results. MKL combines a set of kernels (basis kernels) in a linear, nonlinear or data-dependent way into a composite kernel, where the basis kernels can use different kernel functions or different values for the hyperparameters of a single kernel function [48]. Numerous studies have continuously improved the development of MKL applied in many subjects: classification of hyperspectral images [49], binary classification problems [50], air quality prediction [51], anomaly detection [52], object categorization [53], Alzheimer's disease diagnosis [54], oil painter recognition [55], multiclass classification [56], discriminating early- and late-stage cancers [57], subspace clustering [58], and many others.

In recent years, several methods combining multiple kernels have been proposed. Kannao and Guba [50] identified and modeled distinct local regions of input space, as each kernel has varying discriminative capabilities in distinct regions, naming them as 'regions of success', through a set of instance-dependent success prediction functions having high values in 'regions of success' and low ones otherwise. The use of these success prediction functions as instance-dependent weighing functions promotes locally discriminative base kernels while suppressing others. Zheng *et al.* [51] introduced the multiple kernel support vector classifier, an MKL model, which embodies the characteristics of ensemble learning, kernel learning, and representative learning. The centered alignment approach is used to obtain the weight of each kernel, and a boosting approach is used to determine the proper number of kernels. The kernels are combined by the weighted sum (conic sum restriction). The support vector

classifier is used as the base learner and optimized with a general optimizing algorithm.

An MKL approach for one-class classification was proposed by Gautam *et al.* [52]. The classifier used is Scholkopf's one-class SVM. The weight for each kernel is defined by a gating function. The weight for each kernel is assigned locally. The parameters of the gating function and one-class classifier are optimized simultaneously through a two-step optimization process. First, the optimization problem is solved to find the parameters; later, with the gating function parameters updated, the new weight is computed. Wang *et al.* [53] proposed a data-dependent MKL algorithm based on soft grouping. There are two steps in the training stage: (i) the samples are divided into groups with a soft-grouping algorithm to accommodate the correlation and diversity of the samples; (ii) an alternative optimization method is used to learn the kernel weights and support vector coefficient (classifier). The composite kernel is determined by the kernel weights of groups and the probability of this sample falling to the groups.

A novel structured sparsity, defined by $l_{1,p}$-norm ($p > 1$), regularized MKL method was designed by Peng *et al.* [54]. It represents each feature with a distinct kernel as a basis and captures featurewise importance by learning the weight for each kernel, followed by grouping the kernels according to task-specific criteria (feature modalities). Then, an optimally combined kernel presentation of multimodal features is learned in a data-driven approach. The proposed regularizer enforced on kernel weights is to sparsely (l_1) select a concise feature set within each homogeneous group and fuse the heterogeneous feature groups by taking advantage of dense norms (l_p). Liao *et al.* [55] proposed an MKL algorithm divided into three phases: first, before MKL is carried out, a prelearning process (K-medoids) is used to cluster similar candidate subkernels and select some subkernels with better classification ability in each category, which decreases the size of candidate subkernels; the second step computes the classification ability of each kernel, in each category, to select the subkernel with the best classification performance; the final phase uses the selected subkernel to carry out MKL under l_p-norm ($p > 1$) constraints.

A collaborative and geometric MKL algorithm presented by Wang *et al.* [56], directly classifies multiclass data into corresponding classes. It uses multiple empirical kernel learning to map the sample into multiple kernel spaces and then trains the softmax function in each kernel space. The softmax function can utilize the explicit features in the kernel space efficiently. To improve the collaboration between different kernel spaces, one regularization term (R_U) was designed to require the consistent outputs of samples in different kernel spaces. Moreover, to make the outputs of samples have geometric classification features, a geometric projection regularization term (R_{G_l}) was designed to reduce the within-class distance of the outputs of samples in each kernel space. The two regularization terms were introduced to improve the classification ability further. Rahimi and Gönen [57] formulated a multitask MKL method with a coclustering model on gene sets to identify biological processes and learn task-specific classification models simultaneously. Multitask learning, where different tasks are learned simultaneously, allows cohorts (i.e., tasks) with limited data to benefit from other tasks. Coclustering builds a predefined number of clusters of cohorts and pathways (i.e., tasks and kernels).

Ren *et al.* [58] proposed a novel MKL method that jointly learns an optimal affinity graph and a suitable consensus kernel for clustering purposes. The nonlinear data are mapped into a high-dimensional reproducing kernel Hilbert space where a linear pattern analysis is performed. The kernel matrix H (kernel Gram matrix) is symmetric positive semidefinite and is decomposed via an auxiliary square matrix B. This matrix is used to compose the matrix H with a sparse noise component (E) to deal with noisy data. A weighting strategy is used as the multiple kernel learning process. Note that the proposed algorithm integrates the MKL with local and global structure learning and the Hilbert space self-expressiveness property in a unified optimization problem. A denoising MKL method was presented by Zhou *et al.* [59]. It considers two kinds of noise: local noise, which appears in a small number of elements of the kernel matrix and is often induced by outliers or corrupted instances, and global noise, which appears in most of the elements of the kernel matrix and is often induced by inappropriate kernels. Noise matrices and noise tensors are introduced to capture local and global noise. The cleaned kernels are obtained by subtracting the noise from the candidate kernels. To learn the consensus kernel, the disagreement between the consensus kernel and all the cleaned kernels is minimized.

These different kernels may correspond to using different concepts of similarity or involve information coming from multiple sources, i.e., different representations or feature subsets. The reasoning is similar to combining different classifiers. Different kernels correspond to different concepts of similarity, and instead of attempting to find which kernel works best manually, a learning algorithm is responsible for selecting it, or a combination of both features can be employed. Using a specific kernel may be a source of bias, and by allowing a learner to choose among a set of kernels, an improved solution can be found. The combination function of multiple kernels and its corresponding parameters can be represented as:

$$k_\eta(x_i, x_j) = f_\eta(\{k_m(x_i^m, x_j^m)\}_{m=1}^P | \eta), \tag{4}$$

where the combination function $f_\eta : \mathbb{R}^P \to \mathbb{R}$ can be a linear or a nonlinear function. Kernel functions $\{k_m : \mathbb{R}^{D_m} \times \mathbb{R}^{D_m} \to \mathbb{R}\}_{m=1}^P$ adopt (not necessarily different) P feature representations of data instances $x_i = \{x_i^m\}_{m=1}^P$ where $x_i^m \in \mathbb{R}^{D_m}$, and D_m is the dimensionality of the corresponding feature representation. η parameterizes the combination function [48].

One of the simplest methods for determining the kernel combination function is the fixed rule. This strategy uses functions without any parameters, e.g., summation or multiplication of the kernels, and does not require any training.

Pavlidis *et al.* [60] reported that, on a gene functional classification task, training a support vector machine (SVM) [61] with an unweighted sum of heterogeneous kernels achieves better results than the combination of multiple SVMs, each trained with one kernel. The heuristic approach uses a parameterized combination function and finds the parameters of this function usually by looking at some measure separately obtained from each kernel function. These measures can be calculated from the kernel matrices or taken as the performance values of the single-kernel-based learners that are trained separately using each kernel. Moguerza *et al.* [62] and de Diego *et al.* [63] proposed a matrix functional form of combined kernels:

$$k_\eta(x_i, x_j) = \sum_{m=1}^{P} \eta_m(x_i, x_j) k_m(x_i^m, x_j^m), \tag{5}$$

where $\eta_m(\cdot, \cdot)$ assigns a weight to $k_m(\cdot, \cdot)$ according to x_i and x_j. The aforementioned works propose different heuristics to estimate the weighting function values using conditional class probabilities $Pr(y_i = y_j|x_i)$ and $Pr(y_j = y_i|x_j)$ calculated with a nearest-neighbor approach. However, each kernel function corresponds to a different neighborhood, and $\eta_m(\cdot, \cdot)$ is calculated on the neighborhood induced by $k_m(\cdot, \cdot)$.

It is also possible to use a linear combination instead of a data-dependent combination to formulate the combined kernel function as follows:

$$k_\eta(x_i, x_j) = \sum_{m=1}^{P} \eta_m k_m(x_i^m, x_j^m), \tag{6}$$

where the kernel weights are selected by looking at the performance values obtained by each kernel separately. For instance, Qiu and Lane [64] proposed two simple rules for selecting the kernel weights for regression problems:

$$\eta_m = \frac{R_m}{\sum_{h=1}^{P} R_h} \tag{7}$$

and

$$\eta_m = \frac{\sum_{h=1}^{P} M_h - M_m}{(P-1)\sum_{h=1}^{P} M_h}, \tag{8}$$

where R_m is the Pearson's correlation coefficient between the true outputs and the predicted labels generated by the regressor using the kernel matrix K_m, and M_m is the mean squared error generated by the regressor using the kernel matrix K_m. These heuristics find a convex combination of the input kernels as the combined kernel.

Pearson's correlation coefficient is a common measure of association between two continuous variables. It is defined as the ratio of the covariance of the two variables to the product of their standard deviations, commonly denoted by the Greek letter ρ:

$$\rho = \frac{Cov(X, Y)}{\sigma_X \sigma_Y} \tag{9}$$

The sample correlation coefficient, R, can be obtained by plugging the sample covariance and the sample standard deviations into the previous formula, i.e,.

$$R = \frac{\sum_{i=1}^{n}((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{10}$$

The Pearson's correlation coefficient ranges from -1 to $+1$. When $\rho > 0$, two variables tend to increase or decrease simultaneously; for $\rho < 0$, one variable tends to increase when the other decreases; finally, in $\rho = 0$ corresponds to the absence of association [65].

There are two main differences between the models described previously and the one introduced in this work. First, the base learner of the previous models majority is an SVM, while the proposed one uses KRLS algorithms, which produce much sparser solutions with higher robustness to noise. Moreover, KRLS machines are fully online algorithms designed to operate in real-time environments where data become available one sample at a time. Second, almost all of the previous models use a linear combination, which is the most popular approach with two basic categories: unweighted sum, i.e., using the sum or mean of the kernels as the combined kernel; and weighted sum. In the weighted sum case, the following combination function can be linearly parameterized:

$$k_\eta(x_i, x_j) = f_\eta(\{k_m(x_i^m, x_j^m)\}_{m=1}^{P}|\eta) = \sum_{m=1}^{P} \eta_m k_m(x_i^m, x_j^m), \tag{11}$$

where η denotes the kernel weights. Other versions of this approach differ in the constraints: the linear sum, i.e., $\eta \in \mathbb{R}^P$; the conic sum, i.e., $\eta \in \mathbb{R}_+^P$; or the convex sum, i.e., $\eta \in \mathbb{R}_+^P$ and $\sum_{m=1}^{P} \eta_m = 1$. The author in [63] applied a nonlinear combination that uses nonlinear functions of kernels, e.g., multiplication, power, and exponentiation. The introduced model uses data-dependent combination methods that assign specific kernel weights for each data instance. By doing so, it is possible to identify local distributions in the data and learn proper kernel combination rules for each region.

III. MULTIPLE CHALLENGERS KERNEL RECURSIVE LEAST-SQUARES (MC-KRLS)

The ALD-KRLS can address both difficulties presented in kernel machines, the growing kernel matrix, and the overfitting problem. By applying a sparsification procedure to the kernel matrix, it can limit the size of the dictionary and avoid overfitting, but eventually, the dictionary will reach its maximum size and will not learn from the new information received by the model.

The proposed method, named MC-KRLS, is a method for using multiple ALD-KRLS algorithms or other kernel machines by adopting the same input but with different dictionaries that are related to each other through their size. Thus, it is always possible to learn as new information is

received and to control the size of the dictionaries in the kernel machines. To reach this result, research efforts have been focused on when to create a new kernel matrix, how to compute the combined forecast, and when to delete the kernel matrix.

A. WHEN TO CREATE A NEW KERNEL MATRIX

The adopted method establishes a dependency between the kernel matrices, allowing a new matrix to be created when the previous one reaches a predetermined dictionary size as defined by the user. For instance, let us consider an MC-KRLS with three kernels (M3C-KRLS).

The first kernel is created as in the ALD-KRLS algorithm, but the second kernel is created only after the first kernel reaches 25% of its dictionary size. The third kernel is created when the second kernel reaches 15% or any other proportions set by the user.

B. HOW TO COMPUTE THE COMBINED FORECAST ($\hat{Y}$)

The first method used is the arithmetic mean of each kernel matrix output. This is considered a standard method, and no acronym is assigned to it. The second method is a weighted mean of each kernel matrix output as described in (12). The weighted version is denoted by MC-KRLSW.

$$\hat{Y} = \frac{w_{k_1}\hat{Y}_{k_1} + w_{k_2}\hat{Y}_{k_2} + \ldots + w_{k_n}\hat{Y}_{k_n}}{w_{k_1} + w_{k_2} + \ldots + w_{k_n}}, \tag{12}$$

where w_{k_n} is the weight associated with the n output kernel, which is computed as follows:

$$w_{k_i} = \frac{1}{E_i} / \sum_{j=1}^{n} \frac{1}{E_j}, \tag{13}$$

where E_i is the absolute error (AE) of the previous forecast of the kernel matrix i.

$$E_i = |Y - \hat{Y}_{k_i}|. \tag{14}$$

C. WHEN TO DELETE THE KERNEL MATRIX

An adaptive method for dealing with the kernel machines and choosing the 'best ones' is described in this section. The first approach assumes that when all kernel matrices reach their maximum size, a counter will start, and when the counter stops, the MSE is computed. The kernel matrix with the worst MSE is then deleted. This is the standard method, and no acronym is given to it.

The second method is similar to the first method, but instead of computing the MSE when the counter ends, AE is computed for every iteration. Then, the best kernel matrix, i.e., with the kernel with the smallest AE, will not suffer any changes, whereas the other matrices will have their 'ages' increased. The matrix that reaches its predefined maximum age first is deleted.

Life expectancy (LE) is a statistical measure of how long an organism may live, and at a given age, life is expected to cease. LE is based on many features, such as the year of birth, current age, and other demographic factors. A simplified version of this concept is used here based on two constants called the aging factor (AF) and weakening factor (WF) [40].

In nature, colony leaders are constantly challenged by new individuals. Aging facilitates a leader to be replaced by a younger individual, whereby it is likely to create more opportunities for diversity and improvements. Inspired by this phenomenon, this work adapts the aforementioned idea from nature to the kernel machines and proposes the MC-KRLS aggregating both AF and WF, resulting in the MC-KRLSA algorithm.

Aging is not the only feature that composes the LE of an individual. In this study, WF represents the other features associated with LE. WF plays a role whenever the kernel (K_i) cannot find a new result ($\hat{y}_{k_i}(t)$) better than the previous result ($\hat{y}_{k_i}(t-1)$). Thus, even kernels of the same age will have different LEs and will cease to exist at different moments. First, the kernel age KA_i of K_i is set to zero, and then the kernel machines follow the rules defined as:

$$\text{if } \hat{y}_{k_i}(t) \text{ is worse than } \hat{y}_{k_i}(t-1) \text{ then} \tag{15}$$
$$KA_i(t) = KA_i(t-1) + AF + WF \text{ else} \tag{16}$$
$$KA_i(t) = KA_i(t-1) + AF \tag{17}$$
$$\text{end if} \tag{18}$$

The proposed algorithm aims to overcome the main limitations of similar approaches in tracking changes in the underlying stochastic process, as discussed in Section II. In addition to the number of used kernels (η_K), its remaining parameters are as follows: the maximum value reached by the counter ($counter.max$), which in this work is set to the same value as the maximum dictionary size; and when to create a new matrix, which is investigated for 25%, 50%, 75%, and 100% of the maximum dictionary size (M). For the sake of reproducibility of this paper the source codes are available for download at [66].

Table 1 shows the used acronyms, which comprise a composition of MC-KRLS, the number of used kernels, how the forecast is combined, and the delete method.

TABLE 1. Multiple kernel learning methods used in the study.

Kernels	Combined Forecast	Delete Method	Acronym
2	Arithmetic Mean	Counter	M2C-KRLS
3	Arithmetic Mean	Counter	M3C-KRLS
2	Weighted Mean (W)	Counter	M2C-KRLSW
3	Weighted Mean (W)	Counter	M3C-KRLSW
2	Arithmetic Mean	Aging (A)	M2C-KRLSA
3	Arithmetic Mean	Aging (A)	M3C-KRLSA
2	Weighted Mean (W)	Aging (A)	M2C-KRLSAW
3	Weighted Mean (W)	Aging (A)	M3C-KRLSAW

IV. CASE STUDY AND APPLICATIONS RESULTS

In this section, the search behavior of MC-KRLS machines in solving regression problems is evaluated. The case study is applied to a wind power database and used to predict $(t+5)$ minutes ahead. In particular, answers to the following

two questions are sought: (i) how do multiple challenger kernel machines work on regression problems, and (ii) how do the parameters of the multiple challenger model lead to the best set? All of the algorithms were developed in MATLAB and executed at the Danmarks Tekniske Universitet (DTU) High-Performance Computing (HPC) clusters: Central and Compute.

A. THE DATASET

The dataset is composed of measurements of wind power generation for every 5 minutes, from 01/01/2011 04:05 to 01/01/2013 04:00, of 23 onshore wind farms [67], resulting in 210,518 measurements for each farm as provided by the Australian Energy Market Operator (AEMO). In this particular study, eight wind farms are chosen. All negative values are set to zero, and data are normalized within the range]0,1[. The complete dataset used in this work is available for download at [68].

B. BENCHMARKS

To perform a comparative study and check the performance of the proposed method, five models are used as benchmarks. In addition to the three kernel machines mentioned in Section II-A, the persistence and the online sequential extreme learning machine (OS-ELM) are briefly described.

The persistence forecast assumes that the future wind speed is the same as the most recent measurement. The persistence forecast for a time interval Δ ahead is given by $\hat{Y}_{i+\Delta} = Y_i$. Sequential implementation of the least-squares solution of the output weight vector results in the OS-ELM [69], which uses the recursive least-squares algorithm [70].

C. DEFINITION OF PARAMETERS USED IN THE WIND POWER FORECAST

The best parameter values are found by using a k-fold cross-validation procedure with five blocks of 8,421 points, where four (33,684 points) are used as training sets and one of them (8,421 points) is always left out to serve as the validation set, while the remaining data (168,413 points) are used as a test set to obtain the final evaluation [71]. To make it fair, the same partitions of the training, validation and testing data are used when different comparison algorithms are trained on the dataset. It is observed that there is no rule on data splitting, just common practices. The minimum percent average root-mean-square error (RMSE %) is used as the performance index.

The parameter that controls the dictionary size for each algorithm determines the computational and memory complexities. For each experiment, this parameter is varied over a wide range so that the RMSE can be measured. Performance curves are used for comparison purposes. Section IV-D2 presents the curves of the benchmark provided by ALD-KRLS and their respective MC-KRLS as part of the analysis of parameters ALD threshold (ν) and the optimal LAG length $X = \{X_1, X_2 \ldots, X_k\}$.

It is observed that the optimal selection of parameters is an important open problem in the kernel machine literature, but it falls outside the scope of this work. The remaining parameters are chosen by an exhaustive search to optimize the position of the performance curve, while the results are listed in Table 2.

TABLE 2. Parameters used in the wind power forecast.

Method	Parameters
ALD-KRLS SW-KRLS KRLS-T	M = 20, 30, 40, 50, 60, 70 LAG = 1, 2, 3, 4, 5, 6, 7, 8 Gaussian Kernel Kernel parameter (σ) = 32
ALD-KRLS	ν = 2E-5, 1E-5, 2E-4, 1E-4, 1E-3, 0.05, 0.01, 0.1
SW-KRLS	c = 2E-5, 1E-5, 2E-4, 1E-4, 1E-3, 0.05, 0.01, 0.1
KRLS-T	λ = 0.9, 0.99, 0.999 *jitter* = 5E-6, 1E-6, 1E-5 $sn2$ = 5E-3, 5E-2, 1E-2
OS-ELM	$NOHN$ = 3, 5, 10, 20, 30, 50, 75, 100, 150, 200, 250 LAG = 1, 2, 3, 4, 5, 6, 7, 8 Radial Basis Function (RBF) Number of training data = 33,684

According to Table 2, M is the dictionary size, σ is the kernel width, c is a regularization parameter, λ is the forgetting factor, *jitter* is noise used to avoid round-off error, $sn2$ is a noise-to-signal ratio, and $NOHN$ is the number of hidden nodes assigned.

D. TEST RESULTS

Fig. 1 represents the two experiments in terms of time series, namely, test case 1 (TC1), which is composed of single-input and single-output (SISO) and multiple-input and multiple-output (MIMO) techniques, and test case 2 (TC2), represented by the MC-KRLS, where the proposed algorithm with n MIMO kernel machines tracks the results to provide forecasts for w wind power plants, while an 'operator' (OP) is responsible for combining all of them.

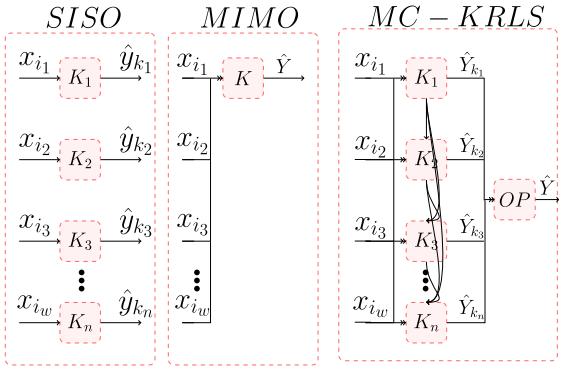

FIGURE 1. Block diagram of all the arrangements to which KRLS machines are applied during the experiment.

1) TEST CASE 1

Table 3 presents the RMSE (%) of each wind farm. The last row contains their respective averages. The differences

between SISO kernel machines are small. However, the kernel machine that uses the SISO ALD threshold as a sparsity tool is the best kernel machine.

The use of MIMO layouts improves the RMSE (%) when compared with their SISO counterparts, except for the SW-KRLS. The MIMO ALD-KRLS proves to be the best kernel machine again. The use of MIMO for the SW-KRLS presents slightly higher RMSE(%) when compared with the respective SISO versions. The KRLS-T shows equal results due to the round-off error, but the MIMO version presents better performances in seven of the eight sites. The exception is wind farm LKBONNY3, which causes the average to be slightly higher than that of the SISO version.

The Pearson correlation coefficient (R) is used to show why some MIMO models achieve better results than those obtained with SISO models. If R is greater than 0.8, then the system is described as strongly correlated, whereas it is described as weakly correlated when R is less than 0.5. Table 4 presents the results for each coefficient. The last three rows present a summary of the parameters classified as strongly, correlated, and weakly correlated. Five of the eight farms are strongly correlated or correlated with four or more wind farms.

For all cases in which ν and $c > 0.001$, there is a significant increase in the RMSE (%). Furthermore, in every case, the smaller the dictionary size and the greater the LAG, the higher the RMSE (%). Finally, when using the ALD-KRLS, two time series are strongly affected by the dictionary size, i.e., HALLWF1 and NBHWF1. The authors of KRLS-T state that λ is usually sensitive within the range [0.95, 1] [47]. The best RMSE (%) is often obtained for $\lambda = 0.9$, as the error increases with λ.

2) TEST CASE 2

The differences between the ALD-KRLS benchmarks (SISO and MIMO) and the proposed MC-KRLS models are small. However, the results achieved with the introduced models are, in general, better than those obtained with the MIMO ALD-KRLS, as shown in Table 5. The SW-KRLS and KRLS-T models now present the same previous behavior observed in the SISO and MIMO models.

Analogous to what is observed with the ALD-KRLS SISO and MIMO machines, there is a significant increase in the RMSE (%) in all cases in which $\nu > 0.001$, but different behaviors are verified. First, the dictionary size or the number of LAGs used does not significantly influence the RMSE (%), thus allowing the use of the smallest dictionary size and the shorter LAG length. Second, when using the ALD-KRLS, two time series are significantly affected by the dictionary size, i.e., HALLWF1 and NBHWF. However, this is not observed in the case of multiple KRLS. Third, the use of any of the multiple challenger models shows more stable results compared with the ALD-KRLS.

Fig. 2 shows the results of MIMO and M3C-KRLSW ALD-KRLS, which are the best results presented in Tables 3 and 5. It is observed that the use of multiple kernels

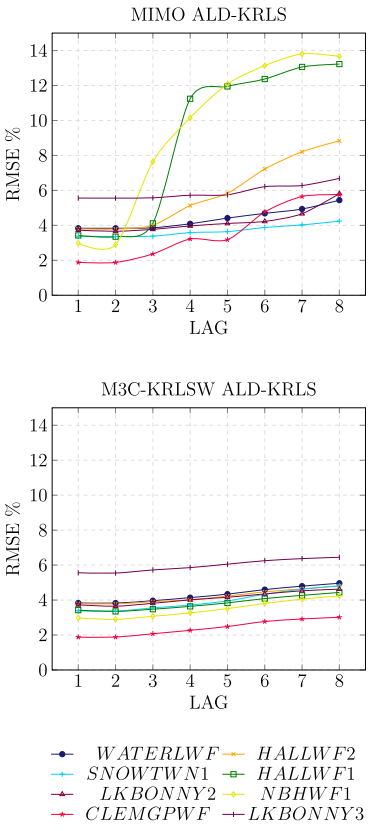

FIGURE 2. Comparison between the MIMO and M3C-KRLSW ALD-KRLS forecasts for ν = 1.00E-05 and varying the number of LAGs.

minimizes the importance of the number of LAGs for any ν value.

The SW-KRLS MC-KRLS presents a significant increase in the RMSE (%) in all cases in which $c > 0.001$. For all conditions, the smaller the dictionary size and the greater the LAG, the higher the RMSE (%). For the KRLS-T, the best RMSE (%) is usually reached using $\lambda = 0.9$, while the error increases with λ.

The first question raised in Section III can be answered with the help of Fig. 3, which is used to analyze the behavior of the RMSE(%) and the need to eventually start a new kernel machine. The best results of the multiple kernel machines using different dictionary sizes and triggers to start a new kernel machine are presented in terms of distinct color scales for each graph. It is a matrix plot that produces a filled net of shaded rectangles, where each matrix position corresponds to one rectangle. The 'Dictionary size' axis represents the maximum number of examples (20, 30, 40, 50, 60, 70) saved in each kernel dictionary. The 'Start new kernel' axis shows the percentage of the maximum number of examples reached by a kernel dictionary (25%, 50%, 75%, 100%) before a new dictionary is created. Finally, the (RMSE(%)) column relates the color scale with the error value.

The best results for the ALD-KRLS M2C-KRLS are found to be 75% for dictionary sizes of 20, 30, and 40 as a trigger to start the new kernel machine and 25% for 50, 60, and 70.

TABLE 3. Forecast of the benchmark models - RMSE (%).

Wind Farm	Persistence	OS-ELM		ALD-KRLS		SW-KRLS		KRLS-T	
		SISO	MIMO	SISO	MIMO	SISO	MIMO	SISO	MIMO
WATERLWF	3.863	3.854	3.879	3.853	3.833	3.959	4.223	3.860	3.859
HALLWF2	3.846	3.837	3.821	3.836	3.767	3.999	4.171	3.852	3.851
SNOWTWN1	3.393	3.352	3.411	3.357	3.341	3.566	3.804	3.409	3.408
HALLWF1	3.450	3.431	3.388	3.433	3.342	3.577	3.727	3.461	3.461
LKBONNY2	3.760	3.678	3.712	3.679	3.614	3.817	4.077	3.777	3.776
NBHWF1	3.000	2.916	2.937	2.926	2.877	3.117	3.273	3.029	3.028
CLEMGPWF	1.893	1.884	1.915	1.888	1.876	2.064	2.220	1.895	1.894
LKBONNY3	5.639	5.552	5.573	5.545	5.514	5.745	6.039	5.618	5.622
AVERAGE	3.606	3.563	3.580	3.565	3.520	3.731	3.942	3.613	3.613

TABLE 4. Pearson product-moment correlation coefficients.

Wind Farm	WATERLWF	HALLWF2	SNOWTWN1	HALLWF1	LKBONNY2	NBHWF1	CLEMGPWF	LKBONNY3
WATERLWF	1							
HALLWF2	0.82	1						
SNOWTWN1	0.41	0.53	1					
HALLWF1	0.79	0.91	0.49	1				
LKBONNY2	0.29	0.29	0.10	0.33	1			
NBHWF1	0.79	0.87	0.54	0.90	0.26	1		
CLEMGPWF	0.50	0.54	0.77	0.50	0.05	0.63	1	
LKBONNY3	0.36	0.34	0.14	0.37	0.90	0.29	0.08	1
STRONGLY C.	1	3	0	2	1	2	0	1
CORRELATED	3	2	3	2	0	3	5	0
WEAKLY C.	3	2	4	3	6	2	2	6

TABLE 5. Forecast of the models - RMSE (%).

	SISO	MIMO							
	KRLS	KRLS	M2C KRLS	M2C KRLSW	M2C KRLSA	M2C KRLSAW	M3C KRLS	M3C KRLSW	M3C KRLSA	M3C KRLSAW
ALD-KRLS	3.565	3.520	3.522	3.520	3.522	3.520	3.519	3.517	3.520	3.519
SW-KRLS	3.731	3.942	3.953	3.920	3.948	3.928	3.921	3.925	3.909	3.914
KRLS-T	3.613	3.613	3.678	3.652	3.661	3.641	3.649	3.636	3.645	3.636

The same behavior is observed in all the other models presented in this work, as denoted in Table 1. The multiple kernel machines that use the SW-KRLS and KRLS-T always create a new kernel machine when the previous dictionary reaches 25% of its size, independent of the dictionary size.

To answer the second question raised in Section III, two solutions are presented: the arithmetic and weighted mean. The weighted version is the best one in 10 of the 12 proposed models, as observed in Table 5. The only exceptions are SW-KRLS M3C-KRLS compared with M3C-KRLSW and SW-KRLS M3C-KRLSAW compared with M3C-KRLSA.

For the third question in Section III, two options are also presented: one using the MSE and the other the AE. Table 5 shows that there is no difference between the results for the ALD-KRLS using two kernel machines. When three kernel machines are adopted, the use of MSE provides better results. For the SW-KRLS using the arithmetic mean, the AE also presents improved performance. Finally, the KRLS-T presents better results using the MSE in all proposed models.

Fig. 4 shows the behavior of the RMSE(%) versus the dictionary size used by the kernel machines. It shows the best result of the multiple kernel machines using different dictionary sizes and triggers to start a new kernel machine. The color scales are different for each graph. Fig. 4 has the same description as Fig.3. The best results for the ALD-KRLS M2C-KRLS are found with smaller dictionary sizes. According to values in the third column, the multiple kernel machines that use the KRLS-T present similar behavior to those obtained with ALD-KRLS. However, the best results are always obtained with a larger dictionary size when using SW-KRLS.

In Fig. 3 and 4, it can be noted that the ALD-KRLS M2C-KRLS obtains the lowest RMSE (%), mainly when 75% of the dictionary size is set to start a new kernel and it uses dictionaries with 20 examples, the smallest one in the range analyzed by this study (20, 30, 40, 50, 60, 70). For SW-KRLS M2C-KRLS machines, the best results are usually 25% of the dictionary size with the largest dictionary size (70). For the KRLS-T M2C-KRLS, the best results are usually 25% of the dictionary size with the smallest dictionary size (20). Similar behavior is found when using any machine listed in Table 1.

To compare computational time was required to run all the methods in a single machine. The results presented before

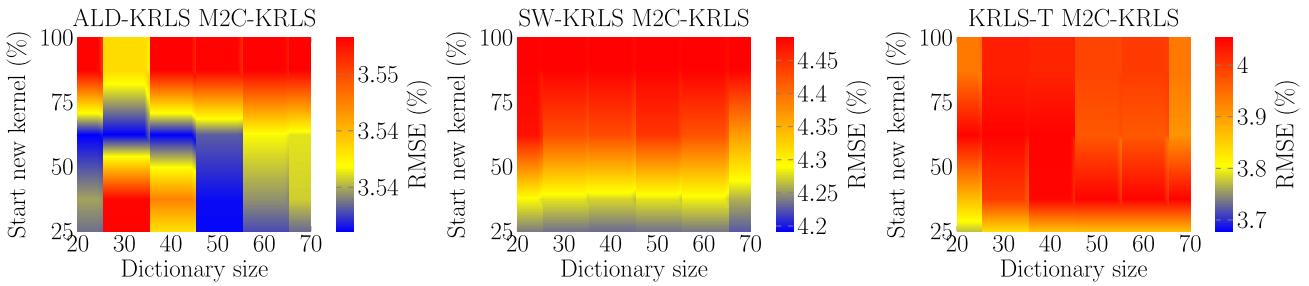

FIGURE 3. Behavior of the RMSE(%) versus when to start a new kernel machine.

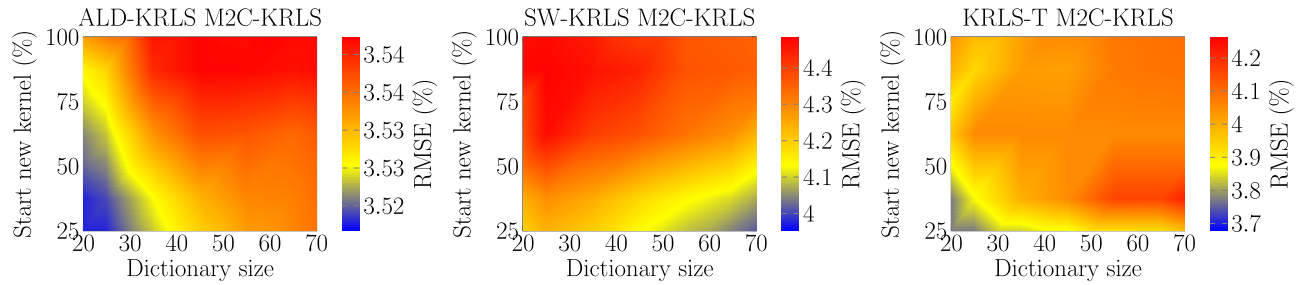

FIGURE 4. Behavior of the RMSE(%) versus dictionary size.

TABLE 6. Computation time (s).

	M2C KRLS	M2C KRLSW	M2C KRLSA	M2C KRLSAW	M3C KRLS	M3C KRLSW	M3C KRLSA	M3C KRLSAW
ALD-KRLS	205.1111	209.3029	206.0297	213.9421	301.7949	309.9622	307.4587	310.7236
SW-KRLS	139.3686	142.5556	139.9083	143.0825	245.3131	246.4561	242.0534	247.2417
KRLS-T	170.1855	172.3439	173.2392	177.2522	289.8388	294.6144	292.6144	297.4468
ALD-KRLS	100%	102.04%	100.45%	104.31%	147.14%	102.71%	101.88%	102.96%
SW-KRLS	100%	102.29%	100.39%	102.66%	176.02%	100.47%	98.67%	100.79%
KRLS-T	100%	101.79%	101.79%	104.15%	170.31%	101.65%	100.96%	102.62%

were from a HPC consisting of different hardware. Then, it was defined: (i) The goal of tracking and analyzing the software metric is to determine the computational time of each method; (ii) Each benchmark was from the same computer, and how long it took (in seconds) across 10 different runs; (iii) Table 6 shows the worst times found for each method using different kernel machines ($M = 70$, $LAG = 8$) for one entire process (33,684 training, 8,421 validation, and 168,413 test points).

The increase of the number of kernel machines directly affects the time presented between the M2C and M3C groups. Independent of the method used, the kind of kernel machine (ALD-KRLS, SW-KRLS, KRLS-T) has a great impact on the running time. Using the arithmetic means (MC-KRLS) as reference in all cases the use of the weighted mean (MC-KRLSW) brought a time increase, the use of the age factor (MC-KRLSA) increased the execution time but not as much as the weighted version, actually M3C-KRLSA (SW-KRLS) had an reduce of 1.33% at execution time. The combination of both (MC-KRLSAW) shows the highest execution time as expected, since the increase of lines of codes and computations.

V. CONCLUSION

A novel and competitive adaptive method that can be used in any kernel machine for short-term forecasting of wind power is introduced in this work. This method uses multiple kernel machines related to each other through the sizes of the respective dictionaries. A competitive adaptive factor is aggregated to the problem when all machines reach the maximum dictionary size and then the worst kernel is deleted.

Multiple kernel machines have been applied to a dataset of wind power plants in Australia over a period of two years.

The proposed algorithm creates new kernel matrices as long as the process continues running, thus identifying changes in the time series, avoiding the abrupt loss of information that typically occurs in tracking methods, and maintaining a controlled number of examples since there is an adaptive selection of active kernels. The kernel size is then fixed rather than limited. The use of the MC-KRLS machine makes the use of smaller dictionary sizes possible, resulting in better results and reducing the probability of overfitting. The susceptibility of the model to the number of LAGs used is also reduced. The kernel machine chosen impacts greatly in the computation time. The computation time could be a

matter during training, validation and test period, however during regular operation, in the worst result to predictor $t + 5$ minutes was computed in 1, 47 milliseconds.

As expected, there is not a single best performing algorithm for all scenarios. The optimal choice of an algorithm depends on the range for the target RMSE (%), the available computational resources, and the particular dataset.

This work was motivated by the need to produce accurate very-short-term forecasts for one or multiple wind farms. Future work will focus on extending this approach to other variables, e.g., temperature, wind speed, wind direction, among others; additional forecast horizons; investigation of other kernel machines; and consideration of the development of other adaptive models, possibly taking into account the similarity results in addition to the AE or MSE.

REFERENCES

[1] G. W. E. C. GWEC, *Global Wind Report 2019*. Brussels, Belgium: Global Wind Energy Council, 2020.

[2] A. R. Araghi, G. H. Riahy, O. Carlson, and S. Gros, "Enhancing the net energy of wind turbine using wind prediction and economic NMPC with high-accuracy nonlinear WT models," *Renew. Energy*, vol. 151, pp. 750–763, May 2020.

[3] E. B. Ssekulima, M. B. Anwar, A. A. Hinai, and M. S. El Moursi, "Wind speed and solar irradiance forecasting techniques for enhanced renewable energy integration with the grid: A review," *IET Renew. Power Gener.*, vol. 10, no. 7, pp. 885–989, Jul. 2016.

[4] S. A. Vargas, G. R. T. Esteves, P. M. Maçaira, B. Q. Bastos, F. L. C. Oliveira, and R. C. Souza, "Wind power generation: A review and a research agenda," *J. Cleaner Prod.*, vol. 218, pp. 850–870, May 2019.

[5] J. S. Roungkvist and P. Enevoldsen, "Timescale classification in wind forecasting: A review of the state-of-the-art," *J. Forecasting*, vol. 39, no. 5, pp. 757–768, Aug. 2020.

[6] S. S. Soman, H. Zareipour, O. Malik, and P. Mandal, "A review of wind power and wind speed forecasting methods with different time horizons," in *Proc. North Amer. Power Symp.*, Sep. 2010, pp. 1–8.

[7] A. Sfetsos, "A comprehensive overview of short term wind forecasting models based on time series analysis," in *Soft Computing in Green and Renewable Energy Systems*. Berlin, Germany: Springer, 2011, pp. 97–116.

[8] X. Zhao, S. Wang, and T. Li, "Review of evaluation criteria and main methods of wind power forecasting," *Energy Procedia*, vol. 12, pp. 761–769, Jan. 2011.

[9] J. Zhang, M. Cui, B.-M. Hodge, A. Florita, and J. Freedman, "Ramp forecasting performance from improved short-term wind power forecasting over multiple spatial and temporal scales," *Energy*, vol. 122, pp. 528–541, Mar. 2017.

[10] J. R. Andrade and R. J. Bessa, "Improving renewable energy forecasting with a grid of numerical weather predictions," *IEEE Trans. Sustain. Energy*, vol. 8, no. 4, pp. 1571–1580, Oct. 2017.

[11] A. M. Foley, P. G. Leahy, A. Marvuglia, and E. J. McKeogh, "Current methods and advances in forecasting of wind power generation," *Renew. Energy*, vol. 37, no. 1, pp. 1–8, 2012.

[12] H. Liu and C. Chen, "Data processing strategies in wind energy forecasting models and applications: A comprehensive review," *Appl. Energy*, vol. 249, pp. 392–408, Sep. 2019.

[13] M. Lydia, S. S. Kumar, A. I. Selvakumar, and G. E. P. Kumar, "Linear and non-linear autoregressive models for short-term wind speed forecasting," *Energy Convers. Manage.*, vol. 112, pp. 115–124, Mar. 2016.

[14] S.-K. Sim, P. Maass, and P. G. Lind, "Wind speed modeling by nested ARIMA processes," *Energies*, vol. 12, no. 1, p. 69, 2019.

[15] S.-X. Wang, M. Li, L. Zhao, and C. Jin, "Short-term wind power prediction based on improved small-world neural network," *Neural Comput. Appl.*, vol. 31, no. 7, pp. 3173–3185, Jul. 2019.

[16] A. Pawar, V. Jape, and S. Mathew, "Wind power forecasting using support vector machine model in rstudio," in *Cognitive Informatics and Soft Computing*. Singapore: Springer, 2019, pp. 289–298.

[17] B. Khorramdel, C. Y. Chung, N. Safari, and G. C. D. Price, "A fuzzy adaptive probabilistic wind power prediction framework using diffusion kernel density estimators," *IEEE Trans. Power Syst.*, vol. 33, no. 6, pp. 7109–7121, Nov. 2018.

[18] F. Liu, R. Li, and A. Dreglea, "Wind speed and power ultra short-term robust forecasting based on Takagi–Sugeno fuzzy model," *Energies*, vol. 12, no. 18, p. 3551, Sep. 2019.

[19] N. Li, F. X. He, and W. T. Ma, "Wind power prediction based on extreme learning machine with kernel mean *p*-power error loss," *Energies*, vol. 12, no. 4, p. 673, 2019.

[20] H. Acikgoz, C. Yildiz, and M. Sekkeli, "An extreme learning machine based very short-term wind power forecasting method for complex terrain," *Energy Sources A, Recovery, Utilization, Environ. Effects*, vol. 42, no. 22, pp. 1–16, 2020.

[21] Z. Qian, Y. Pei, H. Zareipour, and N. Chen, "A review and discussion of decomposition-based hybrid models for wind energy forecasting applications," *Appl. Energy*, vol. 235, pp. 939–953, Feb. 2019.

[22] Z. Tian, S. Li, Y. Wang, and X. Wang, "Wind power prediction method based on hybrid kernel function support vector machine," *Wind Eng.*, vol. 42, no. 3, pp. 252–264, Jun. 2018.

[23] Aasim, S. N. Singh, and A. Mohapatra, "Repeated wavelet transform based ARIMA model for very short-term wind speed forecasting," *Renew. Energy*, vol. 136, pp. 758–768, Jun. 2019.

[24] S. P. Mishra and P. K. Dash, "Short-term prediction of wind power using a hybrid pseudo-inverse Legendre neural network and adaptive firefly algorithm," *Neural Comput. Appl.*, vol. 31, no. 7, pp. 2243–2268, Jul. 2019.

[25] Z. Qu, W. Mao, K. Zhang, W. Zhang, and Z. Li, "Multi-step wind speed forecasting based on a hybrid decomposition technique and an improved back-propagation neural network," *Renew. Energy*, vol. 133, pp. 919–929, Apr. 2019.

[26] Z. S. Tian Li and Y. Wang, "A prediction approach using ensemble empirical mode decomposition-permutation entropy and regularized extreme learning machine for short-term wind speed," *Wind Energy*, vol. 23, no. 7, pp. 177–206, 2020.

[27] Z. Liu, P. Jiang, L. Zhang, and X. Niu, "A combined forecasting model for time series: Application to short-term wind speed forecasting," *Appl. Energy*, vol. 259, Feb. 2020, Art. no. 114137.

[28] G. Giebel, R. Brownsword, G. Kariniotakis, M. Denhard, and C. Draxl, "The state-of-the-art in short-term prediction of wind power. A literature overview," ANEMOS.plus, Porto, Portugal, Tech. Rep. Deliverable D-1.2, 2011.

[29] I. Okumus and A. Dinler, "Current status of wind energy forecasting and a hybrid method for hourly predictions," *Energy Convers. Manage.*, vol. 123, pp. 362–371, Sep. 2016.

[30] J. W. Messner, P. Pinson, J. Browell, M. B. Bjerregård, and I. Schicker, "Evaluation of wind power forecasts—An up-to-date view," *Wind Energy*, vol. 23, no. 6, pp. 1461–1481, 2020.

[31] V. Vapnik, *Estimation of Dependences Based on Empirical Data*. Berlin, Germany: Springer, 2006.

[32] V. Vapnik, *The Nature of Statistical Learning Theory*. Berlin, Germany: Springer, 2013.

[33] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 9, 1997, pp. 155–161.

[34] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

[35] A. Kuh and D. Mandic, "Applications of complex augmented kernels to wind profile prediction," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2009, pp. 3581–3584.

[36] J. Dowell, S. Weiss, and D. Infield, "Kernel methods for short-term spatio-temporal wind prediction," in *Proc. IEEE Power Energy Soc. Gen. Meeting*, Jul. 2015, pp. 1–5.

[37] P. Kumari and R. Wadhvani, "Wind power prediction using KLMS algorithm," in *Proc. Int. Conf. Inventive Res. Comput. Appl. (ICIRCA)*, Jul. 2018, pp. 154–161.

[38] M. Espinoza, J. A. Suykens, R. Belmans, and B. De Moor, "Electric load forecasting: Using kernel-based modeling for nonlinear system identification," *IEEE Control Syst. Mag.*, vol. 27, no. 5, pp. 43–57, Oct. 2007.

[39] J. Che and J. Wang, "Short-term load forecasting using a kernel-based support vector regression combination model," *Appl. Energy*, vol. 132, pp. 602–609, Nov. 2014.

[40] E. C. Bezerra, R. P. S. Leão, and A. P. D. S. Braga, "A self-adaptive approach for particle swarm optimization applied to wind speed forecasting," *J. Control, Autom. Electr. Syst.*, vol. 28, no. 6, pp. 785–795, Dec. 2017.

[41] C. Wei, J. Chen, Z. Song, and C.-I. Chen, "Development of self-learning kernel regression models for virtual sensors on nonlinear processes," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 1, pp. 286–297, Jan. 2019.
[42] R. Boloix-Tortosa, J. J. Murillo-Fuentes, I. Santos, and F. Pérez-Cruz, "Widely linear complex-valued kernel methods for regression," *IEEE Trans. Signal Process.*, vol. 65, no. 19, pp. 5240–5248, Oct. 2017.
[43] Y. Jia, S. Kwong, W. Wu, R. Wang, and W. Gao, "Sparse Bayesian learning-based kernel Poisson regression," *IEEE Trans. Cybern.*, vol. 49, no. 1, pp. 56–68, Jan. 2019.
[44] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.
[45] F. A. Tobar, S.-Y. Kung, and D. P. Mandic, "Multikernel least mean square algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 265–277, Feb. 2014.
[46] S. Van Vaerenbergh, J. Via, and I. Santamaría, "A sliding-window kernel RLS algorithm and its application to nonlinear channel identification," in *Proc. IEEE Int. Conf. Acoust. Speed Signal Process.*, May 2006, p. 5.
[47] V. Van Vaerenberg, M. Lázaro-Gredilla, and I. Santamaria, "Kernel recursive least-squares tracker for time-varying regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1313–1326, Aug. 2012.
[48] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *J. Mach. Learn. Res.*, vol. 12, pp. 2211–2268, Jul. 2011.
[49] T. Liu and Y. Gu, "Multiple kernel learning for hyperspectral image classification," in *Hyperspectral Image Analysis*. Cham, Switzerland: Springer, 2020, pp. 259–293.
[50] R. Kannao and P. Guha, "Success based locally weighted multiple kernel combination," *Pattern Recognit.*, vol. 68, pp. 38–51, Aug. 2017.
[51] H. Zheng, H. Li, X. Lu, and T. Ruan, "A multiple kernel learning approach for air quality prediction," *Adv. Meteorol.*, vol. 2018, pp. 1–15, Jun. 2018.
[52] C. Gautam, R. Balaji, K. Sudharsan, A. Tiwari, and K. Ahuja, "Localized multiple kernel learning for anomaly detection: One-class classification," *Knowl.-Based Syst.*, vol. 165, pp. 241–252, Feb. 2019.
[53] Q. Wang, G. Fu, L. Li, H. Wang, and Y. Li, "Data-dependent multiple kernel learning algorithm based on soft-grouping," *Pattern Recognit. Lett.*, vol. 112, pp. 111–117, Sep. 2018.
[54] J. Peng, X. Zhu, Y. Wang, L. An, and D. Shen, "Structured sparsity regularized multiple kernel learning for Alzheimer's disease diagnosis," *Pattern Recognit.*, vol. 88, pp. 370–382, Apr. 2018.
[55] Z. Liao, L. Gao, T. Zhou, X. Fan, Y. Zhang, and J. Wu, "An oil painters recognition method based on cluster multiple kernel learning algorithm," *IEEE Access*, vol. 7, pp. 26842–26854, 2019.
[56] Z. Wang, Z. Zhu, and D. Li, "Collaborative and geometric multi-kernel learning for multi-class classification," *Pattern Recognit.*, vol. 99, Mar. 2020, Art. no. 107050.
[57] A. Rahimi and M. Gönen, "A multitask multiple kernel learning formulation for discriminating early- and late-stage cancers," *Bioinformatics*, vol. 36, no. 12, pp. 3766–3772, Jun. 2020.
[58] Z. Ren, H. Li, C. Yang, and Q. Sun, "Multiple kernel subspace clustering with local structural graph and low-rank consensus kernel learning," *Knowl.-Based Syst.*, vol. 188, Jan. 2020, Art. no. 105040.
[59] P. Zhou, F. Ye, and L. Du, "Unsupervised robust multiple kernel learning via extracting local and global noises," *IEEE Access*, vol. 7, pp. 34451–34461, 2019.
[60] P. Pavlidis, J. Weston, J. Cai, and W. N. Grundy, "Gene functional classification from heterogeneous data," in *Proc. 5th Annu. Int. Conf. Comput. Biol.*, 2001, pp. 249–255.
[61] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 988–999, Sep. 1999.
[62] J. M. Moguerza, A. Muñoz, and I. M. Diego, "Improving support vector classification via the combination of multiple sources of information," in *Structural, Syntactic, and Statistical Pattern Recognition*. Berlin, Germany: Springer, 2004, pp. 592–600.
[63] I. M. de Diego, A. Muñoz, and J. M. Moguerza, "Methods for the combination of kernel matrices within a support vector framework," *Mach. Learn.*, vol. 78, nos. 1–2, pp. 137–174, Jan. 2010.
[64] S. Qiu and T. Lane, "A framework for multiple kernel support vector regression and its applications to siRNA efficacy prediction," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 6, no. 2, pp. 190–199, Jun. 2009.
[65] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*. Boca Raton, FL, USA: CRC Press, 2000.
[66] E. Bezerra. (2020). *MC-KRLS Collection of Codes*. [Online]. Available: https://github.com/erickbezerra/MC-KRLS
[67] J. Dowell and P. Pinson, "Very-short-term probabilistic wind power forecasts by sparse vector autoregression," *IEEE Trans. Smart Grid*, vol. 7, no. 2, pp. 763–770, May 2016.
[68] J. Dowell. (2015). *AEMO 5 Minute Wind Power Data*. [Online]. Available: http://www.jethrobrowell.com/data-and-code.html
[69] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.
[70] E. K. Chong and S. H. Zak, *An Introduction to Optimization*, vol. 76. Hoboken, NJ, USA: Wiley, 2013.
[71] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, vol. 112. New York, NY, USA: Springer, 2013.

ERICK C. BEZERRA received the B.Sc. degree in electrical engineering from the University of Fortaleza (UNIFOR), Fortaleza, Brazil, in 2002, and the M.Sc. degree in electrical engineering from the Federal University of Ceará, Fortaleza, in 2012, where he is currently pursuing the Ph.D. degree in electrical engineering.

He is currently a Lecturer with the Department of Electrical Engineering, UniFANOR Wyden, Fortaleza. His research is focused on wind power forecasting and the development of computational tools. His research interests include forecasting, computational intelligence, and data analysis.

PIERRE PINSON (Fellow, IEEE) received the M.Sc. degree in applied mathematics from the National Institute for Applied Sciences (INSA), Toulouse, France, and the Ph.D. degree in energetics from École des Mines de Paris, Paris, France.

He is currently a Professor with the Department of Electrical Engineering, Center for Electric Power and Energy, Technical University of Denmark (DTU), Kongens Lyngby, Denmark, and the Head of the Research Group focused on energy analytics and markets. His research interests include forecasting, uncertainty estimation, optimization under uncertainty, decision sciences, and renewable energies. He serves as an Editor for the *International Journal of Forecasting* and *Wind Energy*.

RUTH P. S. LEÃO was born in Fortaleza, Brazil. She received the B.Sc. degree in electrical engineering from the Federal University of Ceará, Brazil, in 1978, and the Ph.D. degree from Loughborough University of Technology, Loughborough, U.K., in 1990.

She was a Postdoctoral Researcher with Kassel University, Kassel, Germany, in 2006. She is currently a Professor with the Department of Electrical Engineering, Federal University of Ceará, Fortaleza. Her research interests include grid integration of renewable energy sources, power quality, and microgrids.

ARTHUR P. S. BRAGA received the B.Sc. degree in electrical engineering from the Federal University of Ceará, Fortaleza, Brazil, in 1995, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of São Paulo (USP), São Carlos, Brazil, in 1998 and 2004, respectively.

He held a postdoctoral position with the University of São Paulo, in 2006. He is currently an Adjunct Professor with the Federal University of Ceará, Fortaleza. His research interests include machine learning, autonomous agents, metaheuristics, and fuzzy systems.