Towards Designing Artificial Neural Networks by Evolution*

Xin Yao and Yong Liu
Computational Intelligence Group, Department of Computer Science
University College, The University of New South Wales
Australian Defence Force Academy, Canberra, ACT, Australia 2600
Email: xin@csadfa.cs.adfa.oz.au

Abstract

Designing artificial neural networks (ANNs) for different applications has been a key issue in the ANN field. At present, ANN design still relies heavily on human experts who have sufficient knowledge about ANNs and the problem to be solved. As ANN's complexity increases, designing ANNs manually becomes more difficult and unmanageable. Simulated evolution offers a promising approach to tackle this problem. This paper describes an evolutionary approach to design ANNs. The ANNs designed by the evolutionary process are referred to as evolutionary ANNs (EANNs). They represent a special class of ANNs in which evolution is another fundamental form of adaptation in addition to learning (also known as weight training). This paper describes an evolutionary programming (EP) based system to evolve both architectures and connection weights (including biases) of ANNs. Five mutation operators have been proposed in our evolutionary algorithm. In order to improve the generalisation ability of evolved ANNs, these five operators are applied sequentially and selectively. Validation sets have also been used in the evolutionary process in order to improve generalisation further. The evolutionary algorithm allows ANNs to grow as well as shrink during the evolutionary process. It incorporates the weight learning process as part of its mutation process. The whole EANN system can be regarded as a hybrid evolution and learning system. Extensive experimental studies have been carried out to test this EANN system. This paper gives some of the experimental results which show the effectiveness of the system.

Keywords — Evolutionary Algorithms, Artificial Neural Networks, Learning, Evolution, Generalisation, Adaptation.

1 Introduction

Designing artificial neural networks (ANNs) through simulated evolution has been investigated for many years [1, 2, 3, 4]. It offers a very promising and automatic alternative to designing ANNs manually. The advantage of the automatic design over the manual design becomes clearer as the complexity of ANNs increases. Manual design of ANNs requires the designer to have very good knowledge in both ANNs and the problem to be solved by the ANN. However, such knowledge is often unavailable for a non-ANN-expert facing a real-world problem.

Evolutionary artificial neural networks (EANNs) [3, 4] refer to a special class of artificial neural networks (ANNs) in which evolution is another fundamental form of adaptation in addition to learning. They provide a general framework for investigating various aspects

*This work is supported by the Australian Research Council through its small grant scheme.

of simulated evolution and learning, including the automatic design of ANNs. The general framework make it clear where the automatic design of ANNs fits into the whole picture of a general adaptive system which can change its behaviours through changing its "hardware", i.e., weights and architectures, and its "software", i.e., learning rule. We will review such a general framework [5, 6] in Section 2. Then we discuss the issue of designing ANNs through simulated evolution in Section 3. Section 4 presents our latest evolutionary system which can evolve ANN's weights and architectures at the same time. Finally, Section 5 gives our conclusions and indicates some future research topics.

2 A General Framework for EANNs

Evolution can be introduced into ANNs at various levels. At the lowest level, evolution can be introduced into weight[1] training, where ANN's weights are evolved. This evolutionary process is similar to the learning process in the connectionist paradigm where weights are adjusted in order to learn certain functions. At the next higher level, evolution can be introduced into ANN's architecture adaptation, where the architecture is evolved, rather than designed by human beings. At the highest level, evolution can be introduced into ANN's learning rule, i.e., the rule which specifies how to adjust weights in weight training. Since the weight training has traditionally been regarded as a learning process, the evolution of learning rules can be regarded as a process of learning to learn (weights). A general framework of EANNs which includes the above three levels of evolution is given in Figure 1 [3].

There have been some discussions on whether the evolution of learning rules is at the highest level among the three in Figure 1 [3, 4]. From the point of view of engineering, the decision on the level of evolution depends on what kind of prior knowledge is available. If there is more prior knowledge about EANN's architectures than that about their learning rules or a particular class of architectures is pursued, it is better to put the evolution of architectures at the highest level because such knowledge can be used to reduce the (architecture) search space and the lower level evolution of learning rules can be more biased towards this kind of architectures. On the other hand, the evolution of learning rules should be at the highest level if there is more prior knowledge about them available or there is a special interest in certain type of learning rules.

Figure 1 provides us with a common framework for discussing various EANN models if we interpret the simulated evolution in a broader sense, that is, if we interpret simulated annealing (SA), gradient descent search, exhaustive search, etc., as special cases of evolutionary algorithms. For example, the traditional back-propagation (BP) network can be considered as a special case of our general framework with one-shot (only-one-candidate) search used in the evolution of architectures and learning rules and the BP algorithm used in the evolution of connection weights. In fact, our general framework defines a three-dimensional space where 0 represents one-shot search and +1 represents exhaustive search along each axis. Each EANN model corresponds to a point in this space, where the three coordinates represent the three algorithms used by the EANN model in searching for the weights, architecture and learning rules. In this paper, we will describe an EANN model which uses one-shot algorithm to search for the learning rule, a variant of evolutionary programming (EP) to search for the architecture, and a hybrid BP-SA algorithm to search for the weights.

[1] Weights include connection weights and biases.

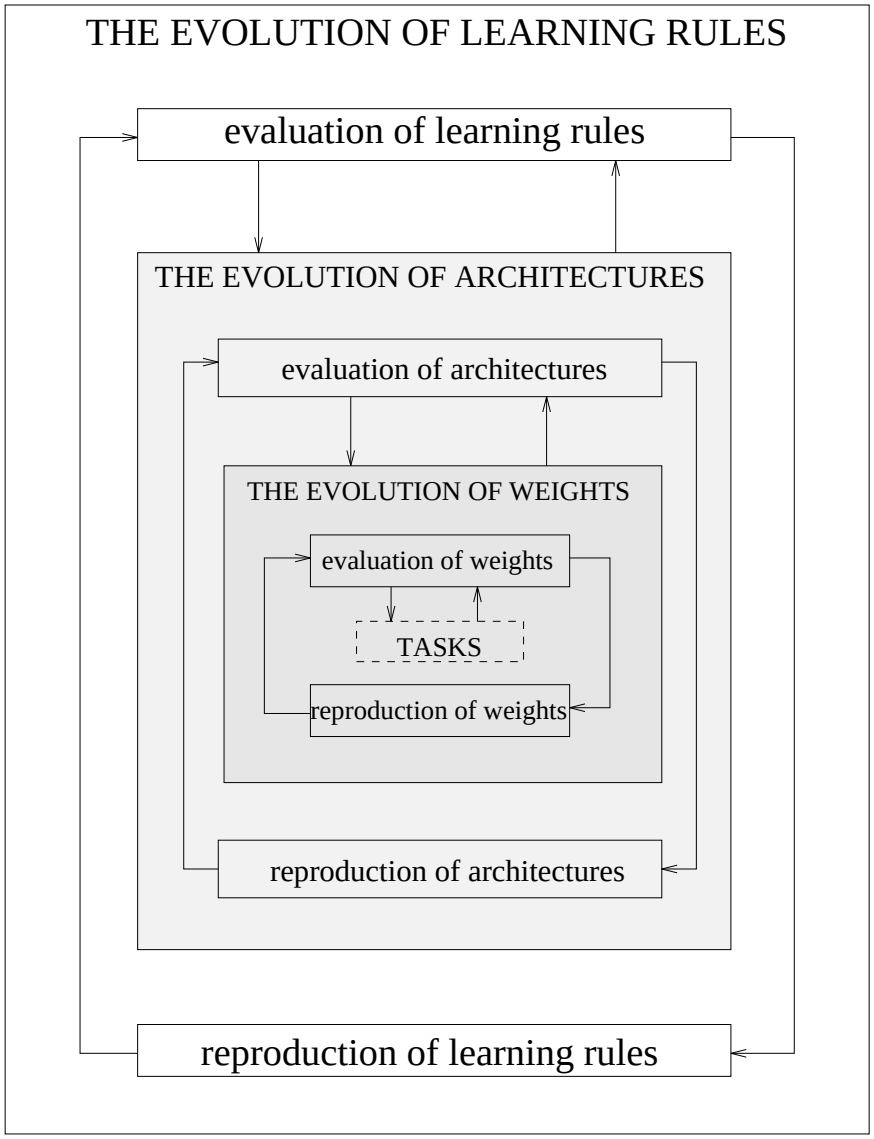


Figure 1: A general framework for EANNs.

3 Automatic Design of ANNs Through Evolution

The evolution of architectures provides an automatic way to design ANNs. It is obvious from the general framework in the previous section that any algorithm can be used to implement such simulated evolution. There are constructive and pruning algorithms which learn ANN's architectures and weights, but they are "susceptible to becoming trapped at structural local optima" [7]. In addition, they "only investigate restricted topological subsets rather than the complete class of network architectures" [7]. Evolutionary algorithms, like genetic algorithms, EP, and evolution strategies, are better suited to the task of evolving ANN's architectures [8].

Evolving ANN's architectures by evolutionary algorithms is not an easy task. The many to many mapping between genes and phenotypes has caused two major problems in evolving ANNs. The first problem is the noisy fitness evaluation problem. That is, we normally use one phenotype's fitness to approximate its genotype's fitness. For example, the fitness of a trained ANN (from a random set of initial weights) is often used to represent the fitness of the ANN's architecture. Such evaluation of the architecture is noisy because it depends on the random initialisation and the training algorithm used. One genotype may have many different phenotypes.

The second problem is the well-known permutation problem (or competing conventions problem). That is, one phenotype may have many different genotypes. For example, two strictly-layered feedforward ANNs which order their hidden nodes differently are functionally equivalent, but may have different genotypic representations.

In order to alleviate the above two problems in the evolutionary design of ANNs, we have decided to evolve the architecture and the weights at the same time using an EP algorithm. The simultaneous evolution of the architecture and weights means that each individual in the population is an ANN with the architecture and weights. Its evaluation is more accurate because we are evaluating an ANN with the architecture and weights. We do not use the fitness of an ANN with the architecture and weights to represent that of the architecture. The use of an EP algorithm avoids crossover operators which will not be effective due to the permutation problem.

4 A New System for Designing ANNs — EPNet

We have developed a new evolutionary system for designing ANNs automatically. The main structure of the system, EPNet, is given in Figure 2.

The EPNet system is built upon an EP algorithm which adopts a rank-based selection scheme [9] and five mutations; hybrid training, node deletion, connection deletion, connection addition and node addition [10, 11, 12]. Hybrid training is the only mutation in EPNet which modifies ANN's weights. It is based on a modified BP (MBP) algorithm with an adaptive learning rate and an SA algorithm. The other four mutations are used to grow and prune hidden nodes and connections. Only feedforward ANNs are considered in EPNet at present.

The number of epochs used by MBP to train each ANNs in a population is defined by two user-specified parameters. There is no guarantee that an ANN will converge to even a local optimum in a generation. Hence this training process is called partial training.

The five mutations are attempted sequentially. If one mutation leads to a better offspring, it is regarded as successful. No further mutation will be applied. In other words,

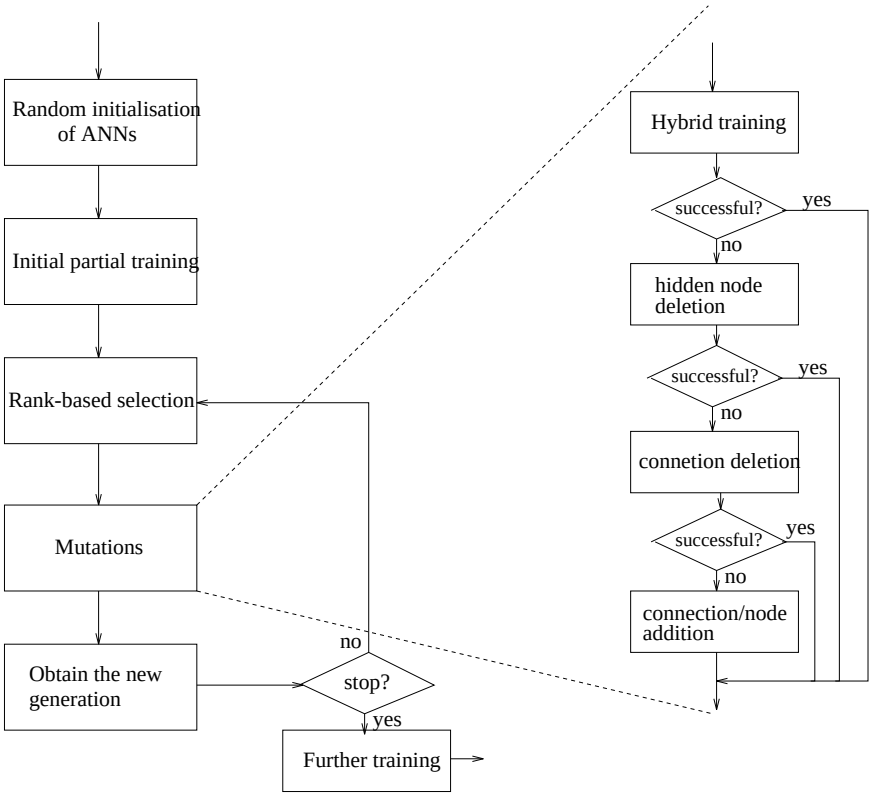


Figure 2: The main structure of EPNet.

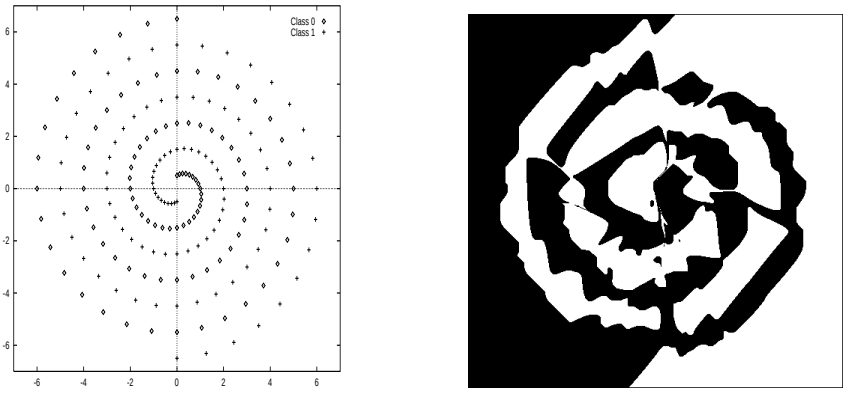
only one of the five mutations will be applied each time. The order of deletion first and addition later encourages the evolution of compact ANNs. EPNet puts a lot of emphasis on the behavioural links between parents and offsprings. It deletes and adds connections probabilistically according to their importance in the ANN. Node deletion is done at random, but node addition is achieved through splitting an existing node.

In order to improve the generalisation ability of evolved ANNs, we have used two validation sets in EPNet. The first one is used to compute the fitness of each ANN. After the evolutionary process, all the ANNs in the final population will be trained further by MBP on the combined training and the first validation set. The ANN which performs the best on the second validation set will be the final output of EPNet.

We have tested our EPNet on a number of artificial and real-world data sets, including the parity problem [10, 12] of various sizes, four medical diagnosis problems [11], and the Australian credit card assessment problem [12]. We also tested our EPNet on another very difficult artificial problem — the two-spiral problem. The result of a typical run by EPNet on the problem is shown in Figure 3.

5 Conclusion

This paper addresses the issue of automatic design of ANNs. It is argued that the evolutionary approach offers a very promising and competitive alternative to designing ANNs manually or by a constructive/pruning algorithm. The evolutionary design of ANNs is discussed in a general framework of EANNs, which provides a common basis for comparing



(a) (b)

Figure 3: (a) The training data set of the two spiral problem with 194 examples. (b) The decision region formed by the ANN evolved by EPNet. There are 14 hidden nodes and 131 connections in the ANN.

and investigating various EANN models. A new evolutionary system for designing ANNs, EPNet, is described. Experimental studies were carried out for a number of problems. This paper reports the result on the well-known two-spiral problem. Our future work includes parallelisation of the EPNet system and improvement of the hybrid training since it is the mutation which consumes most of the computation time.

References

[1] J. D. Schaffer, D. Whitley, and L. J. Eshelman, "Combinations of genetic algorithms and neural networks: a survey of the state of the art," In D. Whitley and J. D. Schaffer, editors, *Proc. of the Int'l Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92)*, pages 1–37. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[2] X. Yao, "A review of evolutionary artificial neural networks," *International Journal of Intelligent Systems*, 8(4):539–567, 1993.

[3] X. Yao, "Evolutionary artificial neural networks," *International Journal of Neural Systems*, 4(3):203–222, 1993.

[4] X. Yao, "Evolutionary artificial neural networks," In A. Kent and J. G. Williams, editors, *Encyclopedia of Computer Science and Technology*, volume 33, pages 137–170. Marcel Dekker Inc., New York, NY 10016, 1995.

[5] X. Yao, "Evolution of connectionist networks," In T. Dartnall, editor, *Preprints of the Int'l Symp. on AI, Reasoning & Creativity*, pages 49–52, Queensland, Australia, 1991. Griffith Univ.

[6] X. Yao, "The evolution of connectionist networks," In T. Dartnall, editor, *Artificial Intelligence and Creativity*, pages 233–243. Kluwer Academic Publishers, Dordrecht, 1994.

[7] P. J. Angeline, G. M. Sauders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. on Neural Networks*, 5(1):54–65, 1994.

[8] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms," In J. D. Schaffer, editor, *Proc. of the Third Int'l Conf. on Genetic Algorithms and Their Applications*, pages 379–384. Morgan Kaufmann, San Mateo, CA, 1989.

[9] X. Yao, "An empirical study of genetic operators in genetic algorithms," *Microprocessing and Microprogramming*, 38:707–714, 1993.

[10] Y. Liu and X. Yao, "A population-based learning algorithm which learns both architectures and weights of neural networks," *Chinese Journal of Advanced Software Research* (Allerton Press, Inc., New York, NY 10011), 3(1):54-65, 1996.

[11] X. Yao and Y. Liu, "Evolving artificial neural networks for medical applications," In *Proc. of 1995 Australia-Korea Joint Workshop on Evolutionary Computation*, pages 1–16. KAIST, Taejon, Korea, September 1995.

[12] X. Yao and Y. Liu, "Evolutionary artificial neural networks that learn and generalise well," Accepted by *the 1996 IEEE International Conference on Neural Networks, Washington, DC, USA*. 3-6 June 1996.