Neural Networks 56 (2014) 10–21

Contents lists available at ScienceDirect

Neural Networks

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

Relative entropy minimizing noisy non-linear neural network to approximate stochastic processes

Mathieu N. Galtier [a,*], Camille Marini [b,c], Gilles Wainrib [d], Herbert Jaeger [a]

[a] School of Engineering and Science, Jacobs University Bremen gGmbH, 28759 Bremen, Germany
[b] Institut für Meereskunde, Zentrum für Meeres- und Klimaforschung, Universität Hamburg, Hamburg, Germany
[c] MINES ParisTech, 1, rue Claude Daunesse, F-06904 Sophia Antipolis Cedex, France
[d] Laboratoire Analyse Géométrie et Applications, Université Paris XIII, France

ARTICLE INFO

Article history:
Received 10 December 2013
Received in revised form 15 April 2014
Accepted 18 April 2014
Available online 26 April 2014

Keywords:
Echo state networks
Linear inverse modeling
Relative entropy
Stochastic processes
El Nino southern oscillation

ABSTRACT

A method is provided for designing and training noise-driven recurrent neural networks as models of stochastic processes. The method unifies and generalizes two known separate modeling approaches, Echo State Networks (ESN) and Linear Inverse Modeling (LIM), under the common principle of relative entropy minimization. The power of the new method is demonstrated on a stochastic approximation of the El Niño phenomenon studied in climate research.

1. Introduction

Black-box modeling methods for stochastic systems have a broad range of applications in physics, biology, economy or the social sciences. Generally speaking, a model of a stochastic system is a representation of the conditional distribution of the system's future given the present state (Markov models) or some part or the entire system past. There is a large variety of such stochastic predictors among which we focus on generic methods which do not depend on the type of data considered. The Auto-Regressive–Moving-Average (ARMA) models (Box, Jenkins, & Reinsel, 2013) form a class of linear stochastic approximators which has led to many derivative works and is widely used in engineering applications. In particular, it covers the case of multivariate linear stochastic differential equations (SDE), or Ornstein–Uhlenbeck processes, which is the basic structure used in the Linear Inverse Modeling (LIM) theory (Penland & Magorian, 1993). ARMA models are generally learnt by optimizing a least squares measure of the prediction error. A notable characteristic of ARMA models is that the dimension of the underlying SDE is identical to the observable dimension of the target time series. By contrast, dynamic Bayesian networks (Murphy, 2002), with Hidden Markov Models (HMM) (Baum & Petrie, 1966; Rabiner, 1989) as the most widely employed special case, rely on hidden variables. HMM are trained by maximum likelihood schemes, typically with some version of the expectation maximization algorithm (Dempster, Laird, & Rubin, 1977; Moon, 1996). A problem with dynamic Bayesian networks, inherited from their simpler static counterparts, is that inference (e.g. prediction) quickly becomes computationally expensive when the dependency structure of hidden variables is not particularly simple (as it is in HMMs). The Temporal Restricted Boltzmann Machine (Sutskever & Hinton, 2006), a recent addition to the spectrum of such models, is a point in case. With the advent of kernel machines in machine learning community, models based on Gaussian Processes have been designed to approximate stochastic processes (Rasmussen, 2006). A critical point regarding these models lies in their computational complexity when working with long time series. There is also a large body of literature about online adaptive predictors, e.g. Kalman filters (Haykin, 2005). In this paper however we focus on non-adaptive models trained on all available training data using a batch algorithm.

Recurrent neural networks (RNNs) have also been used in various ways for approximating stochastic dynamical systems. In their basic forms (Pearlmutter, 1995; Williams & Zipser, 1995), RNNs are models of deterministic dynamical systems; if trained on data sampled from stochastic sources, at exploitation time

* Correspondence to: CR INRIA, EPI NeuroMathComp, Sophia-Antipolis Mditerranee 2004, route des lucioles, 06902 Sophia-Antipolis Cedex, France. Tel.: +33 6 38 044 099.
E-mail address: mathieu.galtier@inria.fr (M.N. Galtier).

http://dx.doi.org/10.1016/j.neunet.2014.04.002
0893-6080/

such RNNs will not propose future distributions but only a single expected mean future trajectory. RNNs represent, in principle, a promising model class because they are dense in interesting classes of target systems, implying that arbitrarily accurate models can in principle be found (Funahashi & Nakamura, 1993; Sontag, 1997). Gradient-descent based learning algorithms for RNNs are typically computationally expensive and cannot be guaranteed to converge. Since about a decade, an alternative approach to RNN design and training, now generally called *reservoir computing* (Jaeger & Haas, 2004; Maass, Natschläger, & Markram, 2002), has overcome the problem of learning complexity. The key idea in this field is not to train all parameters of an RNN but only the weights of connections leading from the RNN "body" (called *reservoir*) to the output neurons. Here we will build on a particular instantiation of reservoir computing called *Echo State Networks* (ESNs).

Although deterministic models at the outset, neural network architectures for predicting future distributions have been variously proposed (Buesing, Bill, Nessler, & Maass, 2011; Husmaier & Taylor, 1997), or neural networks were embedded as components in hybrid models of stochastic systems (Chatzis & Demiris, 2011; Krogh & Riis, 1999). Here we propose a novel way to use RNNs in a stochastic framework based on the way stochasticity is taken into account in LIM. LIM consists in tuning both the drift and the diffusion term of an Ornstein–Uhlenbeck process to approximate a stochastic process. First, the drift is optimized to approximate the time series as if it were deterministic; then, the diffusion is chosen so that the variances of both systems are identical. LIM is widely used in climate research and stands as a simple approach giving relatively good results (Barnston, Tippett, L'Heureux, Li, & DeWitt, 2012; Hawkins, Robson, Sutton, Smith, & Keenlyside, 2011; Newman, 2013; Penland, 1996; Zanna, 2012).

To compare two stochastic processes, and thus to define what it means to approximate a stochastic process, we use the relative entropy (also known as Kullback–Leibler divergence) (Kullback & Leibler, 1951). Although not a true distance, it displays many interesting properties, interpretations and relationships with other quantities such as the mutual information (Cover & Thomas, 2012) or the rate function in large deviations theory (Ellis, 2005). It also is computationally convenient (as opposed to the Wasserstein distance for instance), and has been widely used in machine learning (Ackley, Hinton, & Sejnowski, 1985; Hinton, Osindero, & Teh, 2006). Usually, this measure is used to compare the laws of two discrete or continuous random variables, but it can also be used to compare the laws of two stochastic processes in the path space, which is at the basis of this paper. This way of measuring the difference in law between two stochastic processes amounts in performing a change of probabilities thanks to Girsanov Theorem (Karatzas & Shreve, 1991), whose applications range from mathematical finance (Avellaneda, Friedman, Holmes, & Samperi, 1997) to simulation methods for rare events (Wainrib, 2013). In the context of recurrent neural networks, we have already shown that the learning rule deriving from the minimization of relative entropy has interesting biological features since it combines two biologically plausible learning mechanisms (Galtier & Wainrib, 2013).

In this paper, we show how to train a noise-driven RNN to minimize its relative entropy with respect to a target process. The method consists two steps. First, the drift of the neural network is trained by minimizing its relative entropy with respect to the target (Section 3). Second, the noise matrix of the network is determined based on a conservation principle similarly to LIM (Section 4). We show how this approach extends the existing ESN and LIM theory in Section 5. Numerical approximations to the double well potential and to the El Niño phenomenon studied in climate research are presented in Section 6.

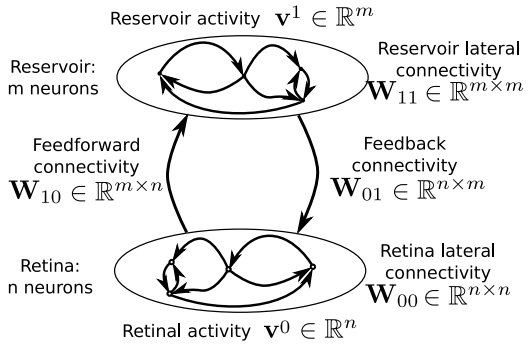

Fig. 1. Structure and main notations of the neural network described in Section 2.

2. Model

We define here two mathematical objects that are, a priori, unrelated: a stochastic time series and an autonomous RNN made of two layers. The time series is assumed to be a sample path of an underlying stochastic process which is the modeling target. The objective is to make the RNN approximate the target process.

The target time series $\mathbf{u}$ is assumed to be the discretization of an n-dimensional ergodic continuous process defined on the time interval $[0, T]$. The discretization step is chosen to be $dt = 1$ which corresponds to fixing the timescale. Imposing $T \in \mathbb{N}$, $\mathbf{u}$ can be seen as a matrix in $\mathbb{R}^{n\times T}$. For each $t \in \{1, \ldots, T\}$, we use the notation $\mathbf{u}_t$ for the n-dimensional vector corresponding to the value of the continuous target time series at time t. Similarly, we write $\delta\mathbf{u}_t \overset{def}{=} \mathbf{u}_{t+1} - \mathbf{u}_t$ and $\delta\mathbf{u} \in \mathbb{R}^{n\times T}$ corresponding to the previous definition (with the convention that $\delta\mathbf{u}_T = 0$).

The two-layer neural network is defined as follows. The first layer, also called retina, has n neurons, as many as the target time series dimension. We take $\mathbf{v}_t^0 \in \mathbb{R}^n$ to be the activity of the retina at time t which will eventually approximate the target time series. The second layer, also called reservoir, has m neurons. Because each reservoir neuron does not directly correspond to a variable of the target, they are said to be hidden neurons. We denote the activity of the reservoir at time t by $\mathbf{v}_t^1 \in \mathbb{R}^m$. Each layer has a complete internal connectivity, recurrent connections, that is, all neurons within a layer are interconnected. The two layers are interconnected with feedforward, i.e. retina to reservoir, connections and feedback, i.e. reservoir to retina, connections, as shown in Fig. 1(a). In this paper, according to a guiding principle in reservoir computing (Lukoševičius & Jaeger, 2009), the feedforward and reservoir matrices $\mathbf{W}_{10}$ and $\mathbf{W}_{11}$ are drawn randomly and remain unchanged. Only connections leading to retina neurons will be adapted. These are collected in $\mathbf{W} = (\mathbf{W}_{00}\ \mathbf{W}_{01}) \in \mathbb{R}^{n\times(n+m)}$.

The activity of each layer is governed by the following differential law:

$$\begin{cases} d\mathbf{v}_t^0 = \left(-\mathbf{v}_t^0 + \mathbf{W}_{00}\mathbf{v}_t^0 + \mathbf{W}_{01}\mathbf{v}_t^1\right)dt + \Sigma dB_t \\ d\mathbf{v}_t^1 = \epsilon\left(-l\mathbf{v}_t^1 + s(\mathbf{W}_{10}\mathbf{v}_t^0 + \mathbf{W}_{11}\mathbf{v}_t^1)\right)dt \end{cases} \tag{1}$$

where $\epsilon, l \in \mathbb{R}_+$, s is a sigmoid function, e.g. tanh, that is applied elementwise, i.e. $s(\mathbf{x})_i = s(\mathbf{x}_i)$, $\Sigma \in \mathbb{R}^{n\times n}$ is the noise matrix and B_t is an n-dimensional Brownian motion.

In order to unify LIM and ESNs, we submit this architecture to certain restrictions. In particular, we choose the first layer to be linear and we choose a tanh non-linearity in the reservoir. Later, we will make a simple choice for a numerical differentiation scheme for the same reason.