Dowell, Jethro and Weiss, Stephan and Infield, David (2015) Kernel methods for short-term spatio-temporal wind prediction. In: 2015 IEEE Power and Energy Society General Meeting. IEEE. ISBN 9781467380409 , http://dx.doi.org/10.1109/PESGM.2015.7285965

Kernel Methods for Short-term Spatio-temporal Wind Prediction

Jethro Dowell, Stephan Weiss and David Infield
University of Strathclyde
Glasgow, UK
Email: jethro.dowell@strath.ac.uk

***Abstract*—Two nonlinear methods for producing short-term spatio-temporal wind speed forecast are presented. From the relatively new class of *kernel methods*, a kernel least mean squares algorithm and kernel recursive least squares algorithm are introduced and used to produce 1 to 6 hour-ahead predictions of wind speed at six locations in the Netherlands. The performance of the proposed methods are compared to their linear equivalents, as well as the autoregressive, vector autoregressive and persistence time series models. The kernel recursive least squares algorithm is shown to offer significant improvement over all benchmarks, particularly for longer forecast horizons. Both proposed algorithms exhibit desirable numerical properties and are ripe for further development.**

I. Introduction

Wind energy penetration is increasing on power systems around the world driven primarily by a desire to reduce reliance on carbon intensive alternatives. To optimally utilise variable renewable generation, such as wind power, power systems, and the way they are operated, are changing: transmission networks must connect distant renewable generation to load centres, distribution networks must accommodate small scale generation, and operators must consider the stochastic nature of this new variable generation when performing scheduling tasks [1]–[4]. Decisions relating to scheduling tasks are informed by forecasts on a variety of temporal and spatial scales, and the upper limit on the level of variable generation that can be accommodated by power systems will ultimately be set by the skill of these forecasts (and their users).

This paper is concerned with short-term wind speed forecasting, a key input when predicting wind power. Unlike day-ahead forecasting, where numerical weather predictions are essential, short-term forecasts (up to approximately 6 hours) produced by purely statistical methods are superior to complex physical models [5]. Spatio-temporal approaches, which model and forecast multiple locations simultaneously to capture space-time dependencies, have been shown to improve skill significantly when compared to forecasts which only consider individual locations.

To date, the majority of statistical methods used for wind speed prediction have been linear despite the well established nonlinear nature of the wind. We therefore explore a relatively new and exciting class of learning algorithms called *kernel methods* which enable the linear processing of nonlinear 'features'. This approach retains many desirable properties of linear processing, such as fast learning algorithms and the existence of a unique optimal solution, while making it possible to capture some nonlinearities.

Over the last decade, many kernel methods have been developed and now represent a distinct class of learning algorithms. Such methods are based on the so called 'kernel trick', a result which allows the inner product of a nonlinear function defined by a Mercer kernel (Mercer's Theorem [6]) to be calculated while the function itself remains unknown [7]. This has advantageous properties in function estimation and classification; support vector machines, for example, rely on kernel methods.

A direct application of nonlinear function estimation is regression, where some nonlinear mapping is followed by linear processing in a high (or infinite) dimensional feature space. The kernel trick removes the need to identify the mapping associated with the a given Mercer kernel which may not be available or be difficult to calculate; the only challenge is selecting an appropriate kernel for the problem at hand.

Several linear methods have been 'kernelised' including the popular least means squares (LMS) [8] and recursive least squares (RLS) [9] algorithms, plus extensions, [10]–[13] for example. Reported applications to forecasting include high frequency wind prediction [13], [14] and load forecasting [15], among others.

In this paper we examine the application of two kernel methods to the short-term wind forecasting problem: a simple kernelised LMS algorithm and the kernel RLS of [9] are studied. The theory of Kernel methods is briefly introduced in Section II and the prediction problem is stated in III followed by descriptions of the KLMS and KRLS algorithms in III-B and III-C, respectively. A case study is then presented in Section IV, including how the algorithms and benchmarks were implemented, and their performance is evaluated. Finally, conclusions are drawn in Section V.

II. Kernel Methods

Kernel methods are a class of learning algorithm which use Mercer kernels in order to produce nonlinear versions of conventional linear learning algorithms. The kernel trick allows the inner product of two input vectors in some high-dimensional Hilbert space $\mathcal{H}$ (often called the feature space) to be calculated without explicit knowledge of the feature vectors, which form a the nonlinear projection of the input vectors in $\mathcal{H}$.

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

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}_1} \quad , \tag{1}$$

where $\mathcal{H}_1$ is a real- or complex-valued reproducing kernel 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 (1) states that if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are mapped onto $\mathcal{H}_1$ by $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$, respectively, then the inner product of these functions can be calculated by evaluating the kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, even if the mapping $\phi(\cdot)$ is unknown. This result is known as the Kernel trick.

The Gaussian kernel is frequently used in real world applications with particular success in time series prediction problems. It is the expansion function for an infinite dimensional feature space and is given by

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2\right) \tag{2}$$

and used throughout this study [7]. While other kernels could be chosen, the Gaussian kernel has a physical interpretation as a measure of similarity, which is fitting here, and has out performed other candidate kernels (triangular and polynomial) in other similar work [9], [13]. The choice, or construction, of kernels is very much an open problem and the subject of ongoing research.

III. Prediction Algorithms

A. Prediction Set-up

The prediction problem is outlined in Fig. 1, whereby the purpose of a potentially nonlinear function $f(\cdot)$ is to look Δ samples ahead in time, estimating $\boldsymbol{y}_t \in \mathbb{R}^n$ (or $\mathbb{C}^n$) from the input vector $\boldsymbol{x}_t \in \mathbb{R}^m$ (or $\mathbb{C}^m$) containing space-time data comprising measurements $\boldsymbol{y}_{t-\Delta},\ \boldsymbol{y}_{t-\Delta-1}, \ldots$. The predictor is written

$$\boldsymbol{y}_t = f(\boldsymbol{x}_t) \quad . \tag{3}$$

The aim in the context of a prediction problem is to find an estimate $\hat{f}(\cdot)$ of $f(\cdot)$ which minimises the estimation error in the mean squared sense, i.e.

$$J = \sum_t |\boldsymbol{e}_t|^2 = \sum_t |\boldsymbol{y}_t - \hat{f}(\boldsymbol{x}_t)|^2 \tag{4}$$

The linear approximation of this problem is given by $\hat{f}(\boldsymbol{x}_t) = \mathbf{A}\boldsymbol{x}_t$ where $\mathbf{A} \in \mathbb{R}^{n\times m}$ (or $\mathbb{C}^{n\times m}$) is a coefficient matrix whose entries are to be determined. Many estimation schemes based on this approximation have been studied.

Alternatively, we can state the approximation in terms of the mapping $\phi(\cdot)$ to place us in a nonlinear setting, writing $\hat{f}(\boldsymbol{x}_t) = \mathbf{A}\phi(\boldsymbol{x}_t)$ with $\mathbf{A} \in \mathbb{R}^{n\times l}$ (or $\mathbb{C}^{n\times l}$). The properties of Mercer kernels make it possible to derive estimation schemes for $f(\cdot)$ in a high l-dimensional feature space without performing calculations in such a space. This combines simple implementation of linear methods with the advantageous properties of working with a nonlinear mapping.

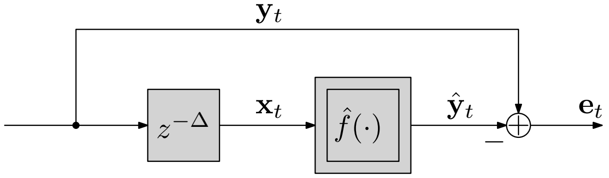

Fig. 1. Nonlinear system $f(\cdot)$ set-up to predict the vector quantity $\boldsymbol{y}_t$ some Δ samples ahead based on input $\boldsymbol{x}_t$.

In the remainder of this section two popular linear algorithms are discussed, the least mean squares (LMS) and recursive least squares (RLS) algorithms, are presented in their conventional linear and kernelised forms. Since we are only concerned with real valued wind speed, the proceeding sections assume all quantities are real valued and the conjugations necessary in the complex case are omitted.

B. Kernel LMS

The LMS algorithm comprises an update scheme based on gradient descent for the coefficient matrix $\mathbf{A}$ given by

$$\begin{aligned} \mathbf{A}_0 &= \mathbf{0}_{n\times m} & (5)\\ \boldsymbol{e}_t &= \boldsymbol{y}_t - \mathbf{A}_t\boldsymbol{x}_t & (6)\\ \mathbf{A}_{t+1} &= \mathbf{A}_t + \mu\boldsymbol{e}_t\boldsymbol{x}_t^{\mathrm{T}} \quad , & (7) \end{aligned}$$

where the prediction $\hat{\boldsymbol{y}}_t = \mathbf{A}_t\boldsymbol{x}_t$ and μ is the positive learning rate which controls the trade-off between confidence in individual samples and convergence speed.

When kernelised, the update step (7) becomes

$$\mathbf{A}_{t+1} = \mathbf{A}_t + \mu\boldsymbol{e}_t\phi^{\mathrm{T}}(\boldsymbol{x}_t) \quad , \tag{8}$$

however to express the algorithm in terms of inner products it is more convenient to wright

$$\mathbf{A}_{t+1} = \mu\sum_{i=1}^{t}\boldsymbol{e}_i\phi^{\mathrm{T}}(\boldsymbol{x}_i) \quad , \tag{9}$$

which allows the prediction to be expressed as

$$\hat{\boldsymbol{y}}_t = \mu\sum_{i=1}^{t}\boldsymbol{e}_i\phi^{\mathrm{T}}(\boldsymbol{x}_i)\phi(\boldsymbol{x}_t) \quad . \tag{10}$$

Finally, if $\phi(\cdot)$ is chosen to be an expansion function of some reproducing kernel Hilbert space, the inner product in (10) can be computed using the corresponding generating kernel $k(\cdot,\cdot)$, i.e.

$$\hat{\boldsymbol{y}}_t = \mu\sum_{i=1}^{t}\boldsymbol{e}_i k(\boldsymbol{x}_i, \boldsymbol{x}_t) \quad . \tag{11}$$

Notice however that as t increases so does the number of terms in the sum required to produce and estimate. This quickly becomes impractical and must be avoided. We therefore impose a sparsity constraint, by retaining a finite *dictionary*, D, of input vectors. At each time step t the input vector is compared to the dictionary D_{t-1}; if the minimum distance between $\boldsymbol{x}_t$ and D_{t-1} exceeds some sparsity parameter ν, it is added to the dictionary, i.e. $D_t := D_{t-1} \cup \{\boldsymbol{x}_t\}$, else $D_t := D_{t-1}$. The estimation is now

$$\hat{\boldsymbol{y}}_t = \mu \sum_{i \in D_{t-1}} \boldsymbol{e}_i k(\boldsymbol{x}_i, \boldsymbol{x}_t) \quad . \tag{12}$$

C. Kernel RLS

The RLS algorithm attempts to minimise the cost function (4) at each time step, rather than the mean squared error as in the LMS algorithm. The cost function is rewritten as

$$J(\boldsymbol{w}) = \sum_{i=1}^{t} (\boldsymbol{y}_i - \mathbf{A}\phi(\boldsymbol{x}_i))^2 = |\mathbf{Y}_t - \mathbf{\Phi}_t^{\mathrm{T}} \boldsymbol{w}|^2 \quad , \tag{13}$$

where $\mathbf{Y}_t$ and $\mathbf{\Phi}_t$ are output and projected input data matrices, and $\boldsymbol{w}$ is a weight vector. As before, working in some high dimensional feature space is undesirable, so writing the optimal weight vector as $\boldsymbol{w}_t = \sum_{i=1}^{t} \alpha_i \phi(\boldsymbol{x}_i) = \mathbf{\Phi}_t \boldsymbol{\alpha}$ we use the kernel trick to express the cost function as

$$J(\boldsymbol{\alpha}) = |\mathbf{Y}_t - \mathbf{K}_t \boldsymbol{\alpha}|^2 \quad , \tag{14}$$

where $[\mathbf{K}_t]_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j);\ i, j = 1, ..., t$, is called the kernel matrix.

In theory the minimiser $\boldsymbol{\alpha} = \mathbf{K}_t^{-1} \mathbf{Y}_t$ could be computed recursively using the conventional RLS algorithm, however, as with the kernalised LMS algorithm, the complexity of the calculation would increase with each new sample, in addition to possible over-fitting when the number of samples becomes large. We therefore sparsify the algorithm by retaining a finite dictionary of samples and replacing $\mathbf{K}_t$ with the dictionary kernel matrix $[\tilde{\mathbf{K}}_t]_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j);\ i, j \in D$, and $\boldsymbol{\alpha}_t$ with the reduced weights $\tilde{\boldsymbol{\alpha}}_t$.

The derivation of the KRLS algorithm is lenghthy and involved, and space here is limited so the update steps are stated without proof. The full derivation, and pseudo code of its implementation, can be found in the original paper [9].

```
*Initialisation:* $\tilde{\mathbf{K}}_1 = [k(\boldsymbol{x}_1, \boldsymbol{x}_1)]$, $\tilde{\mathbf{K}}_1^{-1} = [1/\tilde{\mathbf{K}}_1]$, $\tilde{\boldsymbol{\alpha}}_1 = [\boldsymbol{y}_1/\tilde{\mathbf{K}}_1]$, $\mathbf{P}_1 = [1]$, sparsity parameter: $\nu$.
**for** $t$=2,3,...
  *Compute:* $\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)$ where $[\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)]_i = k(\boldsymbol{x}_i, \boldsymbol{x}_t), i \in D$
  *Make Prediction:* $\hat{\boldsymbol{y}}_t = \tilde{\boldsymbol{k}}_{t-1}^{\mathrm{T}}(\boldsymbol{x}_t)\tilde{\boldsymbol{\alpha}}$
  *Compute:* $\boldsymbol{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)$
  *Compute:* $\delta_t = k(\boldsymbol{x}_t, \boldsymbol{x}_t) + \tilde{\boldsymbol{k}}_{t-1}^{\mathrm{T}}(\boldsymbol{x}_t)\boldsymbol{a}_t$
  *Case 1:* The new sample is approximately linearly independent with respect to the current dictionary, satisfying $\delta_t > \nu$, therefore the new sample, $\boldsymbol{x}_t$, is added to the dictionary. The matrices $\tilde{\mathbf{K}}_t^{-1}$, $\mathbf{P}_t$ and $\tilde{\boldsymbol{\alpha}}_t$ are updated as follows:
      $\tilde{\mathbf{K}}_t^{-1} = \frac{1}{\delta_t}\begin{bmatrix} \tilde{\mathbf{K}}_{t-1}^{-1} + \boldsymbol{a}_t\boldsymbol{a}_t^{\mathrm{T}} & -\boldsymbol{a}_t \\ -\boldsymbol{a}_t^{\mathrm{T}} & 1 \end{bmatrix}$   (15)
      $\mathbf{P}_t = \begin{bmatrix} \mathbf{P}_{t-1} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$   (16)
      $\tilde{\boldsymbol{\alpha}}_t = \begin{bmatrix} \tilde{\boldsymbol{\alpha}}_{t-1} - \frac{\boldsymbol{a}_t}{\delta_t}(\boldsymbol{y}_t - \tilde{\boldsymbol{k}}_{t-1}^{\mathrm{T}}(\boldsymbol{x}_t)\boldsymbol{a}_t) \\ \frac{1}{\delta_t}(\boldsymbol{y}_t - \tilde{\boldsymbol{k}}_{t-1}^{\mathrm{T}}(\boldsymbol{x}_t)\boldsymbol{a}_t) \end{bmatrix}$   (17)
  *Case 2:* The new sample is approximately linearly dependent with respect to the current dictionary, satisfying $\delta_t \leq \nu$, so the new sample is not added to the dictionary and $\tilde{\mathbf{K}}_t^{-1} = \tilde{\mathbf{K}}_{t-1}^{-1}$. The matrices $\mathbf{P}_t$ and $\tilde{\boldsymbol{\alpha}}_t$ are updated as follows:
      $\mathbf{P}_t = \mathbf{P}_{t-1} - \frac{\mathbf{P}_{t-1}\boldsymbol{a}_t\boldsymbol{a}_t^{\mathrm{T}}\mathbf{P}_{t-1}}{1 + \boldsymbol{a}_t^{\mathrm{T}}\mathbf{P}_{t-1}\boldsymbol{a}_t}$   (18)
      $\boldsymbol{q}_t \overset{\text{def}}{=} \frac{\mathbf{P}_{t-1}\boldsymbol{a}_t}{1 + \boldsymbol{a}_t^{\mathrm{T}}\mathbf{P}_{t-1}\boldsymbol{a}_t}$   (19)
      $\tilde{\boldsymbol{\alpha}}_t = \tilde{\boldsymbol{\alpha}}_{t-1} + \tilde{\mathbf{K}}_t^{-1}\boldsymbol{q}_t\left(\boldsymbol{y}_t - \tilde{\boldsymbol{k}}_{t-1}^{\mathrm{T}}(\boldsymbol{x}_t)\boldsymbol{a}_t\right)$   (20)
**end**
```

D. Sparsity

Both the KLMS and KRLS algorithms retain a dictionary of input samples as a sparse representation of the complete history of input samples up to some time. It is an important property of the dictionary that it is finite, and it can be shown to be so with only mild conditions on the data and kernel function. If $\boldsymbol{x} \in \mathcal{X}$ and $\phi(\boldsymbol{x}) \in \mathcal{H}$ then if $\mathcal{X}$ is compact, and the sparsity parameter is positive ($\nu > 0$), the dictionary will be finite. For a rigorous proof, see [9].

It should be noted that for these algorithms to be fully adaptive they should incorporate some forgetting mechanism whereby out-of-date dictionary elements are 'forgotten' in order track changing dynamics of the process being modelled. A sophisticated multi-kernel LMS algorithm is developed [13] which includes this feature, as well as the ability to combine multiple kernel functions. The drawback, of course, is the need to determine the parameters for each of these mechanisms.

IV. Case Study

A. Test Data

The data used for testing is from the Hydra dataset of hourly mean potential wind at multiple locations across the Netherlands. Six locations within 150km of each other are considered here with measurements from 2001 used as a training set and data from 2002 used for testing. The measured wind speed has been corrected for the effects of shelter from buildings or vegetation. The resulting *potential* wind is an estimate of the wind speed that could have been measured at 10m height if the station's surroundings were free of obstacles and flat with a roughness length equal to that of grass onshore (0.03m) and water offshore (0.002m). For more information on this process see [16]. In addition, the data as been normalised so that it occupies the range $[0, 1]$.

This transformation aids spatial prediction by removing biases present at individual measurement locations that would otherwise interfere with the spatio-temporal correlation of the data. The procedure is simple to implement once information regarding the terrain surrounding a weather station is known.

B. Implementation

The KLMS and KRLS are employed to predict the wind speed at the six locations which are embedded in the vector $\boldsymbol{y}_t \in \mathbb{R}^6$. The input vector $\boldsymbol{x}_{\Delta|t}$ for a $\Delta = 1$ step-ahead prediction is the concatenation of p lagged values of $\boldsymbol{y}_t$, i.e. $\boldsymbol{x}_{1|t} = (\boldsymbol{y}_{t-1}^{\mathrm{T}}, ..., \boldsymbol{y}_{t-p}^{\mathrm{T}})^{\mathrm{T}}$, and for horizons of $\Delta > 1$ where not all lags are available, predictions are used such that

$$\boldsymbol{x}_{\Delta|t} = \begin{bmatrix} \hat{\boldsymbol{y}}_{\Delta-1|t-1} \\ \vdots \\ \hat{\boldsymbol{y}}_{1|t-\Delta+1} \\ \boldsymbol{y}_{t-\Delta} \\ \vdots \\ \boldsymbol{y}_{t-p} \end{bmatrix} \quad , \tag{21}$$

where $\hat{\boldsymbol{y}}_{\Delta|t}$ denotes the prediction of $\boldsymbol{y}_t$ made Δ steps ahead. This is the so called *direct forecasting* approach. In this study we are only concerned with forecast horizons up to 6 hours, or $\Delta = 1, ..., 6$ and the forecasts for each horizon are made in parallel by distinct predictors. The number of temporal lags is a trade off between accuracy and computational expense, for the KLMS $p = 6$ and KRLS $p = 3$ since the improvement in accuracy was negligible for greater values.

The sparsification parameter and LMS learning rate are determined heuristically by exhaustive search to minimise the residual error on the training data. The sparsification parameter is $\nu = 0.02$ for the KRLS and $\nu = 0.1$ for the KLMS. The KLMS learning rate was chosen to be $\mu = 0.01$.

C. Benchmarks

An important benchmark is the persistence forecast which supposes that the future wind speed will be the same as the most recent measurement. While its implementation is trivial its performance is still considered acceptable by many practitioners, particularly in situations where more complex approaches offer only modest gains. The persistence forecast Δ-hours ahead is given by

$$\hat{\boldsymbol{y}}_{\Delta|t} = \boldsymbol{y}_{t-\Delta} \quad . \tag{22}$$

In order to compare the kernelised algorithms to conventional techniques and highlight the value of spatial information, two non-recursive linear time series models are used as further benchmarks in addition to the conventional LMS and RLS algorithms. The first is the non-spatial autoregressive (AR) model which is given by

$$\hat{y}_{\Delta|t} = \sum_{i=1}^{\Delta-1} a_i \hat{y}_{\Delta-i|t-i} + \sum_{i=\Delta}^{p} a_i y_{t-i} \tag{23}$$

for each location. The number of lags p is determined by the Akaike information criterion, and the parameters a_i are determined by maximum likelihood estimation assuming independent identically distributed (i.i.d) Gaussian prediction errors.

The second is the vector generalisation of AR, the vector autoregressive model (VAR). As in the multivariate kernelised algorithms, the measurements at multiple locations are embeded in the vector $\boldsymbol{y}_t$ and the model is written

$$\hat{\boldsymbol{y}}_{\Delta|t} = \mathbf{A}\boldsymbol{x}_{\Delta|t} \quad , \tag{24}$$

where $\boldsymbol{x}_t$ is given by Equation (21). Once again the number of lags p is determined by the Akaike information criterion and assuming i.i.d Gaussian errors the coefficient matrix $\mathbf{A} \in \mathbb{R}^{n \times np}$ is determined by maximum likelihood estimation.

The conventional LMS algorithm, with update scheme given in equations (5)–(7) and learning rate $\mu = 0.0005$, and conventional RLS with update scheme

$$\begin{aligned} \boldsymbol{e}_t &= \boldsymbol{y}_t - \mathbf{A}_t \boldsymbol{x}_t & (25) \\ \boldsymbol{k}_t &= \frac{\boldsymbol{x}_t^{\mathrm{T}} \mathbf{Q}_t}{1/\lambda + \boldsymbol{x}_t^{\mathrm{T}} \mathbf{Q}_t \boldsymbol{x}_t} & (26) \\ \mathbf{Q}_{t+1} &= \mathbf{Q}_t - \mathbf{Q}_t \boldsymbol{x}_t \boldsymbol{k}_t & (27) \\ \mathbf{A}_{t+1} &= \mathbf{A}_t + \boldsymbol{e}_t \boldsymbol{k}_t \quad , & (28) \end{aligned}$$

and forgetting factor $\lambda = 0.9995$ are also included for comparison with their kernelised versions. The look-ahead indexing has been dropped here to avoid notational clutter but the principle still applies.

The AR and VAR methods are non-recursive, that is to say that their parameters are estimated directly from the training data and are then fixed throughout the test period. The other methods, with the exception of persistence, are recursive and as such are initialised and then run sequentially through the training and test data, updating their parameter estimates at each step.

D. Results

Performance is evaluated in terms of root mean squared error, which is given by the expression

$$\mathrm{RMSE}_\Delta = \sqrt{\frac{1}{T} \sum_{t=1}^{T} (\hat{y}_{\Delta|t} - y_t)^2} \tag{29}$$

for the samples $y_1, ..., y_T$ in the test dataset at each location and for each forecast horizon $\Delta = 1, ..., 6$.

The performance of the kernelised algorithms and benchmarks is illustrated in Figure 2 in terms of mean RMSE accross the six sited in the dataset. The most simplistic approaches, persistence and AR perform significantly worse at all forecast horizons than the more sophisticated VAR, RLS and KRLS. Both the LMS and its kernalised version (KLMS) have intermediate performance, reflective of their complexity, though the KLMS performs particularly poorly for the 1 and 2 hour ahead predictions.

The improvement over persistence is shown in Figure 3 for the VAR, RLS and KRLS predictions. All three exhibit similar performance for 1 and 2 hour ahead forecasts, but the KRLS

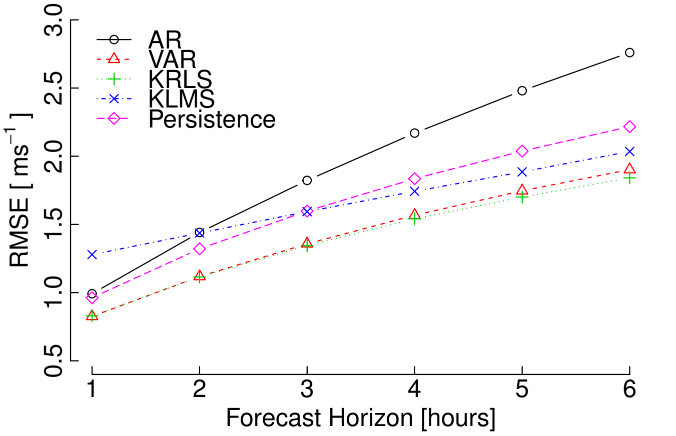

Fig. 2. Mean root mean squared error across all 6 sites for Kernelised algorithms and benchmarks.

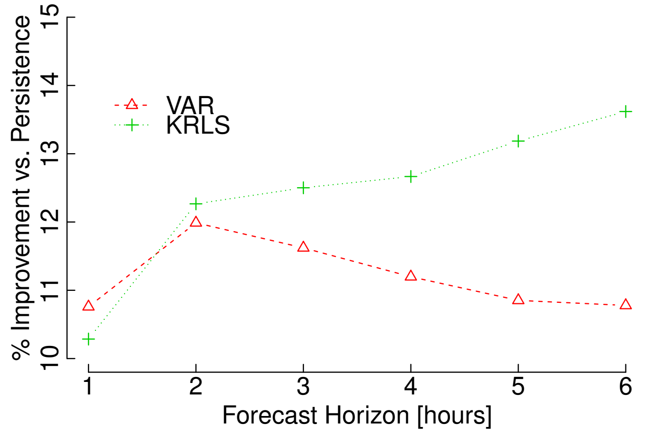

Fig. 3. Percentage improvement vs. persistence for the VAR and KRLS forecasts.

outperforms the two linear methods for the longer horizons. It is also notable that the KLMS improves relative to the LMS at longer forecast horizons. In both cases the kernelised versions of linear algorithms offer improved 5 and 6 hour ahead predictions.

V. Conclusions

For the short-term spatio-temporal prediction of wind speed, we have presented two examples from a new class of learning algorithms called kernel methods. The kernel least mean squares and kernel recursive least squares algorithms are non-linear extensions of their conventional linear forms and have been applied to a dataset comprising wind speed measurements made at six locations in the Netherlands over a period of 2 years. The KRLS in particular shows significant improvement over several established linear benchmarks, especially for longer forecast horizons.

Acknowledgements

The authors gratefully acknowledge the Royal Netherlands Meteorological Institute for their supply of meteorological data, and the support of the UK's Engineering and Physical Sciences Research Council via the University of Strathclyde's Wind Energy Systems Centre for Doctoral Training, grant number EP/G037728/1.

References

[1] S. J. Watson, L. Landberg, and J. A. Halliday, "Application of wind speed forecasting to the integration of wind energy into a large scale power system," *IEE Proceedings—Generation, Transmission and Distribution*, vol. 141, no. 4, pp. 357–362, 1994.

[2] E. Denny and M. O'Malley, "Wind generation, power system operation, and emissions reduction," *IEEE Transactions on Power Systems*, vol. 21, no. 1, pp. 341–347, 2006.

[3] E. Bitar, R. Rajagopal, P. Khargonekar, K. Poolla, and P. Varaiya, "Bringing wind energy to market," *IEEE Transactions on Power Systems*, vol. 27, no. 3, pp. 1225–1235, 2012.

[4] S. Faias, J. De Sousa, F. Reis, and R. Castro, "Assessment and optimization of wind energy integration into the power systems: Application to the portuguese system," *IEEE Transactions on Sustainable Energy*, vol. 3, no. 4, pp. 627–635, 2012.

[5] G. Giebel, R. Brownsword, G. Kariniotakis, M. Denhard, and C. Draxl, *The State-of-the-Art in Short-Term Prediction of Wind Power*. ANEMOS.plus, 2011, project funded by the European Commission under the 6th Framework Program, Priority 6.1: Sustainable Energy Systems.

[6] J. Mercer, "Functions of positive and negative type, and their connection with the theory of integral equations," *Philosophical Transactions of the Royal Society of London, Series A*, vol. 209, no. 441–458, pp. 415–446, 1909.

[7] B. Schølkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[8] W. Liu, P. Pokharel, and J. Principe, "The kernel least-mean-square algorithm," *IEEE Transactions on Signal Processing*, vol. 56, no. 2, pp. 543–554, Feb 2008.

[9] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, Aug 2004.

[10] W. Liu, I. M. Park, Y. Wang, and J. Principe, "Extended kernel recursive least squares algorithm," *IEEE Transactions on Signal Processing*, vol. 57, no. 10, pp. 3801–3814, Oct 2009.

[11] S. Van Vaerenbergh, M. Lazaro-Gredilla, and I. Santamaria, "Kernel recursive least-squares tracker for time-varying regression," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1313–1326, Aug 2012.

[12] F. A. Tobar, A. Kuh, and D. P. Mandic, "A novel augmented complex valued kernel LMS," in *7th Sensor Array and Multichannel Signal Processing Workshop*. Hoboken, NJ, USA: IEEE, June 2012.

[13] F. Tobar, S.-Y. Kung, and D. Mandic, "Multikernel least mean square algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 265–277, Feb 2014.

[14] A. Kuh, C. Manloloyo, R. Corpuz, and N. Kowahl, "Wind prediction using complex augmented adaptive filters," in *International Conference on Green Circuits and Systems*, June 2010, pp. 46–50.

[15] C. Liu and F. Liu, "The short-term load forecasting using the kernel recursive least-squares algorithm," in *3rd International Conference on Biomedical Engineering and Informatics*, vol. 7, Oct 2010, pp. 2673–2676.

[16] Royal Netherlands Meteorological Institute, (2005, October) Hydra potential wind data set. [Online]. Available: http://www.knmi.nl/samenw/hydra.