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

Abstract

Neural networks for implementing large networked systems such as the smart electric power grid consist of multiple inputs and outputs. Many outputs lead to more 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. The letter demonstrates this concept of developing CCN using dimensionality reduction in a DRN for scalability and better performance. The concept has been analytically explained and empirically verified through application.

Keywords: CCN, DRN, Electric Power Network, Large Networked Systems, Scalability

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. Neighborhood can be defined as proximity with respect to certain parameter associated with the component and the system. Examples of such systems, collectively referred to as “large networked systems”, are smart electric grid, transportation network, water distribution network, communication network, 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 decent 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 bigger as the number of output variables, and its 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. \( \vec{O} = f(\vec{I}) \) where \( \vec{I} \) is a vector of input elements consisting of current and past states as well as current and past controls (Narendra and 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 \( \vec{F}_i \) 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 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 et al., 2003). Recurrence provides dynamic neural networks with the memory necessary to store the spatio-temporal data and map inputs to the outputs (Kolen and Kremer, 2001). Cellular neural networks (CNNs) (Chua and Yang, 1988) consist of neurons, called cells, having local connection only to its neighbors. In (Werbos and Pang, 1996; Wunsch, 2000), cellular networks are presented in which each cell is neural networks, and are referred to as CNNs. Cellular computational networks (CCNs) are dynamic recurrent networks (DRNs) consisting of a computational element (neural networks or otherwise) in each cell, and can be used to implement large networked systems. In this letter, CCN, a scalable neural network architecture that exploits on the property of complex networked systems, is described. It is shown later
in the letter that a CCN consisting of a NN in each cell represents a sparsely connected DRN. The development of a CCN for electric power network is described in this letter and is compared with other neural network architectures. The obtained result shows 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 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 multilayer 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. Figs. 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 CCN the power of DRN with the simple architecture of a static MLP.

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:

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

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. Symbol ‘$\times$’ represents cross product and ‘$g$’ represents the activation function in the hidden neurons. Based on above discussion, equation for the CCN of Fig. 2(a) can be written as:

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

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 fewer 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

Figure 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, and (d) Equivalent representation using a CCN.
systems and it is hard to train such networks without sacrificing accuracy and/or speed. By forming a cellular structure taking advantages 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 each variable. Consider a large system consisting of $D$ variables and $N$ dimensions. It can be represented by a CCN of $D$ layers with $N$ cells in each layer as:

$$O_i^j = f\left((\bar{F}_i^j \subset \bar{I}) \cup \bar{C}_i^j\right), 1 \leq i \leq N \text{ and } 1 \leq j \leq D$$  

(3)

$$O_i^j(k) = g_i^k\left((\bar{F}_i^j(k) \subset \bar{I}^j) \times (\bar{W}_{in})_{i}^j \times (\bar{W}_o)_{i}^j\right)$$  

(4)

where, $\bar{I}$ is the set of input elements associated with all the dimensions of each variable and has at least $(D \times N)$ elements, $\bar{F}_i^j(k)$ is the vector of input elements specific to the cell and $\bar{C}_i^j(k) \subset \bar{O}(k - 1)$ is the feedback from the connected cells that provides the context to the DRN. The vector $\bar{O} \supset O_i^j$ is the set of outputs from each cell and consists of exactly $(D \times N)$ elements. This ability to ‘cluster’ a smaller subset of input elements to form cells with single output makes CCN linearly scalable and inherently parallel. Such small cells can be easily implemented using MLPs and trained simultaneously on a parallel computing environment. However, the designer needs to know the network architecture, and the different input variables that influence a particular output variable in order to take advantage of the sparsity and locality.

3. Case Study and Results

The case studied is a reduced order equivalent interconnected model of the New England test system /New York Power system consisting of 16 generators and 68 buses (Pal and Chaudhuri, 2005). The case is simulated on a real-time digital simulator and training data obtained by applying forced perturbation to the system. An MLP, a TLRN and a CCN are each being used to predict the speed deviation of the generators ($\Delta\omega(k + 1)$) in this multemachine power network. The inputs to the MLP are the speed deviation of the generators in the current sample ($\Delta\omega(k)$) and the change in reference voltage applied as the input to the generators ($\Delta V_{ref}(k)$). In addition to these inputs, the TLRN also consists of time-delayed feedback from the outputs ($\Delta\omega(k)$). In case of a CCN, each generator is represented using one cell made up of MLP. The cells are connected to each other based on the nearest neighboring generators that influence the performance of each generator (Fig. 3), which also determines the number of inputs in each cell. Apart from speed deviation and change in reference voltage of the current sample, each cell receives as inputs the outputs of the nearest neighboring cells in the previous sample. The neighboring cells are determined based on the electrical distance between the generators in the physical system that influence each other. In all the neural network implementations, the number of hidden neurons is considered to be equal to the sum of the neurons in the input and the output layers for uniform comparison among different architectures. Linear activation functions are used in the neurons of input and the output layers, and sigmoids in the hidden layer.

The average mean squared errors (MSE) obtained in 25 trials during testing are shown in Table 1. In order to quantify the two-fold improvements in performance, a performance index (PI) is calculated based on the MSE and the number of weights (W) using (5).

$$PI_i = \frac{\max(MSE \times W)}{MSE_i \times W_i}$$  

(5)

The plot of absolute testing errors of all the cells are shown in Fig. 4. The results (MSE) show that the performance of CCN is better than those obtained using MLP and TLRN. The CCN obtains 16 times better performance with 16% less weights than TLRN. Since the size of each cell is constant, the number of weights in CCN increases linearly as opposed to quadratically in case of MLP and TLRN. This reduction will be even more significant when projected for a very large system with hundreds of variables in multiple dimensions. Ability of CCN to disintegrate a large complex problem into multiple simpler problems makes it highly scalable and applicable in a distributed environment with multi-dimensional multiple input/output
Figure 3: Development of a CCN based on the network topology of an electric power network having 16 generators. Each cell representing a generator is implemented using MLP. The electric power network is shown on the background.

Figure 4: Absolute error between the actual speed deviation and the predicted outputs of 16 cells obtained during testing using (a) MLP, (b) TLRN and (c) CCN.

parameters. It should be noted that there is a fair amount of assumption in the use of numbers of neurons in hidden layer, learning and momentum gains and training epochs in the determination of the presented results and hence are not claimed as optimal.

4. Conclusion

Cellular computational networks are shown to be a massively parallel and highly scalable dynamic recurrent network architectures. Performance index shows CCN as a better architecture in terms of accuracy and consistency for learning the dynamics of large networked systems. However, there are several research issues that need to be addressed. Input selection and connectivity is determined by application. Implementation of the proposed method on other networked systems is a potential research area.

Table 1: Comparison of average MSEs obtained using different NN architectures.

<table>
<thead>
<tr>
<th>Epochs</th>
<th>Average MSE (×10^-8)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MLP</td>
</tr>
<tr>
<td>MSE</td>
<td>45.94±27.93</td>
</tr>
<tr>
<td>W</td>
<td>2304</td>
</tr>
<tr>
<td>PI</td>
<td>1.05</td>
</tr>
</tbody>
</table>

Acknowledgment

The funding provided by the National Science Foundation, USA under the CAREER grant ECCS #1231820, #1232070 and EFRI #1238097 is gratefully acknowledged.

References