HYBRID TRAINING USING MICROBIAL GA AND BACKPROPAGATION

Abstract

This paper shows how the speed of convergence of an artificial neural network can be improved using a genetic algorithm. Standard gradient-descent algorithms, such as backpropagation, suffer from various problems connected to the random initialization of the network's weights. Using the global-search ability of a genetic algorithm, it is possible to find an optimal starting point. The starting set of weights is then fine-tuned by the gradient-descent algorithm. The paper shows how a simple genetic algorithm, a microbial genetic algorithm, can effectively reduce the number of epochs needed to learn a task. The experiment has been conducted on three different benchmark tasks: a XOR problem, an NMN encoder problem and a two-spirals problem. The real benefits of this hybrid approach become apparent when the error landscape presents many local minima, which present a problem for gradient descent. The use of this hybrid approach has been proved to be more efficient than standard methods in different fields, such as finance and medicine.

Introduction

Learning and evolution are two fundamental forms of adaptation. Genetic algorithms are a powerful searching method inspired by natural selection. Among the many possible application, they can be used to optimize neural networks. There are three possible combinations between genetic algorithms and artificial neural networks: evolution of the connection weight, evolution of the architecture and evolution of the learning rule. This paper focuses on the first one using hybrid training. It is known that GAs are not particularly good at fine-tuned local search but are very good at global search. Conversely, ANNs are good at the opposite. The efficiency of the two algorithms can be improved significantly by combining GA and ANN. GAs can be used to locate a good starting point for the ANN's training. The GAs' ability to locate the basin of attraction of the global minimum for the error function can significantly reduce the probability of incurring in a local minima solution, this problem is especially true for backpropagation algorithm. Hybrid training has been used successfully in many application areas [1]-[4]. These researches used GA to search for a near-optimal set of initial connection weights and then used BP to perform a local search from these initial weights. Their works suggest that hybrid GA/BP algorithms are more efficient than GA or BP alone. In the following paper, a new algorithm for hybrid training is described. This algorithm is focused on simplicity and it is not meant to be the best configuration, rather it shows that even with the most minimalist approach is possible to demonstrate the strength of the GA/BP approach. Three different tasks have been tested using both BP alone and the GA/BP approach.

Genetic Algorithms

Genetic algorithms (GAs) are a kind of stochastic search process that starts from a set of finite representations (a population of chromosomes), each mapping a possible solution. Each representation is associated with a fitness value. The individuals with the highest fitness have the highest reproduction probabilities. There are many possible implementations for a genetic algorithm, but the main attributes that are common to most of the implementations are: heredity, variation and selection. Heredity means that new individuals are similar to their parents. Variation means that the new individuals are not identical to those they replace. Selection is the process of discrimination between many individuals based on their fitness [5]. The model used for this paper is based on the

microbial genetic algorithm described by Inman Harvey [5]. This is the simplest form that maintains elements of mutation, crossover and elitism. The process of evolution is divided into tournaments. In each tournament the following operations take place:

1. Select the first individual
2. Select the second individual
3. Based on their fitness, individuate Winner and Loser
4. For each gene: overwrite Loser's gene with Winner's gene with probability cR (crossover), mutate Loser's gene (mutation) with probability mR
5. Measure new fitness and repeat till end condition is met

Note that in this algorithm, uniform crossover is implemented; in uniform crossover, each bit is chosen from one of the two parents with a certain probability cR. This is not the only type of crossover that can be implemented. The second individual is selected among the neighbours of the first one to simulate the effect of a geographical distribution [10]. To apply this algorithm to ANNs there are two possible ways to encode the weights: binary and real. It has been used a real value encoding to avoid having very long binary strings. When using real encoding, mutation is created by adding a small δ, where δ has a normal distribution. A population represents a set of neural networks with different weights. The fitness of each individual is calculated by feedforwarding the training inputs and comparing the outputs with the expected values. It is expected that the best set of weights dominates the population through the genetic search process.

Artificial Neural Networks

An artificial neural network, or simply neural network, is a collection of units, called neurons, connected by a set of weights. The simplest NN is a feed-forward neural network, where information flows from the input unit and ends at the output unit. In between the input and output unit, there can be multiple hidden layers. Each neuron applies an activation function, such that the output of the neuron *i* is $z_i = f(\sum w_{ij}x_j + b_i)$. The task of a neural network is to learn a task by generalizing a set of training samples. The learning action can be performed in multiple ways, but it always consists in adjusting the weights following a predefined learning rule. The most common learning rule is backpropagation (BP). Backpropagation updates the weights by subtracting the partial derivative $\frac{\partial C}{\partial w_{ij}}$ for the weights and $\frac{\partial C}{\partial b_i}$ for the biases, where C is the cost function, w_{ij} is the weight from node *j* to node *i* and b_i is the bias on node *i*. BP is an efficient way to train multilayer NN and has been applied to many tasks in science, finance and engineering. However, BP suffers from several problems as described in [6]-[8]. The main problems are: it is slow, performance is sensitive to initial conditions and it can be easily trapped in local minima.

Methods

To improve the performance of backpropagation, it is possible to use a genetic algorithm to locate the best starting point that will lead BP to find the set of weights and biases the minimize the error function. By exploiting the global search ability of genetic algorithms, it is possible to: speed up BP, remove the sensibility from the initial conditions and avoid local minima solutions. Speed performance is improved because only fine-tuning is required. The sensitivity from initial conditions and the probability of finding a local minimum are removed because the initial conditions are already

in the basin of attractions of the global minimum. To test the improvements introduced by the GA/BP approach, three tests have been proposed. The three tasks were classic benchmark tasks for NN; the first task was a XOR problem, the second problem was a N-M-N encoder and the last problem was a two-spirals problem. The genetic algorithm was used to search for a set of weights and biases which yielded an error lower than a predefined condition. The solution was then fine-tuned to match the condition at which the task was considered learned. The number of epochs needed to solve the problem was recorded, one evaluation of an individual in the population was considered as 0.5 epochs since it does not involve the backpropagation process, only the feedforward. Each test was repeated several times, only the solution that converged were recorded.

XOR Problem

The XOR problem is a non-linearly separable problem where the network has to map a couple of binary inputs to the correct output. This problem is one example of a N-parity problem, the output must be on if an odd number of inputs is on. This set of problems require the network to separate similar inputs since the nearest neighbours must produce the opposite answer.

		Output
0	0	0
0	1	1
1	0	1
1	1	0

To solve the problem it has been used a three-layer neural network (2-2-1) with standard backpropagation (learning rate at 0.01). The GA had a population of 20 sets of weights, initialized using a normal distribution with zero mean and variance equal to 5. These values were chosen using trial and error. The GA was terminated when the TSS error reached 1. The best set of weights was used as the starting point for the training. The task was considered learned when the TSS error was less than 0.04.

N-M-N Encoder Problem

This family of problems has been popular in the connectionist community for many years. The neural network is presented with N distinct input patterns, each of which has only one of the input units turned on (set to 1.0); the other input bits are turned off (set to 0.0). The task is to duplicate the input pattern in the output units. Since all information must flow through the hidden units, the network must develop a unique encoding for each of the N patterns in the M hidden units and a set of connection weights that, working together, perform the encoding and decoding operations. When M is less than $log_2 N$, the encoder is called tight encoder [9]. In real-world applications, it is usually more convenient to have enough hidden layers and enough nodes such that the system can easily learn the task, paying attention to avoid overfitting. N-M-N encoders have a particular feature: when using backpropagation, only one input-side weight is modified.

It has been chosen to use a 16-4-16 tight encoder to test the GA/BP approach. For this problem has been used a neural network using backpropagation with momentum. This version of backpropagation is faster than the standard version and it is commonly used in many systems. The GA had a population of 20 random weights, initialized using a normal distribution with zero mean and variance equal to

10. These values were chosen using trial and error. Tournaments continued until the TSS error was 25, 10% of the maximum error $16^2 = 256$. The best set of weights was used as the starting point for the training. The task was considered learned when the TSS error was less than 0.04.

Two-spirals Problem

The two-spirals problem is composed of two intertwined spirals of different classes. The task requires the network to learn a mapping that distinguishes between points of the two spirals. This problem is difficult because it requires the network to create highly nonlinear separation bounds. Using standard backpropagation, it is common to end in a local minimum.

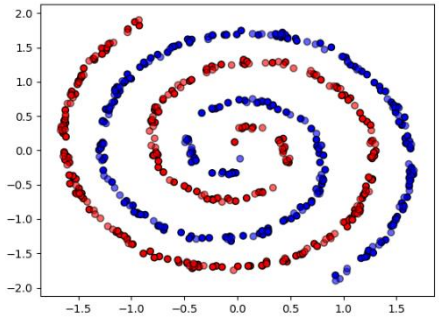

Figure 1 Two-spiral problem

For this problem, it has been used a five-layer neural network (2-10-10-10-1) using backpropagation with momentum (learning rate at 0.001 momentum at 0.9). Each problem had 300 points, divided into two sets: a training set with 180 points and a testing set of 120 points. The population contained 50 sets of random weights, initialized using a normal distribution with zero mean and variance equal to 1. For this problem, the genetic algorithm was stopped when the number of misclassified points was less than 150 or the number of tournaments was more than 1000. These values were chosen using trial and error. The error was measured on the training set. The best set of weight was used as the starting point for the training. The task was considered learned when the number of misclassified points from the testing set was less than 1%.

Results

XOR Problem

Training the network only with standard BP, the network learned the task in 952 epochs averaged over 60 repetitions ($\sigma = 473$). Using the hybrid approach GA/BP, the average number of tournaments was 203 ($\sigma = 289$), and the average number of epochs was 625 ($\sigma = 438$). GA was used to reduce the TSS error up to 1. This value can be varied but using a lower threshold would not lead to a faster convergence for backpropagation. Summing together the number of tournaments and epochs, the result is 2655. Not all the simulations converged within 10000 epochs, these were simply ignored. *Figure 2* shows the fitness increasing during the tournaments, it is common to have a stair shaped graph. *Figure 3* shows the error decreasing during the fine-tuning phase. The shape of the graph is much more continuous because of the effect of the continuous learning of the BP algorithm.

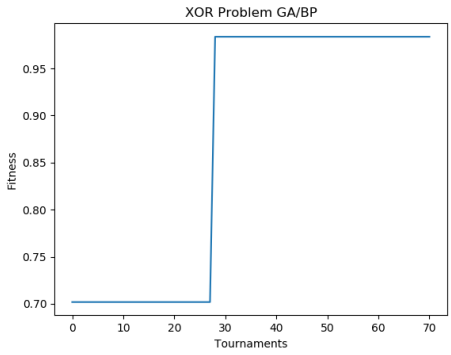

Figure 2 Sample realization from XOR, GA/BP method. GA phase. Fitness was 1/TSS

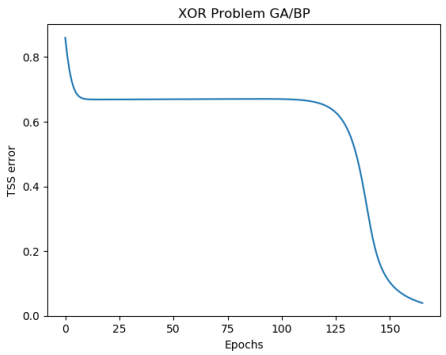

Figure 3 Sample realization from XOR, GA/BP method. BP phase.

N-M-N Encoder

Training the network only with BP, the network learned the task in 3223 epochs averaged over 100 repetitions ($\sigma = 290$). Using the hybrid approach GA/BP, the average number of tournaments was 140 ($\sigma = 1217$), and the average number of epochs was 1854 ($\sigma = 2455$). GA was used to reduce the TSS error up to 25. Summing together the number of tournaments and epochs, the result is 13229. As for the previous experiment, simulations which did not converge within 10000 epochs were ignored. *Figure 4* shows the increasing fitness during the tournaments. *Figure 5* shows the error during the fine-tuning phase, the error approaches the minimum very fast and then plateaus and decreases very slowly.

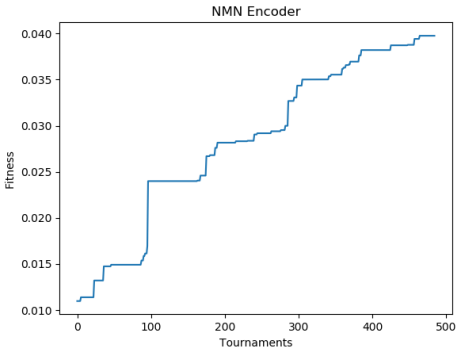

Figure 4 Sample realization from NMN encoder, GA/BP method. GA phase. Fitness was 1/TSS.

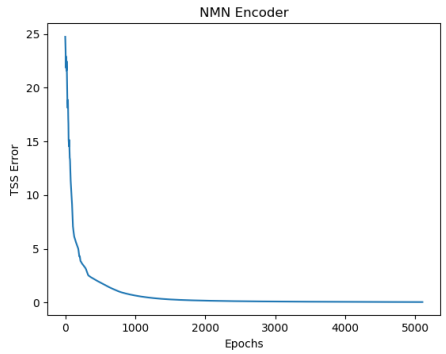

Figure 5 Sample realization from NMN encoder, GA/BP method. BP method.

Two-spirals Problem

Training the network only with BP, the network learned the task in 61320 epochs averaged over 20 repetitions ($\sigma = 17352$). Using the hybrid approach GA/BP, the average number of tournaments, over 10 repetitions, was 766 ($\sigma = 344$), and the average number of epochs was 10315 ($\sigma = 5901$). GA was used to reduce the TSS error up to 0.6. Summing together the number of tournaments and epochs, the result is 29465. Also in this experiment, a minority of the simulations did not converge and were ignored. *Figure 6* shows the fitness increasing during the tournaments. *Figure 7* shows the

error decreasing, the graph is more scattered than the previous ones because the error landscape is much more complex and even a small variation in the weights can change greatly the result.

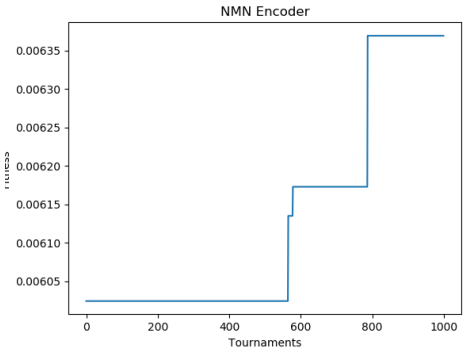

Figure 6 Sample realization from the two-spirals problem, GA/BP method.BA phase. Fitness is 1 over the number of misclassified points.

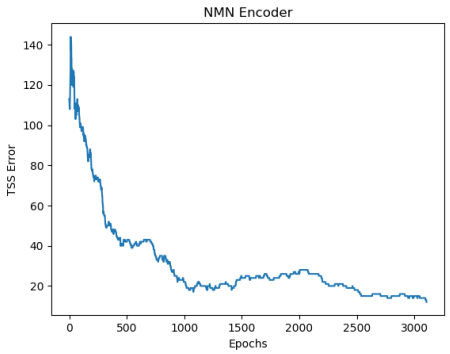

Figure 7 Sample realization from the two-spirals problem, GA/BP method.

Discussion

Using the hybrid approach, the solution at the first task was found using 34% fewer epochs than with the standard approach, however looking at the combined number of epochs, that is the sum of epochs and half the number of tournaments times the size of the population, the hybrid approach was 280% slower than the standard approach. For the second task, the results were similar: the hybrid approach used 40% fewer epochs to train the network, and the combined number of epochs was 4 times higher than the number of epochs using standard backpropagation. The last task was solved in 83% fewer epochs using the hybrid approach, and the combined number of epochs was 48% of the number of epochs needed for the standard backpropagation algorithm. The experiments demonstrate that the use of a gradient-descent algorithm, such as backpropagation, from the point specified by a genetic algorithm, provides faster convergence than starting from a random point of the weight space. The tasks used to test the GA/BP approach were standard benchmarks for neural network, but do not necessarily reflect real-world scenarios, where data presents deep interconnections. The use of hybrid training techniques should be tailored to the specific task, however one possible aspect that could greatly influence the GA performance is the crossover operation. For the microbial genetic algorithm implemented, it has been used a uniform crossover, this type of crossover does not take in consideration possible relations between neighbours' weight. Other types of crossover would be more appropriate if the task intrinsically involves some sort of interdependency between regions of the chromosome since they would avoid disrupting these regions. For example, using a one-point crossover would maintain intact entire regions of the chromosome and would have better performance with respect to a uniform crossover.

Conclusion

In this paper, the speed of convergence of two training methods for neural networks has been systematically tested. The first method was standard backpropagation and the second method was a hybrid training method. Hybrid training methods take advantage of the genetic algorithms' ability to search through vast weight spaces. After the genetic algorithm has found the basin of attraction of the set of weights with minimum error, a gradient-based algorithm, such as backpropagation, is used to fine-tune the network. The results support previous findings[1], using a genetic algorithm to find a good starting point for backpropagation improves the performance and the speed of the training process. For all three tasks, the number of epochs of backpropagation required to meet the learning

condition was lower than using only backpropagation. This result has been obtained using the simplest form of genetic algorithm and neural network, nonetheless, it was possible to prove the benefits of the BA/BP approach. Looking at the combined number of epochs, it appears the only for the two-spiral problem this number was lower than the number of epochs needed by backpropagation to reach the same error. From this result, it is possible to deduce that the GA/BP approach is really beneficial when the error landscape is very complex and the probability of being trapped in a local minimum is high. In these scenarios, the hybrid training approach is much faster than the standard approach (up to 50% faster in the two-spirals problem).

Current applications of genetic algorithms range from credit control to medicine. Bin Sang proposed a method for evaluating the credit risk of supply chain finance using a genetic algorithm and backpropagation[11]. Hybrid training has been proved to have better performance in predicting and evaluating the parameters of the drug-loaded microsphere preparation process[12]. The hybrid approach can be applied to many different types of network. More complex networks, such as CNN and SAE, have been shown to benefit from hybrid training [13]. Moreover, genetic algorithms can be used to tune different aspect of a neural network. All types of hyperparameters can be evolved, from the architecture of the network to the learning rule, as shown in [14] and [15].

References

[1] R. K. Belew, J. McInerney, and N. N. Schraudolph, “Evolving networks: Using genetic algorithm with connectionist learning,” Comput. Sci. Eng. Dep. (C-014), , Univ. of California, San Diego, Tech. Rep. CS90-174 (revised), Feb. 1991.

[2] W. Yan, Z. Zhu, and R. Hu, “Hybrid genetic/BP algorithm and its application for radar target classification,” in Proc. 1997 IEEE National Aerospace and Electronics Conf., NAECON. Part 2 (of 2), pp. 981–984.

[3] Xin Yao, "Evolving artificial neural networks," in Proceedings of the IEEE, vol. 87, no. 9, pp. 1423-1447, Sept. 1999, doi: 10.1109/5.784219.

[4] Dalmia S., Sriram A., Ashwin T.S. (2021) Genetic Algorithm-Based Optimization of Clustering Data Points by Propagating Probabilities. In: Mandal J.K., Mukherjee I., Bakshi S., Chatterji S., Sa P.K. (eds) Computational Intelligence and Machine Learning. Advances in Intelligent Systems and Computing, vol 1276. Springer, Singapore. https://doi.org/10.1007/978-981-15-8610-1_2

[5] Harvey I. (2011) The Microbial Genetic Algorithm. In: Kampis G., Karsai I., Szathmáry E. (eds) Advances in Artificial Life. Darwin Meets von Neumann. ECAL 2009. Lecture Notes in Computer Science, vol 5778. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21314-4_16

[6] Martin Riedmiller. Advanced supervised learning in multi-layer perceptrons from backpropagation to adaptive learning algorithms. Computer Standards and Interfaces Special Issue on Neural Networks, 16(3):265–275, 1994

[7] Martin Riedmiller and Heinrich Braun. A direct method for faster backpropagation learning: The RPROP Algorithm. In IEEE International Conference on Neural Networks 1993 (ICNN93), San Francisco, pages 586–591, 1993.

[8] Dilip Sarkar. Methods to speed up error back propagation learning algorithm. ACM Computing Surveys, 27(4):519–542, 1995.

[9] Fahlman, S.E. (1989). Faster-learning variations on back-propagation: An empirical study. In D. Touretzky, G. Hinton & T. Sejnowski (Eds.), Proceedings of the 1988 Connectionist Models Summer School (pp. 38-51). San Mateo, CA: Morgan Kaufmann.

[10] Spector, L., Klein, J.: Trivial Geography in Genetic Programming. In: Yu, T., Riolo, R.L., Worzel, B. (eds.) Genetic Programming Theory and Practice III, pp. 109–124. Kluwer Academic Publishers, Boston (2005)

[11] Bin Sang, Application of genetic algorithm and BP neural network in supply chain finance under information sharing, Journal of Computational and Applied Mathematics, Volume 384, 2021

[12] Zhang, Xujing & Zhou, Jianping & Xu, Yan. (2020). Optimized parameters for the preparation of silk fibroin drug-loaded microspheres based on the response surface method and a genetic algorithm–backpropagation neural network model. Journal of Biomedical Materials Research Part B: Applied Biomaterials. 109. 10.1002/jbm.b.34676.

[13] Serhat kilicarslan, mete celik, şafak sahin, hybrid models based on genetic algorithm and deep learning algorithms for nutritional anemia disease classification, biomedical signal processing and control, volume 63, 2021

[14] N. J. Radcliffe, “genetic neural networks on mimd computers (compressed edition),” ph.d. Dissertation, dep. Theoretical phys., univ. Edinburgh, u.k., 1990.

[15] David j. Chalmers, the evolution of Learning: An Experiment in Genetic Connectionism, Editor(s): David S. Touretzky, Jeffrey L. Elman, Terrence J. Sejnowski, Geoffrey E. Hinton, Connectionist Models, Morgan Kaufmann, 1991, Pages 81-90,