Neurocomputing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Neurocomputing

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

Wind energy prediction and monitoring with neural computation

Oliver Kramer [a,*], Fabian Gieseke [a], Benjamin Satzger [b]

[a] Computational Intelligence Group, Department of Computing Science, University of Oldenburg, 26111 Oldenburg, Germany
[b] Distributed Systems Group, Information Systems Institute, Vienna University of Technology, A-1040 Wien, Austria

ARTICLE INFO

Keywords:
Wind energy
Wind prediction
Support vector regression
Dimension reduction
Time-series monitoring
Self-organizing feature maps

ABSTRACT

Wind energy has an important part to play as renewable energy resource in a sustainable world. For a reliable integration of wind energy high-dimensional wind time-series have to be analyzed. Fault analysis and prediction are an important aspect in this context. The objective of this work is to show how methods from neural computation can serve as forecasting and monitoring techniques, contributing to a successful integration of wind into sustainable and smart energy grids. We will employ support vector regression as prediction method for wind energy time-series. Furthermore, we will use dimension reduction techniques like self-organizing maps for monitoring of high-dimensional wind time-series. The methods are briefly introduced, related work is presented, and experimental case studies are exemplarily described. The experimental parts are based on real wind energy time-series data from the National Renewable Energy Laboratory (NREL) western wind resource data set.

1. Introduction

Wind has an important part to play in sustainable energy systems. An important aspect for a stable and an efficient energy grid is to predict *when*, and *how strong* the wind is blowing for a single windmill, or a whole wind park. A precise understanding and forecast of wind energy is very important to reduce carbon dioxide emissions caused by reserve energy sources. Advanced monitoring techniques for wind time-series data are required to understand changes and to recognize errors in the grid.

In this paper we employ methods from neural computation, i.e., *support vector regression* and *self-organizing maps*, and show that these methods play a key role for forecasting and monitoring of renewable energy. The work is an extension, and consistent depiction of the research results on SVR-based wind energy prediction [1], and monitoring [2,3]. We hope to demonstrate their potential in contributing to a sustainable system development. We will employ the following neural- and kernel-based machine learning methods:

- Support vector regression for wind energy forecasting in Section 3 and
- self-organizing feature maps for time-series monitoring in Section 4.

Before we come to the methodological parts of this paper, the following section gives a brief introduction to the NREL wind data sets our analyses are based on. In Section 5 we will summarize the main results of this paper.

2. Wind data

The data for analysis is based on the NREL western wind resources data set [4,5]. The western wind resources data set is part of the *Western Wind and Solar Integration Study*, which is a large regional wind and solar integration study in USA. The data has been designed to perform temporal and spatial comparisons like load correlation, or estimation of production from hypothetical wind farms for demand analysis and planning of storage based on wind variability. The western wind resources data set was partly created with the help of numerical weather predictions. The data was sampled every 10 min and every 2 km; 1.2 million grid points have been aggregated to 32,043 locations. Each grid point is estimated to hold 10 Vestas 3 MW turbines, and therefore, 32,043 locations in total employ more than 960 GW of capacity. The set contains data of 2004, 2005, and 2006. The GUI of the western wind data set allows to select grid points and download the corresponding times series data, see Appendix A.

For forecasting and sequence visualization the wind data is aggregated to a time-series training set in a preprocessing step. This training set has a vertical and a horizontal component:

- vertical (rows): the vertical axis corresponds to the choice of grid points, the number of grid points determines the dimension of the data set and

* Corresponding author.
E-mail address: oliver.kramer@uni-oldenburg.de (O. Kramer).

0925-2312/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.07.029

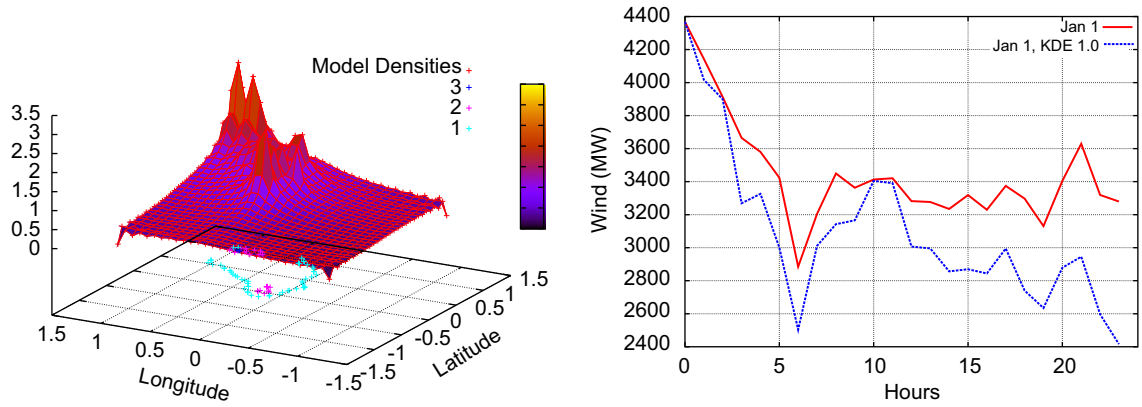

Fig. 1. Left: Kernel density visualization with Gaussian kernel and Silverman's rule of the wind park southeast of Salt Lake City. The figure visualizes the density of grid points with regard to normalized coordinates. Right: Sampled kernel density estimation model of wind changes and original wind times series of the first two days in January 2006 of one grid point in Salt Lake City.

- horizontal (lines): the horizontal axis corresponds to the chosen time window (e.g., aggregation of hours, days, etc.)

The wind data is a time-series of N wind measurements of K grid points, i.e., $\mathbf{x}_1,\ldots,\mathbf{x}_N$ with $\mathbf{x}_i=(x_1,\ldots x_K)^T$. Potter et al. [5] describe how the NREL data has been created. The data has been measured every 10 min, resulting in 52,560 measurements a year.

The left part of Fig. 1 shows a kernel density estimation [6] of the Salt Lake City wind park: the figures illustrate the density of wind energy production. The kernel densities are basis of the regression model using a Gaussian kernel with Silverman's bandwidth adaptation rule [7]. On the right hand side a wind times series of the first two days in January 2006 of one grid point in Salt Lake City, and a corresponding sampled time-series based on its kernel density estimate is shown.

3. Wind forecasting with support vector regression

Up to now, the integration of decentralized energy into the grid is as good as ignored. It is estimated that the stability of the energy grid decreases, if the amount of ignored renewable energy exceeds about 15–20%. But wind resources are steadily increasing. For a reasonable integration of volatile resources like wind, a precise prediction for subhourly scheduling becomes necessary. Precise forecast will allow balancing and integration of multiple volatile power sources at all levels of the transmission and distribution grid [8]. Consequently, the forecast of renewable resources plays an important role for a stable energy grid. The amount of wind power crucially depends on the speed of the wind, as the power is proportional to the cube of the wind speed. Hence, small differences in the wind speed result in significant differences in the produced power.

In this section we investigate the questions if prediction of wind energy can exclusively be based on the existing infrastructure of windmills and their wind speed measurements, and what are the limitations of state-of-the-art regression techniques for wind energy time-series forecasting. To answer these questions we will conduct experiments based on the NREL data employing a state-of-the-art kernel regression method, i.e., support vector regression (SVR) by Vapnik [9], which is related to support vector machines [10,11]. Our analysis will be based on a direct mapping of wind speed measurements on the produced wind energy.

We formulate the wind forecasting task as regression problem. Again, we assume that a time-series $\mathbf{x}_1,\ldots,\mathbf{x}_N$ of N wind measurements of K wind grid points, and corresponding measurements $y_1,\ldots,y_N$ of wind energy production of a target point is given. The task is to predict the wind energy production y_t at time $t=t_i+\theta$ with $\theta\in\mathbb{N}^+$ based on the past wind measurements at time $t_i,t_i-1,t_i-2,\ldots,t_i-\mu$, with $\mu\in\mathbb{N}^+$ past observations. The following questions arise:

- how many observations from the past do we need (i.e., how to choose μ to reduce the test error) and
- how far can we look into the future (i.e., how is the test error increasing with θ).

We concentrate on the production of a single grid point in Section 3.5 and on the large-scale level of a whole wind park in Section 3.6.

3.1. Related work

State-of-the-art techniques in regression have already been applied to energy forecasting. But the results are often limited to simplified case-studies of particular windmills, neglecting parameter analyses, or aspects like how far ahead we can reliably predict wind on a short-term level. As wind forecasting is an important task, a lot of publications can be found in the literature on different methods. Costa et al. [12] review 30 years of short-term prediction concentrating on forecasting methods, mathematical, statistical, and physical models, as well as meteorology. Negnevitsky et al. [13] review forecasting techniques used for power system applications with focus on electricity load, price forecasting, and wind power prediction. Milligan et al. [8] discuss if wind is a capacity resource. They state that for a single wind power plant, predictions on a 1 or 2 h basis can achieve an accuracy level of approximately 5–7% mean absolute error to installed wind capacity, increasing to 20% for day-ahead forecasts.

Machine learning approaches are successful methods for wind forecasting based on past observations. As an overview of all methods is not the scope of this work, we restrict our depiction to selected methods that are closely related to our approach. Many methods are based on neural networks. Shuhui et al. [14] estimate the wind energy production of single wind turbines at Central and South West Services Fort Davis. They discuss the structure and the number of neurons in a multi-layer perceptron for turbine power production of single windmills. Li et al. [15] introduced a robust two-step approach based on a Bayesian combination of three neural networks (e.g., backpropagation and radial basis functions networks). They demonstrate the approach for a one-hour forecast of two wind sites in North Dakota.

Preliminary work on SVR and wind forecasting has recently been introduced. Mohandes et al. [16] compared an SVR approach for

Please cite this article as: O. Kramer, et al., Wind energy prediction and monitoring with neural computation, Neurocomputing (2012), http://dx.doi.org/10.1016/j.neucom.2012.07.029

wind speed prediction to a multi-layer perceptron. The approach is based on mean daily wind speed data from Saudi Arabia. Shi et al. [17] proposed an approach that combines an evolutionary algorithm for parameter tuning with SVR-based prediction. The technique allows a six-hour prediction and is experimentally evaluated on wind data from North China. Recently, Zhao et al. [18] compared SVR to backpropagation for a 10 min prediction of wind speed. Further work concentrates on special aspects like prediction and diagnosis of wind turbine faults. Kusiak and Li [19] introduced an approach based on fault prediction on three levels, e.g., fault category and specific fault prediction in a 5 min to 1 h approach.

3.2. Support vector regression

We will apply the support vector regression (SVR) [9,20,21] model to address our prediction tasks. The approach is one of the state-of-the-art methods in regression. The goal of the learning process is to find a prediction function $\hat{f}: \mathcal{X} \to \mathbb{R}$ that assigns *good* predictions to unseen $x \in \mathcal{X}$ (e.g., $\mathcal{X} = \mathbb{R}^d$). Here, we only sketch the key ideas of the concept and refer to, e.g., Smola and Schölkopf [22] for a comprehensive overview. The SVR technique can be seen as a special case of regularization problems of the form

$$\inf_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2, \tag{1}$$

where $\lambda > 0$ is a fixed user-defined real value, $L: \mathbb{R} \times \mathbb{R} \to [0, \infty)$ is a loss function, and $\|f\|_{\mathcal{H}}^2$ is the squared norm in a so-called reproducing kernel Hilbert space $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{X}} = \{f : \mathcal{X} \to \mathbb{R}\}$ induced by an associated kernel function $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (which can be seen as similarity measure between the patterns). Plugging in different loss functions leads to different (but related) regression models. The so-called ε-insensitive loss $L_\varepsilon(y,t) = \max(|t-y| - \varepsilon, 0)$ with $\varepsilon > 0$ leads to

$$\inf_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \max(|f(\mathbf{x}_i) - y_i| - \varepsilon, 0) + \lambda \|f\|_{\mathcal{H}}^2, \tag{2}$$

and, hence, to the SVR approach.[1] Here, the first term corresponds to the difference between the values predicted by the function (i.e., model) and the corresponding real values given in the training set (residuals). The second term corresponds to the complexity of the model. Ideally, one would like to have a model that fits the data well, and that is not too complex at the same time to avoid overfitting. The ε-insensitive loss does not take into account small residual errors. The choice of ε defines the magnitude of errors that can be neglected. We will experimentally analyze various settings for ε in Section 3.4. Additionally, we will consider the square-loss $L_2(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$ for our experimental evaluation.

Support vector methods have found a broad acceptance in machine learning. Various extensions have been proposed, e.g., the lane classifier GEPSVM (proximal SVM classification via generalized eigenvalues) that classifies points by assigning them to the closest of two nonparallel planes generated by a corresponding eigenvalue problem [23]. A recent application of SVMs is the distributed classification in ad hoc sensor networks [24].

3.3. Experimental analysis

In the following, we will experimentally analyze forecasts with SVR based on the NREL western wind resources data sets. The analysis concentrates on wind grid points in the wind park of Tehachapi in California, USA. We employ the following experimental settings. The SVR is trained on 1/10-th of the observations from 2006. As core of the SVR implementation we employ LIBSVM [25]. In the experiments we make use of an RBF-kernel with kernel width σ. Furthermore, we employ grid search in the parameter space of λ, and σ of the RBF-kernel. Grid search makes use of a test data set based on the second 1/10-th of the one-year data, and tests the following values: 2^α with $\alpha = -15, \ldots, 15$. For time-critical applications we recommend to narrow the grid search bounds, as the successful parameters often lie in the range between $\sigma = 2^\alpha$ with $\alpha = -10, \ldots, -5$, and $\lambda = 2^\alpha$ with $\alpha = 5, \ldots, 10$ for the NREL wind data. The final validation error is computed based on the second 1/5-th of the corresponding data sets, using the L_ε, and the square-loss L_2.

3.4. Loss function parameter study

We start the analysis with tests of different loss function parameters for the SVR training process. The results will determine the choice of the ε-value in the remainder of this work. Table 1 shows the analysis of five values for ε that determine the magnitude of residual errors not contributing to the overall error during training. For comparison we state the L_ϵ and the L_2 loss on the validation set. The experiments are based on a 30-min forecast of wind based on two time steps (data of the last 20 min) from the past measurements of 15 wind grid points. The forecast is computed for the energy production of one wind grid point in the middle of the Tehachapi wind park. The results show that – as expected – the L_ε-error decreases with increasing tolerance threshold ϵ. But the L_2 loss has a minimum at $\epsilon = 0.5$. We assume that this setting is a reasonable choice for the following experiments.

3.5. Small-scale analysis: wind grid point level

The question is how far we can look into the future, and how much information from the past is necessary for a reliable forecast. Intuitively, we would expect that a static shot of the wind situation results in a loss of information, as no development, e.g., no change with regard to successive time steps is put into the model. Nevertheless, this intuition can be misleading as hidden correlations, and dependencies may exist. In the following, we do not rely on preliminary assumptions. We analyze the influence of the number of past time steps on the prediction error for an increasing number of steps we look ahead.

Table 2 shows the validation error for the energy forecast of a wind grid point in Tehachapi. It is based on 15 grid points from Tehachapi and neighbored wind parks within the range of about 50 miles. The figures show the validation error, i.e., L_ε- and L_2-loss on the validation set. From top to bottom the lines show predictions going further into the future. From left to right the figures show predictions that take more past time steps into account. One time step corresponds to 10 min. The results show that the error is increasing the further we try to predict the future energy production. L_ε- and L_2-loss are strongly correlated. Furthermore, the figures confirm a trend that is consistent with our expectations: the more past is taken into account the better the predictions become.

Table 1
Analysis of loss function parameter ϵ on the validation error measures with L_ε and L_2 loss.

Loss	0.01	0.1	0.5	1.0	2.0
L_ε	2.128	2.046	1.795	1.538	1.188
L_2	15.013	14.984	**14.365**	14.571	15.383

[1] Note that in the latter formulation, the offset b is omitted for simplicity.

Please cite this article as: O. Kramer, et al., Wind energy prediction and monitoring with neural computation, Neurocomputing (2012), http://dx.doi.org/10.1016/j.neucom.2012.07.029

Table 2
Forecasts for a single wind grid point in Tehachapi based on wind measurements of 15 grid points of Tehachapi and neighbored parks within a range of ca. 50 miles. The figures show the validation error with regard to increasing steps into the future (lines, top to bottom), and an increasing number of past measurements (columns, left to right).

Steps	1		2		3		6		12	
	L_ε	L_2	L_ε	L_2	L_ε	L_2	L_ε	L_2	L_ε	L_2
1	1.734	15.040	1.679	13.526	1.714	15.384	1.690	13.558	1.807	13.592
2	1.765	14.654	1.767	15.698	1.797	16.022	1.798	14.790	1.860	14.193
3	1.869	17.128	1.868	16.605	1.823	15.571	1.919	16.414	1.955	15.903
6	2.220	20.526	2.149	18.836	2.233	19.996	2.248	19.185	2.259	18.852
12	2.984	30.821	2.884	28.675	2.838	28.798	2.865	27.688	2.814	26.628

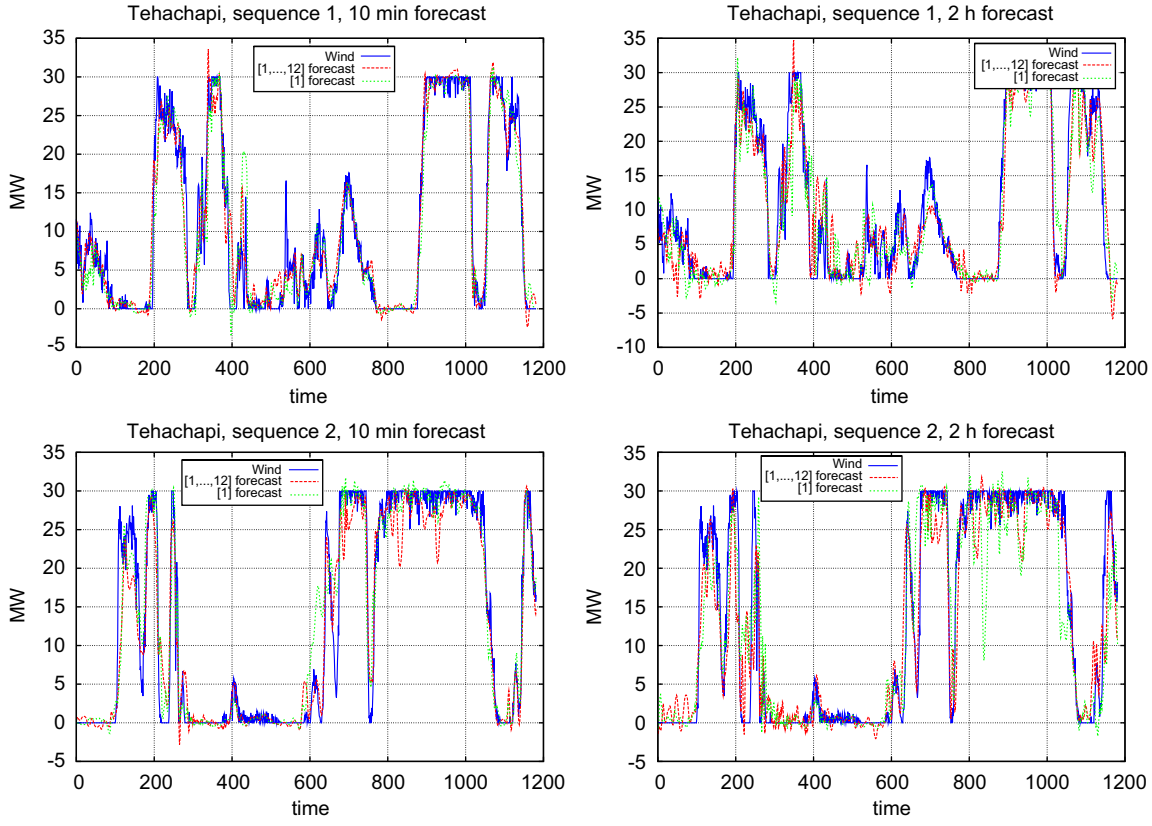

Fig. 2. Ten-minute (left figures) and two-hour (right figures) ahead wind energy forecasts for one Tehachapi wind grid point. Each figure shows the forecast based on 10 min and 2 h of past wind measurements. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Forecasts of sudden changes, e.g., caused by storm fronts passing, belong to the most important aspects. As a measure for the quality of forecasts is no easy undertaking, we employ a visual interpretation of two typical forecasts in the following. Fig. 2 shows two randomly chosen wind time-series from 2006 that are not basis of the training and testing process. The plots show the actual wind (blue/solid lines) and the forecasts based on a trained SVR model. Both plots on the left show 10-min forecasts, the plots on the right show two-hour forecasts. Red (dark dotted) lines show the forecast based on the data from the last 2 h (i.e., based on $12 \times$ 15-dimensional vectors), while green (bright dotted) lines show the forecasts only based on the last measurements 10 min ago (i.e., based on 15-dimensional vectors). In both situations we can observe that the 10 min ahead forecasts lead to very accurate results. In particular, the forecast based on the last 10 min leads to a reliable prediction. More deviations from the true curve can be observed, if we use the last 2 h for predictions. It is known that too much additional data can act like noise, and disturb the prediction [26]. The situation changes on the two-hour level, where the forecast based on wind measurements from the last 2 h leads to a higher accuracy. The forecast based on the 10-min level is much less reliable, and leads to larger deviations.

3.6. *Large-scale analysis: wind park level*

Large-scale forecasting on the level of wind parks has an important part to play for global control strategies. Besides the approach to aggregate the forecasts of all windmills of the whole park, the sum of energy can be taken into account. In the following, we conduct the prediction analysis on the level of a wind park near Salt Lake City that consists of 28 wind grid points. For the forecasts we employ a set of 100 randomly chosen wind grid points in the whole western area.

Table 3 shows the experimental results of the analysis with regard to various combinations of time steps from the past ([1,3]: 10 and 30 min, [3,6]: 30 and 60 min, and [6,12]: 60 and 120 min), and the steps we try to look into the future (from 10 min to 6 h). Similar to the previous section, the results show the corresponding validation error. Based on these figures we can observe the trend that the best forecast is achieved for the experiments

Please cite this article as: O. Kramer, et al., Wind energy prediction and monitoring with neural computation, Neurocomputing (2012), http://dx.doi.org/10.1016/j.neucom.2012.07.029

predicting one hour ahead. Looking further into the future decreases the forecast accuracies, but still results in an acceptable validation error. Short-term forecasts do also not result in the best validation errors. This is probably due to the fact that most of the windmills used for prediction are too far away to determine 10-min forecasts. They are spread across the whole western area of the US. For the one-hour ahead forecast the past information from the last 10 min and 30 min achieves the best validation error. But employing other combinations of time steps does not deteriorate the results significantly.

Fig. 3 shows a visualization of two random sequences, and the corresponding 10-min and 6-h ahead forecasts. The curves show the real wind that was blowing, and the forecasts, each based on the two past time steps. The plots show that all forecasts achieve a relatively high prediction accuracy that should be satisfying for most balancing activities in a smart grid. The predictions based on the last 2 h are even more reliable based on a 10-min forecast than the predictions based on the last 30 min. Also for the six-hour ahead forecast the prediction based on the [6,12]-data set results in the best curve. Local deviations from the true curve are more frequent in the case of the [1,3]-data set forecast.

Table 3
Forecast of wind energy production of wind park near Salt Lake City. The figures show the validation error for increasing time steps (lines, top to bottom) with regard to various time steps from the past (columns) taken into account.

Steps	[1,3]		[3,6]		[6,12]	
	L_ϵ	L_2	L_ϵ	L_2	L_ϵ	L_2
1	57.549	9044.233	57.218	9271.327	58.313	9148.557
6	58.786	9932.734	58.047	9355.095	57.745	9433.448
12	56.113	8774.924	56.879	8899.538	56.649	8822.972
24	58.448	9250.796	57.700	8965.454	56.869	8804.929
36	58.598	9599.905	59.171	9436.259	58.992	9968.387

3.7. Conclusion

Wind energy prediction is an important aspect for a stable grid. The integrity and stability can be improved the better a forecast of volatile energy sources is possible. We have demonstrated that SVR is a successful method for the prediction of wind energy production only based on wind measurements from windmills, in particular without further meteorological data or weather forecasts. SVR turns out to be a fast and robust time-series prediction technique. For the wind resource scenarios we have found recommendable parameters in case the ϵ-loss is employed. For fast training of the SVR model, typical bounds can be identified and the region of interest for grid search can be narrowed. The experiments have shown that a reliable forecast on the level of wind grid points is possible on the two-hour level, while the 10-min prediction leads to almost exact results. On the level of a whole wind park, the results have shown that even a reasonable six-hour forecast is possible.

4. Wind monitoring with dimensionality reduction methods

As wind is a volatile energy source, state observation has an important part to play for grid management, fault analysis, and planning strategies of grid operators. We demonstrate how an approach from unsupervised neural computation help to understand high-dimensional wind resource time-series. We propose a sequence analysis approach for high-dimensional wind time-series based on a quantization process with Kohonen's self-organizing map (SOM) [27]. It generates a topological low-dimensional representation of the

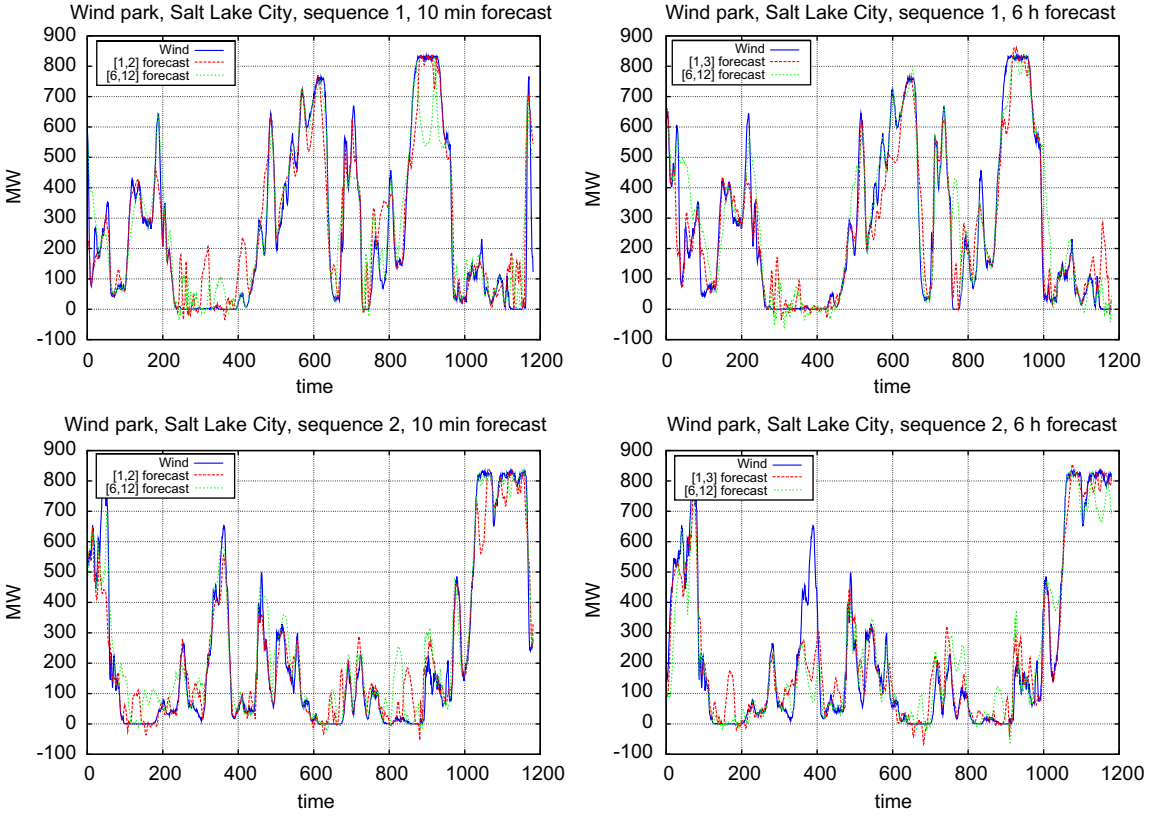

Fig. 3. Ten-minute (left figures) and six-hour (right figures) ahead forecasts for a wind park southeast of Salt Lake City on two randomly selected sequences (upper part, and lower part). Also in the case of the 6-h forecast the achieved accuracy is satisfactory.

Please cite this article as: O. Kramer, et al., Wind energy prediction and monitoring with neural computation, Neurocomputing (2012), http://dx.doi.org/10.1016/j.neucom.2012.07.029

high-dimensional data allowing a *smoother* quantization with regard to the data sample distribution than simply dividing the feature space into grids, or distributing codebook vectors equidistantly. A SOM is able to capture the intrinsic structure of the data. It can be used to visualize sequences or to perform further symbol-oriented analysis like string matching. At the end of this section we will employ dimensionality reduction methods for continuous latent spaces like PCA to visualize wind energy sequences.

4.1. Kohonen's learning algorithm

SOMs are biologically inspired neural models. The SOM algorithm has been proposed by Teuvo Kohonen [27] at the University of Helsinki, and is a dimension reduction method. Ritter et al. [28] motivate the SOM from the point of view of computational neuroscience. In the following, we briefly repeat the SOM learning algorithm. The SOM consists of a map of neurons $n_1,\ldots,n_K$, each with a weight vector $\mathbf{w}_i$. Furthermore, each neuron n_k exhibits a position on a map. Fig. 4 shows the pseudocode of the algorithm. At the beginning, the neural weights are initialized with random values. During the training phase the high-dimensional feature vectors are presented to the SOM. For each input vector the algorithm compares the distance to the weights of each neuron n_i of the map. The neuron n^* with the lowest distance d^* to the weight vector $\mathbf{w}^*$ is the winner neuron

$$d^* = \min_{1 \le k \le K} \{\|\mathbf{w}_j - \mathbf{x}\|^2\}. \tag{3}$$

The weights of the winner neuron n^* and its neighbors are pulled into the direction of the data sample $\mathbf{x}$ depending on the learning rate η and the neighborhood function $h(r)$. The neighborhood function h shall have the following properties:

- h is maximal at the center of neuron n^* and
- outside the range of r, i.e., for $d > r$, it holds $h = 0$.

Parameter r is also called neighborhood radius. A typical neighborhood function that fulfills the constraint is the Gaussian function.

In each generation the SOM updates the weights of the winner and its neighborhood with the help of η and h, so that they are pulled into the direction of x

$$\mathbf{w}_i' = \mathbf{w}_i + \eta \cdot h(n^*, n_i, r) \cdot (\mathbf{x} - \mathbf{w}_i). \tag{4}$$

The algorithms lead to a mapping from the feature space to the neural map. The mapping maintains the topology of the neighborhood: Close data samples in the high-dimensional space lie closely together on the map, too. Usually the radius r is a function of parameter σ, which is decreased in the course of the algorithm. To ensure convergence of the algorithm we have to reduce η as well, e.g., with a decreasing function:

$$\sigma(t) = \sigma_s \left(\frac{\sigma_e}{\sigma_s}\right)^{t/t_e}, \tag{5}$$

with σ_s being the staring value and the target value σ_e that belongs to its function value t_e. SOMs have been applied to a broad field of applications, e.g., real-time monitoring of product properties from spectrophotoscopic measurements [29], and in missing value imputation [30]. SOM-based visualization has been applied to various applications ranging from financial time-series data [31] to GSM network performance data [32]. We have employed a related approach for the visualization of music sequences [33]and employed the SOM-based discretization for recognition of sequences with a dynamic time warping approach for gesture recognition [34]. Using SOMs for monitoring and modeling of complex industrial processes like pulp process, steel production, and paper industry are described by Alhoniemi et al. [35].

```
1   Start
2     Initialize weight vectors w_i of each neuron n_i;
3     Repeat
4       Randomly select data sample x from training set T;
5       Compare x to each weight vector w_i of the SOM;
6       The winner neuron n* with minimal distance
        d* = min_{1≤j≤k} {d(x, w_j)};
7       Update of weight vectors
        w_i' = w_i + η · h(n*, n_i, r) · (x − w_i);
8       Decrease learning rate η, and radius r;
9     Until termination condition
10  End
```

Fig. 4. Pseudocode of the SOM algorithm.

4.2. Visualization of multivariate wind time-series

Task of the SOM is to compute a smooth mapping of the high-dimensional time-series data to the neurons of the map. By smoothness we denote that the quantization maintains the topology of the data: neighbored high-dimensional data is also neighbored on the low-dimensional map. This characteristic is important for the state monitoring, as smoothly changing colors alleviate the recognition of state changes. We employ a squared two-dimensional SOM, whose size, and number of training cycles are stated in each case.

With a trained SOM the sequence of high-dimensional data can be translated to a corresponding sequence of symbols (winner neurons). Let $\mathbf{x}_1,\ldots,\mathbf{x}_N$ be a sequence of high-dimensional feature vectors. Fed to a trained SOM the smooth quantization is generated by computing a sequence of symbols $s_1,\ldots,s_N$ with

$$s_i = \arg \min_{k=1\ldots K} \|\mathbf{w}_k - \mathbf{x}_i\|^2. \tag{6}$$

The symbols of the sequence are colorized with regard to the corresponding winner neuron n^*, whose color has to be user-defined. According to the topological characteristics of the SOM neighbored neurons are assigned to similar colors in the sequence. A function $f : \mathcal{S} \to \mathbb{R}^3$, $s \to (r_1, r_2, \lambda)$ maps a symbol s to a three-dimensional RGB-vector. Values r_1 and r_2 depend on neuron n with corresponding coordinates on the map, while for the third value λ an arbitrary constant can be chosen. With function f a sequence of symbols corresponding to a sequence of winner neurons of the SOM can be translated into a sequence of RGB-vectors.

We have developed an efficient implementation of the SOM-based quantization and a visualization method in C. The developed software tool makes use of an XML-file for specification of the SOM training parameters. The features of the training set have to be stored in a simple CSV-format.

4.3. Micro-level: wind park Tehachapi

In the first data scenario we use the wind resource data set from 20 grid points in the area of Tehachapi, California, in 2006. The points have been randomly chosen, and are aggregated to 20-dimensional vectors. For a better readability we concentrate on the visualization of a period of approximately four days, i.e., 1000 measurements. Large-scale experiments can be found in Section 4.4. The upper two sequences of Fig. 5 show the visualization of 20-dimensional grid points of Tehachapi. The figures compare the visualization based on a 2×2-SOM with 100

Please cite this article as: O. Kramer, et al., Wind energy prediction and monitoring with neural computation, Neurocomputing (2012), http://dx.doi.org/10.1016/j.neucom.2012.07.029

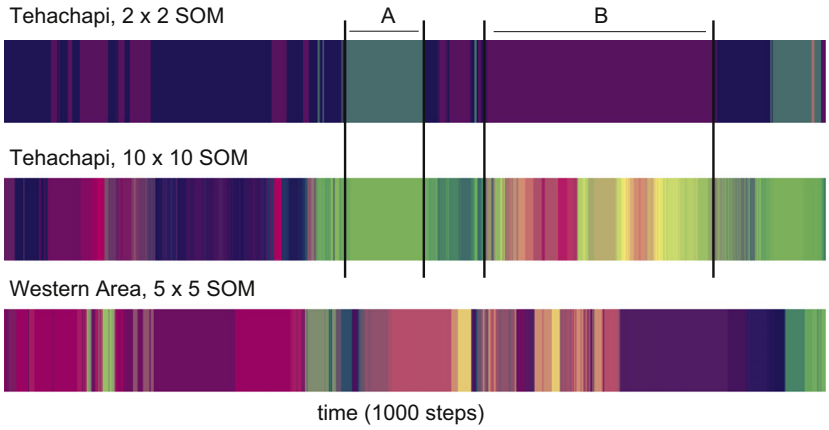

Fig. 5. Upper two sequences: Visualization of times series of 20 points in the area of Tehachapi. The sequences are 20-dimensional time-series of 1000 steps, starting from January 2006. The results are generated with a 2×2-SOM with 100 learning cycles and a 10×10-SOM with 500 learning cycles. Lower part: Visualization of times series of 20 randomly chosen points in the whole area of the western wind data set, i.e., Nampa, Salt Lake City, Casper, Fort Collins, Colorado Springs, Amarillo, Carlsbad in Mexico, Las Vegas, Palm Springs, and Tehachapi using a 5×5-SOM. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

learning cycles to the output of a 10×10-SOM with 500 learning cycles. The SOM-training process leads to structurally similar topological mappings. The color assignment may vary in different runs, as the training is a non-deterministic process.

It can be observed that both sequence visualizations share similar color changes, while the larger SOM allows a finer granularity of states. The green part A left from the middle of both sequences is a period, when the wind is almost not blowing. The corresponding segments of both sequences are marked with a similar color. In contrast, part B shows a segment that is marked with the same color in the 2×2-SOM, but varying colors in the case of the 10×10-SOM. The latter is able to visualize differences with a higher granularity in the same period. It is important to remark that for interpretation purposes the SOM allows a backward mapping by showing the neural weights, corresponding to the multi-dimensional vectors that represent the system states. Furthermore, it is possible to train a SOM with specific system states to visualize typical, and important developments. In our experiments topological defects that would motivate the application of cylindric SOM-topologies or the neural gas heuristic [36] have not been observed.

4.4. Large-scale analysis: whole western area

For the second scenario we have selected 20 grid points in different areas of the western part of USA. We randomly chose two points from each of the following locations: Nampa, Salt Lake City, Casper, Fort Collins, Colorado Springs, Amarillo, Carlsbad in New Mexico, Las Vegas, Palm Springs, and Tehachapi. Fig. 5 shows the corresponding visualization. Similar colors correspond to similar system states. In the middle of the time-series the wind is changing quite often. For interpretation of the colors, the corresponding data has to be analyzed. For example, the color green means that in Palm Springs and Tehachapi the wind is weak, and strong in the other parts of the western area. This state can be observed in the first quarter and at the end of the period. The assignment to reference states and their interpretation *a posteriori* are easily possible with the help of the weight vectors of the neurons of interest.

We demonstrate the scalability of our tool with a large-sale experiment with 100 grid points of the whole western area for one year. For this sake we have trained SOMs of various sizes on a state-of-the-art consumer processor. For the tests we employed an AMD Athlon64 X2 3800+, and an Intel i5 M450 with 2.4 GHz, both with 4 GB memory running Windows 7. The large-scale training data sets consists of 100 wind grid points randomly chosen from the western area. The final data set consists of 52,560 measurements—one measurement every 10 min for one year.

Table 4
Comparison of runtime on two CPUs for training with various SOM dimensions.

SOM	AMD Athlon64	Intel i5 M450
3×3	5:10 min	3:30 min
5×5	13:50 min	9:15 min
10×10	54:26 min	36:29 min
25×25	6:30 h	3:46 h

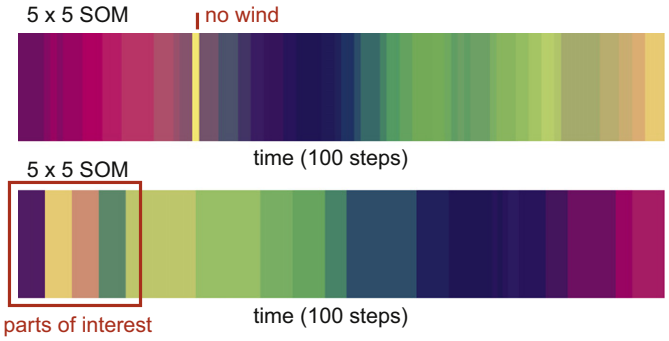

Fig. 6. Upper sequence: Simulated blackout of wind resources in Tehachapi. The alert state at the 28-th time step can clearly be recognized with a similar color like the low-wind level on at the end of the period. Lower sequence: Training of the SOM with states of interest and corresponding visualization of a 100 step times series. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4 compares the runtime of 100 training cycles on the two CPUs. The average memory usage we observed during training of the SOM was around 40–50 MB, and about 100 MB during computation of the visualization. The figures show that the tool can visualize large and high-dimensional data sets on usual CPUs on the day, week, or month basis. Training of large maps with huge data sets can be a time-consuming undertaking.

4.5. Identification of reference motifs

The upper part of Fig. 6 shows an alert state that has artificially been integrated to the data of January 2006. The 28-th data element of the wind resource has been set to 0.0 for all grid points,

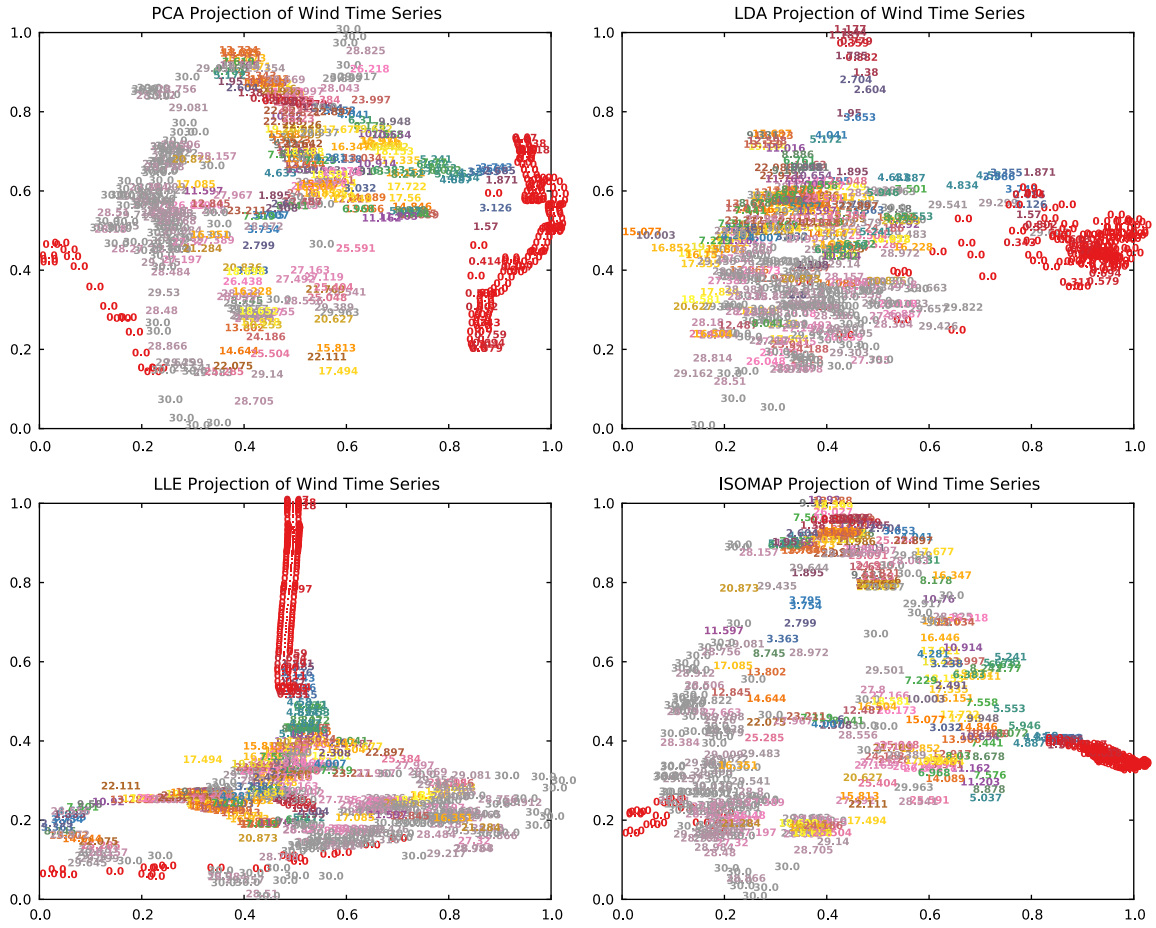

Fig. 7. Dimensionality reduction of 500 wind time-series patterns that are 30-dimensional. The structure of the space of wind energy patterns is visualized. Trajectories reveal the pattern structure. In comparison to the SOM-approach the temporal information is lost. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

simulating a sudden blackout of all wind turbines. This alert state can clearly be recognized, it has a suspicious bright color like the area on the right hand side of the plot that represents times of low wind energy production. A training with representative alert states that have to be selected by the grid operator is possible to identify specific situations of the system. The lower part of Fig. 6 shows the visualization of a sequence that contains training motifs – sequences of special interest – at the beginning. These four segments have artificially been generated, and represent (1) a state where no wind blows in the whole system, (2) a state where the wind blows with full speed in the northern half of the park, and not in the southern part, (3) the inverse state of (2), and (4) the situation that the wind blows everywhere with full speed. The subsequent parts of the sequence show where the corresponding situation can be found in the remainder of the sample sequences. Again, similar colors represent states that are similar to the reference motifs. The other colors (like the dark blue sections) do not have much in common with the reference motifs. The marked states can easily be identified, e.g., as alert states.

4.6. Comparison of wind time-series embeddings

In the case of SOMs the patterns are mapped to discrete neurons, which are assigned to colors. There are further ways to visualize high-dimensional wind time-series. In this section we show how the multivariate wind time-series can be mapped to a two-dimensional latent space employing principal component analysis (PCA) [37], linear discriminant analysis (LDA) [38], locally linear embedding (LLE) [39], and ISOMAP [40]. These methods are very popular dimensionality reduction algorithms, and have found a broad range of applications, e.g., PCA neural models have recently been applied for the blind separation of signals [41].

Fig. 7 shows the embeddings of 500 patterns of 30-dimensional wind time-series steps of January 2005 of the Tehachapi wind park employing the four methods. The plotted values are the corresponding wind energies for a target wind spot in the center (energies again reach from 0.0 MW to 30.0 MW), which also determine the colors (from red for 0.0 MW to gray for 30.0 MW). Interesting structures are visualized: the development from low wind energy to a higher level (successive red patterns) can clearly be identified as trajectory in the space of wind energy patterns in case of PCA, LLE, and also ISOMAP. Strong wind is clearly separated from the other situations. The PCA, LLE, and ISOMAP embeddings have one further thing in common: holes can be identified that show that the space of wind energy situations employs sparse parts. Such a continuous approach is not possible with SOM-approaches, as the embeddings are mapped to the network structure, which is usually organized as a gird. If it is necessary to visualize the space of time-series patterns, then the continuous approach is recommended. If the development w.r.t. a linear time axis is of interest, then we recommend the SOM-approach.

4.7. Conclusion

For grid operators the understanding of the energy system status is very important. Wind is a very volatile resource. Tools

Please cite this article as: O. Kramer, et al., Wind energy prediction and monitoring with neural computation, Neurocomputing (2012), http://dx.doi.org/10.1016/j.neucom.2012.07.029

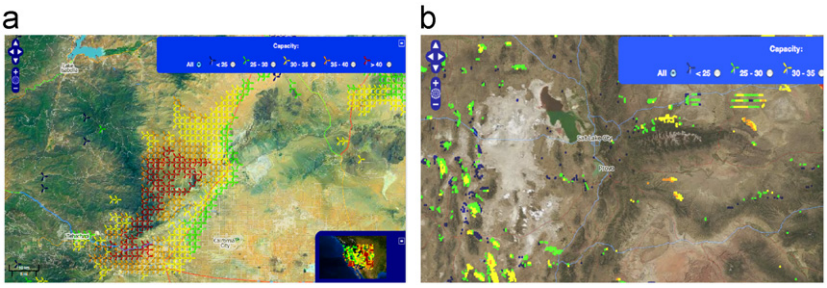

Fig. A1. NREL wind map: screenshot of a section of the Tehachapi wind park in California (left) and of the wind park near Salt Lake City (right).

are required that allow to represent and visualize high-dimensional time-series data in an intuitive kind of way. In this section we have demonstrated how unsupervised neural learning methods can contribute to fulfill this goal. Self-organizing maps can be used as efficient tools that allow the smooth discretization of high-dimensional time-series for visualization and monitoring, maintaining topological characteristics like neighborhood relations, and distances in data space. Training with specific system state sequences that are important for the grid operator is a straightforward undertaking with the introduced approach. For this sake a set of representative training data sets has to be identified and manually generated. The continuous embedding with related dimensionality reduction algorithms turns out to be an interesting approach for the analysis of the overall space of wind energy patterns, not taking into account the temporal information.

5. Summary

Neural computation comprises methods for solving machine learning and optimization tasks. They have proven well in analysis and modeling dynamic systems. This seems natural as neural techniques are motivated by mechanisms of natural neural systems that have to cope with dynamic environments. Wind is a very dynamic system. An important aspect is to manage its volatile, and dynamic nature. The integration of wind energy into smart grids affords balancing capabilities, and balancing affords understanding, and forecasting.

The examples presented in our paper have shown how neural techniques can help to cope with the dynamics of wind time-series data. They turn out to be successful methods in forecasting, and monitoring of wind energy time-series data. Efficient implementations allow to apply them in real-time scenarios thanks to the LIBSVM library [25] and our SOM-visualization tool. It is subject to future projects to show the success of these, and other neural methods in real-world energy applications.

Appendix A. Wind time-series data

In the following, we give a brief overview of the wind data sets employed in the experimental parts.

(1) Tehachapi is a wind park in California, USA, see left part of Fig. A1. The Tehachapi data set consists of approximately 200 windmills. We use particular grid points in Tehachapi. For the sequence monitoring approach in Section 4 we used a data set consisting of 20 randomly chosen grid points from the whole wind farm.

(2) Salt Lake City, see right part of Fig. A1. The small wind park southwest of Salt Lake City consists of 28 wind grid points. It is subject to a large-scale analysis of wind resource prediction in Section 3.2.

(3) Western area: The western area data set consists of 100 grid points that have randomly been selected from the whole NREL data set. In particular, the western area data set is used as indicator for large-scale wind resource prediction on the park level, as the broadly distributed grid points work as indicators for wind developments for Tehachapi, and Salt Lake City in Section 3.

References

[1] O. Kramer, F. Gieseke, Short-term wind energy forecasting using support vector regression, in: International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO), 2011, pp. 271–280.
[2] O. Kramer, F. Gieseke, Analysis of wind energy time series with kernel methods and neural networks, in: The Seventh International Conference on Natural Computation (ICNC), Shanghai, 2011, pp. 2381–2385.
[3] O. Kramer, T. Hein, Monitoring of multivariate wind resources with self-organizing maps and slow feature analysis, in: IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG), IEEE Press, Paris, 2011.
[4] D. Lew, M. Milligan, G. Jordan, L. Freeman, N. Miller, K. Clark, R. Piwko, How do wind and solar power affect grid operations: the western wind and solar integration study, in: The Eighth International Workshop on Large Scale Integration of Wind Power and on Transmission Networks for Offshore Wind Farms, 2009.
[5] C.W. Potter, D. Lew, J. McCaa, S. Cheng, S. Eichelberger, E. Grimit, Creating the dataset for the western wind and solar integration study (USA), in: The Seventh International Workshop on Large Scale Integration of Wind Power and on Transmission Networks for Offshore Wind Farms, 2008.
[6] E. Parzen, On the estimation of a probability density function and mode, Ann. Math. Stat. 33 (1962) 1065–1076.
[7] B.W. Silverman, Density Estimation for Statistics and Data Analysis, Monographs on Statistics and Applied Probability, vol. 26, Chapman and Hall, London, 1986.
[8] M. Milligan, K. Porter, E. DeMeo, P. Denholm, H. Holttinen, B. Kirby, N. Mille, A. Mills, M. OMalley, M. Schuerger, L. Soder, 〈http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5233741&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5233741〉 (February).
[9] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[10] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.
[11] B. Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, USA, 2001.
[12] A. Costa, A. Crespo, J. Navarro, G. Lizcano, H. Madsen, E. Feitosa, A review on the young history of the wind power short-term prediction, Renew. Sustain. Energy Rev. 12 (6) (2008) 1725–1744.
[13] M. Negnevitsky, P. Mandal, A. Srivastava, Machine learning applications for load, price and wind power prediction in power systems, in: Intelligent System Applications to Power Systems (ISAP), 2009, pp. 1–6.
[14] P.G. Shuhui, S. Li, D.C. Wunsch, E.A. Ohair, M.G. Giesselmann, Using neural networks to estimate wind turbine, J. Guid. Control Dyn. 16 (3) (2001) 276–282.
[15] G. Li, J. Shi, J. Zhou, Bayesian adaptive combination of short-term wind speed forecasts from neural network models, Renew. Energy 36 (1) (2011) 352–359.
[16] M. Mohandes, T. Halawani, S. Rehman, A.A. Hussain, Support vector machines for wind speed prediction, Renew. Energy 29 (6) (2004) 939–947.
[17] J. Shi, Y. Yang, P. Wang, Y. Liu, S. Han, Genetic algorithm-piecewise support vector machine model for short term wind power prediction, in: Proceedings of the Eighth World Congress on Intelligent Control and Automation, 2010, pp. 2254–2258.
[18] P. Zhao, J. Xia, Y. Dai, J. He, Wind speed prediction using support vector regression, in: Industrial Electronics and Applications (ICIEA), 2010, pp. 882–886.
[19] A. Kusiak, W. Li, The prediction and diagnosis of wind turbine faults, Renew. Energy 36 (1) (2011) 16–23.

Please cite this article as: O. Kramer, et al., Wind energy prediction and monitoring with neural computation, Neurocomputing (2012), http://dx.doi.org/10.1016/j.neucom.2012.07.029

[20] B. Schölkopf, R. Herbrich, A.J. Smola, A generalized representer theorem, in: D.P. Helmbold, B. Williamson (Eds.), Proceedings of the 14th Annual Conference on Computational Learning Theory, 2001, pp. 416–426.
[21] I. Steinwart, A. Christmann, Support Vector Machines, Springer-Verlag New York, NY, USA, 2008.
[22] A.J. Smola, B. Schölkopf, B. Scholkopf, A Tutorial on Support Vector Regression, 1998.
[23] X. Yang, S. Chen, B. Chen, Z. Pan, Proximal support vector machine using local information, Neurocomputing 73 (1–3) (2009) 357–365.
[24] D. Wang, J. Zheng, Y. Zhou, J. Li, A scalable support vector machine for distributed classification in ad hoc sensor networks, Neurocomputing 74 (1–3) (2010) 394–400.
[25] C.-C. Chang, C.-J. Lin, LIBSVM: A Library for Support Vector Machines, Software Available At 〈http://www.csie.ntu.edu.tw/cjlin/libsvm〉, 2001.
[26] P.F. Evangelista, M.J. Embrechts, B.K. Szymanski, Taming the curse of dimensionality in kernels and novelty detection, in: Ajith Abraham, Bernard de Baets, Mario Kppen, Bertam Nickolay (Eds.), Applied Soft Computing Technologies: The Challenge of Complexity, Springer-Verlag, 2006, pp. 431–444.
[27] T. Kohonen, Self-Organizing Maps, Springer, 2001.
[28] H. Ritter, T. Martinetz, K. Schulten, Neural Computation and Self-Organizing Maps: An Introduction, Addison Wesley, 1992.
[29] F. Corona, E. Liitiäinen, A. Lendasse, L. Sassu, S. Melis, R. Baratti, A SOM-based approach to estimating product properties from spectroscopic measurements, Neurocomputing 73 (1–3) (2009) 71–79.
[30] P. Merlin, A. Sorjamaa, B. Maillet, A. Lendasse, X-SOM and l-SOM: a double classification approach for missing value imputation, Neurocomputing 73 (7–9) (2010) 1103–1108.
[31] D. Yu, Y. Qi, Y.-H. Xu, J.-Y. Yang, Kernel-SOM based visualization of financial time series forecasting, in: ICICIC, vol. 2, 2006, pp. 470–473.
[32] P. Lehtimaeki, K. Raivio, A SOM based approach for visualization of gsm network performance data, in: Proceedings of the 18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 2005, pp. 588–598.
[33] T. Hein, O. Kramer, Recognition and visualization of music sequences using self-organizing feature maps, in: KI, 2010, pp. 160–167.
[34] O. Kramer, B. Satzger, J. Lässig, Power prediction in smart grids with evolutionary local kernel regression, in: HAIS, vol. 1, 2010, pp. 262–269.
[35] E. Alhoniemi, J. Hollmen, O. Simula, J. Vesanto, Process monitoring and modeling using the self-organizing map, Integrated Comput. Aided Eng. 6 (1999) 3–14.
[36] T. Martinetz, K. Schulten, A neural gas network learns topologies, in: Artificial Neural Networks, Elsevier, 1991, pp. 397–402.
[37] I. Jolliffe, Principal Component Analysis, Springer Series in Statistics, Springer, New York, USA, 1986.
[38] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer, Berlin, 2009.
[39] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323–2326.
[40] J.B. Tenenbaum, V.D. Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.
[41] K.I. Diamantaras, T. Papadimitriou, Applying pca neural models for the blind separation of signals, Neurocomputing 73 (1–3) (2009) 3–9.

Dr. Oliver Kramer is junior professor for computer science at the Department of Computing Science, University of Oldenburg, Germany. He received his diploma in computer science from the University of Dortmund in 2003, and his PhD from the University of Paderborn in 2008. His research interests include optimization, machine learning, and applications in energy systems.

Dr. Benjamin Satzger is an assistant professor of computer science in the Distributed Systems Group at Vienna University of Technology. He received his Master degree and Doctorate in Computer Science in 2005 and 2008, respectively, both from Augsburg University, Germany. From 2009 to 2010 he was a Research Fellow at the International Computer Science Institute, Berkeley. His research interests include management of complex systems and data stream processing.

Dr. Fabian Gieseke received his Diploma degrees in mathematics and computer science from the University of Münster, Germany, and his PhD in computer science from the Carl von Ossietzky University Oldenburg, Germany. He is currently working as a post-doctoral researcher at the University of Oldenburg. His research interests include support vector machines and their extensions to semi- and unsupervised learning settings, and applications in astronomy and energy systems.

Please cite this article as: O. Kramer, et al., Wind energy prediction and monitoring with neural computation, Neurocomputing (2012), http://dx.doi.org/10.1016/j.neucom.2012.07.029