Chapter 1
A Theoretical Framework for Parallel Implementation of Deep Higher Order Neural Networks

Shuxiang Xu
University of Tasmania, Australia

Yunling Liu
China Agricultural University, China

ABSTRACT

This chapter proposes a theoretical framework for parallel implementation of Deep Higher Order Neural Networks (HONNs). First, we develop a new partitioning approach for mapping HONNs to individual computers within a master-slave distributed system (a local area network). This will allow us to use a network of computers (rather than a single computer) to train a HONN to drastically increase its learning speed: all of the computers will be running the HONN simultaneously (parallel implementation). Next, we develop a new learning algorithm so that it can be used for HONN learning in a distributed system environment. Finally, we propose to improve the generalisation ability of the new learning algorithm as used in a distributed system environment. Theoretical analysis of the proposal is thoroughly conducted to verify the soundness of the new approach. Experiments will be performed to test the new algorithm in the future.

INTRODUCTION

HONNs (Higher Order Neural Networks) [Lee et al 1986; Giles et al 1987] are Artificial Neural Networks (ANNs) in which the net input to a computational neuron is a weighted sum of its inputs plus products of its inputs. Such neuron is called a Higher-order Processing Unit (HPU) [Lippman 1989]. It was known that HONNs can implement invariant pattern recognition [Psaltis et al 1988; Reid et al 1989; Wood et al 1996]. It was shown in [Giles et al 1987] that HONNs have impressive computational, storage and generalization capabilities.

DOI: 10.4018/978-1-5225-0788-8.ch001

One of the most important Artificial Intelligence technologies, ANN attempts to mimic the computational power of biological brain, such as the human brain, for image recognition, sound recognition, natural language processing, complicated decision making, etc. by interconnecting simple computational units. Like the human brain, an ANN needs to be trained before it can be used to make decisions. However, nearly all of the current ANN implementations involve using a software program, running on a standalone computer, to learn training examples out of a dataset. Depending on the size of the dataset, this training could take days or even weeks. This is becoming a more serious issue in the current Big-Data era when huge datasets are available for ANNs to learn. Therefore, this chapter proposes a theoretical framework for parallel implementation of Deep HONNs by answering the following research questions:

1. How to develop a new partitioning approach for mapping HONNs to individual computers within a master-slave distributed system (a local area network)? This will allow us to use a network of computers (rather than a single computer) to train a HONN to drastically increase its learning speed: all of the computers will be running the HONN simultaneously (parallel implementation). We will use the master computer to control the overall learning process by distributing learning tasks to the individual slave computers.
2. How to develop a new learning algorithm so that it can be used for HONN learning in a distributed system environment? A HONN model needs to be trained using a learning algorithm before it can be used to make decisions. All the current HONN learning algorithms are intended for use on a standalone computer. We will develop a new algorithm to allow HONN learning/training in a distributed system environment. This involves maintaining communication among individual computers within the system so that they collectively run the same task.
3. How to improve the generalisation ability of the new learning algorithm as used in a distributed system environment? Like the human brain, after a HONN is well trained it may be able to generalise, i.e. producing outputs based on new (previously unseen) inputs. This is the ultimate goal of training a HONN.

BACKGROUND AND LITERATURE REVIEW

Human brain processes information in parallel ways. Parallel processing is the ability of the brain to simultaneously process incoming stimuli of differing quality. For example, in human vision, the brain divides what it sees into several components: colour, motion, shape, and depth. These are individually, but simultaneously analysed, and then combined, and then compared to stored memories, which helps the brain identify what we are viewing [Myers 2001].

Parallel processing in computers is the simultaneous use of more than one CPU to execute a program (such as an ANN learning algorithm). This makes a program run faster because there are more engines (CPUs) running it (with the help of distributed processing software).

To understand why parallel processing is required for ANN implementation, it is necessary to understand how an ANN learns and then makes decisions. Generally speaking, an ANN session consists of two phases: learning (or training) and testing (or recall). In the learning phase, a set of training examples is presented to the network, and weights are adjusted in accordance with input values, their correlation, and output values. By these, an ANN learns a problem or remembers existing patterns (the training examples). In the recall phase, new input sets are presented and an ANN produces appropriate results.