PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY B

Phil. Trans. R. Soc. B (2007) **362**, 339–353
doi:10.1098/rstb.2006.1962
Published online 8 January 2007

Neural networks for perceptual processing: from simulation tools to theories

Kevin Gurney*

Adaptive Behaviour Research Group, Department of Psychology, University of Sheffield, Sheffield S10 2TP, UK

Neural networks are modelling tools that are, in principle, able to capture the input–output behaviour of arbitrary systems that may include the dynamics of animal populations or brain circuits. While a neural network model is useful if it captures phenomenologically the behaviour of the target system in this way, its utility is amplified if key mechanisms of the model can be discovered, and identified with those of the underlying system. In this review, we first describe, at a fairly high level with minimal mathematics, some of the tools used in constructing neural network models. We then go on to discuss the implications of network models for our understanding of the system they are supposed to describe, paying special attention to those models that deal with neural circuits and brain systems. We propose that neural nets are useful for brain modelling if they are viewed in a wider computational framework originally devised by Marr. Here, neural networks are viewed as an intermediate mechanistic abstraction between 'algorithm' and 'implementation', which can provide insights into biological neural representations and their putative supporting architectures.

Keywords: neural networks; connectionism; perception; meta-theory; methodology; modelling

1. INTRODUCTION

This paper has two main aims. First, to give an introduction to some of the techniques—the 'nuts-and-bolts' as it were—of neural networks deployed by the authors in this issue of the Journal. Our intention is to emphasize conceptual principles and their associated terminology, and to do this wherever possible without recourse to detailed mathematical descriptions. However, the term 'neural network' has taken on a multitude of meanings over the last couple of decades, depending on its methodological and scientific context. The second aim, therefore, given that the application of the techniques described in this issue may appear rather diverse, is to supply some meta-theoretical landmarks to help understand the significance of the ensuing results.

In general terms, neural networks are tools for building models of systems that are characterized by datasets which often (but not always) are derived by sampling a system input–output behaviour. While a neural network model is of some utility if it mimics the behaviour of the target system, it is far more useful if key mechanisms underlying the model functionality can be unearthed, and identified with those of the underlying system. In other words, the modeller can 'break into' the model, viewed initially as an input–output 'black box', and find internal representations, variable relationships and structures which may correspond with the underlying target system. This target system may be entirely non-biological (e.g. stock market prices) or be of biological origin, but have nothing to do with brains (e.g. ecologically driven patterns of population dynamics). In these instances, we can ask whether the internal network machinations are informative of specific relationships between system inputs and outputs, and any internal variables. However, the mechanistic elements of a network have names which are evocative of processing in the animal brain; there is talk of 'artificial neurons', their interconnection strengths and 'learning'. If, therefore, a neural network is a model of parts of the brain, the problem of interpretation of internal mechanisms is particularly acute. For, if these mechanisms are based on those in the brain, is it the case that they reflect genuine, biological neural mechanisms? These and related questions are explored in the second half of the paper.

2. NEURAL NETWORK PRINCIPLES

This section gives a high-level view of some of the principles and techniques used in the papers in this issue of the Journal. A more comprehensive treatment at this level can be found in Gurney (1997), while the books by Haykin (1999) and Bishop (1996) take a more mathematical approach.

We start with a pragmatic, working definition of a neural network: a neural network is an interconnected assembly of simple processing elements, *units* or *nodes* whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the inter-unit connection strengths, or *weights*, obtained by a process of adaptation to, or *learning* from, a set of *training patterns*. The rest of this section is devoted to unwrapping these terms with special emphasis on those networks that appear in subsequent articles in this issue.

*k.gurney@shef.ac.uk

One contribution of 15 to a Theme Issue 'The use of artificial neural networks to study perception in animals'.

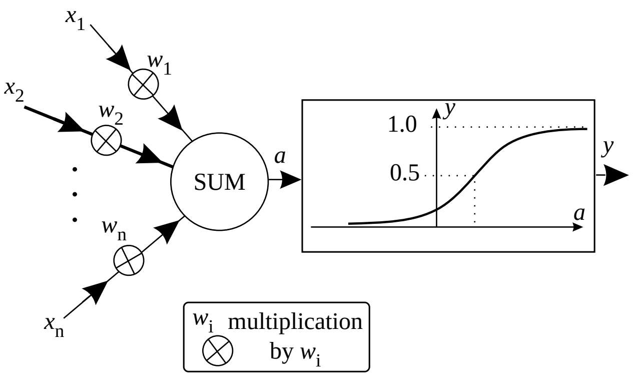


Figure 1. Simple model neuron. A weighted sum of inputs is compressed by a logistic sigmoid.

(a) *Artificial neurons*

Figure 1 is a graphical description of a typical neural network node. Input signals $x_1, x_2, \ldots, x_n$ are combined to form an output y via an *activation* variable a. The latter is formed by taking a weighted sum of each input x_i, i.e.

$$a = \sum_i w_i x_i. \tag{2.1}$$

The weights w_i may be positive or negative. The activation is then usually transformed by some kind of squashing function which limits the output y to a specified range (usually the interval [0,1]) and introduces a nonlinearity; this latter feature proves to be crucial in endowing neural nets with their powerful functionality (see §2*b*). In the figure, the squashing function has been chosen to be the logistic sigmoid

$$y = \frac{1}{1 + \exp(-(a - \theta))}, \tag{2.2}$$

although other, similar functions are occasionally used. The constant θ defines the point at which y takes its mid-point value. Moreover, it is the point where the function is changing most rapidly and is therefore the value of the activation at which the node is most sensitive to small changes in the inputs. The negative of θ is therefore sometimes referred to as the *bias*. Note that y approaches 0 and 1 asymptotically as the activation decreases and increases, respectively (so, y is never equal to 0 or 1, but may be made as close to these as we please).

The basic node described above has a long lineage. The first artificial neural node was the threshold logic unit (TLU) introduced by McCulloch & Pitts (1943). This was also a two-stage device with the first stage given by equation (2.1) but with the output nonlinearity defined by a discontinuous step function, rather than the smooth ramp described by equation (2.2). Thus, the output of the TLU had only two values, 0 or 1, depending on whether the activation was less than or greater than the threshold θ, respectively. A more complex node, the *Perceptron*, was introduced by Rosenblatt (1958) which, retained the Boolean (0,1) output of the TLU, but allowed pre-processing of Boolean input variables with arbitrary functions (so-called 'association units') whose outputs then formed the variables x_i in equation (2.1). The TLU is therefore a special case of the Perceptron when 'association units' just pass a single input through to each weight.

The neurobiological inspiration for the structure of figure 1 is as follows. The input x_i corresponds to the presynaptic input on afferent i, while the weight w_i encapsulates the corresponding synaptic strength. The product $w_i x_i$ is akin to the postsynaptic potential (PSP) which is inhibitory/excitatory according to whether w_i is negative/positive. The integration of PSPs over the dendritic arbour and soma is represented by simple arithmetic addition, and the quantity a corresponds to the somatic membrane potential. This is then transformed by the squashing function to give a firing rate y. Clearly, some of these correspondences are, at best, merely qualitative analogues. The issue of realism is revisited in the second half of the paper.

(b) *Feedforward networks and classification*

A ubiquitous problem in perception is that of classification or pattern recognition. As an example, consider the problem of identifying letters of the alphabet. Humans are able to recognize letters in many sizes, orientations and fonts (including handwritten variations) with ease. Any individual person can never see all possible letter variants, but, instead, will learn idealized letter shapes from a very small set of possibilities (usually a plain font in childrens' reading books). This latter point demonstrates that *generalization* is a key component in the classification process, that is, the ability to generalize knowledge of specific pattern exemplars to a wide variety of related cases.

Based on this example, we now formalize the general problem of classification as follows. Given an arbitrary sensory input pattern drawn from some universal set of patterns, is it possible to place that pattern in its appropriate class or category, where there are generally many fewer classes than the patterns themselves? Further, we suppose that we do not have an exhaustive list of the entire universe of patterns; rather, we only have immediate access to some subset of patterns *P*, and knowledge of the category that each member of *P* belongs to. By way of terminology *P*, is the *training set* referred to in the motivational definition of a neural network at the start of this section. The problem is to construct an input–output model, based on this limited knowledge, which will generalize so that, if it is presented with a pattern not in *P*, it will elicit the correct classification for that pattern. Note that a 'model' which simply classifies *P*, but does not generalize, is easy to construct but of no real interest—it is just a lookup table of pattern–class pairings. We will return to the relationship between neural processing and generalization later (see §2*d*). In the meantime, we will look at how the classification problem may be solved in principle by a neural network.

Figure 2 shows a *feedforward network* which consists of a layered structure with information flowing from the inputs, at the bottom of the diagram, to the outputs at the top. The inputs have no functionality as such, but are simply points which receive pattern information and distribute this information to the first layer of neural nodes *per se* (of the type described previously).

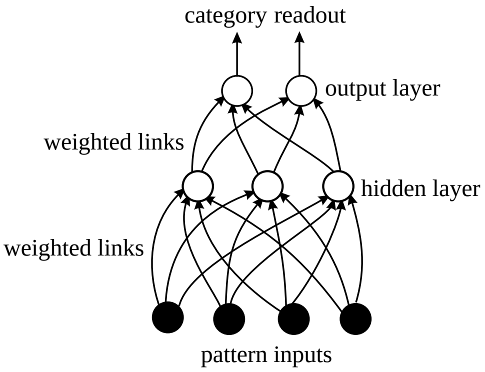


Figure 2. Simple feedforward neural network. The hidden and output layers are comprised of nodes described in figure 1.

In the example, there are four inputs and so all patterns for classification would have been defined by a list of four numbers. In more formal analyses, these lists of numbers are properly referred to as *vectors* with numeric *components*, and we sometimes speak of *pattern vectors*. This first layer of functional nodes is sometimes referred to as a *hidden layer*, since we are not supposed to inspect or control the output values on these nodes (y in equation (2.2)) during the process of setting the network weights, that is, during *training* or *learning* (see §2*c*). The outputs of the hidden layer are subsequently processed by an *output layer* which is used to read-out the category in which the input pattern is placed. There are several ways of doing this, depending on the way information is represented in the network (see §2*e*). We will refer to a network of the kind shown in figure 2 as a two-layer network, since it contains two layers of processing nodes. Some authors include the input layer in the layer count so that the network in figure 2 would constitute a three-layer net. The final point to make here is that networks of the kind shown in figure 2 are sometimes called *multi-layer Perceptrons* (*MLPs*), in deference to the important role played by Rosenblatt's original Perceptron in shaping the theory of neural network learning (Minsky & Papert 1969).

Note that processing by any particular node can be performed independently of that in any other. Thus, processing could, in principle, be performed in parallel if we had the necessary hardware resources to assign to each node. In spite of this, most networks find their implementation in software simulation in a conventional computer in which each node has to be visited serially to compute its output.

There is a mathematical framework which is particularly useful for describing quantitatively the process of classification in networks. It is based on the notion that patterns reside in some *pattern space* and is evocative of geometric analogies that enable the problem to be visualized. Suppose, for example, we have patterns belonging to two classes, A and B. If each pattern was defined by only two numerical components, then it could be represented quantitatively as a point in Cartesian axes as shown in figure 3*b*. If, in fact, each pattern is a vector with $n>2$ components, figure 3 is just a cartoon schematic which is simply illustrative of the case in n-dimensions. In figure 3*a*, the patterns are shown as being separated by a straight line. In three dimensions, this situation implies a plane, and in n-dimensions ($n>3$) a hyperplane. In all these cases, we say that the patterns are *linearly separable*, and the straight line is schematically indicative of this.

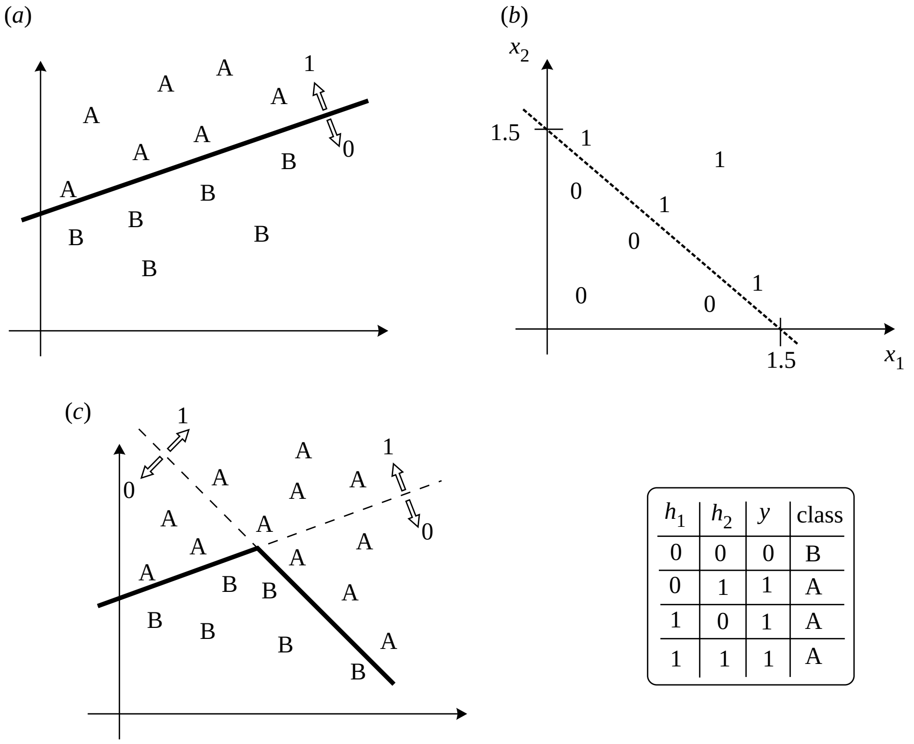
Suppose we have a single artificial neuron with n-inputs, then it could attempt to solve the classification problem in figure 3*a* by indicating output values of 1,0 for classes A,B, respectively. This could occur exactly if the node was a TLU, and approximately using a node of the form shown in figure 1 (since, in this case, the output approaches 0 and 1 asymptotically). It may be shown that linearly separable problems can indeed be solved by a single artificial neuron: a result which follows from the linearity of signal combination in equation (2.1). To see this, consider the two-input case and assume, for simplicity, a TLU. The critical condition that defines classification occurs when the activation a equals the threshold θ, since small changes in a around this value cause the node to switch its output between 0 and 1. Putting $a=\theta$, gives $w_1x_1+w_2x_2=\theta$. This may be solved for x_2 in terms of x_1 to give

$$x_2=-\left(\frac{w_1}{w_2}\right)x_1+\left(\frac{\theta}{w_2}\right),$$

which is a straight line with slope $-w_1/w_2$ and intercept θ/w_2. Now put, for example, $w_1=w_2=1$ and $\theta=1.5$. This defines a line $x_2=-x_1+1.5$ as shown in figure 3*b*. Here, pair of values (x_1, x_2) defining points on the same side of the line as the origin gives TLU outputs of 0, while values defining points on the other side of this line give TLU outputs of 1. In particular, the Boolean inputs (1,1) give an output of 1, while the other three Boolean input pairs give an output of 0 (in this case, the TLU is acting as a classical logic AND gate).

Figure 3*c* shows a harder problem in pattern space which may only be solved by a *decision line* (in n-dimensions, a *decision surface*) consisting of two straight, but non-collinear segments (shown by the solid line in the figure). The dotted lines show the extension of the line segments which make each of them a continuous straight line throughout pattern space (similar to the line in figure 3*a*). Each extended straight line then defines a linearly separable problem which may be solved by nodes with outputs, h_1 and h_2. While each of these separate classifications mixes patterns A and B together, the table in the figure shows how the original classification problem may now be solved by taking suitable combinations of h_1 and h_2; that is, class B is signalled if and only if both h_1 and h_2 are 0. This two-component classification problem is linearly separable and may be solved with a single two-input neural node. The original A/B classification problem has therefore been decomposed into two stages which may be solved by a two-layer net with two hidden nodes (yielding h_1 and h_2) and a single output node.

As the classification becomes more complex, we may now ask the following question: is it possible to solve an arbitrary classification problem with a two-layer net—or do we need to resort to more complex structures? In other words, in an analogous way to the example mentioned above, can we describe the



h_1	h_2	y	class
0	0	0	B
0	1	1	A
1	0	1	A
1	1	1	A

Figure 3. Geometric view of pattern classification; binary decision between classes A and B. (*a*) The classes are linearly separable. (*b*) Linear separability for two-input TLU (see text for details); the dotted line shows the decision line for the TLU. (*c*) The classes are not linearly separable but may be classified by a two-layer net with two hidden nodes.

decision surface of the problem in a piecewise linear way, solve the resulting decomposition using hidden units, and then combine their outputs in a linearly separable fashion which solves the original problem? A series of general results (Lippmann 1987; Wieland & Leighton 1987; Makhoul *et al.* 1989) has answered this in the affirmative so that, in principle, we never need a network with more than two layers. Further, the two-layer network capability may be extended to more than two classes. There is certainly one solution to such problems, since each class X may be used to define a binary classification between X and non-X classes at a single output node. It is worth noting that the nonlinearity of the squashing function is essential to these results, for it may be shown that a two-layer net with linear nodes reduces to a single layer net (which may only solve linearly separable problems).

While this result appears to confer enormous power on neural networks as models of classification, it is a double-edged sword as far as interpreting the network in biological terms is concerned. Thus, we are not forced to explore more biologically realistic architectures simply to achieve a desired functionality (these issues are discussed further in §3). Further, the result is only an existence proof; no method is supplied for identifying the detailed structure of the net (specifically how many hidden nodes one needs), how to find the weights (training procedure), or how to specify the size or nature of the training set P needed to find the solution. These problems are the subject of §2*c*,*d*.

(c) *Training feedforward networks*

In this section, we describe methods for determining the weights in a network, given that it should attempt to classify a training set P, each member p of which belongs to a known category t_p. The general idea is to iteratively supply members of the training set as input to the network, compare the network's output with the desired target performance t_p, and make adjustments to the weights to gradually bring the network's output ever closer to the target. Since we have access to the desired output for each pattern, this general paradigm is referred to as *supervised training* (or *supervised learning*).

We consider first the simple case of a 'network' consisting of a single node of the form described by equations (2.1) and (2.2). In this case, the output in response to pattern p is just a single scalar number y_p, as is also the target t_p. The latter will, in general, be 0 or 1 to flag one of the two possible classes. A quantitative way to compare the network's output and the target is to take the square of the difference $y_p - t_p$ which guarantees a positive measure. Thus, we define a pattern error $e_p = (y_p - t_p)^2$, and an overall *error* E to be the sum of the e_p over all members of the training set, i.e.

$$E = \sum_{p \in P} (y_p - t_p)^2. \tag{2.3}$$

The problem can now be restated as one of attempting to minimize E by making changes in the network weights, a process which is shown schematically in figure 4*a*. The dependence of E on the weights $E(w)$ is shown in cartoon form in figure 4*a* by the 'U'-shaped line. In general, there will be n-inputs ($n > 1$) each with its own weight w_i, that helps determine the error; that is, the error–weight function is described by a surface in $n+1$ dimensions. In spite of this, the schematic form in the figure gives a useful insight into the training process. Thus, starting with an initial weight set and error, training proceeds by altering the weights in a series of small steps until the ideal or desired weight set is found

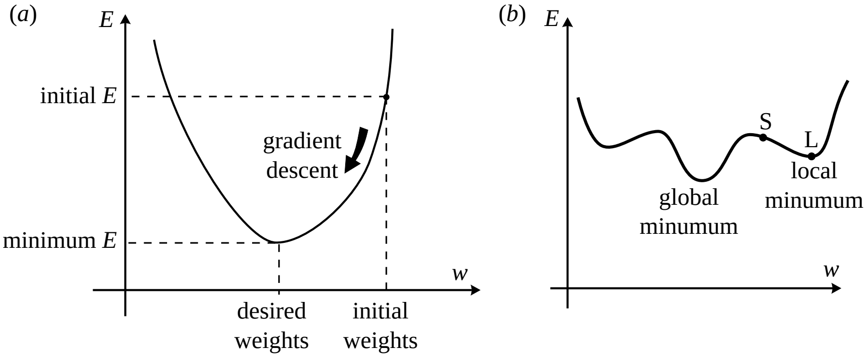


Figure 4. Gradient descent. (*a*) Schematic of a simple situation in error-weight space (no local minima). (*b*) Schematic of local minima.

which corresponds to the minimum error the net can achieve. The figure illustrates the point that, if we could always guarantee each step giving a 'downhill' move along the error–weight function $E(w)$, then we would eventually reach the target weight set. It is worth noting one proviso to this, however, which arises because we have assumed that the path from initial to final weights is continually decreasing. In general, the error surface may contain many 'bumps' and not just the global minimum as shown in figure 4*a*. In this case, moving downhill may result in the network becoming trapped in a local (rather than the desired global) minimum. This phenomenon is illustrated in figure 4*b* in which, starting at point S, training which induces movement down the error surface results in a suboptimal network at the local minimum L.

Notwithstanding these potential difficulties, the required process is one of *gradient descent* (moving 'downhill') and is enabled if we can compute the gradient or slope of the function $E(w)$ for each weight w_i. This is indeed possible using processes in the differential calculus. Using these techniques, it is possible to verify the intuitively plausible result that, since E has contributions from all patterns (see equation (2.3)), so too do the error gradients. In principle, therefore, we should evaluate all these contributions and sum them to find the true gradients for each weight. However, it transpires that it is possible to sidestep this rather compute-intensive process and train the network 'online' using estimates for the gradients found by calculating them for the current input pattern only. Under this regime, the required change in the ith weight Δw_i is given by

$$\Delta w_i = \alpha\sigma'(t_p - y_p)x_p^i, \tag{2.4}$$

where x_{p}^i is the component of pattern p on input i; σ' is the slope of the squashing function (with the current input) and α is a constant referred to as the *learning rate* which controls how large the weight updates are at each step. Training then consists of repeatedly supplying input patterns from P (together with associated targets) and updating the weights according to the *learning rule* in equation (2.4). The difference term $(t_p - y_p)$ is sometimes referred to as the 'delta', so that this particular method is referred to as the *delta rule*. Nominally, the iterative process continues until there is no appreciable change in the error or until all the outputs are a reasonable match with the target. However, we will see in §2*d* why it might be sensible to curtail training before this point.

The method described above for a single node may immediately be extended to several nodes forming a single-layer network by simply applying equation (2.4) to each node. Extending to the case of a two-layer network is, however, non-trivial. The problem is that an error on a particular output node could be due to the weights on that node being poorly trained, or to the hidden nodes which supply its inputs being poorly trained; where does the blame lie? This *credit assignment problem* is in fact solvable using regular techniques in the calculus and, because the solution involves computations at the hidden nodes which make use of error information originally at the output nodes, it is referred to as *error backpropagation* or simply *backpropagation*. Several papers presented in this issue make use of this learning technique. Backpropagation has been criticized on the grounds that it is not biologically plausible but, unless we are specifically interested in developmental processes (rather than simply the final, fully functional network), this is not a relevant issue. Further, recent evidence (Fitzsimonds *et al.* 1997) has shown that synaptic plasticity (the biological analogue of weight changes in neural nets) is indeed able to propagate in local neuronal circuits in ways not dissimilar to that envisaged in backpropagation.

The delta rule was first reported in a meeting abstract by Widrow & Hoff (1960); it was therefore occasionally referred to as the Widrow–Hoff rule in earlier work. For a derivation and overview of the delta rule, see Widrow & Stearns (1985). Backpropagation was discovered by Werbos (1974) who reported it in his Ph.D. thesis. It was later rediscovered by Parker (1982) but this version languished in a technical report that was not widely circulated. It was discovered again and made popular by Rumelhart *et al.* (1986) in their seminal book *Parallel distributed processing*. Many variants and improvements have since been made to the backpropagation algorithm since these first formulations (for a review, see Tveter 1996).

What happens if we do not have access to individual target information for each output node, but simply have, instead, information as to whether the net has performed 'well' or 'badly'; that is, if we have a single scalar reinforcement signal which flags 'reward' if the network performs better than expected in approximating the target, and 'penalty' if the net performs worse than expected? The key to using this approach is to allow the network to 'explore' the space of possible solutions to the problem by adding noise to the node activation. Training then proceeds by increasing the magnitude of weights during learning trials that result in 'reward', and decreasing their magnitude (and eventually changing their sign) during 'penalty' trials. In this way, the network learns by 'trial and error'. Andrew Barto and co-workers have developed a range of algorithms for training both single nodes (Barto 1985; Barto & Anandan 1985) and networks (Williams 1987). More complex variants make use of *temporal difference* learning (Sutton 1988) and *actor–critic* models (Barto *et al.* 1983). All these methods fall within the remit of the general theory of reinforcement learning (Sutton & Barto 1998). These methods are not intrinsically supervised—that is, the reward signal is not necessarily derived from knowledge of what the 'right answer' should be, although reward signals *may* be derived from error functions of the form in equation (2.3).

Neural network learning is an example of the class of problems known as 'function optimization' or 'parameter search'. As such, neural net learning may take advantage of any of the general purpose techniques developed therein. In particular, those methods which do not suffer from becoming trapped in local minima are especially attractive. These include simulated annealing and evolutionary computing methods such as genetic algorithms; for a review of some of these techniques, see Shang & Wah (1996).

(d) *Networks and generalization*

As noted previously, models of perceptual classification are only useful if they can generalize and discover the underlying model that gave rise to the training set. It is therefore crucial to determine whether neural networks are able to exhibit this key property. To explore this question, consider again the model neuron defined in equations (2.1) and (2.2). The first thing to note is that the functionality is defined by the processing of continuously graded signals using continuous, smoothly varying relationships. This means that small changes in the inputs will, in general, result in small changes in the output. Moreover, suppose a model neuron (with continuous output function) is responding decisively to a particular pattern so that its output lies close to the asymptotes (0 and 1) of the squashing function. In this regime, the slope of this function is very small, and any change in activation will make a negligible change in the output. Since variations in the inputs cause activation changes, we conclude that a model neuron which is responding decisively is relatively insensitive to its input. This will, in turn, foster generalization, since patterns similar to that currently being applied will not significantly alter the neuron's output.

In summary, it is the 'smooth' or analogue style signal processing paradigm of artificial neurons, together with their intrinsic nonlinearity, which promotes generalization. In a full well-trained network (one which has clear responses to each pattern), these properties work synergistically across individual model neurons, so that similar input patterns will evoke similar patterns of activity over the hidden layer which will, in turn, elicit similar network outputs.

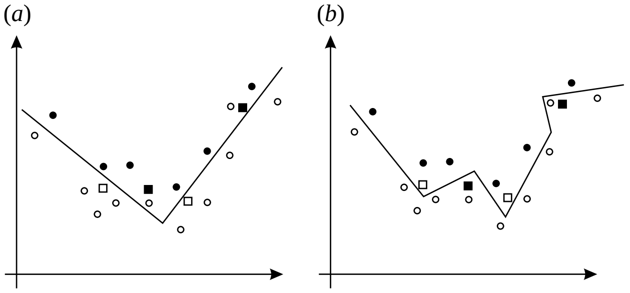


Figure 5. Overtraining in pattern space. (*a*) Simple decision surface separating two classes shown by light and dark symbols. The round symbols (for both classes) are the training set, square symbols are test patterns. There is no misclassification of the test patterns. (*b*) Same set of patterns with a complex decision surface. There is misclassification of the test set.

Further, while the particular artificial neuron we have studied so far is a rather impoverished model of the real animal neuron, the nonlinear combination of analogue signals is almost certainly found in real neurons. We therefore conjecture that generalization is an intrinsic property of real brain circuits arising through some quite general properties of biological neural processing.

Generalization in a network can, however, be severely reduced if the network is not well suited to modelling the underlying pattern data. In particular, if there are too many hidden nodes, then generalization can be curtailed—a point which is best made in the context of pattern space. Consider the classification problem with two classes, shown schematically in figure 5.

In figure 5*a*, there are two line segments indicating two hyperplane fragments in the decision surface, implemented using two hidden units. Two members of the training set have been misclassified, apparently indicating poor performance. However, suppose that four previously unseen *test patterns* (i.e. patterns not in *P*) are presented as shown by the square symbols. These have been classified correctly and the net has generalized from the training data. Thus, if we interpret the two misclassified training patterns as outliers or noisy data, the net has implemented a good model of the data which captures its essential characteristics in pattern space.

Consider now figure 5*b*, in which there are six line segments associated with the use of six hidden units. The training set is identical to that used in the previous example and each one has successfully been classified. However, all four test patterns have incorrectly been classified so that, even though the training data are all dealt with correctly, there may be many patterns which are misclassified. The problem here is that the net has too much freedom (via its abundance of hidden nodes) to choose its decision surface and has *overfitted* it to accommodate any noise and intricacies in the data without regard to the underlying model. Thus, while a two-layer net with an essentially unlimited supply of hidden nodes can tackle arbitrary tasks, inferring the underlying model with a limited training set *P* is a problem of some delicacy.

Some of the techniques for dealing with the problem of overfitting are outlined in (Gurney 1997, section 6.10); however, one prominent technique deserves mention here, and is based on the method of *cross-validation* found in conventional statistical modelling. Consider again the network with too many hidden units whose decision surface is shown schematically in figure 5*b*. The diagram shows the decision surface after exhaustive training, but what form does this take in the early stages of learning? It is reasonable to suppose that a smoother form (something more like that in figure 5*a*) would be developed at this time before the net has had a chance to learn the details in the training set. If we then curtail the training at a suitable stage, it may be possible to 'freeze' the net in a form that generalizes more appropriately. Rosin & Fierens (1995) showed that this is exactly what does happen in a simple example with two inputs.

Assuming that this process of gradual approximation is a general one when training feedforward nets, how are we to know when to stop training the net? One approach is to divide the available training data into two sets: one training set proper *P*, and one so-called *validation set V*. The idea is to train the net in the normal way with *P* but, every so often, to determine the error with respect to the validation set *V*. The typical behaviour of a network under this process of cross-validation is shown in figure 6. One criterion for stopping training, therefore, is to do so when the error over *V* reaches a minimum, since this is indicative that generalization with respect to patterns not in the nominal training set *P* is optimal. Cross-validation is a technique borrowed from regression analysis in statistics and has a long history (Stone 1974). That such a technique should find its way into the 'toolkit' of supervised training in feedforward neural networks should not be surprising because networks are also fitting models to statistical data. While we have emphasized the pattern space viewpoint of network function in the paper, it is also possible to conceive of networks as performing function approximation (Gurney 1997, section 6.7.2) and so they can be thought of as performing nonlinear regression. These similarities are explored further in the review article by Cheng & Titterington (1994).

(e) *Knowledge representations in neural networks*

At the end of training a network, any knowledge or long-term memory is stored in the weights or connection strengths. This has given rise to the term *connectionism* to describe the neural network modelling approach in biological areas (especially in psychology). However, while knowledge is stored in the weights, there is an additional way in which knowledge representation occurs in networks. On applying an external input and allowing it to be processed, a characteristic pattern of the activity will be developed across the net that may be thought of as representing knowledge about the current input. For feedforward nets, it is the intermediate hidden layers that provide a particular focus of interest here, for it is across these nodes that an *internal representation* of the training set occurs. In this view, these representations are then operated on, or decoded by, the output layer into a form which is to be interpreted as the 'answer' or response to the input. As noted previously, the role of the hidden layer(s) may be thought of as ensuring that perceptually similar inputs are re-represented so as to be close together in the pattern vector sense, a perspective of hidden unit functionality emphasized by Rumelhart & Todd (1993). A related viewpoint conceives of hidden units as 'feature detectors' that extract the underlying aspects of the training set while ignoring irrelevant clutter or noise.

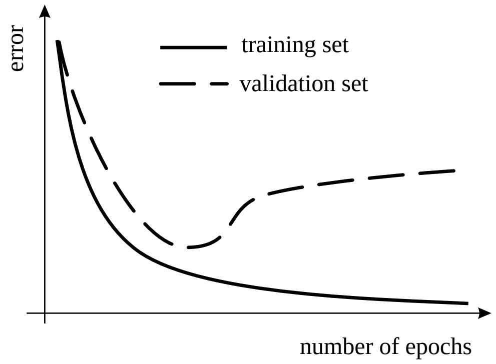


Figure 6. Cross-validation. Use of a validation set allows periodic testing to see whether the model has overfitted. Optimal performance occurs where the error for the validation set is minimal.

Whatever interpretation is adopted in respect of the hidden layer, there are essentially three different types of activity profile that can occur over any neural layer (hidden or otherwise). In a *localist* representation, each semantically discrete item, concept or idea is associated with the activity of a single node. For example, in a classification task, the occurrence of a particular class would be signalled in a neural layer by a single node of that layer being active (output close to 1) while all others are inactive (output close to 0). Figure 7*a* shows an example of this scheme using four nodes to categorize objects into one of the four classes—horizontal rectangle/ellipse, vertical rectangle/ellipse. Activities which are close to 1/0 are shown by filled/open circles, respectively. Figure 7*b* shows a *semilocalist* or *feature-based* representation of the same classes. Each node now stands for a feature of the object class—'horizontal', 'vertical', 'rectangle' and 'ellipse'. Classes are now designated by suitable combinations of active feature nodes. Note that in both localist and semilocalist representations, a minimum of four nodes are required in each case. Finally, figure 7*c* shows a *distributed* representation, in which every node within the layer plays some role in the activity profile representing each class. Here, nodes may be partially active (indicated by shades of grey) and we do not necessarily need four nodes to obtain a unique pattern of activity for each class (the example uses three). There is now no clear interpretation of the significance of activity on any particular node (as indicated by the question marks) and we talk of the nodes designating *micro-features* or *sub-symbolic* entities.

The emphasis by some on distributed representations, combined with the fact that, in principle, artificial neurons can process their information in parallel, has led to the term parallel distributed processing (PDP) to be applied to the computational paradigm established using neural networks (Rumelhart *et al.* 1986).

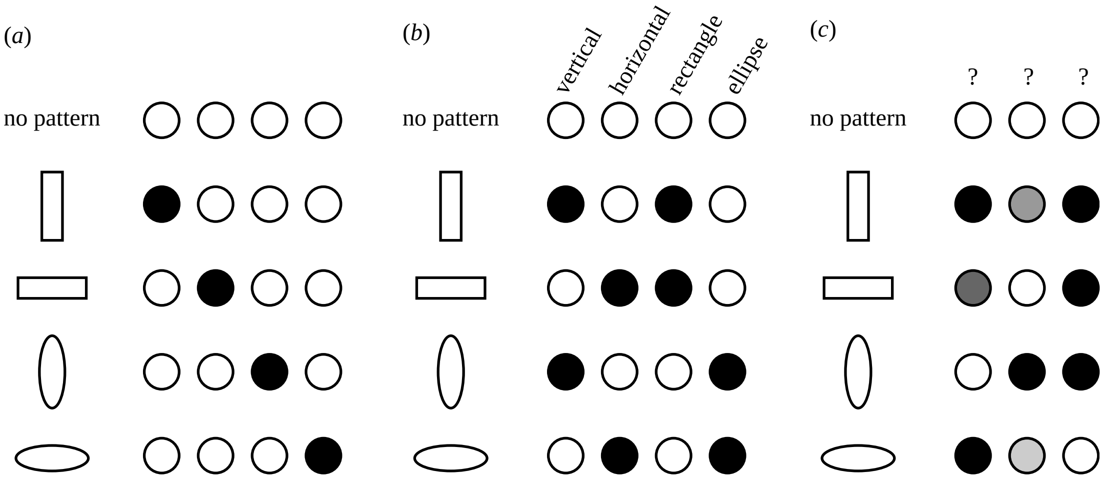


Figure 7. Neural network representations: (*a*) localist, (*b*) semilocalist or feature-based and (*c*) distributed.

(f) *Learning temporal sequences*

The networks we have dealt with so far are unable to deal with temporal dependencies; they respond immediately to their input with an output response. However, many perceptual processes rely on discovering temporal sequences in the input. For example, language, animal noises, and calls in general may be thought of as a sequence of aural stimulus primitives, concatenated to produce an overall temporal pattern. To learn this patterning requires that a network stores some kind of 'memory' or previous state information that can be combined with its present input, thereby conditioning that input on its temporal context within a sequence. Figure 8 shows a simple example of a network that can perform this kind of task. Structurally, it consists of a two-layer network of hidden and output nodes as shown in figure 2, but it is augmented by another set of nodes—the *state* or *context* nodes—which supply input to the hidden layer. The state nodes are also hidden in the sense that they do not interact with the environment, and receive a copy of the activity on the hidden layer via fixed weights with value 1. The net is also endowed with a clock which sequences operations in the network over a series of discrete time-steps as follows. At the start of all operations, the context nodes are initialized to output value 0.5. At the first time-step, the first input pattern in the sequence is supplied to the external input, the network produces an output, and the hidden layer copies its pattern of activity to the state nodes. If the network is being trained, then backpropagation could be applied at this point. At the second time-step, the next input pattern is presented but, since the context nodes have a record of the previous hidden-layer activity, the hidden nodes also receive their own previous state as input. In this way, the network can learn temporal context and sequential dependencies.

The first example of this kind of network appeared in a technical report by Jordan (1986) and it was Elman (1990) who popularized the technique by showing how this general architecture could learn sequences in a variety of abstract temporal patterns as well as simple examples in the English language. Pfennig & Ryan present a paper in this issue which uses a network of the kind described by Elman.

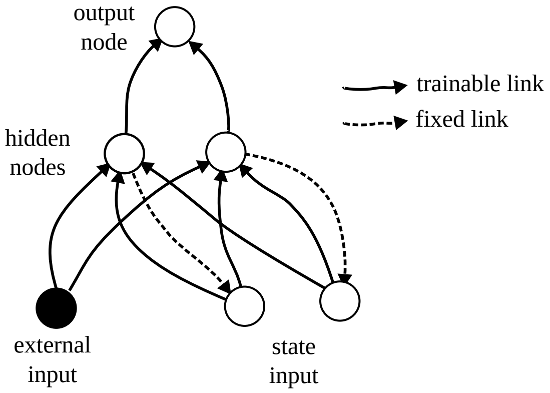


Figure 8. Net used by Elman (1990) to encode temporal sequences. It is drawn so that the state or context nodes (that encode the previous hidden-layer state) appear as another set of inputs.

Quite generally, any network with feedback or *recurrent* connections (like those between the state and hidden nodes in the Elman net) will support 'memory' of some kind. Indeed, networks with massively recurrent interconnection have been extensively studied as models of associative memory in both abstract (e.g. Hopfield 1982) and psychological (e.g. McClelland & Rumelhart 1985) settings.

This completes the technical exposition part of the paper. In §3, we move on to discuss the theoretical status of network models and their structural and functional components.

3. META-THEORETICAL ISSUES

It has already been noted that a two-layer net can, in principle, perform any input–output mapping. The question to ask when building a neural network model is therefore not 'will it work?' but, rather, 'does the model shed any light on the target system?' More formally, we ask whether the model gives any theoretical insight into the mental and neural processes underlying perception and cognition. The relationship between neural network models and theories of mental processing has been hotly debated—for example, Smolensky (1988), McCloskey (1991) and Green (2001). However, as a prelude to our discussion, it is useful to distinguish, quite generally, between two rather different types of computational model, without specific reference to neural networks.

(a) *Description versus governance*

A phenomenological or *descriptive* model of a system is one which replicates the behaviour of the system but, in doing so, does not make recourse to the mechanisms that are believed to *govern* the system's behaviour. This distinction is scientifically very general, but has been explored in relation to brain modelling by Dayan (2002). For example, the time course of the voltage of a neural membrane when an action potential is produced may be described reasonably accurately by some high-order polynomial function of distance along the axon and time. However, this makes no reference to the sodium and potassium currents which are supposed to underlie action potential generation. In contrast, the account of Hodgkin & Huxley (1952) and Koch (1999) invokes just these mechanisms to provide an account of action potential generation. As another example, this time from economics, consider financial indicators such as the Dow–Jones or FTSE indices. It may be possible to fit their time-series approximately by arbitrarily complex functional forms (indeed a linear trend with positive slope is a good first approximation), but these quantities are ultimately governed by the action of thousands of independent 'agents'—stock-brokers who buy and sell shares on the financial markets. A mechanistic model might attempt to invoke the dynamic interaction of these agents and the markets they generate.

These two examples differ from each other in so far as one (the neural membrane) is deterministic, while the other (stock prices) is subject to noise and is a problem in statistical pattern analysis. Neural networks are a class of tools that are well suited to tackling problems of the latter kind. For example, using training data generated by recording market prices over the recent past, a network could take as inputs a set of raw share prices and other financial indicators, and attempt to generate the next day's prices or FTSE index. The resulting model will make no reference to agent-based mechanisms but will describe, phenomenologically, the trends in the data.

Neural networks considered as general statistical modelling tools share similarities with other techniques in this general area. Thus, the outputs of the hidden layer in a feedforward network—the internal representation of the data—are analogous to the factors in factor analysis, and the hidden unit weights akin to the loadings on the variables. Moreover, in the same way that it is possible to try and interpret the factors of factor analysis (under 'rotation' of the loading vectors), it makes sense to try and understand what the hidden nodes in a network are representing. In other words, rather than conceive of the network as a 'black box' that simply replicates behaviour (which is nevertheless still useful) one can probe the model to see what combinations of input are primarily responsible for determining the output. In this way, neural networks as statistical models are endowed with explanatory power. Olden (2007) provides an example of a neural network as a statistical model in this issue of the Journal, and goes on to examine methods for discovering causal relationships among the model variables.

(b) *The problem of biological realism*

Suppose we have a neural network model which takes inputs that could directly represent sensory stimuli of some kind, and whose outputs represent a perceptual classification. The function of the network is, therefore, potentially mirroring a processing task performed in the brain. We might then reasonably ask to what extent any explanatory power of the neural network relates to the mechanisms in the brain that underlie the perceptual task. The problem here however is that, while biological perception is mediated by the brain, and neural networks are somewhat brain-like, they are in many respects biologically implausible. The functionality of real neurons is enormously more complex than that implied in our model neuron (equations (2.1) and (2.2)) and brain circuits are usually more complex than the simple homogeneous feedforward nets described here. Neural networks might therefore appear *prima facie* to have little or no explanatory power with regard to the computations performed by brain circuits underlying perception (Crick 1989).

There are two possible ways to tackle this problem. First, we could choose to meet the demands of biological realism head on, as it were, and constrain our models to be biologically plausible. This route takes us into the newly established discipline of computational neuroscience in which it is acknowledged that, unlike the stereotypical and homogeneous networks described so far, brain systems typically make use of complex circuits with many layers, together with both inter- and intra-layer feedback connections (Shepherd & Koch 1998). Further, the functionality of real neurons is more powerful than that of the artificial nodes in neural networks, and shows a much richer diversity (Koch *et al.* 2002). Thus, the output of real neurons is best characterized as a series of discrete voltage 'spikes' (action potentials) rather than a continuous valued variable. Information may therefore be encoded, not only in the mean firing rate (one interpretation of the node output y), but also in the specifics of inter-spike timing (Rieke *et al.* 1996). Neural behaviour is mediated by a profusion of different ionic currents traversing the neural membrane, and real neurons have an extended morphology allowing complex computation to be performed over the dendritic arbour. The possible variations of membrane function and morphology give rise to a huge diversity of cell types in the brain with a corresponding diversity of computations. This may include the simple 'weight and add' of artificial neural net nodes, but also extends to include nonlinear combinations of inputs (Shepherd & Koch 1998). Models which are explicitly constrained by detailed biological data form the subject of the papers in this issue by Borst (2007) and by Williamson (2007).

An alternative approach to tackling the problem of biological realism and neural networks is to step back and try to define the problem space of 'brains and computation' more precisely. This was the remit of Dror & Gallogly (1999) when they explored the problem of biological plausibility in the general arena of cognitive modelling. Here, we focus specifically on neural network models of brain function but, nevertheless, start with a general contextual framework for our analysis.

(c) *Marr's hierarchical analysis*

In a technical report (Marr & Poggio 1976), and later in his seminal book on vision (Marr 1982), Marr described a hierarchical framework for understanding computational processes in perception and cognition. At the top of the hierarchy is the *computational* level. This attempts to answer the questions—what is being computed and why? It describes the essential characteristics of the input–output transforms and any constraints that apply therein. The next level is the *algorithmic* which describes precisely how the computation is being carried out, and finally, there is the *implementation* level which gives a detailed description of what hardware the algorithm makes use of. Marr's original example in his book *Vision* (Marr 1982), provides a very clear illustration of this framework. Consider the computation of the bill in a supermarket with a cash register. In answer to the top-level question of 'what' is being computed, it is the arithmetical operation of addition. As to 'why' this is being done, it is simply that the laws of addition reflect or model the way we should accumulate prices together from piles of goods in a trolley; it is incorrect, for example, to multiply the prices together. Next, we wish to know exactly how this arithmetic operation is performed. The answer is that it is done by the normal procedure taught at school where we add individual digits in columns and carry to the next column if required. Further, in cash registers, this will be done in the decimal representation rather than binary (normally encountered in machine arithmetic) because rounding errors are incurred when converting between the normal (decimal) representation of currency and binary. As for the implementation, this occurs using logic gates made out of silicon, silicon oxide and metal. Note that choices at different levels are, in principle, independent of each other. For example, we could have chosen to use a binary representation and alternative implementations might make use of mechanical machines or pencil and paper. The importance of discovering good representations for solving the problem is crucial; the ancient Romans failed to develop a positional number system and so struggled to fully develop arithmetic.

We now consider an example in neural networks models of perception. It concerns the computation of the apparent velocity of moving objects in the visual field. This is ethologically useful since we might want to know how fast a car, or falling piece of fruit, is travelling in order to avoid it, or catch it, respectively. In Marr's scheme, then, the computation being performed is to find the velocity of the object in retinotopic coordinates (find how fast the object is moving with respect to the eye).

One algorithm for doing this computation is based on Fourier analysis of the image. Just as one-dimensional temporal audio waveforms may be decomposed into Fourier (sinusoidal) components, so too can two-dimensional spatial images be decomposed into components which consist of gratings with sinusoidally varying luminance. Figure 9*a* shows an example of such a grating which is supposed to be moving in a direction indicated by the velocity vector (arrow) $\boldsymbol{v}_{\mathrm{g}}$. This direction is defined by the perpendicular to the grating bars, and the length of this vector is proportional to the speed of the grating. However, if the grating is the only component in the image, it nominally extends infinitely in all directions in the plane. Therefore, any motion component of the grating along its bars is undetectable. Even if the grating was attached to some large but finite surface, any realizable motion detection system will only be able to sample a small part of this image (indicated in the figure by the circular aperture through which the grating is being viewed). Thus, both theoretically and practically, the only observable motion component is that which is perpendicular to the grating bars (Wallach 1976; Vallortigara & Bressan 1991). The grating motion is therefore indistinguishable from that of any other grating which has the same perpendicular velocity component (for example, the one with velocity v^* in figure 9*a*) and there is a family of gratings compatible with the moving image, having velocities whose vectors lie on a *constraint line* (shown as a dotted line in the figure). Of course, real moving surfaces consist of the superposition of many sinusoidal components, the simplest of which consists of a combination of two gratings. This results in a moving plaid, an example of which is shown in figure 9*b*. The velocity of the plaid must be the same as that of its components and so its vector must lie at the intersection of the two constraint lines associated with the two component gratings. The algorithm for computing the image velocity is therefore a two-stage process: evaluate the perpendicular motion vectors of each spatial component and then compute the velocity consistent with the intersection of constraints (IOC) of these components (Adelson & Movshon 1982).

In Marr's scheme, there are several possibilities at the implementational level for executing the IOC algorithm. It could be done, for example, with paper and pencil using a drawing of the geometry (as in figure 9*b*), or algebraically using the associated trigonometric relations. We now show that there is a neural network implementation, which can solve IOC (Gurney & Wright 1992). The network has a layer of inputs that encode preferential responses to specific gratings and motions. Thus, each input responds best to a grating of given direction, speed and spatial frequency. The output layer is trained to preferentially encode true image velocity in which each node responds most strongly when encoding a particular image speed v, and direction θ. For example, the plaid of figure 9*b* would give rise to two inputs responding to the component gratings, and a single output encoding the plaid velocity.[1] Suppose we arrange the output nodes on a pair of Cartesian axes for speed, and direction, so the location of each node is determined by its preferred values for these quantities (see figure 9*c*). By dint of the arguments above, the response of the network to an input representing a single grating with speed v_{g} and direction θ_{g} is ambiguous; the grating does not uniquely define an image velocity and it will stimulate many output nodes. The resulting pattern of activity in the output layer is shown schematically in figure 9*c* by the 'U'-shaped curve (the geometry

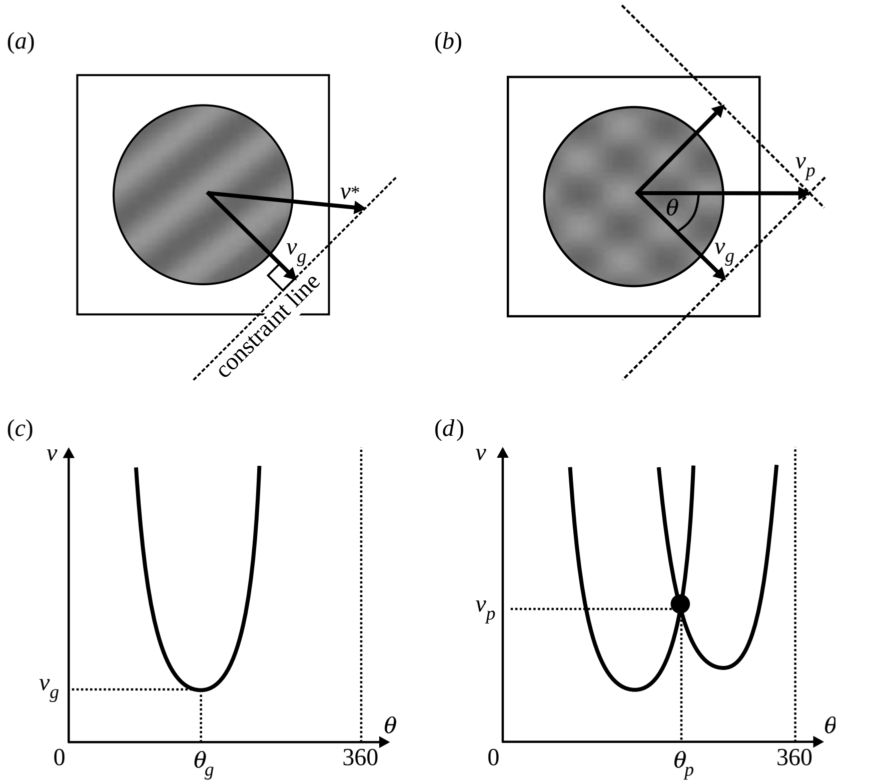


Figure 9. Intersection of constraints. (*a*) A single image component (moving sinusoidal grating) viewed through a circular aperture. The motion of the grating is consistent with all motion vectors that fall on the dotted constraint line. (*b*) Two moving gratings superposed produce a plaid whose image velocity is unambiguously given by the intersection of the constraint lines for each component. (*c*) Response of a network (trained on image motion) to single moving grating. (*d*) Response of network to a plaid with largest response shown by the filled circle.

in figure 9*b* shows that this function takes the form $v_g/\cos(\theta_p-\theta_g)$ where θ_p is the associated pattern direction). When two gratings are presented, as input, they each give rise to 'U'-shaped pattern of activity but, at the point where these functions overlap, the response of the network is enhanced since it sees twice the input (figure 9*d*). There is therefore a unique 'hump' of activity in the network corresponding to a velocity defined by the IOC, and the network implements the IOC algorithm.

(d) *Mapping from networks to biological neural circuits*

The image velocity network, as well as illustrating an application of Marr's hierarchy to perceptual processing, also shows that it is possible to think of networks as implementing algorithms rather than simply combining signals in apparently unstructured ways. This is important because it points to the possibility that network models could be thought of as quantitative tests of computational hypotheses about perceptual and cognitive processing in the brain. This idea is predicated on the following assumption: that neural network models have sufficient points of contact with real brain circuits that the hypotheses they are founded on can allude directly to the brain, albeit at a fairly high level of abstraction. Thus, in the example of the neural network model of IOC, we demonstrated that, since an abstract neural network can implement the proposed algorithm, a real brain system could also *in principle* do the same thing. In other words, we hypothesize that the brain solves the computational problem of determining image motion velocity using a two-stage process with IOC applied to analysis of local Fourier components.

To test this hypothesis, we are now charged with trying to *map* the abstract neural network onto real brain circuits. Gurney & Wright (1992) advanced the argument that there is evidence for correspondence between the input layer and visual cortical area V1, and the output layer and visual area MT in primates. This evidence is based on V1- and MT-expressing sensitivity to moving gratings (Hubel & Wiesel 1968) and image motion (Movshon *et al.* 1985), respectively, which is consistent with the representations used in the network. The project of mapping the neural network for IOC onto biological neuronal circuits is, however, far from complete. Neocortex (including MT) is a complex six-layered structure with a variety of neural cell types and microcircuits that our impoverished single layer of simplified neurons cannot hope to emulate. In addition, while we know V1 innervates MT, we do not know if the precise connectivity required by the model is to be found anatomically. However, generalizing the foregoing analysis from the IOC example implies that, in so far as neural networks are used to model brain processes, they occupy a position between the algorithmic level in the Marr hierarchy and the implementational level (i.e. biological neural tissue). The abstract neural network is an additional *mechanistic* level, and we refer to the process of attempting to realize the neural network in brain circuits as one of *mechanism mapping* (Gurney *et al.* 2004).

(e) *Networks and neural representations*

We now explore further the role played by representations in linking neural networks with brain circuits. The first step in attempting to map the IOC net onto the visual system in the brain depended on an identification of the representations used in the net with counterparts in the visual system. In this model, the input representation was explicitly based on knowledge of the way in which gratings are known to be represented in area V1. However, the network was trained in a way rather different to those outlined in the first half of this article. Rather than supplying the net with a supervisory target output, the net had to 'discover' structure in the training set under a process of *self-organization* using learning rules that had biological plausibility.

The function of this kind of network has similarities with statistical techniques like principal component analysis (PCA), since the network performs a *dimension reduction* or compression of the patterns in the input space. Thus, the network consists of a single layer of artificial neurons and, like PCA, discovers features of the training set that are sufficient to describe the essentials of this set of patterns. In the IOC net, for example, these features are speed and direction. Each node becomes maximally responsive to patterns with a particular speed and direction, and tends to ignore other aspects of the input. To this extent, it has 'discovered' that the input set is essentially two dimensional.

The learning takes place using rules based on the principles outlined by Hebb (1949) that are supposed to govern synaptic change in real neurons. Hebb proposed that if a presynaptic neuron was simultaneously active with a postsynaptic partner, then the synapse between these neurons would increase its strength. In the network, the amount of change on a weight is governed by the size of the corresponding input and the output of the neuron to which the weight belongs; the weight grows if there is a strong correlation between input and output, and decays if there is little or none. Self-organizing networks of this kind were developed and popularized by Kohonen (1984); an introduction to these ideas may also be found in Gurney (1997).

Based on the observations above, self-organizing networks can be used in situations where we suspect that the input patterns have too many components and that they may essentially be described more naturally using only a few parameters or dimensions. The utility of the IOC network model was, however, motivated by a more biologically grounded developmental perspective. Thus, it aimed to demonstrate that the representation of image velocity, observed in area MT, could emerge naturally from a biologically plausible developmental process using a prior stage of processing whose encoding was also biologically plausible. The use of self-organizing neural networks to demonstrate the emergence of known neural representations has a long history associated especially with topographic maps of retinotopy (Willshaw & von der Malsburg 1976), orientation selectivity (von der Malsburg 1973) and ocular dominance (Swindale 1980). For a more recent review of the field, see Price & Willshaw (2000).

In contrast to this, Zipser (1992) has shown how feedforward networks trained with supervised methods to perform a perceptual task often develop representations in their hidden layer which appear similar to those observed experimentally in the neural circuits believed to implement the same computational task in the brain. For example, neurons in posterior parietal area 7a in monkeys are believed to compute a head centred representation of object location in space, by combining retinotopic information about object spatial location, together with gaze angle (Andersen *et al.* 1985). Zipser & Andersen (1988) constructed a network model with two layers to perform the same computation. They discovered that the hidden layer yielded nodes whose response properties mimicked well that of certain neurons in area 7a.

Zipser (1992) supplies further examples of this kind, and conjectures an explanation for why the representations found in networks might mirror the biology. First, he notes that the setting of parameters (the weights) in a network, in order to achieve some given input–output behaviour, is an example of the general process of system identification. In the general case, forcing a model to show the same behaviour as a target system will not necessarily force the behaviour of the model's *constituents* to mimic those of the system. Indeed, it may not even be sensible to make that comparison because the constituents in each case will be too dissimilar. However, in the case of neural networks and biological neural circuits, there is enough similarity to at least attempt a comparison, and Zipser (1992) speculates that this similarity is, in fact, sufficient to force a close correspondence between the model representations and those in the target neural system.

The ability of neural networks to exhibit biologically plausible representations leads to hypotheses that can guide experimental programmes, and can help explain complex patterning in physiological data from cells that have non-trivial receptive field properties as a result of taking part in complex representations.

(f) *Networks and neural architectures*

We now turn to another way in which neural networks can test computational hypotheses about perceptual processing. In this instance, it concerns the gross structural properties or architecture of the network and the target brain system. We illustrate it with an example based on the connectionist model of Stroop processing by Cohen *et al.* (1990). In the Stroop task, the subject has to name the colour used to render stimulus words which are colour names (e.g. 'red', 'blue', etc.). In some instances, the word and the colour ink are congruent (e.g. 'red' in red ink) in others they conflict (e.g. 'red' in blue ink). The subject has to speak aloud the name of the ink colour as quickly as possible, and reaction times and error rates are used as performance measures. There is also a control condition in which no word is presented but, rather, a meaningless pattern like 'XXXX'. In addition, it is possible to contrast the colour-naming task with another one in which the subject has to read the word. The main result is that the mean reaction time in the colour-naming task for the conflict condition is usually longer than that for the congruent condition—a phenomenon known as the Stroop effect (Stroop 1935; MacLeod 1991). Typical reaction time data are shown in figure 10*a*.

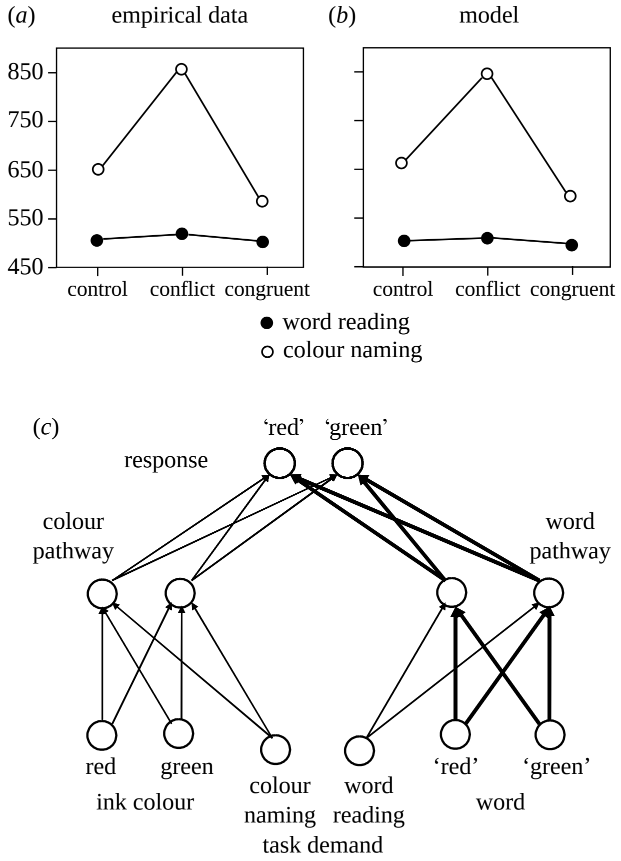


Figure 10. The model of Cohen *et al.* (1990) for the Stroop task. (*a*) Stroop task, experimental data. (*b*) Replication of data by the model. (*c*) The model architecture.

In their model, Cohen *et al.* (1990) advanced the hypothesis that the Stroop effect may be explained by the difference in processing strength between the pathways in the brain that process colour information and word information. Note that this hypothesis is couched in terms of neural processing because it refers to signal 'pathways' and the relative strength of transmission within these pathways. It therefore makes sense to test this hypothesis with a neural network. The model consisted of two subnetworks—one each for colour and word information (see figure 10*c*). The pathways had an identical minimal architecture with two inputs (for two colours), two hidden nodes and a common output layer. Crucially, the pathways for colour and word information differed in their connection strengths, with the word pathway having the stronger links (as indicated by thicker lines in the figure). The task (word reading or colour naming) was specified by sensitizing or biasing the hidden nodes using task-demand nodes. When the latter became active, the hidden nodes required less input from elsewhere to generate significant output.

The model was successful in several respects, including its ability to replicate the basic pattern of Stroop data (see figure 10*b*). At the very least, it is therefore a successful phenomenological model of the Stroop task. However, the model also provides evidence for the differential strength of processing hypothesis for, while there is clearly more to the biological neural processing of colour and word than the simple subnetworks used in this model, the hypothesis is not contingent on the details within those pathways. Thus, any arbitrarily complex pair of pathways would give the same result, so long as that used for word reading transmitted information more efficiently than that for colour processing.

In a similar way to the use of networks in discovering neural representations, the Stroop example highlights the utility of the core similarity between abstract nets and real neuronal circuits. In the Stroop network, this similarity (connection strengths, pathways, signal combinations, etc.) is sufficient to frame a hypothesis about biological neural architectures (different 'strengths' in two pathways) in a simplified setting. Indeed, the simplification implicit in the abstract network endows the model with explanatory power that it would not have if encumbered with too much detail. Of course, in going further and attempting to map the colour- and word-reading pathways onto their biological counterparts, we would discover whether the pathways really were different in the way suggested, and a more stringent test of the hypothesis would ensue.

The Stroop network of Cohen *et al.* (1990) is but one example of many where neural networks have been used to explore architectural issues. Jacobs *et al.* (1991) examined networks that could simultaneously determine the location and identity ('where' and 'what') of objects in visual space. Undifferentiated networks did not perform as well as split or modular networks that dealt with each sub-problem independently, thereby giving insight into why these problems also appear to be solved independently in the animal visual system. Hinton & Shallice (1991) described a network model which was able to exhibit many of the phenomena associated with deep dyslexia in humans. More significantly, in the current context, they experimented with a variety of architectures and discovered that many of the key results were contingent only on there being a layer of nodes with internal recurrent (feedback) connections able to support 'memory' or state information. Finally, we draw attention to the work of McClelland *et al.* (1995) which used a neural network-based analysis of the need for two memory systems in animals: a short-term memory system which can learn 'isolated' or episodic items rapidly, and a long-term system that integrates each episode into a wider knowledge base in a prolonged process of consolidation. Once again, the arguments used by the authors of the model were quite general, making reference to only fundamental properties of neural network development and learning.

4. CONCLUDING REMARKS

In the fist half of the paper, we outlined some of the technical issues in neural networks. We focused largely on feedforward nets (and variants) because these form the basis of many of the models described in this issue. However, there are a plethora of network architectures we have not covered and the reader is referred to the references at the beginning of that section for more information. The concept of generalization leads to a better understanding of what neural networks might buy us in terms of modelling power, and enables us to develop principled ways of training them to take advantage of this power (using cross-validation and the like). The theory of knowledge representation is an important one if we are to understand network mechanisms and their relation to any corresponding features in the target system (be it the animal brain or an ecosystem).

Our aim in the second half of the paper was to demonstrate that neural networks are not limited to 'mere simulation' of input–output behaviour, but that they have a role to play in developing theories of cognition and perception. Some neural network models are purely phenomenological descriptions of a target system and make no claim to establish links with internal mechanisms in that system. Rather, the network is a statistical model of the system which, nevertheless, provides explanatory power by highlighting relationships between variables, and suggesting new internal (hidden layer) variable combinations, features or 'factors'. If a network models some computational task performed by the animal brain, it is tempting to make correspondences between the network and the corresponding brain mechanisms underlying the computation. At first glance, this approach appears flawed because neural networks lack the biological realism to directly model real neural tissue. However, careful examination of a principled approach to computational modelling due to Marr suggests otherwise. Thus, neural networks appear to occupy a place in Marr's hierarchy somewhere between the abstract algorithmic level and implementation in biological neural circuits. Construction of neural networks at this 'abstract mechanistic' level is therefore one part of much larger modelling strategy that seeks to understand high-level computational questions, and algorithms, as well as details of implementation in real neural tissue. As such, neural networks seem to have sufficient core similarities to biological neural circuits to offer insights in two general areas: first, in discovering and understanding the role of neural representations; and second, in testing hypotheses about large-scale neural connectivity or architectures.

I would like to thank Tom Stafford for reading a draft of the paper. This work was supported in part by EPSRC grant EP/C516303/1.

ENDNOTE

[1] In fact the net uses so-called 'course coding' (Touretzky 1995). Thus small clusters of nodes are active rather than individuals, but this is not crucial to the argument.

REFERENCES

Adelson, E. H. & Movshon, J. A. 1982 Phenomenal coherence of moving visual patterns. *Nature* **300**, 523–525. (doi:10.1038/300523a0)

Andersen, R. A., Essick, G. K. & Siegel, R. M. 1985 Encoding of spatial location by posterior parietal neurons. *Science* **230**, 456–458. (doi:10.1126/science.4048942)

Barto, A. G. 1985 Learning by statistical cooperation of sele-interested neuron-like computing elements. *Hum. Neurobiol.* **4**, 229–256.

Barto, A. G. & Anandan, P. 1985 Pattern-recognizing stochastic learning automata. *IEEE Syst. Man Cybern.* **SMC-15**, 360–375.

Barto, A. G., Sutton, R. S. & Anderson, C. W. 1983 Neuronlike elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cybern.* **13**, 835–846.

Bishop, C. M. 1996 *Neural networks for pattern recognition*. Oxford, UK: Oxford University Press.

Borst, A. 2007 Correlation versus gradient type motion detectors: the pros and cons. *Phil. Trans. R. Soc. B* **362**, 369–374. (doi:10.1098/rstb.2006.1964)

Cheng, B. & Titterington, D. M. 1994 Neural networks: a review from a statistical perspective. *Stat. Sci.* **9**, 2–54.

Cohen, J. D., Dunbar, K. & McClelland, J. L. 1990 On the control of automatic processes—a parallel distributed-processing account of the Stroop effect. *Psychol. Rev.* **97**, 332–361. (doi:10.1037/0033-295X.97.3.332)

Crick, F. 1989 The recent excitement about neural networks. *Nature* **337**, 129–132. (doi:10.1038/337129a0)

Dayan, P. 2002 Levels of analysis in neural modeling. In *Encyclopedia of cognitive science* (ed. L. Nadel). London, UK: Nature Publishing Group; John Wiley and Sons Ltd.

Dror, I. E. & Gallogly, D. P. 1999 Computational analyses in cognitive neuroscience: in defense of biological implausibility. *Psychon. Bull. Rev.* **6**, 173–182.

Elman, J. L. 1990 Finding structure in time. *Cogn. Sci.* **14**, 179–211. (doi:10.1016/0364-0213(90)90002-E)

Fitzsimonds, R. M., Song, H. J. & Poo, M. M. 1997 Propagation of activity-dependent synaptic depression in simple neural networks. *Nature* **388**, 439–448. (doi:10.1038/41267)

Green, C. D. 2001 Scientific models, connectionist networks, and cognitive science. *Theor. Psychol.* **11**, 97–117.

Gurney, K. 1997 *An introduction to neural networks*. London, UK: UCL Press (Taylor and Francis group).

Gurney, K. N. & Wright, M. J. 1992 A self-organising neural network model of image velocity encoding. *Biol. Cybern.* **68**, 173–181. (doi:10.1007/BF00201439)

Gurney, K., Prescott, T. J., Wickens, J. R. & Redgrave, P. 2004 Computational models of the basal ganglia: from robots to membranes. *Trends Neurosci.* **27**, 453–459. (doi:10.1016/j.tins.2004.06.003)

Haykin, S. 1999 *Neural networks: a comprehensive foundation*. Englewood Cliff, NJ: Prentice Hall.

Hebb, D. 1949 *The organization of behaviour*. New York, NY: Wiley.

Hinton, G. E. & Shallice, T. 1991 Lesioning an attractor network—investigations of acquired dyslexia. *Psychol. Rev.* **98**, 74–95. (doi:10.1037/0033-295X.98.1.74)

Hodgkin, A. L. & Huxley, A. F. 1952 A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* **117**, 500–544.

Hopfield, J. J. 1982 Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl Acad. Sci. USA* **79**, 2554–2558. (doi:10.1073/pnas.79.8.2554)

Hubel, D. H. & Wiesel, T. N. 1968 Receptive fields and functional architecture of monkey striate cortex. *J. Physiol.* **195**, 215–243.

Jacobs, R. A., Jordan, M. I. & Barto, A. G. 1991 Task decomposition through competition in a modular connectionist architecture. *Cogn. Sci.* **15**, 219–250. (doi:10.1016/0364-0213(91)80006-Q)

Jordan, M. I. 1986 *Serial order: a parallel distributed approach*. San Diego, CA: University of California, Institute for Cognitive Science.

Koch, C. 1999 *The biophysics of computation: information processing in single neurons*. New York, NY: Oxford University Press.

Koch, C., Mo, C. & Softky, W. 2002 Single-cell models. In *The handbook of brain theory and neural networks* (ed. M. Arbib). Cambridge, MA: MIT Press.

Kohonen, T. 1984 *Self-organization and associative memory.* Berlin, Germany: Springer.

Lippmann, R. 1987 An introduction to computing with neural nets. *ASSP Mag. IEEE* **4**, 4–22.

MacLeod, C. M. 1991 Half a century of research on the Stroop effect—an integrative review. *Psychol. Bull.* **109**, 163–203. (doi:10.1037/0033-2909.109.2.163)

Makhoul, J., El-Jaroudi, A. & Schwartz, R. 1989 Formation of disconnected decision regions with a single hidden layer. In *Int. Joint Conf. on Neural Networks*, vol. 1, pp. 455–460, Seattle.

Marr, D. 1982 *Vision: a computational investigation into human representation and processing of visual information*. New York, NY: W.H. Freeeman and Co.

Marr, D. & Poggio, T. 1976 *From understanding computation to understanding neural circuitry.* Cambridge, MA: MIT AI Laboratory.

McClelland, J. L. & Rumelhart, D. E. 1985 Distributed memory and the representation of general and specific information. *J. Exp. Psychol. Gen.* **114**, 159–188. (doi:10.1037/0096-3445.114.2.159)

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. *Psychol. Rev.* **102**, 419–457. (doi:10.1037/0033-295X.102.3.419)

McCloskey, M. 1991 Networks and theories—the place of connectionism in cognitive science. *Psychol. Sci.* **2**, 387–395. (doi:10.1111/j.1467-9280.1991.tb00173.x)

McCulloch, W. S. & Pitts, W. 1943 A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **7**, 115–133. (doi:10.1007/BF02478259)

Minsky, M. & Papert, S. 1969 *Perceptrons*. Cambridge, MA: MIT Press.

Movshon, J. A., Adelson, E. H., Gizzi, M. S. & Newsom, W. T. 1985 The analysis of moving visual patterns. In *Pattern recognition mechanisms* (eds C. Chagas, R. Gattas & C. G. Gross), pp. 117–151. Berlin, Germany: Springer.

Olden, J. D. 2007 Critical threshold effects of benthiscape structure on stream herbivore movement. *Phil. Trans. R. Soc. B* **362**, 461–472. (doi:10.1098/rstb.2006.1974)

Parker, D. B. 1982 *Learning-logic: office of technology licensing*. Stanford, CA: Stanford University.

Price, D. & Willshaw, D. 2000 *Mechanisms of cortical development*. Oxford, UK: Oxford University Press.

Rieke, F., Warland, D., de Ruyter van Steveninck, R. & Bialek, W. 1996 *Spikes*. Cambridge, MA: MIT Press.

Rosenblatt, F. 1958 The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **65**, 386–408. (doi:10.1037/h0042519)

Rosin, P. L. & Fierens, F. 1995 Improving neural net generalisation. In *Proc. IGARSS'95*. Firenze, Italy.

Rumelhart, D. E. & Todd, P. 1993 Learning and connectionist representations. In *Attention and performance XIV* (eds D. Meyer & S. Kornblum), pp. 3–31. Cambridge, MA: MIT Press.

Rumelhart, D. E., McClelland, J. L. & The PDP Research Group 1986 *Parallel distributed processing: explorations in the microstructure of cognition*. Cambridge, MA: MIT Press.

Shang, Y. & Wah, B. W. 1996 Global optimization for neural network training. *IEEE Comput.* **29**, 45–54.

Shepherd, G. M. & Koch, C. 1998 Introduction to synaptic circuits. In *The synaptic organization of the brain* (ed. G. M. Shepherd), pp. 1–36. Oxford, UK: Oxford University Press.

Smolensky, P. 1988 On the proper treatment of connectionism. *Behav. Brain Sci.* **11**, 1–23.

Stone, M. 1974 Cross-validatory choice and assessment of statistical predictions. *J. R. Stat. Soc.* **B36**, 111–133.

Stroop, J. R. 1935 Studies of interference in serial verbal reactions. *J. Exp. Psychol.* **18**, 643–662. (doi:10.1037/h0054651)

Sutton, R. S. 1988 Learning to predict by the method of temporal differences. *Mach. Learn.* **3**, 9–44.

Sutton, R. S. & Barto, A. G. 1998 *Reinforcement: an introduction*. Cambridge, MA: MIT Press.

Swindale, N. V. 1980 A model for the formation of ocular dominance stripes. *Proc. R. Soc. B* **208**, 243–264.

Touretzky, D. S. 1995 Connectionist and symbolic representations. In *The handbook of brain theory and neural networks* (ed. M. Arbib), pp. 243–247, 1st edn. Cambridge, MA: MIT Press.

Tveter, D. R. 1996 Backpropagator's review. See http://www.dontveter.com/bpr/bpr.html.

Vallortigara, G. & Bressan, P. 1991 Occlusion and the perception of coherent motion. *Vis. Res.* **31**, 1967–1978. (doi:10.1016/0042-6989(91)90191-7)

von der Malsburg, C. 1973 Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik* **14**, 85–100. (doi:10.1007/BF00288907)

Wallach, H. 1976 On perceived identity 1: the direction of motion of straight lines. In *On perception* (ed. H. Wallach). New York, NY: Quadrangle.

Werbos, P. 1974 *Beyond regression: new tools for prediction and analysis in the behavioral sciences*. Boston, MA: Harvard University.

Widrow, B. & Hoff Jr, M. E. 1960 Adaptive switching circuits. In *IRE WESCON Convention Record*, pp. 96–104.

Widrow, B. & Stearns, S. D. 1985 *Adaptive signal processing*. Englewood Cliffs, NJ: Prentice Hall.

Wieland, A. & Leighton, R. 1987 Geometric analysis of neural network capabilities. In *1st IEEE Int. Conf. on Neural Networks*, vol. III, pp. 385–392. San Diego, CA.

Williams, R. J. 1987 *Reinforcement learning connectionist systems*. Boston, MA: Northeastern University.

Williamson, R. & Chrachri, A. 2007 A model biological neural network: the cephalopod vestibular system. *Phil. Trans. R. Soc. B* **362**, 473–481. (doi:10.1098/rstb.2006.1975)

Willshaw, D. J. & von der Malsburg, C. 1976 How patterned neural connections can be set up by self-organization. *Proc. R. Soc. B* **194**, 431–445.

Zipser, D. 1992 Identification models of the nervous system. *Neuroscience* **47**, 853–862. (doi:10.1016/0306-4522(92)90035-Z)

Zipser, D. & Andersen, R. A. 1988 A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature* **331**, 679–684. (doi:10.1038/331679a0)