Neural Networks 50 (2014) 120–123

Contents lists available at ScienceDirect

Neural Networks

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

Cellular computational networks—A scalable architecture for learning the dynamics of large networked systems

Bipul Luitel, Ganesh Kumar Venayagamoorthy *

Real-Time Power and Intelligent Systems Laboratory, Holcombe Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634, USA

ARTICLE INFO

Article history:
Received 21 May 2013
Received in revised form 14 September 2013
Accepted 6 November 2013

Keywords:
CCN
DRN
Electric power network
Large networked systems
Scalability

ABSTRACT

Neural networks for implementing large networked systems such as smart electric power grids consist of multiple inputs and outputs. Many outputs lead to a greater number of parameters to be adapted. Each additional variable increases the dimensionality of the problem and hence learning becomes a challenge. Cellular computational networks (CCNs) are a class of sparsely connected dynamic recurrent networks (DRNs). By proper selection of a set of input elements for each output variable in a given application, a DRN can be modified into a CCN which significantly reduces the complexity of the neural network and allows use of simple training methods for independent learning in each cell thus making it scalable. This article demonstrates this concept of developing a CCN using dimensionality reduction in a DRN for scalability and better performance. The concept has been analytically explained and empirically verified through application.

1. Introduction

In many networked systems, each component's behavior is directly affected by the behavior of other components in its neighborhood and indirectly by other components that are connected to the network far away from its neighborhood. The neighborhood can be defined as the proximity with respect to certain parameters associated with the component and the system. Examples of such systems, collectively referred to as "large networked systems", are smart electric grids, transportation networks, water distribution networks, communication networks, and sensor networks (surveillance, monitoring, etc.). Implementation of such systems using a DRN requires a learning method to adapt the sets of weights such that all the functions are simultaneously approximated using the same set of weights. Using a gradient descent based learning method such as backpropagation, weights in a multi-layered neural network are adapted based on the gradient of the output error with respect to the weights. Therefore, the sizes of the input and output weight matrices become larger as the number of output variables, and their corresponding inputs, increases.

In most real world problems, however, each output variable is only dependent on a subset of input variables. Suppose $\vec{O} = [O_1, O_2, \ldots, O_i, \ldots, O_N]$ is an output vector of N output variables where each output is a function of inputs, i.e. $\vec{O} = f\left(\vec{I}\right)$ where $\vec{I}$ is a vector of input elements consisting of current and past states as well as current and past controls (Narendra & Mukhopadhyay, 1997) and has a size of $M \geq N$. Implementation of large networked systems using DRNs is computationally intensive and training challenging. In many practical systems, each output O_i of $\vec{O}$ is a function of input elements $\vec{F}_i \subset \vec{I}$. The input element vector $\vec{F}$ consists of m elements such that $m \ll M$ as N becomes large. Therefore, implementing with smaller independent neural networks is more efficient, and training is easier and more accurate. The drawback, however, is that connectivity of the components and hence the associated dynamics may be lost.

Dynamic neural networks are necessary to learn the spatial and temporal dynamics of complex nonlinear systems in real life (Gupta, Jin, & Homma, 2003). Recurrence provides dynamic neural networks with the memory necessary to store the spatio-temporal data and map inputs to the outputs (Kolen & Kremer, 2001). Cellular neural networks (CNNs) (Chua & Yang, 1988) consist of neurons, called cells, having local connection only to their neighbors. In Werbos and Pang (1996) and Wunsch (2000), cellular networks are presented in which each cell is a neural network, and these are referred to as CNNs. Cellular computational networks (CCNs) are dynamic recurrent networks (DRNs) consisting of a computational element (neural network or otherwise) in each cell, and can be used to implement large networked systems. In this article, the CCN, a scalable neural network architecture that exploits on the property of complex networked systems, is described. It is shown later in the article that a CCN

* Corresponding author. Tel.: +1 864 656 5936.
E-mail addresses: iambipul@ieee.org (B. Luitel), ganeshv@mst.edu, gkumar@ieee.org (G.K. Venayagamoorthy).

0893-6080/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.neunet.2013.11.001

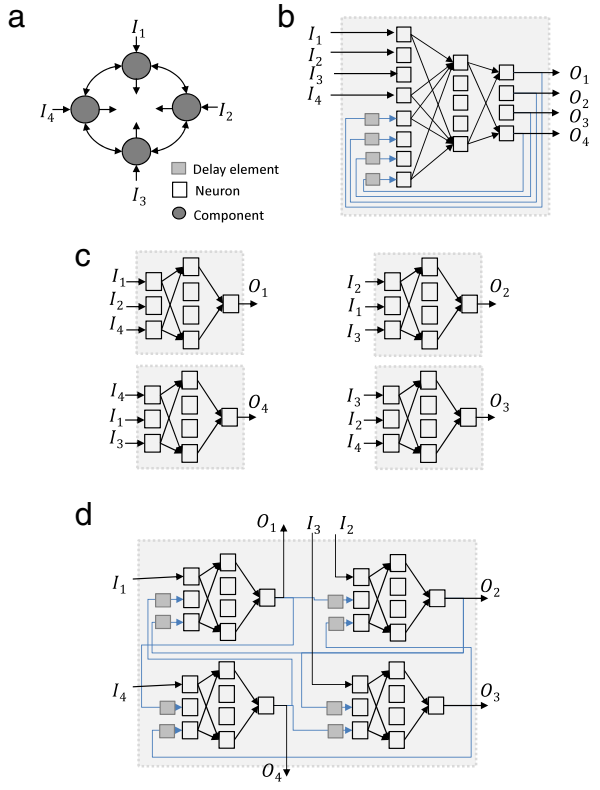

Fig. 1. (a) A system of four interconnected components. (b) A TLRN for learning the dynamics of the system. (c) Equivalent representation using four MLPs, one for each component. (d) Equivalent representation using a CCN.

consisting of a NN in each cell represents a sparsely connected DRN. The development of a CCN for an electric power network is described in this article and is compared with other neural network architectures. The obtained result shows the CCN as a promising architecture for learning the dynamics of large networked systems.

2. CCN as a sparsely connected DRN

Consider an input $\vec{I}$ of four elements $[I_1, I_2, I_3, I_4]$ corresponding to four components of a networked system, as shown in Fig. 1(a). The output $\vec{O} = [O_1, O_2, O_3, O_4]$ is to be mapped to the input $\vec{I}$ using a neural network. Fig. 1(b) shows an implementation using a time lagged recurrent neural network (TLRN) (Elman, 1990) having a context as a feedback from the outputs. For system identification of a component, the variables that affect its performance are used as inputs. This involves selection of a subset of input elements from a set of input data. Assuming that each component of the system is affected by its nearest neighbors, the behavior of each component can be modeled independently using a multi-layer perceptron (MLP) as shown in Fig. 1(c) using common inputs between neighbors. Finally, the same system is modeled using a CCN as shown in Fig. 1(d) where delayed outputs from the neighbors are used as the inputs to each cell. Fig. 1(c) and (d) are almost identical in terms of the architecture (number of neurons used in different layers). However, Fig. 1(d) can be re-arranged as shown in Fig. 2(a) and is functionally equivalent to a sparsely connected TLRN. Fig. 2(b) shows that setting the weights associated with some feedback elements to zero (as shown by the dashed lines) and adding additional connections between the neurons (as shown by the dotted connections) to the CCN, it will be equivalent to a TLRN. This gives the CCN the power of a DRN with the simple architecture of a static MLP.

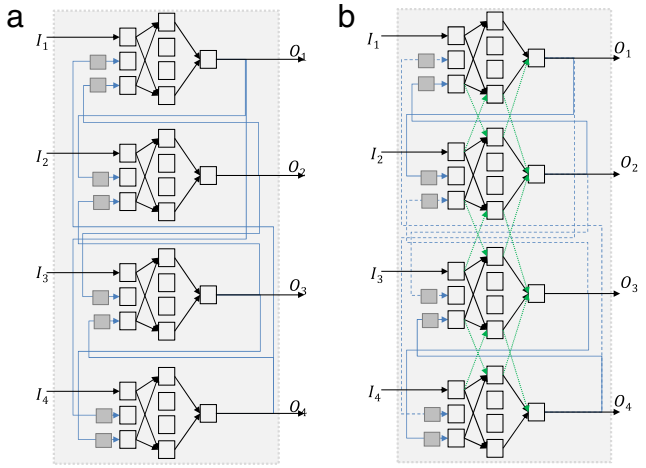

Fig. 2. (a) Re-arrangement of the cells of a CCN for comparing it against a DRN. (b) Addition of synaptic weights between the neurons in the input and the hidden layers across different cells will render the CCN equivalent to a DRN.

It can be inferred from the above observation that CCNs are DRNs with sparse synaptic weights between the neurons in different layers. A DRN can be described in vector notation by the following equation:

$$\vec{O}(k) = g\left(\vec{W}_{in} \times \vec{I}(k) + \vec{W}_c \times \vec{O}(k-1)\right) \times \vec{W}_o \tag{1}$$

where $\vec{W}_{in}$ and $\vec{W}_c$ represent the synaptic weights from the input and context layer neurons respectively to the hidden layer neurons, and $\vec{W}_o$ represents the set of weights from the hidden layer neurons to the output neurons. The symbol '$\times$' represents a cross product and 'g' represents the activation function in the hidden neurons. Based on the above discussion, the equation for the CCN of Fig. 2(a) can be written as

$$O_i = g_i\left((\vec{W}_{in})_i \times \vec{F}_i(k)\right) \times (\vec{W}_o)_i \tag{2}$$

where $\vec{F}_i(k) = [\vec{I}_i(k), \vec{O}_{i-1}(k-1), \vec{O}_{i+1}(k-1)]$ is the input to each cell and $\vec{O}_{i-1}(k-1)$ and $\vec{O}_{i+1}(k-1)$ are the time lagged outputs of connected neighbors used as inputs and provide the recurrence to the overall structure of a CCN. The sets of input and output weights, $(\vec{W}_{in})_i$ and $(\vec{W}_o)_i$ respectively, are synaptic weights between the fully connected neurons of each cell and are each equivalent to the subset of DRN weights $\vec{W}_{in}$ and $\vec{W}_o$. Each cell is independently trained using backpropagation. Because of only one output and a smaller number of weights in each cell, learning is more accurate.

A very large system consists of many multi-dimensional variables. It is very hard to develop a neural network solution for such a system using a single DRN that represents all the variables, or with individual DRNs that represent each variable in one dimension, or by reservoir computing using different sets of readout weights for each variable in each dimension. Such solutions do not scale up for large systems and it is hard to train such networks without sacrificing accuracy and/or speed. By forming a cellular structure taking advantage of the common input elements between different variables in a given system, a multi-layered (multi-dimensional) CCN can be developed where different cells in each 'layer' represent the dimensions of one variable and different layers represent the different variables. The cells are connected to each other within and across the layers based on the common input elements necessary for system identification of