To Modularize or Not To Modularize?

John A. Bullinaria

School of Computer Science, The University of Birmingham
Edgbaston, Birmingham, B15 2TT, UK
j.bullinaria@physics.org

Abstract

There is a considerable degree of modularity in the human brain and numerous reasons why it might have evolved to be that way. One may be tempted to build such modularity into our artificial systems. Indeed, Rueckl, Cave & Kosslyn (1989) demonstrated how a clear advantage in having a modular architecture can exist in neural network models. Here I present a series of simulations of the evolution of such neural systems that show how the advantage *can* cause modularity to evolve. However, I shall also show that it is possible to evolve even more efficient systems that are not modular. It seems that making the decision whether to build modularity into our systems is not as easy as is often thought.

1 Introduction

When building complex intelligent systems, one has numerous design decisions to make. Given the obvious potential for disruptive interference, it seems quite reasonable that two independent tasks will be more efficiently carried out separately by two dedicated modules, rather than together by a homogeneous (fully distributed) system. Certainly there is considerable neuropsychological evidence that human (and other animal) brains do operate in such a modular manner (e.g. Shallice, 1988). There is still some controversy over the precise form that modularity takes, and it now seems unlikely to be of the strong (hard-wired, innate and informationally encapsulated) type of Fodor (1983). Nevertheless, some form of modularity has apparently arisen as the result of evolution by natural selection, and one might think it a good strategy to use the end point of such an evolutionary process (i.e. modularity) in building our artificial systems. In this paper I begin to question whether this really is a good idea.

Connectionist neural network systems are the natural starting point for AI systems built to operate in a manner analogous to human brains, so I shall look at that type of system. And, for concreteness, I shall concentrate on one particularly well studied instance of modularity in the human brain, namely the fact that visual perception involves two distinct cortical pathways (e.g. Mishkin, Ungerleider & Macko, 1983; Goodale & Milner, 1992; Ungerleider & Haxby, 1994) – one running ventrally for identifying objects ("what"), and another running dorsally for determining their spatial locations ("where").

Some time ago, Rueckl, Cave & Kosslyn (1989) considered the interesting question of why "what" and "where" should be processed by separate visual systems in this way. By performing explicit simulation and analysis of a series of simplified neural network models they were able to show that modular networks were able to generate more efficient internal representations than fully distributed networks, and that they learned more easily how to perform the two tasks. The implication was that any process of evolution by natural selection would result in a modular architecture and hence answer the question of why modularity has arisen in human brains. It also implied that hard wiring modularity into our AI systems, at least for neural network systems, is likely to be a good idea.

Now, twelve years later, the power of modern computer technology has reached a level whereby the relevant explicit evolutionary simulations are now feasible. Already Di Ferdinando, Calabretta & Parisi (2001) have established that modularity *can* evolve as expected. In this paper, I present the results of further simulations and conclude that, whilst modularity may arise under certain circumstances, the situation is not quite as straight-forward as the original computational investigation of Rueckl et al. (1989) suggested.

2 Avoiding Disruptive Interference

The basic structure of simple feed-forward neural network models is now widely known. One typically uses a three layer network of simplified neurons. The input layer activations represent the system's input (e.g. a simplified retinal image). These activations are passed via weighted connections to the hidden layer where each unit sums its inputs and passes the result through some form of squashing function (e.g. a sigmoid) to produce its own activation level. Finally, these activations are passed by a second layer of weighted connections to the output layer

where they are again summed and squashed to produce the output activations (e.g. representations of "what" and "where"). The connection weights are typically learnt by some form of gradient descent training algorithm whereby the weights are iteratively adjusted so that the network produces increasingly accurate outputs for each input in a representative set of training data.

From this point of view, the question of modularity relates to the connectivity between the network's hidden and output layers. During training, a hidden unit that is being used to process information for two or more output units is likely to receive conflicting weight update contributions for the weights feeding into it, with a consequent degradation of performance relative to a network that has a separate set of hidden units for each output unit (e.g. Plaut & Hinton, 1987). However, avoiding this kind of disruptive interference by using such an extreme version of modularity that allocates a separate set of hidden units (or module) for each output unit is likely to be rather inefficient in terms of computational resources, and an efficient learning algorithm should be able to deal appropriately with the conflicting weight update signals anyway. Nevertheless, splitting the hidden units up into disjoint sets corresponding to distinct output tasks, may be an efficient option. Indeed, it is hard to imagine how it could be optimal to expect a single set of hidden units to form more than one distinct internal representation.

When a neural network is trained using standard gradient descent type learning algorithms, one generally finds that the processing at the hidden layer tends to become fully distributed – in other words, there is no spontaneous emergence of modularity (e.g. Plaut, 1995; Bullinaria, 1997). However, the human brain is some-what more sophisticated than a simple feed-forward network learning by gradient descent, and Jacobs, Jordan & Barto (1991) have shown explicitly how it is possible to set up gated mixtures of expert networks that can learn to process two tasks in a modular fashion. Such systems appear to have advantages in terms of learning speed, minimizing cross-talk (i.e. spatial interference), minimizing forgetting (i.e. temporal interference), and generalization. In a further computational study, Jacobs & Jordan (1992) have shown how a simple bias towards short range neural connectivity can also lead to the learning of modular architectures.

For the present study I am more interested in identifying optimal fixed architectures for learning to perform the given tasks, rather than in building systems that are able to learn an appropriate architecture. Evolving learning systems by natural selection seems an obvious way to proceed. The old Nature-Nurture debate has certainly come a long way in recent years (e.g. Elman et al., 1996), but it is still important to distinguish which characteristics are innate and which need to be learnt during an individual's lifetime. Fortunately, as computer technology becomes more powerful, we are able to explore these issues by increasingly realistic simulations, and old ideas about the interaction of learning and evolution (e.g. Baldwin, 1896) can now be confirmed explicitly (e.g. Hinton & Nowlan, 1987). In suitably simplified systems, it has been possible to observe the genetic assimilation of learnt characteristics without Lamarckian inheritance, to see how appropriate innate values for network parameters and learning rates can evolve, to understand how individual differences across evolved populations are constrained, and so on (e.g. Bullinaria, 2001). In the remainder of this paper I shall investigate the evolution of modularity in the neural network models of the "what" and "where" vision tasks previously studied by Rueckl et al. (1989). The lessons we learn here should be applicable to the learning and evolution of modularity more generally, and give us some ideas of what structures might be most appropriate to build into our AI systems.

3 The "What" and "Where" Model

Rueckl et al. (1989) have already carried out an extensive series of investigations into the internal representations that are learned in simple neural network models of the "what" and "where" vision tasks. For ease of comparison, and minimization of repetition, I shall study exactly the same simplified model that they used. I shall explore whether the advantages of modularity they observed are sufficient to drive the evolution of modularity, and test the robustness of their results. I shall also follow Rueckl et al. (1989) and Jacobs et al. (1991) in emphasizing that the simulated tasks are vast over-simplifications of what real biological visual systems have to cope with. It makes sense to use them, however, despite their obvious unrealistic features, since they allow us to illustrate the relevant factors with simulations we can perform on current computational hardware in a reasonable amount of time.

Our chosen task consists of mapping a simplified retinal image (a 5×5 binary matrix) to a simplified representation of "what" (a 9 bit binary vector with one bit 'on') and a simplified representation of "where" (another 9 bit binary vector with one bit 'on'). I shall use the same 9 input patterns and 9 positions as in the previous studies, giving the same 81 retinal inputs for training on. Each of the 9 patterns consist of a different set of 5 cells 'on' within a 3×3 sub-retina array, and the 9 positions correspond to the possible centers of a 3×3 array within the full 5×5 array.

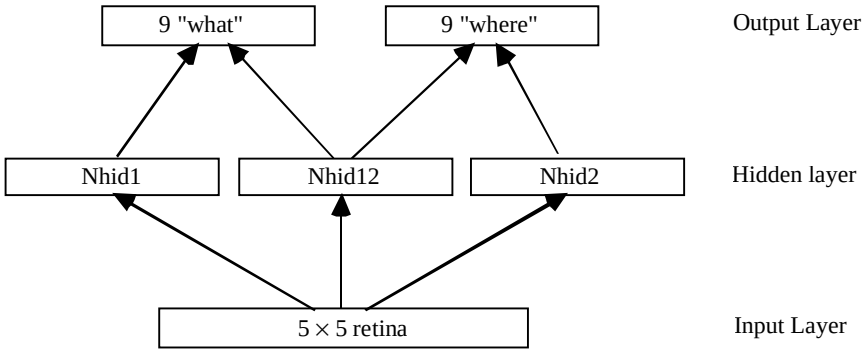

Figure 1: The architecture of the basic neural network model for the "what" and "where" tasks.

The basic network originally investigated by Rueckl et al. (1989) is shown in Figure 1. It has 25 input units, 18 output units and 81 training examples. The arrowed lines represent full connectivity, and *Nhid1*, *Nhid12*, *Nhid2* specify how many hidden units in each block. Rueckl et al. (1989) studied in detail the fully distributed network (*Nhid1 = Nhid2 =* 0) and the purely modular network (*Nhid12* = 0). Our characterization allows an exploration of the full continuum between these extremes. If the maximum number of hidden units *Nhid1 + Nhid12 + Nhid2 = Nhid* is fixed, we need only define two innate architecture parameters *Con1 = Nhid1 + Nhid12* and *Con2 = Nhid2 + Nhid12* corresponding to the number of hidden units connecting to each output block.

4 Evolving the Model

Simulating an evolutionary process for the models discussed above simply involves taking a whole population of individual instantiations of each model and allowing them to learn, procreate and die in a manner approximating these processes in real (living) systems. The genotype of each individual will depend on the genotypes of its two parents, and contain all the appropriate innate parameters. Then, throughout its life, the individual will learn from its environment how best to adjust its weights to perform most effectively. Each individual will eventually die, perhaps after producing a number of children.

For biological evolution, the ability of an individual to survive or reproduce will depend on a number of factors which can vary in a complicated manner with that individual's performance over a range of related and unrelated tasks (food gathering, fighting, running, and so on). For the purposes of our simplified model, however, we shall consider it to be a sufficiently good approximation to assume a simple relation between our single task fitness function and the survival or procreation fitness. Whilst any monotonic relation should result in similar evolutionary trends, we often find that, in simplified simulations, the details can have a big effect on what evolves and what gets lost in the noise.

A more natural approach to procreation, mutation and survival will be followed than many evolutionary simulations have used in the past (e.g. in Belew & Mitchell, 1996). Rather than training each member of the whole population for a fixed time and then picking the fittest to breed and form the next generation, the populations will contain competing learning individuals of all ages, each with the potential for dying or procreation at each stage. During each simulated year, each individual will learn from their own experience with the environment (i.e. set of training/testing data) and have their fitness determined. A biased random subset of the least fit individuals, together with a flat random subset of the oldest individuals, will then die. These are replaced by children, each having one parent chosen randomly from the fittest members of the population, who randomly chooses a mate from the rest of the whole population. Each child inherits characteristics from both parents such that each innate free parameter is chosen at random somewhere between the values of its parents, with sufficient noise (or mutation) that there is a reasonable possibility of the parameter falling outside the range spanned by the parents. Ultimately, the simulations might benefit from more realistic encodings of the parameters, concepts such as recessive and dominant genes, learning and procreation costs, different inheritance and mutation details, different survival and procreation criteria, more restrictive mate selection regimes, protection for young offspring, different learning algorithms and fitness functions, and so on, but for the purposes of this paper, the simplified approach outlined above seems adequate. A similar regime has already been employed successfully elsewhere (Bullinaria, 2001) to study the Baldwin effect in the evolution of adaptable control systems.

Each genotype in our simulation will naturally include all the innate parameters needed to specify the network details, namely the architecture, the learning algorithm, the learning rates, the initial

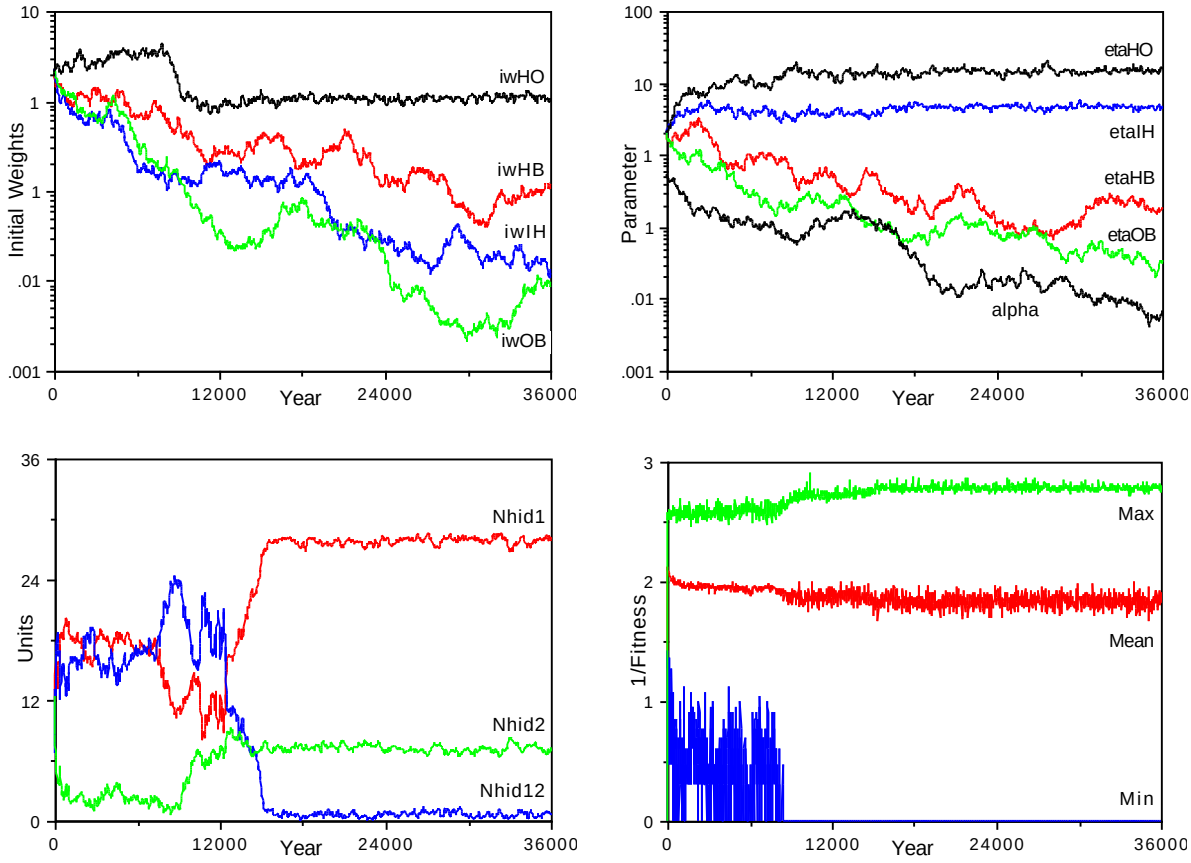

Figure 2: A typical evolution of the model in Figure 1 with 36 hidden units, Sum-Squared Error cost function and Log Error Count fitness function.

connection weights, and so on. In real biological evolution, all these parameters will be free to evolve. In simulations that are designed to explore particular issues, it makes sense to fix some of these parameters to avoid the complication of un-foreseen interactions (and also to speed up the simulations). In my earlier study of genetic assimilation and the Baldwin effect (Bullinaria, 2001), for example, it made sense to keep the architecture fixed and to allow the initial innate connection weights and learning rates to evolve. Here it is more appropriate to have each individual start with random initial connection weights and allow the architecture to evolve. Then, since the optimal learning rates will vary with the architecture, we must allow these to evolve along with the architecture. Moreover, the appropriate ranges for the random initial weights may also depend on the architecture and learning rates, so we must allow these to evolve as well.

Clearly it is important that we fix all the evolutionary parameters appropriately according to the details of the problem and the speed and coarseness of the simulations. For example, if all individuals were able to learn the task perfectly by the end of their first year, and we only tested their performance once per year, then the advantage of those that learn in two months over those that take ten is lost and our simulated evolution will not be very realistic. Since the networks were allowed to evolve their own learning rates, this had to be controlled by restricting the number of presentations of the training data set to two per simulated year for each individual. A fixed population size of 200 was a trade-off between maintaining genetic diversity and running the simulations reasonably quickly. The death rates were set in order to produce reasonable age distributions. This meant about 20 deaths per year due to competition, and another 5 individuals over the age of 20 dying each year due to old age. The mutation parameters were chosen to speed the evolution as much as possible by maintaining genetic diversity without introducing too much noise into the process. These parameter choices led to coarser simulations than one would like, but otherwise the simulations would still be running.

5 Results for the Basic Model

The obvious starting point was to simulate the evolution of the basic system as described above. For comparison purposes, this involved fixing the learning algorithm to be that used by Rueckl et al. (1989), namely online gradient descent with momentum using the Sum Squared Error cost

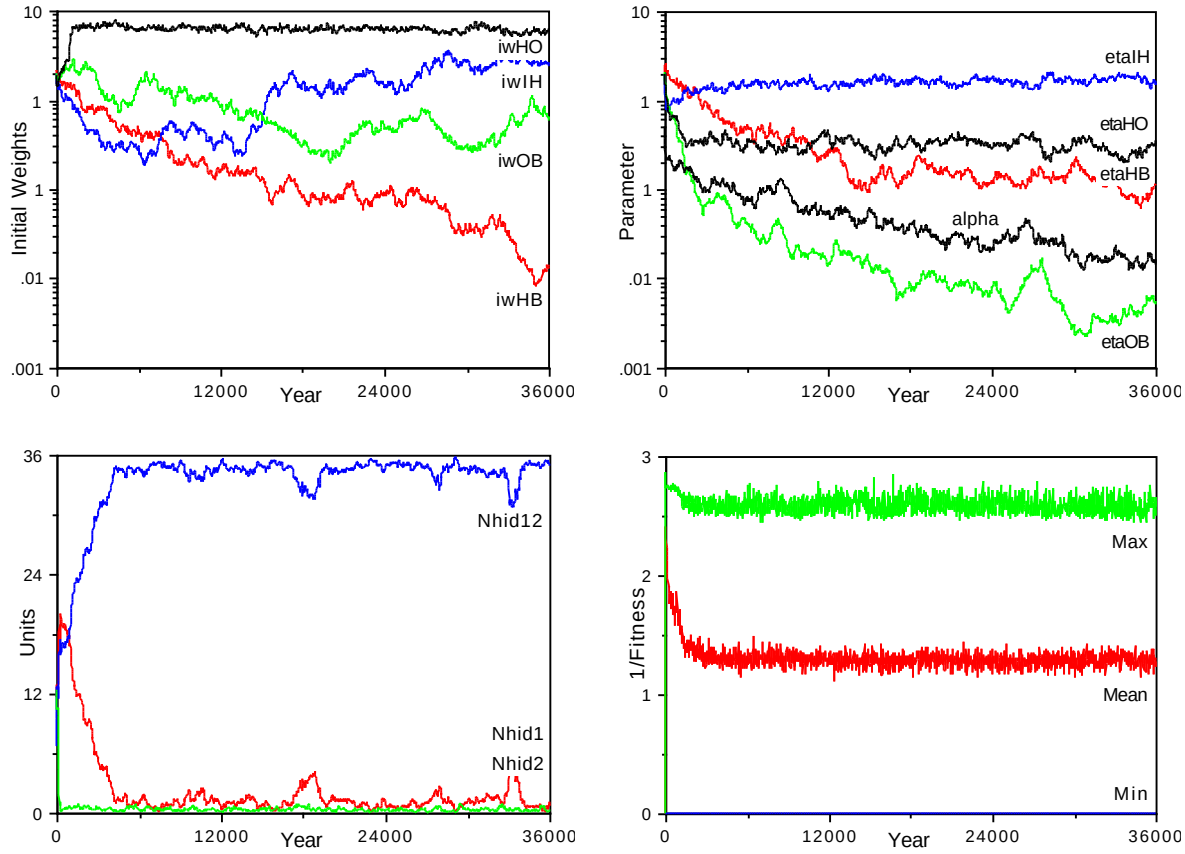

Figure 3: A typical evolution of the model in Figure 1 with 36 hidden units, Cross Entropy cost function and Log Error Count fitness function.

function *E* (Hinton, 1989). Also following them, the target outputs were taken to be 0.1 and 0.9, rather than 0 and 1, and appropriate outputs beyond these targets were deemed errorless. Past experience indicates that the networks learn better if they have different learning rates for each of the different connection layers, and each of the different bias sets. So, to ensure that the architecture comparisons were fair in the sense that they were all learning at their full potential, each network had five learning parameters: the learning rate η_{IH} for the input to hidden layer, η_{HB} for the hidden layer biases, η_{HO} for the hidden to output layer, and η_{OB} for the output biases, and the momentum parameter α. These appear in the standard weight update equation

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

The initial network weights $w_{ij}(0)$ were generated randomly with a uniform distribution from the range [*-iw*, *+iw*]. Naturally, different range parameters *iw* were allowed for the input to hidden layer connections, the hidden layer biases, the hidden to output layer connections, and the output biases. Each genotype thus contained two parameters to control the network architecture, five to control its learning rates, and four to control the distribution of random initial weights.

The next consideration was the choice of fitness measure. The obvious measure is in terms of the number of network outputs over the whole training set that are significantly wrong (e.g. more than 0.2 from their binary targets). The distribution of errors actually becomes very skewed over the population so an appropriate fitness measure was chosen to be 1/log(1+*ErrorCount*).

My earlier evolutionary studies (Bullinaria, 2001) indicated that the evolution can have a strong dependence on the initial conditions, i.e. on the distribution of the innate parameters across the initial population, and that the population settles into a near optimal state more quickly and reliably if it starts with a wide distribution of initial learning rates, rather than expecting the mutations to carry the system from a state in which there is little learning at all. Thus, in all the following experiments, the initial population learning rates were chosen randomly from the range [0.0, 4.0], the momentum parameters randomly from the range [0.0, 1.0], and the random initial weight ranges from the range [0.0, 4.0].

Figure 2 shows how the innate parameters evolved for a typical simulation when there were 36 hidden units in total. Rueckl et al. (1989) used only 18 hidden units, but this was considered too near the

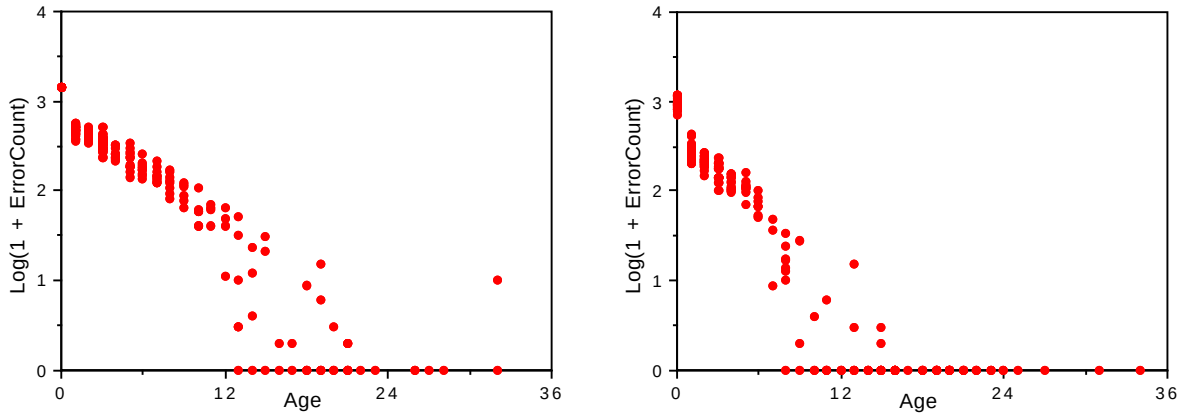

Figure 4: Comparison of learning speeds for the evolved populations with Sum Squared Error (left) and Cross Entropy (right) cost functions.

minimal number needed to perform the task for small scale artifacts to be reliably avoided (e.g. Bullinaria, 1997). The top two graphs show that the learning parameters soon settle down to appropriate values and confirm our intuition that different learning rates and ranges of initial weights are appropriate for the different sets of connections and biases. For comparison, Rueckl et al. (1989) chose all their random initial weights and biases uniformly from within the range [–0.3,+0.3], all the learning rates to be 0.75, and the momentum to be 0.8. The third graph shows that after a hesitant non-modular start, the population quickly evolves to take on a modular architecture with *Nhid12* near zero. This is exactly what we would expect from the Rueckl et al. (1989) study, right down the to way the hidden units are split between *Nhid1* and *Nhid2* according to the relative difficulties of the two tasks. The final graph shows how the log(1+*ErrorCount*) = 1/*Fitness* varies during the evolutionary process, and confirms that members of the evolved population do manage to learn the given tasks.

6 Dependence on Cost Function

The fact that the above results confirm what we would expect from the Rueckl et al. (1989) study may make the evolution of modularity seem almost inevitable. However, the aim of this paper is to begin an investigation into whether this is really the case. The starting point I chose was to note that, if we treat our network outputs as specifying a probability distribution over the binary targets, then we should really use the Cross-Entropy cost function for our gradient descent learning rather than the Sum-Squared Error cost function used above (Hinton, 1989). Figure 3 shows the result of a typical evolutionary simulation using the Cross Entropy cost function, with everything else the same as before. Not surprisingly, different patterns of initial weights and learning parameters evolve. What was less expected was the evolution of a completely non-modular architecture with *Nhid1* and *Nhid2* both very close to zero. Moreover, the mean *Fitness* indicates that this population is much fitter than the population trained with the Sum Squared Error cost function.

Figure 4 compares explicitly the speeds of learning for the final populations evolved using the Sum Squared Error and Cross Entropy cost functions. This shows the clear superiority of the non-modular Cross-Entropy population, and explains why that population has a better average fitness. Members of the modular population typically perfect the tasks between the ages of 13 and 23, whereas the non-modular population reach those performance levels between the ages of 8 and 16. Further simulations that explored other potential system variations, such as changing the fitness measure between 1/log(1+*ErrorCount*), 1/*ErrorCount*, –*Cost*, 1/*Cost*, and 1/log(*Cost*) had little effect on the results.

7 Analysis of Learning Speeds

Although we cannot rely on the mean learning times to predict what will evolve (since the worst and best cases, the standard deviations, the reliability, and so on, will also be important), plots of mean learning times as a function of the network architecture do show quite clearly where the different optimal configurations are situated. Obviously we have to be careful, because the different architectures may have significantly different optimal learning and initial weight parameters, but we shall see that it is possible to proceed with this approach.

Figure 5 shows contour plots of the mean number of simulated years of training required to achieve zero *ErrorCount* for the evolved populations from the same simulations presented in Figures 2, 3 and 4. In the left hand graph, for the Sum Squared Error cost function, we see that the optimal configurations (shown darkest) are situated in the *Nhid12* = 0, i.e. totally modular, region. In the right hand graph, for

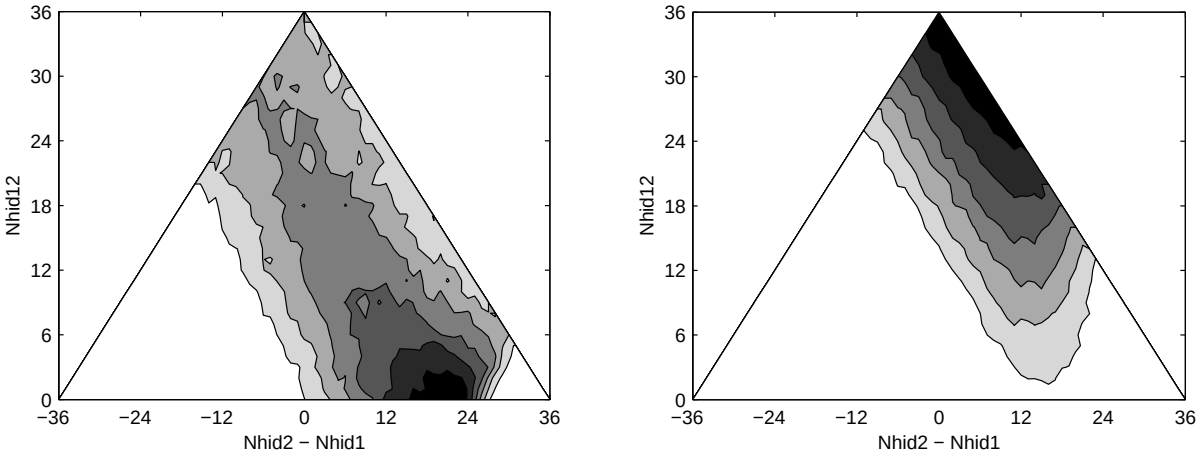

Figure 5: Mean learning times using evolved learning parameters as a function of architecture with Sum Squared Error (left) and Cross Entropy (right) cost functions. Darkest shading indicates fastest learning.

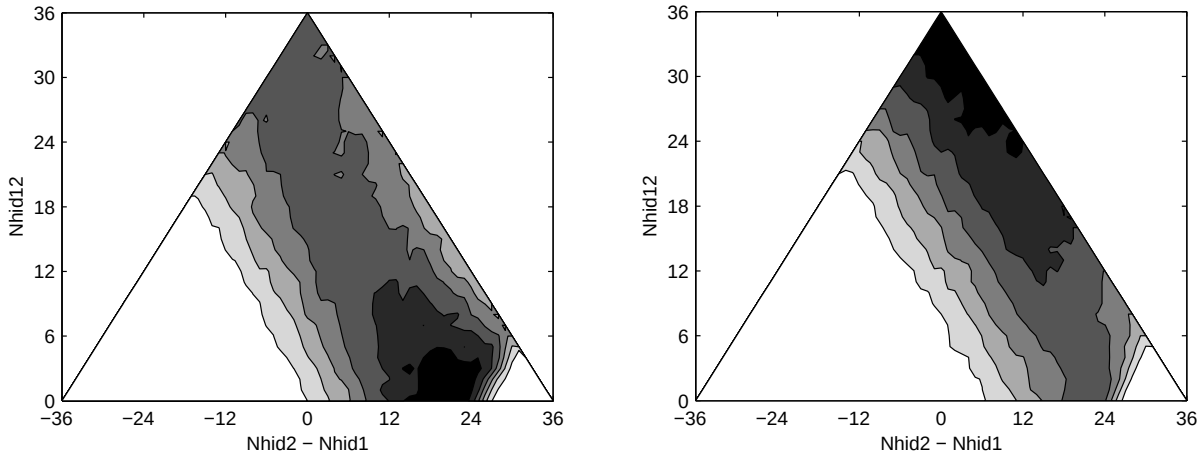

Figure 6: Mean learning times using control learning parameters as a function of architecture with Sum Squared Error (left) and Cross Entropy (right) cost functions. Darkest shading indicates fastest learning.

the Cross Entropy cost function, the optimal configurations (shown darkest) are situated in the *Nhid12 = Nhid*, i.e. total distributed, region. The mean best learning time is about 19 simulated years for the Sum Squared Error case, compared with about 10 years for the Cross Entropy case. Given these results are what the evolutionary simulations told us, they are not surprising.

The important thing we need to test is how robust these optimal configurations are with respect to using non-optimal learning rate and initial weight parameters. It is clearly possible to re-run the evolutionary simulations with the architecture fixed, to determine appropriate parameters for each configuration. The natural control conditions against which to compare the results of Figure 5 will be to have the architecture constrained to be fully distributed for the Sum Squared Error case, and constrained to be fully modular for the Cross Entropy case. In the non-modular case all *Nhid* = 36 hidden units would be used for both tasks, and in modular case we would just allow the proportion of hidden units dedicated to each task to vary. Using the parameters evolved in this way resulted in the mean learning time plots shown in Figure 6. We see that, even with the parameters optimized for the opposite kind of architecture, the optimal architectures are still in the same regions as before. The high performance regions become slightly more spread out and the mean best learning times are increased by about one simulated year in each case, but still we have a clear dependence of the optimal architecture on the learning cost function, and still we find the non-modular architecture far superior in terms of learning speeds.

8 Conclusions

By using simulated evolution by natural selection to optimize the architecture and learning parameters for simple neural network systems performing simplified versions of the “what” and “where” vision tasks, we

have seen that the appropriate choice between modular and non-modular architecture depends crucially on the cost function we choose for the neural network's learning algorithm. Moreover, contrary to earlier indications (Rueckl et al., 1989), it seems that a non-modular architecture with an appropriate learning cost function is far superior in terms of perfecting the given tasks as quickly and reliably as possible.

The human brain does, nevertheless, appear to have evolved to be modular with respect to the full scale versions of these tasks. However, drawing conclusions from our simulations about the modularity in human brains is not straightforward. If the results (i.e. modularity versus non-modularity) depend so crucially on such non-biologically plausible details as the learning algorithm, then it is clearly going to be rather difficult to extrapolate from them to biological systems. Moreover, it is not hard to imagine other crucial constraints on biological evolution that are not present in our current artificial systems (e.g. Jacobs & Jordan, 1992).

As far as understanding the human brain is concerned, the general simulation approach I have presented appears promising, but future simulations in this area will clearly have to be much more realistic if we are to draw reliable conclusions from them. When it comes to choosing appropriate architectures for artificial neural network (and quite likely, other computational intelligence) systems, the simulations presented here indicate that we have to be much more careful about choosing appropriate architectures than was previously realized.

References

Baldwin, J.M. (1896). A New Factor in Evolution. *The American Naturalist*, **30**, 441-451.

Belew, R.K. & Mitchell, M. (Eds) (1996). *Adaptive Individuals in Evolving Populations*. Reading, MA: Addison-Wesley.

Bullinaria, J.A. (1997). Analysing the Internal Representations of Trained Neural Networks. In A. Browne (Ed.), *Neural Network Analysis, Architectures and Applications*, 3-26. Bristol: IOP Publishing.

Bullinaria, J.A. (2001). Exploring the Baldwin Effect in Evolving Adaptable Control Systems. In: R.F. French & J.P. Sougné (Eds), *Connectionist Models of Learning, Development and Evolution*, 231-242. London: Springer.

Di Ferdinando, A., Calabretta, R, & Parisi, D. (2001). Evolving Modular Architectures for Neural Networks. In R.F. French & J.P. Sougné (Eds), *Connectionist Models of Learning, Development and Evolution*, 253-262. London: Springer.

Elman, J.L., Bates, E.A., Johnson, M.H., Karmiloff-Smith, A., Parisi, D. & Plunkett, K. (1996). *Rethinking Innateness: A Connectionist Perspective on Development*. Cambridge, MA: MIT Press.

Fodor, J.A. (1983). *The Modularity of the Mind*. Cambridge, MA: MIT Press.

Goodale, M.A. & Milner, A.D. (1992). Separate Visual Pathways for Perception and Action. *Trends in Neurosciences*, **15**, 20-25.

Hinton, G.E. (1989). Connectionist Learning Procedures. *Artificial Intelligence*, **40**, 185-234.

Hinton, G.E. & Nowlan, S.J. (1987). How Learning Can Guide Evolution. *Complex Systems*, **1**, 495-502.

Jacobs, R.A. & Jordan, M.I. (1992). Computational Consequences of a Bias Toward Short Connections. *Journal of Cognitive Neuroscience*, **4**, 323-336.

Jacobs, R.A., Jordan, M.I. & Barto, A.G. (1991). Task Decomposition Through Competition in Modular Connectionist Architecture: The What and Where Vision Tasks. *Cognitive Science*, **15**, 219-250.

Mishkin, M., Ungerleider, L.G. & Macko, K.A. (1983). Object Vision and Spatial Vision: Two Cortical Pathways. *Trends in Neurosciences*, **6**, 414-417.

Plaut, D.C. (1995). Double Dissociation Without Modularity: Evidence from Connectionist Neuropsychology. *Journal of Clinical and Experimental Neuropsychology*, **17**, 291-321.

Plaut, D.C. & Hinton, G.E. (1987). Learning Sets of Filters Using Back-Propagation. *Computer Speech and Language*, **2**, 35-61.

Rueckl, J.G., Cave, K.R. & Kosslyn, S.M. (1989). Why are "What" and "Where" Processed by Separate Cortical Visual Systems? A Computational Investigation. *Journal of Cognitive Neuroscience*, **1**, 171-186.

Shallice, T. (1988). *From Neuropsychology to Mental Structure*. Cambridge: Cambridge University Press.

Ungerleider, L.G. & Haxby, J.V. (1994). 'What' and 'Where' in the Human Brain. *Current Opinion in Neurobiology*, **4**, 157-165.