PERGAMON

Progress in Neurobiology 61 (2000) 557–579

progress in Neurobiology

www.elsevier.com/locate/pneurobio

On the design of neural networks in the brain by genetic evolution

Edmund T. Rolls*, Simon M. Stringer

Department of Experimental Psychology, Oxford University, South Parks Road, Oxford OX1 3UD, UK

Received 1 October 1999

Abstract

Hypotheses are presented of what could be specified by genes to enable the different functional architectures of the neural networks found in the brain to be built during ontogenesis. It is suggested that for each class of neuron (e.g., hippocampal CA3 pyramidal cells) a small number of genes specify the generic properties of that neuron class (e.g., the number of neurons in the class, and the firing threshold), while a larger number of genes specify the properties of the synapses onto that class of neuron from each of the other classes that makes synapses with it. These properties include not only which other neuron classes the synapses come from, but whether they are excitatory or inhibitory, the nature of the learning rule implemented at the synapse, and the initial strength of such synapses. In a demonstration of the feasibility of the hypotheses to specify the architecture of different types of neuronal network, a genetic algorithm is used to allow the evolution of genotypes which are capable of specifying neural networks that can learn to solve particular computational tasks, including pattern association, autoassociation, and competitive learning. This overall approach allows such hypotheses to be further tested, improved, and extended with the help of neuronal network simulations with genetically specified architectures in order to develop further our understanding of how the architecture and operation of different parts of brains are specified by genes, and how different parts of our brains have evolved to perform particular functions.

Keywords: Genetic algorithm; Natural selection; Cerebral cortex; Pattern association; Autoassociation; Competitive networks

Contents

* Corresponding author. Tel.: +44-1865-271348; fax: +44-1865-310447.
E-mail address: edmund.rolls@psy.ox.ac.uk (E.T. Rolls).

0301-0082/00/$ - see front matter
PII: S0301-0082(99)00066-0

1. Introduction

Analysis of the structure and function of different brain areas is starting to reveal how the operation of networks of neurons may implement the functions of each brain area. In particular, the operation of a number of simple types of network, which are each found in a number of different brain areas, are becoming understood (Rolls and Treves, 1998). One example of these networks is pattern association memories in which one input (e.g., an unconditioned stimulus) drives the output neurons through unmodifiable synapses, and a second input (e.g., a conditioned stimulus) has associatively modifiable synapses onto the output neurons, so that by associative learning it can come to produce the same output as the unconditioned stimulus (see Fig. 1). Pattern association memory may be implemented in structures such as the amygdala and orbitofrontal cortex to implement stimulus-reinforcement learning, e.g. associating the sight of a stimulus with pain (an example of the learning involved in emotion), or the sight of another stimulus with the taste of food (an example of the role of this type of learning in motivational behavior) (Rolls and Treves, 1998; Rolls, 1999). Pattern association learning may also be used in the connections of backprojecting neurons in the cerebral cortex onto the apical dendrites of neurons in the preceding cortical area as these connections may be associatively modifiable (Rolls and Treves, 1998). A second example is autoassociation networks characterized by recurrent collateral axons with associatively modifiable synapses which may implement functions such as short term memory in the cerebral cortex, and episodic memory in the hippocampus (see e.g., Fig. 2). A third example is competitive learning, where there is one major set of inputs to a network connected with associatively modifiable synapses, and mutual (e.g., lateral) inhibition between the output neurons (through e.g., inhibitory feedback neurons) (see Fig. 3). Competitive networks can be used to build feature analyzers by learning to respond to clusters of inputs which tend to co-occur, and may be fundamental building blocks of perceptual systems (Rolls and Treves, 1998). Allowing short range excitatory connections between neurons (as in the cerebral cortex) and longer range inhibitory connections can lead to the formation of topographic maps where the closeness in the map reflects the similarity between the inputs being mapped (Kohonen, 1995; Hertz et al., 1990; Rolls and Treves, 1998).

The question then arises as part of understanding brain function of how the networks found in different brain areas actually have evolved, and how they may be specified by genes. A principal aim of this paper is

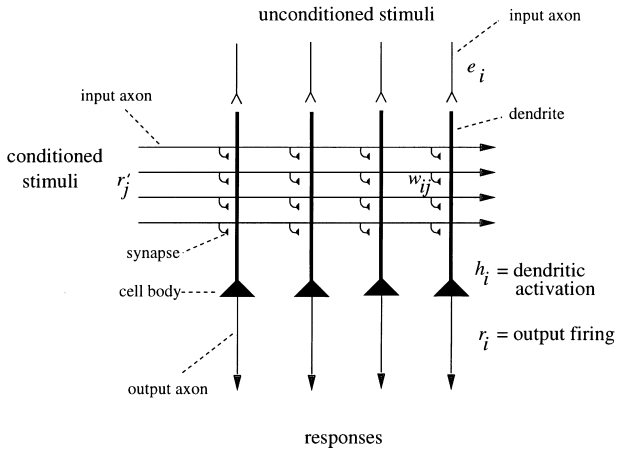

Fig. 1. A pattern association neural network. An unconditioned stimulus has activity or firing rate e_i for the ith neuron, and produces firing r_i of the ith neuron. The conditioned stimuli have activity or firing rate r'_j for the jth axon.

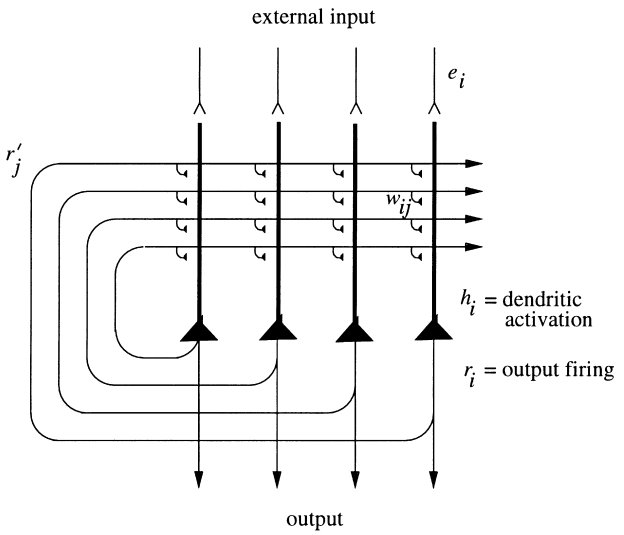

Fig. 2. The architecture of an autoassociative neural network.

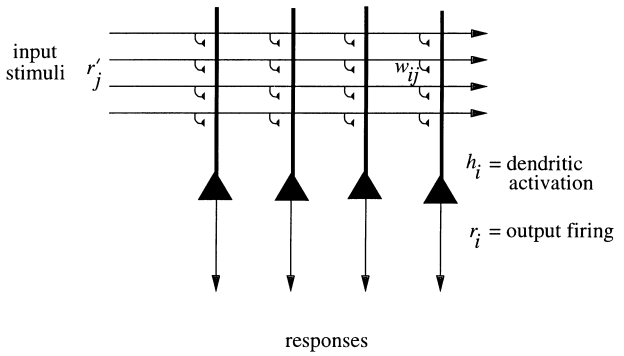

Fig. 3. The architecture of a competitive neural network.

by a process of comparing the networks found in different brain areas, and in particular in how they differ from each other, to suggest a set of parameters being specified by genes which could lead to the building during development of the neuronal network functional architectures found in different brain regions. The choice of parameters is guided not only by comparison of the functional architecture of different brain regions, but also by what parameters, if specified by genes, could with a reasonably small set actually build the networks found in different brain regions. The idea is that if these parameters are specified by different genes, then genetic reproduction and natural selection using these parameters could lead to the evolution of neuronal networks in our brains well adapted for different functions.

A second principal aim of the paper is to show how the sufficiency of the general approach, and the appropriateness of the particular parameters selected, can be tested and investigated using genetic algorithms which actually use the hypothesised genes to specify networks. We show this by implementing a genetic algorithm, and testing whether it can search the high dimensional space provided by the suggested parameters to specify and build neuronal networks that will solve particular computational problems. The computational problems we choose for this paper are simple and well defined problems that can be solved by one-layer networks. These problems are pattern association, autoassociation, and competition, which require quite different architectures for them to be solved (Rolls and Treves, 1998; Hertz et al., 1990) (see Figs. 1–3). Because the problems to be solved are well specified, we can define a good fitness measure for the operation of each class of network, which will be used to guide the evolution by reproduction involving genetic variation and selection in each generation. Although we do not suppose that the actual parameters chosen here for illustration are necessarily those specified by mammalian genes, they have been chosen because they seem reasonable given the differences in the functional architecture of different brain areas, and allow illustration of the overall concept described in this paper about how different network architectures found in different brain regions evolve. Although the computational problems to be solved here have been chosen to have well understood one-layer neural network solutions, once this general approach has been established, it would be of great interest in future to examine problems which are solved by multilayer networks in the brain e.g., invariant object recognition (Ullman 1996; Rolls and Treves 1998; Wallis and Rolls 1997), and episodic memory (Rolls and Treves, 1998; McClelland et al., 1995), in order to understand how brain mechanisms to solve complex problems may evolve and how they may operate.

In recent years there has been increasing interest in the application of genetic algorithms to neural networks (see Vonk et al., 1997). Some of the applications that are potentially relevant to understanding biological nervous systems range from computer simulated animats (Nolfi et al., 1994; Gracias et al., 1996; Huber et al., 1996; Lund and Parisi, 1996) to behavioural robotics (Floreano and Mondada, 1994; Hoshino and Tsuchida, 1996; Kuwana et al., 1996; Husbands et al., 1998), with the genetic algorithm used as a tool to discover neural networks capable of solving certain kinds of task. However, the purpose of this paper is to explore how genes could specify the actual neuronal network functional architectures found in the mammalian brain, such as those found in the cerebral cortex. Indeed, this paper takes examples of some of the actual architectures and prototypical networks found in the cerebral cortex, and explores how these architectures could be specified by genes which allow the networks when built to implement some of the prototypical computational problems that must be solved by neuronal networks in the brain. Moreover, the three fundamental kinds of biological network considered here, pattern associators, auto-associators and competitive nets, are potentially the heterogeneous building blocks for the multilayer architectures found in mammalian brain systems such as the hippocampus and cerebral cortex (Rolls and Treves, 1998). Investigation of how genes could specify the operation of these single layer components of multilayer architectures is a necessary first step towards understanding how more complex multilayer architectures could be built.

2. Description of the genes that could build different types of neuronal network in different brain areas

The hypotheses for the genes that might specify the different types of neuronal network in different brain

areas are introduced next. The hypotheses are based on knowledge of the architectures of different brain regions, which are described in sources such as in Rolls and Treves (1998), Shepherd (1990), Braitenberg and Schuz (1991), Peters and Jones (1984) and Somogyi et al. (1998), and on a knowledge of some of the parameters that influence the operation of neuronal networks. These genes were used in the simulations in which evolution of the architectures was explored using genetic algorithms. The emphasis in this section is on the rationale for suggesting different genes. A more formal specification of the genes, together with additional information on how they were implemented in the simulations, is provided in Section 4.2. It may also be noted that large numbers of genes will be needed to specify for example the operation of a neuron. This paper focusses on those genes that, given the basic building blocks of neurons, may specify the differences in the functional architecture of different brain regions. In addition, we note that here we do not try to describe the operation and properties of simple neural networks, nor the empirical phenomena such as synaptic long-term potentiation and long-term depression which provide empirical support for some of the parameters chosen, but instead refer the reader to sources such as Rolls and Treves (1998) to which cross references are given, and where in addition references to the background literature are provided, and in Koch (1999).

The overall conception is as follows. There are far too few genes (in humans 60,000–80,000, Maynard Smith and Szathmary, 1999) to specify each synapse (i.e., which neuron is connected to which other neuron, and with what connection strength) in the brain. (The number of synapses in the human brain, with 10^{10}–10^{11} neurons in the brain, and perhaps an average of 10,000 synapses each, is in the order of 10^{14}–10^{15}). In any case, brain design is likely to be more flexible if the actual strength of each synapse is set up by self-organisation and experience. On the other hand, brain connectivity is far from random. Some indications about what is specified can be gathered by considering the connectivity of the hippocampus (see Rolls and Treves, 1998, Chapter 6). The CA3 pyramidal cells each receive approximately 50 mossy fibre synapses from dentate granule cells, 12,000 synapses from other CA3 cells formed from recurrent collaterals, and 3600 perforant path synapses originating from entorhinal cortex neurons (see Fig. 4, after Rolls and Treves (1998), Fig. 6.6). In the preceding stage, the 1,000,000 dentate granule cells (the numbers given are for the rat) receive one main source of input, from the entorhinal cortex, and each makes approximately 14 synapses with CA3 cells (see Fig. 4). On the basis of considerations of this type for many different brain areas, it is postulated that for each class of cell the genome specifies the approximate numbers of synapses the class will receive from a specified other class (including itself, for recurrent collaterals), and the approximate number of synapses its axons will make onto specified classes of target cells (including itself). The individual neurons with which synapses are made are not specified, but are chosen randomly, though sometimes under a constraint (specified by another gene) about how far away the axon should travel in order to make connections with other neurons. One parameter value of the latter gene might specify the widespread recurrent collateral system of the CA3 neurons (Rolls and Treves 1998, Section 6.4). Another value for the latter might specify much more limited spread of recurrent collaterals with the overall density decreasing rapidly from the cell of origin, which, as in the cerebral cortex and if accompanied by longer range inhibitory processes implemented by feedback inhibitory neurons, would produce center-surround organisation, and tend to lead to the formation of topological maps (see Kohonen (1995) and Rolls and Treves (1998), Section 4.6). The actual mechanism by which this is implemented is not the subject of this paper, but would presumably involve some (genetically specified) chemical recognition process, together with the production of a limited quantity of a trophic substance that would limit the number of synapses from, and made to, each other class of cell. Some of these processes would of course be taking stage primarily during development (ontogenesis), when simple rules such as making local connections as a function of distance away would be adequate to specify the connectivity, without the need

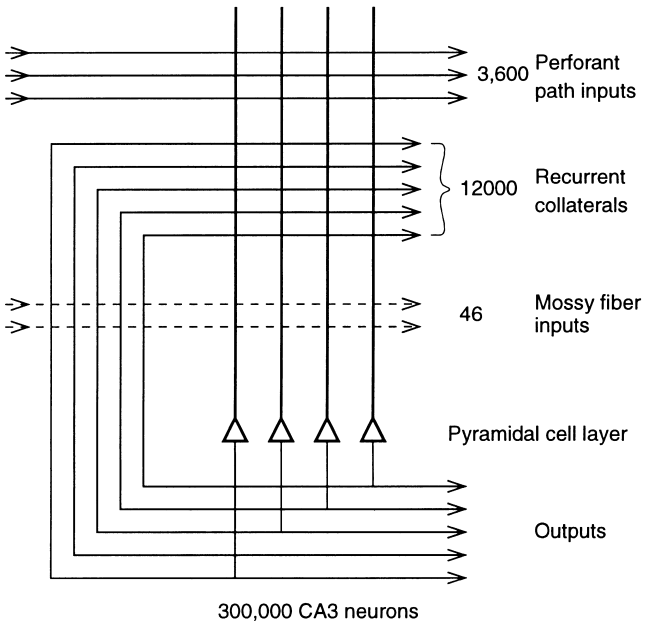

Fig. 4. Schematic architecture of hippocampal CA3 pyramidal cell network.

for long pathway connection routes (such as between the substantia nigra and the striatum) to be genetically encoded. It is presumably because of the complex genetic specification that would be required to specify in the adult brain the route to reach all target neurons that epigenetic factors are so important in embryological development, and that as a general rule the formation of new neurons is not allowed in adult mammalian brains.

It is a feature of brain neuronal network design that not only is which class of neuron to connect to apparently specified, but also approximately where on the dendrite the connections from each other class of neuron should be received. For example, in the CA3 system, the mossy fibres synapse closest to the cell body (where they can have a strong influence on the cell), the CA3 recurrent collaterals synapse on the next part of the dendrite, and the entorhinal inputs synapse on the more distal ends of the dendrites. The effect of the proximal relative to the more distal synapses will depend on a number of factors, including for distal inputs whether proximal inputs are active and thereby operating as current shunts producing division, and on the diameter of the dendrite, which will set its cable properties (Koch, 1999). If the dendritic cable diameter is specified as large, all inputs (including distal inputs) will sum reasonably linearly to inject current into the cell body, whereas if specified as small will tend to result in local computation on a dendrite, with perhaps strong local depolarization produced by a set of locally coactive inputs and synaptic modification local to a close coactive set of synapses on a dendrite, resulting in higher order association performed by what are called Sigma-Pi neurons (see Rolls and Treves, 1998, Section 1.5; Koch, 1999, Section 14.4.2). (Such higher order neural computation is not the main theme of Rolls and Treves (1998), nor of the networks considered in this paper, which generally operate by a uniform post-synaptic learning term, rather than one local to a patch of dendrite, though the genetic specification can easily be extended to incorporate the cable properties of neurons, by for example specifying parameters such as the diameter of the dendrite, see Koch, 1999).

Although inhibitory neurons have not been included in the examples already given, similar specifications would be applicable, including for example a simple specification of the different parts of the dendrite on which different classes of inhibitory neuron synapse in the hippocampus (Buhl et al., 1994), and hence whether the effect is subtractive or shunting (Koch, 1999). In cortical areas, both feedforward and feedback inhibition (the latter from pyramidal cells via inhibitory neurons back to the same population of pyramidal cells) could be produced by a simple genetic specification of this type.

Next, the nature of the synaptic connections, and the learning rule for synaptic modifiability, must be specified. One gene specifies in the simulation described later whether a given neuron class is excitatory or inhibitory. In the brain, this gene (or genes) would specify the transmitter (or transmitters in some cases) that are released, with the actual effects of for example glutamate being excitatory, and gamma-amino-butyric acid (GABA) inhibitory. The learning rule implemented at each synapse is determined by another gene (or genes). One possible effect specified is no synaptic modification. Another is a Hebb rule of associative synaptic plasticity (increase the synaptic strength if both presynaptic and postsynaptic activity are high, the simple rule implemented by associative long-term potentiation (LTP)). For this, the genes might specify NMDA (*N*-methyl-D-aspartate) receptors on the post-synaptic neurons together with the linked intracellular processes that implement LTP (Rolls and Treves, 1998; Wang et al., 1997; Buonomano and Merzenich, 1998). Another possible effect is long-term depression (LTD). This may be heterosynaptic, that is the synaptic weight may be decreased if there is high post-synaptic activity but low presynaptic activity. Part of the utility of this in the brain is that when combined with LTP in pattern associators and autoassociators the effect is to remove the otherwise positive correlation that would be produced between different input patterns if all patterns are specified by positive-only firing rates, as they are in the brain (see Rolls and Treves, 1998, Sections 2.4.7 and 3.3.6). The effect of removing this correlation is to reduce interference between patterns, and to maximize the storage capacity. A further useful property of heterosynaptic LTD is that it can help to maintain the total synaptic strength onto a neuron constant, by decreasing synaptic strengths from inputs to a neuron which are inactive when the neuron is currently firing. This can be very useful in competitive networks to prevent some winning neurons from continually increasing their synaptic strength so that they win to all patterns, and can in addition be seen as part of the process by which the synaptic weight vector is moved to point in the direction of a current input pattern of neuronal activity (see Rolls and Treves, 1998, Chapter 4). Another type of LTD is homosynaptic, in which the synaptic strength decreases if there is high presynaptic activity but low or moderate postsynaptic activity. This might be useful in autoassociative synaptic networks, as combined with LTP and heterosynaptic LTD it can produce a covariance-like learning rule (see Rolls and Treves, 1998, Section A3.1.4), and it may also be useful in forcing neurons to be allocated to one feature or another in feature analyzers if combined with a sliding sensi-

tivity factor that depends on the average post-synaptic activity of a neuron (Bienenstock et al., 1982). Another learning rule of potential biological importance is a trace learning rule, in which for example the post-synaptic term is a short term average in an associative Hebb-like learning rule. This encourages neurons to respond to the current stimulus in the same way as they did to previous stimuli. This is useful if the previous stimuli are different views etc of the same object (which they tend statistically to be in our visual world), because this promotes invariant responses to different versions of the same object. Use of such a rule has been proposed to be one way in which networks can learn invariant responses (Földiák, 1991; Rolls, 1992; Wallis and Rolls, 1997; Rolls and Treves, 1998; Rolls and Milward, 2000; Rolls and Stringer, 2000). Such a trace could be implemented in real neurons by a number of different mechanisms, including slow unbinding of glutamate from the NMDA receptor (which may take 100 ms or more), and maintaining a trace of previous neuronal activity by using short term autoassociative attractor memories implemented by recurrent collaterals in the cerebral neocortex (Rolls and Treves, 1998). Other types of synaptic modification that may be genetically specified include non-associative LTP (as may be implemented by the hippocampal mossy fibre synapses), and non-associative LTD. Other genes working with these may set parameters such as the rapidity with which synapses learn (which in a structure such as the hippocampus may be very fast, in one trial, to implement memory of a particular episode, and in structures such as the basal ganglia may be slow to enable the learning of motor habits based on very many trials of experience); and the initial and maximal values of synapses (e.g., the mossy fibre synapses onto hippocampal CA3 cells can achieve high values).

Another set of genes specifies some of the biophysical parameters that control the operation of individual neurons (see Koch (1999) for background). One gene (in a simulation, or biologically perhaps several) specifies how the activation h_i of a neuron i is calculated. A linear sum of the inputs r'_j weighted by the synaptic weights w_{ij} is the standard one used in most models of neuronal networks (Rolls and Treves, 1998), and those simulated here, as follows:

$$h_i = \sum_j r'_j w_{ij} \tag{1}$$

where $\sum_j$ indicates that the sum is over the C input axons (or connections) indexed by j. An alternative is that there is non-linearity in this process, produced for example by local interactions in dendrites, including local shunting, affected most notably by the cable diameter of the dendrite, which is what such genes may control. For most pyramidal cells, the dendrite diameter is sufficiently large that linear summation to produce the net current injected into the cell bodies is a reasonable approximation (Koch, 1999). Several further genes set the activation function of the neuron. (The activation function is the relation between the activation of the neuron and its firing. Examples are shown in Fig. 5). One possibility is linear. A second, and the most biologically plausible, is to have a threshold, followed by a part of the curve where the firing rate increases approximately linearly with the activation, followed by a part of the curve where the firing rate gradually saturates to a maximum. This function can be captured by for example a sigmoid activation function as follows:

$$r_i = \frac{1}{1 + e^{-2\beta(h_i - \alpha)}} \tag{2}$$

where α and β are the sigmoid threshold and slope, respectively. The output of this function, also sometimes known as the logistic function, is 0 for an input of $-\infty$, 0.5 for h_i equal to α, and 1 for $+\infty$. For this type of activation function, at least two genes would be needed (and biologically there would probably be several to specify the biophysical parameters), one to control the threshold, and a second to control the slope. A third possibility is to have a binary threshold, producing a neuron which moves from zero activity below threshold to maximal firing above threshold. This activation function is sometimes used in mathematical modelling because of its analytic tractability. One variable which is controlled by the threshold (and to some extent the slope) of the activation function is the proportion of neurons in a population that are likely to be firing for any one input. This is the sparseness of the representation. If the neurons have a binary activation function, the sparseness may be measured just by the proportion of active neurons, and takes the value 0.5 for a fully distributed representation, in which half of the neurons are active. For neurons with continuous activation functions, the sparseness a may be defined as

$$a = \frac{\left(\sum_i r_i / N\right)^2}{\sum_i r_i^2 / N} \tag{3}$$

where N is the number of neurons in the layer (Rolls and Treves, 1998). This works also for binary neurons. To provide precise control of the sparseness in some of the simulations described below we made provision for this to be controlled as an option directly by one gene, rather than being determined indirectly by the

threshold and slope gene-specified parameters of the sigmoid activation function shown in Eq. (2).

3. Genetic selection of neural network parameters to produce different network architectures with different functions

Given the proposals just described for the types of parameter that are determined by different genes, we describe next how gene selection is postulated to lead to the evolution of neuronal networks adapted for particular (computational) functions. The description is phrased in terms of simulations of the processes using genetic algorithms, as investigations of this type enable the processes to be studied precisely. The processes involve reproduction, ontogenesis followed by tests of how well the offspring can learn to solve particular problems, and natural selection. First, a selection of genes is made for a set of G genotypes in a population, which should be of a certain minimum size for evolution to work correctly. The genes are set out on a chromosome (or chromosomes). (Effects of gene linkage on a chromosome are not considered here.) Each set of genes is a genotype. The selection of individual genotypes from which to breed is made a probabilistic function which increases with the fitness of the genotype, measured by a fitness function that is quantified by how well that genotype builds an individual that can solve the computational problem that is set. Having chosen two genotypes in this way, two genotypes to specify two new (haploid) offspring for the next generation are made by the genetic processes of sexual reproduction involving both gene recombination and mutation, which occur with specified probabilities. This process is repeated until G genotypes have been produced. Then G individuals are built with the network architectures specified by the G genotypes. The fitness of these individuals is then measured by how well they perform at the computational problem set. In order to solve the computational problem, the networks are trained by presenting the set of input patterns, and adjusting the synaptic weights in the network according to the learning rules specified by the genotype of that individual. The individuals then breed again with a probability of being selected for reproduction that is proportional to their fitness relative to that of the whole population. This process is allowed to proceed for many generations, during which the fitness of the best individual in the population, and the average fitness, both increase if evolution is working. This type of genetic process is an efficient method of searching through a high dimensional space (the space specified by the genes), particularly where the space has many local optima so that simple hill climbing is inefficient, and where there is a single measure of fitness (Holland, 1975; Ackley, 1987; Goldberg, 1989). An additional useful feature of gen-

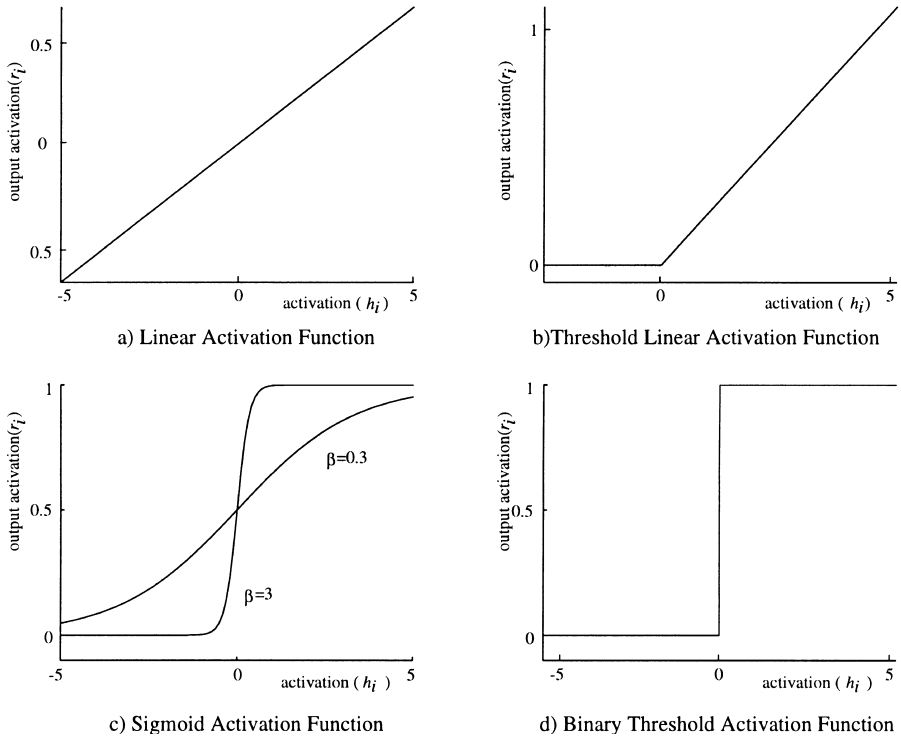

Fig. 5. Activation functions.

etic search is that past gene combinations, useful possibly in other contexts, can remain in the population for a number of generations, and can then be reused later on, without the need to search for those gene combinations again. This re-use of past combinations is one of the features of genetic search that can make it powerful, rapid, and show sudden jumps forward.

4. Simulation of the evolution of neural networks using a genetic algorithm

Next in this paper we describe a simulation of these processes using a genetic algorithm. The aims of the simulations are to demonstrate the feasibility and details of the proposal; to provide an indication of whether the proposed genes can be used to guide efficiently the evolution of networks that can solve computational problems of the type solved by the brain; and to provide a tool for investigating in more detail both the parameters that can best and biologically plausibly be genetically specified to define neural networks in the brain, and more complicated multilayer networks that operate to solve computationally difficult problems such as episodic memory or invariant visual object recognition. In this paper, the proposals are tested to determine whether three different types of one-layer network with different architectures appropriate for solving different computational problems evolve depending on the computational problem set. The problems are those appropriate for the pattern associator, autoassociator, and competitive networks shown schematically in Figs. 1–3.

4.1. The neural networks

The neural networks we consider have a number of classes of neuron. Within a class, a gene allows the number of neurons to vary from 1 up to N. For the simulations described here, N was set to 100. Within a class, the genetic specification of a neuron is homogeneous, with the neurons for that class having, for example, identical activation functions and connectivity probabilities. The number of classes will depend on the individual task, e.g. pattern association, autoassociation, competitive nets, etc, which can be described as one-layer networks, with one layer of computing elements between the input and output (see Figs. 1–3). However, in our simulations there is typically an input layer where patterns are presented, and an output layer where the performance of the network is tested. For the simulations described here, the number of classes was set to allow one-layer networks such as these to be built, but the number of layers that could be formed is in principle under genetic control, and indeed multilayer networks can be formed if there is a sufficient number of classes of neuron. Regarding the implementation of inter-layer connection topologies, the neurons in individual layers exist in a circular arrangement, and connections to a cell in one layer are derived from a topologically related region of the preceding layer. Connections to individual neurons may then be established according to either a uniform or Gaussian probability distribution centered on the topologically corresponding location in the sending layer.

On discrete time steps each neuron i calculates a weighted sum of its inputs as shown in Eq. (1). This is in principle the subject of genetic modification (see below), but the gene specifying this was set for the simulations described here to this type of calculation of the neuronal activation. Next the neuronal firing r_i is calculated, using for example the standard sigmoid function shown in Eq. (2) and allowing α and β the sigmoid threshold and slope to evolve genetically. Next, there is an optional procedure that can be specified by the experimenter to be called to set the sparseness of the firing rates r_i of a class of neuron according to Eq. (3), with a being allowed to evolve. After the neuronal outputs r_i have been calculated, the synaptic weights w_{ij} are updated according to one of a number of different learning rules, which are capable of implementing for example both long term potentiation (LTP) and long term depression (LTD), as described below, and which are genetically selected. For example, the standard Hebb rule takes the form

$$\Delta w_{ij} = k r_i r'_j, \tag{4}$$

where r'_j is the presynaptic firing rate and k is the learning rate.

4.2. The specification of the genes

The genes that specify the architecture and operation of the network are described next. In principle, each gene evolves genetically, but for particular runs, particular genes can be set to specified values to allow investigation of how other genes are selected when there are particular constraints. Each neural network architecture is described by a genotype consisting of a single chromosome of the following form

$$\text{chromosome} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_n \end{bmatrix} \tag{5}$$

where $\mathbf{c}_l$ is a vector containing the genes specifying the properties of neurons in class l, and n is the total number of classes that is set manually at the beginning of a run of the simulation. The vectors $\mathbf{c}_l$ take the form

$$\mathbf{c}_l = \begin{bmatrix} \mathbf{g}_l \\ \mathbf{h}_{l1} \\ \mathbf{h}_{l2} \\ \vdots \\ \mathbf{h}_{ln} \end{bmatrix} \quad (6)$$

where the vector $\mathbf{g}_l$ contains the *intra*-class properties for class l, and the vectors $\mathbf{h}_{lm}$ contain the *inter*-class connection properties to class l from class m where m is in the range 1–n.

The vector of intra-class properties takes the form

$$\mathbf{g}_l = \begin{bmatrix} b_l \\ \alpha_l \\ \beta_l \\ a_l \end{bmatrix} \quad (7)$$

where we have the following definitions for intra-class genes.

(1) b_l is the number of neurons in class l. b_l is an integer bounded between 2 and N, which was set to 100 for the simulations described here. Individual classes are restricted to contain more than one neuron since a key strategy we have adopted for enhancing the biological plausibility is to evolve classes composed of a number of neurons with homogeneous genetic specifications rather than to specify genetically the properties of individual single neurons.

(2) α_l is the threshold of the sigmoid transfer function (2) for class l. α_l is a real number bounded within the interval [0.0, 200.0].

(3) β_l is the slope of the sigmoid transfer function (2) for class l. β_l is a real number bounded within the interval [0.0, 200.0]. We note that low values of the slope will effectively specify a nearly linear activation function, whereas high values of the slope specify a nearly binary activation function (see Fig. 5).

(4) a_l is the sparseness of firing rates within class l as defined by Eq. (3). By definition a_l is a real number bounded within the interval [0.0, 1.0]. However, in practice we ensure a minimum firing sparseness by setting a_l to lie within the interval $[1.0/b_l, 1.0]$, where b_l is the number of neurons in class l. With binary neurons with output r equal to either 0 or 1, setting $a_l \geq 1.0/b_l$ ensures that at least one neuron is firing. For the simulations described in this paper, when this gene was being used, a was set to 0.5 and produced a binary firing rate distribution with a sparseness of 0.5 unless otherwise specified. The alternative way of calculating the firing rates was to allow genes α and β specifying the sigmoid activation function to evolve.

The vectors of inter-class connection genes specify the connections to class l from class m, and take the form

$$\mathbf{h}_{lm} = \begin{bmatrix} r_{lm} \\ s_{lm} \\ c_{lm} \\ e_{lm} \\ z_{lm} \\ t_{lm} \\ p_{lm} \\ \sigma_{lm} \\ q_{lm} \\ f_{lm} \\ k_{lm} \\ d_{lm} \\ u_{lm} \\ v_{lm} \end{bmatrix} \quad (8)$$

where we have the following definitions for inter-class genes.

(1) r_{lm} controls the inter-layer connection topology in that it helps to govern which neurons in class m make connections with individual neurons in class l. The exact definition of r_{lm} depends on the type of probability distribution used to make the connections, which is governed by the gene s_{lm} described below. For $s_{lm} = 0$, connections are established according to a uniform distribution, and r_{lm} specifies the number of neurons within the spatial region from which connections may be made. For $s_{lm} = 1$, connections are established according to a Gaussian distribution, where r_{lm} specifies the standard deviation of the distribution. For the Gaussian distribution, individual connections originate with 68% probability from within a region of 1 standard deviation away. These real valued variates are then rounded to their nearest integer values. As noted in Section 2, connection topologies are characteristically different for different connection types such as CA3 recurrent collateral connections, which are widespread, and intra-module connections in the cerebral cortex. This parameter may also be set by genetic search to enable the region of effect of inhibitory feedback neurons to be just greater than the region within which excitatory neurons receive their input; and to enable topographic maps to be built by arranging for the lateral excitatory connections to operate within a smaller range than inhibitory connections. This parameter may also be set to enable multilayer networks to be built with feedforward connectivity which is from a small region of the preceding layer, but which over many layers allows an output neuron to receive from any part of the input space, as happens in many sensory systems in which topology is gradually lost and as is implemented in a model of the operation of the visual cortical areas (Wallis and Rolls, 1997; Rolls and Milward, 2000). For the simulations described here, this gene was set to 100 to allow global connectivity, and thus to not specify local connectivity.

(2) s_{lm} specifies for the region gene r_{lm} the connec-

tion probability distribution to class l from class m. s_{lm} takes the following values.

- $s_{lm} = 0$: Connections are established according to a uniform probability distribution within the region r_{lm}.
- $s_{lm} = 1$: Connections are established according to a Gaussian probability distribution. r_{lm} specifies the size of the region from which there is a 68% probability of individual connections originating.

In the experiments described here, the values of s were set to 0.

(3) c_{lm} is the number of connections each neuron in class l receives from neurons in class m. c_{lm} takes integer values from 0 to b_m (the number of neurons within class m). These connections are made probabilistically, selecting for each class l of neuron c_{lm} connections from class m. The probabilistic connections are selected either from a uniform probability distribution (as used here), or according to a Gaussian probability distribution. In both cases the local spatial connectivity within the two classes l and m is specified by the region parameter gene r_{lm}.

(4) e_{lm} specifies whether synapses to class l from class m are excitatory or inhibitory. e_{lm} takes the following values.

- $e_{lm} = 0$: Connections are inhibitory
- $e_{lm} = 1$: Connections are excitatory

For inhibitory connections with $e = 0$, the weights in Eq. (1) are scaled by -1, and no learning is allowed to take place.

(5) z_{lm} specifies whether connections from class m to class l are additive or divisive. z_{lm} takes the following values.

- $z_{lm} = 0$: Connections are divisive
- $z_{lm} = 1$: Connections are additive

This gene selects whether the neuronal activation h_i is calculated according to linear summation of the input firing rates r'_j weighted by the synaptic strengths w_{ij}, as specified by Eq. (1), and as used throughout this paper by setting $z_{lm} = 1$; or whether with $z_{lm} = 0$ there is local computation on a dendrite including perhaps local summation of excitatory inputs and shunting effects produced by inhibitory inputs or inputs close to the cell body. The most important physical property of a neuron that would implement the effects of this gene would be the diameter of the dendrite (see Section 2 and Koch, 1999). If set so that $z_{lm} = 0$ for local multiplicative and divisive effects to operate, it would be reasonable and in principle straightforward to extend the genome to include genes which specify where on the dendrite with respect to the cell body synapses are made from class m to class l neurons, the cable properties of the dendrite, etc.

(6) t_{lm} specifies how the synaptic weights to class l from class m are initialised at the start of a simulation. t_{lm} takes the following values.

- $t_{lm} = 0$: Synaptic weights are set to zero. This might be optimal for the conditioned stimulus inputs for a pattern associator, or the recurrent collateral connections in an autoassociator, for then existing connections would not produce interference in the synaptic connections being generated during the associative learning. This would produce a non-operating network if used in a competitive network, for an input could not produce any output.
- $t_{lm} = 1$: Synaptic weights are set to a uniform deviate in the interval [0, 1] scaled by the constant value encoded on the genome as q_{lm} (described below).
- $t_{lm} = 2$: Synaptic weights are set to the constant value encoded on the genome as q_{lm} (described below). This could be useful in an associative network if a learning rule that allowed synaptic weights to decrease as well as increase was specified genetically, because this would potentially allow correlations between the input patterns due to positive-only firing rates to be removed, yet prevent the synapses from hitting the floor of zero, which could lose information (Rolls and Treves, 1998).
- $t_{lm} = 3$: Synaptic weights are set to a Gaussian function of distance x away from the afferent neuron in class l. This function takes the form

$$f(x) = \frac{1}{\sigma\sqrt{2\Pi}} e^{-x^2/2\sigma^2}$$

where σ is the standard deviation. This would be an alternative way to implement local spatial connectivity effects to those implemented by r_{lm}.

(7) p_{lm} is the scale factor used for synaptic weight initialisation with a Gaussian distribution (i.e., for $t_{lm} = 3$). p_{lm} is a real number bounded within the interval [0.0, 100.0].

(8) σ_{lm} is the standard deviation used for synaptic weight initialisation with a Gaussian distribution (i.e., for $t_{lm} = 3$). σ_{lm} is a real number bounded within the interval [0.001, 10.0]. σ_{lm} is restricted to be greater than 0.001 to avoid a singularity that would occur in the Gaussian function for $\sigma_{lm} = 0.0$.

(9) q_{lm} is the scale factor used for synaptic weight initialisation for $t_{lm} = 1$ and $t_{lm} = 2$ (see above). q_{lm} is a real number bounded within the interval [0.0, 100.0]. If t_{lm} is 1 or 2, q_{lm} would be expected to be small for pattern associators and autoassociators (as described above).

(10) f_{lm} specifies which learning rule is used to

update weights at synapses to class l from class m. f_{lm} takes the following values.

- $f_{lm} = 0$: Learning rule 0 (no learning) $\Delta w_{ij} = 0$.
- $f_{lm} = 1$: Learning rule 1 (Hebb rule) $\Delta w_{ij} = k r_i r'_j$.
- $f_{lm} = 2$: Learning rule 2 (modified Hebb rule) $\Delta w_{ij} = k r_i (r'_j - \langle r'_j \rangle)$ where $\langle \cdot \rangle$ indicates an average value. In this rule LTP is incorporated with heterosynaptic LTD set to keep the average weight unchanged. Use of this rule can help to remove the correlations between the input patterns produced by positive-only firing rates (Rolls and Treves, 1998). If this rule is selected, it will work best if the synaptic weight initialisation selected genetically is not zero (to allow the weights to decrease without hitting a floor), and is optimally a constant (so that noise is not added to the synaptic weights which implement the association).
- $f_{lm} = 3$: Learning rule 3 (modified Hebb rule) $\Delta w_{ij} = k r_i (r'_j - w_{ij})$. This rule has the effect of helping to maintain the sum of the synaptic weights on a neuron approximately constant, and is therefore helpful in competitive networks (Rolls and Treves, 1998). This rule accords with the neurophysiological observation that it is easier to demonstrate LTD after LTP has been produced.
- $f_{lm} = 4$: Learning rule 4 (covariance rule) $\Delta w_{ij} = k(r_i - \langle r_i \rangle)(r'_j - \langle r'_j \rangle)$.
- $f_{lm} = 5$: Learning rule 5 (homosynaptic LTD with LTP) $\Delta w_{ij} = k(r_i - \langle r_i \rangle) r'_j$.
- $f_{lm} = 6$: Learning rule 6 (non-associative LTP) $\Delta w_{ij} = k r'_j$. This rule may be implemented in the hippocampal mossy fibre synapses.
- $f_{lm} = 7$: Learning rule 7 (non-associative LTD) $\Delta w_{ij} = -k r'_j$.
- $f_{lm} = 8$: Learning rule 8 (trace learning rule) $\Delta w_{ij} = k \bar{r}_i^{\tau} r'^{\tau}_j$ where the trace $\bar{r}_i^{\tau}$ is updated according to

$$\bar{r}_i^{\tau} = (1 - \eta) r_i^{\tau} + \eta \bar{r}_i^{\tau - 1} \quad (9)$$

and we have the following definitions: $\bar{r}_i^{\tau}$ is the trace value of the output of the neuron at time step τ, and η is a trace parameter which determines the time or number of presentations of exemplars of stimuli over which the trace decays exponentially. The optimal value of η varies with the presentation sequence length. Further details, and a hypothesis about how this type of learning rule could contribute to the learning of invariant representations, are provided by Wallis and Rolls (1997), Rolls and Treves (1998) and Rolls and Milward (2000). For the simulations described in this paper, this learning rule was disabled.

In some simulations described later, the synaptic weights were clipped to be ≥ 0, consistent with what is biologically plausible. However, in some simulations we explored the impact on evolution and performance of allowing the synaptic weights to vary from positive to negative. In addition, an implementation detail for rules $f = 2, 4$ or 5, where an estimate of the average firing rate of a set of neurons $\langle r \rangle$ has to be subtracted, is that $\langle r \rangle$ is taken to be 0.5, which is the average firing of a binary vector with the sparseness value of 0.5 which was used for the test patterns and was that of the desired output patterns except where stated otherwise.

(11) k_{lm} is the learning rate k for synaptic weights to class l from class m. k_{lm} is a real number bounded within the interval [0.0, 10.0].

(12) d_{lm} is the maximum size allowed for the absolute value of synaptic weight updates from class m to class l. d_{lm} is a real number bounded within the interval [0.0, 10.0].

(13) u_{lm} is the maximum size allowed for synaptic weights to class l from class m. u_{lm} is a real number bounded within the interval [−100.0, 100.0].

(14) v_{lm} is the minimum size allowed for synaptic weights to class l from class m. v_{lm} is a real number bounded within the interval [−100.0, 100.0].

Implementation details are that v is bounded to be less than u; and that during initialisation of the synaptic weights, v and u are applied after the other initialisation parameters have operated.

4.3. The genetic algorithm, and general procedure

The genetic algorithm used is a conventional one described by Goldberg (1989). Each individual genotype was a haploid chromosome of the type just defined. The values for the genes in the initial chromosome were chosen at random from the possible values. (In this simulation, if a value in the genotype was one of a large number of possible values, then a single integer or real-valued gene was used, whereas in the brain we would expect several genes to be used, with perhaps each of the several genes coding for different parts of the range with different precision. For example, c_{lm}, the number of synaptic connections to a neuron in class l from a neuron in class m, is to be chosen from 2 to 100 in the simulation, and uses a single integer. In the brain, where we might expect the number of connections per neuron to vary in the range 5–50,000, one possible scenario is that five different genes would be used, coding for the values 5, 50, 500, 5000 and 50,000. We would thus expect that the number of genes required to specify the part of the genotype described here would increase by a factor in the order of 5 in the brain.) There was a set number of genotypes (and hence individuals) per generation, which was 100 for the simulations described. (Too small a number reduces the amount of diversity in the population sufficiently to lead to poor performance of the genetic evolution.) From each genotype an individual

network was constructed, with all connections being chosen probabilistically within the values specified by the genotype, and with a uniform probability distribution (i.e., s_{lm} was set to 0 for the simulations described), and the initial values for the synapses were also chosen as proscribed by the genotype. That individual was then tested on the computation being studied, e.g. pattern association, and the fitness (performance) of the network was measured. The actual testing involved presenting the set of patterns to be learned (as specified below for each network) to the network while allowing learning to occur, and then testing the performance of the network by presenting the same, similar, or incomplete test patterns and measuring the output of the network. The fitness measure used for each class of network is defined below. Unless otherwise stated, each genotype was tested 20 times to obtain an average fitness measure for that genotype. (This step, though not necessary at all for successful evolution, did ensure, given the probabilistic way in which individual connections and their values were assigned, that an accurate measure of the fitness of that genotype was obtained, and hence allowed smoother evolution curves to be plotted. In real life, the numbers specified for the networks would be larger, and the statistical effects of small numbers would be smaller. On the other hand, chance environmental factors might be larger in real life.) This process was repeated until a fitness measure had been obtained for each of the genotypes.

The next generation of genotypes was then bred by reproduction from the previous genotypes. Two genotypes were selected for reproduction with a probability that was proportional to their fitness relative to the sum of the fitnesses of all individuals in that generation. That is, the probability P_i of an individual i being selected is given by

$$P_i = \frac{F_i}{\sum_j F_j} \tag{10}$$

where F_j is the fitness of an individual genotype j, and the sum $\sum_j F_j$ is over all individuals in the population. The genotypes were then bred to produce two offspring genotypes using the processes of recombination and mutation. The probability of recombination was set to 0.4 (so that it occurred with a 40% chance for every breeding pair in every generation), and the probability of mutation of each gene was set to 0.05. The computational background to this is that recombination allows a non-linear search of a high dimensional space in which gene complexes or combinations may be important, allowing local performance hills in the space to be detected, and performance once in a local hill to improve. Mutation on the other hand enables occasional moves to be tried to other parts of the space, usually with poorer performance, but occasionally moving a genotype away from a well explored location to a new area to explore. Both processes enable the search to be much more efficient in a high dimensional space with multiple performance hills or mountains than with simple gradient ascent, which will climb the first hill found, and become stuck there (Ackley, 1987; Goldberg, 1989). The recombination took place with respect to a random point on the chromosome each time it occurred. The mutation also took place at a random place on the chromosome each time it occurred, and the single gene being mutated was altered to a new random value within the range specified for that gene. This reproduction process was repeated until sufficient individual genotypes for the next generation (100 unless otherwise stated) had been produced. The fitness of those genotypes was then measured by building and testing the networks they specified as previously outlined. Then the whole reproduction, ontogenesis, and testing to measure fitness, processes were repeated for many generations, to determine whether the fitness would increase, and if so, what solutions were found for the successful networks.

4.4. Pattern association networks

The computational problem set was to associate pairs of random binary patterns of 1s and 0s with sparseness of 0.5 (i.e. with half the elements 0 and the other half 1s), an appropriate problem for a pattern association net (Rolls and Treves, 1998). There were 10 pattern pairs, which places a reasonable load on the network (Rolls and Treves, 1998). There were two classes of neuron. One of the patterns of the pair, the unconditioned stimulus, was set during learning to activate the 100 output neurons (class 2) during learning, while the other pattern of the pair, the conditioned stimulus (see Fig. 1), was provided by the firing of 100 input neurons (class 1). The fitness measure was obtained by providing each conditioned stimulus (or a fraction of it), and measuring the correlation of the output vector of firing rates, r_i in Fig. 1 where $i = 1$ to 100, with that which was presented as the unconditioned stimulus during learning. This correlation measure ranges from 1.0 for perfect recall to 0.0 (obtained for example if the output of the pattern associator is 0). The fitness measure that was used in most runs, and is plotted in the Figures and Tables was the square of this correlation (used to increase the selectivity of the fitness function). (All of the possible activation functions produced firing rates greater than or equal to 0.) (In practice, a pattern associator with associatively modifiable synapses, random patterns, and the sizes and loading of the networks used here

can achieve a maximum correlation of approximately 1.000.) It may be noted that the genetics could form connections onto individual neurons of any number (up to $N = 100$, the number of input and output neurons in the network) and type between classes 1 to 2 (feedforward connections), and 2 to 2 (recurrent collaterals), 2 to 1, and 1 to 1. (However, in the simulations described in this Section, during retrieval the firing rates of class 2 neurons were set to 0 initially corresponding to no retrieved pattern before the cue is applied, and this had the effect that any recurrent collateral synapses from class 2 to class 2 would not affect retrieval.) For the simulations performed, the operation of inhibitory feedback neurons was not explicitly studied, although when using the sigmoid activation function the neurons had to select a threshold (by altering α_l). (If the alternative way of calculating the firing rates using the sparseness parameter a_l to allow only some output neurons to fire was selected for a particular run, then a_l was itself either allowed to evolve, or set to the same value as that of the unconditioned stimulus. When setting the output sparseness in this way using a_l, the activation function of the neurons was linear.)

The values of the fitness measures for a typical run of the simulations is shown in Fig. 6, along with the values of specific genes. The genes shown are key connection properties from class 1 to class 2, and are: learning rate k_{21}, number of connections c_{21}, weight scaling q_{21}, weight initialisation option t_{21}, and the learning rule f_{21}. In the figure the values of the different variables are normalised in the following way. The net fitness, number of connections c_{21}, weight initialisation t_{21}, and learning rule f_{21} are normalised by their maximum possible values (1.0, 100, 3, 8, respectively), while the learning rate k_{21} and weight scaling q_{21} are normalised by the maximum values that actually occurred during the simulation (7.9 and 52.99, respectively). The fitness measure is the fitness of the best genotype in the population, and for this run the sparseness of the output representation was set to the value of 0.5 by using the sparseness gene set to this value. The synaptic weights were clipped to be 0 for the run shown in this figure and in the other figures

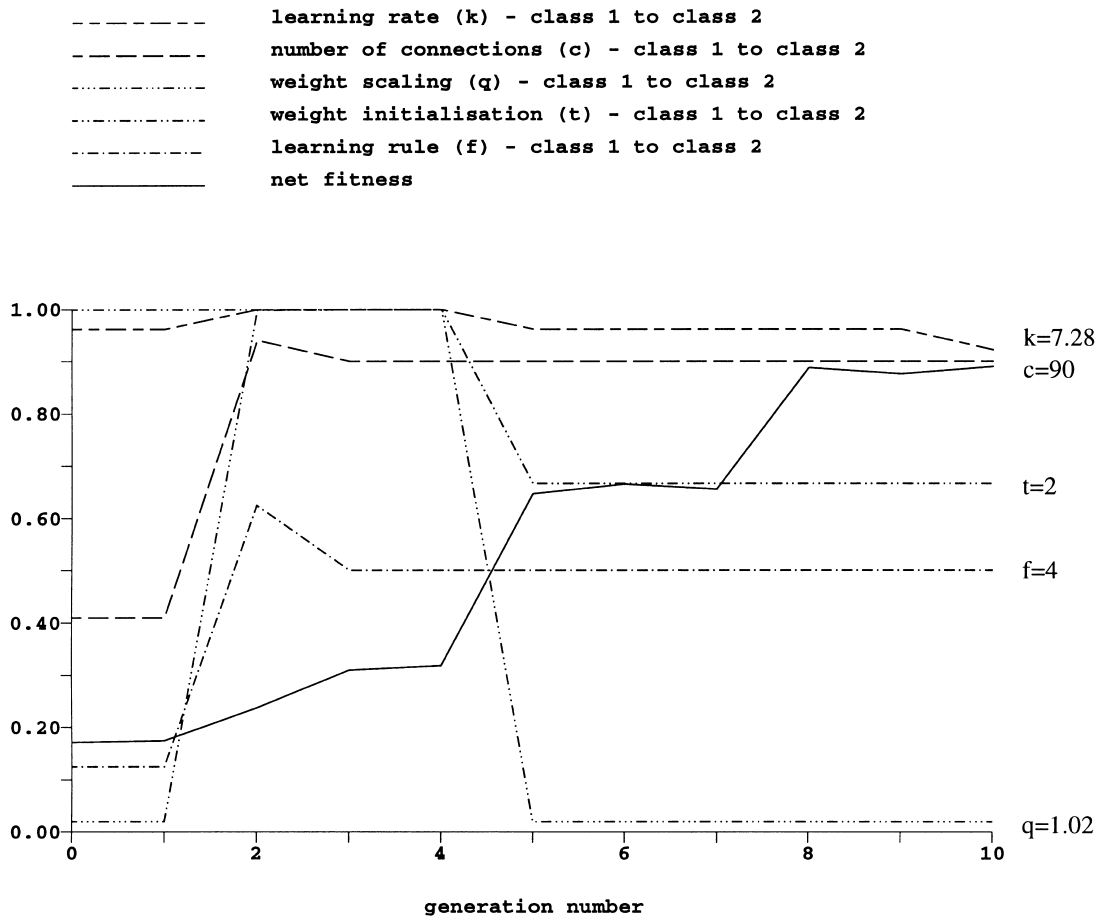

Fig. 6. Fitness of the best genotype in the pattern association task as a function of the number of generations, together with the values of selected genes. The simulation is performed with the firing rates assigned binary values of 0 or 1 according to the neuronal activations, with the proportion of neurons firing set by the firing rate sparsity gene a.

included in this paper. It is shown that evolution to produce genotypes that specify networks with high performance occurred over a number of generations. Some genes, such as that specifying the learning rule, settled down early, and changes in such genes can sometimes be seen to be reflected in jumps in the fitness measure. An example in Fig. 6. is the learning rule, which settled down (for the best genotype) by generation 3 to a value of 4 which selects the covariance learning rule. Another example is that there is a jump in the fitness measure from generation 4 to 5 which occurs when the synaptic weight initialisation gene t_{21} changes from a Gaussian distribution to a constant value. This would increase the performance, as described below. The simultaneous jump of the synaptic weight scaling q_{21} from a high value to a low value on the other hand would not be expected to increase the fitness with learning rule 4 and a constant weight initialisation, again as made clear below. The increase in fitness from generation 7 to 8 occurred when the minimum weight gene v_{21} changed from 11.6 to −62.5 (not plotted), which would also be expected to increase performance, as described below. For other genes, where the impact is less or zero, the values of the genes can fluctuate throughout evolution, because they are not under selection pressure, in that they do not contribute greatly to fitness. Every run of the simulation eventually produced networks with perfect performance. The selection pressure (relative to other genotypes) could be increased by taking a power greater than 1 of the correlation fitness measure. A high value could produce very rapid evolution, but at the expense of minimising diversity early on in evolution so that potentially good genes were lost from the population, with then a longer time being taken later to reach optimal performance because of the need to rely on mutations. Low values resulted in slow evolution. In practice, raising the correlation fitness measure to a power of 1 (no change) or 2 produced the best evolution. (For the pattern associator and autoassociator, the fitness measure used and plotted in the figures was the correlation measure raised to the

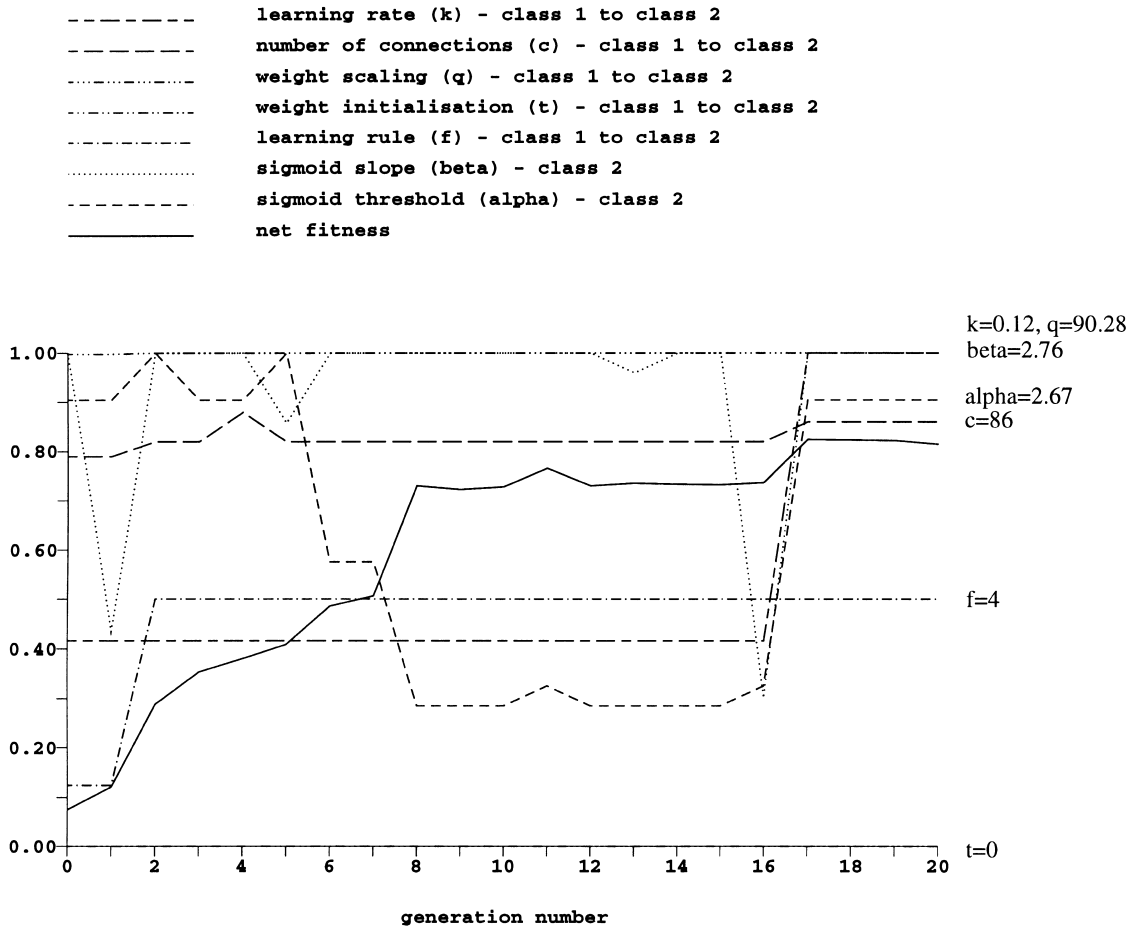

Fig. 7. Fitness of the best genotype in the pattern association task as a function of the number of generations, together with the values of selected genes. The simulation is performed with the firing rates calculated according to the sigmoid activation function (2) using the values of the genes α and β.

power 2.) The best networks all had close to 100 excitatory feedforward connections (gene c_{21}) to each class 2 (output) neuron from the class 1 (input) neurons, which is the minimal number necessary for optimal performance of a fully loaded pattern associator and the maximum number allowed. Moreover, the excitatory feed forward connections for all the successful networks were specified as associative, and in particular f_{21} was selected for these networks as 2 or 4. In addition, with the synaptic weights initialised according to a uniform probability distribution ($t_{21} = 1$) relatively successful networks required genes which specified for the associative feedforward connections low initial synaptic weights (using appropriate values for q_{21}), and/or high learning rates k_{21}, to minimise the synaptic initialisation noise introduced into the associatively learned connections (see further discussion below). However, through the processes of recombination and mutation, such genes might also become combined with a learning rule and weight initialisation combination (e.g., $f_{21} = 4$ and $t_{21} = 2$) that would not itself require low initial synaptic weights for good performance. This is demonstrated in Fig. 6. where the best fit genotype of the last generation has $f_{21} = 4$, $t_{21} = 2$, $q_{21} = 1.02$ and $k_{21} = 7.28$.

An example of a run of the pattern association network using the sigmoid activation function in which the genes that specify the parameters for the threshold α_l and the slope β_l were allowed to evolve for classes 1 and 2 is shown in Fig. 7. In this figure the genes are normalised in a similar manner to Fig. 6, with the additional genes sigmoid slope β_2 and sigmoid threshold α_2 for class 2 also normalised by their maximum values that occurred during the simulation. (For Fig. 7 the values of the learning rate k_{21}, weight scaling q_{21}, sigmoid slope β_2 and sigmoid threshold α_2 are normalised by 0.12, 90.34, 2.76, 2.95, respectively.) When the fitness increased from 7 to 8, the sigmoid threshold α_2 decreased. Between generations 16 and 17, when there was only a small improvement in fitness, the threshold increased (from 0.96 to 2.67), so also did the learning rate k_{21}, thus maintaining the ratio of activation to threshold approximately constant. This illustrates the way in which genes can interact with each other. If in evolution the value of a gene settles or becomes relatively fixed early on, other genes may evolve to values that allow the network to operate well even given the constraints forced by relatively fixed values of other genetically set parameters. The data shown in Fig. 7 are typical of many runs in which genetic evolution could find appropriate values for the threhold α_2 and slope β_2 of the activation function of neurons (as well as of all the other genes) to lead to the production of output firing rate patterns with the appropriate sparseness or more generally firing rate distributions to achieve optimal results in a pattern associator.

To help understand in full detail the actual gene selections found by the genetic algorithm in the simulations, we simulated the pattern associator with manually selected gene combinations for the feedforward connections from class 1 to class 2. In particular, we calculated the network fitnesses for particular combinations of learning rule f_{21}, weight initialisation option t_{21}, and weight scaling q_{21}. (This optional facility of the simulator allowed systematic investigation of the effects of one gene, which could be held fixed at a particular value, on the values of the other genes selected during evolution.) The results of some of these simulations are shown in Tables 1 and 2, both for the normal situation biologically and in most of the runs performed where the weights were clipped to be ≥ 0, and when they were allowed to become negative. (In all cases, the firing rates were ≥ 0, and the learning rate k_{21} was set to 1.) It is shown in Table 1 that with the weights clipped at zero, the best performance (1.000) was obtained with the learning rule f_{21} set to 2 (LTP

Table 1
Performance of pattern association networks with clipping of synaptic weights to be ≥ 0[a]

	Synaptic weight initialisation option for different values of t						
	0	1 ($q = 0.5$)	1 ($q = 5.0$)	1 ($q = 50.0$)	2 ($q = 0.5$)	2 ($q = 5.0$)	2 ($q = 50.0$)
Rule 1	0.261	0.263	0.243	0.047	0.259	0.257	0.256
Rule 2	0.573	0.695	0.581	0.025	0.826	1.000	1.000
Rule 3	0.135	0.139	0.130	0.002	0.136	0.135	0.001
Rule 4	0.878	0.965	0.568	0.023	0.996	1.000	1.000
Rule 5	0.240	0.252	0.241	0.051	0.253	0.258	0.256
Rule 6	0.000	0.007	0.009	0.007	0.000	0.000	0.000
Rule 7	0.000	0.000	0.007	0.007	0.000	0.000	0.000

[a] The performance measure is the fitness, which was r^2, where r is the average over the 10 training patterns of the correlation of the pattern retrieved from the network with the training pattern. The different columns are for the different types of weight initialisation gene ($t = 0$, weights initialised to 0; $t = 1$, weights initialised to a uniform probability distribution; $t = 2$, weights initialised to a constant value) and weight scaling gene q.

plus heterosynaptic long-term depression), or to 4 (covariance), provided that the weights were initialised to a high enough positive constant value (for the simulations performed to a value of 5 as indicated in the table) that when decremented by the learning rule during learning the weights did not hit the floor of 0. (If the weights were not clipped to be ≥ 0 during learning, then as shown in Table 2 both these rules were again the best, but weight initialisation could in addition be set to zero or any constant.) These results are in accord with what would be predicted (Rolls and Treves, 1998), and help to show why the genetic algorithm tended to produce gene combinations which included rule 2 or rule 4 in combination with weight initialisation to a positive constant sufficiently great to enable the synapses not to hit the floor of 0. In practice most of the runs had best genotypes that specified $f_{21} = 4$, the covariance learning rule, because, as shown in Table 1, this would be more likely to work for a wide range of values of the genes, including t_{21} and q_{21}, and therefore probabilistically would be likely to be selected. We note that one reason that the covariance learning rule can produce good performance even when the weight initialisation is to small constant values (e.g., $t_{21} = 2$, $q_{21} = 0.5$), is because with 0 pre- and post-synaptic firing rates, the weight will be given a positive increment by the learning rule, which helps to lift the weights off the floor of 0. The Hebbian associative rule without weight decrementing $f_{21} = 1$ was not chosen for the fit genotypes given the maximum fitness obtainable without weight decrementing shown in Table 2 and required to remove the positive correlation introduced by firing rates ≥ 0 (Rolls and Treves, 1998). The associative learning rule 3 envisaged to be useful in competitive networks was not selected for the successful genotypes in pattern association, as it produced only poor maximum fitness values as shown in Tables 1 and 2. In addition, Tables 1 and 2 show that if the weight initialisation is drawn from a continuous probability distribution $t_{21} = 1$, then performance is less good because this introduces noise into the synaptic matrix, the effects of which can be partly ameliorated by a high value of the learning rate constant k_{21}. Consistent with this, the optimal genotypes found during evolution always selected $t_{21} = 2$ (weights initialised to a positive constant), or, if the weights were not clipped to be ≥ 0 during learning, sometimes t_{21} was selected to be 0 (weights initialised to 0).

4.5. Autoassociative networks

The computational problem set was to learn random binary pattern vectors (of 1s and 0s with sparseness of 0.5, i.e. with half the elements 0 and the other half 1s), and then to recall the whole pattern when only half of it was presented. This is called completion, and is an appropriate test for an autoassociator (Rolls and Treves, 1998). An additional criterion, inherent in the way the simulation was run, was to perform retrieval by using iterated processing in an attractor-like state of steady memory retrieval like that of continuing neuronal firing in a short-term memory (Rolls and Treves, 1998). A one-layer autoassociation network capable of this is shown in Fig. 2, and in our simulations there was a single class of neurons forming an output layer. There were 10 patterns to be learned, which loads the network to approximately 70% of its theoretical capacity (Rolls and Treves, 1998). During learning, a 100-element pattern was presented as the external input to force the 100 output neurons to fire with that pattern, and synaptic modification was allowed to occur by whichever learning rule was selected for that genotype. The fitness measure was obtained by providing half (the first 50 elements) of each pattern to the network, allowing the network to iterate 10 times (which is sufficient for retrieval in a normally operating autoassociation memory with discrete time steps) with the input removed after the first iteration, and measuring the correlation of the output vector of firing rates, r_i in Fig. 2 where $i = 1$–100, with the original complete pattern that was learned. This correlation measure ranges from 1.0 for perfect

Table 2
Performance of pattern association networks without clipping of synaptic weights[a]

	Synaptic weight initialisation option for different values of t						
	0	1 ($q = 0.5$)	1 ($q = 5.0$)	1 ($q = 50.0$)	2 ($q = 0.5$)	2 ($q = 5.0$)	2 ($q = 50.0$)
Rule 1	0.254	0.261	0.239	0.050	0.255	0.258	0.258
Rule 2	1.000	1.000	0.597	0.023	1.000	1.000	1.000
Rule 3	0.138	0.140	0.132	0.001	0.141	0.135	0.001
Rule 4	1.000	1.000	0.586	0.023	1.000	1.000	1.000
Rule 5	0.260	0.258	0.242	0.051	0.255	0.257	0.257
Rule 6	0.000	0.007	0.007	0.007	0.000	0.000	0.000
Rule 7	0.000	0.007	0.007	0.007	0.000	0.000	0.000

[a] Conventions as in Table 1.

recall to 0.0. (In practice, an autoassociator with associatively modifiable synapses, random fully distributed binary patterns, and the sizes and loading of the networks used here, can achieve on average a maximum fitness of approximately 0.996 when half the original pattern is presented as the recall cue.)

The values of the fitness measure and selected genes for a typical run of the simulations are shown in Fig. 8 (In Fig. 8 the genes are normalised in a similar manner to Fig. 6, with the values of the learning rate k_{11} and weight scaling q_{11} normalised by 8.74 and 81.10, respectively.) The fitness measure is the fitness of the best genotype in the population. It is shown that evolution to produce genotypes that specify networks with optimal performance occurred over a number of generations. Most runs of the simulation produced networks with optimal performance. The best networks all had close to 100 excitatory feedback connections from the output neurons (class 1) to themselves, which is the minimal number necessary for optimal performance of a fully loaded autoassociator given that there are 100 output neurons and that this is the maximum number allowed. For the run shown in Fig. 8 the sparseness of the output representation was set to the value of 0.5 by using the sparseness gene set to this value. One jump in performance occurred between generations 8 and 9 when the learning rule f_{11} altered from 2 (LTP/heterosynaptic LTD) to 4 (covariance), the weight initialisation t_{11} changed from 1 (uniform probability distribution) to 2 (constant), and the learning rate k_{11} decreased from 8.0 to 0.98. All these factors, as discussed for the pattern associator, would be expected to improve performance, with the decrease in learning rate ensuring that the weights did not reach the floor or ceiling set by the genes v_{11} and u_{11}. The increase in fitness from generation 3 to 4 was related to the change of the learning rule f_{11} from 3 (LTP with LTD depending on the existing synaptic weight) to 2 (LTP with heterosynaptic LTD), to an increase in the number of connections c_{11} from 71 to 97, and to a change of weight initialisation t_{11} from 3 (gaussian distribution) to 0 (constant value of 0). The increase in fitness from generation 4 to 5 was related to a change of the weight initialisation gene t_{11} from 0 to 1 (a uni-

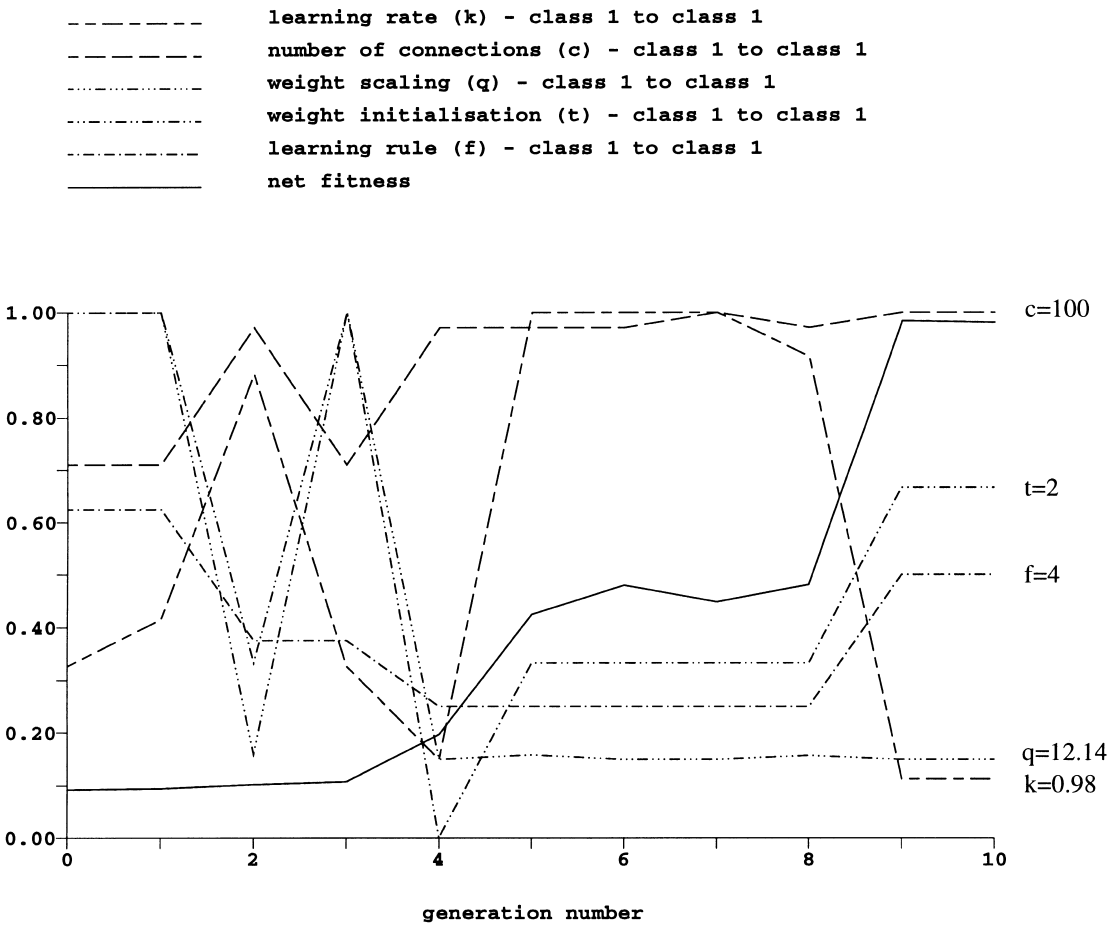

Fig. 8. Fitness of the best genotype in the autoassociation task as a function of the number of generations, together with the values of selected genes. The simulation is performed with the firing rates assigned binary values of 0 or 1 according to the neuronal activations,with the proportion of neurons firing set by the firing rate sparsity gene *a*.

Table 3
Performance of autoassociation networks with clipping of synaptic weights to be ≥ 0[a]

	Synaptic weight initialisation option for different values of t						
	0	1 ($q = 0.5$)	1 ($q = 5.0$)	1 ($q = 50.0$)	2 ($q = 0.5$)	2 ($q = 5.0$)	2 ($q = 50.0$)
Rule 1	0.158	0.152	0.091	0.025	0.154	0.151	0.155
Rule 2	0.240	0.315	0.163	0.008	0.420	0.994	0.994
Rule 3	0.109	0.109	0.108	0.001	0.110	0.108	0.001
Rule 4	0.645	0.745	0.135	0.008	0.924	0.992	0.992
Rule 5	0.102	0.100	0.089	0.025	0.096	0.152	0.147
Rule 6	0.000	0.007	0.007	0.007	0.000	0.000	0.000
Rule 7	0.000	0.000	0.007	0.007	0.000	0.000	0.000

[a] Conventions as in Table 1.

form random positive distribution which would lift the weights in general above the floor of 0), with the noise introduced by the random initialisation being made small relative to the signal by an increase in the learning rate k_{11}. The most successful networks (using synaptic weights clipped to be ≥ 0) typically had genes which specified for the autoassociative feedback connections a learning rule of 4 (covariance) or 2 (LTP with heterosynaptic LTD); and positive constant initial synaptic weights ($t_{11} = 2$), or if a weight initialisation with a uniform probability distribution ($t_{11} = 1$) was chosen then a high learning rate k_{11}, to minimise the synaptic initialisation noise introduced into the autoassociatively learned connections.

To help understand in full detail the actual gene selections found by the genetic algorithm in the simulations, we simulated the autoassociator with manually selected gene combinations for the recurrent connections from class 1 to class 1. In particular, we calculated the network fitnesses for particular combinations of learning rule f_{11}, weight initialisation option t_{11}, and weight scaling q_{11}. The results of some of these simulations are shown in Tables 3 and 4, both for the normal situation biologically and in most of the runs where the weights were clipped to be ≥ 0, and when they were allowed to become negative. (In all cases, the firing rates were ≥ 0.) It is shown in Table 3 that with the weights clipped to be ≥ 0, the best performance (0.99) was obtained with the learning rule f_{11} set to 2 (LTP plus heterosynaptic long-term depression), or to 4 (covariance), provided that the weights were initialised to a high enough positive constant value (for the simulations performed with $q_{11} = 5$ as indicated in the tables) that when decremented by the learning rule during learning the weights did not hit the floor of 0. (If the weights were not clipped to be ≥ 0 during learning, then both these rules were again the best, but weight initialisation could in addition be set to zero ($t_{11} = 0$) or any constant ($t_{11} = 2$), as shown in Table 4.) These results are in accord with what would be predicted (Rolls and Treves, 1998), and help to show why the genetic algorithm tended to produce gene combinations which included rule 2 or rule 4 in combination with weight initialisation to a positive constant sufficiently great to enable the synapses not to hit the floor of 0. (Without clipping of the synaptic weights to be ≥ 0 during learning, weight initialisation to uniform random values could produce good performance with rules 2 and 4 provided that the weights were scaled to low initial values ($q_{11} = 0.5$) so that only low amounts of noise were introduced. The learning rate was in all cases in the tables set to 1.)

Table 4
Performance of autoassociation networks without clipping of synaptic weights[a]

	Synaptic weight initialisation option for different values of t						
	0	1 ($q = 0.5$)	1 ($q = 5.0$)	1 ($q = 50.0$)	2 ($q = 0.5$)	2 ($q = 5.0$)	2 ($q = 50.0$)
Rule 1	0.145	0.150	0.092	0.025	0.149	0.150	0.147
Rule 2	0.991	0.990	0.146	0.008	0.991	0.992	0.993
Rule 3	0.109	0.108	0.107	0.001	0.109	0.108	0.001
Rule 4	0.992	0.990	0.153	0.008	0.992	0.996	0.991
Rule 5	0.154	0.150	0.092	0.026	0.148	0.152	0.148
Rule 6	0.000	0.007	0.008	0.007	0.000	0.000	0.000
Rule 7	0.000	0.008	0.007	0.008	0.000	0.000	0.000

[a] Conventions as in Table 1.

The Hebbian associative rule without weight decrementing $f_{11} = 1$ was not chosen for the fit genotypes given the maximum fitness obtainable without weight decrementing to remove the positive correlation introduced by firing rates ≥ 0 shown in Table 3. The associative learning rule 3 envisaged to be useful in competitive networks was not selected for the successful genotypes in autoassociation, as it produced only poor maximum fitness values as shown in Tables 3 and 4.

Some simulation runs were performed in which instead of using a fixed value of the gene a_1 to set the sparseness of the output firing, a sigmoid activation function with genetically specified threshold α_1 and slope β_1 were allowed to evolve. The part of the solution space containing good genotypes was very small, because with the iterative processing performed by an autoassociation net, any inaccuracy of the threshold would be greatly multiplied to produce cumulative error in the level of firing of the output neurons. For this reason, successful solutions were rarely found by the simulations. We note that in the real brain, feedback inhibition using inhibitory interneurons is used to help control the overall firing of a population of neurons, and believe that when these neurons are introduced into such simulations in future work, solutions will be found regularly with the sigmoid activation function.

4.6. Competitive networks

The computational problem set was to learn to place into separate categories 20 binary pattern vectors each 100 elements long. First, five exemplar patterns were randomly generated with 50 1s and 50 0s. Then, for each exemplar, three further patterns were created by mutating 20 of the bits from 0 to 1 or 1 to 0. This gave a total of 20 patterns that could be placed in five similarity categories. There were two classes of neuron. Class 1 was the input class, consisted of 100 neurons, and was set to fire according to which input pattern was being presented. Class 2 was the output class. Connections were possible from class 1 to 2, from class 1 to 1, from class 2 to 1, and from class 2 to 2. The network shown in Fig. 3, if trained with learning rule 3 and given a powerful and widespread lateral in-

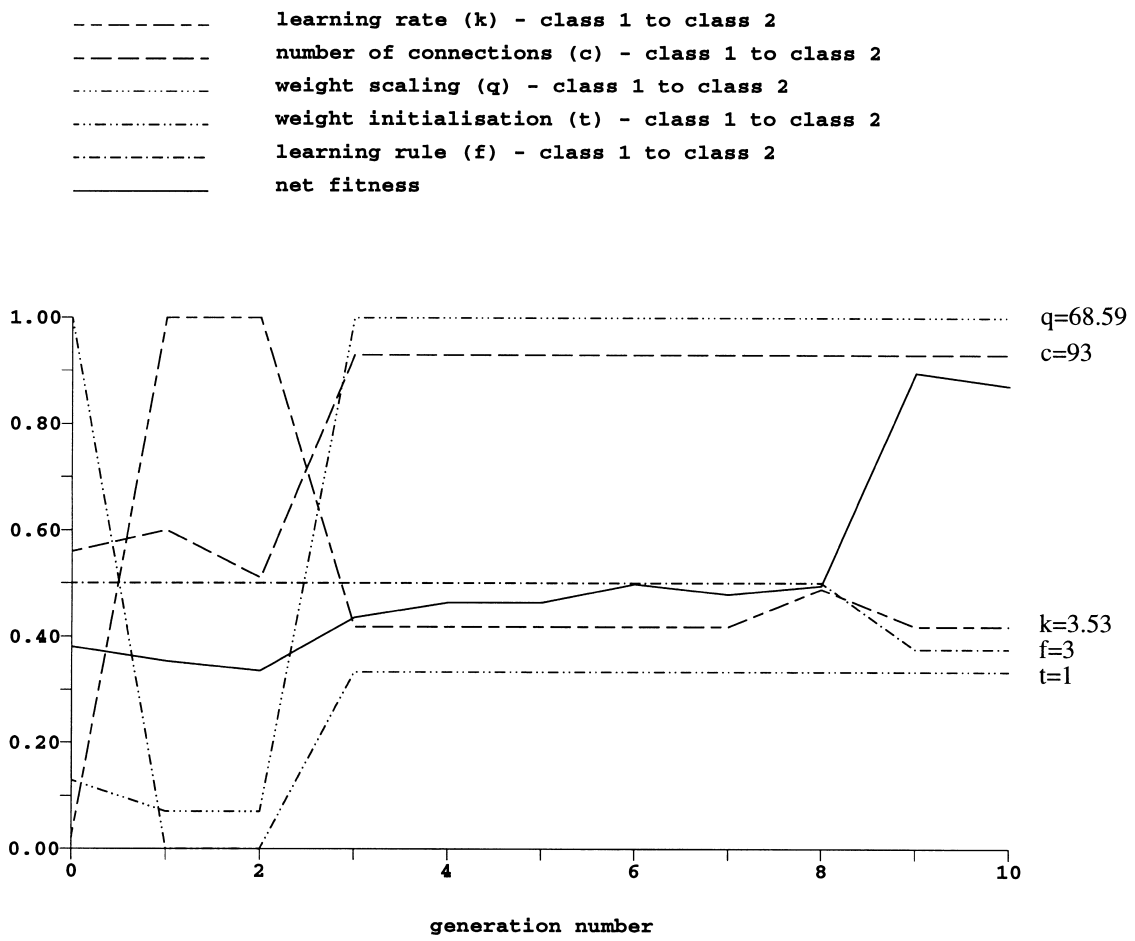

Fig. 9. Fitness of the best genotype in the competitive network task as a function of the number of generations, together with the values of selected genes. Evolution led to the choice of learning rule 3 for the best genotype.

hibition scheme can solve this problem (Rolls and Treves, 1998). During learning, each 100-element input pattern was presented to force class 1 neurons to fire, then the firing rates of the class 2 neurons were calculated, and then synaptic modification was allowed to occur by whichever learning rules were selected for that genotype. However, during learning and testing, the firing rates of neurons in class 2 were initially set to 0 corresponding to no firing before the cue was applied, and this had the effect of eliminating the effect of connections from class 2 to classes 1 and 2. The fitness measure used assessed how well the competitive net was able to classify the 20 patterns into their respective five categories. A suitable measure of how closely two output firing rate vectors are aligned with each other is the cosine of the angle between them; this measure ranges from 0 (for orthogonal vectors) to 1 (for parallel vectors) (Rolls and Treves, 1998). Hence, the fitness measure was set equal to the average of the cosine of the angle between the output firing rate vectors for patterns from the same category, minus the average of the cosine of the angle between the output firing rate vectors for patterns from different categories (clipped at zero). Each network was trained for 20 epochs, where each epoch consisted of training with every input pattern chosen in sequence.

The values of the fitness measure and selected genes for a typical run of the simulation are shown in Fig. 9. The genes shown are key connection properties from class 1 to class 2, and are: learning rate k_{21}, number of connections c_{21}, weight scaling q_{21}, weight initialisation option t_{21}, and the learning rule f_{21}. (In Fig. 9 the genes are normalised in a similar manner to Fig. 6, with the values of the learning rate k_{21} and weight scaling q_{21} normalised by 8.44 and 68.64, respectively.) The fitness measure is the fitness of the best genotype in the population. This run is for a fixed value of the sparseness in class 2, which was set to 0.2. It is shown that evolution to produce a genotype that specifies networks with good performance occurred over a number of generations. Runs typically converged to use learning rules 2 or 3, although learning rule 3 appeared to offer the best performance. Learning rule 3 involves LTP, plus LTD proportional to the existing value of the synaptic weight. This is the only rule which actually encourages the synaptic weight vectors of the different output (class 2) neurons to have a similar and limited total value. Use of a learning rule such as this (or explicit normalization of the length of the synaptic weight vector of each neuron), is advantageous in competitive networks, because it prevents any one neuron from learning to respond to all the input patterns (Rolls and Treves, 1998). The fact that this learning rule was never chosen by the successful genotypes in the associative nets, but was generally chosen for the competitive nets, is evidence that the genetic evolution was leading to appropriate architectures for different computational problems. Learning rule 0 (no learning) was never chosen by the successful genotypes, and in addition we verified that good solutions to the problem set could not be produced without learning. With other more regular patterns the actual values of the synaptic weight vectors produced after learning in the networks specified by the successful genotypes confirmed that conventional competitive networks operating as described by Rolls and Treves (1998) were being produced in the simulations. Another characteristic of the successful genotypes was that there appeared to be less selection pressure on c_{21} the number of connections to a class 2 from a class 1 neuron, with some highly fit nets having relatively low values for c_{21}. Such diluted connectivity can be useful in competitive networks, because it helps to ensure that different input patterns activate different output neurons (Rolls and Treves, 1998). Low values were never chosen for the best genotypes in the associative networks, where full connectivity is advantageous for full capacity. The fact that partial connectivity was often chosen for the competitive networks accounts for the fact that these nets could allow the synaptic weight initialisation to have a constant positive value. (With full connectivity, the weight initialisation has to produce different weight vectors for different neurons, so that the first time a set of patterns is applied, each pattern produces different firing of the output neurons.)

5. Discussion

The simulations described here show that the overall conception of using genes which control processes of the type described can, when combinations are searched through using genetic evolution utilising a single performance measure, lead to the specification of neural networks that can solve different computational problems, each of which requires a different functional architecture. The actual genes hypothesized were based on a comparison of the functional neuronal network architectures of many mammalian brain regions (see e.g. Rolls and Treves, 1998 for examples of these), and were well able to specify the architectures described here. However, the research described is intended as a foundation for further exploration of exactly which genes are used biologically to specify real architectures in the mammalian brain which perform particular computations. We note in this context that it may be the case that exhaustive analysis of the genetic specification of an invertebrate with a small number of neurons does not reveal the principles involved in building complex nervous systems and networks of the type addressed here. The genetic specification for simple nervous systems could be

considerably different, with much more genetic specification of the properties of particular synapses to produce a particular more fixed network in terms both of which identified neuron is connected to which, and what the value of the synaptic connection strength between each neuron is. In contrast, in the approach taken here, there are not identifiable particular neurons with particular connections to other identifiable neurons, but instead a specification of the general statistics of the connectivity, with large numbers of neurons connecting according to general rules, and the performance of the network being greatly influenced by learning of the appropriate values of the connection strengths between neurons based on the co-activity of neurons, and the nature of the modifiable synaptic connection between them.

The particular hypotheses about the genes that could specify the functional architecture of simple networks in the brain were shown by the work described here to be sufficient to specify some different neuronal network architectures for use in a system that builds computationally useful architectures using a gene selection process based on fitness. The genes hypothesized may be taken as a guide to the types of genes that could be used to specify the functional architecture of real biological nervous systems, but the gene specification postulated and simulated can be simply revised based on empirical discoveries about the real genes. One choice made here was to specify the genes with respect to the receiving neuron. The reason for this, rather than specification with respect to the sending neuron, was that many of the controlling properties of the network architecture and performance are in the post-synaptic neuron. Examples of these properties include the type of post-synaptic receptor (e.g., NMDA receptors to specify associatively modifiable connections), and the cable properties of the postsynaptic neuron. The specification of the receiving neuron does include the appropriate information about the sending neuron, in that for example gene c_{lm} does specify (by identifier m) the identity of the sending population. This implies that when a neuron makes an output connection (through its synapses), the neuron class of the sending neuron is available (as a chemical recognition identifier) at the synaptic terminal. However, in principle, in terms of actually building a network, it would be possible to have the genetic specifiers listed as being properties of the sending neuron. Another aspect of the gene specifiers hypothesized here is that there was no gene for how many output connections a class of neurons should make to another class. One reason for this is that the networks can be built numerically without such a specifier, as it is unnecessary if the total numbers of neurons of two classes is specified, and the number of connections received by one class of neuron from the other is specified. However, the number of output connections made to another class of neuron might be specified in real biological systems, and neuron numbers might adjust to reflect this type of constraint, which is a possible way to think about cell death during development. If there is local output connectivity within a certain spatial range, this will also act as a factor that determines the number of output connections made by neurons, and indeed this does seem a possible gene in real biological systems. Indeed, in real systems the single gene specifying the region of connectivity (r_{lm}) here might be replaced by different spatial genes specifying the extent of the dendrites of the receiving neuron, and extent of the axonal arborization for the outputs.

We emphasise that a number of the genes specified here for the simulations might be replaced in real biological systems with several genes operating at a lower level, or in more detail, to implement the function proposed for each gene specified here. The reason for selecting the particular genes hypothesized here is that they serve as a guide to how in principle neuronal networks could be specified genetically in complex neural systems where the functional architecture is being selected for by genetic evolution. Specification of genes as operating at the level described in this paper provides a useful heuristic for investigation by simulation of how evolution may operate to produce neuronal networks with particular computational properties. Once the processes have been investigated and understood with genes operating at the level described here, subsequent investigations that use genes that are more and more biologically accurate and detailed would be facilitated and are envisaged. The approach allows continual updating of the genotype used in the simulations to help understand the implications of new discoveries in neurobiology.

We also emphasise that the way in which the genetics specifies the neural network is very different from that used in some previous approaches, in which there was a fixed architecture, and the genes were allowed to evolve the values of the synaptic weights (Nolfi et al., 1994; Floreano and Mondada, 1994; Lund and Parisi, 1996; Kuwana et al., 1996; Huber et al., 1996). The approach taken here in contrast is to allow the genes to specify the functional architecture, including which classes of neuron are connected to each other, and the synaptic learning rule involved in each such set of connections. The network is then built according to the architecture specified genetically, and tested to determine whether it can learn the appropriate values of the synaptic weights to solve the problem. The approach used here thus requires many fewer genes than would be required to specify actual synaptic weight values, and moreover allows the individual phenotype to adapt itself by learning in the particular environment

in which it finds itself. There is thus some adaptability within the lifetime of the phenotype, which is a great advantage. The adaptive changes which occur within the lifetime of the individual can therefore be used for such processes as using early visual experience to help shape the visual system; learning by pattern association which visual stimuli are actually associated with primary reinforcers; semantic learning; and even learning of particular languages, all of which are likely to take place optimally by specifying a general architecture, and then allowing learning by the individual in the actual environment it finds, which is the approach taken here.

We now consider the number of genes used in simulations of the type described here, remembering that biologically several genes might be required to specify at least some of the genes used in the simulations. The number of genes for each class of neuron in the simulations is currently $4+$ (the number of classes of neuron from which the class receives connections $\times 14$). In the brain, if there were on average five classes of neuron from which each class of neuron received connections, this would result in $4 + 5 \times 14 = 74$ genes per class of neuron. If there were on average five classes of neuron per network, this would yield 370 genes per network. 100 such networks would require 37,000 genes, though in practice fewer might be needed, as some parameters might be assigned to be the same for many classes of neuron in different networks. For example, the overall functional architecture of the cerebral neocortex appears to follow the same general design for different architectonic areas (Rolls and Treves, 1998; Braitenberg and Schuz, 1991; Peters and Jones, 1984; Somogyi et al., 1998; Douglas and Martin, 1990), so much of the neocortex may use one generic network specification, with a few genes used to tweak the parameters for each different architectonic area. (An extension of the idea presented in this paper would allow such a generic specification of the architecture of an area of the cerebral neocortex, and then allow genes to connect different areas with the same generic though locally tweaked architecture.) A factor working in the opposite direction is the fact that a single gene taking an integer or real value was used to specify some of the parameters in the simulation, whereas biologically several genes might be needed to represent such a wide range of parameter values with low, though sufficient, precision as described in Section 4.3.

Although with 100 individuals per generation, and approximately 64 genes for each genotype (for networks with two classes of genes, each connected to each other as well as to themselves) each with several at least possible values, the search might be predicted to be long, in practice the genetic search was often quite short. Part of the reason for this is that some genes made a crucial difference, for example the gene specifying the learning rule, and were selected early on in evolution, usually being present in at least some of the genotypes in the initial population. Because these genes had such a dramatic effect on fitness, they were almost always selected for the next generation, and successive generations then had to search the smaller space of the remaining genes only some of which had a substantial impact on performance. This effect can be seen in Figs. 6–9, which show for the different networks examples in the variation in the value of different genes as a function of the generation number. Some genes, such at that specifying the learning rule, do indeed often settle down early, and changes in these genes can sometimes be seen to be reflected in jumps in the fitness measure. For other genes, where the impact is less or zero, the values of the genes can fluctuate throughout evolution, because they are not under selection pressure, in that they do not contribute greatly to fitness.

Although therefore in some senses the actual evolution investigated here takes place rapidly and apparently quite easily, this is part of the point of this paper, that is to demonstrate that the particular genes identified as those which might allow the networks to be found in different brain areas to be built, and implemented in the simulations, can actually build these networks, and moreover can do so efficiently when using genetic search. However, the networks tested here are single layer networks, and multilayer networks, with correspondingly more parameter combinations to be explored, will doubtless take longer to evolve in future simulations. Genetic search of the parameter space should then become even more evident as a good procedure, because of its power in searching high dimensional spaces with many local optima and in which the parameters combine non-linearly; because of its use of a single measure of performance; and because of its ability to reuse genetic codes still present in a diverse population but originally developed for a different purpose. The intention however here is to specify the principles involved. (In fact, we were on a number of occasions surprised by the power of the genetic search when it sometimes found unanticipated solutions to problems by combining particular genes originally provided in the genome by us for quite different anticipated uses.) Of course, there are many more details of the networks that could be explored by the existing gene specification, including inhibitory interneurons. Simulations of this type will not only enable the evolution and development of more complex multilayer networks to be explored, but will also enable the utility of the specification of different parameters by different genes to be explored, including details of where inputs end on dendrites, dendritic biophysical properties, and thus eventually networks that

operate to simulate also the dynamics of the operation of real neural networks in the brain (Treves, 1993; Rolls and Treves, 1998; Battaglia and Treves, 1998; Panzeri et al., 2000).

Acknowledgements

This research was supported by the Medical Research Council, grant PG8513790, by the MRC Interdisciplinary Research Centre for Cognitive Neuroscience, and by the Human Frontier Science Program.

References

Ackley, D.H., 1987. A Connectionist Machine for Genetic Hill-Climbing. Kluwer Academic Publishers, Dodrecht.

Battaglia, F., Treves, A., 1998. Rapid stable retrieval in high-capacity realistic associative memories. Neural Computation 10, 431–450.

Bienenstock, E.L., Cooper, L.N., Munro, P.W., 1982. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. Journal of Neuroscience 2, 32–48.

Braitenberg, V., Schuz, A., 1991. Anatomy of the Cortex. Springer-Verlag, Berlin.

Buhl, E.H., Halasy, K., Somogyi, P., 1994. Diverse sources of hippocampal unitary inhibitory postsynaptic potentials and the number of synaptic release sites. Nature 368, 823–828.

Buonomano, D.V., Merzenich, M.M., 1998. Cortical plasticity: from synapses to maps. Annual Review of Neuroscience 21, 149–186.

Douglas, R., Martin, K.A.C., 1990. Neocortex. In: Shepherd, G. (Ed.), The Synaptic Organisation of the Brain, 3rd ed. Oxford University Press, New York, Oxford, pp. 389–438 (Chapter 12).

Floreano, D., Mondada, F., 1994. Automatic creation of an autonomous agent: genetic evolution of a neural-network driven robot. In: Cliff, D., Husbands, P., Meyer, J., Wilson, S. (Eds.), From Animals to Animats 3. Proceedings of the Third International Conference on Simulation of Adaptive Behaviour. MIT Press, Cambridge, MA.

Földiák, P., 1991. Learning invariance from transformation sequences. Neural Computation 3, 194–200.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading.

Gracias, N., Pereira, H., Lima, J.A., Rosa, A., 1996. Gaia: an artificial life environment for ecological systems simulation. In: Langton, C.G., Shimohara, K. (Eds.), Artificial Life V. Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems. MIT Press, Cambridge, MA.

Hertz, J., Krogh, A., Palmer, R.G., 1990. Introduction to the Theory of Neural Computation, Santa Fe Institute. Addison-Wesley, Reading, MA.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press.

Hoshino, T., Tsuchida, M., 1996. Manifestation of neutral genes in evolving robot navigation. In: Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems. MIT Press, Cambridge, MA.

Huber, S.A., Mallot, H.A., Bülthoff, H.H., 1996. Evolution of the sensorimotor control in an autonomous agent. In: Maes, P., Mataric, M.J., Meyer, J., Pollack, J., Wilson, S.W. (Eds.), From Animals to Animats 4, Proceedings of the Fourth International Conference on Simulation of Adaptive Behaviour. MIT Press, Cambridge, MA.

Husbands, P., Smith, T., Jakobi, N., O'Shea, M., 1998. Better living through chemistry: evolving gasnets for robot control. Connection Science 10 (4), 1–40.

Koch, C., 1999. Biophysics of Computation. Oxford University Press, Oxford.

Kohonen, T., 1995. Self-Organizing Maps. Springer-Verlag, Berlin.

Kuwana, Y., Shimoyama, I., Sayama, Y., Miura, H., 1996. A robot that behaves like a silkworm moth in the pheromone stream. In: Langton, C.G., Shimohara, K. (Eds.), Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems. MIT Press, Cambridge, MA.

Lund, H.H., Parisi, D., 1996. Generalist and specialist behaviour due to individual energy extracting abilities. In: Langton, C.G., Shimohara, K. (Eds.), Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems. MIT Press, Cambridge, MA.

Maynard Smith, J., Szathmary, E., 1999. The Origins of Life. Oxford University Press, Oxford.

McClelland, J.L., McNaughton, B.L., O'Reilly, R.C., 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. Psychological Review 102, 419–457.

Nolfi, S., Elman, J., Parisi, D., 1994. Learning and evolution in neural networks. Adaptive Behaviour 3, 5–28.

Panzeri, S., Rolls, E.T., Battaglia, F., Lavis, R., 2000. Speed of information retrieval in multilayer networks of integrate-and-fire neurons (submitted).

Peters, A., Jones, E.G. (Eds.), 1984. Cerebral Cortex, vol. 1. Plenum Press, New York.

Rolls, E.T., 1992. Neurophysiological mechanisms underlying face processing within and beyond the temporal cortical areas. Philosophical Transactions of the Royal Society London [B] 335, 11–21.

Rolls, E.T., 1999. The Brain and Emotion. Oxford University Press, Oxford.

Rolls, E.T., Milward, T., 2000. A model of invariant object recognition in the visual system: learning rules, activation functions, lateral inhibition and information-based performance measures. Neural Compulation (in press).

Rolls, E.T., Stringer, S.M., 2000. Invariant object recognition in the visual system with error correction and temporal difference learning (submitted for publication).

Rolls, E.T., Treves, A., 1998. Neural Networks and Brain Function. Oxford University Press, Oxford.

Shepherd, G.M., 1990. The Synpatic Organisation of the Brain, 3rd ed. Oxford University Press, Oxford.

Somogyi, P., Tamas, G., Lujan, R., Buhl, E.H., 1998. Salient features of synaptic organisation in the cerebral cortex. Brain Research Reviews 26, 113–135.

Treves, A., 1993. Mean-field analysis of neuronal spike dynamics. Network 4, 259–284.

Ullman, S., 1996. High-Level Vision. MIT Press, Cambridge, MA.

Vonk, E., Jain, L.C., Johnson, R.P., 1997. Automatic Generation of Neural Network Architecture Using Evolutionary Computation. World Scientific, Singapore.

Wallis, G., Rolls, E.T., 1997. A model of invariant object recognition in the visual system. Progress in Neurobiology 51, 167–194.

Wang, J.H., Ko, G.Y., Kelly, P.T., 1997. Cellular and molecular bases of memory: synaptic and neuronal plasticity. Journal of Clinical Neurophysiology 14, 264–293.