In: A. Cangelosi, G. Bugmann & R. Borisyuk (Eds), *Modeling Language, Cognition and Action*, pp. 375-384. Singapore: World Scientific. 2005.

EMBRYOLOGICAL MODELLING OF THE EVOLUTION OF NEURAL ARCHITECTURE

CHRIS P. BOWERS & JOHN A. BULLINARIA
School of Computer Science, The University of Birmingham
Birmingham, B15 2TT, UK
{c.p.bowers, j.a.bullinaria}@cs.bham.ac.uk

Attempts in the past to use evolutionary simulations to model the emergence of modular neural architectures has led to conflicting results. Here, we present some preliminary work on the use of computational embryogeny to model the evolution of neural architecture at a less coarse level of description. We believe that such an approach will lead to much more reliable and biologically realistic simulations of brain evolution.

1. Introduction

The architecture of the human brain, in particular the existence of functionally specialised neural modules, is becoming increasingly well mapped out. However, some of the early results from neural computational modelling in this area (Rueckl, Cave & Kosslyn, 1989) have recently been thrown into doubt by more detailed evolutionary simulations (Bullinaria, 2001). There remain a range of feasible computational approaches to understanding neural architecture (Jacobs, 1999), and we believe the best way forward is to look at the evolution of these structures at a much lower level of description than is often used.

Embryogeny is the process by which an embryo structure develops from an initial stem state. This genotype to phenotype mapping process is adaptive in that the rules of growth are dependent upon the state of the system and vice-versa, thus forming a dynamical system. The modelling of such processes in a simulated evolutionary framework is known as computational embryogeny.

In this paper we shall present work in which such an approach is utilised to grow neural architectures from an initial stem state, based upon a simple model encapsulating individual cell states and the diffusion of chemicals in a three-dimensional space. This formulation is sufficiently expressive to account for features such as topology and connectivity, as well as the initial weight and learning parameters. A systematic series of computational experiments is carried out to explore the possibility of using this approach to study the emergence of neural structures appropriate for the simplified 'what-where' task used previously (Rueckl, et al., 1989; Bullinaria, 2001). The resulting structures are then compared against a series of more abstract directly coded models of the type studied previously by Bullinaria (2001), but with a matched evolutionary

regime. In this way we can gain a better understanding of the reliability of, and the relationships between, models at different levels of description.

2. Modelling the Emergence of Modularity

We ground our work in the simplified 'what' and 'where' vision tasks, for which extensive investigations into the internal representations learned by simple neural networks have already been carried out (Rueckl, et al., 1989), as have a series of evolutionary simulations (Bullinaria, 2001). These tasks consist of mapping from a 5x5 binary retinal image, to a set of 9 'what' and 9 'where' binary classification outputs. The problem is to identify 9 different 3x3 sub-retinal images that are positioned in the 9 different possible locations upon the 5x5 retina, which yield a total set of 81 possible input patterns.

As in the previous work, the basic neural network structure takes the form of a 3-layer network consisting of 25 input units, 18 output units and various hidden units (Figure 1). The hidden units are all fully connected to each of the inputs, but their connectivity to outputs can be described as belonging to one of three different groups: connected to only the 'what' outputs (*Hid1*), connected to only the 'where' outputs (*Hid2*), or connected to both (*Hid12*). In this respect, a network where all the hidden units belong to the group *Hid12* would be considered totally non-modular, whereas a network where there are no hidden units in *Hid12* would be considered to have totally modular connectivity.

The analysis of Reuckl et al. (1989) appeared to show a clear advantage for modular architectures. However, taking a population of networks, and using simulated evolution by natural selection to determine all the learning and architecture parameters, the emergent architecture was found to depend on the learning algorithm cost function used (Bullinaria, 2001). If the most efficient (cross entropy based) learning algorithm is used, or if the learning algorithm is allowed to evolve, we always end up with a non-modular architecture. These conflicting results have led us to look for more biologically realistic approaches for understanding the emergence of modularity in neural systems.

3. The Embryological Model

Embryogeny is the process by which an embryo develops from a single stem cell. Computational embryogeny refers to the simulation, within computational systems, of the kind of embryonic processes observed in nature. The embryogeny mapping requires that the phenotype consist of discrete processing blocks (cells). In our model the cell environment will be represented as a three-dimensional space that can accommodate various levels of proteins and cell states at any given real valued position.

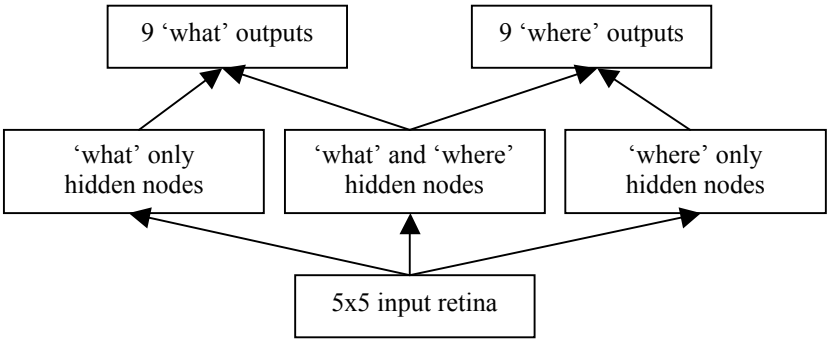

Figure 1: The architecture of the basic neural network model for the 'what' and 'where' task.

In nature, the functionality of genes is determined by the state of a cell and the surrounding environment, i.e. by proteins and other chemicals. The function of a gene may, in turn, be to alter the cell state upon which these genes are dependent. In order to capture the essence of this process in a computer simulation, a number of simplifications over real biological systems must be made to render the computational costs feasible. All cell sizes and shapes are taken to be identical, and are represented as spheres with diameters in proportion to the dimensions of the overall space. Direct cell-cell interaction beyond simple chemical diffusion is prohibited, so there is no cell-adhesion or any other form of force between cells. If a cell is to divide, or move into the location of an existing cell, it simply overwrites the existing cell. This is clearly not biologically plausible, but it is necessary to reduce the computational costs of a more realistic physical model.

In nature, any limits on the size of the physical system are set by physical interaction. Since, in our model, no such forces exist, the environment must be set at some finite size to impose some constraint on the size of the emergent system. This poses the problem of what to do when a cell attempts to move or divide outside of this finite environment. A common approach is to cycle around the edges so that, for example, in two-dimensional space a toroidal shape is formed, removing the need to worry about finite limitations. However, this can often introduce unwanted cyclic behaviour, so a different approach is used in which any cells dividing outside of the modelled space are simply ignored.

The genetic representation of the embryogeny system is based upon a simplified model of the genetic regulatory network (GRN), using the Operon model as a basis for individual gene function (Alberts, et al., 1994). Biologists use this model to describe how groups of individual genes can form complex GRNs. In our work, a much-simplified version of the Operon model is used in which a gene has two dependents, an inducer/repressor and a product. If the

inducer/repressor exists, the gene is switched on/off respectively. If the gene is switched on, it releases a product pre-defined by the gene. This product may result in the increase or decrease of the concentration of some protein, or it may lead to some function upon the cell. For example, a gene that promotes axonal growth may cause an axon head to grow in the direction of the protein gradient as defined by the product, as long as no repressor protein is present. The GRN comes into play since the function of a gene may be to produce an inducer or repressor itself, and in this way a gene can control the expression of other genes and so form a network of interaction between genes. In our model, there are 34 gene functions consisting of concepts such as cell division and movement, protein emission, protein absorption, axon growth, etc. These allow the cells to adapt and change their environment and cell states.

The mapping process starts with an initial stem state, consisting of pre-determined input and output neurons and a single stem cell. The growth process is performed in fixed steps, in parallel, across the space, such that the proteins and genome interact to produce some cell functionality. This interaction is the critical part of the model which defines what is expressible in the phenotype.

4. Computational Embryogeny

Computational embryogeny creates a complex mapping between genotype and phenotype spaces that exhibits some key characteristics. First, the mapping is many-to-many in that a single genotype can form a variety of different phenotypes dependent upon the environmental conditions. Similarly, a number of different genotypes can result in identical phenotypes. Second, the neighbourhood structure is not necessarily preserved by the mapping, in that two similar genotypes can result in very different phenotypes, or vice-versa.

From these characteristics, it is easy to assume that such a mapping would be disadvantageous for a simulated evolutionary process, since it would introduce a hugely rugged landscape. In some respects, this is true. However, this is based on the viewpoint that the fitness landscape is a fixed entity through which a population of individuals drift and jump around based on some heuristics, or search operators, biased toward fitter individuals. These search operators are, however, more powerful than this viewpoint suggests, since it is not only the genetic representation that determines the structure of the fitness landscape, but also the search operators themselves. If the effects of the search operators can be adapted in some way during the evolutionary process, then this would effectively cause a restructuring of the fitness landscape. Thus the analogy of traversing around landscapes begins to break down.

A further complication is the huge number of possible individuals with equivalent fitness. The result is that the search space has vast networks of

neutrality, where there are threads, planes, or even higher dimensional structures of equal fitness. Traditionally, this has been viewed as a bad thing to incorporate, since it results in convergence of the population or dead time in terms of evolutionary change. However, such neutral networks have been shown to be extremely useful with the right kind of algorithm, and evidence suggests they are also critical to natural evolution (Harvey, 1997). Since there can be no selection pressure in these neutral areas, neutral evolution has a different behaviour to that of adaptive evolution, which relies on selection pressure to drive evolution to fitter individuals. These neutral networks can provide a way to freely traverse the search space, without selection pressure driving toward some fixed convergence point. The result is that individuals converge onto the most optimal neutral network, and then diverge across that network. When an individual finds a new neutral network within its reach, it jumps across, and the rest of the population converges on this new network. This results in staggered jumps in fitness through evolutionary time, and gradual increases in diversity followed by a sudden drop as the population jumps to the new network (Smith, Husbands & O'Shea, 2001).

An important point to note here is that, in the absence of selection pressure, evolution does not simply stop. Since individuals are selected randomly, search operators are still used to produce new offspring during neutral evolution. The evolutionary drive is no longer towards fitter individuals, but individuals that can maintain or improve their fitness through mutations and crossover. In other words, genetic material that is more susceptible to damage will be less likely to propagate to the next generation, regardless of its contribution to fitness. It could be said that adaptive evolution applies evolutionary drive to fitter individuals, whilst neutral evolution drives towards more evolvable individuals.

To take advantage of these elements, our evolutionary algorithm consists of the following processes:

- The mutation operator consists of point mutations applied to each allele of the genome with a given mutation probability, resulting in random replacement with a value from within the range of validity. Crossover involves the selection of two random crossover points which may only fall in between genes. This helps to prevent destruction of genes and ensures the conservation of genome length.
- Recombination is based on a steady state approach, consisting of the replacement of the last individual in a linearly ranked population only if its fitness is equal or less than the individual that replaces it.
- Fitness is defined as the grown network's performance on the training data after gradient descent learning using our chosen cost function.

It has been shown that for a complex mapping, optimal neutral evolution will occur when the population is phenotypically converged. In other words, when the entire population is drifting on the same neutral network there is more chance of finding a bridge to an individual/network of higher fitness (Harvey, 1997). However, in order for adaptive evolution to be successful, it must maintain diversity amongst the population to perform effective search across the search landscape. Obviously, these are two conflicting ideals, and so the solution has been to use an algorithm based on an island population model (Gordon, Whitley & Bohn, 1992). This consists of a number of separate sub-populations being allowed to evolve and swap individuals. Populations swap individuals under the following conditions:

- An individual is chosen as a migrant to the receiving population if it is the best individual from another population having the greatest genetic distance from the best individual in the receiving population.
- The chosen migrant replaces the individual in the receiving population with the closest genetic distance.

Genetic distance is measured as the number of differences between two individuals. This allows a group of individuals to converge on a neutral network, whilst ensuring different groups are effectively forced to diverge from each other. The result is a system that can swiftly adapt to suit either neutral or adaptive evolution, or even a combination of both. An added advantage of the island model, and one of the more common reasons for its use, is that it is easy to distribute across numerous machines and so speed up the evaluation time of the overall population. This is a significant advantage in this case where fitness evaluation can be particularly computationally expensive.

5. Experimental Details

Our aim for this paper is to extend the previous work on evolving modularity (Bullinaria, 2001) by allowing the evolutionary process greater flexibility in the architectures and learning parameters it can express. In particular, we wish to gain a better understanding of the relationship between modularity and learning for the simplified 'what-where' vision task discussed above.

Although the embryogeny model is capable of expressing any of the parameters required for describing and evaluating the performance of a neural network, in our study it does not specify values for all the network weights and biases, only the ranges for their random initial values. As with most biological networks, the actual values are learned from appropriate training data. In our experiments, we use an online gradient descent algorithm, based on either the

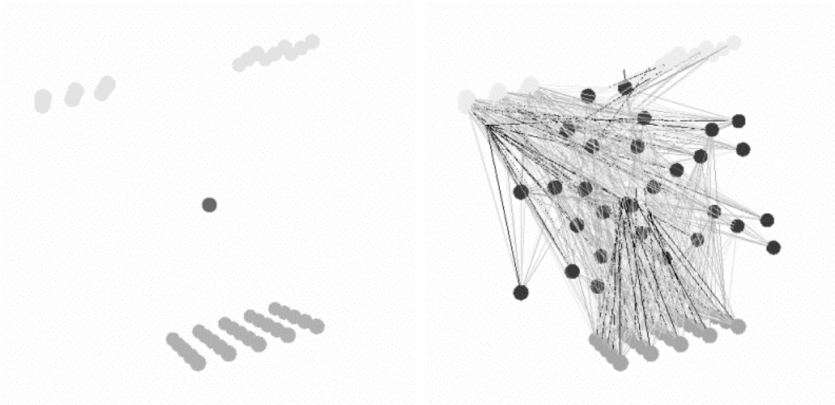

Figure 2: (a) The initial stem state of the model. (b) A typical fully grown evolved neural network.

Sum Squared Error (SSE) or the Cross Entropy (CE) cost function.

The types of architectures that can be found by our evolutionary system are extremely flexible, encapsulating all forms of connectivity, including lateral and recurrent connections. The learning algorithm must account for this possibility, and so a windowed propagation approach is used whereby neural activations can only propagate through a single neuron per iteration. This means that several iterations of the training algorithm are required before activation at the input neurons finally reaches the output neurons, but recurrent looping and output settling problems are eliminated.

The embryogeny process requires a stem state from which to start. This stem state consists of a set of predefined input and output neurons on opposing faces of the three-dimensional environment with a single stem cell positioned at the centre (Figure 2a). The input neuron positioning is consistent with the problem description forming a 5x5 array, whilst the two groups of output neurons are arranged in 3x3 arrays to minimise the connectivity radius required to connect to all 9 outputs. For simplicity, we took the dendrite lengths and overall environment size to be measured in units of cell widths. This removes any need to assign definitive sizes, rather than just proportionality ratios between different aspects of the model. The network growing process is allowed a fixed number of growth steps sufficient to allow any cell to re-position itself to any location in the environment.

Our modeling approach is also flexible enough to accommodate the idea, and consequences for modularity, of constraints on neural connectivity, including those discussed by Jacobs and Jordan (1992). By constraining the distance from which an axon head can create a synaptic connection to a neuron, it is possible to effectively constrain the number of neurons a particular neuron can connect to. This is represented in Figure 3.

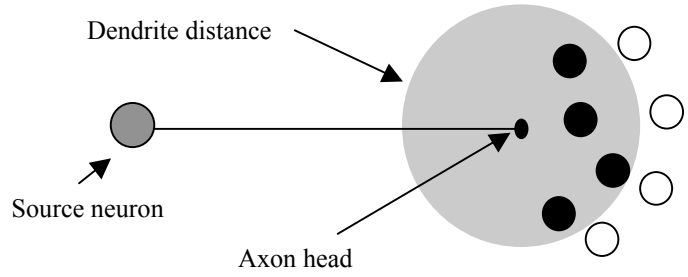

Figure 3: The effect of dendrite distance on connectivity. Only cells within the dendritic radius create synaptic connections with the source neuron.

6. Simulation Results

We now present our results from a series of simulated evolutionary processes obtained for various dendrite distances, and for both the SSE and CE learning algorithm cost functions. Figure 4a shows the hidden to output connectivity plot for CE, where *pHid1*, *pHid2*, and *pHid12* specify the proportion of hidden units connected to each output block. Each point in the plot represents an optimal network found by an evolutionary process. For larger dendrite distances, the results agree with those of the coarser evolutionary simulations discussed above (Bullinaria, 2001), with networks tending to be located in the *pHid12=1* region, indicating that non-modular fully distributed networks are preferred.

However, if we reduce the dendrite distance, and so constrain the connectivity, the results show a distinct difference, with far fewer neurons connected to both output blocks. Two points can be made from these results. First, for larger dendrite distances, the results show non-modular networks emerging, and so, even with greater expressivity in terms of network structure and learning parameters, non-modular networks are still found to be optimal. Second, it is possible to encourage the emergence of less distributed and more modular architectures, simply by decreasing the allowable dendrite distances.

Figure 4b shows the average performance rates throughout the evolutionary process. We see that for shorter dendrite distances it takes more evolutionary time to find an optimal solution. Thus, although we can force less distributed and more modular architectures on networks trained with CE, the evolutionary progress is worse than for networks that use a fully distributed approach.

If we look at the connectivity plot results for the corresponding SSE simulations (Figure 5a), we see that for longer dendrite distances the network connectivity behaves much like those for CE. However, finding optimal networks for smaller dendrite distances becomes extremely hard, and this is reflected by the lack of plotted points representing optimal networks with

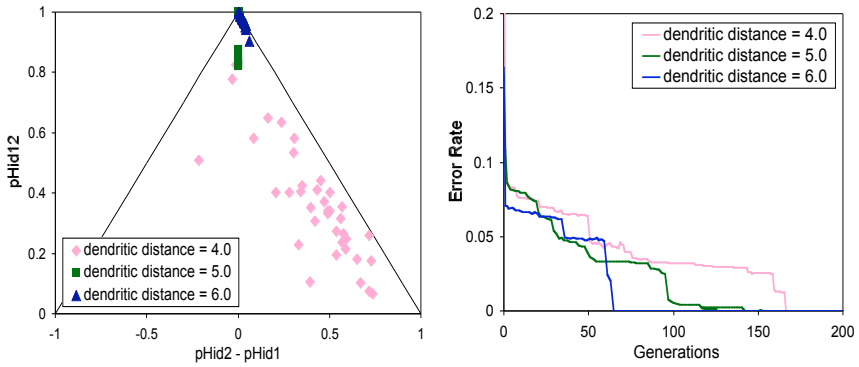

Figure 4: (a) Connectivity plot of optimal networks and (b) average evolutionary learning curves for the CE based learning algorithm.

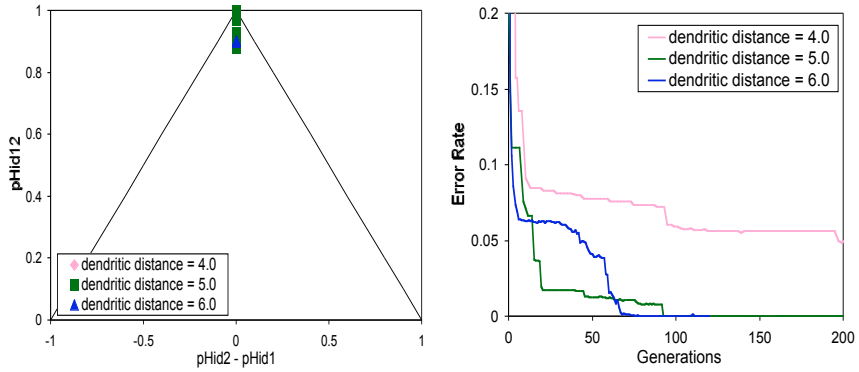

Figure 5: (a) Connectivity plot of optimal networks and (b) average evolutionary learning curves for the SSE based learning algorithm.

dendrite distance of 4.0. This is confirmed in Figure 5b where we can see that the learning behaviour during evolutionary time is similar to CE except that, for smaller dendrite distances, optimal networks are never actually found and evolution gets stuck on poorly performing networks.

Using more directly encoded representations, the CE cost function always produced non-modular optimal networks, whilst SSE tended to produce modular optimal networks, as Bullinaria (2001) found. When comparing this with our embryological modelling results for CE, it is clear that the behaviour is similar until the networks connectivity is constrained by short dendrite distances. Comparison of the results for SSE show the greatest differences, since our work suggests that, under a more expressive representation, non-modular networks are optimal in this case too. This appears to be at odds with all the previous work (Rueckl et al., 1989; Bullinaria, 2001). Moreover, constraining the connectivity

by restricting the dendrite distances has a serious effect for SSE networks, in that it results in a failure of the evolutionary algorithm to find networks of any description that can learn to perform the given tasks.

7. Conclusion

We began by reviewing the Rueckl et al. (1989) and Bullinaria (2001) studies into the emergence of modularity in neural systems, and argued that performing more detailed simulations was a sensible approach to gaining a better understanding of the problem. After an outline of a computational embryogeny approach to evolving networks that grow and learn to perform two distinct tasks (the 'what' and 'where' tasks studied previously), we found that the emerging architectures always take on a distinctly non-modular form, unless restrictions are imposed on the dendritic distances. This ties in with the work of Jacobs and Jordan (1992) on the consequences of a natural bias to short neural connections. We clearly have much more work to do to explore fully all the relevant issues, but we believe we have a promising approach for studying the evolution of neural architectures.

References

Alberts, A., Bray, D., Lewis, J., Raff, M., Roberts, K. & Watson, J.D. *Molecular Biology of the Cell.* 3rd edition, Garland. 1994.

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

Gordon, V., Whitley, D. & Bohn, A. Dataflow Parallelism in Genetic Algorithms. In Manner, R. & Manderick, B. (Eds), *Parallel Problem Solving from Nature 2*, 553-542. Amsterdam: Elsevier. 1992.

Harvey, I. Artificial Evolution for Real Problems. In *Evolutionary Robotics: From Intelligent Robots to Artificial Life*. 187-220. AAI. 1997.

Jacobs, RA. Computational Studies of the Development of Functionally Specialized Neural Modules. *Trends in Cognitive Science*, **3**, 31-38. 1999.

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

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

Smith, T. Husbands, P. & O'Shea, M. Neutral Networks and Evolvability with Complex Genotype-Phenotype Mapping. *European Conference on Artificial Life*. 272-281. Springer. 2001.