A better formatted version appears in the journal: *Neural Computing & Applications*, 2007, vol. 16, pages 209-226.

Using Evolution to Improve Neural Network Learning: Pitfalls and Solutions

John A. Bullinaria

School of Computer Science, University of Birmingham

Birmingham, B15 2TT, UK

j.a.bullinaria@cs.bham.ac.uk

Abstract: Autonomous neural network systems typically require fast learning *and* good generalization performance, and there is potentially a trade-off between the two. The use of evolutionary techniques to improve the learning abilities of neural network systems is now widespread. However, there are a range of different evolutionary approaches that could be applied, and no systematic investigation has been carried out to find which work best. In this paper, such an investigation is presented, and it is shown that a range of evolutionary techniques can generate high performance networks, but they often lead to unwanted side effects, such as occasional instances of very poor performance. The nature of these problems are explored further, and it is shown how the evolution of age dependent plasticities and/or the use of ensemble techniques can alleviate them. A range of techniques are thus identified, with differing properties, that can be matched to the specific requirements of each application.

Keywords: Neural networks, Learning algorithms, Evolutionary computation, Age dependent plasticity, Ensemble methods.

1. INTRODUCTION

The power of neural network systems is now well known, but getting them to operate efficiently requires the specification of a number of details that tend to be rather application dependent [1]. For example, a simple feed-forward network learning particular classes of input-output mapping requires appropriate architecture, initial weight distributions, learning rates, regularization parameters, and so on, to maximize its performance. Which aspect of performance is important, and how to measure it, will also be application dependent. Usually it will be good final generalization performance that is most relevant, but often fast learning is also required, particularly for real-time autonomous systems. This paper considers the problem of how to set up neural network systems that learn to generalize as well as possible, as quickly as possible, from data drawn randomly from particular classes of data distributions. The aim is not to introduce new neural network learning algorithms, but to study the techniques that might be used to make sure that existing learning algorithms (e.g., standard gradient descent) are working most effectively, particularly by finding good values for the various associated parameters.

Given the noise in neural network learning performance measurements (due to the random initial weights, choice of data samples, and such like) a population based optimization approach tends to work better than attempting to build the models by hand, or by application of gradient descent style changes to the learning parameters. Consequently, an increasingly common and successful approach has been to use simulated natural selection to evolve such systems. The evolutionary neural network field now spans an enormous range of approaches covering the evolution of all aspects of the networks and their learning algorithms [2, 3], and application areas are as diverse as pattern recognition [4], VLSI system design [5], vehicle control [6], and board games like Go [7]. However, we are concerned here with just one particular aspect, namely the evolution of neural networks that learn quickly to generalize well. There appear not to have been any general investigations into which evolutionary approaches are best to use for this, nor any checks for potentially deleterious side-effects of the evolution, though comparative studies of other factors do exist (e.g. [3, 8, 9, 10, 11]). This paper attempts to fill this gap, with particular emphasis on the potential pitfalls of the most obvious evolutionary approaches, and their associated solutions.

It will become clear that the interaction of learning and evolution can result in unfortunate side effects whereby some of the natural approaches lead to the evolution of rather risky learning strategies that sometimes result in very poor performance. Of course, for applications employing a whole population of essentially disposable autonomous devices, occasional instances of disastrous performance may not be too problematic. However, if one had a single autonomous system, sent at great expense to another planet (say), the possibility of an occasional lapse that could lead to its destruction would obviously be unacceptable. For such applications, one clearly needs to augment the basic evolutionary neural network approaches with other techniques that can alleviate any learning difficulties.

The remainder of this paper begins by describing the type of neural network systems to be studied, and outlining the broad classes of evolutionary approaches appropriate for optimizing their learning parameters. The results from a systematic series of computational experiments investigating the advantages and disadvantages of each evolutionary approach are then presented. The emergence of problematic risky learning strategies then becomes apparent, and further simulations are described that explore two promising techniques for alleviating those problems. First the idea of age dependent plasticities inspired by the critical periods found in human learning, and second a range of ensemble techniques that work on the principle that a committee will be able to out-vote any poorly performing individuals. The paper ends with some general discussion and conclusions.

2. THE NEURAL NETWORK SYSTEMS

For concreteness, this study will be restricted to standard fully connected feed-forward networks of sigmoidal units with one hidden layer trained using gradient descent to perform simple classification tasks. The standard weight w_{ij} update equation for each training data sample n is therefore

$$\Delta w_{ij}(n) = -\eta_L \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(n-1) \qquad (1)$$

where E is the cost function that measures the network's error on the training data sample [1]. As usual, the training starts from random initial network weights, which are typically chosen from appropriate uniform or Gaussian distributions. Previously [12] it has been demonstrated empirically that the networks can learn significantly faster if they are allowed different learning parameters for each of their four distinct components L (the input to hidden weights IH, the hidden unit biases HB, the hidden to output weights HO, and the output unit biases OB). Consequently, to ensure that the networks can learn at their full potential, each has four learning rates η_L, a momentum parameter α, and initial weights $w_{ij}(0)$ generated randomly from the four uniform ranges $[-r_L, +r_L]$.

For classification problems, it is appropriate to use the cross-entropy cost function [1, 13], and add a standard weight decay regularization term to prevent over-fitting [1], so

$$E = -\sum_j \left[t_j.\log(o_j) + (1-t_j).\log(1-o_j) \right] + \lambda \sum_{k,l} (w_{kl})^2 \tag{2}$$

which results in the output layer weight derivatives

$$\frac{\partial E}{\partial w_{ij}} = h_i.(t_j - o_j) + \lambda w_{ij} \tag{3}$$

where t_j are the binary target network outputs, o_j are the actual outputs, h_i are the hidden unit activations, and λ is the regularization parameter.

The aim here is to evolve appropriate values for the various learning parameters, rather than attempting to set them by hand, and then use the gradient descent algorithm (1) with those parameters to learn the network weights. Clearly there would be little point in evolving a system just to learn one particular training set quickly, since it would be quicker overall to tolerate a slow learner than to wait for a lengthy evolutionary process to finish. What one generally wants to do is evolve a system that can learn to generalize quickly from randomly drawn data samples in a novel environment, that is itself drawn randomly from a whole class of possible environments. Since most real world classification tasks involve learning non-linear classification boundaries in a space of real valued inputs, this paper uses the class of simple training data sets in which each set has two inputs normalized into a continuous two dimensional space $[0.0, 1.0]^2$, and two outputs representing classes specified by random circular classification boundaries in that input space. Three typical data distributions of this type are shown in Figure 1. This set-up is simple enough to allow extensive simulations in a reasonable time, yet covers many of the crucial features and difficulties of real world applications.

Each individual neural network is given its own randomly chosen data set, and during its lifetime experiences and learns from a sequence of new random samples from that distribution. The aim of this *online learning* process is to acquire quickly the ability to generalize from previous data samples to each new data sample [14], i.e. to perform well on each new data sample *before* learning from it and moving on to the next sample. The evolution is expected to produce the online learning parameters that result in good generalization ability as quickly as possible. Since the networks will presumably evolve to avoid over-fitting of the training data, the gradient descent error function is not necessarily the best measure of evolutionary fitness. A better and obvious performance measure is in terms of the percentage of network output classifications that are deemed *correct* (e.g., output activations within 0.2 of their binary targets), and these will typically be averaged over a particular number of training samples representing the experience during a fixed length period of an individual's life. Throughout, the network performances are conveniently displayed in terms of "error counts", the percentage of classifications that are *incorrect*.

3. APPROACHES FOR THE SIMULATED EVOLUTION

The objective of this paper is to explore the best approaches for improving neural network learning by using simulated evolution by natural selection. The general idea is to take a whole population of individual neural network instantiations, and allow them to learn, procreate and die in a manner inspired by natural (biological) systems [2]. Each individual is 'born' with a genotype that specifies all the appropriate innate parameters (including the learning rates and random initial weight distributions), and is derived from the genotypes of its two parents via 'crossover' and random mutations. Then, throughout its 'life', each individual learns from its environment how best to adjust their weights to perform most effectively. The fittest individuals parent a number of children. Eventually, all individuals 'die' and are removed from the population. If natural selection is successfully simulated, good innate properties (e.g., parameter values) will tend to proliferate in the population, and poor ones will be lost.

For biological individuals, the ability to survive or reproduce will generally be a complex function of their performance over a range of tasks (fighting, fleeing, feeding, and so on). Current neural network systems do not involve such high level abilities, so a simpler measure is required, and it proves sufficient to use the single task performance as the survival or procreation fitness. It is also necessary to use simplified versions of genetic encoding and recombination. A number of previous studies [13, 15, 16, 17] have shown that it is sufficient to use direct real valued coding of each evolvable parameter, and a simple recombination approach whereby each child inherits characteristics from its two parents such that each parameter is chosen at random somewhere between the values of its parents, with sufficient added noise (or mutation) to give a reasonable possibility of the parameter falling outside the range spanned by the parents. In this study, each network will have ten such evolved parameters specified in its genotype, and many learnable weights/biases, as discussed above and summarised in Table 1. All the other network parameters are fixed across the whole population and all generations. In principle, the number of hidden units can also be evolved, but this usually results in the maximum allowed number being used anyway, so it is fixed at a value that is sure to be large enough to avoid under-fitting of the expected training data, yet not so large as to place unnecessary demands on the computational resources.

The primary objective here is to investigate which evolutionary approaches work best for neural networks of the type specified above, to identify and solve any emergent problems, and then attempt to formulate some general guidelines. There already exist many different evolutionary computation schemes that draw on various aspects of biological evolution [18], but here we shall consider from first principals what are the natural approaches for evolving good neural network learners. The first issue to consider is the procedure for managing the turnover of individuals in the populations. There are two broad classes of approach: *generational* approaches in which all the children are produced at once to create the next generation, and *steady state* approaches where a only a few children are introduced into the population at each stage [18, 19]. The distinction is often blurred by the idea of *elitism* whereby some of the fittest individuals are kept from one generation to the next in the generational approach. In cases, such as here, where individuals are able to improve their fitness by learning during their lifetime, there is a natural concept of *age* for each individual, and the crucial distinction is between generational approaches in which the populations always consists of individuals with the same age, and steady state approaches in which the population is made up of individuals with an emergent distribution of different ages. This kind of distinction was not studied in the previous comparisons [19, 20, 21]. Elitism will be used in the generational approaches here, to help preserve good genetic information, but still the populations will have fundamentally different structure to our steady state approach.

Most biological systems follow some kind of *steady state* approach, and such a nature inspired approach has previously proved successful for evolving good neural network learners [13, 15, 16]. A natural formal specification is:

SS Every individual learns to improve their performance for as long as they live. High fitness corresponds to good generalization performance. After each simulated year, a small random subset of individuals over a certain simulated age die of old age, and a small random subset of the least fit individuals are killed through competition with other individuals. The dead individuals are replaced by children that have parents chosen from among the fittest individuals. The need to compete with older individuals indirectly encourages faster learning.

The concept of "simulated year" is crucial in this study to allow the learning and evolutionary time-scales to be measured in the same units, which in turn is necessary to set up fair comparisons between the various evolutionary approaches. There are two obvious *generational* approaches that can be used to evolve systems that learn [2, 22], one directly encouraging good generalization performance, and one directly encouraging fast learning:

G1 Every individual learns to improve their performance for a set number of simulated years. High fitness corresponds to good generalization performance. After the set number of years, all individuals die and are replaced by the next generation of children that have parents chosen from among the fittest individuals.

G2 Every individual learns to improve their performance until it reaches a set target level of generalization performance. High fitness corresponds to reaching the target level quickly. After all the individuals have achieved the target performance, or reached some time limit, they all die and are replaced by the next generation of children that have parents chosen from among the fittest individuals.

These broad specifications still leave much room for variation, such as in the precise details of the fitness measures, selection schemes, and the various evolutionary parameters.

In the *SS* approach, it is important to realise that if all the individuals were able to learn their task perfectly by the end of their first year, and their performance was only tested once per year, the advantage of those that learn in two months over those that take ten is lost, and the evolutionary process would not encourage faster learning. Consequently, "one simulated year" needs to be defined as the time taken to experience some sufficiently small fraction of the total number of training data samples required to learn the given task. Then the death rates need to be set to result in reasonable age distributions, with the best performing adults having time to reproduce, yet not dominating the population and killing off most of the children before they have had a chance to learn how to perform well. This is conveniently achieved by having fixed tournament competition death rates with 10% of the population losing random pair-wise fitness comparisons and dying each simulated year, and old age death rates dependant on how fast the best individuals learn, with 20% of the individuals aged more than three times the average earliest age of best performance dying each simulated year.

For the *G1* approach, one needs to specify the number of training samples, or simulated years N_G, per generation. The difficulty is that, if this is set too low, none of the individuals will ever get close to finishing learning, and if it is set too high, it is not clear what will drive the individuals to reach their peak performance any earlier. While this approach might be useful for evolving other factors [2], it is not obviously any good for the problem of evolving fast learners. A more reliable variation would be to fix N_G at some sufficiently large value, and measure the average performance over the last N_M years rather than just the final year. If one starts with $N_M = 1$, and increases N_M by one whenever a reasonable fraction (say half) of each generation consistently settle down to a clear maximum performance over that period, the drive to learn faster will never be lost. The $N_M = 1$ phase will concentrate on optimizing the performance, and then the $N_M > 1$ phase will speed the reaching of that performance. A potential alternative variation would be to start with N_G at some sufficiently large value, and then decrease that by one whenever a reasonable fraction (say half) of each generation consistently achieve a clear maximum performance by that age. The problem with this is that the random training data sampling includes easier than average periods that cause N_G to fall too quickly, and this results in too many individuals not completing their learning. The $N_M > 1$ approach also has the advantage of providing a more accurate measure of the average performance, so we shall concentrate on studying that two stage *G1* approach.

In the *G2* approach, the crucial parameter that needs specifying is the target level of performance. The difficulty here is that, if the target is not set to be the best level of performance possible, the individuals will never reach it, and in general one will not know in advance what that level is. Moreover, even if one did know it (e.g., because it was known that perfect performance was always achievable), it is quite possible that that performance level would not actually be reachable by the individuals in the first generation. It would appear that this approach on its own is not feasible at all for evolving fast learners. However, if one starts off with a large N_G and $N_M = 1$ in the *G1* approach as before, one can switch to the *G2* approach once a reasonable fraction (say half) of each generation is consistently achieving a clear maximum performance at the end of that period, with the target level of performance set equal to that maximum. Here, the *G1* phase with $N_M = 1$ again optimizes the performance, but now a *G2* phase is used to optimize the speed of reaching that performance.

A complication facing both of the two-stage generational approaches is that, for many applications, there may not be a 'clear maximum performance level', and so it may be difficult to determine when to switch from the first to second stage. For example, if the data samples involve a significant level of noise, as most real world applications do, the generalization performance will reflect that, and the random data sampling will lead to variations in that generalization performance. This will obviously also complicate the measuring of the learning time in the *G2* stage of the *G1+G2* approach. One will need to formulate procedures for providing suitable estimates of when the 'maximum performance level' has been reached, and use those. The details will inevitably be somewhat application dependent, depending on which particular aspects of performance are most crucial for the given application, but in principle, finding appropriate criteria should be possible. In this paper, however, this complication, and the associated problem dependence, is avoided by choosing a task that allows perfect (zero error) performance, and hence an unambiguous maximum performance level.

The above discussion has identified three distinct, natural, and apparently feasible, approaches one could follow for evolving fast neural network learners: the nature inspired steady-state *SS* approach, the two stage *G1+G1* generational approach, and the hybrid *G1+G2* generational approach. The remainder of this paper explores empirically how well each of these approaches actually work for a particular representative class of learning tasks, studies in more detail some unwanted emergent properties revealed by that comparison, and investigates some potential solutions to those problems.

4. BASELINE SIMULATION RESULTS

Whichever evolutionary approach one follows, obtaining reliable results requires careful setting of all the evolutionary parameters according to the details of the application and the speed and coarseness of the simulations. To keep the comparisons fair, as many details as possible were kept constant across all the evolutionary runs: A fixed population size of 200 was a trade-off between maintaining genetic diversity and completing the simulations reasonably quickly. The genotype specification and procreation process were the same throughout, as described above. The generational approaches used elitism, whereby 50% of each new generation had genotypes that were exact copies of the fittest parents'. The neural network architecture and training data were as described above, and a few simple tests determined that 20 hidden units were adequate to learn the training data, and that 1200 data samples per simulated year provided adequate performance estimates for both testing and training purposes. Similar studies [12, 13, 15] have found empirically that the evolved parameters range over many orders of magnitude, so it was computationally efficient to apply the crossovers and mutations to the logarithms of the parameters, rather than the parameters themselves. The additive mutations were based on a mixture of two sufficiently wide Gaussian distributions (98% of width 0.1 plus 2% of width 1.0) to speed the evolution as much as possible by maintaining genetic diversity without introducing an excessive amount of noise into the process.

Earlier evolutionary studies [13, 15, 16] found that the evolutionary efficiency depends rather strongly on the initial conditions, i.e. on the distribution of innate parameters across the initial population. In particular, the populations are found to settle into their final configurations more reliably and quickly if they start with wide distributions of initial learning rates, rather than expecting the mutations to carry them from a state in which there is little learning at all. So, for each evolutionary run, the random initial population learning rates η_L were taken uniformly from the range [0.0, 4.0], the momentum parameters α from [0.0, 1.0], the regularization parameters λ from [0.0, 0.1], and the random initial weight ranges r_L from [0.0, 4.0]. These ranges were chosen to cover the values typically used in hand-crafted networks, in order to provide a good chance of starting with a range of reasonably competent individuals.

To complete extensive and fair comparisons it is important to set generational timescales that don't waste resources (e.g., by having the number of training samples per generation far too high), so we begin by establishing a baseline performance level using the steady state *SS* evolutionary approach. In view of the number of random factors involved, all the simulations were run many times, with different random numbers, to determine the statistical variability. Figure 2 shows the evolution of the initial weight ranges, learning rates, momentum and regularization parameters, and the resulting generalization performance, averaged over the populations from 30 runs. The momentum and regularization parameters quickly take on values that are so low that they have very little effect on the networks' performance, as one might expect from the nature of the training data, so we shall not discuss them any further. The four learning rates, two of the initial weight ranges, and the error counts appear to settle down quickly, but actually the variances across simulations are so great that plotting the standard deviations would render the graphs unreadable. A detailed investigation reveals underlying bimodal distributions, with each run settling down into one of two final configurations. Nine runs resulted in a fast learning population, and 21 in a slow learning population. The bimodal nature is evident in the learning rate means and standard deviations for those two sets shown in Figure 3. It is as though two distinct ‘species’ evolve, with no populations nor individuals falling in between them. The initial weight ranges r_{IH} and r_{HB} also take on specific narrow and vastly different values for the two ‘species’, but r_{HO} and r_{OB} vary wildly, even within a single run, indicating that they have little influence on the individuals’ performance. The ranges r_L follow this pattern in all the simulations, and so will not be discussed further. Note that it is important to run the simulations for much longer than the initial settling phase (about ten times longer in this case), because there are often long periods of equilibrium before a rapid switch to an alternative configuration [23]. In fact, the brief increase in variance in η_{HO} at around 80000 years in the left hand graph of Figure 3 corresponds to an aborted attempt at such a switch in just one run.

It is worth noting how different the learning parameters are between ‘species’ and across the different network components, and how far they all are from the values typically used when building networks by hand. Of course, it is the performance levels of the evolved populations that are of primarily interest, since it is they that justify the use of such far from traditional learning parameters [12]. Figure 4 shows the minimum, average and maximum individual generalization error counts for the populations throughout evolution, with the variances across runs. There are several points of interest: For the first 4000 years there are rarely any individuals who achieve perfect (zero error) performance on new data samples, but after that, every population has at least one such individual. This has obvious implications for getting a *G2* style generational evolution started. The faster learning ‘species’ have significantly lower average error levels, as one would expect given that the individuals are lowering their error levels more quickly. However, the maximum error levels of the fast learners are higher and much more variable than the slow learners, which is potentially problematic and warrants further study.

To evaluate fully the performance results, an accurate and statistically reliable measure of the individuals’ learning abilities is needed, so each individual from each evolved population was tested on each of 50 different training data sets. This is because the variances result from differing task difficulties as well as individual differences. By the age of 50 simulated years (i.e. 60,000 data samples), even the slow learning populations have their peak in the generalization error count distribution at zero errors, but the distributions have very long tails which make the means and standard deviations seem misleadingly large. The performances of the two evolved steady state ‘species’ are represented more clearly by the median, upper and lower quartiles of errors per year at each age as shown in Figure 5.

5. COMPARISON OF THE EVOLUTIONARY APPROACHES

Having found significant differences in the learning rates and generalization performance levels between the two steady state ‘species’, and also relative to traditional hand-built networks, these results are now compared with those from the two generational approaches. The generational time-scale was set at $N_G = 60$, which the steady state approach established to be adequate for learning perfect performance, yet not too wasteful of computational resources. The evolution of the learning rates for the two generational approaches are shown in Figure 6, with means and variances over 20 runs. The populations take somewhat longer to settle down into their final states compared with the steady state approach, but no longer end up with multi-modal distributions. The final learning rates (and other parameters) are remarkably similar to those of the fast learning steady state populations, as are the average performance levels. Figure 7 shows the error rates per year for the evolved populations in each case, averaged over 50 training data distributions as for Figure 5. The median and lower quartile error rates are very similar across generational approaches, and also to the fast learning steady state populations, but there are clear differences in the ages at which the upper quartiles reach zero in the three cases.

Although differences have been found between the *G1+G1* and *G1+G2* two stage generational results, it is not yet clear how crucial the second stage of evolution really is. A further set of 20 runs were therefore carried out using only the initial *G1* stage, with the fitness determined only by the performance at age $N_G = 60$. The learning rate evolution and evolved error rates in this case are plotted in Figure 8. Typically, the transition to stage 2 would take place at between 32,000 and 44,000 simulated years, but the parameters continue to evolve beyond that, even without the second stage fitness criteria, though at a noticeably slower pace. By 240,000 simulated years the median and quartile error plots show very similar performance to the two stage generational approaches (of Figure 7), with only slightly worse upper quartile performance, and slightly better lower quartile performance. We conclude that, given a long enough period of evolution, the pressure in the *G1* approach to consistently do well at age $N_G = 60$ is alone sufficient to evolve fast learners.

It seems that all of the obvious evolutionary approaches are capable of evolving neural networks that perform well, but there *are* significant differences. In Figure 9, the median error rates appear similar, but the distribution of ages at which individuals reach their first full year of perfect performance reveal worthwhile improvements across approaches. Moreover, clear differences are also apparent in the means and standard deviations of the error count distributions during learning shown in Figure 10, again for 200 individuals and 50 different sets of training data. The standard errors on the means are 100 times lower than the standard deviations, so the differences between the means are significant, despite the learning rates looking rather similar.

Another important aspect of performance difference is in the distribution of errors per simulated year after training. Figure 11 shows two views of the average generalization error distributions for individuals between the ages of 50 and 60. On the left, the peaks of all the distributions are around zero, as one would hope; and, as one would expect from the quartiles plotted above, the slow learning steady state populations have a significantly wider peak than the other four cases. The differences are more noticeable in the tails of the distributions, shown on the right. The fast learning steady state populations have around forty times as many large error counts (above 30) as the slow learning steady state populations, which is one evolutionary advantage that the slow learners have, and also explains the unexpected pattern of maximum errors seen in Figure 4. The *G1+G2* generational populations' error distribution levels off between that of the two steady state groups, while the *G1+G1* generational populations' distribution continues to fall with increasing numbers of errors. The single stage *G1* populations are similar to the two stage *G1+G1* populations at the peak, and the *SS slow* populations at the tail.

The differences between the evolved populations from the various approaches are easily understood by considering the different evolutionary pressures involved. The *G1* approach only has the performance at age 60 driving the evolution, and consequently it ends up with the best performance at that age. Since it has nothing to directly encourage fast learning, one might expect it to be slower to learn at earlier ages, but in fact it only ends up slightly slower than the other approaches. The *G1+G1* approach measures the fitness as the average performance over an increasing number of years prior to age 60, which will naturally improve the rate at which most individuals learn, but this also indirectly encourages a more risky learning strategy with some individuals occasionally failing to learn properly at all, leaving large errors at all ages. The *G1+G2* approach encourages fast learning even more directly, by measuring the fitness as the number of epochs required to reach zero errors, and this does lead to even faster learning, but with even more high error instances as a result. Finally, the *SS* approach encourages faster learning at all ages, resulting in better learning again, but even more of the problematic high error cases. This is the same hierarchy as seen in the times taken for the four evolutionary processes to settle down.

6. FURTHER EVOLUTIONARY SIMULATIONS

While it is easy to understand how the different evolutionary pressures result in slightly different evolved populations for the three generational approaches, it is not so clear why two distinct types of population should emerge from the steady state approach. Are the slow learning *SS* populations just a poorly performing local maximum of fitness configuration, or do they have some real advantage, as suggested by their ability to learn to avoid very large errors? To test this, 100 fast learning and 100 slow learning individuals were taken from the evolved *SS* populations, and the mixed population was allowed to evolve further. For five different mixed populations of this type, for all of the evolutionary approaches, the fast learning individuals quickly came to dominate the whole population, sometimes in as few as three generations. This confirmed that the evolved slow learning individuals really were inferior to the faster learning individuals in all the approaches.

The next question was whether the slow learning *SS* populations would evolve into fast learning individuals if the evolution was continued using any of the generational approaches. Five slow learning populations were evolved for a further 240,000 simulated years (i.e. the length of the original evolutionary runs) using each of the generational approaches, and in all cases the populations stayed in their slow learning configuration. This indicated that these populations did indeed constitute a true local maximum of fitness, rather than a quirk of the *SS* approach. Normally, such premature convergence of evolutionary populations to local optima indicates that the parameter space is not being explored effectively, either because of a loss of genetic diversity, or because sufficient diversity was not there in the first place [24]. The obvious solutions are to start with a wider distribution of parameters in the initial populations, and/or widen the distribution of mutations to keep the populations diverse during evolution.

Since the evolved parameters span many orders of magnitude, the uniform distributions of initial population parameters η_L and r_L do not easily scale up to covering the whole of their final parameter spaces. It makes more sense here for the logarithms $log\,\eta_L$ and $log\,r_L$ to set be randomly from uniform ranges, in this case from [–4.0 +4.0] and [–2.0, +2.0] respectively. Doing this resulted in ten out of ten steady state runs evolving fast learning populations, with final parameters like those found in only 30% of runs previously, as shown on the left of Figure 12. This graph also shows that the populations take on their final configurations much more quickly than before, which is another advantage of this variation, but also a potential source of serious difficulties. The problem is that once outside the traditional range of small initial weights and learning rates, a large part of the parameter space corresponds to extremely poorly performing networks [25], and so using relatively small population sizes can easily lead to the characteristics of just one individual taking over the whole population, leading to the reduced diversity that we set out to avoid. The simulations of Figure 12 have clearly managed to avoid this difficulty, but this is not something that can be relied on happening more generally, particularly since one usually has very little idea of appropriate initial population parameter distributions for new application domains.

The alternative variation is to keep the smaller uniform initial parameter ranges, and instead modify the mutation distributions. If the mutations are increased by too small an amount, the evolution still results in one set of fast learners and one set of slow learners. If they are increased by too much, the evolving parameters vary widely and the populations never settle down. An intermediate widening of the mutation distribution from 98% of width 0.1 plus 2% of width 1.0 used originally, to 95% of width 0.2 plus 5% of width 2.0, led to different evolved populations, but not the improvement hoped for. Ten such runs resulted in evolved populations that fell into one of two classes, with the evolving learning rates and evolved error rates shown in Figure 13. The top graphs correspond to populations of fast learners (averaged over 6 runs) similar to those observed earlier (Figures 3 and 5), but with the wider high-error tail to be expected of a population experiencing many large mutations. The bottom graphs represent populations with ever increasing learning rates (averaged over 4 runs) driven by individuals which can sometimes learn extremely quickly, but resulting in poor average performance by the population as a whole. It can be concluded that increasing the mutation rates in this way is not the best way to proceed.

The findings so far can be summarized as follows: The *SS fast* and *G1+G2* evolved populations contain the fastest learners, but these also adopt the riskiest learning strategies that result in the highest numbers of individuals still having very poor performance at old ages. The *G1+G1* populations don't have such fast learners, but have fewer poorly performing individuals. The single stage *G1* populations are slower still to learn, as one might expect, but these individuals have much more consistently good performance by age 60. The *SS slow* populations contain the slowest learners, but have very few individuals that are still very poor performers at age 60. A closer investigation of the pattern of errors reveals that the problematic instances of very poor performance observed in Figure 11 are not due to a small number of consistently very poorly performing individuals, nor due to brief periods of very poor performance during lifetimes of otherwise reasonable performance. They actually arise from different individuals not being able to cope at all on a small number of tasks, and it seems difficult to predict which individuals will have problems with which tasks. The remainder of this paper will look at ways of avoiding this unfortunate situation. The *SS slow* populations will not be considered any further, since they do not exhibit the problems we are trying to solve, and because their relatively poor performance renders them unlikely to be used in practice anyway.

7. AGE DEPENDENT PLASTICITY

It seems likely, given the relative lack of problems with the *G1* and slower *SS* populations, that it is the pressure to learn quickly that is at the root of the problems with the faster learning populations. For many human abilities, it is well known that there are critical periods for learning, with high learning rates initially, and much lower rates at older ages (e.g., [26, 27]). Learning rate adaptation has also been shown to be beneficial for artificial neural network systems (e.g., [28]). The idea here is that by taking inspiration from these and allowing learning rates that can vary with age, it may be possible to evolve higher learning rates and faster learning at early ages, whilst maintaining lower learning rates later, resulting in less of an incentive to adopt the risky learning strategies observed above.

Such an age dependence can be implemented using a simple age t dependent scale factor to multiply all the learning rates. Previously, a general 40 parameter piecewise linear scale factor was found to evolve to take a form that is not far from an exponential fall to a baseline value [16]. To simplify matters here, we therefore consider the two-parameter scale factor

$$s(t) \; = \; \beta \, + \, (1-\beta)\,\mathrm{e}^{-t/\tau} \qquad , \qquad \eta_L(t) \; = \; s(t)\,\eta_L(0) \tag{4}$$

in which the baseline β and the time constant τ can evolve to take on any positive values. It is then relatively straightforward to repeat all of the above simulations with everything else the same, apart from the insertion of this scale factor and its two evolvable parameters.

The evolved scale factors, with variances across 10 populations of 200 individuals, are shown on the left of Figure 14. Note how the relatively problem free *G1* populations have learning rates that fall off with age much more slowly than for the other populations. On the right of Figure 14 is shown, for the *SS fast* populations, how the age dependent learning rates compare with the fixed learning rates that evolved before. The initial learning rates have adjusted themselves by different amounts to take advantage of the age dependent scaling. Not surprisingly, given the rather different emergent scale factors, the evolved adjustments vary with approach, so there are no longer similar learning rates across the four evolutionary approaches, which in turn leads to somewhat different learning performances.

The mean generalization error rates and standard deviations during learning are shown in Figure 15, and exhibit massive reductions compared with the corresponding results of Figure 10. So allowing age dependent plasticity produces clear overall improvements in the average error rates, but does it achieve the objective of reducing the occasional problematic high error cases? Figure 16 shows the error count distributions for the trained evolved networks for comparison with Figure 11. The peaks now have much wider variation across the evolutionary approaches than before, with the widths of the peaks in proportion to how much the learning speed is taken into account in the evolutionary fitness evaluation. In the tails, all four approaches show the hoped for reduction in the numbers of very large errors. Naturally, deciding whether that level of reduction is sufficient will be rather application dependent.

The *G1* populations, whose evolution is only driven by final performance, not surprisingly end up with a much improved final performance at age 60, but the initial rate of learning, which was previously comparable to the evolved fast learners, is now significantly poorer. The *G1+G1* populations, which are evolved to have good performance for as many years prior to age 60 as possible, now have much lower and consistent error rates across those final years, but the earlier reductions in errors are slower. The *SS* and *G1+G2* populations, which are evolved to reduce their error rates as quickly as possible, do now learn much more quickly, but still suffer from the most cases of persistent small errors that are not eliminated by age 50. We see the consequences of the inherent trade-offs whereby evolving improvement in one aspect leads to degradation of others, but we do now have a much clearer picture of how the different factors interact within the various evolutionary approaches, and what the trade-offs are.

8. ENSEMBLE TECHNIQUES

Since the instances of very poor performance in the fixed plasticity networks are fairly rare, it is possible that a simple committee or ensemble approach could be sufficient to remove them. The idea is that if an ensemble of three or more individual neural networks arrive at a combined output by a simple voting procedure, the occasional instances of very poor performance will be out-voted and cease to be a problem (e.g., as discussed in [29, 30]). Such an approach has certainly proved successful with other types of evolved neural networks in the past (e.g., [31]). However, if the individual networks all make the same rare mistakes, this approach will not be helpful.

Three is clearly the minimum number that can form a workable voting committee, so that case is investigated first. There are two obvious ways to proceed: one can take three independent individual networks from each evolved population, or take a single individual and train it three times starting from different sets of random initial weights drawn from its innate distribution. Even if parallel processing is not possible, and the total learning time is increased by a factor of three, this may still result in better overall performance. Figure 17 shows the mean error counts during learning, and the error distributions between ages 50 and 60, for voting over three runs of each individual. Compared with the corresponding results for single runs of each individual, there are reductions in errors by factors of three or four, which is encouraging, but there remain significant numbers of very large errors. Figure 18 shows the corresponding results for voting ensembles made up of three independent individuals. This approach is clearly much more successful at avoiding the large error cases, as expected given the reduced likelihood for error correlations across the ensemble [32], so we shall use this approach from here on.

The natural extension of voting in threes is to consider larger voting ensembles. On the left of Figure 19 is shown how the mean error rates at age 60 are reduced with increasing ensemble sizes. If the ensemble members are carrying out their computation in parallel, or if the limiting factor in the training times is the rate at which the training data arrives, rather than the number of compute cycles required, then this is the true picture of the ensemble performance. However, taking into account the extra time required by multiple serial training runs, by computing the error rates at ages 60/*EnsembleSize,* gives the rather different pattern of results seen on the right of Figure 19. There is now relatively little improvement overall for ensemble sizes greater than three, and the *G1* and *G1+G1* approaches actually get worse.

A useful measure of occurrences of the problematic very large error cases is the total proportion of errors above 30 arising between the ages of 50 and 60. This is plotted for the various ensemble sizes on the left of Figure 20, and shows clearly the expected benefit of the ensemble approach. In this case, taking the computational effort into account, by dividing each of the specified ages by the *EnsembleSize*, makes very little difference. This is because the large errors persist throughout a small number of training runs, but are absent from a very early age in the majority of runs. It would seem therefore, that ensembles of five or seven do have significant benefits in this respect, even if additional computational costs need to be taken into account.

The important question remaining is whether the ensemble approach can further improve the variable plasticity results too, or has the limit of what is achievable already been reached? Figure 21 shows the mean error rates during learning and the resultant error distributions for ensembles of three variable plasticity networks, for comparison with the fixed plasticity results in Figure 18. The slower initial reduction in errors for the variable plasticity *G1* and *G1+G1* cases will clearly be problematic if one needs to take the computational cost into account, but otherwise, the ensembles do give further improvement. Figure 22 shows in the mean error rates at age 60 for various ensemble sizes, for comparison with Figure 19. The proportion of very large error cases shown in Figure 23 confirms that variable plasticity and ensembles together can completely eliminate the problem of occasional very large errors.

9. DISCUSSION AND CONCLUSIONS

The increasing use of evolutionary techniques for improving neural network systems [2, 3] makes it important to have a good understanding of which evolutionary approaches are likely to be best for each particular type of application, what potential problems might emerge, and how one can best alleviate them. The aim of this paper has been to explore these issues for the range of evolutionary approaches that are most natural for producing neural networks that learn quickly to generalize well. The simulation results presented are consistent with the expectation that the different approaches *do* optimize different aspects of performance, and consequently yield different evolved behaviours. The important findings concern the reliability of the various approaches, and the discovery of which factors are most crucial for the various aspects of good performance. For a given application, it will generally be known which aspects of learning and generalization performance are most important, and the simulation results presented here can provide guidelines as to which approach is most likely to achieve the desired results.

Several fundamental issues, that need to be considered when choosing a particular evolutionary approach for a given neural network application, can be identified:

1. Is the aim to simulate natural biological evolution? If so, the steady state *SS* approach is appropriate, but particular care is needed to prevent the populations becoming stuck in local maxima of fitness.

2. Is it important to obtain individuals that tend to learn good performance as quickly as possible? If so,

the steady state *SS* and generational *G1+G2* approaches are best, with the generational evolution likely to be more reliable.

3. Is it acceptable to wait longer for better performance levels to be learned? If so, the generational *G1* and *G1+G1* approaches are best, but single stage *G1* evolution is likely to take longer.

4. Is there a lack of reliable criteria for two stage evolution, for example, because of particularly noisy training data? If so, the *SS* or *G1* approaches should be used, and the *SS* evolution is likely to result in faster learners. However, this depends on how well one can generate an appropriate initial generation, and maintain diversity during evolution.

5. How problematic are risky learning strategies and occasional instances of very poor performance? If they must be avoided, variable plasticity and/or ensemble techniques are usually worth using. Otherwise, the standard *G1* approach is safest, but it will lead to slower learners.

Inevitably there are trade-offs between the various criteria and approaches, and one will have to refer to the detailed results presented here for an indication of which approach is likely to be best overall for each particular application. Often one may be able to tolerate occasional very poor performance in return for faster learning most of the time, or faster learning on average. Certainly, in natural systems it can be a good strategy overall for individuals to respond as quickly as possible, even if it leads to occasional deaths. This will be particularly true if it allows more children to be produced before an untimely death than would otherwise be possible in a full lifetime. On the other hand, if one is using evolution to design a single autonomous system, such a strategy would be unacceptably foolhardy. The more one encourages faster learning, the riskier the learning strategies that emerge, and the more the need to introduce additional techniques to compensate for them. Allowing age dependent plasticity and/or using ensemble techniques with any of the evolutionary approaches can reduce, or even totally eliminate, the evolution of risky learning strategies, but these interact in complex ways with the other aspects of performance, and again one needs to refer to the detailed results presented here.

A major consideration for all evolutionary techniques is the computational costs involved. In this study, the tasks were chosen to be simple enough to render feasible a full systematic investigation, but those classification tasks are typical of more realistic applications, and it is reasonable to expect analogous results for them. Other studies indicate that the evolutionary efficiency does scale up well with task and neural network size, but it is difficult to avoid the increased training times that will be required for larger networks and more complex training data [17]. The relatively slow evolution rates in the two stage generational approaches, compared with the steady state approach, are probably not something to be too concerned about. It is the different emergent behaviours arising from the different uses of the fitness measures that are of primary interest, not how fast the evolution achieves those behaviours. Moreover, there remain many other variations that could be employed to speed up the evolution: different selection schemes, different elitism proportions, alternative mutation distributions, and such like. The slowness of the single stage generational approach might be more problematic, particularly if reasonably fast learners are required, simply because of the reduced fitness pressure involved. However, that increased cost may still be small in relation to the increased tendency of the steady state approach to settle into far from optimal fitness configurations, and the consequent need to carry out multiple runs from different initial populations. Ensuring greater diversity in the initial populations was found to be a more reliable way of avoiding such poor performing populations than attempting to maintain population diversity with larger mutation rates. Knowing the best evolutionary approach for each application will clearly be of no use without an appropriately diverse initial population to start from.

Earlier studies have already shown that evolution can be used to create high performance neural network systems [2–13, 15–17], and this paper has confirmed that even a less than optimal choice of evolutionary approach can still result in the emergence of good individual networks. However, it is now clear from this study that to generate the best possible neural networks, it is important to choose the right evolutionary approach for each particular application. It has also been shown how additional techniques, in particular, age dependent plasticity and ensemble voting, can alleviate some of the unwanted side effects of the most efficient of the obvious evolutionary approaches. The detailed results presented in this paper demonstrate clearly the trade-offs between the various approaches, techniques and performance measures, and provide some useful indications of how best to proceed once one has decided which aspects of evolved performance are most crucial for the particular application in question.

ACKNOWLEDGEMENTS

This paper brings together into a unified framework, and expands upon, a number of ideas and results previously scattered across several conference papers [12, 15, 17, 22, 33, 34].

REFERENCES

[1] C.M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, UK: Oxford University Press, 1995.

[2] X. Yao, Evolving Artificial Neural Networks. *Proceedings of the IEEE,* vol. 87, pp. 1423-1447, 1999.

[3] E. Cantû-Paz & C. Kamath, An Empirical Comparison of Combinations of Evolutionary Algorithms and Neural Networks for Classification Problems. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics,* vol. 35, pp. 915-927, 2005.

[4] C.A. Perez, C.A. Salinas, P.A. Estevez & P.M. Valenzuela, Genetic Design of Biologically Inspired Receptive Fields for Neural Pattern Recognition. *IEEE Transactions on Systems, Man, and Cybernetics, B,* vol. 33, pp. 258-270, 2003.

[5] N. Funabiki, J. Kitamichi & S. Nishikawa, An Evolutionary Neural Network Approach for Module Orientation Problems. *IEEE Transactions on Systems, Man, and Cybernetics, B,* vol. 28, pp. 849-855, 1998.

[6] S. Baluja, Evolution of an Artificial Neural Network Based Autonomous Land Vehicle Controller. *IEEE Transactions on Systems, Man, and Cybernetics, B,* vol. 26, pp. 450-463, 1996.

[7] M. Richards, D. Moriarty & R. Miikkulainen, Evolving Neural Networks to Play Go, *Applied Intelligence*, vol. 8, pp. 85-96, 1997.

[8] A. Agogino, K. Stanley & R. Mikkulainen, Online Interactive Neuro-Evolution. *Neural Processing Letters*, vol. 11, pp. 29-37, 2000.

[9] P.A. Castillo-Valdivieso, J.J. Merelo, A. Prieto, I. Rojas & G. Romero, Statistical Analysis of the Parameters of a Neuro-Genetic Algorithm. *IEEE Transactions on Neural Networks*, vol. 13, pp. 1374-1394, 2002.

[10] K. Stanley & R. Miikkulainen, Evolving Neural Networks Through Augmenting Topologies, *Evolutionary Computation*, vol. 10, pp. 99-127, 2002.

[11] H.A. Abbass, Speeding Up Backpropagation Using Multiobjective Evolutionary Algorithms. *Neural Computation*, vol. 15, pp. 2705-2726, 2003.

[12] J.A. Bullinaria, Evolving Neural Networks: Is it Really Worth the Effort? In *Proceedings of the European Symposium on Artificial Neural Networks*, Evere, Belgium: d-side, 2005, pp. 267-272.

[13] J.A. Bullinaria, Evolving Efficient Learning Algorithms for Binary Mappings. *Neural Networks*, vol. 16, pp. 793-800, 2003.

[14] L. Bottou & Y. Le Cun, Large Scale Online Learning, In *Advances in Neural Information Processing Systems 16*, Cambridge, MA: MIT Press, 2004.

[15] J.A. Bullinaria, Simulating the Evolution of Modular Neural Systems. In *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society,* Mahwah, NJ: Lawrence Erlbaum Associates, 2001, pp. 146-151.

[16] J.A. Bullinaria, From Biological Models to the Evolution of Robot Control Systems. *Philosophical Transactions of the Royal Society of London* A, vol. 361, pp. 2145-2164, 2003.

[17] T. Seipone & J.A. Bullinaria, Evolving Improved Incremental Learning Schemes for Neural Network Systems. In: *Proceedings of the 2005 IEEE Congress on Evolutionary Computing (CEC 2005)*, Piscataway, NJ: IEEE, 2005, pp. 273-280.

[18] A.E. Eiben & J.E. Smith, *Introduction to Evolutionary Computing*, Berlin, Germany: Springer-Verlag, 2003.

[19] D. Whitley, The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA: Morgan Kaufmann, 1989, pp. 116-123.

[20] G. Syswerda, A Study of Reproduction in Generational and Steady-State Genetic Algorithms. In G Rawlins (Ed.), *Foundations of Genetic Algorithms*, San Mateo, CA: Morgan Kaufmann, 1991, pp. 94-101.

[21] K.A. De Jong & J. Sarma, Generation Gaps Revisited. In L.D. Whitley (Ed.), *Foundations of Genetic Algorithms 2*, San Mateo, CA: Morgan Kaufmann, 1993, pp. 19-28.

[22] J.A. Bullinaria, Generational versus Steady-State Evolution for Optimizing Neural Network Learning. In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2004)*, pp. 2297-2302. Piscataway, NJ: IEEE, 2004.

[23] S.J. Gould & N. Eldredge, Punctuated Equilibria: The Tempo and Mode of Evolution Reconsidered. *Paleobiology*, vol. 3, pp. 115-151, 1977.

[24] R.K. Ursem, Diversity-Guided Evolutionary Algorithms. In J.J Merelo Guervós et al. (Eds), *Parallel Problem Solving from Nature*, vol. 2439 of *LNCS*, Heidelberg, Germany: Springer, 2002,

pp. 462-471.

[25] Y. Lee, S.H. Oh & M.W. Kim, An Analysis of Premature Saturation in Back Propagation Learning. *Neural Networks*, vol. 6, pp. 719-728, 1992.

[26] B. Julesz & I. Kovacs (Eds), *Maturational Windows and Adult Cortical Plasticity*. Reading, MA: Addison-Wesley, 1995.

[27] D.B Bailey, J.T. Bruer, F. Symons & J.W Lichtman. (Eds), *Critical Thinking About Critical Periods*. Baltimore: Brookes, 2000.

[28] R.A. Jacobs, Increased Rates Of Convergence Through Learning Rate Adaptation. *Neural Networks*, vol. 1, pp. 295-307, 1988.

[29] L.K. Hansen, & P. Salamon, Neural Network Ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 993-1000, 1990.

[30] R. Battiti, & A.M. Colla, Democracy in Neural Networks: Voting Schemes for Classification. *Neural Networks*, vol. 7, pp. 691-709, 1994.

[31] X. Yao, & Y. Liu, Making Use of Population Information in Evolutionary Artificial Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics, B*, vol. 28, pp. 417-425, 1998.

[32] K. Turner, & J. Ghosh, Error Correlation and Error Reduction in Ensemble Classifiers. *Connection Science*, vol. 8, pp. 385-403, 1996.

[33] J.A. Bullinaria, Evolved Age Dependent Plasticity Improves Neural Network Performance. In *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems (HIS 2005)*, Piscataway, NJ: IEEE, 2005, pp. 79-84.

[34] J.A. Bullinaria, Ensemble Techniques for Avoiding Poor Performance in Evolved Neural Networks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006),* Piscataway, NJ: IEEE, 2006, pp. 9857-9864.

Table 1
The evolved parameters that are specified in the genotype and fixed for each individual neural network, and the learned parameters that vary during each individual's lifetime

Evolved parameters (unrestricted real valued)		Learned parameters (unrestricted real valued)	
4 initial weight/bias distributions	$[-r_L, +r_L]$,	many connection weights/biases	w_{ij}
4 learning rates	η_L,		
1 learning momentum	α		
1 regularization parameter	λ		

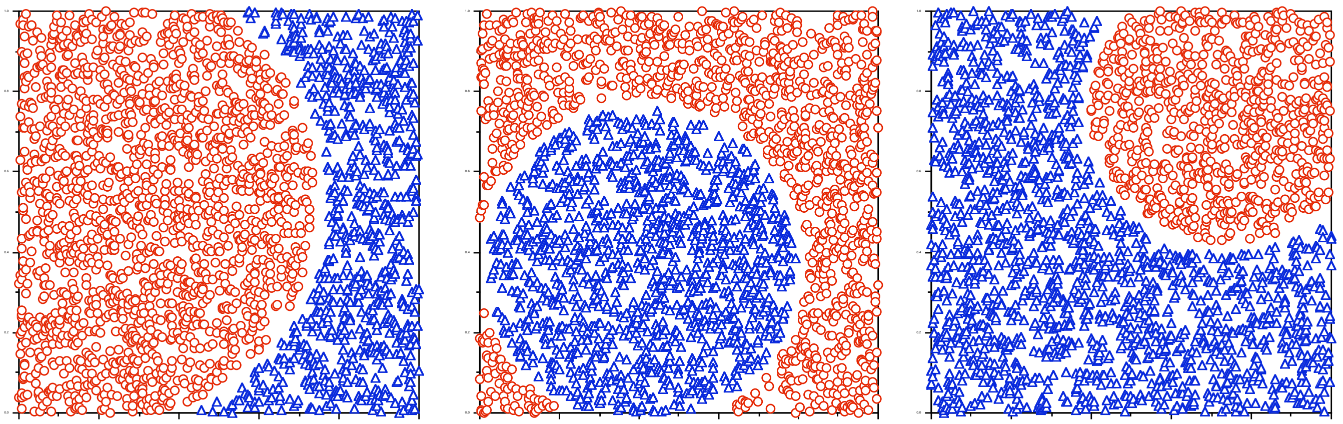


Figure 1: Some typical two-class classification data distributions which the neural networks are expected to learn from randomly chosen samples.

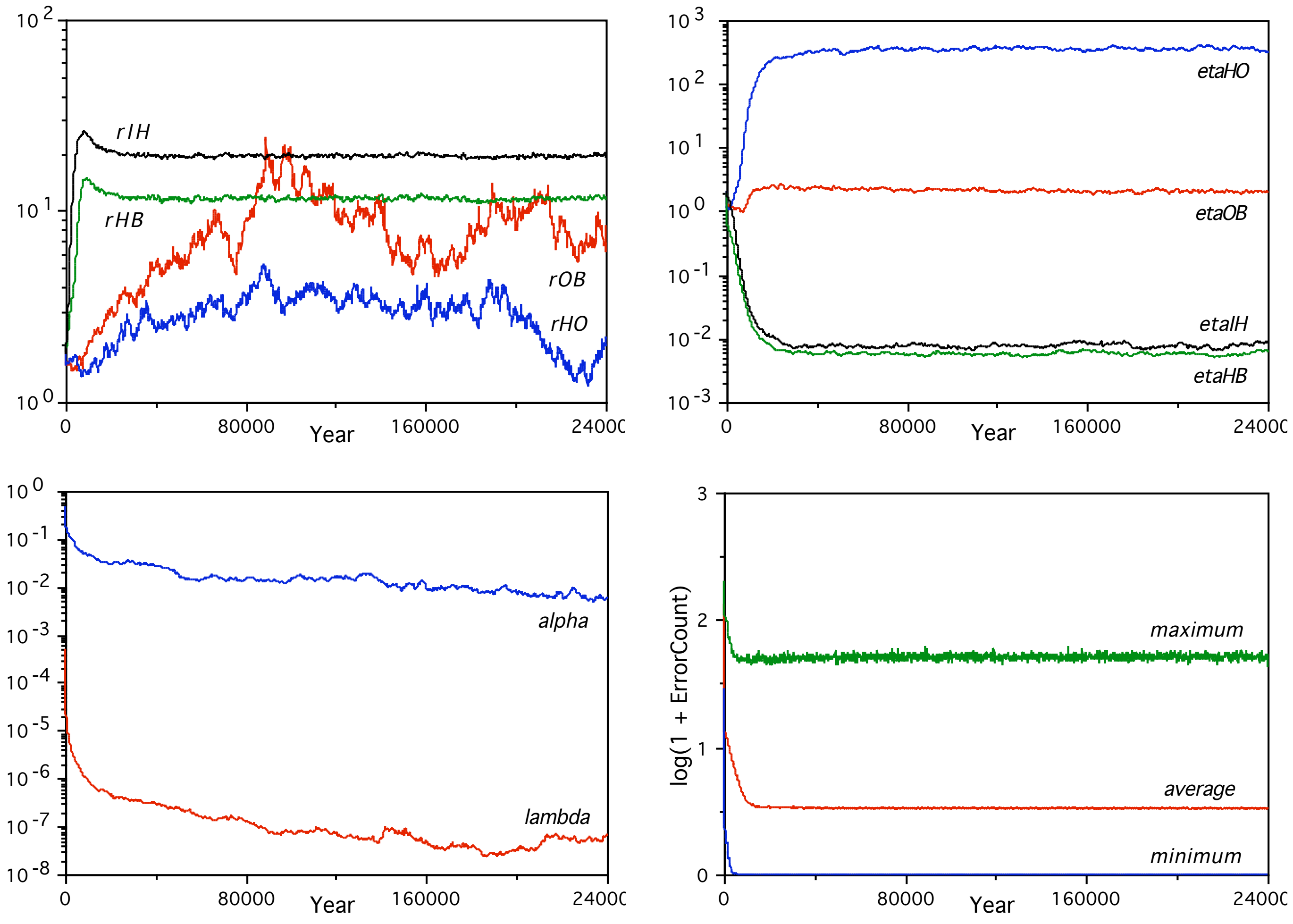


Figure 2: Evolution of the average initial weight ranges r_L, learning rates η_L, momentum α and regularization parameter λ, and associated error rates, for the steady state *SS* approach (averaged over the populations from 30 runs).

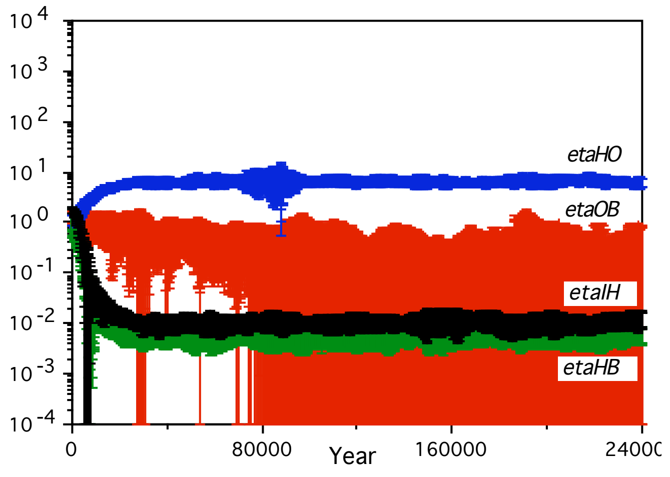


Figure 3: Evolution of the average learning rates η_L, and their variances across different runs, for the slow learning (left) and fast learning (right) steady state *SS* populations.

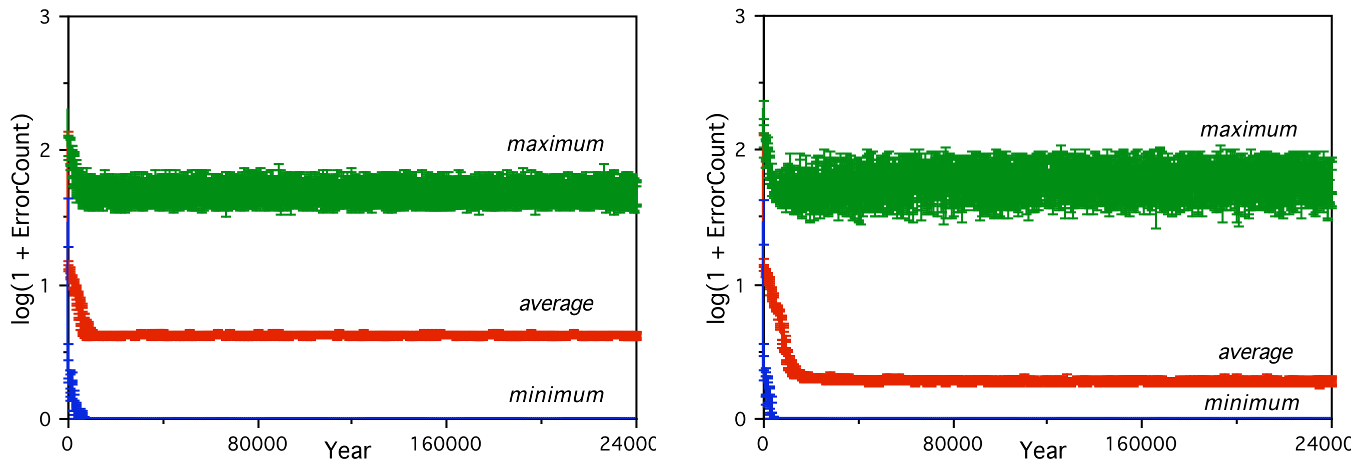
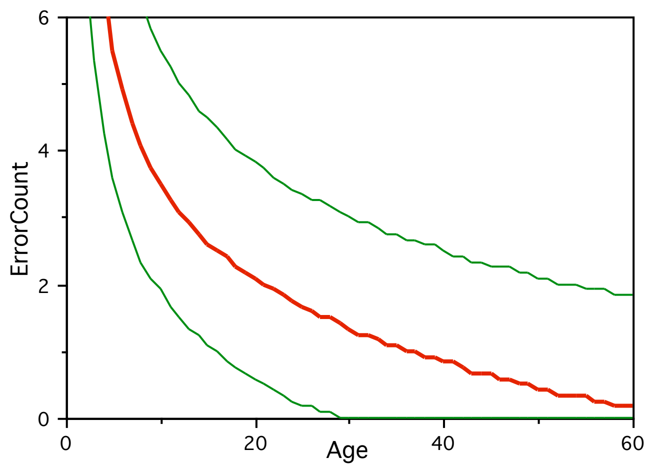


Figure 4: Evolution of the minimum, average and maximum individual error levels, and their variances over different runs, for the slow learning (left) and fast learning (right) steady state *SS* populations.



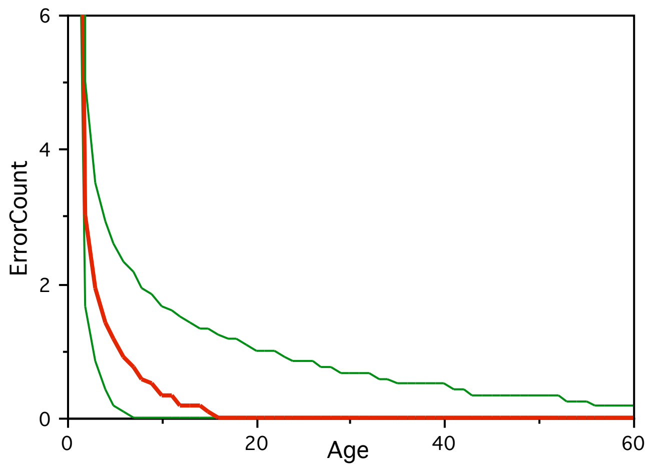


Figure 5: Median and quartile error rates for the individuals from the slow learning (left) and fast learning (right) steady state *SS* populations at year 240,000.

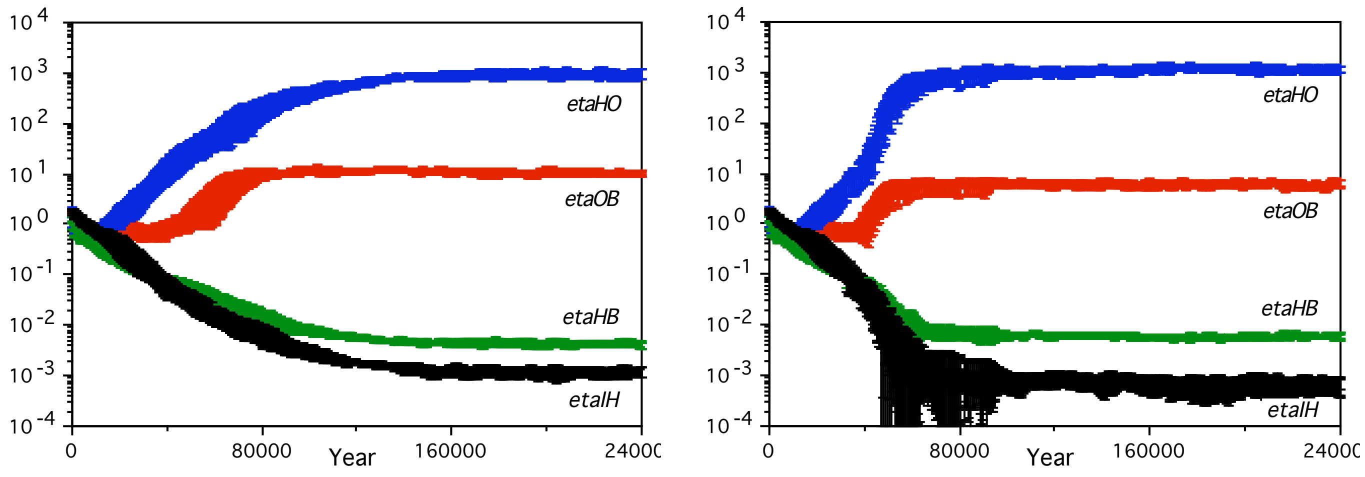


Figure 6: Evolution of the average learning rates η_L, and their variances across 20 runs, for the *G1+G1* (left) and *G1+G2* (right) generational populations.

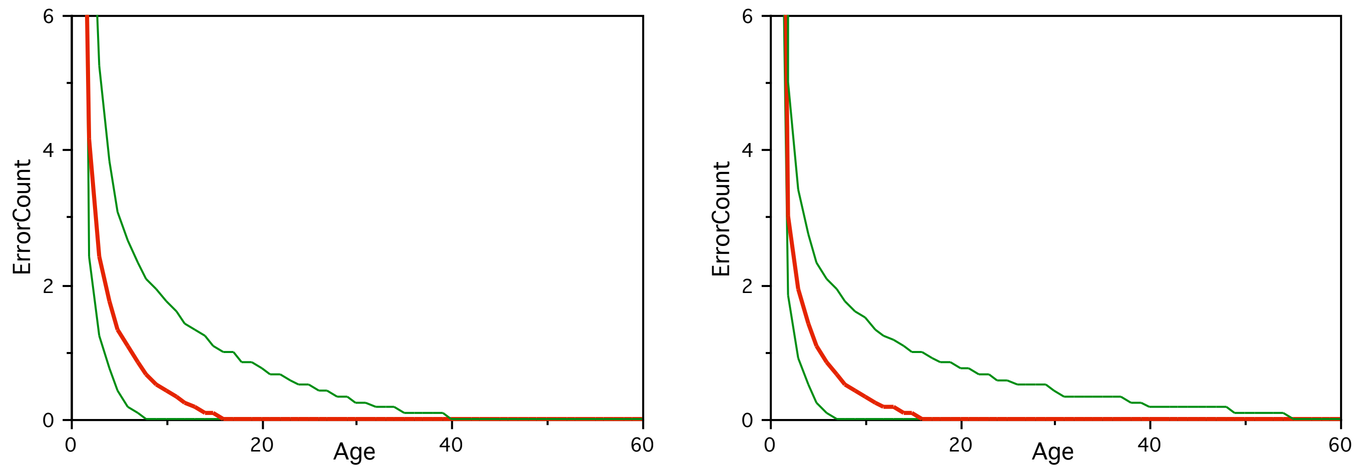
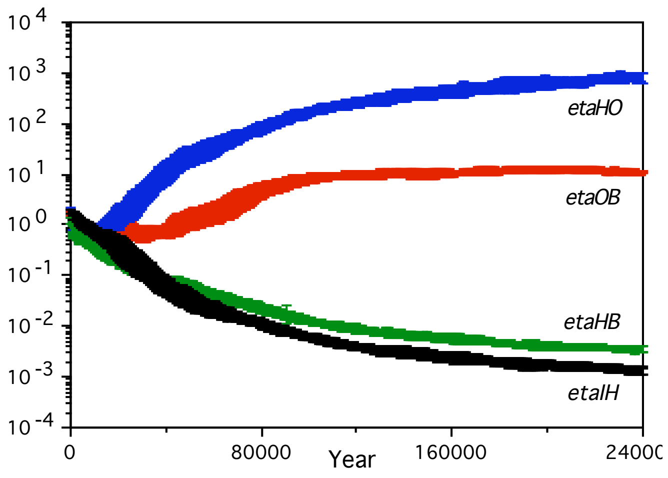


Figure 7: Median and quartile error rates for the individuals from the *G1+G1* (left) and *G1+G2* (right) generational populations at year 240,000.



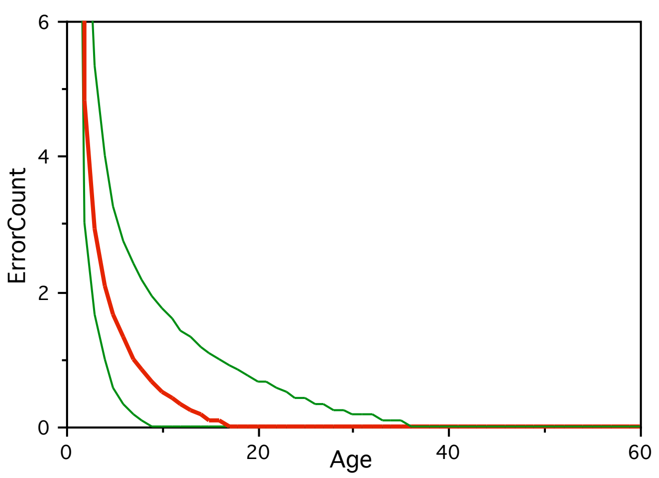
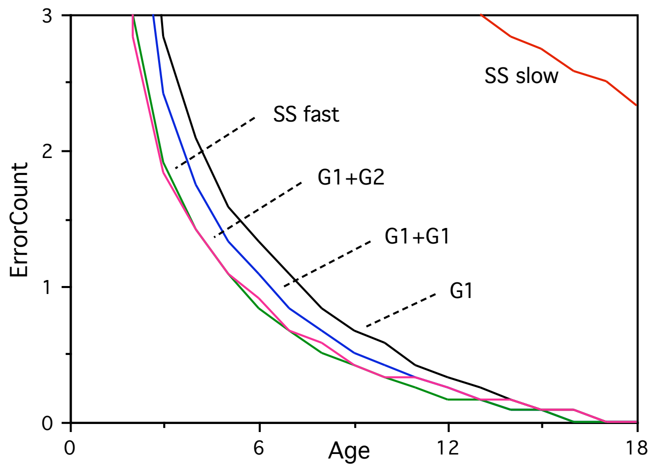


Figure 8: Evolution of the learning rates η_L (left) and the resulting median and quartile error rates at year 240,000 (right), in the generational *G1* approach when there is no second stage of evolution to encourage faster learning.



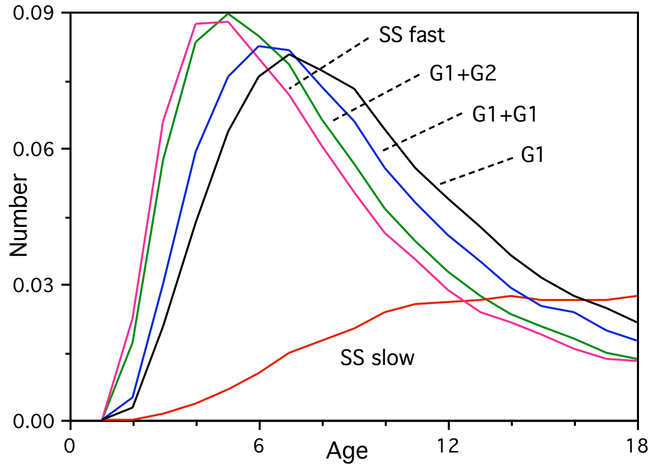
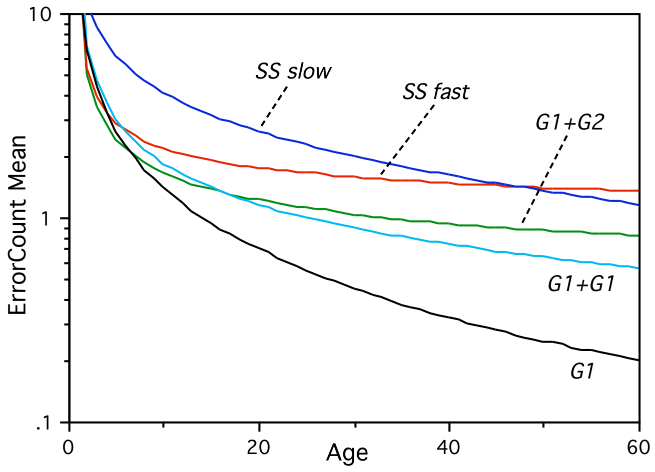


Figure 9: The median error rates during learning (left), and the distribution of ages at which individuals reach their first complete year of perfect performance (right), for the five sets of evolved populations.



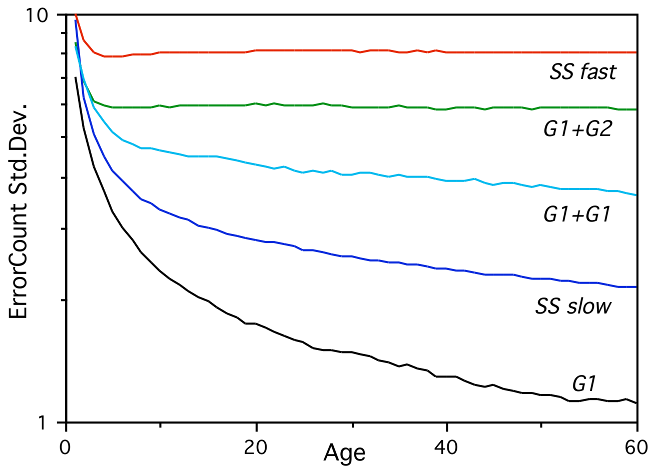


Figure 10: The means (left) and standard deviations (right) of the error count distributions for individuals during learning, for the five sets of evolved populations.

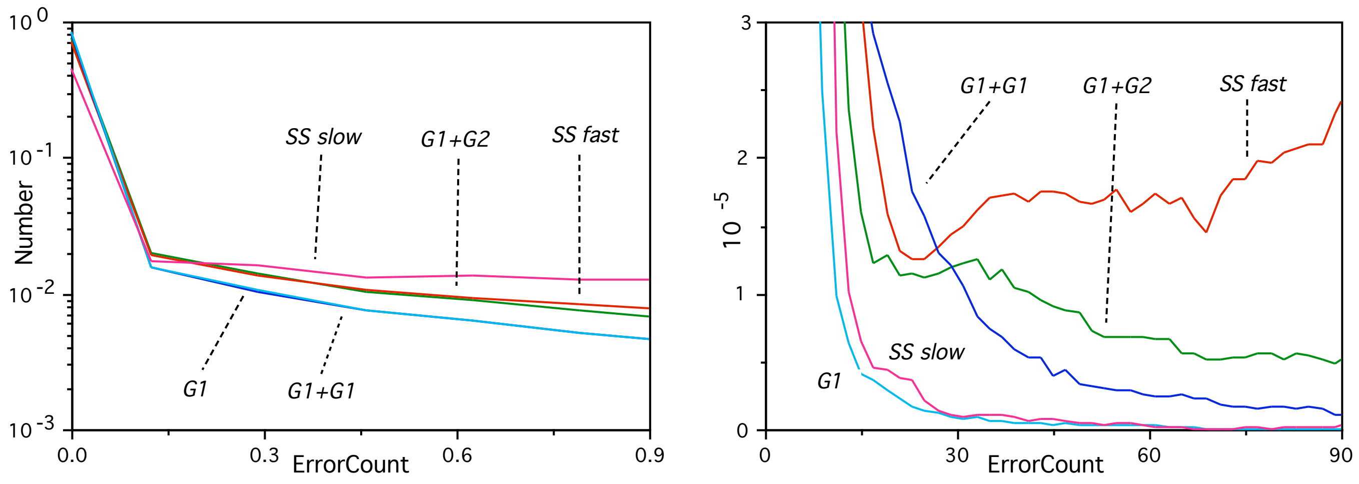


Figure 11: Two views of the average error count distributions for individuals between the ages of 50 and 60 for the five sets of evolved populations: peaks (left) and tails (right).

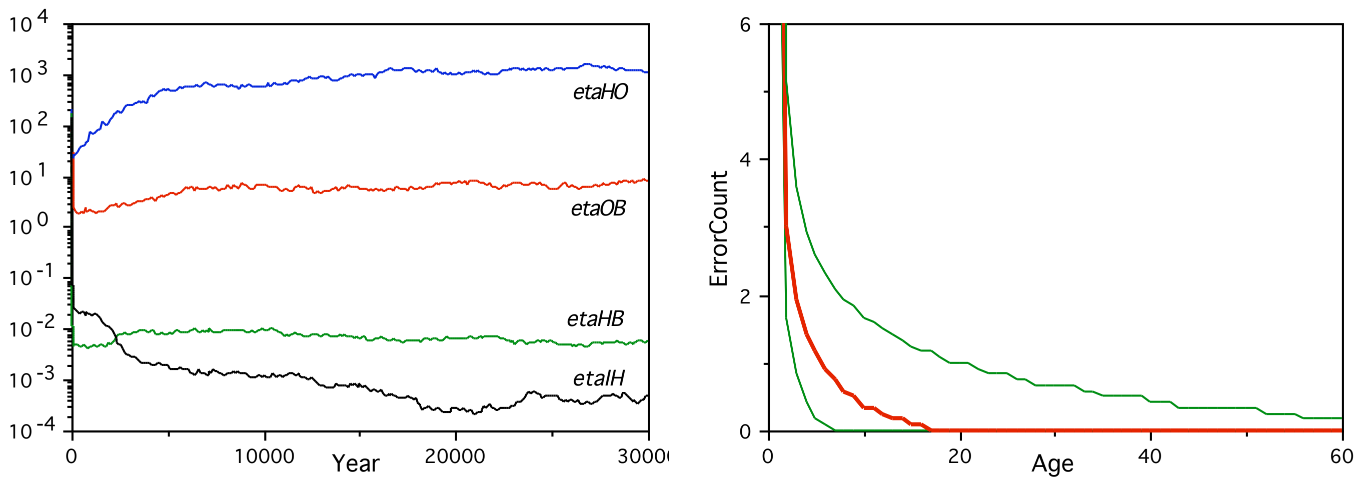


Figure 12: Evolution in the steady state *SS* approach when the initial population parameters are set randomly in logarithm space (left), and the resulting median and quartile error rates (right).

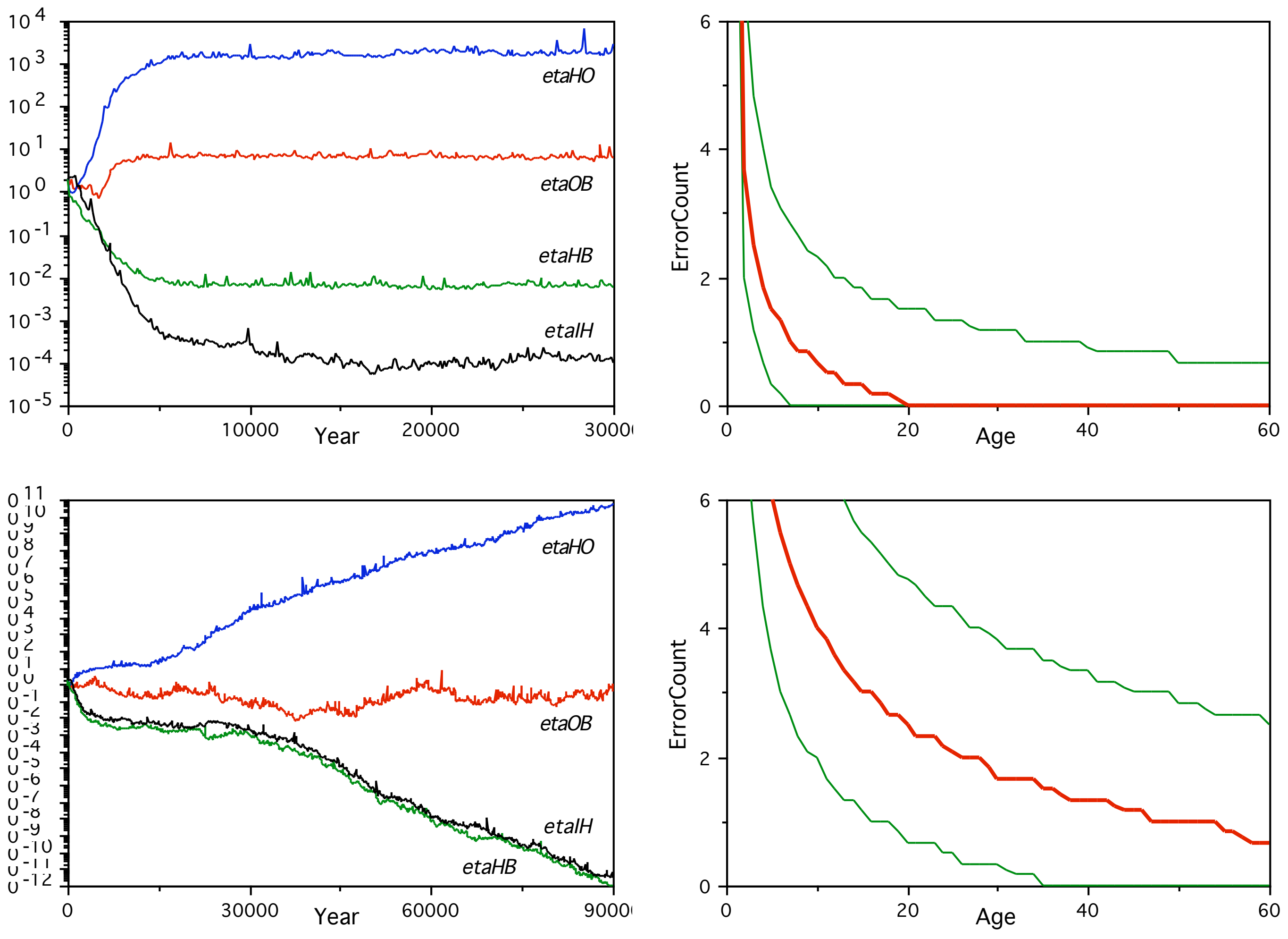


Figure 13: Evolution in the steady state *SS* approach with much larger mutation rates and magnitudes leads to two distinct types of evolved populations: fast learners (top graphs) and too-fast learners (bottom graphs).

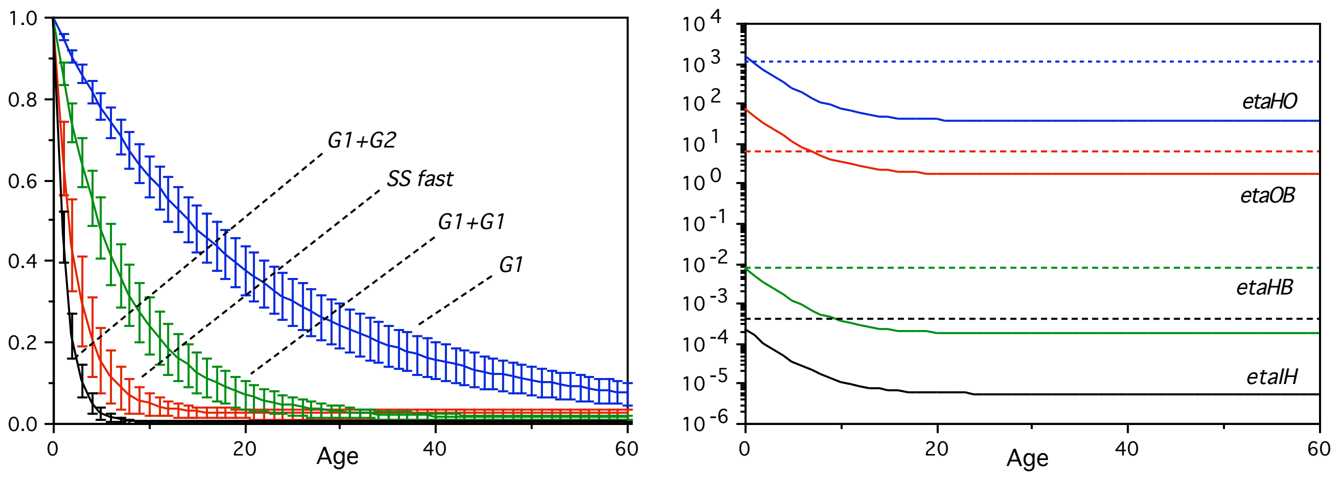


Figure 14: The evolved age dependent learning rate scale factors for the four evolutionary approaches (left), and the actual learning rates compared with the corresponding constant learning rates for the *SS fast* case (right).

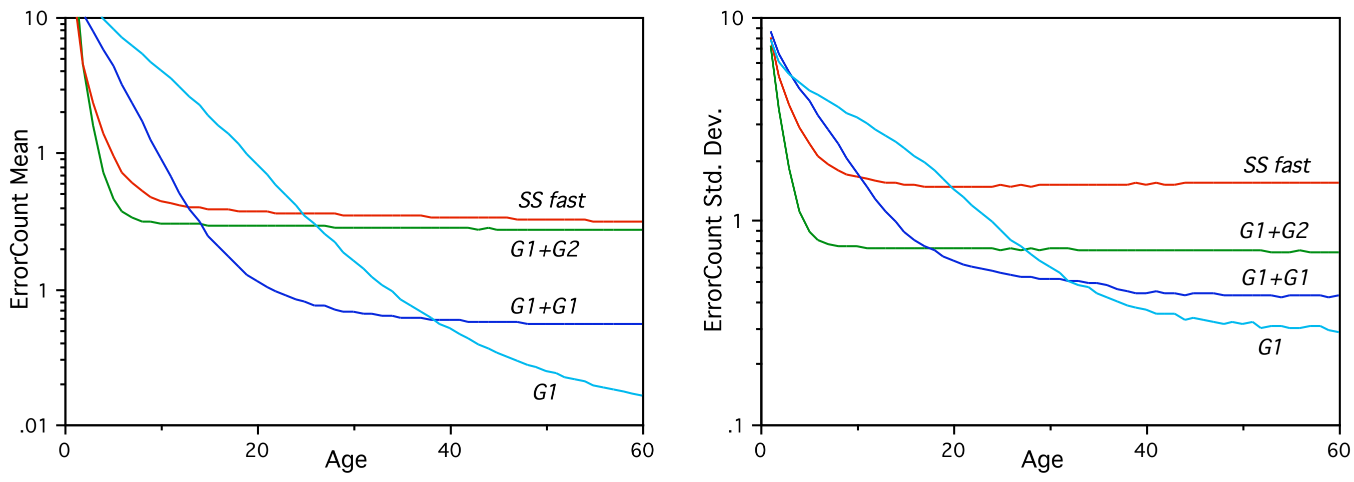


Figure 15: The means (left) and standard deviations (right) of the error count distributions for individuals during learning, for the four sets of evolved age dependent plasticity populations, for comparison with Figure 10.

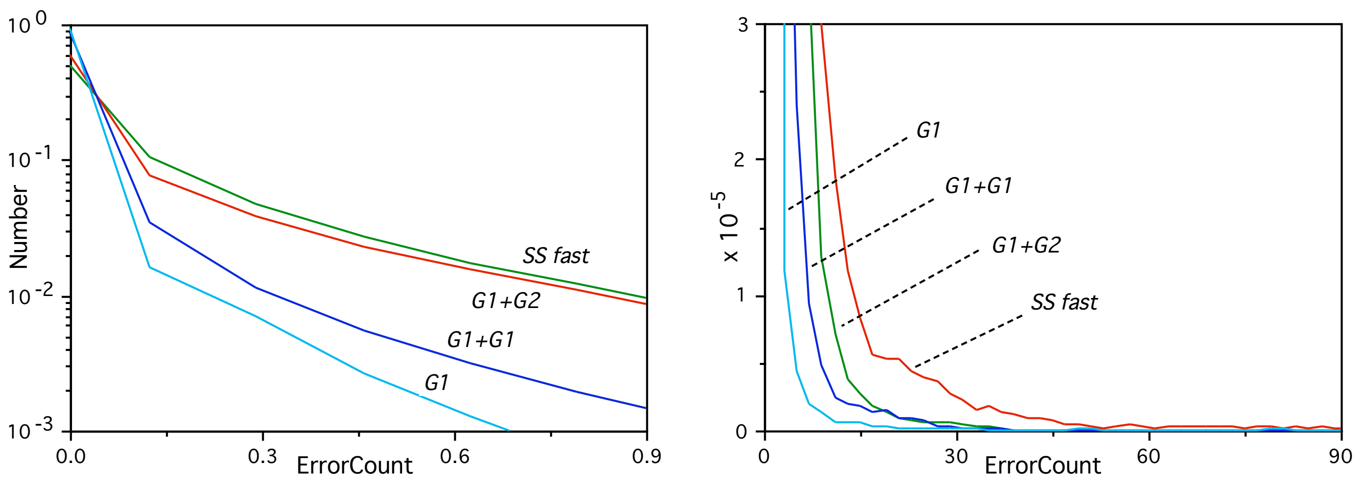


Figure 16: Two views of the average error count distributions for individuals between the ages of 50 and 60 for the evolved age dependent plasticity populations, for comparison with Figure 11: peaks (left) and tails (right).

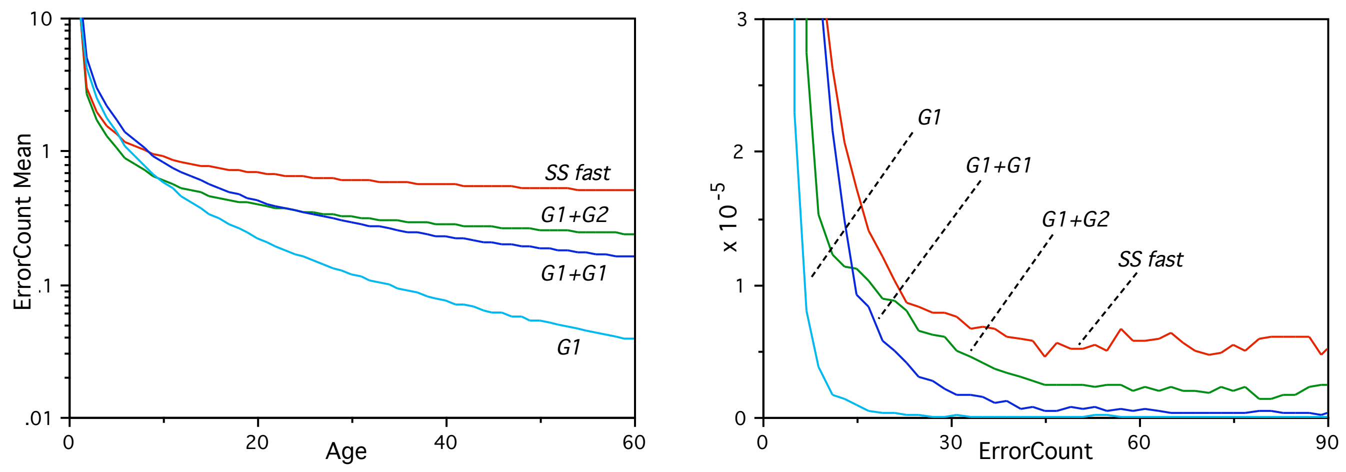


Figure 17. The mean error counts during learning when each evolved network uses voting over three runs starting from different initial weights (left), and the corresponding average error count distributions between ages 50 and 60 (right).

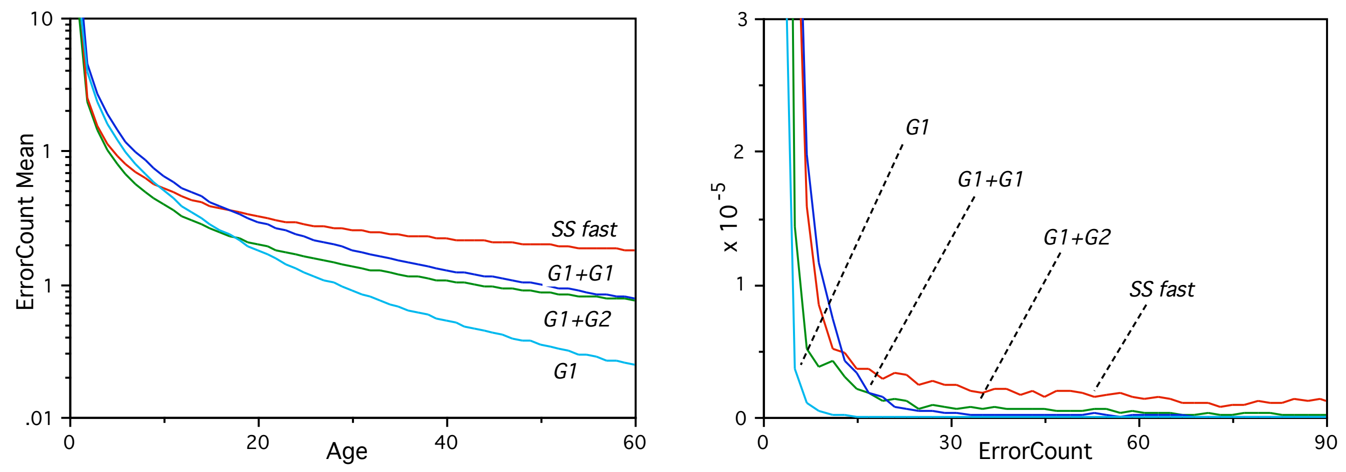
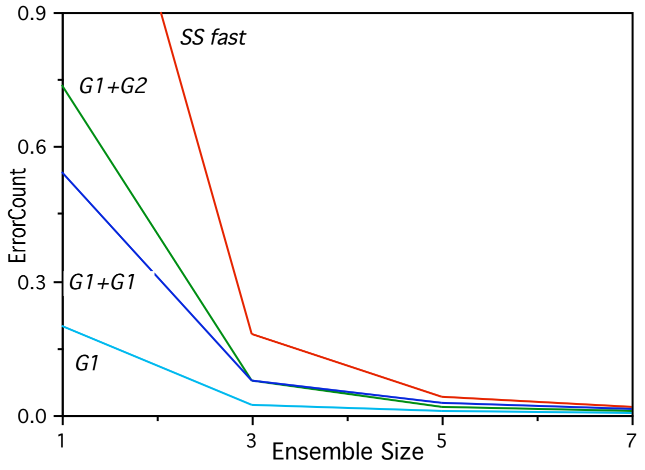


Figure 18. The mean error counts during learning for the evolved networks using voting in groups of three independent individuals (left), and the corresponding average error count distributions between ages 50 and 60 (right).



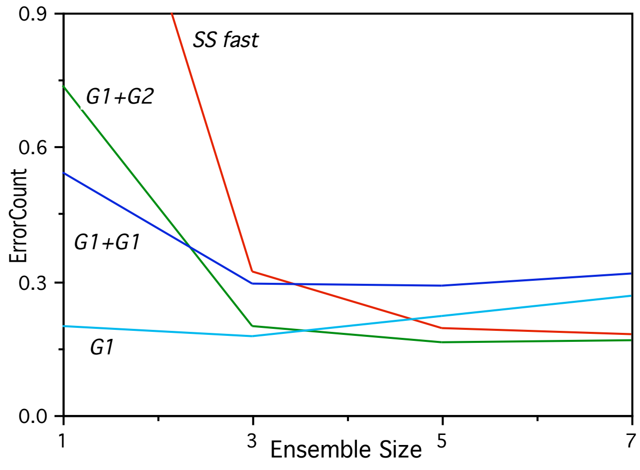
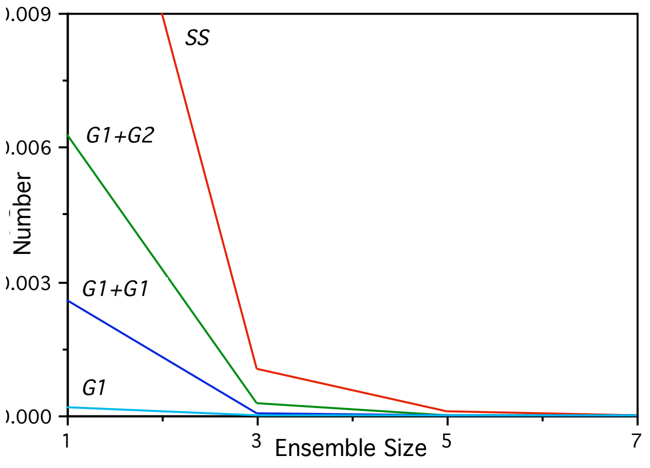


Figure 19. The mean error counts at age 60 for increasing ensemble sizes (left), and the corresponding error counts when the computational effort is taken into account by counting errors at age 60/*EnsembleSize* (right).



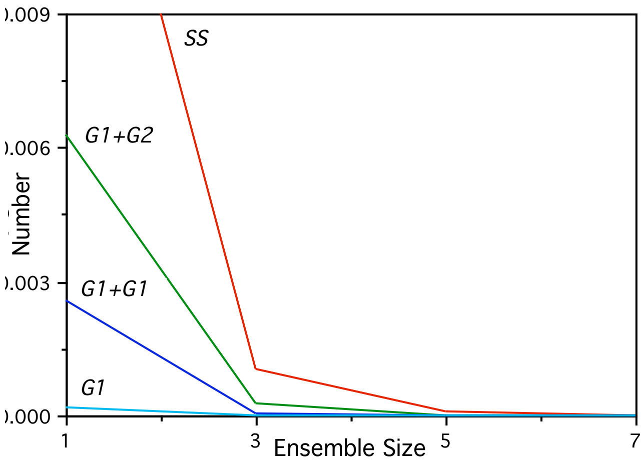


Figure 20. The total proportion of large error counts (above 30) between ages 50 and 60 (left), and the corresponding proportions when the computational effort is taken into account by using ages/*EnsembleSize* (right).

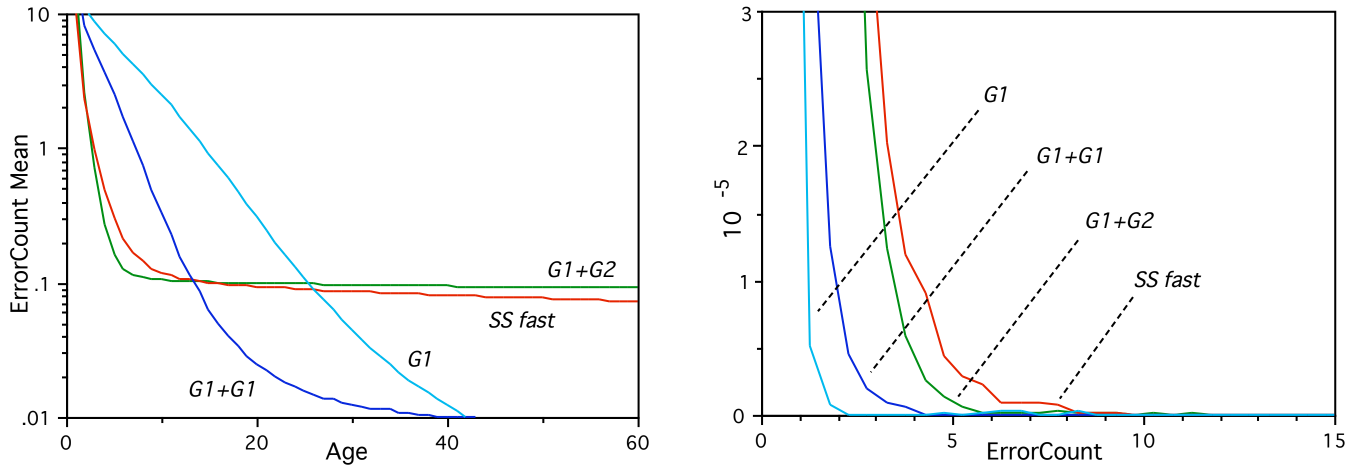


Figure 21. The mean error counts during learning for ensembles of three independent variable plasticity networks (left), and the corresponding average error count distributions between ages 50 and 60 (right).

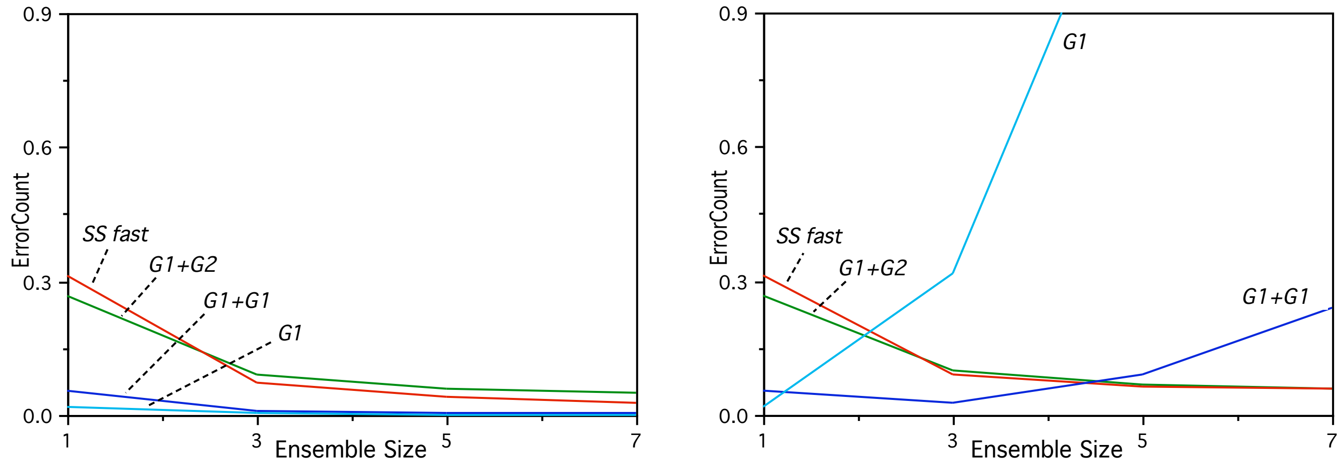


Figure 22. The mean error counts at age 60 for ensembles of variable plasticity networks (left), and the corresponding error counts when the computational effort is taken into account by counting errors at age 60/*EnsembleSize* (right).

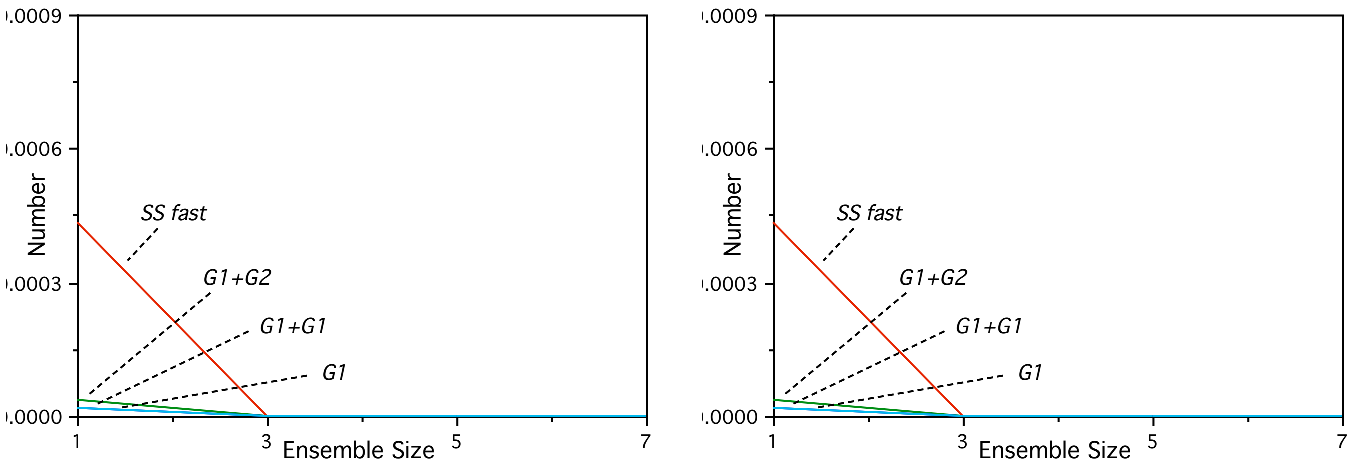


Figure 23. The total proportion of large error counts (above 30) between ages 50 and 60 for ensembles of variable plasticity networks (left), and the corresponding proportions found when using ages/*EnsembleSize* (right).