Neurocomputing 192 (2016) 139–146

Contents lists available at ScienceDirect

Neurocomputing

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

Model-coupled autoencoder for time series visualisation

Nikolaos Gianniotis [a,*], Sven D. Kügler [a], Peter Tiňo [b], Kai L. Polsterer [a]

[a] Astroinformatics Group, Heidelberg Institute for Theoretical Studies (HITS), Schloss-Wolfsbrunnenweg 35, 69118 Heidelberg, Germany
[b] School of Computer Science, The University of Birmingham, Edgbaston, Birmingham B15 2TT, United Kingdom

ARTICLE INFO

Article history:
Received 10 July 2015
Received in revised form
19 January 2016
Accepted 27 January 2016
Available online 2 March 2016

Keywords:
Time series
Dimensionality reduction
Echo state network
Autoencoder

ABSTRACT

We present an approach for the visualisation of a set of time series that combines an echo state network with an autoencoder. For each time series in the dataset we train an echo state network, using a common and fixed reservoir of hidden neurons, and use the optimised readout weights as the new representation. Dimensionality reduction is then performed via an autoencoder on the readout weight representations. The crux of the work is to equip the autoencoder with a loss function that correctly interprets the reconstructed readout weights by associating them with a reconstruction error measured in the data space of sequences. This essentially amounts to measuring the predictive performance that the reconstructed readout weights exhibit on their corresponding sequences when plugged back into the echo state network with the same fixed reservoir. We demonstrate that the proposed visualisation framework can deal both with real valued sequences and binary sequences. We derive magnification factors in order to analyse distance preservations and distortions in the visualisation space. The versatility and advantages of the proposed method are demonstrated on datasets of time series that originate from diverse domains.

1. Introduction

Time series[1] are sequences of observations that exhibit short or long term dependencies between them in time. These dependencies can be thought of as manifestations of a latent regime (e.g. natural law) governing the behaviour of the time series. Machine learning approaches designed to deal with data of a vectorial nature have no knowledge of such temporal dependencies. By introducing a model that accounts for temporal behaviour, algorithms can become “aware” of the sequential nature of the data and make the most of the available information.

Echo state networks (ESNs) [1] are discrete time recurrent neural networks that have emerged as a popular method to capture the latent regime underlying a time series. ESNs have the great advantage that the hidden part, the reservoir of neurons, is fixed and only the output weights need to be trained. The ESN is thus essentially a linear model and so the output weights, also known as readout weights, can thus be easily optimised via least squares. The processing of structured data has been a topic of research for a long time [2,3]. Regarding time series, recent attempts [4–6] have exploited the predictive capabilities of ESNs in regression and classification tasks. In the unsupervised setting, the work in [7] suggests compressing a linear state space model through a linear autoencoder in order to extract vectorial representations of structured data. The work in [8] considers the visualisation of individual observations belonging to a *single* sequence by temporally linking them using an ESN.

In this work, we employ the ESN in the formulation of a dimensionality reduction algorithm for visualising a dataset of time series (we extend previous work presented in [9]). Given a fixed reservoir, the only free parameters in the ESN are in the readout weight vector which maps the state space to the sequence space. Thus, an optimised (i.e. trained) readout weight vector uniquely addresses an instance of the ESN (always for the same fixed reservoir) that best predicts on a given sequence. We can also reason backwards: given an observed sequence, we can train the ESN (Section 2.1) and identify the readout weight vector that best predicts the given sequence. Hence, each sequence in the dataset can be mapped to the readout weight vector that exhibits the best predictive performance on it. These readout weight vectors in conjunction with the common and fixed reservoir, capture temporal features of their respective sequences. Representing sequences as weight vectors constitutes the first part of our proposed approach (Section 3.1).

The second stage of our approach involves training an autoencoder [10] on the obtained readout weight vectors in order to induce a two-dimensional representation, the visualisation, at the bottleneck. At the heart of the autoencoder lies the reconstruction error function which drives the visualisation induced at the bottleneck. During training, the autoencoder receives readout weights as inputs, compresses them at the bottleneck, and outputs an approximate version of the inputs, the reconstructed readout

* Corresponding author.
[1] We interchangeably use the terms time series and sequence.

http://dx.doi.org/10.1016/j.neucom.2016.01.086
0925-2312/

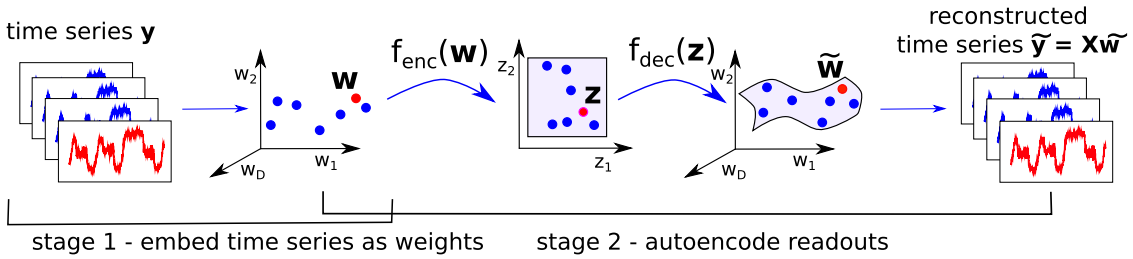

Fig. 1. Sketch of proposed method. In a first stage, time series $\boldsymbol{y}$ are cast to readout weights $\boldsymbol{w}$ in the weight space (see Section 3.1). In a second stage, the autoencoder projects readout weights $\boldsymbol{w}$ onto coordinates $\boldsymbol{z}$ residing in a two-dimensional space, and reconstructs them again as $\tilde{\boldsymbol{w}}$ (see Section 3.2). By multiplying with the state space, given by $\boldsymbol{X}$, we map the reconstructed readout weights $\tilde{\boldsymbol{w}}$ to the sequence space where reconstruction error is measured (see Eq. (7)).

weights. Typically, one would take as the reconstruction error function the L_2 norm between the original readout weights and reconstructed readout weights. In the proposed work, we equip the autoencoder with a different reconstruction function that assesses how well the reconstructed readout weights still predict on the sequence that it represents. If it predicts well, we deem it a good reconstruction; if it predicts poorly, we deem it a poor reconstruction (Section 3.2). An overview of the proposed method is displayed in Fig. 1.

In Section 6, we show that the autoencoder with the proposed reconstruction error function is capable of interpreting similarities between time series better than other dimensionality reduction algorithms. In Section 7, we discuss the possibility of alternative formulations of the proposed approach before concluding with some remarks on future work in Section 8.

2. Preliminary

This section introduces some notation and terminology while briefly reviewing ESNs and the autoencoder.

2.1. Echo state networks

An ESN is a discrete-time recurrent neural network with fading memory. It processes time series composed of a sequence of observations $y(t) \in \mathbb{R}$ over time t that we denote here by $\boldsymbol{y} = (y(1), \ldots, y(T))$, where T is the length[2] of the sequences. Hence $\boldsymbol{y} \in \mathbb{R}^{T\times 1}$. Given an input $y(t)$, the task of the ESN is to make a prediction $\hat{y}(t+1)$ for the observation $y(t+1)$ in the next time step. Similar to a feedforward neural network, the ESN comprises an input layer with weights $\boldsymbol{v} \in \mathbb{R}^{D\times 1}$, a hidden layer with weights $\boldsymbol{U} \in \mathbb{R}^{D\times D}$ (hence D is the size of the reservoir) and an output layer with weights $\boldsymbol{w} \in \mathbb{R}^{D\times 1}$, the latter weights $\boldsymbol{w}$ also known as readout weights. However, in contrast to feedforward networks, ESNs equip the hidden neurons with feedback connections. The operation of an ESN is specified by the equations:

$$\boldsymbol{x}(t+1) = h(\boldsymbol{U}\boldsymbol{x}(t) + \boldsymbol{v}y(t)), \tag{1}$$

$$\hat{y}(t+1) = \boldsymbol{w}^T\boldsymbol{x}(t+1), \tag{2}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^{D\times 1}$ are the hidden activations of the reservoir at time t, and $h(\cdot)$ is a nonlinear function commonly chosen as the $\tanh(\cdot)$ function. Bias terms have been omitted in the formulation for the sake of clarity in notation.

According to standard ESN methodology [1], parameters $\boldsymbol{v}$ and $\boldsymbol{U}$ in Eqs. (1), (2) are randomly generated[3] and fixed. The only trainable parameters are the readout weights $\boldsymbol{w}$. Training involves feeding at each time step t an observation $y(t)$ and recording the resulting activations $\boldsymbol{x}(t)$ row-wise into a matrix $\boldsymbol{X} \in \mathbb{R}^{T\times D}$. Usually, some initial observations are dismissed in order to "washout" [1] dependence on the initial arbitrary reservoir state (e.g. $\boldsymbol{x}(1) = \boldsymbol{0}$). Given matrix $\boldsymbol{X}$, the following objective function is minimised:

$$\ell(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2. \tag{3}$$

The above objective can be supplemented by a regularisation term and so the combined objective is $\ell(\boldsymbol{w}) + \mu^2\|\boldsymbol{w}\|^2$. The combined objective can be exactly minimised by solving the pertaining least squares problem and obtaining $\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \mu^2\boldsymbol{I}_D)^{-1}\boldsymbol{X}^T\boldsymbol{y}$ as the solution, where $\boldsymbol{I}_D$ is the $D \times D$ identity matrix. Given this result, we introduce function $g(\boldsymbol{y})$ that maps a given time series to the optimal readout weights:

$$g(\boldsymbol{y}) = (\boldsymbol{X}^T\boldsymbol{X} + \mu^2\boldsymbol{I}_D)^{-1}\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{w}. \tag{4}$$

2.2. Deterministically constructed echo state networks

In the original formulation of the ESN [1] the weights in $\boldsymbol{v}$ and $\boldsymbol{U}$ are generated stochastically and so are the connections between the hidden neurons in the reservoir. This makes the training and use of the ESN dependent on random initialisations. In order to avoid this source of randomness, we make use of a class of ESNs that are constructed in a deterministic fashion [11].

Deterministic ESNs make several simplifications over standard ESNs. All entries in $\boldsymbol{v}$ have the same absolute value of a single scalar parameter $v > 0$. The signs of the entries in $\boldsymbol{v}$ are deterministically generated by an aperiodic sequence: e.g. a pseudorandom binary sequence (coin flips), with outcomes 0 and 1 corresponding to $-$ and $+$ respectively. Similarly, the entries in $\boldsymbol{U}$ are parametrised by a single scalar $u > 0$. As opposed to random connectivity, deterministic ESNs impose a fixed regular topology on the hidden neurons in the reservoir. Amongst possible choices, one can arrange the neurons in a cycle. A cyclic arrangement imposes the following structure on $\boldsymbol{U}$: the only nonzero entries in $\boldsymbol{U}$ are on the lower subdiagonal $\boldsymbol{U}_{i+1,i} = u$, and at the upper-right corner $\boldsymbol{U}_{1,D} = u$. An illustration of a cyclic deterministic ESN is shown in Fig. 2.

In this work we employ deterministic ESNs with a cyclic connectivity. Deterministic ESNs have three degrees of freedom: the reservoir size D, the input weight v and reservoir weight u. Hence, the triple (D, v, u) completely specifies an ESN. It has been shown empirically and theoretically (memory capacity) [11] that deterministic ESNs perform up to par with their stochastic counterparts. Training of a deterministic ESN is performed in exactly the same fashion as in stochastically constructed ESNs using the objective $\ell(\boldsymbol{w})$ in Eq. (3).

2.3. Autoencoder

The autoencoder [10] is a feedforward neural network that defines a three hidden layer architecture[4] with the middle layer,

[2] In general, each sequence can have its own length T_n. For ease of exposition, here all sequences have the same T.

[3] The spectral radius of the reservoir's weight matrix $\boldsymbol{U}$ is made < 1 to encourage *the Echo State Property*.

[4] To be perfectly precise, we use what is widely considered the standard autoencoder specified in [12, Section 12.4.2]).