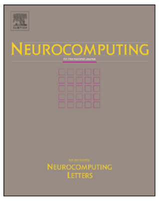
Contents lists available at ScienceDirect

Neurocomputing

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



Fast learning neural networks using Cartesian genetic programming

Maryam Mahsal Khan [a], Arbab Masood Ahmad [b], Gul Muhammad Khan [a,*], Julian F. Miller [c]

[a] Department of Electrical Engineering, UET Peshawar, Pakistan
[b] Department of Computer System Engineering, UET Peshawar, Pakistan
[c] Department of Electronics, University of York, UK, YO10 5DD

A R T I C L E I N F O

Article history:
Received 17 August 2011
Received in revised form
24 December 2012
Accepted 27 April 2013
Communicated by H. Zhang
Available online 15 May 2013

Keywords:
Artificial neural network
Pole balancing
Breast cancer
Neuroevolution
Recurrent networks

A B S T R A C T

A fast learning neuroevolutionary algorithm for both feedforward and recurrent networks is proposed. The method is inspired by the well known and highly effective Cartesian genetic programming (CGP) technique. The proposed method is called the CGP-based Artificial Neural Network (CGPANN). The basic idea is to replace each computational node in CGP with an artificial neuron, thus producing an artificial neural network. The capabilities of CGPANN are tested in two diverse problem domains. Firstly, it has been tested on a standard benchmark control problem: single and double pole for both Markovian and non-Markovian cases. Results demonstrate that the method can generate effective neural architectures in substantially fewer evaluations in comparison to previously published neuroevolutionary techniques. In addition, the evolved networks show improved generalization and robustness in comparison with other techniques. Secondly, we have explored the capabilities of CGPANNs for the diagnosis of Breast Cancer from the FNA (Finite Needle Aspiration) data samples. The results demonstrate that the proposed algorithm gives 99.5% accurate results, thus making it an excellent choice for pattern recognitions in medical diagnosis, owing to its properties of fast learning and accuracy.

The power of a CGP based ANN is its representation which leads to an efficient evolutionary search of suitable topologies. This opens new avenues for applying the proposed technique to other linear/non-linear and Markovian/non-Markovian control and pattern recognition problems.

1. Introduction

Artificial neural networks (ANNs) not only have the ability to extract information from complex data but also have the potential to resolve linear/non-linear problems in fields such as chemical processes, control, robotics, pattern recognition, computer vision, oil and gas, etc. [1–3].

Control Systems are conventionally developed by constructing a mathematical model that represents all dynamics of the system [4]. However, some systems that are complex and non-linear cannot be mathematically modeled, as there are no standard and conventional procedures to represent them. Thus the efficient design and development of a controller becomes difficult. Intelligent control is an unconventional process that provides the flexibility of generating abstract models without any indication of hidden dynamics of the system.

One training method used in ANNs is a genetic algorithm. This is broadly known as neuroevolution. For decades, the performance of the neuroevolutionary algorithms has been tested on the non-linear control problem e.g. developing linear and non-linear controller of a standard benchmark problem ‘the inverted pendulum’ for multiple scenarios of poles, optimum localization of mobile robots, auto-pilot helicopter or aircraft controller, automobile crash warning system, rocket control, routing over a data network, coordinating multi-rover systems, time-series prediction, lung sound detection, speech recognition, chemical processes and manufacturing, and the octopus arm task [5–11]. Many of the developed controllers have been found efficient and useful when compared to the controllers developed by conventional methods.

Genetic Programming is a form of automatic program induction where evolutionary algorithms are used to build computer programs and complex data structures [12–14]. In this paper, we are using a graph based form of genetic programming called Cartesian genetic programming (CGP) [15–17]. CGP has been explored in a range of diverse application domain and has shown to produce competitive performance [16,18–22]. A powerful aspect of CGP is its representation of graphs coupled with a high degree of genetic redundancy.

We have adapted CGP for representation and evolution of neural networks. We call our neuroevolutionary technique, CGP artificial neural networks (CGPANNs). Both feedforward and recurrent representations are examined. It is applied on a standard benchmark problem—pole balancing ranging from a simple setup to extremely difficult versions. The results obtained demonstrate that

* Corresponding author. Tel.: +92 91 9216796; fax: +92 91 9216663.
E-mail addresses: gk502@nwfpuet.edu.pk, engineergul@yahoo.com (G. Muhammad Khan).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.04.005

the proposed technique has a fast learning capability as it consistently outperforms all the previous methods explored to date.

The CGPANN technique is also applied to the problem of detecting breast cancer. Features from breast mass are extracted using fine needle aspiration (FNA) and the information is applied as input to CGPANN. FNA data available at the Wisconsin Diagnostic Breast Cancer web site is used for training and testing the network capabilities. The system developed produces fast and accurate results when compared to contemporary work done in the field. The error of the model comes out to be as low as 1% for Type-I (wrongly classifying benign samples as malignant) and 0.5% for Type- II (wrongly classifying malignant samples as benign). The paper is organized as follows. Section 2 is an overview of neuroevolution and the different algorithms developed so far. Section 3 describes the background information on Cartesian genetic programming. Section 4 describes the neuroevolutionary algorithm based on Cartesian genetic programming in detail. Section 5 describes the algorithm applied on the standard benchmark problems i.e. single and double pole balancing tasks along with simulation, analysis and results. Section 6 represents in detail the performance of the algorithm on the diagnosis of breast cancer. Section 8 concludes with discussions and future work.

2. Neuroevolution

Artificial neural networks (ANNs) are computational systems that map complex relationships between inputs and outputs. They are made up of interconnected nodes (neurons) and weighted connections. The properties of these nodes are inspired by biological neurons, and can exhibit complex global behaviour.

A number of evolutionary algorithms have been applied in the past decade to evolve either weights, topology or both of the parameters of artificial neural networks also known as TWEANN (Topology and Weight Evolving Artificial Neural Network). TWEANN algorithms can be 'direct' encoding schemes with the genotype representing all connections and nodes that appear in the phenotype, or they can be 'indirect' where the genotype specifies the rules for constructing the phenotype. Direct encodings are generally better at fine tuning and generating compact architectures [23–30], whereas indirect encodings are better at finding a particular type of ANN architecture (i.e. topology) [23,31–34].

Yao reviewed different combinations of ANNs and evolutionary algorithms (EAs) that evolved ANN connection weights, architectures, learning rules, and input features [35]. He used both direct and indirect encoding scheme and pointed out that the direct encoding scheme of ANN architectures is very good at fine tuning and generating a compact architecture, whereas the indirect encoding scheme is suitable for finding a particular type of ANN architecture quickly. In further analysis, he identified that separating the evolution of architectures and connection weights can cause fitness evaluation to mislead evolution, whereas simultaneous evolution of ANN architectures and connection weights produces better results [35].

In symbiotic adaptive neural evolution (SANE), the neuron population along with the representations of network topologies are evolved together. SANE has successfully been applied to pole balancing task obtaining the desired behaviour within relatively few evaluations [10]. ESP referring to enforced subpopulations is an extension to SANE where instead of one population of neurons, a subpopulation of hidden layer neurons is evolved. This has produced better results than SANE [36].

In conventional neuroevolution (CNE) a genotype represents the whole neural network similar to Wieland's algorithm [37]. It uses rank selection and burst mutation. The conventional algorithm has advantages over ESP as it evolves genotypes at the entire network level rather than neuron level, thus allowing potential global solutions to be found for a predefined network topology and size.

Cooperative synapse neuroevolution (CoSyNE) evolves neural network at the level of synaptic weights only. The CoSyNE approach has been shown to out-perform all the previous approaches on pole balancing problem [9,36].

Stanley presented a TWEANN, known as neuroevolution of augmenting topologies (NEAT) [38]. NEAT was shown to perform faster than many other neuro-evolutionary techniques. Unlike a conventional neural network whose topology is defined by the user, NEAT allows the user to evolve the network topology. Thus the complexity of the network can change. The algorithm is notable in that it evolves both network weights and structure, and efficiently balances between the fitness and diversity of evolved solutions. NEAT has also produced competitive results on pole balancing problem.

Cooperative co-evolution networks (COVNET) focus on the evolution of co-adapted subcomponents. Instead of evolving complete networks only subnetworks are evolved. COVNET uses this technique and has been shown to produce better generalization and smaller networks [11].

Tsoy and Spitsyn introduced a neuroevolutionary algorithm known as NeVA and applied it to the pole balancing problem. In NeVA, adaptive mutation of connections and weights of a network is performed [39].

Kazuhiro also looked at the pole balancing problem using MBEANN (Mutation Based Evolutionary ANN), which is a TWEANN algorithm where evolution is based on mutation only [40].

Continuous time recurrent neural networks (CTRNNs) are derived from dynamical neuron models known as leaky integrators and their behaviour is modeled mathematically using a system of differential equations. It is an intermediate step between sigmoidal and spiking neuron. Angeline et al. [41] evolve both the weights and topology of CTRNNs and argued that this provides a number of advantages over traditional approaches. In [42] CTRNNs are evolved using NEAT and applied on the pole balancing problem. This produced better results than the standard NEAT algorithm.

The HyperNEAT algorithm consists of a neural network called the substrate comprising nodes that are connected together [43,44]. The weights of these connections are encoded using another network called CPPN (Compositional Pattern Production Network) which is evolved using NEAT (Neuro-Evolution of Augmenting Topology) algorithm. The inputs to CPPN are the coordinates of the nodes of the ANN while the output is the weight of the connection between these nodes. The output is in the form of a pattern in a four dimensional Hyper cube, where pairs of dimensions correspond to the coordinates of the source and target neurons. The HyperNEAT algorithm has features in common with biological networks of neurons in which neurons corresponding to neighboring sensors are located closer together. Buk et al. [45] compared the evolutionary computation technique of Hyper NEAT with the modified form of it which they named HyperGP. In HyperGP the CPPN is evolved using Genetic Programming. While the desired fitness reached by evolving the weights of the ANN using NEAT algorithm in CPPN takes 92 generations, the same fitness is achieved with GP for the same population size in only 20 generations. Risi et al. [46] developed a new form of HyperNEAT called ES-HyperNEAT in which the number and location of nodes is decided by the algorithm. This differs from the original HyperNEAT which consists of fixed substrate where the number of hidden nodes is decided a priori, while only the connection weights are evolved by CPPN. In ES-HyperNEAT the locations of these nodes are determined by finding the variance of weights throughout the substrate. Nodes are automatically removed from the substrate at locations that have a variance lower than a threshold. Thus the algorithm takes care of the number of nodes and their associated weights.

Pujol and Poli used a form of Genetic Programming (GP) called Parallel Distributed GP (PDGP) to evolve weights, topology and