Simulating the Evolution of Modular Neural Systems

John A. Bullinaria (j.bullinaria@physics.org)
School of Computer Science, The University of Birmingham
Edgbaston, Birmingham, B15 2TT, UK

Abstract

The human brain is undoubtedly modular, and there are numerous reasons why it might have evolved to be that way. Rueckl, Cave & Kosslyn (1989) have shown how a clear advantage in having a modular architecture can exist in neural network models of a simplified version of the "what" and "where" vision tasks. In this paper I present a series of simulations of the evolution of such neural systems that show how the advantage *can* cause modularity to evolve. However, a careful analysis indicates that drawing reliable conclusions from such an approach is far from straightforward.

Introduction

Intuitively, 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 brains do operate in such a modular manner (e.g. Shallice, 1988). In particular, the inference from double dissociation to modularity is one of the corner stones of cognitive neuropsychology, and over recent years double dissociation between many tasks have been established, with the implication of associated modularity.

Some early neural network models seemed to indicate that fully distributed systems could also result in double dissociation (e.g. Wood, 1978) and hence cast some doubt on the inference of modularity. Since then, the potential for double dissociation in connectionist systems with and without modularity has been well studied (e.g. Plaut, 1995; Bullinaria & Chater, 1995; Bullinaria, 1999), and the early connectionist double dissociations have been seen to be merely the result of small scale artefacts. Several later studies (e.g. Devlin, Gonnerman, Andersen & Seidenberg, 1998; Bullinaria, 1999) have shown how weak double dissociation can arise as a result of resource artifacts (e.g. Shallice, 1988, p232) in fully distributed systems, but it seems that strong double dissociation does require some form of modularity, though not necessarily in the strong (hard-wired, innate and informationally encapsulated) sense of Fodor (1983). Plaut (1995), for example, has shown that double dissociation can result from damage to different parts of a single neural network, and Shallice (1988, p249) lists a number of systems that could result in double dissociation without modularity in the conventional sense. In this paper, I am not so much interested in showing how double dissociation can arise in connectionist systems without modularity, but rather, how modularity can arise in connectionist systems and hence have the potential for exhibiting double dissociation.

Of particular interest to us here is the discovery that visual perception involves two distinct cortical pathways (Mishkin, Ungerleider & Macko, 1983) – 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 is that any process of evolution by natural selection would result in a modular architecture and hence answer the question of why modularity has arisen.

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

Learning Multiple Tasks

Nowadays, the basic structure of simple feed-forward neural network models is well known. We typically use 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 set of training data.

In this context, the question of modularity relates to the connectivity between the network's hidden and

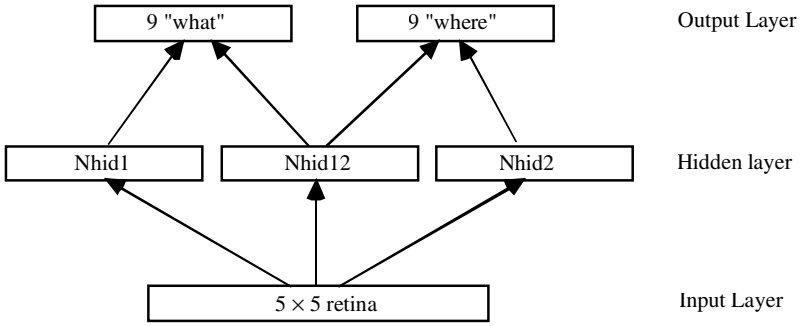

Figure 1: Architecture of the basic neural network model for the "what" and "where" tasks.

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 (Plaut & Hinton, 1987). However, such an extreme version of modularity with a 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.

It is well known that, when one trains a neural network using standard gradient descent type learning algorithms, 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 somewhat 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.

In this paper, I am more interested in the *evolution* of modularity than the *learning* of modularity. The old Nature-Nurture debate has come a long way in recent years (e.g. Elman et al., 1996), but it is still important to understand which characteristics are innate and which need to be learnt during ones lifetime. Moreover, as computer technology becomes more powerful, we are able to explore these issues by increasingly realistic simulations. 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, we have been able to observe the genetic assimilation of learnt characteristics without Lamarckian inheritance, see how appropriate innate values for network parameters and learning rates can evolve, understand how individual differences across evolved populations are constrained, and so on (e.g. Bullinaria, 2001). In the remainder of this paper I shall consider the evolution of modularity in neural network models of the "what" and "where" tasks previously studied by Rueckl et al. (1989). The lessons we learn here will be applicable to the learning and evolution of modularity more generally.

The "What" and "Where" Model

To avoid the need to repeat the extensive analyses of the learnt internal representations carried out by Rueckl et al. (1989), I shall study exactly the same simplified neural network model that they used, and explore whether the advantages of modularity they observed are sufficient to drive the evolution of modularity. I shall also follow Rueckl et al. (1989) and Jacobs et al. (1991) in emphasizing that the tasks we are simulating 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.

The 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 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 shows the basic network that was originally investigated by Rueckl et al. (1989). We have 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 will allow us to explore the full continuum between these extremes. If the maximum number of hidden units *Nhid* = *Nhid1* + *Nhid12* + *Nhid2* is fixed, then we need define only two innate architecture parameters *Con1* = *Nhid1* + *Nhid12* and *Con2* = *Nhid2* + *Nhid12* corresponding to the number of hidden units connecting to each output block.

Simulating Evolution

To simulate an evolutionary process for the models discussed above, we take a whole population of individual instantiations of each model and allow 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.

In more realistic situations, the ability of an individual to survive or reproduce will rely on a number of factors which can depend in a complicated manner on that individual's performance over a range of related 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.

I shall follow a more natural approach to procreation, mutation and survival than many evolutionary simulations have done 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.

The simulated genotypes naturally include all the innate parameters needed to specify the network details, namely the architecture, the learning algorithm, the learning rates, the initial 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.

It is clearly important to fix the evolutionary parameters appropriately according to the details of the problem and the speed and coarseness of the simulations. For example, if all individuals learn the task perfectly by the end of their first year, and we only test 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 training data presentations per year to 10 for each individual. Choosing 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 5 deaths per year due to competition, and another 5 individuals over the age of 30 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.

Experiment 1 – The Basic Model

I began by simulating the evolution of the system as stated 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 on the Sum Squared Error cost function *E* (Hinton, 1989). As before, the target outputs were taken

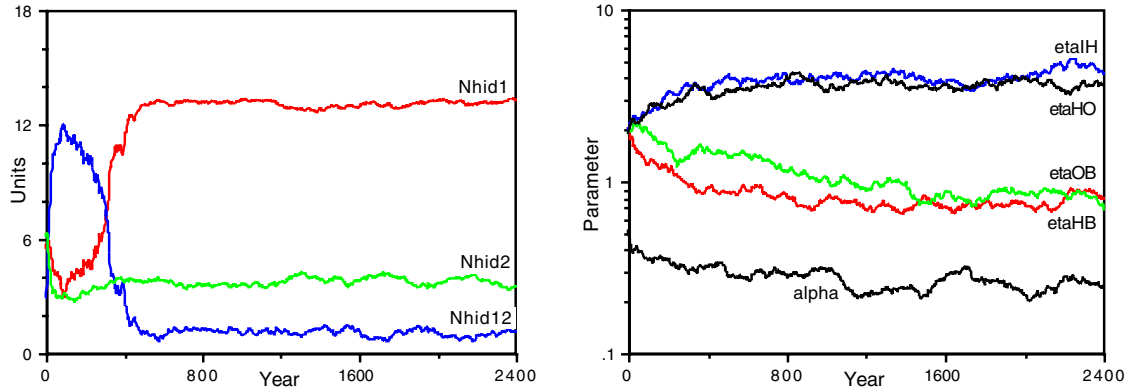

Figure 2: Evolution of the model in Figure 1 with Sum-Squared Error cost function and Log Cost fitness function.

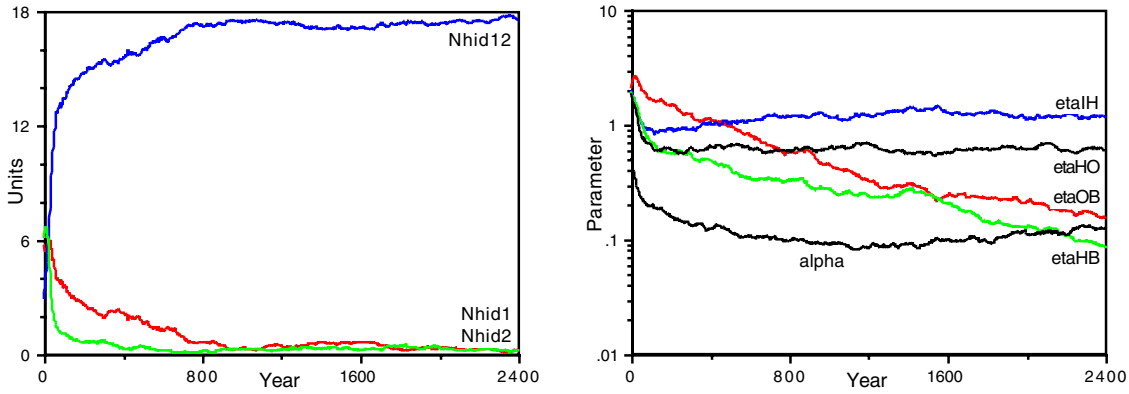

Figure 3: Evolution of the model in Figure 1 with Cross Entropy cost function and Log Error Count fitness.

to be 0.1 and 0.9, rather than 0 and 1, and appropriate outputs beyond these targets were deemed errorless. 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

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

Each genotype thus contained two parameters to control the network architecture, and five to control its learning rates. The Sum Squared Error cost distribution turns out to be rather skewed across the population, so the individual evolutionary fitnesses were defined to be –log(Cost).

I have found in my earlier studies (Bullinaria, 2001) that the evolution can depend 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. 2.0] and the momentum parameters randomly from the range [0.0, 1.0]. Following Rueckl et al. (1989), the initial weights were chosen randomly within the range [0.0,0.3].

Figure 2 shows how the innate parameters evolved when there were 18 hidden units in total (which is how many Rueckl et al., 1989, used). We see that the learning parameters soon settle down and, after a 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 optimal values for *Nhid1* and *Nhid2*.

Experiment 2 – Different Costs

The results of Experiment 1 make the evolution of modularity look almost inevitable. However, it would be misleading not to report on the countless simulations in which modularity did not evolve, and which could equally well correspond to human evolution, with the implication that modularity in the human brain must originate in some other manner. Figure 3 shows what can happen with one particularly reasonable alternative choice for the gradient descent cost function and

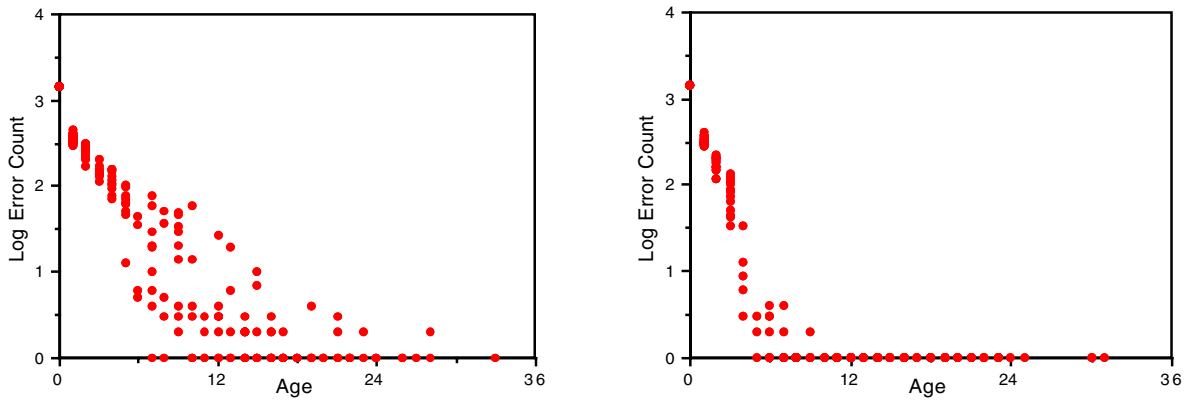

Figure 4: Comparison of evolved populations with Sum Squared Error (left) and Cross Entropy (right) cost functions.

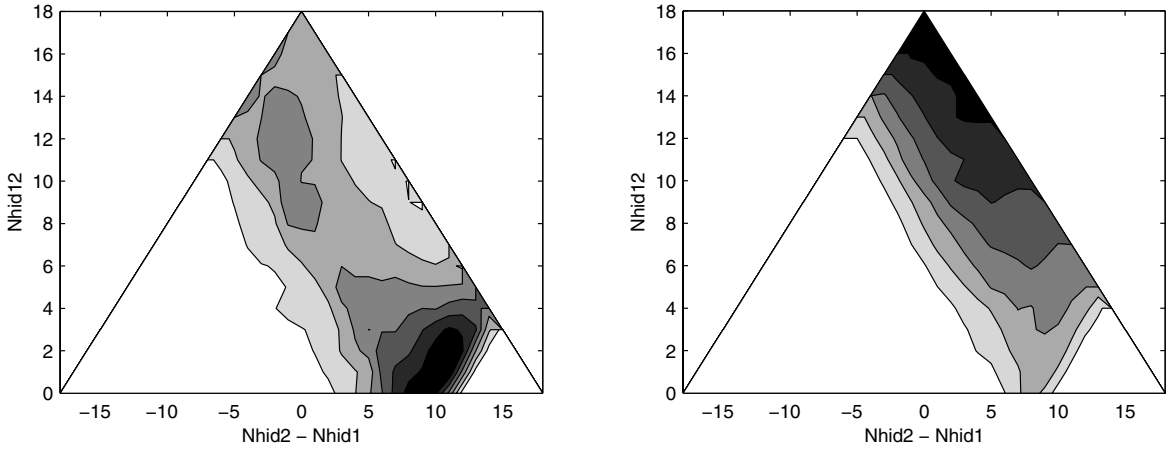

Figure 5: Mean learning times with Sum Squared Error (left) and Cross Entropy (right) cost functions.

evolutionary fitness function, namely the standard Cross-Entropy cost function (Hinton, 1989), and fitness defined by counting the total number of output units with errors above some small fixed value (0.2 say). This results in the evolution of a completely non-modular architecture. A systematic study reveals that changing the fitness between –Cost, –log(Cost), 1/Cost, –ErrorCount, and –log(1+ErrorCount) and has little effect on the results. However, the choice of cost function is crucial. Figure 4 compares the learning in the evolved populations for the Sum Squared Error and Cross Entropy cost functions with –log(1+ErrorCount) fitness. The non-modular Cross-Entropy population shows a clear superiority.

Although we should not rely on the mean learning rates to predict what will evolve (since the standard deviations, the worst and best cases, and so on, are also important), the plots in Figure 5 of the mean learning times as a function of the architecture do show quite clearly where the different optimal configurations (shown darkest) are situated.

Experiment 3 – Larger Networks

A final worry was that our simulations were suffering from small scale artefacts. Often when a network has barely enough hidden units to solve the task at hand, it behaves differently to when it has plenty of spare resources (e.g. Bullinaria & Chater, 1995; Bullinaria, 1997). Since 18 hidden units is near minimal for our task, all of the above simulations were repeated with 36 hidden units. This had little effect on the Cross Entropy simulations, but the results were rather variable with Sum Squared Error costs. Sometimes modularity evolved, sometimes it didn't, and often mixed populations arose. Apparently minor variations in the implementational details, or even just different random number seeds, could change the results completely.

Figure 6 shows the mean learning times here for comparison with those for the smaller networks in Figure 5. We see the Cross-Entropy plot has the same non-modular optimum as before, but the Sum-Squared Error case is now much noisier, with further, roughly equivalent, minima appearing in the non-modular regime. This is presumably why the evolutionary simulation results were so variable.

Conclusions

I have shown how it is possible to simulate the evolution of modularity in simple neural network models.

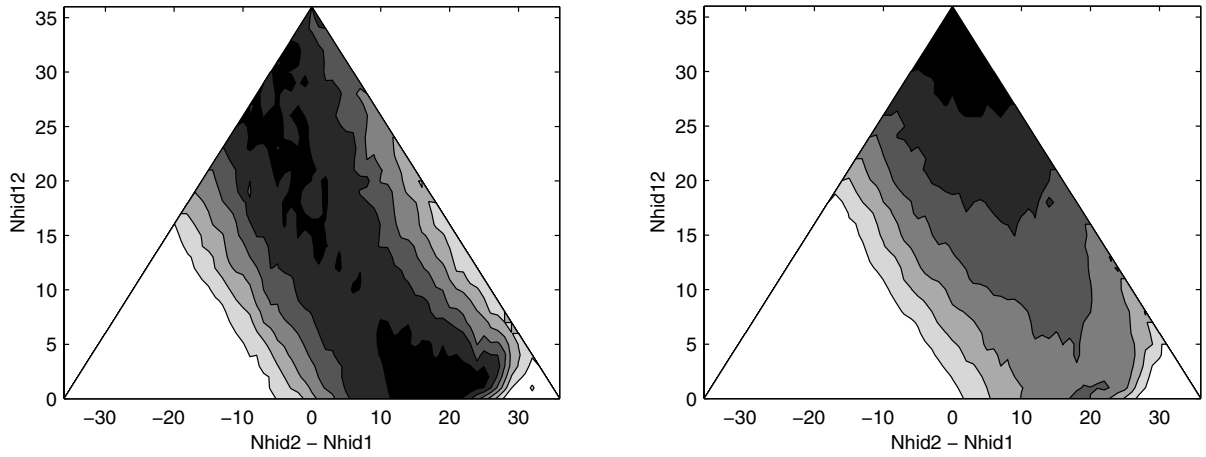

Figure 6: Large network learning times with Sum Squared Error (left) and Cross Entropy (right) cost functions.

However, drawing conclusions from them about the modularity in human brains is not so 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. On one hand, we might expect that the human brain has evolved particularly efficient learning algorithms, in which case we could argue that the more efficient non-modular cross-entropy populations are the more realistic. On the other hand, real tasks are considerably harder than those used in our simulations, and so the modular populations might be deemed a more reliable representation of the actual relation between the human learning algorithm power and task complexity. 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.

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. (1999). Connectionist Dissociations, Confounding Factors and Modularity. In D. Heinke, G.W. Humphreys & A. Olsen (Eds), *Connectionist Models in Cognitive Neuroscience,* 52-63. Springer.

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

Bullinaria, J.A. & Chater N. (1995). Connectionist Modelling: Implications for Cognitive Neuropsychology. *Language and Cognitive Processes*, **10**, 227-264.

Devlin, J.T., Gonnerman, L.M., Andersen, E.S. & Seidenberg, M.S. (1998). Category-Specific Semantic Deficits in Focal and Widespread Brain Damage: A Computational Account. *Journal of Cognitive Neuroscience*, **10**, 77-94.

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

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.

Wood, C.C. (1978). Variations on a Theme of Lashley: Lesion Experiments on the Neural Model of Anderson, Silverstein, Ritz & Jones. *Psychological Review,* **85,** 582-591.