ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

Autoencoding Time Series for Visualisation

Nikolaos Gianniotis[1], Dennis Kügler[1], Peter Tiňo[2], Kai Polsterer[1] and Ranjeev Misra[3]

1- Astroinformatics - Heidelberg Institute of Theoretical Studies
Schloss-Wolfsbrunnenweg 35 D-69118 Heidelberg - Germany

2 - School of Computer Science - The University of Birmingham
Birmingham B15 2TT - UK

3 - Inter-University Center for Astronomy and Astrophysics
Post Bag 4, Ganeshkhind, Pune-411007 - India

Abstract. We present an algorithm for the visualisation of time series. To that end we employ echo state networks to convert time series into a suitable vector representation which is capable of capturing the latent dynamics of the time series. Subsequently, the obtained vector representations are put through an autoencoder and the visualisation is constructed using the activations of the "bottleneck". The crux of the work lies with defining an objective function that quantifies the reconstruction error of these representations in a principled manner. We demonstrate the method on synthetic and real data.

1 Introduction

Time series are often considered a challenging data type to handle in machine learning tasks. Their variable-length nature has forced the derivation of feature vectors that capture various characteristics, e.g. [1]. However, it is unclear how well such (often handcrafted) features express the potentially complex latent dynamics of time series. Time series exhibit long-term dependencies which must be taken into account when comparing two time series for similarity. This temporal nature makes the use of common designs, e.g. RBF kernels, problematic. Hence, more attentive algorithmic designs are needed and indeed in classification scenarios there have been works [2, 3, 4] that successfully account for the particular nature of time series.

In this work we are interested in visualising time series. We propose a fixed-length vector representation for representing sequences that is based on the Echo State Network (ESN) [5] architecture. The great advantage of ESNs is the fact that the hidden part, the reservoir of nodes, is fixed and only the readout weights need to be trained. In this work, we take the view that the readout weight vector is a good and comprehensive representation for a time series.

In a second stage, we employ an autoencoder [6] that reduces the dimensionality of the readout weight vectors. However, employing the usual L_2 objective function for measuring reconstruction is inappropriate. What we are really interested in is not how well the readout weight vectors are reconstructed in the L_2 sense, but how well each reconstructed readout weight vector can still reproduce its respective time series when plugged back to the *same, fixed* ESN reservoir. To that end, we introduce a more suitable objective function for measuring the reconstruction quality of the autoencoder.

2 Echo state network cost function

An ESN is a recurrent discrete-time neural network. It processes time series composed by a sequence of observations which we denote by[1] $\boldsymbol{y} = (y(1), y(2), \ldots, y(T))$. The task of the ESN is given $y(t)$ as an input to predict $y(t+1)$. An ESN is typically formulated using the following two equations:

$$\begin{aligned} \boldsymbol{x}(t+1) &= h(\boldsymbol{u}\boldsymbol{x}(t) + \boldsymbol{v}y(t)) \ , & (1) \\ y(t+1) &= \boldsymbol{w}\boldsymbol{x}(t+1) \ , & (2) \end{aligned}$$

where $\boldsymbol{v} \in R^{N\times 1}$ is the input weight, $\boldsymbol{x}(t) = [x_1, \ldots, x_N] \in R^{N\times 1}$ are the latent state activations of the reservoir, $\boldsymbol{u} \in R^{N\times N}$ the weights of the reservoir units, $\boldsymbol{w} \in R^{N\times 1}$ the readout weights[2]. N is the number of hidden reservoir units. Function $h(\cdot)$ is a nonlinear function, e.g. tanh, applied element-wise. According to ESN methodology [5] parameters $\boldsymbol{v}$ and $\boldsymbol{u}$ in Eq. (1) are randomly generated and fixed. The only trainable parameters are the readout weights $\boldsymbol{w}$ in Eq. (2).

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 $\boldsymbol{X}$. Typically, some initial observations are dismissed in order to "washout" [5] the initial arbitrary reservoir state. The following objective function ℓ is minimised:

$$\ell(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2 + \frac{1}{2}\lambda^2\|\boldsymbol{w}\|^2 \ , \tag{3}$$

where λ is a regularisation term. How well vector $\boldsymbol{w}$ models $\boldsymbol{y}$ with respect to the fixed reservoir is measured by objective ℓ. The optimal solution is $\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda^2\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ where $\boldsymbol{I}$ is the identity matrix. We express this as a function g that maps time series to readout weights:

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

3 Vector representation for time series

Given a fixed ESN reservoir, for each time series in the dataset we determine its best readout weight vector and take it to be its new representation *with respect to this reservoir.*

3.1 ESN reservoir construction

Typically, parameters $\boldsymbol{v}$ and $\boldsymbol{u}$ in Eq. (1) are set stochastically [5]. To eliminate dependence on random initial conditions when constructing the ESN reservoir, we strictly follow the deterministic scheme[3] in [7]. Accordingly, we fix the topology of the reservoir by organising the reservoir units in a cycle using the *same*

[1]For brevity we assume univariate time series, i.e. $y(t) \in R$.

[2]Bias terms can be subsumed into weight vectors $\boldsymbol{v}$ and $\boldsymbol{u}$ but are ignored here for brevity.

[3]We stress that our algorithm is not dependent on this deterministic scheme for constructing ESNs; in fact it also works with the "standard" stochastically constructed ESN type as in [5].

coupling weight a. Similarly, all elements in $\boldsymbol{v}$ are assigned the same absolute value $b > 0$ with signs determined by an aperiodic sequence as specified in [7]. Further, following this methodology we determine values for a and b by cross-validation. The combination a, b with the lowest test error is used to instantiate the ESN reservoir that subsequently encodes the time series as readout weights.

3.2 Encoding time series as readout weights

Given the fixed reservoir, specified by a and b, we encode each time series $\boldsymbol{y}_j$ in the dataset by the readout weights $\boldsymbol{w}_j$ using function $g(\boldsymbol{y}_j) = \boldsymbol{w}_j$ (see Eq. (4)). We emphasise that *all* time series $\boldsymbol{y}_j$ are encoded with respect to the *same* fixed reservoir. Hence dataset $\{\boldsymbol{y}_1, \dots, \boldsymbol{y}_J\}$ is now replaced by $\{\boldsymbol{w}_1, \dots, \boldsymbol{w}_J\}$.

4 Autoencoding with respect to the fixed reservoir

The autoencoder [6] learns an identity mapping by training on targets identical to the inputs. Learning is restricted by the bottleneck that forces the autoencoder to reduce the dimensionality of the inputs, and hence the output is only an approximate reconstruction of the input. By setting the number of neurons in the bottleneck to two, the bottleneck activations can be interpreted as two-dimensional projection coordinates $\boldsymbol{z} \in R^2$ and used for visualisation.

The autoencoder is the composition of an encoding f_{enc} and a decoding f_{dec} function. Encoding maps inputs to coordinates, $f_{enc}(\boldsymbol{w}) = \boldsymbol{z}$, while decoding approximately maps coordinates back to inputs, $f_{dec}(\boldsymbol{z}) = \tilde{\boldsymbol{w}}$. The complete autoencoder is a function $f(\boldsymbol{w};\boldsymbol{\theta}) = f_{dec}(f_{enc}(\boldsymbol{w})) = \tilde{\boldsymbol{w}}$, where $\boldsymbol{\theta}$ are the weights of the autoencoder trained by backpropagation.

4.1 Training mode

Typically, training the autoencoder involves minimising the L^2 norm between inputs and reconstructions over the weights $\boldsymbol{\theta}$:

$$\frac{1}{2}\sum_{j=1}^{J} \|f(\boldsymbol{w}_j;\boldsymbol{\theta}) - \boldsymbol{w}_j\|^2. \tag{5}$$

However, this objective measures only how good the reconstructions $\tilde{\boldsymbol{w}}_j$ are in the L_2 sense. What we are really interested in is how well the reconstructed weights $\tilde{\boldsymbol{w}}_j$ are still a good readout weight vector when plugged back to the fixed reservoir. This is actually what the objective function ℓ in Eq. (3) measures. This calls for a modification in the objective function Eq. (5) of the autoencoder:

$$\frac{1}{2}\sum_{j=1}^{J} \ell_j(f(\boldsymbol{w}_j;\boldsymbol{\theta})) = \frac{1}{2}\sum_{j=1}^{J} \|\boldsymbol{X}_j f(\boldsymbol{w}_j;\boldsymbol{\theta}) - \boldsymbol{y}_j\|^2 + \frac{1}{2}\lambda^2\|f(\boldsymbol{w}_j;\boldsymbol{\theta})\|^2 \ , \tag{6}$$

where ℓ_j and $\boldsymbol{X}_j$ are the objective function and hidden state activations associated with time series $\boldsymbol{y}_j$ (see Eq. (3)). The weights $\boldsymbol{\theta}$ of the autoencoder can now be trained via backpropagation using the modified objective in Eq. (6).

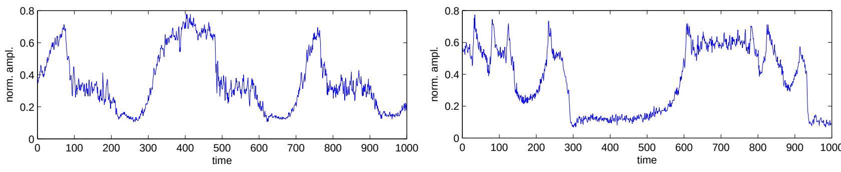


Fig. 1: Example X-ray radiation regimes β (left) and κ (right).

4.2 Testing mode

Having trained the autoencoder f via backpropagation, we would like to project new incoming time series $\boldsymbol{y}^*$ to coordinates $\boldsymbol{z}^*$. To that end we first use the fixed ESN reservoir to encode the time series as a readout weight vector $g(\boldsymbol{y}^*) = \boldsymbol{w}^*$ (see Eq. (4)). The readout weight vector $\boldsymbol{w}^*$ can then be projected using the encoding part of the autoencoder to obtain the projection $f_{enc}(\boldsymbol{w}^*) = \boldsymbol{z}^*$.

5 Experiments and Results

We present results on two synthetic datasets and on a real astronomical dataset. In all experiments we constructed the ESN reservoir deterministically according to [7] and fixed the size of the reservoir to $N = 200$. We used a washout period of 200 observations. Regularisation parameter λ for the ESNs was fixed to 10^{-4}. The number of neurons in the hidden layers of the autoencoder was set to 10. The proposed algorithm can handle out-of-sample data and hence apart from projecting training data only, we also project unseen test data. We apply no normalisation to the datasets. Moreover, we also constructed visualisations using the popular t-SNE algorithm [8] on the raw signals. We found the visualisation produced by t-SNE did not differ greatly over a range of perplexities $\{5, 10, \ldots, 50\}$.

NARMA: We generated sequences from the following NARMA classes [7] of order $10, 20, 30$, of length 800, using the following equations respectively:

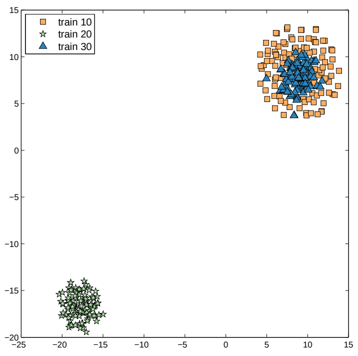
$$y(t+1) = 0.3y(t) + 0.05y(t)\sum_{i=0}^{9} y(t-i) + 1.5s(t-9)s(t) + 0.1,$$

$$y(t+1) = \tanh(0.3y(t) + 0.05y(t)\sum_{i=0}^{19} y(t-i) + 1.5s(t-19)s(t) + 0.01) + 0.2,$$

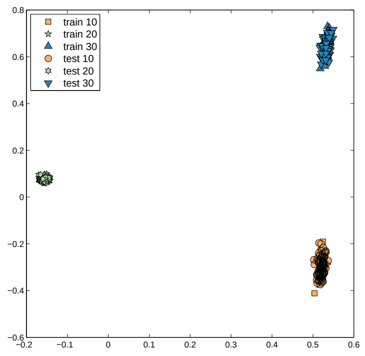
$$y(t+1) = 0.2y(t) + 0.004s(t)\sum_{i=0}^{29} y(t-i) + 1.5s(t-29)s(t) + 0.201,$$

where $s(t)$ are exogenous inputs generated independently and uniformly in the interval $[0, 0.5)$. These time series constitute an interesting synthetic example due to the long-term dependencies they exhibit.

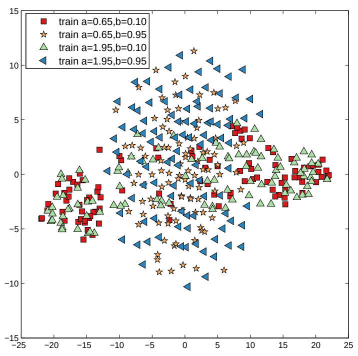
ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.



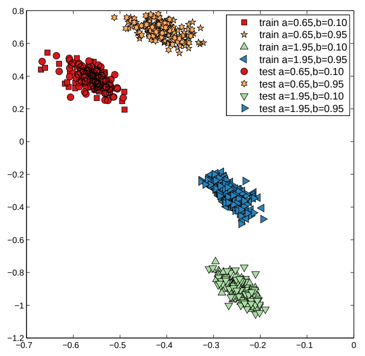
(a) NARMA by t-SNE.



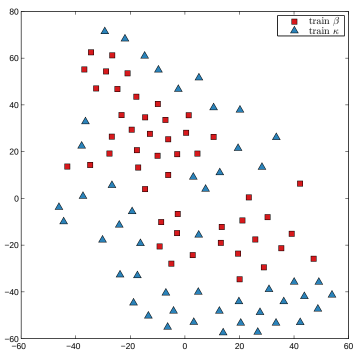
(b) NARMA by our method.



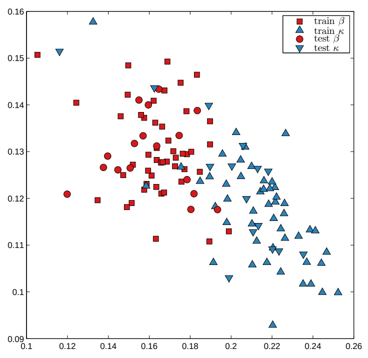
(c) Cauchy by t-SNE.



(d) Cauchy by our method.



(e) X-ray radiation by t-SNE.



(f) X-ray radiation by our method.

Fig. 2: Colours represent classes. The proposed algorithm supports out-of-sample visualisation, hence markers • and ▲ are the projections of the training and testing data respectively. Note that in the NARMA and Cauchy plots • and ▲ heavily overlap.

Cauchy class: We sampled sequences from a stationary Gaussian process with correlation function given by $c(x_t, x_{t+h}) = (1 + |h|^a)^{-\frac{a}{b}}$ [9]. We generated 4 classes of such time series by the permutation of parameters $a \in \{0.65, 1.95\}$ and $b \in \{0.1, 0.95\}$. We generated from each class 100 time series of length 2000.
X-ray radiation from black hole binary: We used data from [10] concerning a black hole binary system that expresses various types of temporal regimes which vary over a wide range of time scales. We extracted subsequences of length 1000 from regimes β and κ that were chosen on account of their similarity (see Fig. 1).

6 Discussion and Conclusion

We show the visualisations in Fig. 2. Unlike t-SNE which operates directly on the raw data, the proposed algorithm can capture the differences between the time series in the lower dimensional space. This is because our method explicitly accounts for the sequential nature of time-series; learning is performed in the space of readout weight representations and is guided by an objective function that quantifies the reconstruction error in a principled manner. Of course, the perfectly capable t-SNE is used here as a mere candidate from the class of algorithms designed to visualise vectorial data in order to demonstrate this issue. Moreover, we demonstrate that our method, by its very nature, is capable of projecting also unseen hold-out data. Future work will focus on processing large datasets of astronomical light curves.

References

[1] J. W. Richards, D. L. Starr, N. R. Butler, J. S. Bloom, J. M. Brewer, A. Crellin-Quick, J. Higgins, R. Kennedy, and M. Rischard. On machine-learned classification of variable stars with sparse and noisy time-series data. *The Astrophysical Journal*, 733(1):10, 2011.

[2] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, pages 487–493. The MIT Press, 1998.

[3] T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004.

[4] H. Chen, F. Tang, P. Tino, and X. Yao. Model-based kernel for efficient time series analysis. In *KDD*, pages 392–400, 2013.

[5] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical report, German National Research Center for Information Technology, 2001.

[6] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AICHE Journal*, 37:233–243, 1991.

[7] A. Rodan and P. Tino. Minimum complexity echo state network. *IEEE Transactions on Neural Networks*, 22(1):131–144, 2011.

[8] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[9] T. Gneiting and M. Schlather. Stochastic models that separate fractal dimension and the hurst effect. *SIAM Review*, 46(2):269–282, 2004.

[10] K. P. Harikrishnan, R. Misra, and G. Ambika. Nonlinear time series analysis of the light curves from the black hole system grs1915+105. *Research in Astronomy and Astrophysics*, 11(1), 2011.