Renewable and Sustainable Energy Reviews 98 (2018) 189–198

Contents lists available at ScienceDirect

Renewable and Sustainable Energy Reviews

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

Deep Learning for fault detection in wind turbines

Georg Helbing*, Matthias Ritter

Department of Agricultural Economics, Humboldt Universität zu Berlin, Philippstr. 13, Haus 12, 1. OG, Raum 26, D-10115 Berlin, Germany

ARTICLE INFO

Keywords:
Deep Learning
Artificial neural network
Fault detection
Condition monitoring
Wind turbine

ABSTRACT

Condition monitoring in wind turbines aims at detecting incipient faults at an early stage to improve maintenance. Artificial neural networks are a tool from machine learning that is frequently used for this purpose. Deep Learning is a machine learning paradigm based on deep neural networks that has shown great success at various applications over recent years. In this paper, we review unsupervised and supervised applications of artificial neural networks and in particular of Deep Learning to condition monitoring in wind turbines. We find that – despite a promising performance of supervised methods – unsupervised approaches are prevalent in the literature. To explain this phenomenon, we discuss a range of issues related to obtaining labelled data sets for supervised training, namely quality and access as well as labelling and class imbalance of operational data. Furthermore, we find that the application of Deep Learning to SCADA data is impeded by their relatively low dimensionality, and we suggest ways of working with higher-dimensional SCADA data.

1. Introduction

As wind turbines are becoming larger and more complex, detecting technical faults in wind turbines that require unscheduled maintenance is increasingly important [1,2]. [1] point out that major faults, which cause more than one day of downtime, represent only 25% of all failures, but account for 95% of the downtime incurred by wind turbines. It is therefore in the interest of wind farm operators to predict those major errors in advance, to prevent critical damage and to optimise maintenance schedules. In recent years, the detection of malfunctioning in wind turbines has been subject to intense research by the scientific community. In numerous studies, fault detection has been examined at system level [3,4] as well as for various components such as rotor blades [5–7] and gearbox [8–10].

Most approaches can be subdivided into model-based (using a numerical model of the wind turbine or its subcomponents), signal processing (based on vibration data measured by a condition monitoring system) and data-driven approaches (based on supervisory control and data acquisition (SCADA) data) [11]. A thoroughly explored data-driven approach is to train a normal behaviour model (NBM) that predicts the value of some state variable of the wind turbine, e.g. active power or temperatures, as a function of environmental variables and other state variables. The predicted value is then compared online to the actually measured value, yielding a residual. Large values or trends of the residual time series may indicate incipient failure of the turbine. The NBM approach is closely related to statistical process control (SPC),[1] and indeed the residual time series are frequently transformed into a control chart such as the exponentially-weighted moving average (EWMA) chart [3,5,13].

For the estimation of NBM, many authors have used so-termed artificial neural networks (ANNs) e.g. [14–16]. These are a family of methods from machine learning that are capable of detecting highly non-linear relationships in data. When trained in a supervised manner, they can be used for classification and for regression tasks. The most widely applied ANN of this type is the multi-layer perceptron (MLP). On the other hand, unsupervised ANN architectures allow to smooth out noise and to learn compressed representations of the data. The term Deep Learning refers to ANNs that are particularly complex and which have been applied to the fields of image processing [17], natural language processing [18] and in particular to machine health monitoring

Abbreviations: ANN, Artificial Neural Network; API, Application Programming Interface; AUC, Area Under the Curve; CNN, Convolutional Neural Network; EWMA, Exponentially-Weighted Moving Average; GPU, Graphical Processing Unit; GRU, Gated Recurrent Unit; LSTM, Long-Short Term Memory; MD, Mahalanobis Distance; MLP, Multi-Layer Perceptron; NBM, Normal Behaviour Model; PCA, Principal Component Analysis; RBM, Restricted Boltzmann Machine; RNN, Recurrent Neural Network; ROC, Receiver Operating Characteristics; SAE, Stacked Autoencoder; SCADA, Supervisory Control and Data Acquisition; SDAE, Stacked Denoising Autoencoder; SMLDAE, Stacked Multilevel-Denoising Autoencoder; SPC, Statistical Process Control

* Corresponding author.
E-mail addresses: georg.helbing@agrar.hu-berlin.de (G. Helbing), matthias.ritter@agrar.hu-berlin.de (M. Ritter).

[1] For an introduction to SPC, see for instance Montgomery [12].

https://doi.org/10.1016/j.rser.2018.09.012
Received 29 January 2018; Received in revised form 4 September 2018; Accepted 7 September 2018
1364-0321/

[19] with noteworthy success.[2] There have also been some first applications to fault detection in wind turbines [5,8,21], which use unsupervised ANN such as stacked autoencoders (SAE) and stacked restricted Boltzmann machines (RBM) to detect outliers that indicate an incipient failure in SCADA and vibration data.

In this paper, we review the more recent literature on fault detection in wind turbines for applications of classical ANNs and Deep Learning. It is our goal to assess to what degree Deep Learning is applied and to identify further potentials. Several reviews exist that summarise the various model-based, signal-processing and data-driven approaches applied in this field. [22] review papers that assess the performance of wind turbine gearboxes. Their review, however, considers ANNs only as one method amongst others and does not focus on ANNs. The same can be said for [23], who provide a general overview of prediction, operation and condition monitoring in wind turbines. The papers they review use a broad variety of data-mining and further methods. [24] provide a review on papers studying fault detection based on SCADA data. Their focus is on SCADA data rather than any particular method used for their evaluation. On the contrary, [25] specifically review applications of ANNs in wind energy systems. With most papers being dedicated to power prediction, though, they do not focus on fault detection. None of the reviews mentioned here put a focus on Deep Learning.

The rest of this paper is structured as follows. In the next section, we provide a concise introduction to ANNs and Deep Learning, also touching on some common programming frameworks for implementing these models. We continue by describing NBM and control charts, which are often used in this field (Section 3). In the fourth section, we review a selection of studies that apply ANNs and Deep Learning to fault detection in wind turbines. Finally, we discuss the reviewed articles with regard to perspectives for applying Deep Learning in this field.

2. Artificial neural networks and Deep Learning

ANNs are quite a seasoned tool, dating back as far as the 1950s [26]. They come in a great range of different flavours, but all have in common that they are machine learning algorithms that can be used for regression and classification (supervised learning) and/or for representation learning (unsupervised) tasks. The most common types of ANNs for supervised learning are the MLP, the convolutional neural network (CNN) and the recurrent neural network (RNN). The most common types of ANNs for unsupervised learning are the (stacked) RBM and the (stacked) autoencoder. The literature also contains various combinations and sub-types of these. For an overview see for instance [20,27–30].

2.1. Multi-layer perceptron

An MLP is a fully-connected feed-forward neural network consisting of the input layer, one or more hidden layers and the output layer.[3] Each layer consists of several nodes, which are computational units, and the nodes of one layer are connected to all nodes of the subsequent layer via links that are weighted with real numbers as in a directed weighted graph, as illustrated in Fig. 1. The numeric input data, such as an observation of SCADA variables, are fed into the MLP, where each input variable is represented by one node of the input layer. Via the weighted links, the values ‘flow’ to the nodes of the next layer, where in each node the weighted sum of all inputs plus a constant bias is computed. This can be expressed as

$$f_j(x_1, \ldots, \quad x_n; w_{1j}, \ldots, \quad w_{nj}, b_j) = \sum_{i=1}^{n} w_{ij} x_i + b_j \tag{1}$$

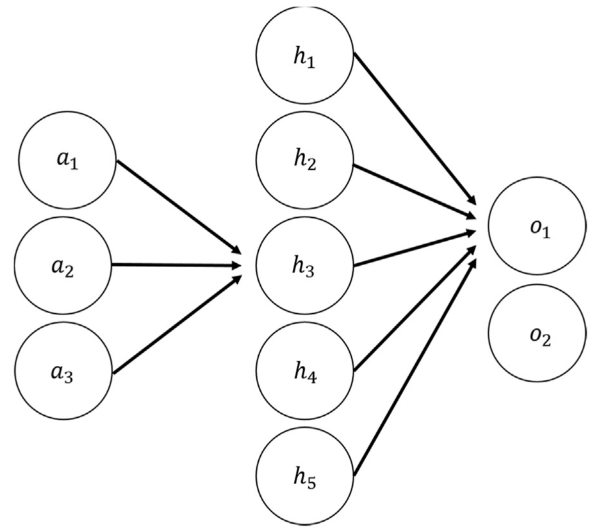

Fig. 1. Schematic of a 3-5-2 MLP. Only links to h_3 and o_1 are displayed. Data flow from the input nodes a_i via weighted links to each node of the hidden layer. In each node h_j the weighted sum of the incoming links is computed and transformed via a non-linear activation function, whose output is again forwarded to the output layer via weighted links. In the output nodes o_k, the weighted sum of all incoming links is computed and transformed via an activation function, whose output represents the output of the MLP.

where f_j is the output of node j, x_i are its inputs, w_{ij} the weights of the links connecting it to the previous layer and b_j is the bias. The results of these affine transformations then pass through a non-linear activation function, which is typically the sigmoid function, the hyperbolic tangent or the so-called rectified linear unit. For instance, the frequently used sigmoid function yields the transformed output of node j as

$$g(f_j) = \frac{e^{f_j}}{1 + e^{f_j}} \tag{2}$$

After this transformation, the values are forwarded to all nodes of the subsequent layer via weighted links and so the process continues, until the transformed data reach the output layer and pass through the last activation function. These values are the output of the network. In summary, an MLP is a non-linear function $f:\mathbb{R}^n \to \mathbb{R}^m$, where n is the dimension of the input data and m the dimension of the output data. Training an MLP means adjusting its weights and biases so that the MLP's output over a training set approximates the true values (the so-called labels) as well as possible. Typically, the squared error ρ is used as a measure of prediction error. Given the label $o(x_1, \ldots, x_n) \in \mathbb{R}^m$, the squared error ρ of the output $\tilde{o}(x_1, \ldots, x_n) \in \mathbb{R}^m$ is computed as

$$\rho(x_1, \ldots, \quad x_n) = \frac{1}{2} \sum_{i=1}^{m} (o_i - \tilde{o}_i)^2 \tag{3}$$

where the factor of $\frac{1}{2}$ is used to facilitate the derivation of certain properties. The backpropagation algorithm [32] is used to compute the gradient of the squared error with respect to the weights and biases. These gradients can then be used by an optimization algorithm – e.g. stochastic gradient descent – to adjust the weights. A schematic of training an MLP is displayed in Fig. 2.

Often, the weights of the MLP are initialised with random values at the beginning of the training and then iteratively optimized. It has been found, however, that this procedure leads to increasingly poor results as the MLP becomes deeper (i.e. as it has more layers), because it is the nature of the backpropagation algorithm that gradients tend to become smaller the farther their layer is from the output layer, which is referred to as the problem of vanishing gradients [27]. This difficulty in training MLP with more than a few layers might explain why many applications of MLP use only one hidden layer. However, deeper MLP carry out more non-linear transformations on the data, so their capacity of

[2] See LeCun et al. [20] for an introduction to Deep Learning.
[3] For a general introduction to ANN and MLP in particular, see Kriesel [31].