Empirical Studies on the Speed of Convergence of Neural Network Training using Genetic Algorithms

Hiroaki Kitano
Center for Machine Translation
Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.
hiroaki@cs.cmu.edu

Abstract

This paper reports several experimental results on the speed of convergence of neural network training using genetic algorithms and back propagation. Recent excitement regarding genetic search lead some researchers to apply it to training neural networks. There are reports on both successful and faulty results, and, unfortunately, no systematic evaluation has been made. This paper reports results of systematic experiments designed to judge whether use of genetic algorithms provides any gain in neural network training over existing methods. Experimental results indicate that genetic search is, at best, equally efficient to faster variants of back propagation in very small scale networks, but far less efficient in larger networks.

1 Introduction

Genetic algorithms are an efficient searching method inspired by the principle of natural selection. There are a few studies on the efficiency of search by genetic algorithms, including a few reports applied to neural network training [Montana and Davis, 1989] [Whitley and Hanson, 1989], and some reports claim that genetic algorithms are more efficient than back propagation [Montana and Davis, 1989]. However, it is controversial whether application of genetic algorithms to train neural networks can be an efficient alternative to gradient-descent methods such as back propagation. One cause for this confusion is that we have not seen any systematic investigation of the efficiency of neural network training using genetic algorithms.

In [Whitley and Hanson, 1989] there are reports that show convergence characteristics change if population size changed or an adaptive mutation method was used, but no comparison has been made as to whether their method provides any faster convergence than back propagation or its variants. [Montana and Davis, 1989] reported that training using genetic algorithms was substantially faster than back propagation. However, lack of description on the specific back propagation method and its parameters, and inaccessability to their task domain made their experiments unreplicable by other researchers.

This paper is, perhaps, the first attempt to systematically evaluate the efficiency of genetic algorithms for neural network training. The goal of our experiments is to draw an overall picture as to relative strengths of back propagation and genetic algorithms for neural network training, and to evaluate the speed of convergence of both methods. We designed our experiments to be replicated by other researchers so that our report could serve as a common forum for discussion to resolve this issue. For this purpose, we decided to use the XOR problem, various scales of encoder/decoder problems, and the two-spirals problem as tasks by which we evaluate the speed of convergence. Of course, we recognized the need to assess this with large and real-world domains, and thus we also conducted experiments in phoneme recognition tasks. However, reports from such a domain are not replicable by other researchers because the training data and the network for the task are not easily accessible. Thus, we report our experiments from the XOR, various encoder/decoder problems and the two-spirals problem which, at least, serve as a starting point for further systematic evaluations.

Although there are applications of genetic algorithms for neural network designing as seen in [Miller et. al., 1989] [Harp et. al., 1989] and [Stork et. al., 1990], we focus on weight training in this paper. This is because neural network designing tasks can be subsumed by weight training where weight values in a specific value range or connectivity bits represent unconnected links. This view is biologically plausible since real connection strengths between neural groups can be determined by the distribution of cell and substrate adhesion molecules (CAMs and SAMs)[Edelman, 1987]. Computationally, convergence characteristics of weight training and configuration design have similar property since the only differences between them is the size of search spaces.

This paper has three parts. First, we examine characteristics of genetic algorithm-based weight training using the XOR problem. As we will describe later, we found that simple use of genetic algorithms for neural network would generally be outperformed by back propagation. Second, we propose and examine the GA-BP method as a remedy for the problem of local search. The GA-BP method combines genetic algorithms and back propagation to offset problems of local search. We found that the GA-BP method consistently converges faster than genetic algorithms alone. Here, we found that as network size gets bigger, convergence of genetic algorithms degrades to the extent that even the GA-BP method underperforms back propagation and its faster variants. The third part of the paper is central. We examine convergence characteristics using various scales of encoder/decoder problems and the two-spirals problem to examine whether genetic algorithms converge faster than back propagation in its early stages of training. While the weakness of genetic algorithms in local fine-tuning is obvious, although this problem was circumvented in the GA-BP method, the speed of convergence in the early stages of training is the critical factor in evaluating their utility as a method of training neural networks.

2 Genetic Algorithms: Overview

Genetic algorithm is a kind of stochastic search process that starts from a set (or a population) of finite string representations, called chromosomes, each of which maps a possible solution to the problem. Given an evaluation function, a reproduction probability for each chromosome is assigned based on the fitness of that chromosome. Chromosomes with higher fitness values will have higher reproduction probabilities. This simulates the survival of the fittest. Two operations are performed stochastically: crossover and mutation. The crossover operation chooses two chromosomes from a current population as parents, and reproduces descendants by mixing chromosomes. Since this is a probabilistic process, and not all chromosomes are crossovered, a parameter called a crossover parameter defines probability. Mutation is the random change of an element of a chromosome. As a result, a new population is formed as a descendant of the current population. The next generation will be formed by going through evaluation, crossover, and mutation processes. For detail, refer to [Goldberg, 1989].

In our experiments, each chromosome represents a distribution of weights of each network. Thus, a population of chromosomes is a population of neural networks with different weight distributions. Since we employ a real value encoding, a chromosome is a list of real values each of which maps onto a connection weight. We expect chromosomes which represent highly optimized weight distribution to dominate the population through genetic search processes.

3 Problems of Genetic Algorithms for Neural Network Training

There are two major problems in applying genetic algorithms to training neural networks: weakness in fine-tuned local search, and a trade-off between population size and speed of convergence.

The weakness of genetic algorithms in performing fine-tuned local search is widely recognized. Although genetic algorithms exhibit very fast convergence to a point of approximate solution in a search space, a genetic algorithm itself does not entail a mechanism for local fine-tuning as seen in back propagation. When a population reaches a state where it is dominated by the best chromosome, finding a better solution must depend upon mutations. This would result in very inefficient search.

We have confirmed this problem by training a neural network with a genetic algorithm. In figure 1, movements of sum square error are shown for back propagation, a serial genetic

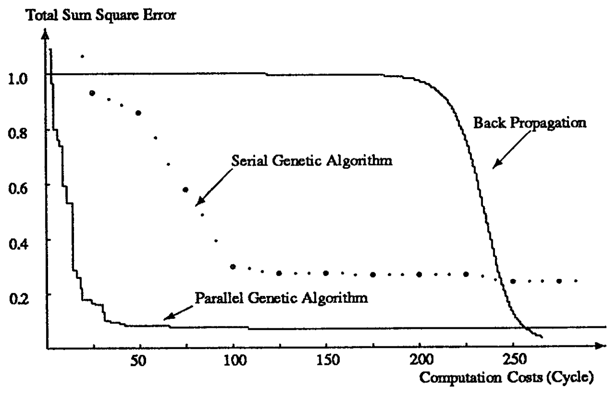

Figure 1: Convergence of GA- and BP-based training

algorithm, and a parallel genetic algorithm. Although genetic algorithms successfully reduce the error measure at the beginning of the training, the speed of convergence was drastically decreased in searching for a near-optimal solution. Eventually, back propagation outperformed genetic algorithm-based training. This characteristic of genetic algorithms would not be a problem when the required converge criteria of the error measure is relatively high, but this assumption is often not the case.

One way to solve this problem is to have a larger population. [Whitley and Hanson, 1989] reported that an error measure was reduced to 0.0025 in the XOR problem using populations of 100 and 200, but not with a population of 50. We have also confirmed these results. However, the problem of using a larger population is that it requires extensive computation for each generation, and, even though the outcome converges into an optimal solution in a few generations, genetic algorithms using large populations would be outperformed by back propagation.

4 Combining Genetic Algorithms and Back Propagation: The GA-BP Method

We propose the GA-BP method as a remedy for this problem. In the GA-BP method, neural networks are trained in two stages. First, genetic algorithms train weights and biases of nodes to locate a point in a weight-bias space which is close to the optimal solution. Then, back propagation starts from that point, and conducts an efficient local search. This combination would be an efficient method of training neural networks because it takes advantage of the strengths of genetic algorithms and back propagation (the fast initial convergence of genetic algorithms and the powerful local search of back propagation), and circumvents the weaknesses of the two methods[1] (the weak fine-tuning capability of genetic algorithms and a flat spot in back propagation). Since the problem of weak fine-tuning capability has been circumvented by employing back propagation for local search, the

[1]Ideas of combining local search methods with genetic algorithms are not new. It has been discussed in literatures such as [Grefenstette, 1987] and [Ackley, 1987].

central issue is whether the speed of convergence of genetic algorithms outperforms the speed of convergence of back propagation in locating a point near the solution.

4.1 Experimental Settings

We have tested the GA-BP method using the XOR problem and the 4-2-4 encoder/decoder problem. For the XOR problem, we used the 2-2-1 feedforward network.

In genetic algorithms, we represent the network with an array of floating point numbers, each of which corresponds to weights and biases. At the initialization, these floating point numbers are randomly generated in the range of ± 5.0, and follow a simple distribution. For the XOR problem and the 4-2-4 encoder/decoder problem, we assigned higher probability in the range from +2.5 to +5.0, and from -2.5 to -5.0. A range from +2.5 to -2.5 has a lesser chance of being generated. This distribution of initial values was decided based upon heuristic observation of weights and biases in the final state of the networks, and adjusted for each network. It is important to ensure that the optimal value of weights and biases is covered by the range of initial distribution. The size of the population is 50 unless otherwise stated. To evaluate the fitness of each chromosome, feedforward computation is performed by presenting patterns which consist of one epoch. Total sum square (TSS) error is used as an error measure. The fitness of a chromosome is: fitness = $1/\mathrm{TSS}^2$. Reproduction strategy is a proportional reproduction which normalizes fitness, and assigns higher reproduction probability for higher fitness chromosomes. In addition, we introduced an elitist reproduction which always chooses the two best chromosomes and simply copies them, without crossover or mutation, to the population of the next generation. Crossover is either a single crossover or multiple crossover (two point or more) with a probability of 50.0%. Mutation probability is 5.0% for a static mutation unless otherwise stated. For an adaptive mutation, the value changes depending upon a similarity of paired chromosomes. For this experiment, we used two-point crossover, adaptive mutation, proportional reproduction based on a fitness measure, and elitist copy strategy.

After training by the genetic algorithm is completed, the best chromosome is sent to a back propagation module. The genetic algorithm stage was terminated when total sum square error was reduced to less than 0.6. This is because for chromosomes whose error measure is higher than this figure, significant numbers of trials did not converge. We used a standard back propagation as defined in [Rumelhart, Hinton and Williams, 1986]. Learning parameters were default values of [McClelland and Rumelhart, 1988]. The learning was judged to have converged when the TSS measure went below 0.04 unless otherwise specified. The continuous weights update mode was used which conducts error back propagation after each presentation of the pattern. For back propagation, the initial values of weights and biases were randomly generated within a range ± 1.0, instead of ± 5.0. This is because if we generate initial weights and biases with a range of ± 5.0, more than 30% of the trials do not converge.

We have also used a quickprop, a faster variant of back propagation, proposed in [Fahlman, 1988]. Quickprop is known to be the fastest variation of back propagation at present. For quickprop, we used default parameters described in [Fahlman, 1988].

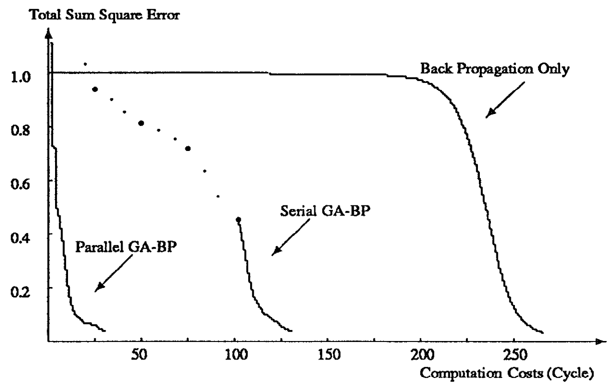

Figure 2: Convergence of GA-BP and BP only methods

In measuring computational costs, we counted one evaluation in a genetic algorithm as 1/2 an epoch of back propagation because the evaluation of TSS does not involve the error back propagation process; only feedforward is involved. Thus, evaluating one generation consisting of 50 chromosomes is considered to be equal to 25 epochs in back propagation.

4.2 Experimental Results

We ran the standard back propagation (BP), quickprop with hyper error (QP-Hyper), quickprop with sum square error (QP-Sum), serial GA-BP with varying population size (GA-BP-50 and GA-BP-20), and parallel GA-BP with varying threshold value (PGA-BP-0.6 and PGA-BP-0.3). For each learning method, we ran 20 trials, and the average value and standard deviation was computed to compare the speed of convergence and the stability of the method.

The result is shown in table 1, and typical convergence curves for parallel GA-BP, serial GA-BP, and standard back propagation are shown in figure 2. ECC is an expected convergence cycle, and STD is a standard deviation.

In table 1, an experimental set of GA-BP-50 was conducted with a population size of 50, and an experimental set of GA-BP-20 with a population size of 20 was conducted, but with a very high mutation probability (50.0%). We used a relatively small population because the GA-BP method does not require a genetic algorithm-based process to converge into a very low sum square error measure. We only require it to converge down to 0.6. To this level of error measure, in the XOR problem, smaller populations converge faster than larger populations. It should be noted that for trials by quickprop 45% of the trials using sum square error, and 29% of trials using hyper error function did not converge within pre-decided limitations. These trials had to be restarted and numbers of iteration for unconverged trials were added to the total computation cost. PGA-BP-0.6 and PGA-BP-0.3 are parallel GA-BP methods with thresholds of stopping genetic algorithms of 0.6 and 0.3 in total sum square error, respectively. We also found that the threshold value of 0.6

	ECC	STD
BP	285.8	90.9
QP (Hyper)	91.9	60.9
QP (Sum)	226.6	123.6
GA-BP-50	130.9	72.4
GA-BP-20	100.5	37.5
PGA-BP-0.6	33.1	4.6
PGA-BP-0.3	46.7	28.2

Table 1: Results from the XOR problem

	ECC	STD
BP	700.2	197.4
QP	21.1	2.1
Serial GA-BP	5087.4	603.6
Parallel GA-BP	136.2	74.1

Table 2: Results from the 4-2-4 encoder/decoder problem

converges faster than the 0.3 setting. This is due to the fact that the genetic algorithm slows down its convergence speed if it needs to fine-tune weights and biases, and the speed of convergence of the back propagation process is not significantly reduced even if it starts from 0.3 of the TSS measure.

Next, we carried out experiments with the 4-2-4 encoder/decoder problem. In the genetic algorithm stage, it took an average of 58 generations to converge to 0.6 sum square error with a population size of 200. Trials with a population size of 50 did not converge to 0.6 within 200 generations. In table 2, we show the results of 10 trials. Data for back propagation is the average of the trials which converged: almost 20% of trials did not converge within 1500 cycles and these trials were simply ignored. The convergence criteria was 0.16 in TSS allowing 0.01 TSS error for each output unit per pattern.

In the 4-2-4 encoder/decoder problem, the GA-BP method was outperformed by quickprop. There are two possible explanations for this. First, the threshold to stop genetic algorithms is too low for this task so that efficiency was lost at the plateau. The second possibility is that quickprop is faster even in the initial search of near-optimal points. If the latter is the case, use of genetic algorithms for neural network training could generally be concluded to be inefficient compared to faster variants of back propagation.

In the following section, we describe our experiments to critically test whether genetic algorithms can converge faster in the initial stage of training.

5 Scaled Up Experiments

In order to examine the possibility that quickprop could be faster than genetic algorithms even in initial stage of training, we have conducted a series of experiments on the encoder/decoder problems ranging from 2-2-2 to 16-4-16 and the two-spirals problem. Chromosome length for these experiments ranges from 11 to 165 for encoder/decoder problems, and 138 for the two-spirals problem. Experimental settings are the same as stated in the previous section. Since

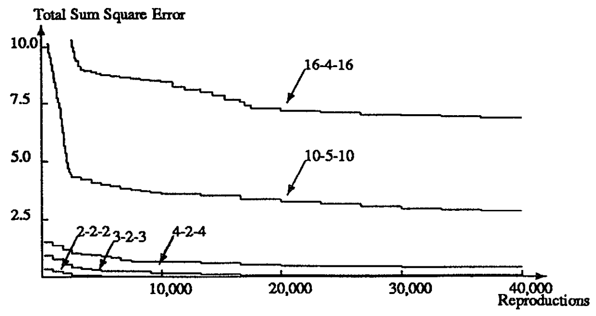

Figure 3: Total sum square error with GA-based training

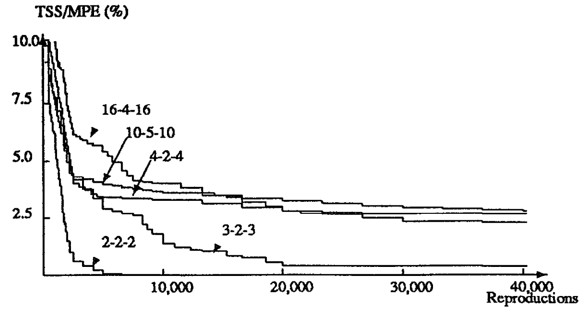

Figure 4: Percentage of total sum square error against maximum potential error

we are going to compare genetic algorithms with a quickprop which is an optimized version of back propagation, we have also optimized genetic algorithms using multi-point crossover, and an adaptive mutation rate method. Although these methods improved performance of genetic algorithms, its effects were not significant enough to alter the essential results of our experiments.

5.1 The Encoder/Decoder Problems

The encoder/decoder problems are one of the standard benchmarks for neural network learning algorithms. An *"N-M-N encoder/decoder"* means a network of three-layers with N units for input and output layer and M units for a hidden layer. The network is given with N distinct input patterns, each of which only one bit is turned on, and all other bits are turned off. The network should duplicate the input pattern in the output units.

Figure 3 shows total sum square error for each task plotted along with numbers of reproductions required [2]. These are expected values averaged over 10 trials each. Clearly, TSS measure degrades as the size of the network increases.

Figure 4 shows the percentage ratio of TSS compared to maximum potential TSS error. It should be noticed that the size of the network did not alter the speed of convergence

[2]For a population consists of 50 individuals, evolving one generation requires 50 reproductions, for most cases, involving recombination and mutation. Thus, 10,000 reproduction means 200 generations with a population of 50 individuals.

Tasks	Chromo. Length	Max. Error	GA	QP
2-2-2	14	4.0	133	8.8
3-2-3	20	9.0	2950	9.2
4-2-4	26	16.0	3800	9.6
8-3-8	67	64.0	2800	9.7
10-5-10	125	100.0	1150	2.4
16-4-16	164	256.0	1950	7.2

Table 3: Convergence speed to 10% of MPE

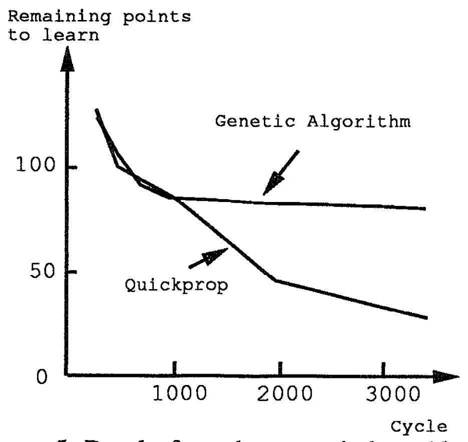

Figure 5: Results from the two-spirals problem

measured by percentage bases, except with very small size networks such as the 3-2-3 problem and the 2-2-2 problem which converged into very low error measure.

Table 3 shows expected numbers of reproductions for genetic algorithms to converge at 10% of the maximum potential error (MPE). Length of the chromosome does not seem to have a strong correlation with the speed of convergence, but, again smaller networks such as the 2-2-2 problem converge much faster than in other tasks. Expected convergence cycles for quickprop to converge at 10% of the maximum error are shown. Quick prop was run with sum square error, not with hyper error which is faster than with sum square error, and still converged much faster than genetic algorithms. In addition, substantial numbers of trials from points specified by genetic algorithms were trapped in the local minima when the error measure of the point which stops genetic algorithm stage was more than 10% of the maximum potential error.

5.2 The Two-Spirals Problem

The two-spirals problems is a task that, for each 194 training points belonging to one of two interwined spirals, a network must tell in which spiral a given point belongs to. This is a hard task for a back propagation due to its complicated error landscape. We use a training point generation program and a network described in [Lang and Witbrock, 1988]. The network is a five-layered fully connected configuration with short cut connections. The network consists of 19 nodes and, counting threshold of each node, there are 138 trainable weights.

In figure 5, plots of average numbers of points remain to be learned from 10 trials are shown. Although the task is extremely non-linear, quickprop outperforms GA-based training. For this experiment, network parameters used are the same as [Lang and Witbrock, 1988]. A population size for GA-based training is 50. Adaptive mutation and multiple crossover are used. Change in population size, mutation rate, number of crossover points and its probability, and initial weight distribution did not alter basic results of the experiments.

6 Discussions

In essence, our experiments demonstrated the following three points:

- Although genetic algorithms converge efficiently at the initial stage, the speed of convergence is severely undermined later due to its weak local fine-tuning capability.
- Use of gradient-descent procedure, such as back propagation, from the point specified by a chromosome generally provides faster convergence than starting from points determined at random.
- Speed of convergence of genetic algorithms for larger networks is very slow, and the point of convergence is shifted upward. Thus, substantial numbers of trials for back propagation from these points ran the risk of being trapped by local minima.

One clear message from our experiment is the need for faster convergence methods of genetic algorithms. Although weakness of local fine-tuning can be overcome by combining a gradient-descent scheme, such as back propagation, and a genetic algorithm as seen in the GA-BP method, if genetic algorithms' initial convergence is slower than gradient-descent schemes, the utility of the genetic algorithm can hardly be seen. Unfortunately, for a range of problems we have tested here, this was the case. Convergence of genetic algorithm based neural network training was so slow that it was consistently outperformed by quickprop. Varying parameters such as mutation rate, crossover rate, and population size, or the introduction of faster convergence methods, such as adaptive mutation and multipoint crossover, made only limited contributions in reversing the results.

However, it should be noted that our experiments were conducted only on relatively small tasks with rather simple error landscapes. It is conceivable that for this class of problem, a gradient-descent method works very efficiently. This is especially true for encoder/decoder problems. But, this may not be the case where the error landscape has many local minima and pathetic curvature so that the speed of convergence of the gradient-descent method degrades substantially, and even runs a high risk of being trapped in the local minima. For this class of tasks, genetic algorithms may have a reasonable chance of outperforming even faster variants of back propagation. In fact, [Ackley, 1987] reports genetic algorithms have a better chance of outperforming hillclimbing on tasks with many local mimina. However, it should be noticed that our experiments include the two-spiral problems which is expected to be hard for gradient-descent methods. The fact that GA-based training was outperformed by quickprop even

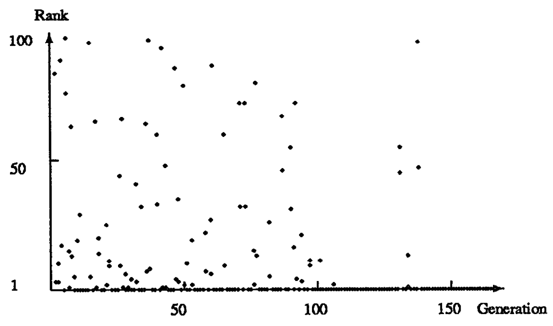

Figure 6: Distribution of parent chromosomes

in the two-spirals problem indicates that existence of strong non-linearity itself is not a sufficient condition to judge that the task is suitable for the GA-based training. We are currently investigating whether the GA-BP method outperforms faster variants of back propagation in the real-world domain — phoneme recognition with time-delay neural networks.

It is conceivable, however, that such a task as weight training of neural network is not a good task for genetic algorithms due to its interdependencies between regions of chromosomes. In neural network, modification of weights in a part of the network affects outcomes of other parts through activation or inhibitory links. This property of neural networks counters to a basic idea of genetic search that a partial solution in a lower-dimensional subspace is likely to be a part of a good overall solution. Thus, combining good chromosomes does not necessarily results in a better chromosome. It is an interesting question to ask to what extend a *compositionality* of solution is kept or disturbed in the neural network training task. In order to quantitatively analyze contribution by good partial solutions, we took statistics of ranking distribution of parents which created the best chromosome in each generation. In the figure 6, rank-based distribution of parents which created the best chromosome in each generation is mapped. This distribution is taken from one of the runs for the XOR problem.

If a partial solution is likely to be a part of an overall good solution, there should be strong bias of distribution toward high ranking (lower half of the graph) chromosomes. 59.3% of the parents are within the top 25 range in rank[3]. Distribution between the range from 26th to 100th was almost uniformal. Obviously, we have statistically significant bias toward higher ranking parents, although almost 40% of contribution comes from not-so-high ranked parents. However, since we are using the proportional reproduction rule, which is $1/TSS^2$, the above figure does not answer the question whether higher ranking chromosomes has a better chance of contributing to optimization. We would further analyze this by measuring an *effective reproduction rate*. The *effective reproduction rate* is a ratio of reproduction which created the best chromosome out of an entire reproduction.

[3]This exclude the dead copy of the best chromosome of the previous generation, because it merely shows any new recombination did not gain better fitness than existing one's – only the parents of chromosomes which updated the best fitness should be counted.

Range	2 - 25	26 - 50	51 - 75	76 - 100
XOR	0.15	0.11	0.05	0.09
Linear Bits	0.47	0.11	0.11	0.07

Table 4: Effective reproduction rate

Table 4 show the effective reproduction rate for each range of chromosomes in the XOR problem and a linear bits problem. The linear bits problem is a task that for N-length bit string, obtain a string with maximum numbers of bits on. Thus, in this problem partial solution is always a part of a better overall solution. In the XOR problem, although there is a statistically significant bias toward high ranking chromosomes (0.15%), there is substantial contribution from low ranking chromosomes. The linear bits problem is more dependent upon high ranking chromosomes (0.47%). It seems that a strong interdependency of the neural network task suppresses an effective reproduction rate of high rank chromosomes, thus overall speed of convergence is slower than that of weak- or non-interdependent tasks.

It should be noticed that successful reports on applications of genetic algorithms come mostly from the tasks having the compositional property assumed in the genetic algorithms. From this aspects, using genetic algorithms for neural networks design would be more promising than weight training because well-formed local circuits are expected to be a part of more functional structure.

The fact that genetic algorithms are not capable of conducting local fine-tuning has interesting biological implications. Supposing our somatic system exhibits similar characteristics, it is computationally plausible that genetically encoded information on neural circuitry and connectivity is not sufficient to create an optimally tuned neural system.

First, the fact that genetic algorithms alone are not capable of precisely determining weights of networks indicates that some gradient-descent type or equivalent learning scheme is needed. By the same token, genetic algorithms may not be able to fully optimize a network even in neural network design tasks. A need for a supplementary scheme for fine-tuning is clear.

Second, however, simply combining genetic algorithms and local fine-tuning may not provide sufficient optimization. We might have to insert a more flexible scheme for adaptation in between genetic search and local gradient-descent search scheme. This can be concluded from two facts: (1) degradation of convergence of genetic algorithms on larger scale problems, and (2) many trials from points specified by genetic algorithms, in these degraded cases, are trapped in the local minima.

Fortunately, our neural system has such mechanisms. One is a scheme such as that modeled on the theory of neural group selection or *neural darwinism* [Edelman, 1987], and the other is synaptic weight modification and plasticity. Assuming that our experimental results could be considered as being a simpler, but analogical simulation of evolution and development of the brain, a cascaded scheme of natural selection, adaptation, and learning is a necessary consequence for survival.

7 Conclusion

In this paper, we have reported results of systematic experiments designed to compare the speed of convergence for training neural networks using genetic algorithms and gradient descent methods. At the outset of this paper, we pointed out that simple application of genetic algorithms would be outperformed by back propagation due to its weak local fine-tuning capability and computational cost for evaluating numbers of chromosomes in one generation. Thus, instead of directly comparing genetic algorithms and back propagation, we have compared the GA-BP method and quickprop, both of which are faster methods of training neural networks.

As a rationale for using the GA-BP method, we have shown briefly that the GA-BP method converges consistently faster than genetic algorithms alone. However, experimental results using the XOR and the 4-2-4 encoding/decoding problems revealed a possibility that the scaling property of genetic algorithms is not desirable for training larger networks even in the initial stage of convergence. Thus, the critical test should examine whether genetic algorithms converge faster than back propagation and its faster variants in the initial stage of training.

Experiments with various scales of the encoder/decoder problems range from 2-2-2 to 16-4-16 and the two-spiral problem. As a result of these experiments, we discovered that the scaling property of genetic algorithms is undesirable for neural network training. Although the performance largely depends upon the error landscape and evaluation functions used, degradation of performance was so significant that even the parallel GA-BP method underperformed quickprop. Also, asymptotic convergence curves indicate genetic algorithms require far greater numbers of recombination to converge to desired error criteria compared to the quickprop. On the other hand, we have a report by [Montana and Davis, 1989] indicating that neural network training by genetic algorithms converge much faster than training by back propagation. This conflict needs to be resolved by carrying out fair and well controlled experiments.

As discussed in the previous section, from the scope of our experiments, we reached the conclusion that:

- Dramatically faster methods of convergence need to be discovered for genetic algorithms to be used as an efficient alternative to faster variants of back propagation.
- The neural network training task was comfirmed to have a less compositional feature — combinations of good partial solutions do not necessarily create a better overall solution. Thus, the speed of convergence of genetic algorithm-based training was significantly undermined.
- The relation between the convergence speed of genetic algorithms and back propagation against various classes of error landscape needs to be systematically investigated to identify cases where genetic algorithms perform better than back propagation. By identifying classes of tasks and error landscapes which genetic algorithm based neural network training could perform better than back propagation or its faster variants, we can resolve conflicts of results between successful and faulty reports, and this would lead to appropriate use of genetic algorithms for neural network training.
- The fact that genetic algorithms fail to converge into optimal points in larger networks has interesting biological implications. Assuming our model simulates, even in a very simplified manner, real evolutional process, the convergence property discovered computationally supports neural plasticity, such as neural group selection and weight modification, as a necessary mechanism to sustain the survival of our spices.

References

[Ackley, 1987] Ackley, D., *A Connectionist Machine for Genetic Hillclimbing*, Kluwer Academic Publishers, 1987.

[Edelman, 1987] Edelman, G., *Neural Darwinism: Theory of Neuronal Group Selection*, Basic Books, New York, 1987.

[Fahlman, 1988] Fahlman, S., *An Empirical Study of Learning Speed in Back-Propagation Networks*, CMU-CS-88-162, Carnegie Mellon University, 1988.

[Goldberg, 1989] Goldberg, D., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, 1989.

[Grefenstette, 1987] Grefenstette, J., "Incorporating Problem Specific Knowledge into Genetic Algorithms," In Davis, L. (Ed.), *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann, 1987.

[Harp et. al., 1989] Harp, S., Samad, T. and Guha, A., "Towards the Genetic Synthesis of Neural Networks," In *Proceedings of the Third International Conference on Genetic Algorithms*, 1989.

[Lang and Witbrock, 1988] Lang, K. and Witbrock, M., "Learning to Tell Two Spirals Apart," In *Proceedings of the 1988 Connectionist Models Summer School*, 1988.

[McClelland and Rumelhart, 1988] McClelland, J. L. and Rumelhart, D. E., *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises*, The MIT Press, 1988.

[Miller et. al., 1989] Miller, G., Todd, P. and Hegde, S., "Designing Neural Networks using Genetic Algorithms," In *Proceedings of the Third International Conference on Genetic Algorithms*, 1989.

[Montana and Davis, 1989] Montana, D. and Davis, L., "Training Feedforward Neural Networks Using Genetic Algorithms," In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-89)*, 1989.

[Rumelhart, Hinton and Williams, 1986] Rumelhart, D. E., Hinton, G. E. and Williams, R.J., "Learning Internal Representation by Error Propagation," In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, (Eds.) Rumelhart, D. E. and McClelland, J. L., The MIT Press, 1986.

[Stork et. al., 1990] Stork, D., Walker, S., Burns, M. and Jackson, B., "Preadaptation in Neural Circuits," In *Proceedings of the International Joint Conference on Neural Networks*, 1990.

[Whitley and Hanson, 1989] Whitley, D. and Hanson, T., "Optimizing Neural Network Using Faster, More Accurate Genetic Search," In *Proceedings of the International Conference on Genetic Algorithms*, 1989.