Engineering Applications of Artificial Intelligence 39 (2015) 1–13

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm

Fardin Ahmadizar [a,*], Khabat Soltanian [b], Fardin AkhlaghianTab [b], Ioannis Tsoulos [c]

[a] Department of Industrial Engineering, University of Kurdistan, Pasdaran Boulevard, Sanandaj, Iran
[b] Department of Software Engineering & Information Technology, University of Kurdistan, Pasdaran Boulevard, Sanandaj, Iran
[c] Department of Computer Engineering, Technological Institute of Epirus, Arta, Greece

ARTICLE INFO

Article history:
Received 28 April 2014
Received in revised form
24 September 2014
Accepted 7 November 2014
Available online 6 December 2014

Keywords:
Neural networks
Grammatical evolution
Genetic algorithm
Adaptive penalty approach
Classification problems

ABSTRACT

The most important problems with exploiting artificial neural networks (ANNs) are to design the network topology, which usually requires an excessive amount of expert's effort, and to train it. In this paper, a new evolutionary-based algorithm is developed to simultaneously evolve the topology and the connection weights of ANNs by means of a new combination of grammatical evolution (GE) and genetic algorithm (GA). GE is adopted to design the network topology while GA is incorporated for better weight adaptation. The proposed algorithm needs to invest a minimal expert's effort for customization and is capable of generating any feedforward ANN with one hidden layer. Moreover, due to the fact that the generalization ability of an ANN may decrease because of overfitting problems, the algorithm utilizes a novel adaptive penalty approach to simplify ANNs generated through the evolution process. As a result, it produces much simpler ANNs that have better generalization ability and are easy to implement. The proposed method is tested on some real world classification datasets, and the results are statistically compared against existing methods in the literature. The results indicate that our algorithm outperforms the other methods and provides the best overall performance in terms of the classification accuracy and the number of hidden neurons. The results also present the contribution of the proposed penalty approach in the simplicity and generalization ability of the generated networks.

1. Introduction

Artificial neural networks (ANNs) have risen from 1960s as a novel way to mimic the human brain. The learning ability of ANNs makes them a powerful tool for various applications, such as classification (Zhang, 2000; Cantu-Paz and Kamath, 2005; Castellani and Rowlands, 2009; Rivero et al., 2010; Castellani, 2013), clustering (Du, 2010), vision (Weimer et al., 2013; Zaidan et al., 2014), control systems (Li et al., 2014; Czajkowski et al., 2014), prediction (Muttil and Chau, 2006; Chen et al., 2008; Chen and Wang, 2010; Taormina et al., 2012; Weizhong, 2012; Khashei and Bijari, 2012; Coban, 2013), and many others (Rabuñal and Dorado, 2005). Designing the network architecture and training it are the most important problems with exploiting ANNs. In supervised problems, the task of training is the adaptation of the network weights so that the ANN can map a set of predefined input patterns to the desired corresponding outputs. BackPropagation (BP) algorithm (Rumelhart et al., 1986) is the most known ANNs training algorithm. Selecting effective input features, determining the number of hidden neurons, and the connectivity pattern of neurons is the task of designing the ANN topology that affects the network's leaning capacity and generalization, and usually needs to be performed by experts; they have to experiment on different topologies to find a suitable one.

Evolutionary neural networks (ENNs) refer to a class of research in which evolutionary algorithms (EAs) are used in the ANN designing and/or training. EAs are population-based, stochastic search algorithms stimulating the natural evolution. As EAs search globally and can handle infinitely large, non-differentiable and multimodal search space of architectures, this field has been interested by many researchers and ANN developers.

ENNs can be followed in three major groups. The first is the adaptation of the ANN weights by EAs, a form of the ANN training. The second group of works is the use of EAs for designing the architecture of ANNs, and the third is the Topology and Weight Evolving of ANNs (TWEANN) that includes the methods proposed to simultaneously evolve the weights and architecture. In the

* Corresponding author. Tel./fax: +98 871 6660073.
E-mail address: f.ahmadizar@uok.ac.ir (F. Ahmadizar).

http://dx.doi.org/10.1016/j.engappai.2014.11.003
0952-1976/

following, some of the important research lines in ENNs are reviewed; the more complete and state-of-the art reviews can be found in Yao (1999) and Azzini and Tettamanzi (2011).

1.1. ANN training algorithms

In the evolutionary training of ANNs, the network architecture is firstly determined, usually by experts. As the training of ANNs can be formulated as a search problem, the EA is used for exploring and exploiting the search space for finding an optimized set of weights. Binary representation of the connection weights is one of the earlier works in this area (Whitley, 1989; Caudell and Dolan, 1989; Whitley et al., 1990; Cantu-Paz and Kamath, 2005). However, some of researchers have used suitable real encoding EAs (Montana and Davis, 1989; Deb et al., 2002). Since evolution startegy (ES) and evolutionary programming (EP) algorithms are well-suited for real vector optimization, they have been employed in this area (Fogel et al., 1995; Heidrich-Meisner and Igel, 2009). An ANN training algorithm is always a well-known benchmark for new approaches developed in EA's research area; the generalized generation gap parent-centric recombinatin (G3PCX) algorithm introduced by Deb et al. (2002), for example, has been used in the empirical study performed by Cantu-Paz and Kamath (2005) to compared with a binary encoded genetic algorithm (GA). In order to improve the ANN training process, a local search may be embedded into an EA (Topchy and Lebedko, 1997).

1.2. ANN designing algorithms

The architecture of an ANN is of great importance because it affects the learning capacity and generalization capability of the ANN. Gradient-based search approaches such as constructive and destructive algorithms may be used in the automatic design of the architecture of ANNs (Frean, 1990; Sietsma and Dow, 1991). Nevertheless, the main drawback of such methods is that they are quite susceptible to fall in local optima (Angeline et al., 1994).

In the evolutionary design of the architecture, there are two approaches for representing solutions. In the first approach, called direct encoding, all of the network architecture details are encoded in a chromosome. Assuming the ANN as a directed graph and using an adjacent matrix for representing its genotype is a common direct encoding method. In this way, each entry is a binary number representing the presence or absence of a connection between two nodes (Miller et al., 1989; Kitano, 1990; Belew et al., 1991; Cantu-Paz and Kamath, 2005). This representation can also be employed to prune the connections of a network trained with full connectivity pattern (Reed, 1993). The other approach is indirect encoding, where only some characteristics of the architecture of an ANN are encoded in a chromosome. In the indirect encoding approach, some aspects of the destination network and/or network generation (mapping process) are predefined. For example, if we know that a fully connected architecture is suitable for our problem, it is sufficient to only encode the number of hidden layers, the number of neurons in each layer and the parameters of BP algorithm in a chromosome. The knowledge added to the algorithm reduces the search space and usually leads to a compact encoding.

There are several types of indirect encoding in the literature. Kitano (1990) has introduced a grammar based indirect representation encoding production rules in a chromosome instead of an adjacent matrix of the network. In the genotype to phenotype mapping, the production rules are decoded to generate the adjacent matrix; transforming the matrix to the corresponding network is then straightforward. Compact genotype is the main advantage of this method. Siddiqi and Lucas (1998) have shown that direct encoding can be at least as good as the Kitano's method. Fractal representation inspired by regularity, symmetry and self-similarity of live organisms is another indirect encoding scheme that may be more plausible than other encoding schemes (Merrill and Port, 1991). Gruau (1993, 1994) has introduced a cellular encoding as an indirect representation that is motivated by cell division in biology.

Cantu-Paz and Kamath (2005) have empirically evaluated, in addition to ENN methods for ANN training, also various ENN methods for feature selection (Kohavi and John, 1997; Yang and Honavar, 1998) and ANN designing on classification problems. Yang and Chen (2012) have proposed an evolutionary constructive and pruning algorithm to design the network topology, where its weights are optimized through BP. Furthermore, Soltanian et al. (2013) have applied a grammatical evolution (GE) algorithm (Ryan et al., 1998), called GE-BP, to design the architecture of an ANN, where the BP algorithm is used for the network evaluations against training data.

1.3. Simultaneous evolution of the ANN topology and weights

Yao and Liu (1997) have developed a TWEANN method based on EP, called EPNet, for simultaneously evolving the architecture and weights of an ANN, in which no crossover operation is utilized. NeuroEvolution of Augmenting Topologies (NEAT), presented by Stanley and Miikkulainen (2002), is another successful system in this area. A hypercube based NEAT with an indirect encoding has also been introduced by Stanley et al. (2009). Compact genotype, input as well as output scalability, and utilizing the geometry of the problem domain are the most important advantages of their system. Motsinger et al. (2006) and Tsoulos et al. (2008) have used GE for construction and training ANNs with one hidden layer. Castellani and Rowlands (2009) have developed a TWEANN method for recognizing wood veneer defects, and reported that there are no differences in accuracy between architectures using one and two hidden layers. Rivero et al. (2010) have applied genetic programming (GP) to design and train a feedforward ANN with any arbitrary architecture, and Oong and Isa (2011) have presented a global-local balanced GA to simultaneously design and train an arbitrary connected feedforward ANN. More recently, Castellani (2013) has compared evolutionary ANN designing and whole ANN development algorithms with classical feature selection and designing methods.

However, there are also other types of ANNs and EAs combination in the literature. Evolving the neuron transfer functions (Stork et al., 1990), the learning rule and parameters of BP (Bengio et al., 1990; Baxter, 1992), and ANN ensembles (Yao and Islam, 2008; Huanhuan and Yao, 2010; Felice and Yao, 2011; Donate et al., 2013; Ghazikhani et al., 2013) are other research lines in the ANN evolution. ANN ensembles originate from the idea of divide-and-conquer algorithms. Experiments have shown that EAs are good choice to automatically divide the problem space, and thus, ENN ensembles have attracted many researchers attention in the literature (Yao and Islam, 2008).

The major drawback of the ANN training algorithms described in Section 1.1 is the expert's effort needed for designing the network topology. The major disadvantage of the ANN designing algorithms described in Section 1.2 is that the search space is complex and noisy because the fitness assigned to a given architecture is dependent to the learning method. The methods described in Section 1.3 are free from these problems. However, most of them suffer from inability of using the problem domain knowledge, while some of them are extremely dependent to the expert.

As mentioned earlier, the TWEANN method proposed by Tsoulos et al. (2008) uses GE for designing the network topology as well as optimizing its weights. That is, their method encodes