Self-Evolving Neural Networks with Minimal Human Intervention

Ms.R.Jaseenash[1], Assistant Professor
Department of Computer Science and Engineering, Sri Shanmugha College of Engineering and Technology, Pullipalayam, Morur (Post), Sankari (Tk), Salem, Tamil Nadu, India
jaseenash.r@shanmugha.edu.in
K.P.Vijayalakshmi[2], Assistant Professor
Department of Computer Science and Engineering, Sri Shanmugha College of Engineering and Technology, Pullipalayam, Morur (Post), Sankari (Tk), Salem, Tamil Nadu, India
vijayalakshmi.s@shanmugha.edu.in
Mrs.S.Abirami[3], Assistant Professor
Department of Computer Science and Engineering, Sri Shanmugha College of Engineering and Technology, Pullipalayam, Morur (Post), Sankari (Tk), Salem, Tamil Nadu, India
abiramii.s@shanmugha.edu.in
Mrs.S.Gokulapriya[4], Assistant Professor
Department of Computer Science and Engineering, Sri Shanmugha College of Engineering and Technology, Pullipalayam, Morur (Post), Sankari (Tk), Salem, Tamil Nadu, India
gokulapriya@shanmugha.edu.in

Abstract- Learning and evolution are two fundamental forms of adaptation. There has been a great interest in combining learning and evolution with artificial neural networks (ANN's) in recent years. This paper: 1) reviews different combinations between ANN's and evolutionary algorithms (EA's), including using EA's to evolve ANN connection weights, architectures, learning rules, and input features; 2) discusses different search operators which have been used in various EA's; and 3) points out possible future research directions. It is shown, through a considerably large literature review, that combinations between ANN's and EA's can lead to significantly better intelligent systems than relying on ANN's or EA's alone.

Keywords – Evolutionary computation, intelligent systems, neural networks, connection weights, architectures, learning rules.

INTRODUCTION

Evolutionary artificial neural networks (EANN's) refer to a special class of artificial neural networks (ANN's) in which evolution is another fundamental form of adaptation in addition to learning[1]. Evolutionary algorithms (EA's) are used to perform various tasks, such as connection weight training, architecture design, learning rule adaptation, input feature selection, connection weight initialization, rule extraction from ANN's, etc. One distinct feature of EANN's is their adaptability to a dynamic environment. In other words, EANN's can adapt to an environment as well as changes in the environment[2]. The two forms of adaptation, i.e., evolution and learning in EANN's, make their adaptation to a dynamic environment much more effective and efficient. In a broader sense, EANN's can be regarded as a general framework for adaptive systems, i.e., systems that can change their architectures and learning rules appropriately without human

intervention[3]. This paper is most concerned with exploring possible benefits arising from combinations between ANN's and EA's. Emphasis is placed on the design of intelligent systems based on ANN's and EA's. Other combinations between ANN's and EA's for combinatorial optimization will be mentioned but not discussed in detail. ANN's can be divided into feed forward and recurrent classes according to their connectivity. An ANN is feed-forward if there exists a method which numbers all the nodes in the network such that there is no connection from a node with a large number to a node with a smaller number[4]. All the connections are from nodes with small numbers to nodes with larger numbers. An ANN is recurrent if such a numbering method does not exist. The architecture of an ANN is determined by its topological structure, i.e., the overall connectivity and transfer function of each node in the network[5].

1. Generate the initial population $G(0)$ at random, and set $i = 0$;
2. REPEAT
   (a) Evaluate each individual in the population;
   (b) Select parents from $G(i)$ based on their fitness in $G(i)$;
   (c) Apply search operators to parents and produce offspring which form $G(i+1)$;
   (d) $i = i + 1$;
3. UNTIL 'termination criterion' is satisfied

Figure 1- A general framework of EA's[1]

THE EVOLUTION OF CONNECTION WEIGHTS

Weight training in ANN's is usually formulated as minimization of an error function, such as the mean square error between target and actual outputs averaged over all examples, by iteratively adjusting connection weights. Most training algorithms, such as BP and conjugate gradient algorithms, are based on gradient descent[6]. There have been some successful applications of BP in various areas, but BP has drawbacks due to its use of gradient descent. It often gets trapped in a local minimum of the error function and is incapable of finding a global minimum if the error function is multimodal and/or nondifferentiable. A detailed review of BP and other learning algorithms can be found in. One way to overcome gradient-descent-based training algorithms' shortcomings is to adopt EANN's, i.e., to formulate the training process as the evolution of connection weights in the environment determined by the architecture and the learning task. EA's can then be used effectively in the evolution to find a near-optimal set of connection weights globally without computing gradient information[7]. The fitness of an ANN can be defined according to different needs. Two important factors which often appear in the fitness (or error) function are the error between target and actual outputs and the complexity of the ANN. Unlike the case in gradient-descent-based training algorithms, the fitness (or error) function does not have to be differentiable or even continuous since EA's do not depend on gradient information[8]. Because EA's can treat large, complex, non-differentiable, and multimodal spaces, which are the typical case in the real world, considerable research and application has been conducted on the evolution of connection weights. The evolutionary approach to weight training in ANN's consists of two major phases[9]. The first phase is to decide the representation of connection weights, i.e.,

whether in the form of binary strings or not. The second one is the evolutionary process simulated by an EA, in which search operators such as crossover and mutation have to be decided in conjunction with the representation scheme. Different representations and search operators can lead to quite different training performance. A typical cycle of the evolution of connection weights is shown in Fig. 2[10]. The evolution stops when the fitness is greater than a predefined value (i.e., the training error is smaller than a certain value) or the population has converged.

(i) Binary Representation- The canonical genetic algorithm (GA) has always used binary strings to encode alternative solutions, often termed chromosomes[11]. Some of the early work in evolving ANN connection weights followed this approach. n such a representation scheme, each connection weight is represented by a number of bits with certain length. An ANN is encoded by concatenation of all the connection weights of the network in the chromosome. A heuristic concerning the order of the concatenation is to put connection weights to the same hidden/output node together[12]. Hidden nodes in ANN's are in essence feature extractors and detectors. Separating inputs to the same hidden node far apart in the binary representation would increase the difficulty of constructing useful feature detectors because they might be destroyed by crossover operators. It is generally very difficult to apply crossover operators in evolving connection weights since they tend to destroy feature detectors found during the evolutionary process[13].

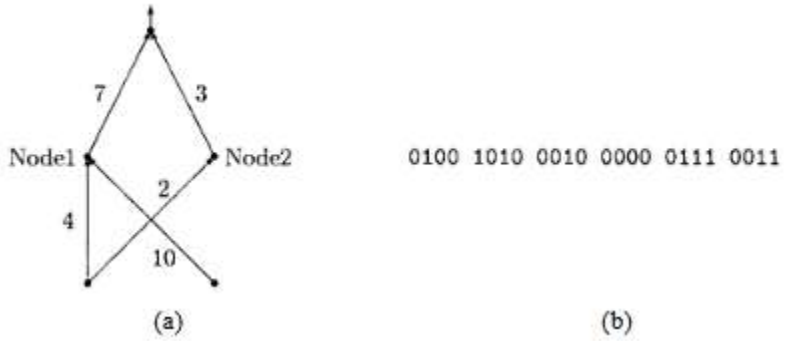

Figure 2 – (a) An ANN with connection weights shown. (b) A binary representation of the weights, assuming that each weight is represented by four bits[4]

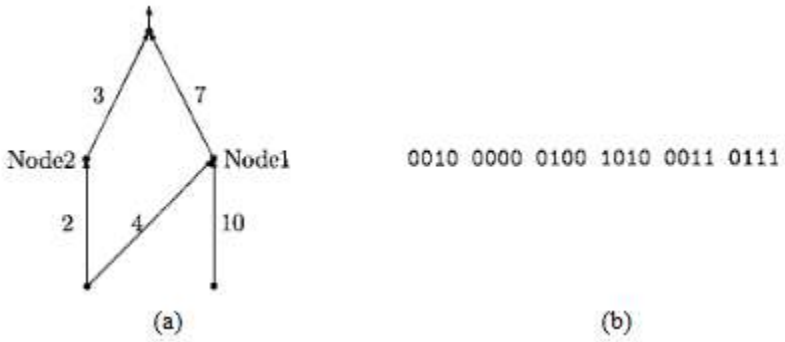

Figure 3 – (a) An ANN which is equivalent to that given in Figure 2- (a), (b) Its binary representation under the same representation scheme[5]

(ii) Real-Number Representation- There have been some debates on the cardinality of the genotype alphabet. Some have argued that the minimal cardinality, i.e., the binary representation, might not be the best. Formal analysis of nonstandard representations and operators based on the concept of

equivalent classes has given representations other than *k*ary strings a more solid theoretical foundation. Real numbers have been proposed to represent connection weights directly, i.e., one real number per connection weight. For example, a real-number representation of the ANN given by Fig. 2(a) could be (4.0,10.0,2.0,0.0,7.0,3.0)[14].

THE EVOLUTION OF ARCHITECTURES

The architecture of an ANN is predefined and fixed during the evolution of connection weights. This section discusses the design of ANN architectures. The architecture of an ANN includes its topological structure, i.e., connectivity, and the transfer function of each node in the ANN[15]. As indicated in the beginning of this paper, architecture design is crucial in the successful application of ANN's because the architecture has significant impact on a network's information processing capabilities[16]. Given a learning task, an ANN with only a few connections and linear nodes may not be able to perform the task at all due to its limited capability, while an ANN with a large number of connections and nonlinear nodes may over fit noise in the training data and fail to have good generalization ability. Up to now, architecture design is still very much a human expert's job[17]. It depends heavily on the expert experience and a tedious trial-and-error process. There is no systematic way to design a near-optimal architecture for a given task automatically. Research on constructive and destructive algorithms represents an effort toward the automatic design of architectures. Roughly speaking, a constructive algorithm starts with a minimal network (network with minimal number of hidden layers, nodes, and connections) and adds new layers, nodes, and connections when necessary during training while a destructive algorithm does the opposite, i.e., starts with the maximal network and deletes unnecessary layers, nodes, and connections during training[18]. However, as indicated by Angeline et al., "Such structural hill climbing methods are susceptible to becoming trapped at structural local optima." In addition, they "only investigate restricted topological subsets rather than the complete class of network architectures"[19].

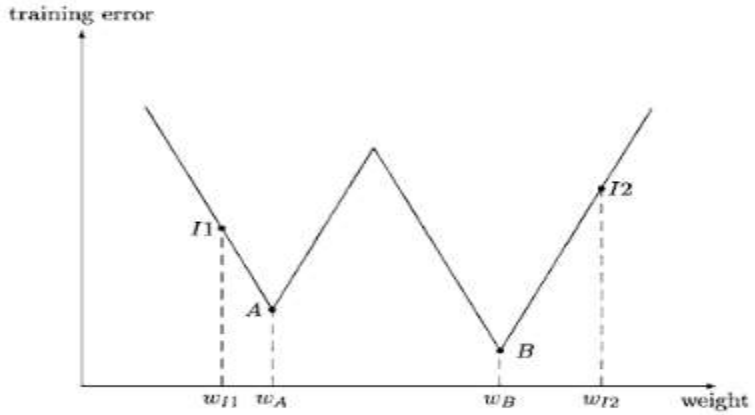

Figure 4- An illustration of using an EA to find good initial weights such that a local search algorithm can find the globally optimal weights easily[11]

Design of the optimal architecture for an ANN can be formulated as a search problem in the architecture space where each point represents architecture[20-24]. Given some performance (optimality) criteria, e.g., lowest training error, lowest network complexity, etc., about architectures, the performance level of all architectures forms a discrete surface in the space. The optimal architecture design is equivalent to finding the highest point on this surface. There are several characteristics of such a surface as indicated by Miller et al. which make EA's a better

candidate for searching the surface than those constructive and destructive algorithms mentioned above[25].

CONCLUSION

Evolution can be introduced into ANN's at many different levels. The evolution of connection weights provides a global approach to connection weight training, especially when gradient information of the error function is difficult or costly to obtain. Due to the simplicity and generality of the evolution and the fact that gradient-based training algorithms often have to be run multiple times in order to avoid being trapped in a poor local optimum, the evolutionary approach is quite competitive. Evolution can be used to find a near-optimal ANN architecture automatically. The direct encoding scheme of ANN architectures is very good at fine tuning and generating a compact architecture. The indirect encoding scheme is suitable for finding a particular type of ANN architecture quickly. Separating the evolution of architectures and that of connection weights can make fitness evaluation inaccurate and mislead evolution. Simultaneous evolution of ANN architectures and connection weights generally produces better results. It is argued that crossover produces more harm than benefit in evolving ANN's using distributed representation (e.g., multilayer perceptrons) because it destroys knowledge learned and distributed among different connections easily. Crossover would be more suitable for localized ANN's, such as RBF networks. Evolution can also be used to allow an ANN to adapt its learning rule to its environment. In a sense, the evolution provides ANN's with the ability of learning to learn. It also helps to model the relationship between learning and evolution. Preliminary experiments have shown that efficient learning rules can be evolved from randomly generated rules. Current research on the evolution of learning rules normally assumes that learning rules can be specified. While constraints on learning rules are necessary to reduce the search space in the evolution, they might prevent some interesting learning rules from being discovered. Global search procedures such as EA's are usually computationally expensive. It would be better not to employ EA's at all three levels of evolution. It is, however, beneficial to introduce global search at some levels of evolution, especially when there is little prior knowledge available at that level and the performance of the ANN is required to be high, because the trial-and-error and other heuristic methods are very ineffective in such circumstances. With the increasing power of parallel computers, the evolution of large ANN's becomes feasible. Not only can such evolution discover possible new ANN architectures and learning rules, but it also offers a way to model the
creative process as a result of ANN's adaptation to a dynamic environment.

REFERENCES

[1]. X. Yao, "Evolution of connectionist networks," in Preprints Int. Symp. AI, Reasoning & Creativity, Queensland, Australia, Griffith Univ., 2023, pp. 49–52.
[2]. "A review of evolutionary artificial neural networks," Int. J. Intell. Syst., vol. 8, no. 4, pp. 539–567, 2024.
[3]. Evolutionary artificial neural networks," Int. J. Neural Syst., vol. 4, no. 3, pp. 203–222, 2023.
[4]. The evolution of connectionist networks," in Artificial Intelligence and Creativity, T. Dartnall, Ed. Dordrecht, The Netherlands: Kluwer, pp. 233–243, 2024.
[5]. Evolutionary artificial neural networks," in Encyclopedia of Computer Science and Technology, vol. 33, A. Kent and J. G. Williams, Eds. New York: Marcel Dekker, 2023, pp. 137–170.
[6]. G. E. Hinton, "Connectionist learning procedures," Artificial Intell., vol. 40, no. 1–3, pp. 185–234, Sept. 2024.

[7]. Sanjay Kumar Suman, Dhananjay Kumar and L. Bhagyalakshmi, "Non Cooperative Power Control Game with New Pricing for Wireless Ad hoc Networks", International Review on Computers and Software, vol. 9, no. 1, pp. 18-28, 2014. ISSN: 1828-6003,

[8]. S. Porselvi, Sanjay Kumar Suman and L. Bhagyalakshmi, "Harvesting RF energy for mobile charging", Australian Journal of Basic and Applied Science, vol. 9, no. 20, pp. 454-465, June 2015.

[9]. K. Swapna, P. Rajalakshmi and Sanjay Kumar Suman, "Security Enhancement in MANET using Game Theory", Middle East Journal of Scientific Research, vol. 23, pp. 190-195, 2015.

[10]. VinaySrivatsan, Sanjay Kumar Suman, L. Bhagyalakshmi and S. Porselvi, "Non radiative wireless power transfer", Journal of Advances in Natural and Applied Sciences, vol. 10, no. 16, pp. 147-153, Nov. 2016.

[11]. Sujeetha Devi, Bhagyalakshmi L and Sanjay Kumar Suman, "Cluster based energy efficient joint routing algorithm for delay minimization in wireless sensor networks", International Journal of Pure and Applied Mathematics, vol. 119, no. 15, 307-313, 2018

[12]. J. H. Holland, Adaptation in Natural and Artificial Systems. Ann Arbor, MI: Univ. Michigan Press, 2024.

[13]. D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley, 2024.

[14]. D. B. Fogel, "An introduction to simulated evolutionary optimization," IEEE Trans. Neural Networks, vol. 5, pp. 3–14, Jan. 2023.

[15]. T. B¨ack, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," IEEE Trans. Evolutionary Computation, vol. 1, pp. 3–17, Apr. 2024.

[16]. D. R. Hush and B. G. Horne, "Progress in supervised neural networks," IEEE Signal Processing Mag., vol. 10, pp. 8–39, Jan. 2023.

[17]. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in Parallel Distributed Processing: Explorations in the Microstructures of Cognition, vol. I, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 2023, pp. 318–362.

[18]. M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," Neural Networks, vol. 6, no. 4, pp. 525–533, 2023.

[19]. K. J. Lang, A. H. Waibel, and G. E. Hinton, "A time-delay neural network architecture for isolated word recognition," Neural Networks, vol. 3, no. 1, pp. 33–43, 2021.

[20]. S. S. Fels and G. E. Hinton, "Glove-talk: A neural network interface between a data-glove and a speech synthesizer," IEEE Trans. Neural Networks, vol. 4, pp. 2–8, Jan. 2023.

[21]. S. Knerr, L. Personnaz, and G. Dreyfus, "Handwritten digit recognition by neural networks with single-layer training," IEEE Trans. Neural Networks, vol. 3, pp. 962–968, Nov. 2024.

[22]. R. S. Sutton, "Two problems with backpropagation and other steepest-descent learning procedures for networks," in Proc. 8th Annual Conf. Cognitive Science Society. Hillsdale, NJ: Erlbaum, 2023, pp. 823–831.

[23]. D. Whitley, T. Starkweather, and C. Bogart, "Genetic algorithms and neural networks: Optimizing connections and connectivity," Parallel Comput., vol. 14, no. 3, pp. 347–361, 2023.

[24]. Y. Chauvin and D. E. Rumelhart, Eds., Backpropagation: Theory, Architectures, and Applications. Hillsdale, NJ: Erlbaum, 2023.