Neural Learning of Chaotic Dynamics: The Error Propagation Algorithm

Rembrandt Bakker, Jaap C. Schouten, Cor M. van den Bleek
Delft University of Technology,
Department of Chemical Process Technology
Julianalaan 136, 2628 BL Delft, The Netherlands
r.bakker@stm.tudelft.nl

C. Lee Giles[1,2]
NEC Research Institute
4 Independence Way
Princeton, NJ 08540
giles@research.nj.nec.com

Technical Report
UMIACS-TR-97-77 and CS-TR-3843
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

Abstract

An algorithm is introduced that trains a neural network to identify chaotic dynamics from a single measured time-series. The algorithm has four special features:

1. The state of the system is extracted from the time-series using delays, followed by weighted Principal Component Analysis (PCA) data reduction.
2. The prediction model consists of both a linear model and a Multi-Layer-Perceptron (MLP).
3. The effective prediction horizon during training is user-adjustable, due to ‘error propagation’: prediction errors are partially propagated to the next time step.
4. A criterion is monitored during training to select the model that has a chaotic attractor most similar to the real system’s attractor.

The algorithm is applied to laser data from the Santa Fe time-series competition (set A). The resulting model is not only useful for short-term predictions but it also generates time-series with similar chaotic characteristics as the measured data.

Keywords — time series, neural networks, chaotic dynamics, laser data, Santa Fe time series competition, Lyapunov exponents, principal component analysis, error propagation.

1. Introduction

A time-series measured from a deterministic chaotic system has the appealing characteristic that its evolution is fully determined and yet its predictability is limited, due to exponential growth of errors in model or measurements. A variety of data-driven analysis methods for this type of time-series was collected in 1991 during the Santa Fe time-series competition [1]. The methods focused either on characterization or prediction of the time-series. No attention was given to a third, much more powerful objective: given the data and the assumption that it was produced by a deterministic system, find a set of model equations that will produce a time-series with identical chaotic characteristics, having the

---

[1] http://www.neci.nj.com/homepages/giles.html

[2] Also with the Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742

same chaotic attractor. The model could be based on first principles if the system is well understood, but here we assume knowledge of just the time-series and use a neural network based, black-box, model. In concise neural network jargon we formulate our goal: train a network to learn the chaotic attractor. A number of authors addressed this issue [2-6]. The common approach is to identify a prediction model, generate a time-series with it and investigate its characteristics. Principe et al. [2] found that in many cases this approach fails; learning of the attractor is not guaranteed and remains a matter of trial and error. To improve the common approach, Kuo and Principe [3] proposed to minimize multi-step instead of just one-step-ahead prediction errors. We proceed in this line of thought by introducing an algorithm that effectively increases the prediction horizon of the model during training; however, it differs in that it does not compute multi-step-ahead predictions but, more efficiently, propagates previous prediction errors. The algorithm is explained in section 4.1. It is implemented along with three other features that make the modeling more effective: the reduction of the number of model inputs by weighted PCA (section 2); use of a combined linear model and MLP (section 3); and monitoring during training of a criterion to compare model and measured attractor (section 4.2). Section 5 describes how the algorithm was successfully applied to laser data (set A) from the Santa Fe competition. The paper ends with concluding remarks and directions for future research (section 6).

2. Method of delays and PCA

The discrete time evolution equation for a nonlinear, autonomous, deterministic and stationary system is

$$\vec{x}_{n+1} = F(\vec{x}_n), \tag{1}$$

where F is a nonlinear function, and $\vec{x}_n$ is the state of the system at time $t_n = t_0 + n\tau$, τ is the sampling time. In most real applications only a single variable is measured, even if the evolution of the system depends on several variables. In this case, we extract the state from the time-series of the single measured variable y by taking a delay vector $\vec{x}_n = (y_{n-m+1}, \ldots, y_n)$, where m is the number of delays, called embedding dimension. Takens [7] showed that this *method of delays* will lead to evolutions of the type of Eq. (1) if we choose $m \geq 2D + 1$, where D is the dimension of the system's attractor.

The choice of the delay time τ and the embedding m is highly empirical. First we fix their product $T = \tau m$, the time window, and consider that: (i) it is too short if the variation within the window is dominated by noise; and (ii) it is longer than necessary if the first and last part of the delay vector have no correlation at all. Next we choose the delay time τ. Here we must realize that τ is not only the time interval between successive delays but it will also be the *one-step-ahead prediction time* of our model. That time must be small to enable generation of smooth time series and to make sure that the mapping F in Eq. (1) will be smooth.

2.1. Principal Component embedding

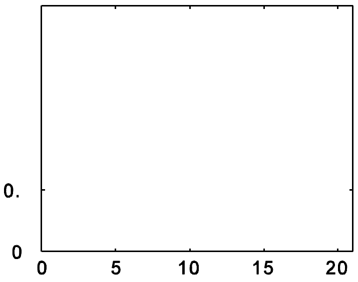
The consequence of choosing a small delay time in a fixed time window is a large embedding, $m = T/\tau$, causing the model to have an excess number of inputs that are highly correlated. A convenient way to reduce the dimension of a set of linearly correlated variables is Principal Component Analysis (PCA), as introduced in this context by Broomhead and King [8]. PCA transforms the set of delay vectors $\vec{y}_n$ to a new set of uncorrelated variables of reduced dimension. The transformation matrix $\boldsymbol{U}$ is computed by singular value decomposition (SVD) of the covariance matrix $\boldsymbol{S}$ of the input data. SVD decomposes $\boldsymbol{S}$ into the product of three matrices $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are both orthogonal and Σ is diagonal. The principal component vectors $\vec{z}_n$ are computed by

$$\vec{z}_n \approx \boldsymbol{U}^T \vec{y}_n, \tag{2}$$

and the original delay vectors $\vec{y}_n$ are reconstructed by

$$\vec{y}_n \approx \boldsymbol{U}\vec{z}_n. \tag{3}$$

The accuracy of the reconstruction in Eq. (3) depends on how small we choose the reduced dimension of $\vec{z}_n$. Intuitively, one would like the reconstruction of recent measurements to be more accurate than that of older ones. This can be done by using *weighted PCA* [9]. In our experience, the standard weighting profile shown in Fig. 1 works well. The curve is computed $w(d) = \exp\left(-\frac{d-1}{m-1}\right)$, where $w(d)$ is the weight for the d^{th} delay. The weight of the most recent delay is doubled; without weighting it tends to be less well reconstructed because it has only one neighbor in the delay vector.


0.
0
0
5
10
15
20

```
ERROR: stackunderflow
OFFENDING COMMAND: rlineto

STACK:

3
```