Expert Systems with Applications 40 (2013) 6033–6040

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

Predicting time series of railway speed restrictions with time-dependent machine learning techniques

Olga Fink [a,*], Enrico Zio [b,c], Ulrich Weidmann [a]

[a] Institute for Transport Planning and Systems, ETH Zurich, Zurich, Switzerland
[b] Chair on Systems Science and the Energetic Challenge, European Foundation for New Energy-Electricité de France (EDF) at École Centrale Paris and SUPELEC, France
[c] Department of Energy, Politecnico di Milano, Italy

ARTICLE INFO

Keywords:
Neural networks
Echo state networks
Conditional restricted Boltzmann machines
Railway operations disruptions
Tilting system
Speed reductions
Discrete-event diagnostic data
Binary time series predictions

ABSTRACT

In this paper, a hybrid approach to combine conditional restricted Boltzmann machines (CRBM) and echo state networks (ESN) for binary time series prediction is proposed. Both methods have demonstrated their ability to extract complex dynamic patterns from time-dependent data in several applications and benchmark studies. To the authors' knowledge, it is the first time that the proposed combination of algorithms is applied for reliability prediction.

The proposed approach is verified on a case study predicting the occurrence of railway operation disruptions based on discrete-event data, which is represented by a binary time series. The case study concerns speed restrictions affecting railway operations, caused by failures of tilting systems of railway vehicles. The overall prediction accuracy of the algorithm is 99.93%; the prediction accuracy for occurrence of speed restrictions within the foresight period is 98% (which corresponds to the sensitivity of the algorithm). The prediction results of the case study are compared to the prediction with a MLP trained with a Newton conjugate gradient algorithm. The proposed approach proves to be superior to MLP.

1. Introduction

European railway networks have increasingly become more complex and interconnected due to increased service frequency of passenger transportation and increased rail freight demand. The increased frequency resulted partly in decreased buffering capacities for preventing failure propagation over the networks. Therefore, railway networks have become more vulnerable to failures that could cause severe operational disruptions (Dorbritz, 2012). As one of the countermeasures, both railway infrastructure systems and rolling stock systems are being increasingly equipped with monitoring and diagnostic systems. These systems allow operators to identify locations of failures in shorter times and thereby provide higher availability of the systems.

Data measured by remote systems give an insight into the processes occurring in the specific components or subsystems, as opposed to considerations which are valid for average systems under average operating conditions. Monitored data can be used to detect faults or failures, to diagnose them and to predict the considered systems remaining useful life (Vachtsevanos, 2006). In order to use data effectively, at least one parameter (which can be measured and processed cost-efficiently) must exist that directly describes the systems condition, or the system condition must be able to be derived indirectly from measurable parameters.

In addition to the continuously measured diagnostic data, there are also automatically recorded events which occur at discrete random times. The recorded discrete events can range from confirming an enabled function to indicating that a signal exceeds defined limits. Some of these events are used to warn the train driver or to assist the maintenance crew in their fault finding and corrective actions. Discrete-event diagnostic data contains less information than continuously measured parameters. However, they provide complementary information that cannot only be used for a posteriori analyses, but also for predictions. The sequence of occurring discrete events can be represented by a binary time series.

Different types of models for binary time series regression have been proposed in the literature and used in different fields of application, including reliability prediction (Kedem & Fokianos, 2002). The regression models include state-space approaches, such as hidden Markov models (MacDonald & Zucchini, 1997), generalized linear models (Green & Silverman, 1994; McCullagh & Nelder, 1989) and logistic regression models (Liang & Zeger, 1989). Machine learning techniques, such as Support Vector Machines (SVM) and different types of neural networks, are also increasingly applied for time series regressions (Yadav, Mishra, Yadav, Ray, &

* Corresponding author. Address: ETH Zurich, Wolfgang-Pauli-Str. 15, 8093 Zurich, Switzerland. Tel.: +41 44 633 27 28.
E-mail address: ofink@ethz.ch (O. Fink).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.04.038

Kalra, 2007; Zainuddin & Pauline, 2011; Zhang & Wan, 2007), also in the field of reliability prediction where they have proved to be powerful tools (Chatterjee & Bandopadhyay, 2012; Pai, 2006; Xu, Xie, Tang, & Ho, 2003). However, the most commonly used neural network model, multilayer perceptrons (MLP), generally, suffer from drawbacks which include obtaining sub-optimal solutions due to local minima and requiring long computation time.

In this paper, we propose a combination of conditional restricted Boltzmann machines (CRBM) and echo state networks (ESN) for binary time series prediction. Both methods have demonstrated their ability to extract complex dynamic patterns from time-dependent data in several applications (Lin, Yang, & Song, 2009; Salakhutdinov, Mnih, & Hinton, 2007) and benchmark studies (Ferreira, Ludermir, & de Aquino, 2013; Jaeger & Haas, 2004). The principal component analysis (PCA) is applied complementarily in this approach, to reduce the dimensionality and extract the features. To the authors' knowledge, it is the first time that the proposed combination of algorithms is applied for reliability prediction.

The proposed approach is applied to a case study concerning the prediction of the occurrence of railway operations disruptions caused by failures or malfunctions of the tilting system. In this work, diagnostic event data of a railway vehicle fleet are used to forestall the future behavior of speed restrictions affecting railway operations based on the evolution of the diagnostic event sequence in time. The prediction precision of the prosed combination of algorithms is compared to the prediction performance achieved with a MLP which is trained with a Newton conjugate gradient algorithm.

The remainder of the paper is organized as follows. Section 2 describes the case study and the applied data, which are derived from the tilting system of a railway vehicle fleet. Section 3 presents the theoretical and computational background of each of the applied algorithms and introduces the proposed combination of algorithms to predict the occurrence of speed restrictions. Section 4 presents the results of the case study obtained with the proposed hybrid approach and compares these results to those obtained with a MLP trained with a Newton conjugate gradient algorithm. Section 5 discusses the obtained results and presents the conclusions of this research.

2. Case study

2.1. Object of the case study: railway rolling stock tilting system

In this research, the suitability of the proposed combination of algorithms for binary time series predictions is verified on a case study predicting speed restrictions caused by railway rolling stock tilting system.

Failures of tilting systems directly affect schedule adherence and thereby service reliability. Therefore, from the practical point of view, it is worthwhile for the operators to be able to predict the occurrence of these failures, to be able to prevent them or, if the failures cannot be prevented, to communicate the disturbances in advance to the passengers and infrastructure operators.

When trains run in track curves, passengers and objects inside experience centrifugal forces. The higher the speed, the higher the experienced forces in curve. In order not to affect passenger's comfort above acceptable limits, speeds are reduced in track curves. Through targeted tilting of the car body, the forces affecting the passengers and their comforts are reduced, and the trains are able to run with high speeds in curves (Luebke & Hecht, 2008) with the same comfort level for passengers. This can shorten the travelling times, especially on lines with numerous curves. Tilting systems induce 8–12% in travel time savings, compared to conventional rolling stock systems.

However, if the tilting system is not active or fails, the normal speed restrictions in curves apply and the travelling times are increased compared to the ones with functioning tilting system. If only even one of the tilting systems on any of the cars throughout the train is affected, the whole train has to reduce the speed. The recovery times, that are included in the timetables to compensate for deviations in travelling times and to buffer some unexpected events, are less than travel time savings induced by tilting systems. Therefore, failures of tilting system directly affect schedule adherence and operational stability.

If the occurring speed reductions can be predicted, either they can be anticipated and prevented by scheduling specific maintenance actions or the passengers can be informed in advance. Infrastructure operators can also use the predictions on reduced speeds and increased travel times for rescheduling and improving the overall level of service. Therefore, anticipating and preventing tilting system failures leads to increased schedule adherence and improved service reliability.

2.2. Diagnostic event data

In this study, diagnostic discrete-event data from a European railway fleet consisting of 52 train sets were considered, part of which consists of 9 and the other part of 11 coaches. The available observation period was 313 days (approximately ten months). The states of the trains show a wide variability as the vehicles have been commissioned at different times, their preventive maintenance is not performed at the same time and they operate under different operating conditions. Then, given the size of the fleet, the dataset can be believed to cover all relevant combinations of the different pertinent parameters whose consequences are reflected in the occurrences of the diagnostic events. Hence, the data are considered sufficient to demonstrate the feasibility of the approach (Fink, Nash, & Weidmann, 2013).

Data were collected automatically by event recorders which start recording parameters when a predefined diagnostic event occurs. The number and character of parameters is predefined for each specific event in the design process of the diagnostic system and is supposed to provide relevant information required to resolve the occurred failure or malfunction. There are different types of recorded information associated with a diagnostic event. These can be either measured parameters, such as outside temperature, overhead line voltage, battery voltage etc., or the current state of different related systems is recorded, e.g. if the doors are available to be opened or the heating and air conditioning system operates in a reduced power mode.

If the failures are critical and require some action of the driver, the occurring events are communicated to the driver, otherwise they are just transmitted to the maintenance personnel. For the tilting system, the occurring failures directly affect operation and thereby also schedule adherence. Events affecting the tilting system are therefore directly communicated to the driver. Furthermore, it is also one of the systems that generates a significant percentage of all diagnostic events affecting operation.

2.3. Data generation and preprocessing

The observation time period of 313 days is divided into time intervals of 6 h and the time domain is equally discretized. The length of the time interval has been chosen in a way of balancing practical application needs and the requirement of having sufficiently informative data for the algorithm to learn the evolving patterns within the time series. From the practical point of view, the possibilities of a maintenance crew for anticipating impending failures during scheduled trips are limited. The reduction of the time interval to e.g. 3 h would not provide significant additional

flexibility to the operator, given the duration of single trips of up to 4.5 h and the typical operational and planning conditions. On the other hand, a reduced time interval would also increase the sparsity of the informative patterns in the time series. On the contrary, the extension of the interval to, say, 12 h would reduce the information content of the prediction for the operator. Therefore, discrete time intervals of 6 h have been considered as a good level of aggregation.

For each of the 6-h time intervals, it is determined if at least one speed restriction due to a malfunctioning tilting system has occurred. In that case a numerical label equal to 1 is assigned to the time interval, otherwise it is assigned a numerical label equal to 0. By so doing, a binary time series is obtained, which represents which of the time intervals experienced a speed reducing event and which did not. An alternative approach to generate a time series would be to assign the exact number of speed restrictions in the considered time interval. Assuming that the speed restrictions are caused by a technical failure that can only be resolved by a specific maintenance action and given that the possibilities of a maintenance crew to intervene during scheduled trips are limited, once a speed restriction has occurred, it will persist at least for the duration of a single trip. Therefore, in practice, usually, only the first occurring speed restriction influences the train operation.

Patterns included in the data set for training and testing are those for which a speed restriction has occurred during the corresponding time period: in total, there were 501 patterns. The data patterns are sparse: only in 2.5% of the time intervals at least one speed restriction event occurred.

The foresight period of the algorithm, which is the time frame for which the predictions are performed, was set to seven days. This time frame is considered suitable for practical purposes to anticipate the occurring disruptions for operational and maintenance planning and scheduling. In practical applications, the prediction could be performed on a rolling basis.

The algorithm learns to predict time dependent patterns of the last seven days in the observation period, based on the input of the preceding 306 days. In other words, the algorithm learns to allocate the occurrence of speed restrictions for each of the 6-h time intervals for a seven-day time period.

This procedure results in an input dimension of 1216 (306 days with time intervals of 6 h) and an output dimension of 28 (seven - days with time intervals of 6 h).

3. Applied algorithms

3.1. General concepts of artificial neural networks

Artificial neural networks are computational models that imitate basic features of biological neural networks and deduce functional behavior from data. They are composed of basic (mostly) nonlinear processing units, usually referred to as neurons, interconnected by information storing entities, the so-called weights (Haykin, 2009). Neural networks are ideally suited to problems for which little or no exact information is known regarding the input–output functional relationships, but where data samples (patterns of input–output) are available. These features make artificial neural networks a potentially valuable method for reliability analysis tasks such as predicting degradation processes and failure behaviors of components and systems, like railway rolling stock and infrastructure systems.

Different types of artificial neural networks are able to model static and dynamic processes. Static processes are often modeled by feed-forward neural networks (Haykin, 2009), dynamic processes by recurrent neural networks, which are capable of integrating memory in the learning process (Mandic & Chambers, 2001).

Artificial neural networks can implicitly learn functional relationships from data patterns in three different ways: supervised, unsupervised and reinforced (Haykin, 2009). Supervised learning relates to learning with a teacher. Inputs and corresponding outputs are presented to the neural network in a learning process, during which the information storing weights (the parameters of the neural model) are updated so as to minimize a predefined function measuring the error between the true output and the output predicted by the neural network (Haykin, 2009). In the case of unsupervised learning, the true outputs are not known, and the task of the artificial neural network is to identify structure and patterns existing within the data (Haykin, 2009). A mixed learning paradigm is reinforcement learning: the learning algorithm rewards the network if the computed outputs are correct, without actually presenting the true output (Haykin, 2009).

In the work presented in this paper, we use conditional restricted Boltzmann machines (CRBM) and echo state networks (ESN). CRBM usually learn in an unsupervised way (Ackley, Hinton, & Sejnowski, 1985). ESN can be used for both, supervised as well as unsupervised learning (Jaeger, 2005).

3.2. Echo state networks

3.2.1. General concepts of ESN

ESN are a specific type of recurrent neural networks. Similar to other recurrent neural networks, ESN are able to exhibit dynamic temporal behavior and have a memory. ESN are typically applied for modeling complex dynamic systems. They have been used in practical applications, such as iterated prediction of time series, signal classification and dynamic pattern recognition tasks (Jaeger & Haas, 2004). The ESN algorithm overcomes some shortcomings of more commonly used networks, such as Jordan–Elman networks, and it has demonstrated a capability of achieving remarkable results in several benchmark studies (Verstraeten, 2009).

In contrast to many other neural networks, such as MLP, the main structural element of ESN is a reservoir rather than a structure of neurons organized in layers (Verstraeten, 2009). The ESN reservoir contains a defined number of randomly and sparsely connected neurons. The weights between the connected neurons within the reservoir are fixed and are not trained during the training process.

For an ESN, the training process works by presenting input signals (training signals) into the reservoir to induce nonlinear responses. The single neurons exhibit an echoed response to the training signal and generate a variation or a transformation of the induced training signal. Subsequently, a desired output signal is determined by a trainable linear combination of all the generated response signals. In supervised learning, a teacher signal is fed back to the reservoir.

The ESN main difference from other neural networks is that only the weights of the reservoir output signals are trained. The weights of the connections within the reservoir are not trained but are generated randomly. This approach significantly reduces the learning process compared to other algorithms (e.g. backpropagation through time) (Jaeger, 2005).

One of the major properties of the echo state network is the so-called echo state property. This property ensures that the network has a fading memory and thus that the influence of initial conditions, which are randomly generated, diminishes asymptotically (Jaeger, 2005). Therefore, at the end of the training process only learned relationships influence the output.

3.2.2. Mathematical background of ESN

The theoretical concepts and the training algorithms for the echo state networks were derived from Jaeger and Haas (2004), Jaeger (2005), Jaeger, Lukosevicius, Popovici, and Siewert (2007),

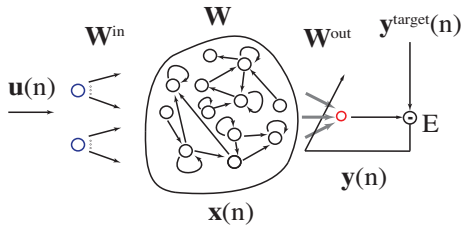

Fig. 1. Functioning principle of an echo state network based on Lukosevicius (2012).

Lukosevicius (2012) and Verstraeten (2009) and are presented in the following.

Echo state networks are applied to learn a functional relationship between the given input $\mathbf{u}(n) \in \mathbb{R}^{N_u}$ and a desired output $y^{target}(n) \in \mathbb{R}^{N_y}$, where $n = 1,\ldots,T$ is the discrete time and T is the number of data points in the training dataset $\aleph = \{(\mathbf{u}(n), y^{target}(n))\}$. The task is to learn $y(n) \in \mathbb{R}^{N_y}$, minimizing an error measure $E(\mathbf{y},\mathbf{y}^{target})$ and generalizing on data that was unseen by the algorithm. The main principles of the ESN along with the main annotations are presented in Fig. 1.

The error measure is typically a Mean-Square Error (MSE), such as the Root-Mean-Square Error (RMSE):

$$E(\mathbf{y},\mathbf{y}^{target}) = \frac{1}{N_y}\sum_{i=1}^{N_y}\sqrt{\frac{1}{T}\sum_{n=1}^{T}\left(y_i(n) - y_i^{target}(n)\right)^2}, \tag{1}$$

RMSE is additionally averaged over the N_y dimensions i of the output.

The basic procedure to generate ESN (Fig. 1) comprises following steps (Jaeger, 2005):

1. Generate a large random reservoir ($\mathbf{W}^{in}$,$\mathbf{W}$, α);
2. Run it using the training input $\mathbf{u}(n)$ and collect the corresponding reservoir activation states $\mathbf{x}(n)$;
3. Compute the linear readout weights $\mathbf{W}^{out}$ from the reservoir using linear regression, minimizing the MSE between $\mathbf{y}(n)$ and $\mathbf{y}^{target}(n)$;
4. Use the trained network on new input data $\mathbf{u}(n)$, computing $\mathbf{y}(n)$ by employing the trained output weights $\mathbf{W}^{out}$.

In case of unsupervised learning the target output $\mathbf{y}^{target}(n)$ is not available and therefore only steps 1 and 2 are performed.

The annotation and the detailed procedures of the single steps are explained in the following.

The basic discrete-time, sigmoid-unit echo state network with N reservoir units, K inputs and L outputs is governed by the state update equation

$$\mathbf{x}(n) = f(\mathbf{W}^{in}\mathbf{u}(n) + \mathbf{W}\mathbf{x}(n-1)), \quad n = 1,\ldots,T \tag{2}$$

where $\mathbf{x}(n) \in \mathbb{R}^{N_x}$ is a vector of reservoir neuron activations at a time step n; $f(\cdot)$ is the neuron activation function, usually the symmetric tanh $(\cdot)$, whereby tanh $(\cdot)$ is applied element-wise; $\mathbf{W}^{in} \in \mathbb{R}^{N_x \times N_u}$ is the input weight matrix and $\mathbf{W} \in \mathbb{R}^{N_x \times N_x}$ is a weight matrix of the internal network connections. The network is usually started in the initial state $\mathbf{x}(0) = \mathbf{0}$. Bias values are omitted in Eq. (2).

It is also possible to include memory in the single neurons by performing leaky integration of its activation from previous time steps. Usually, the leaky integration is performed before the activation $f(\cdot)$ is applied.

$$\mathbf{x}(n) = (1-\alpha)\mathbf{x}(n-1) + \alpha\mathbf{f}(\mathbf{W}^{in}\mathbf{u}(n) + \mathbf{W}\mathbf{x}(n-1)) \tag{3}$$

where $\alpha \in [0,1]$ is the leaking rate. If $\alpha = 1$, the leaky integration is not applied. Eq. (3) corresponds to the exponential moving average.

The readout from the reservoir is usually linear and is defined by Lukosevicius (2012):

$$\mathbf{y}(n) = f^{out}(\mathbf{W}^{out}[\mathbf{u}(n)|\mathbf{x}(n)]) \tag{4}$$

where $\mathbf{W}^{out} \in \mathbb{R}^{N_y \times (N_u + N_x)}$ is the learned output weight matrix, $f^{out}(\cdot)$ is the output activation function (which is usually identity function). $f^{out}(\cdot)$ is applied component-wise, whereby $\cdot|\cdot$ stands for a vertical vector (or matrix) concatenation. Feedback connections $\mathbf{W}^{fb}$ from $\mathbf{y}(\mathbf{n}-\mathbf{1})$ to $\mathbf{x}(n)$ in Eq. (2) can be additionally applied.

In batch learning, output weights $\mathbf{W}^{out}$ (Eq. (4)) can be obtained by solving a system of linear equations:

$$\mathbf{W}^{out}\mathbf{X} = \mathbf{Y}^{target}, \tag{5}$$

with respect to $\mathbf{W}^{out}$. In this equation, $\mathbf{X} \in \mathbb{R}^{N \times T}$ and $\mathbf{Y}^{target} \in \mathbb{R}^{N_y \times T}$ are matrices of $\mathbf{x}(n)$ and $\mathbf{y}^{target}(n)$ over the training period $n = 1,\ldots,T$. Linear regression is applied to minimize the error $E(\mathbf{Y}^{target},\mathbf{W}^{out}\mathbf{X})$. *Moore–Penrose pseudoinverse* $\mathbf{X}^+$ of $\mathbf{X}$ can be applied to calculate $\mathbf{W}^{out}$:

$$\mathbf{W}^{out} = \mathbf{Y}^{target}\mathbf{X}^{+}. \tag{6}$$

Although pseudoinverse show high numerical stability, they are expensive memory-wise. To solve this *normal equations* the following approach can be applied:

$$\mathbf{W}^{out}\mathbf{X}\mathbf{X}^T = \mathbf{Y}^{target}\mathbf{X}^T \tag{7}$$

This equation can be solved as follows:

$$\mathbf{W}^{out} = \mathbf{Y}^{target}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1} \tag{8}$$

It is worth noting that $\mathbf{Y}^{target}\mathbf{X}^T \in \mathbb{R}^{N_y \times N}$ and $\mathbf{X}\mathbf{X}^T \in \mathbb{R}^{N \times N}$ do not depend on the length T of the training sequence, and can be calculated incrementally while the training data are passed through the reservoir.

A regularization factor can be added to Eq. (8) to impose rigidity (Hoerl & Kennard, 1970):

$$\mathbf{W}^{out} = \mathbf{Y}^{target}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \beta^2\mathbf{I})^{-1} \tag{9}$$

where $\mathbf{I} \in \mathbb{R}^{\mathbb{N} \times \mathbb{N}}$ is the identity matrix and β is the regularization factor.

3.3. Conditional restricted Boltzmann machines

3.3.1. General concepts of CRBM

Restricted Boltzmann machines (RBM) comprise symmetrically connected neuron-like units, composed to network structures. RBM are also referred to as stochastic neural networks. They consist of two layers: a visible layer and a hidden layer. Each unit in the visible layer is connected to all units in the hidden layer and vice versa. The visible layer contains the input parameters; the hidden layer contains the latent parameters that the networks learn (Ackley et al., 1985). The hidden layer learns to model the distribution of the visible layer of variables. However, the units within one layer are not interconnected. Therefore, the networks are called restricted. This restriction simplifies the learning process.

Hidden units in RBM learn the structure and features that are contained in the data. The extracted features can be, for example, parameters that influence the input data and cause a higher-order correlation between the input dimensions but cannot be observed or measured.

The conditional restricted Boltzmann machines (CRBM) extend the concept of RBM. They enable capturing temporal dependencies (Salakhutdinov et al., 2007). Furthermore, it is possible not only to use the information of visible variables but also to include context variables and thereby to enrich the content of information in the model. CRBM are particularly suitable for time series applications

and have been applied to different kinds of problems, such as collaborative filtering (Salakhutdinov et al., 2007), classification and modeling motion capture data (Taylor & Hinton, 2009).

CRBM are capable of learning and reproducing complex dynamics directly from data (Zeiler, Taylor, Troje, & Hinton, 2009). Their concept regarding hidden layers is similar to Hidden Markov Models, which also have hidden states (Zeiler et al., 2009). However, the binary hidden state of CRBM is much more powerful, which enables the CRBM capturing longer-term dependencies (Zeiler et al., 2009).

3.3.2. Mathematical background of RBM and CRBM

The concepts of RBM and CRBM are closely related. The difference between the methods is the integration of context variables in CRBM. Therefore, first the background of RBM and subsequently the additional concepts of CRBM are introduced. The theoretical concepts and the training algorithms for the RBM and CRBM were derived from Ackley et al. (1985), Taylor and Hinton (2009), Salakhutdinov et al. (2007) and Zeiler et al. (2009) and are presented in the following.

A RBM network consists of visible and hidden units. Hidden units can be either modeled as binary or as Gaussian units (Taylor & Hinton, 2009).

A joint configuration $(\mathbf{v},\mathbf{h})$ of the visible and hidden units has an energy given by

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i\in visible} a_i v_i - \sum_{j\in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{10}$$

where v_i, h_j are the binary states of visible unit i and hidden unit j, a_i, b_j are their biases and w_{ij} is the weight between them. The network assigns a probability to every pair of a visible and a hidden vector via the following energy function:

$$p(\mathbf{v},\mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})} \tag{11}$$

where the *partition function* Z is a normalization term given summing over all possible pairs of visible and hidden vectors:

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{12}$$

The probability that the network assigns to a visible vector, $\mathbf{v}$, is given by summing over all possible hidden vectors:

$$p(\mathbf{v}) = \frac{1}{Z}\sum_{h} e^{-E(\mathbf{v},\mathbf{h})} \tag{13}$$

Given a training set of state vectors (the data), learning consists of finding the pertinent parameters (weights and biases) that define a Boltzmann distribution in which the training vectors have high probability.

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{model} \tag{14}$$

where $\langle\cdot\rangle_{data}$ is an expected value in the data distribution and $\langle\cdot\rangle_{model}$ is an expected value when the Boltzmann machine is sampling state vectors from its equilibrium distribution. This leads to a very simple learning rule for performing stochastic steepest ascent in the log probability of the training data:

$$\triangle w_{ij} = \epsilon \frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \epsilon(\langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{model}) \tag{15}$$

where ϵ is a learning rate.

Because there are no direct connections between hidden units in an RBM, it is very easy to get an unbiased sample of $\langle v_i h_j\rangle_{data}$. Given a randomly selected training pattern, $\mathbf{v}$, the binary state, h_j, of each hidden unit, j, is set to 1 with probability

$$p(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_i v_i w_{ij}\right) \tag{16}$$

where $\sigma(x)$ is the logistic sigmoid function $1/(1+exp(-x))$. $v_i h_j$ is then an unbiased sample.

Because there are no direct connections between visible units in an RBM, it is also very easy to get an unbiased sample of the state of a visible unit, *given a hidden vector*

$$p(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_j h_j w_{ij}\right) \tag{17}$$

To get an unbiased sample of $\langle v_i h_j\rangle_{model}$ an approach to reconstruct $\langle v_i h_j\rangle_{model}$ has been proposed. In this case, $\langle v_i h_j\rangle_{model}$ is replaced by $\langle v_i h_j\rangle_{recon}$. In this case firstly, the states of the visible units to a training vector are set. Secondly, the binary states of the hidden units are computed by applying Eq. (16). Once binary states have been chosen for the hidden units, a reconstruction is produced by setting each v_i to 1 with a probability given by Eq. (17). Therefore, an incremental learning process is required.

For Gaussian hidden units Eq. (10) is adjusted to:

$$E(\mathbf{v},\mathbf{h}) = -\sum_i a_i v_i + \sum_j \frac{(h_j - b_j)}{2\sigma_j^2} - \sum_{i,j} v_i \frac{h_j}{\sigma_j} w_{ij} \tag{18}$$

where σ_j^2 is the variance of the hidden unit j. The variances of all hidden units are usually fixed to $\sigma_j^2 = 1$.

The difference between a RBM and a CRBM is that CRBM takes additional context information into account. The model defines a joint probability distribution over $\mathbf{v}$ and $\mathbf{h}$, conditional on the context variables $\mathbf{c}$ and the model parameters, θ:

$$p(\mathbf{v},\mathbf{h}|\mathbf{c},\theta) = \frac{e^{-E(\mathbf{v},\mathbf{h}|\mathbf{c},\theta)}}{Z} \tag{19}$$

where Z is again the *partition function*. However, in this case it is exponentially expensive to compute exactly. The energy term can be computed similarly to Eq. (18).

The same learning rule as for RBM applies (Eq. (15)), with the difference that in the case of CRBM it maximizes the log probabilities of the observed output vectors conditional on the input vectors.

3.4. Principal component analysis

Variables are often correlated in large data sets. This means that the content of information in the correlated variables is lower than in uncorrelated variables. Furthermore, high dimensional input data increases computational resources needed for data analysis. The process of principle component analysis (PCA) can be applied to reduce the dimensionality of a data set with many correlated variables, without any significant loss of information (Jolliffe, 2002). In the PCA process, the original data is converted by orthogonal transformation into a set of uncorrelated variables referred to as principle components (PC) without inducing any significant loss in variation compared to the original data set. Principal components can be considered as a lower dimensional representation of the input data.

The principal component analysis is based on the principle of the change of basis, in which a basis in obtained that is a linear combination of the original basis and that re-expresses the data optimally, with a maximal independence between the single components. The independence is in this case defined by considering the variance of the data in the original basis and the PCA seeks to de-correlate the original data. This is performed by finding the directions in which the variance is maximized. Subsequently these directions are used to define the new basis.

Given $\mathbf{X}$, a $m \times n$ matrix, where the n columns represent the samples and the m rows represent the variables, the principal components of $\mathbf{X}$ are given by the eigenvectors of the covariance matrix $\mathbf{C_X} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$.

3.5. Combining CRBM with ESN and PCA

CRBM have proven to have benefits for uncovering time dependencies in data, especially for long-term dependencies (Salakhutdinov et al., 2007). ESN have proven to be powerful in extracting dynamical features (Jaeger & Haas, 2004). These two methods are, then, here originally combined for binary time series predictions. While CRBM is an unsupervised algorithm, ESN is applied in this approach both in an unsupervised and in a supervised way. Principle component analysis is applied to extract the uncorrelated features and to reduce the dimensionality of the data.

Generally, the input of CRBM consists of visible data and additional explanatory input. Explanatory influencing parameters can be used to create a context, and to enrich and facilitate the pattern extraction. The context variables are usually variables that directly or indirectly influence the output values. These can be, for instance, complementary environmental parameters. For this case study, it is assumed that ESN are able to extract relevant dynamic features of the time series in an unsupervised way. The extracted dynamic features are used as context variables for the CRBM. Therefore, the binary time series was presented to the reservoir of the ESN to extract complementary features. The applied ESN consisted of a reservoir with leaky integrator neurons, which enable integrating memory in single neurons. The applied leak rate was 0.88. The applied activation function within the neurons was a standard sigmoid function. No explicit bias was included in the model. The reservoir size was set to 500 neurons. Nonlinear responses are induced in each of the neurons by presenting the input to the reservoir (Eq. (2)).

In the next step, principal components of the output of the ESN were extracted. Principal components are uncorrelated features of the inputs. By applying PCA, a part of variability within the data set is lost. However, this is accepted in this case study. The dimension of the temporal features extracted by the ESN could be reduced from 500 features to 150 principal components. The lower dimensional representation of the dynamic responses of the ESN reservoir were then used as complementary inputs to the CRBM. The observed time series patterns with the input dimension of 1216 were used as visible input to the CRBM (Fig. 2). The one-layer CRBM contained 600 hidden units, which were trained for 10 epochs. In the learning process of CRBM, the logarithms of the probabilities of the hidden and visible variables are maximized conditional on the context variables (Eqs. (19) and (15)).

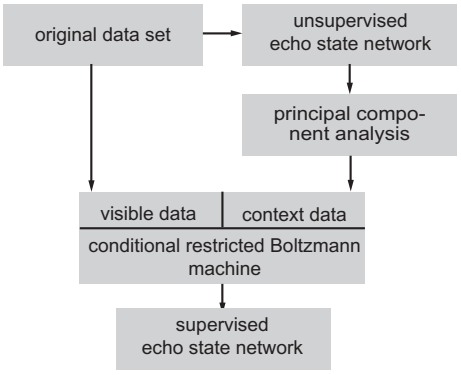

Fig. 2. Applied combination of algorithms.

In the successive step, the output of the CRBM was presented to an echo state reservoir, where the output of the CRBM was induced in 500 neurons of the ESN reservoir. Similarly to first reservoir, leaky integrator neurons with a leak rate of 0.88 and standard sigmoid activation function were applied. No explicit bias was included in the model. The ESN learned to extract the dynamic patterns from the presented input in a supervised way (Eq. (9)). Target output was the binary time series pattern of occurring speed restrictions in the last seven days of the time pattern. Ridge regression with a regularization term was applied to determine the optimal combination of reservoir signals that generate the desired output signal. In this case study a regularization factor of 0.01 was applied.

In this research the network parameters were determined by performing a hyperparameter optimization with grid search (Bengio, 2000). Cross-validation with 10 folds was applied within the grid search process.

The steps of the procedure are presented in Fig. 2.

Through the combination of these steps, the learning of the process is very efficient, because only the last layer is trained in a supervised way. Additionally, for an ESN only the weights between the reservoir and the output are trained by a linear regression with a regularization term.

4. Results

Holdout has been used to evaluate the generalization ability of the algorithm (Haykin, 2009). The data set is divided in two subsets. Both subsets are assumed to be independent and representative of the underlying data distribution of the entire data set. In this research, 90% of data were used for training and 10% for testing.

In total, 451 patterns were applied for training and 50 were kept for testing.

Fig. 3 shows a part of the patterns of the testing data set (only 25 patterns with 28 time intervals (the intervals to be predicted) are visualized). In the Figure, each of the squares represents a time interval of 6 h for a selected coach. The time intervals when speed restrictions occurred are marked grey; when no speed restrictions occurred the patterns are marked white.

For the testing data set, the algorithm had to make in total 1400 binary decisions (28 six-hour time intervals in the foresight period of seven days $\times$ 50 data patterns in the testing data set), if a speed restriction would occur within the considered time interval of 6 h or not. The overall prediction precision of the algorithm is very good: all of the time intervals without speed restrictions were recognized correctly by the algorithm and only one speed restriction was not recognized. With 1400 six-hour time intervals in the foresight period for all the considered coaches in the testing data set, this resulted in 0.07% of incorrectly predicted occurring events. Considering only the prediction precision of the algorithm for the occurrence of speed restrictions in the foresight period of seven days, 98% of all the speed restrictions were recognized correctly by the algorithm.

The results confirm a very good generalization ability of the algorithm. As the data is sparse, the parameter defining the performance of the algorithm is the sensitivity, which is high in this case study. This shows that the algorithm does not purely assign no occurring speed restrictions to each of the discrete time intervals, but is able to precisely allocate the occurring speed restrictions to the specific time intervals. Usually, overfitting occurs when the algorithm fits the training data perfectly but is not able to generalize and transfer the patterns to data that have not yet been presented to the algorithm. However, the proposed combination of algorithms was able to extract the patterns within the binary time series and transfer these patterns to previously unseen patterns. The performance of the algorithm on the testing data set was very good.

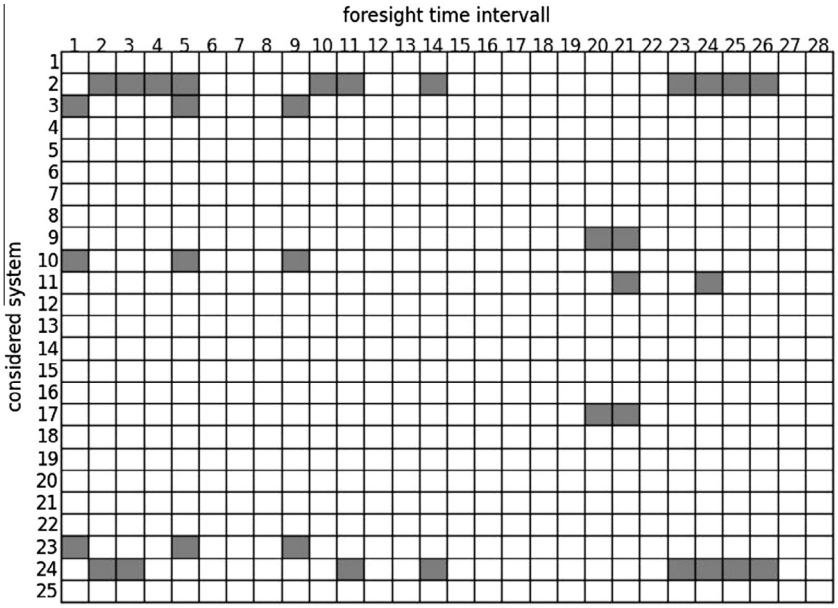

Fig. 3. Example of data patterns in the testing data set.

Additionally to the proposed algorithm, the same data set was analysed with a MLP for comparison. A MLP with two hidden layers was chosen with 50 neurons in the first layer and 30 neurons in the second layer. MLP with only a single hidden layer have been proven to be universal approximators (Hornik, Stinchcombe, & White, 1989). However, the number of neurons required to achieve the desired performance can be substantial. MLP with two hidden layers have been proven to be capable of approximating functions with the desired precision with a finite number of neurons (Huang et al., 2003).

The MLP was trained with a Newton conjugate gradient algorithm, which has been shown to have superior performance over a standard backpropagation learning algorithm (Johansson, Dowla, & Goodman, 1991). The optimally trained MLP could correctly predict 98.8% of all the occurring events within the foresight period of the testing data set: this corresponds to 1.2% of incorrectly predicted events, compared to 0.07% predicted by the proposed algorithm. Considering only the occurring speed restrictions within the foresight period, MLP could predict 77.8% of the occurring events correctly, compared to 98% correctly predicted occurring events by the proposed algorithm.

5. Discussion and conclusions

In this paper, we have proposed to combine ESN and CRBM for binary time series predictions. Complementarily, the hidden distribution of the variables is processed with a principal component analysis. The proposed approach has been applied for predicting the occurrence of train speed restrictions caused by the tilting system. In the case study analysed, the combination of algorithms proposed shows a very good ability to generalize the patterns in the binary time series, with a superior performance compared to MLP.

The interval of 6 h selected appears to be a good level of aggregation, balancing the requirements of the application perspective, providing a sufficient resolution given operational and planning conditions, and the requirements of the algorithms, providing a sufficient number of informative patterns to learn from.

The proposed approach is applicable for dynamic time series patterns with an adequately frequent occurrence of events or a sufficiently long observation period, in order to ensure that a sufficient quantity of informative patterns is presented to the algorithm. As shown in the case study, even though the occurrence of speed restricting events is seldom, so that the data set contained a lot more time intervals in which no event occurred, the combination of algorithms was capable of learning to predict the occurring speed restrictions on a foresight period of seven days.

Acknowledgments

The authors thank Alstom Transportation for providing the data for this research project.

The participation of Olga Fink to this research is partially supported by the Swiss National Science Foundation (SNF) under Grant No. 205121_147175.

The participation of Enrico Zio to this research is partially supported by the China NSFC under Grant No. 71231001.

References

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science, 9*, 147–169.

Bengio, Y. (2000). Gradient-based optimization of hyperparameters. *Neural Computation, 12*, 1889–1900.

Chatterjee, S., & Bandopadhyay, S. (2012). Reliability estimation using a genetic algorithm-based artificial neural network: An application to a load-haul-dump machine. *Expert Systems with Applications, 39*, 10943–10951.

Dorbritz, R. (2012). Methodology for assessing the structural and operational robustness of railway networks. Ph.D. thesis, ETH Zurich.

Ferreira, A. A., Ludermir, T. B., & de Aquino, R. R. B. (2013). An approach to reservoir computing design and training. *Expert Systems with Applications*.

Fink, O., Nash, A., & Weidmann, U. (2013). Predicting potential railway operations disruptions caused by critical component failure using echo state neural networks and automatically collected diagnostic data. In: *Transportation Research Record: Journal of the Transportation Research Board*, in press.

Green, P., & Silverman, B. (1994). *Nonparametric regression and generalized linear models: A roughness penalty approach* (1st ed.). London and New York: Chapman and Hall.

Haykin, S. S. (2009). *Neural networks and learning machines* (3rd ed.). Upper Saddle River: Pearson Education.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics, 12*, 55–67.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*, 359–366.

Huang, G. B. (2003). Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks, 14*, 274–281.

Jaeger, H. (2005). Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network approach. Technical report, German National Research Center for Information Technology.

Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science, 304*, 78–80.

Jaeger, H., Lukosevicius, M., Popovici, D., & Siewert, U. (2007). Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks, 20*, 335–352.

Johansson, E., Dowla, F., & Goodman, D. (1991). Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method. *International Journal of Neural Systems, 02*, 291–301.

Jolliffe, J. T. (2002). *Principal component analysis* (2nd ed.). New York: Springer.

Kedem, B., & Fokianos, K. (2002). *Regression models for time series analysis*. New York, N.Y: Wiley.

Liang, K. Y., & Zeger, S. L. (1989). A class of logistic regression models for multivariate binary time series. *Journal of the American Statistical Association, 84*, 447–451.

Lin, X., Yang, Z., & Song, Y. (2009). Short-term stock price prediction based on echo state networks. *Expert Systems with Applications, 36*, 7313–7317.

Luebke, D., & Hecht, M. (2008). *Das system Bahn Handbuch*. Hamburg: Eurailpress.

Lukosevicius, M. (2012). A practical guide to applying echo state networks. *A practical guide to applying echo state networks. Lecture notes in computer science* (Vol. 7700). Berlin Heidelberg: Springer.

MacDonald, I. L., & Zucchini, W. (1997). *Hidden markov and other models for discrete-valued time series*. New York: Chapman and Hall.

Mandic, D. P., & Chambers, J. A. (2001). *Recurrent neural networks for prediction learning algorithms, architectures and stability*. Chichester: Wiley.

McCullagh, P., & Nelder, J. A. (1989). *Generalized linear models* (2nd ed.). London: Chapman and Hall.

Pai, P. F. (2006). System reliability forecasting by support vector machines with genetic algorithms. *Mathematical and Computer Modelling, 43*, 262–274.

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on machine learning, ICML '07* (pp. 791–798). New York, NY, USA: ACM.

Taylor, G. W., & Hinton, G. E. (2009). Factored conditional restricted Boltzmann machines for modeling motion style. In *Proceedings of the 26th annual international conference on machine learning, ICML '09* (pp. 1025–1032). New York, NY, USA: ACM.

Vachtsevanos, G. (2006). *Intelligent fault diagnosis and prognosis for engineering systems*. Hoboken, NJ: Wiley.

Verstraeten, D. (2009). Reservoir computing: Computation with dynamical systems. Ph.D. thesis, Ghent University.

Xu, K., Xie, M., Tang, L. C., & Ho, S. L. (2003). Application of neural networks in forecasting engine systems reliability. *Applied Soft Computing, 2*, 255–268.

Yadav, A., Mishra, D., Yadav, R. N., Ray, S., & Kalra, P. K. (2007). Time-series prediction with single integrate-and-fire neuron. *Applied Soft Computing, 7*, 739–745.

Zainuddin, Z., & Pauline, O. (2011). Modified wavelet neural network in function approximation and its application in prediction of time-series pollution data. *Applied Soft Computing, 11*, 4866–4874.

Zeiler, M., Taylor, G., Troje, N., & Hinton, G. (2009). Modeling pigeon behaviour using a conditional restricted boltzmann machine. In: *17th European symposium on artificial neural networks (ESANN)*, d-side publishing.

Zhang, Y. Q., & Wan, X. (2007). Statistical fuzzy interval neural networks for currency exchange rate time series prediction. *Applied Soft Computing, 7*, 1149–1156.