Neural Networks 79 (2016) 12–19

Contents lists available at ScienceDirect

Neural Networks

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

When are two multi-layer cellular neural networks the same?

Jung-Chao Ban[a], Chih-Hung Chang[b,*]

[a] Department of Applied Mathematics, National Dong Hwa University, Hualien 970003, Taiwan, ROC

[b] Department of Applied Mathematics, National University of Kaohsiung, Kaohsiung 81148, Taiwan, ROC

ARTICLE INFO

Article history:
Received 5 July 2015
Received in revised form 13 March 2016
Accepted 16 March 2016
Available online 28 March 2016

MSC:
primary 34A33
37B10
secondary 11K55
47A35

Keywords:
Multi-layer cellular neural networks
Hidden layers
Sofic shifts
Topological conjugacy
Total amalgamation
Williams classification theorem

ABSTRACT

This paper aims to characterize whether a multi-layer cellular neural network is of deep architecture; namely, when can an n-layer cellular neural network be replaced by an m-layer cellular neural network for $m < n$ yet still preserve the same output phenomena? From a mathematical point of view, such characterization involves investigating whether the topological structure of two (or multiple) layers is conjugate. A decision procedure that addresses the necessary and sufficient condition for the topological conjugacy between two layers in a network is revealed.

1. Introduction

This paper focuses on the following dynamical systems.

$$\begin{cases} \dfrac{d}{dt}x_i^{(n)}(t) &= -x_i^{(n)}(t) + z^{(n)} + \displaystyle\sum_{k\in\mathcal{N}}(a_k^{(n)} f(x_{i+k}^{(n)}(t)) \\ &\quad + b_k^{(n)} f(x_{i+k}^{(n-1)}(t))), \\ &\vdots \\ \dfrac{d}{dt}x_i^{(2)}(t) &= -x_i^{(2)}(t) + z^{(2)} + \displaystyle\sum_{k\in\mathcal{N}}(a_k^{(2)} f(x_{i+k}^{(2)}(t)) \\ &\quad + b_k^{(2)} f(x_{i+k}^{(1)}(t))), \\ \dfrac{d}{dt}x_i^{(1)}(t) &= -x_i^{(1)}(t) + z^{(1)} + \displaystyle\sum_{k\in\mathcal{N}} a_k^{(1)} f(x_{i+k}^{(1)}(t)), \end{cases} \tag{1}$$

for some integer $n \geq 2$, $i \in \mathbb{N}$, and $t \geq 0$. Herein $x_i^{(\ell)}(t) = 0$ for $1 \leq \ell \leq n$ and $t \geq 0$ provided $i \leq 0$. The so-called *neighborhood* $\mathcal{N}$ is a finite subset of integers $\mathbb{Z}$; the output function

$$f(x) = \frac{1}{2}(|x+1| - |x-1|) \tag{2}$$

is a piecewise linear map. $\mathbb{A} = [A^{(1)}, \dots, A^{(n)}]$ and $\mathbb{B} = [B^{(2)}, \dots, B^{(n)}]$ are the feedback and controlling templates, respectively, where $A^{(j)} = [a_k^{(j)}]_{k\in\mathcal{N}}$, $B^{(l)} = [b_k^{(l)}]_{k\in\mathcal{N}}$ for $1 \leq j \leq n, 2 \leq l \leq n$; $\mathbb{z} = [z^{(1)}, \dots, z^{(n)}]$ is the threshold. The template $\mathbb{T}$ of (1) consists of the feedback and controlling templates and the threshold, namely, $\mathbb{T} = [\mathbb{A}, \mathbb{B}, \mathbb{z}]$. Note that (1) are standard cellular neural networks (CNNs) if we let $n = 1$; in this case we call them *single layer* CNNs. The main reason one chooses (2) to be the output function for (1) is the application of the pattern recognition and image processing (Chua & Yang, 1988a, 1988b).

(1) are called *multi-layer cellular neural networks* (MCNNs, Chua & Shi, 1990) for $n \geq 2$. For the last few decades, MCNNs have received considerable attention due to the fact that they have been successfully applied to many areas such as signal propagation between neurons and image processing (Chua & Yang, 1988a; Crounse & Chua, 1995; Murugesh, 2010; Yang, Nishio, & Ushida, 2001, 2002), pattern recognition (Chua & Roska, 2002; Crounse, Roska, & Chua, 1993; Peng, Zhang, & Liao, 2009), CMOS realization (Carmona, Jimenez-Garrido, Dominguez-Castro, Espejo, & Rodriguez-Vazquez, 2002; Xavier-de Souza, Yalcin, Suykens, & Vandewalle, 2004), VLSI implementation (Chua & Shi, 1991), and self-organization phenomena (Arena, Baglio, Fortuna, & Manganaro, 1998). The sufficient conditions for the complete stability of (1) for $n \geq 1$ can be found in Li (2009), Paolo-Civalleri and Gilli

* Corresponding author.
E-mail address: chchang@nuk.edu.tw (C.-H. Chang).

http://dx.doi.org/10.1016/j.neunet.2016.03.005
0893-6080/© 2016 Elsevier Ltd. All rights reserved.

(1999), Savaci and Vandewalle (1992), Török and Roska (2004), Wu and Chua (1997), Xu, Pi, Cao, and Zhong (2007) and Zou and Nossek (1991).

Some kind of stationary solution for (1) is essential, namely, the *mosaic solution*, due to the wide range of complete stability in the parameter space and the application to image processing. For single layer CNNs, a *mosaic solution* $\bar{x}$ is a stationary solution of (1) which satisfies $|\bar{x}_i| \geq 1$ while its corresponding pattern $\bar{y} = (\bar{y}_i) = (f(\bar{x}_i))$ is called a *mosaic output pattern*. Since the output function (2) is a piecewise linear function with $f(x) = 1$ (resp. -1) if $x \geq 1$ (resp. $x \leq -1$), the output of a mosaic solution $\bar{x} = (\bar{x}_i)_{i\in\mathbb{N}}$ is an element in $\Sigma = \{-1, +1\}^{\mathbb{N}}$, and that is why we call it a *pattern*.

Given an n-layer CNN with $n \geq 2$, a stationary solution $\mathbf{x} = (x_i^{(1)}, \dots, x_i^{(n)})_{i\in\mathbb{Z}} \in \mathbb{R}^{\infty\times n}$ of (1) is called *mosaic* if $|x_i^{(k)}| > 1$ for $1 \leq k \leq N$, $i \in \mathbb{Z}$. The output $\mathbf{y} = (y_i^{(1)} \cdots y_i^{(n)})_{i\in\mathbb{Z}} \in \{-1, 1\}^{\infty\times n}$ of a mosaic solution is called a pattern, where $y_i^{(k)} = f(x_i^{(k)})$. The *solution space* $\mathbf{Y}$ of (1) stores the mosaic patterns $\mathbf{y}$, and the *output space* $\mathbf{Y}^{(n)}$ of (1) is the collection of the output patterns in $\mathbf{Y}$, or, more precisely,

$$\mathbf{Y}^{(n)} = \{(y_i^{(n)})_{i\in\mathbb{Z}} : (y_i^{(1)} \cdots y_i^{(n)})_{i\in\mathbb{Z}} \in \mathbf{Y}\}.$$

There are two important problems for a given MCNN: (i) How can we characterize whether an MCNN has deep architecture?[1] and (ii) How can we train a deep architecture MCNN? Those two problems are closely related to AI design, since deep architecture may be required for any such design. As a general reference, readers are referred to Bengio (2009) for more details.

This work is intended as an attempt to answer problem (i) and study the learning algorithm of (ii). First we try to formulate (i) and (ii) mathematically. Let Σ be a shift space; that is, Σ is a subset of $\mathcal{A}^{\mathbb{N}}$ for some finite set $\mathcal{A}$. $C(\Sigma, \Sigma)$ denotes the collection of maps from Σ to Σ. A map $\tau \in C(\Sigma, \Sigma)$ is called a *factor* (resp. an *embedding*) if it is onto (resp. one-to-one). τ is called a *conjugacy* if it is both a factor and an embedding. Then the above problem is formulated as follows.

Problem 1. Given an n-layer CNN (1) with $n \geq 2$.

(1) Corresponding to a given $\mathbf{Y}^{(1)}$, what kind of $\mathbf{Y}^{(n)}$ can be shown? To be precise, what is the symbolic space $\mathbf{Y}^{(n)}$ according to $\mathbf{Y}^{(1)}$?
(2) Given $\mathbf{Y}^{(i)}$ and $\mathbf{Y}^{(j)}$ for $1 \leq i \neq j \leq n$, does there exist a conjugacy τ between them?

Problem 1-(1) is closely related to the learning algorithm of MCNNs since one can figure out which kind of output solutions of (1) can be produced from a given input. It is also worth pointing out that if τ in Problem 1-(2) exists and $i < j$, then one can merge the ith layer to the jth layer to form one layer since conjugacy ensures that the phenomena exhibited by these two layers are **dynamically the same**. By continuing this process one would obtain a new MCNN such that each layer completely performs a different function between the other layers (thus each layer cannot be removed). Thus, one can characterize the depth of such an MCNN. In Ban, Chang, and Lin (2012), Ban and Chang provided a necessary and sufficient condition for determining whether $\mathbf{Y}^{(i)}$ and $\mathbf{Y}^{(j)}$ are conjugated for some $1 \leq i < j \leq n$; in this case, an n-layer CNN can be replaced by an $(n - j + i)$-layer CNN. Their criterion only works for the case where the symbolic transition matrices of $\mathbf{Y}^{(i)}$ and $\mathbf{Y}^{(j)}$ are both right-resolving (defined later). Later on, Chang (2015) obtained a necessary and sufficient condition for determining whether a multi-layer neural network can be reduced to one with fewer layers without the assumption in Ban et al. (2012); instead of right-resolving of the symbolic transition matrix, the criterion proposed in Chang (2015) hinges on the existence of the so-called factor-like matrix. This work proposes an algorithm, which provides a necessary and sufficient condition for determining the depth of an MCNN, for answering Problem 1-(2); the main contribution of the proposed algorithm is to demonstrate a workable criterion which can be realized by programming, such that the result can be derived in seconds.

Meanwhile, recall that the well-known Hopfield neural networks (Hopfield, 1982, 1984) can be formulated as

$$\begin{cases} C_i\dot{x}_i &= -\dfrac{x_i}{R_i} + \displaystyle\sum_{j=1}^{N} \omega_{ij}y_j + \theta_i, \\ y_i &= g_i(\lambda_i x_i), \end{cases} \quad \text{for } i = 1, \dots, N, \tag{3}$$

where x_i stands for the state of neuron i with each activation function g_i being sigmoid. It should be highlighted that if $n = 1$, then the main difference between (1) and (3) is their output functions and the weights between neurons. Hence the investigation of the MCNNs in this paper can be extended to Hopfield neural networks (HNNs) with some modification. Roughly speaking, MCNNs are hybrids between conventional neural networks, such as HNNs, and continuous automata; the behavior of the overall systems of both MCNNs and HNNs is driven by the weights of the processing unit's linear interconnection. The major discriminator is that the connections between MCNN processors are local, while all the HNN processors are fully interconnected. Beyond that, our methodology can also be applied to determine whether two stable multi-layer neural networks are topologically conjugate, in other words, whether or not two different neural networks, such as CNN and HNN, recognize the same images up to the change of color. The related work is in preparation.

As stated above, one of the applications of our algorithm is to see whether two completely stable networks are likely to be conjugated, that is, to determine if two networks exhibit the same dynamical behavior eventually. More precisely, the output of a network converges to patterns whenever it is completely stable; in this case, the output patterns are realized symbolically and can be analyzed via our algorithm and theorem. To the best of our knowledge, there is no such elucidation investigating multi-layer neural networks from this perspective. Furthermore, in Rakkiyappan, Chandrasekar, Lakshmanan, and Park (2014), Rakkiyappan, Zhu, and Chandrasekar (2014), the authors demonstrated the asymptotic stability of some types of stochastic neural networks with time-dependent delays and Markovian jump parameters. A natural question is to ask under what conditions a stochastic neural network with delays possesses similar behavior to a multi-layer cellular neural network, up to conjugacy. It is also interesting to elaborate the cost to simulate a stochastic neural network with delays by a deterministic multi-layer network; more precisely, to answer the question of how many layers we need, for instance, for a multi-layer cellular neural network to exhibit the dynamical behavior of a stochastic Cohen–Grossberg neural network with delays. The related work remains in preparation.

The rest of this paper is organized as follows. In Section 2, we consider the simplest case, i.e., $n = 2$ and elucidate how to produce the symbolic space of $\mathbf{Y}^{(2)}$ according to a given $\mathbf{Y}^{(1)}$. This method can be easily extended to the general case where $n > 2$. The so-called *symbolic transition matrices* $\mathbf{S}^{(i)}$ of $\mathbf{Y}^{(i)}$, for $i = 1, 2$, are defined therein, which is helpful for the study of Problem-(2). We prove that $\mathbf{S}^{(i)}$ is the complete invariant for the existence of conjugacy between $\mathbf{Y}^{(1)}$ and $\mathbf{Y}^{(2)}$ (Theorem 2.1). This gives the affirmative answer for Problem-(2) of $n = 2$. Finally we extend this result to the general case for arbitrary $n \geq 2$ (Theorem 3.1) in Section 3, and further discussion and our conclusion are addressed in Section 4.

[1] Deep architectures are composed of multiple levels of nonlinear operations, such as in neural networks with many hidden layers or in complicated propositional formulae re-using many sub-formulae.