Using Redundancy to Improve the Performance of Artificial Neural Networks*

David A. Medler and **Michael R. W. Dawson**
Biological Computation Project
Department of Psychology, University of Alberta
Edmonton, Alberta
CANADA T6G 2E9

Abstract

For Artificial Neural Networks (ANNs) to be effective modelling tools, they must draw upon biological characteristics: One characteristic often overlooked in the design of ANNs is the replication, or redundancy, of processes within the brain. This paper examines the effects of redundancy on the performance of ANNs trained on either a pattern classification task (e.g. parity, encoder) or a function approximation task (e.g. forward kinematics). Results suggest that there is an optimal level of redundancy that increases the likelihood of network convergence while decreasing overall network processing time. ANNs with this level of redundancy consistently perform better than standard ANNs on pattern classification tasks. Furthermore, redundant ANNs trained on the function approximation task are more accurate in terms of overall system error than standard ANNs. These results imply that redundancy may be effectively used to increase the performance of ANNs, both in accuracy and speed.

1 Introduction

The design of Artificial Neural Networks (ANNs) is normally based on engineering principles-- make the system as simple and efficient as possible. From a strict computing science viewpoint, there is little wrong with this approach; however, from a cognitive science viewpoint, this approach is highly suspect. A growing number of scientists are now questioning this performance approach on the grounds that ANNs are moving away from their biological basis and are therefore losing validity as models in cognitive science [e.g. Lewandowsky, 1993; McCloskey, 1991]. To be effective models, ANNs must draw upon biological characteristics, especially those associated with the brain [Dawson and Shamanski, 1994; Dawson, Shamanski, and Medler, 1993]. One biological characteristic often overlooked in the design of ANNs is redundancy: Redundancy is the replication of processes within the brain.

Redundancy in biological systems has been documented since the nineteenth century when it was proposed that functional recovery following unilateral brain lesions was facilitated by replicated processes between the left and right hemispheres of the brain [Gall and Spurzheim, 1810-1819; cited in Almli and Finger, 1992]. Although we now know that the two hemispheres of the brain perform vastly different functions, redundancy within the two hemispheres is still held as a viable theory of functional recovery [Almli and Finger, 1992; Marshall, 1984]. More recently, studies of hydrocephalus patients suggest that normal psychological functioning is still evident with only half the normal brain tissue mass [Berker, Lorber, and Smith, 1983; cited in Glassman, 1987]. This implies that the brain is at least twice as large as is needed for immediate survival, and the extra baggage of the normal brain simply replicates functions already present.

Further neurophysiological evidence for redundancy in biological systems comes from recent studies of animal physiology. Kovac, Davis, Matera, and Croll [1983] found several physiological systems within the nervous system of *Pleurobranchaea californica* that produced essentially the same behavior; when combined, though,these systems greatly enhanced the precision of simple and complex movements. Furthermore,Strehler and Lestienne [1986] examined the firing patterns of neurons within a monkey's visual cortex and found redundant coding in the regularity of triplets of impulses triggered by specific stimuli. Similarly, Swindale [1986] noted that orientation selectivity in the visual cortex is produced by more than one mechanism, and in more than one location. This physiological evidence is complemented by a sizable theoretical literature on the biological relevance of redundancy.

The vast majority of theoretical work on the relevance of biological redundancy has centered on the factors surrounding the evolution of redundancy. Although one of the earlier assumptions concerning redundancy was that it allowed recovery of functioning following brain trauma, it is unlikely that such a rarely survived event like brain damage could exert any natural selection pressure for neural spare capacity [c.f.

* This research was supported by National Science and Engineering Research Council of Canada (NSERC) Research Grant 2038 and NSERC Equipment Grant 138704, awarded to MRWD.

Glassman, 1987]. If we assume that recovery from brain damage is just a convenient side effect of redundancy, we must consider other evolutionary stresses.

Most evolutionary changes involve small, simple changes that allow better adaptation to the surrounding environment; The faster a system can evolve, the greater the chance of survival. Therefore, it would be more advantageous to evolve several small systems that work in parallel to achieve a goal, than a large and highly specialized system [Swindale, 1986]. In fact, such a system is evident in the neural wirings of *Pleurobranchaea californica* [Kovac et al., 1983]. This view is echoed by Calvin [1983] who considered the evolution of neural timing systems required by early hominids for hunting with thrown objects. At short distances, a single timing neuron is sufficient to allow the proper release time needed for a hit; however, the timing precision required for a strike increases eight-fold with a mere doubling of throwing distance. Therefore, the brain combined the efforts of several timing neurons to increase the precision above that of any single neuron. Consequently, redundancy may have evolved not because brain damage was anticipated, but because it was easier to replicate, and thus improve, what was already present than to develop a single system beyond reproach.

A slightly different theoretical approach to redundancy comes from Jacobson [1976] and Glassman [1987]. Jacobson [1976] considered the connections between neurons involved in a memory trace based on Hebb's model of the cortex, and defined redundancy as "to mean the condition that pairs of cells joined along one effective pathway are joined again along another" (p. 150). Using mathematical calculations and assuming initial random connections between neurons, Jacobson showed that redundancy is an inevitable consequence of the connections within the cortex. Glassman [1987], on the other hand, looked at the probability of overall system failure for any large structure. With no redundancy, failure within a finite time is guaranteed; therefore, if the brain had no redundancy, the chances of it functioning for any significant amount of time are slim.

We have seen that redundancy is a viable, if not necessary, biological property, but can it be effectively implemented in ANNs? Recently, there has been a flurry of connectionist research on using multiple nets to solve problems (e.g. committees, agent teams, stacked generalization, model averaging, error correcting codes). As an example, Baxt's [1992] medical diagnosis network is based on two networks working in parallel: one network is trained to classify positive examples of myocardial infarction, and the other network is trained to classify negative examples. By combining their outputs, Baxt has produced a network that has a hit rate of 97.50% and a false alarm rate of 1.63%, which is considerably better than any human. Although this network is not strictly redundant (i.e. each network is trained on different pattern sets), it gives some indication of the increased accuracy associated with redundancy.

Another form of computational redundancy widely studied today centers around committee machines. Committee machines are based on the principle of using several computers (or networks) at once to solve the same problem. The training algorithm for such machines is rather unique [see Schapire, 1990]: Briefly, the first machine is trained on one pattern set, and then subsequent machines are trained on new pattern sets composed of equal amounts of correctly and incorrectly classified patterns that have been passed through previous machines. Once trained, however, there is little agreement as to the best way of combining the outputs of the different committee machines. Several alternatives have been suggested, from a simple "winner-take-all" or "voting" strategy, to summing the outputs, to calculating the mean output, to implementing a separate network to choose which machine's output is the most appropriate. Regardless of the combining strategy used, the committee machines invariably perform better than single networks alone.

The above research examples, however, have centered on improving the performance of ANNs from an engineering perspective solely. For example, it is not clear that any of the output strategies listed above, or even the training algorithms used for committee machines, are biologically plausible.

Constraints borrowed from biological networks, nevertheless, may have positive effects on the performance of ANNs as illustrated by Izui and Pentland's [1990] research on redundant networks. Using biological redundancy as a model, they mathematically analyzed the functional effects of one of neuronal duplication. Their mathematical calculations predict that redundant networks are more accurate, faster, and stable than standard networks. These predictions were confirmed by both a feedforward neural network trained on the XOR problem, and a feedback neural network trained on the travelling salesman problem. From these results, Izui and Pentland claim that the "highly redundant nature of biological systems is *computationally* important and not merely a side-effect of limited neuronal transmission speed and lifetime" (p. 237). Although Izui and Pentland's research has laid the mathematical foundations of network redundancy, their practical work requires expansion before redundancy is accepted as a useful addition in ANN design. For example, larger problem sets should be considered as well as the applicability of redundancy to different network architectures.

Three different questions are addressed by this current research. First, is there an optimal level of redundancy that improves ANN convergence without increasing the amount of processing required? Second, how do redundant ANNs fare on different classes of problems (e.g. pattern classification versus function approximation)? Third, can redundancy be effectively used with different types of network processing units (e.g. monotonic versus non-monotonic)? It is hypothesized that when the optimal level of redundancy is used, redundant networks will have better convergence on pattern classification problems than standard networks. Furthermore, redundant networks should be more accurate than standard networks on function approximation tasks. Finally, redundancy should be effective with both types of processing units.

2 Experiment 1: Levels of Redundancy

Adding redundancy to a network creates an interesting question from an engineering viewpoint: Are the added hardware requirements of adding extra processing nodes more than compensated for by an increase in performance? In other words, can we trade simplicity for efficiency? Theoretically, redundancy will not increase the overall network processing time as all redundant layers are working in parallel; however, the number of processing steps required will increase proportional to the number of redundant layers. Therefore, we can compare the performance of redundant ANNs to either the total processing time of a standard ANN, or to *1/N* processing sweeps of a standard ANN, where *N* is equal to the level of redundancy.

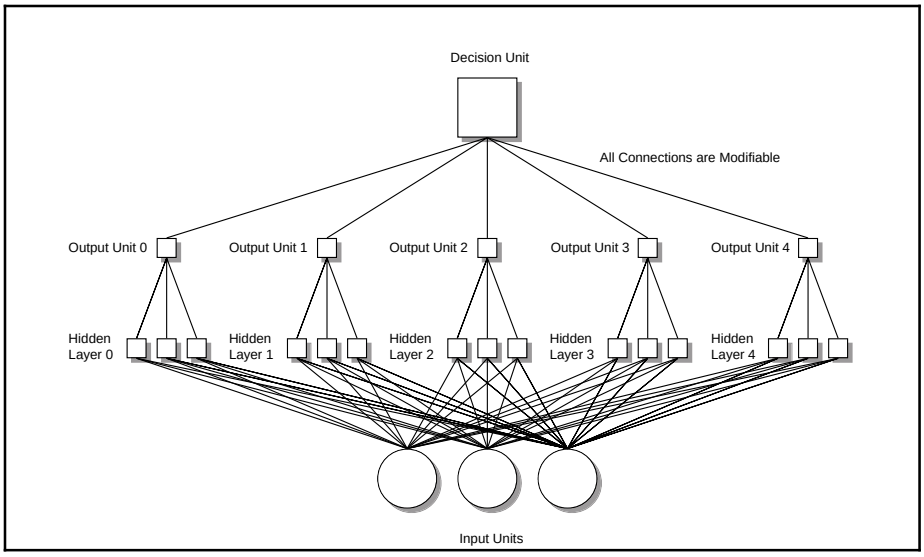

Figure 1. 3-Parity Network Structure with 5 Levels of Redundancy

The problem that now exists is to find the optimal level of redundancy where the increase in hardware requirements is offset by an equal or greater increase in performance. It has been calculated that the brain has at least two, and as many as seven, different layers of redundancy [Glassman, 1987]. Therefore, to assess the optimal level of redundancy for an ANN, the performance of a standard ANN trained on varying levels of a difficult pattern classification task (i.e. 2- to 8-parity) will be compared to the performance of ANNs with from two to eight levels of redundancy.

2.1 Method

2.1.1 Network Architecture

The *standard* network architecture consisted of an input layer, a hidden unit layer, and an output layer: The number of input units and hidden units was equivalent to the size of the parity problem (e.g. ANNs trained on 3-parity had 3 input units, 3 hidden units, and 1 output unit). Connection weights were randomly assigned from a rectangular distribution over the range [-1, +1], and processing unit biases were initialized to 0. All biases and connections within the network were modifiable.

The *redundant* network architecture was created by replicating the hidden unit layer and the output unit layer a set number of times. Each of the replicated output units was then connected to a *Decision Unit*, which acts as the redundant network's output unit. All connections leading into the Decision Unit are modifiable; therefore, the Decision Unit's response is a weighted sum of the replicated output units. Figure 1 shows the redundant network structure for an ANN with five levels of redundancy trained on a 3-parity problem. It should be noted that, as opposed to a three-layer network, no connections exist directly between each of replicated networks. Furthermore, each of the replicated networks was initialized independent of the others. Connection weights were randomly distributed over the range [-5, +5] to introduce more variability in the network, and all unit biases were set to 0. Seven different levels of redundancy were tested: 2, 3, 4, 5, 6, 7, and 8.

2.1.2 Training Stimuli

Parity is a linearly inseparable pattern classification task defined by the number of active input units: If the number of 1's in the input pattern is odd, then the output is 1, otherwise it is 0 [Minsky and Papert, 1969]. ANNs were trained on 2-, 3-4-, 5-, 6-, 7-, or 8-parity problems which had training set sizes of 2, 8, 16, 32, 64, 128, and 256 respectively. Each training set had equal numbers of positive and negative examples of parity.

2.1.3 Training Procedure

The network was trained with the backpropagation algorithm using the *generalized delta rule* (GDR) [see Rumelhart, Hinton, & Williams, 1986]. Backpropagation is described as a steepest descent optimization algorithm for traversing the surface of a weight space whose height measures error. Descent through the weight space is aided by two parameters: momentum (α) and rate-of-learning (η). Momentum is a technique for escaping local minima within the weight space by averaging the weight change for one item with the weight change for the previous item. The rate-of-learning parameter is used to dictate how large a "step" to make when traversing the weight space. For all parity problems, $\alpha = 0.9$, and $\eta = 0.1$.

To train the network, a pattern was randomly sampled--without replacement-- from the pattern set and presented to

the network. The network's actual output was then compared to the desired output, and connection weights and unit biases were modified according to the above algorithm. If the absolute difference between the actual output and desired output was less than 0.05 then a "hit" was recorded. One *sweep* of the network was completed once all patterns were presented to the network. Training of the network continued either until the maximum number of sweeps was completed (30,000) or until each pattern in a sweep produced a hit.

2.2 Results and Discussion

The performance of standard ANNs versus redundant ANNs was compared using three measures: probability of convergence, sweeps to convergence, and total processing steps to convergence. As can be seen from Table 1, the standard networks failed to classify 5-parity and above within 30,000 processing sweeps, whereas the redundant networks found a solution from 30% to 100% of the time on all parity problems. It should be noted that as the level of redundancy increased, so did the probability of convergence; however, only networks with 5 levels of redundancy converged 100% of the time on all sizes of parity problems.

Similarly, when sweeps to convergence is considered, there is a general decrease in sweeps with an increase in redundancy. Again, though, this decrease begins to asymptote around 5 levels of redundancy, which suggests a floor effect. A slightly different function appears with the total processing steps to convergence, calculated by multiplying the number of sweeps by the level of redundancy. This time, there is a slight quadratic function with its lowest points being around 4, 5, or 6 levels of redundancy depending on the problem difficulty. When the number of processing steps is averaged across all parity problems, networks with 4 levels of redundancy perform best, followed by networks with 5 and then 6 levels of redundancy.

Taking all of the above results into consideration, it appears that networks give best overall performance with 5 levels of redundancy for this type of linearly inseparable pattern classification problem. It does appear, however, that networks trained on easier problems (e.g. 2- and 3-parity) do not benefit, and may actually suffer, from redundancy. Nevertheless, these results show that the added hardware requirements of redundancy are more than compensated for by an impressive increase in performance. Although these results only generalize to the parity problem, all further experiments within this paper will adopt a redundancy level of 5.

Table 1. Median Processing Time and Steps to Convergence as a Function of Parity and Redundancy

	Level of Redundancy							
Problem	0	2	3	4	5	6	7	8
2 Parity								
Time	2551	932	841	819	663	641	660	491
Steps	2551	1864	2522	3276	3315	3846	4620	3928
n	10	10	10	10	10	10	10	10
3 Parity								
Time	1214	2235	973	630	587	688	447	526
Steps	1214	4470	2919	2520	2935	4128	3129	4208
n	10	9	10	10	10	9	10	10
4 Parity								
Time	14126	1732	1051	709	608	680	550	543
Steps	14126	3464	3153	2836	3040	4080	3850	4344
n	4	9	10	10	10	10	10	10
5 Parity								
Time	-----	1548	997	831	634	748	747	693
Steps	-----	3096	2991	3324	3170	4488	5229	5544
n	0	6	10	10	10	10	10	10
6 Parity								
Time	-----	1846	1492	932	738	821	720	470
Steps	-----	3692	4476	3728	3690	4926	5040	3760
n	0	8	10	10	10	10	10	10
7 Parity								
Time	-----	1826	1663	874	724	579	716	664
Steps	-----	3652	4989	3496	3620	3474	5012	5312
n	0	3	7	9	10	9	9	10
8 Parity								
Time	-----	1811	1835	884	758	579	692	566
Steps	-----	3622	5505	3536	3790	3474	4844	4528
n	0	4	9	8	10	7	10	9

Note. Maximum number of sweeps = 30000; n = number of converged networks out of 10.

3 Experiment 2: Pattern Classification

Experiment 1 has shown that redundancy improves the performance of ANNs trained on the parity problem; however, we do not know if theses results will generalize to other types of pattern classification problems, or if redundancy only improves the performance of networks with monotonic activation functions. Experiment 2 looked at the effects of redundancy on the number of network sweeps required for ANNs to learn two different types of difficult pattern classification tasks: 3-, 5-, 7-, and 9-parity, and 424-, 838-, and 16416-encoder. Furthermore, the effect of redundancy on the "standard" ANN architecture [e.g. Rumelhart, Hinton, and Williams, 1986] was compared to the effect of redundancy on a different architecture [Dawson and Schopflocher, 1992].

Originally, the backpropagation algorithm was developed under the assumption that the activation function for processing units had to be differentiable and monotonic [Rumelhart, Hinton, and Williams, 1986]; Such processing units are termed *integration devices* by Ballard [1986]. Recently, however, Dawson and Schopflocher [1992] have shown that processing units with a non-monotonic activation function-- called *value units* [Ballard, 1986]-- can learn pattern classification problems much faster than integration devices. Consequently, it is hypothesized that redundant ANNs will converge faster than standard ANNs, and that value unit ANNs will perform better then integration device ANNs. Therefore, the best performance is expected from the redundant value unit network.

3.1 Method

3.1.1 Network Architecture

The standard networks used for the parity classification problems were two-layer networks with one output unit. The number of input units and hidden units, however, were equivalent to the size of problem being solved: namely 3, 5, 7, or 9 units for the respective parity problem. Networks for the encoder problems had 4, 8, or 16 input and output units , and 2, 3, or 4 hidden units as dictated by the size of problem. Connection weights were randomized from a rectangular distribution over the range [-5, +5] for integration device networks using a sigmoidal activation function, or [-1, +1] for value unit networks using a Gaussian activation function. Processing unit biases, regardless of activation function, were initialized to zero. The redundant networks were created as described in Experiment 1.

3.1.2 Training Stimuli

The 3-, 5-, and 7-parity training sets used in this experiment were the same as in Experiment 1: A 9-parity set with 512 training patterns was also included. The training sets for the encoder problems consisted of 4, 8, or 16 orthogonal input patterns composed of a single 1 and filler 0's (e.g. 1 0 0 0, 0 1 0 0, 0 0 1 0, 0 0 0 1). The output patterns and input patterns were equivalent.

3.1.3 Training Procedure

The networks were trained with the backpropagation algorithm using either the GDR for processing units with a sigmoidal activation function [Rumelhart, Hinton, and Williams, 1986], or a modification of the GDR for processing units with a Gaussian activation function [Dawson and Schopflocher, 1992]. For the integration device networks, the parameters were set at $\alpha = 0.9$ and $\eta = 0.1$. Parameters for the value unit networks were $\alpha = 0$ and $\eta = 0.05$.

Training of the ANNs proceeded as described in Experiment 1. A hit was recorded if the actual output was 0.95 or higher when a 1 was desired, or 0.05 or lower when a 0 was desired, and the maximum number of sweeps before failure was set at 30,000. Second, the maximum number of sweeps allowed was held constant at 30000 for all networks. Training continued until all patterns in the set were learned or until the maximum number of sweeps was reached. As the initial random assignment of connection weights introduces variability in learning, each of the four different networks (standard vs. redundant, integration device vs. value unit) was initialized and trained 10 times. The minimum, median, and maximum number of sweeps to convergence, and the number of ANNs reaching convergence were recorded.

3.2 Results and Discussion

Table 2 shows the minimum, median, and maximum number of sweeps required to reach convergence and the total number of networks out of 10 to reach convergence for the 3-, 5-, 7- and 9-parity problems. The redundant networks solved the problems in fewer sweeps than the standard networks. Furthermore, the redundant networks converged on a solution 100% of the time while the standard networks often failed to converge on a solution even after 30000 sweeps. When equalized for the total number of network processing steps, the redundant networks only outperform the standard networks as the problem difficulty increases. Finally, it should be noted that the value unit networks converged much faster than the integration device networks for both the standard network architecture and the redundant network architecture. Also, the standard value unit networks converged on a solution more often than the standard integration device networks, particularly with the more difficult problems.

Table 2. Parity Problem: Sweeps to Convergence as a Function of Network Architecture and Processing Unit Type

	Network Architecture							
	Standard				Redundant			
Sweeps	3	5	7[a]	9	3	5	7	9
Integration Device ANNs								
Minimum	661	5817	-----	-----	261	576	353	397
Median	2599	6943	-----	-----	623	717	1237	1378
Maximum	24850	8068	-----	-----	1017	1047	2853	3695
n	8	2	0	0	10	10	10	10
Value Unit ANNs								
Minimum	49	213	1042	3052	37	34	71	360
Median	81	258	2744	5848	67	84	97	918
Maximum	200	1015	28322	14310	156	130	302	1449
n	10	9	8	6	10	10	10	10

Note. Maximum number of sweeps = 30000; n = number of converged networks out of 10.
a. Due to the difficulty of the 7- and 9-parity problems, different values of η were tried. Value units learned best with $\eta = 0.01$, whereas integration devices failed to learn at all values of η.

Similar results are obtained when we look at the different encoder problems. Redundancy decreases the amount of processing time and increases the likelihood of convergence for both integration device networks and value unit networks (see Table 3). When the number of processing steps are taken into consideration, however, redundancy does not help the integration device networks; On the other hand, value unit networks profit greatly from redundancy. In fact, the worst performance of the redundant value unit networks is better than the best performance of the redundant integration device

networks.

Table 3. Encoder Problem: Sweeps to Convergence as a Function of Network Architecture and Processing Unit Type

	Network Architecture					
	Standard			Redundant		
Sweeps	424	838	16416	424	838	16416
Integration Device ANNs						
Minimum	1200	2232	5143	353	588	827
Median	1399	3377	7177	514	787	2265
Maximum	4024	10399	22636	879	3060	8884
n	10	9	6	10	10	10
Value Unit ANNs						
Minimum	360	482	772	52	63	74
Median	545	775	944	77	96	95
Maximum	975	1088	1921	145	154	127
n	10	10	10	10	10	10

Note. Maximum number of sweeps = 30000; n = number of converged networks out of 10.

In conclusion, convergence on pattern classification problems is much faster with redundant ANNs than with standard ANNs. Furthermore, redundant ANNs converge on a solution 100% of the time regardless of problem type or size, whereas the standard ANNs often failed to reach convergence. Also, standard value unit networks converged more often than standard integration device networks When the networks are equalized for total number of processing steps, the redundant integration device networks only outperform the standard networks on the more difficult problems, whereas redundancy always improves the performance of value unit networks. These findings support Dawson and Schopflocher's (1992), conclusions the value unit networks outperform integration device networks on linearly inseparable pattern classification problems.

4 Experiment 3: Function Approximation

Experiment 1 and Experiment 2 have conclusively shown that redundancy can improve the performance of ANNs trained on difficult pattern classification tasks. The last question to be addressed is whether or not redundancy will improve the performance of ANNs trained on a function approximation task. With function approximation, however, the number of sweeps to reach convergence is no longer an appropriate measure of network performance; therefore, performance will be evaluated via overall network error.

The function approximation task chosen is based on Churchland's [1992] crablike creature which is programmed to reach to a point in space. Our simulated robotic arm, however, will use a neural network to essentially learn forward kinematics. Previous research [Calvin, 1983; Kovac et al., 1983] has suggested that redundancy in biological systems increases the accuracy of simple movements. Therefore, it is hypothesized that redundant ANNs trained on a function approximation task will have less error in responding than standard ANNs.

4.1 Method

4.1.1 Network Architecture

The standard network used was a two-layer network with two input units, two hidden units, and two output units. Connection weights were randomly assigned from a rectangular distribution over the range [-5,+5], and processing unit biases were initialized to zero. The redundant network was created as before with the exception that there were now two Decision Units to correspond with each replicated network's two output units.

4.1.2 Training Stimuli

A schematic diagram of the simulated robot and the problem space is shown in Figure 2. An object was placed randomly in front of the simulated robot: If the object fell within an unreachable area (i.e.grey area in Figure 2) then a new position was randomly chosen. Inputs to the network were the two angles (μ, ω) that the eyes subtended when converged on the object, while the desired network outputs were the angles (ρ, ϕ) that the shoulder joint and elbow joint made

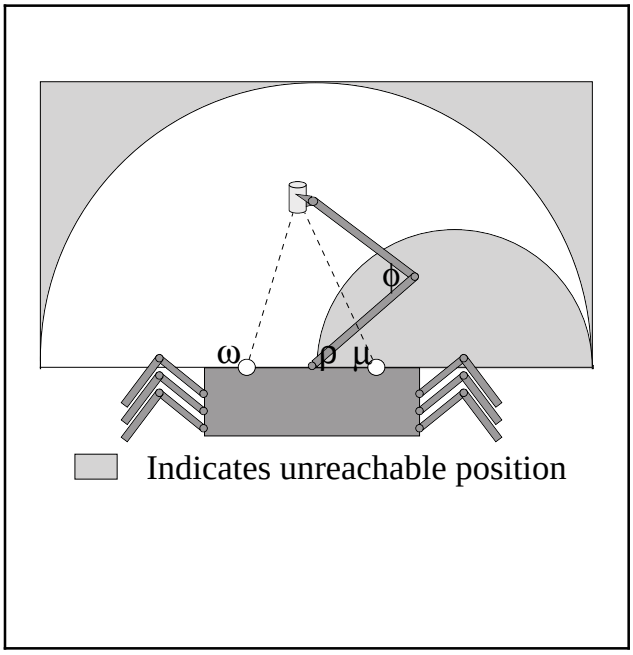

Figure 2. Definition of Problem Space for Simulated Robotic Arm.

in order for the arm to contact the object. All angles were normalized to fall within the range of 0 to 1. The inputs could be considered two-dimensional sensory-state space coordinates, and the outputs would then be considered as separate two-dimensional motor-state space coordinates. The network, therefore, learns the appropriate mapping between the two state spaces [see also Zipser and Anderson, 1988]. As the mapping of the two state spaces forms a continuous function, there are an infinite number of input/output pairs; however, practicality limited the training set to 50 randomly chosen pairs.

4.1.3 Training Procedure

Training of the networks proceeded as in Experiment 1 with some minor changes. First, a "hit" was defined when the absolute error between the desired output and the actual output was less than 0.001. Second, the networks were trained for 50,000 sweeps. Momentum and rate of learning for the simulated robotic arm were $\alpha = 0.9$, and $\eta = 0.1$.

To assess the network's ability to learn the function approximation problem, maximum network sweeps were increased from 100 to 50000 in $\log_{10}$ steps. Total network sum of squared errors (SSE)-- as measured by the difference between desired and actual network response-- as well as the SSE for each individual output (ρ, ϕ), were recorded at each maximum sweep step. As the initial randomness of connection weight assignment produces great variability in network learning, five different networks were trained for both the standard network architecture and the redundant network architecture.

4.2 Results and Discussion

The total SSE range and median for the simulated robotic arm for both the standard and redundant networks are shown in Figure 3. As can be seen, median SSE decreases faster for the redundant networks than for the standard networks. In fact, the average median SSE for the redundant network is significantly less than the average median SSE for the standard network ($\bar{x}$ = 0.355, 0.892 respectively; $\underline{F}$ (1,44) = 23.899, $\underline{p} < .001$). Furthermore, the range of the total SSE is significantly less for the redundant network than for the standard network ($\underline{F}$ (1,44) = 23.90, $\underline{p} < .001$). This holds true for both the elbow joint ϕ ($\underline{F}$ (1,44) = 15.95, $\underline{p} < .001$) and the shoulder joint ρ ($\underline{F}$ (1,44) = 6.91, $\underline{p} < .05$).

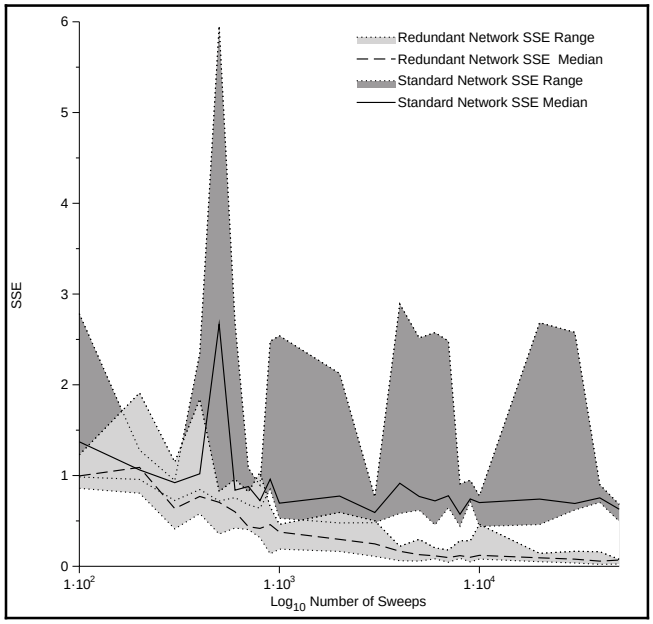

Figure 3. Median and Range of Network Error for the Simulated Robotic Arm

As hypothesized, redundant networks are significantly more accurate than standard networks on function approximation problems. Not only is the median SSE less for the redundant ANNs than for the standard ANNs, but the SSE range is significantly less for redundant networks as well. This means that the responding of the redundant ANNs is much less variable-- or more stable-- than the responding of the standard ANNs. Also, the advantage for the redundant ANNs increases with the number of sweeps completed when total network processing time is considered. This advantage even holds at the higher end of the sweep scale when the redundant and standard networks are equalized for total number of processing steps.

5 General Discussion

The results from both the PC problem and the FA problem confirm Izui and Pentland's [1990] mathematical analysis of redundant networks: Redundancy produces faster convergence, more accurate results, and more stable networks than comparable standard networks. In terms of the relevance of redundancy to the performance of ANNs in general, redundant networks should be considered as a viable alternative to standard networks. The initial cost of the extra hardware associated with redundancy is far out-weighed by the savings in training, accuracy in responding, and network stability produced by redundant processes.

Our results have shown that there is another alternative to the combining algorithms used by committee machines (e.g. mean response, winner-take-all, median response, etc.). The modifiable connections from the individual output units to the decision unit allows the network to train itself. As opposed to taking the mean output response of individual networks, which gives equal weighting to all networks, the amount of contribution is weighted according to how well the individual networks classify the problem. Furthermore, all individual networks contribute to the final result, unlike winner-take-all or median response methods. Consequently, the modifiable connections of the decision unit have proven to be a functional alternative to those methods conventionally used while preserving some semblance of biological systems.

Some evolutionary theories are supported by the performance of the redundant ANN. For example, the increased precision of the redundant network over the standard network on the FA problem

lends credence to Calvin's [1983] hypothesis about redundancy evolving to increase the precision of a system. In fact, as the upper limit of network sweeps increases, the worst redundant network is more precise than the best standard network. Also, the number of sweeps to train both the FA network and the PC network suggests that it is easier to evolve several crude mechanisms working in parallel than one extremely effective mechanism [Swindale, 1986].

Further research will consider the possibility of loss of redundancy accounting for loss of functioning in patients with debilitating diseases. As stated earlier, it is widely held that redundancy in the brain allows for functional recovery after brain damage [Almli and Finger, 1992]. It follows that loss of redundancy may cause loss of functioning. Modelling redundancy via computer simulation has a distinct advantage over biological models, in that precise ablations can be performed on artificial neural networks. Therefore, predications can be made about the performance of biological functioning when redundancy is compromised.

For example, the results of Experiment 3 show that the variability in making a response is much greater for a non-redundant network than a fully redundant network; therefore, as redundancy decreases, variability in responding should increase. Monitoring the variability changes should be an effective tool for estimating how much damage the system has suffered, and should even predict when terminal drop will occur. A practical application of this theory has already been hinted at by Patterson, Foster, and Heron, who conclude that for assessing damage by Multiple Sclerosis, "variability is a more sensitive indicator of visual pathway damage than the usual measure of mean" (1980, p.143). By attempting to model this increase in variability, we may be in a better position to understand the underlying damage associated with such diseases as Multiple Sclerosis and Alzheimers.

References

Almli, C. R., & Finger, S. (1992). Brain injury and recovery of function:Theories and mechanisms of functional reorganization. *Journal of Head Trauma Rehabilitation, 7*, 70-77.

Ballard, D. (1986). Cortical structures and parallel processing: Structure and function. *TheBehavioral and Brain Sciences*, *9*, 67-120.

Baxt, W. G. (1992). Improving the accuracy of an artificial neural network using multipledifferently trained networks. *Neural Computation, 4*, 772-780.

Calvin, W. H. (1983). A stone's throw and its launch window: Timing precision and its implications for language and hominid brains. *Journal of Theoretical Biology, 104*, 121-135.

Churchland, P. M. (1992). *A neurocomputational perspective: The nature of mind and the structure of science*. Cambridge, MA: MIT Press.

Dawson, M. R. W., & Schopflocher, D. P. (1992). Modifying the Generalized Delta Rule to train networks of non-monotonic processors for pattern classification. *Connection Science, 4*, 19-31.

Dawson,M.R.W., & Shamanski, K. S. (1994). Connectionism, confusion, and cognitive science. *Journal of Intelligent Systems, In press.*

Dawson, M.R.W., Shamanski, K. S., & Medler, D. A. (1993). From connectionism to cognitive science. In L. Goldfarb (Ed.) *Proceedings of the Fifth University of New Brunswick Artificial Intelligence Symposium.* Fredericton, NB: UNB Press.

Glassman, R. B. (1987). An hypothesis about redundancy and reliability in the brains of higher species: Analogies with genes, internal organs, and engineering systems. *Neuroscience & Biobehavioral Reviews, 11*, 275-285.

Izui, Y., & Pentland, A. (1990). Analysis of neural networks with redundancy. *Neural Computation, 2*, 226-238.

Jacobson, J. Z. (1976). Relative possibilities of loops and redundant connections in neural nets. *Journal of Mathematical Psychology, 13*, 148-162.

Kovac, M.P., Davis, W.J., Matera, E.M., & Croll, R.P. (1983). Organization of synaptic inputs to paracerebral feeding command interneurons of *Pleurobranchaea californica.* I. Excitatory inputs. *Journal of Neurophysiology, 49*, 1517-1538.

Lewandowsky, S. (1993). The rewards and hazards of computer simulations. *Psychological Science, 4*, 236-243.

McCloskey, M. (1991). Networks and theories: The place of connectionism in cognitive science. *Psychological Science, 2*, 387-395.

Marshall, J. F. (1984). Brain function: neural adaptations and recovery from injury. *Annual Review of Psychology, 35*, 277-308.

Minsky, M. L., & Papert, S. A. (1969). *Perceptrons*. Cambridge, MA: MIT Press.

Patterson, V. H., Foster, D. H., & Heron, J. R. (1980). Variability of visual threshold in Multiple Sclerosis: Effect of background luminance on frequency of seeing. *Brain: A Journal of Neurology, 103*, 139-147.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D E. Rumelhart, J. L. McClelland, and the PDP Group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition. Vol 1*. (pp 318-362). Cambridge, MA.: MIT Press.

Schapire, R. (1990). The strength of weak learnability. *Machine Learning, 5*, 197-227.

Strehler, B. L., & Lestienne, R. (1986). Evidence on precise time-coded symbols and memory of patterns in monkey cortical neuronal spike trains. *Proceedings of the National Academy of Sciences of the United States of America, 83*, 9812-9816.

Swindale, N. V. (1986). Parallel channels and redundant mechanisms in visual cortex. *Nature, 322*, 775-776.

Zipser, D., & Andersen, R. A. (1988). A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature, 331*, 679-684.