Chapter 48
Designing Multilayer Feedforward Neural Networks Using Multi-Verse Optimizer

Mohamed F. Hassanin
Fayoum University, Egypt

Abdullah M. Shoeb
Taibah University, Saudi Arabia

Aboul Ella Hassanien
Cairo University, Egypt

ABSTRACT

Artificial neural network (ANN) models are involved in many applications because of its great computational capabilities. Training of multi-layer perceptron (MLP) is the most challenging problem during the network preparation. Many techniques have been introduced to alleviate this problem. Back-propagation algorithm is a powerful technique to train multilayer feedforward ANN. However, it suffers from the local minima drawback. Recently, meta-heuristic methods have introduced to train MLP like Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Cuckoo Search (CS), Ant Colony Optimizer (ACO), Social Spider Optimization (SSO), Evolutionary Strategy (ES) and Grey Wolf Optimization (GWO). This chapter applied Multi-Verse Optimizer (MVO) for MLP training. Seven datasets are used to show MVO capabilities as a promising trainer for multilayer perceptron. Comparisons with PSO, GA, SSO, ES, ACO and GWO proved that MVO outperforms all these algorithms.

INTRODUCTION

Artificial neural networks (ANNs) are very powerful computational models. These models have great capabilities to represent properties of various interdisciplinary problems. ANN is computationally universal; so it is at least as powerful as Turing machine (McCulloch & Pitts, 1943). In fact, ANN is more powerful than the Turing machine (Siegelmann, 2012). The ANN consists of highly interconnecting

DOI: 10.4018/978-1-5225-2229-4.ch048

simple processing elements. These processing elements simulate the biological neurons. ANN can be considering ANN as a structure of parallel processing. There are three shared points in almost all ANNs, namely, distributed representation, local processing, and nonlinear processing.

McCulloch and Pitts (1943) produce the first mathematical model to simulate the neuron. This mathematical model is developed by F. Rosenblatt and called later a perceptron (Rosenblatt, 1958). The perceptron is the building unit of more complicated and advanced architectures of ANNs. ANN architecture is determined by the way to connect and arrange perceptrons. There are many common ANN architectures like single-layer feed-forward ANNs, in which one input layer and one output layer of processing units and no feedback connections. Another example is multilayer feed-forward ANNs, in which one input layer, one output layer, and one or more hidden layers of processing units and no feedback connections. Recurrent ANN is another architecture in which at least one feedback connection. It may, or may not, have hidden units.

To get results from an ANN, it has to pass through two phases. Phase one is to set its parameters like number of layers, number of perceptrons per a layer, and weights of the network connections. The iterative process of assigning weights to the connection is called network training. The target of training is to find the optimal weights that yield minimum errors regarding the used sample. The second phase is ANN operation, in which we implement the trained ANN to the underlying problem.

There are three types of training, namely supervised, unsupervised, and reinforcement training. In supervised training, we provide the ANN both of input samples and of the corresponding desired output. The ANN is allowed to adapt itself through a suitable algorithm to decrease the gap between the actual and desired outputs. In reinforcement training, we provide the ANN with limited feedback. The supervisor scores the ANN performance of training examples. The third type is unsupervised training, in which the ANN miss the supervisor feedback completely.

We are interested in feed forward multilayer perceptron (MLP). ANN with two hidden layers can solve problems of recognition, separation, and classification whatever the problem complexity (Kůrková, 1992). The traditional training algorithm used with MLP is back-propagation (BP). BP algorithm utilizes – back-propagates – the sample errors to adapt the weights iteratively to minimize the network overall error.

However BP algorithm proves success in many problems, it is still suffers from a serious drawback. BP algorithm uses gradient descent notion and hence if the error function has a local minimum differs than the global one, the training may end at this local minimum rather than the overall deeper valley. The usual solution to this drawback is to start with a range of different initial weight sets. This chapter is dedicated to utilize multi-verse optimizer as a nature-inspired heuristics method to avoid BP drawback.

BACKGROUND

Recent studies alleviate this drawback by randomly initialize the weights and use nature-inspired heuristics methods to improve them. Among of others meta-heuristic methods are Genetic Algorithm (GA) (Leung, Lam, Ling, & Tam, 2003), Particle Swarm Optimization (PSO) (Prasad, Mohanty, Naik, Nayak, & Behera, 2015; Das, Pattnaik, & Padhy, 2014), Cuckoo Search (CS) (Nawi, Khan, & Rehman, 2013; Nawi, Khan, Rehman, Herawan, & Deris, 2014), Ant Lion Optimizer (ALO) (Mirjalili, 2015a), Social Spider Optimization (SSO) (Pereira, Rodrigues, Ribeiro, Papa, & Weber, 2014), Evolutionary Strategy (ES) (Beyer & Schwefel, 2002) and Grey Wolf Optimization (GWO) (Mirjalili, 2015b).