Pre-synaptic Lateral Inhibition Provides a Better Architecture for Self-Organising Neural Networks

Michael W Spratling

Division of Informatics, University of Edinburgh, 5 Forrest Hill, Edinburgh. EH1 2QL. UK.

Abstract. Unsupervised learning is an important property of the brain and of many artificial neural networks. A large variety of unsupervised learning algorithms have been proposed. This paper takes a different approach in considering the architecture of the neural network rather than the learning algorithm. It is shown that a self-organising neural network architecture using pre-synaptic lateral inhibition enables a single learning algorithm to find distributed, local, and topological representations as appropriate to the structure of the input data received. It is argued that such an architecture not only has computational advantages but is a better model of cortical self-organisation.

Submitted to: *Network: Comput. Neural Syst.*

1. Introduction

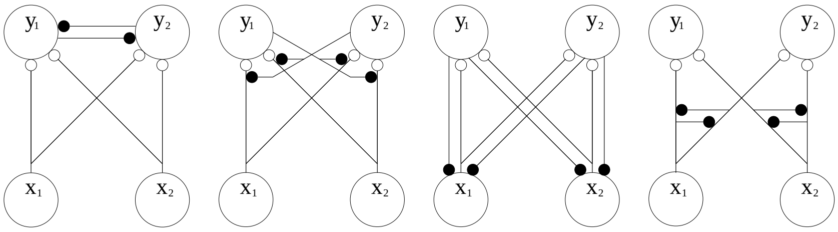
The pattern of activity of the nodes in a neural network constitutes a representation of the input to that network. In unsupervised learning, unlike supervised learning, what node activations constitute a good representation of the current input is not explicitly defined by the training data. Thus many different algorithms have been proposed (Becker & Plumbley 1996; Barlow 1989) which impose various, application independent, measures of 'goodness' on the representation formed; such as information preservation, minimum redundancy, minimum entropy, density estimation over priors, maximum likelihood parameter estimation, topology preservation *etc.* Such algorithms may be implemented using local learning rules or via the optimisation of an objective function. By modifying the learning rules and activation function, or by modifying the objective function, the same neural network architecture may be used to find representations that preserve different measures of 'goodness' (*e.g.* (Oja 1995), *cf.* (Földiák 1990) and (Földiák 1989), *cf.* (Fyfe 1997) and (Harpur & Prager 1996), *cf.* (Sirosh & Miikkulainen 1994) and (Földiák 1990)).

In broad terms these algorithms can be classified with respect to the general form of representation that is learnt. In a distributed code the activity of a population of nodes provides the representation. Depending on the size of this active population, in relation to the total number of nodes, this may be a dense or sparse code. In the limit a sparse code becomes a local code in which a single node provides the representation as the only active node (through hard, or winner-take-all, competition). In addition, the topological structure of the input space may be preserved by placing an ordering on the nodes. With a distributed code all the nodes may represent the same feature of the input, however, it is also possible that different nodes represent different aspects of the input pattern, or (equivalently) the simultaneous presentation of multiple inputs. Such a code is described as factorial. In contrast, the simultaneous presentation of multiple inputs, or the underlying factors of a particular input, can only be represented by a single node when using a local code, requiring distinct 'grandmother cells' to represent each combination of inputs.

Within the diversity of unsupervised learning algorithms which have been proposed all these forms of representation can be learnt, however, no single algorithm learns them all. Thus, in order to learn a good representation for a particular task the appropriate form of encoding must be known beforehand in order to select an appropriate algorithm. In addition, an algorithm which finds a specific, predefined, form of representation fails to model the diversity of structure found in the cortex. Cortical representations vary with sparse distributed coding (Földiák & Young 1995; Olshausen & Field 1996), possibly, where appropriate, local coding (Thorpe 1995), as well as topological maps (Swindale 1996; Knudsen *et al.* 1990) all being used. Since there is evidence to suggest that neurons, from all areas of the neocortex, apply the same learning mechanisms (Sur 1989; O'Leary 1993; Johnson 1997), it would seem to be important to have a single artificial neural network model which can generate all these forms of representation. This paper introduces a self-organising neural network architecture which, using a single learning algorithm, can generate distributed, local, and topological representations, as appropriate to the structure of the input data received.

2. Models of Lateral Inhibition

Lateral inhibition is an essential feature of many artificial, self-organising, neural networks. It provides a mechanism through which neurons compete to represent distinct patterns within the input space. It can be used to decorrelate the activations of nodes, so that the response of the network is a distributed representation of the input (Földiák 1990). Stronger lateral inhibition can be used to generate a local encoding, where a single node inhibits all others (Yuille & Geiger 1995). In addition, by allowing the lateral inhibition strength to vary as a function of the distance between nodes it is possible to preserve the topology of the input space (Swindale 1996; Sirosh & Miikkulainen 1994).



(a) (b) (c) (d)

Figure 1. Models of lateral inhibition. Very simple neural networks consisting of two nodes which each receive two inputs are shown. Nodes are shown as large circles, excitatory synapses as small open circles and inhibitory synapses as small filled circles. (a) The standard model of lateral inhibition. (b) The pre-synaptic lateral inhibition model. (c) The negative feedback model. (d) The SONNET-2 model.

2.1. The Standard Model

The majority of self-organising neural networks use lateral inhibition to provide competition between the outputs of nodes (Földiák 1989; Földiák 1990; Marshall 1995; Sirosh & Miikkulainen 1994; Swindale 1996), or produce the same effect without explicitly representing connections between nodes (Kohonen 1997; Ritter *et al.* 1992; Rumelhart & Zipser 1985). This 'standard' architecture (as shown for a simple network of two nodes in figure 1(a)) consists of a layer of neurons each of which receives excitatory inputs (x_j) from a receptive field (RF). The feedforward synaptic weights (q_{ij}) are refined by a learning algorithm to generate a representation of the input space. Lateral inhibition, via inhibitory synaptic weights (w_{ik}), occurs between the outputs of nodes (y_i) to provide competition for the refinement of the feedforward weights. At equilibrium the output of each node will be:

$$y_i = \sum_{j=1}^{m} (q_{ij} x_j) - \sum_{k=1 (k \neq i)}^{n} (w_{ik} y_k) .$$

This standard architecture has been used (with variations to the learning rules used) to find distributed, local, and topological representations. However, no single algorithm learns them all, due to conflicting requirements for lateral inhibition in forming local and distributed representations.

To learn a local representation the lateral weights simply need to become sufficiently strong to allow the activity of a single node to be dominant at any one time, while ensuring that all nodes do respond to some inputs (to prevent a subset of nodes representing all the patterns). Various methods of ensuring each node represents an input are used (such as decreasing the activation threshold (Földiák 1990; Intrator & Cooper 1992; Rumelhart & Zipser 1985) or dishabituating (Desieno 1988; Ahalt & Fowler 1993) nodes which respond too seldom). These methods presume that the input patterns occur with approximately equal probability and so constrain all nodes to be the most highly active (or the winner of the competition) with approximately equal frequency.

To learn a distributed code lateral weights must be found that are sufficiently strong to prevent all nodes representing the same patterns, but that are sufficiently weak to allow more than one node to be active if multiple inputs are presented simultaneously. Thus, there is a balance to be struck between too little and too much competition. The activity of the nodes in response to the presentation of each input is ambiguous as to whether the weights have found the correct compromise: when two nodes are simultaneously active it can indicate either that lateral weights are too weak and have allowed both units to respond to the same input, or that each node has responded correctly to the simultaneous presentation of distinct inputs; when a single node is active it can indicate either that the lateral weights are too strong to allow other nodes to respond to other inputs, or that only one input pattern is present. To resolve this ambiguity it can be assumed that each pair of input patterns occur with equal probability and the learning rules can be designed to modify the lateral weights to ensure pairs of nodes are coactive with equal frequency, *e.g.*: $\Delta w_{ik} = \alpha y_k \left(y_i - w_{ik}\right)$ (Marshall 1995) or $\Delta w_{ik} = \alpha \left(y_k y_i - p^2\right)$ (Földiák 1990) (where p is the probability of each input pattern occurring, assumed *a priori*). A drawback with this assumption is that a representation will be unstable if there are more nodes than input patterns. In such a case nodes which do not represent an input pattern will receive weaker and weaker inhibition until they do become responsive after which they may become inhibited and unresponsive again, for a short time, or may take over representing one of the input patterns from another node (Marshall 1995). It is thus necessary to know beforehand how many patterns are present in the data. A more significant problem is that such a network will modify its lateral weights to ensure that all pairs of representations are coactive with approximately equal frequency, whether or not this is appropriate. Hence, if certain pairs of stimuli never occur together in the input data then the network will adjust the lateral weights to force the representations of these mutually-exclusive pairs to be coactive and so will generate spurious representations. For example, in order for a network to represent the colour and shape of objects it should generate a factorial code in which nodes representing different colours and different shapes can be coactive. Using this form of lateral inhibition such a network will correctly respond to white squares and black triangles, but will also generate representations of black-white squares and black square-triangles even though such stimuli never exist in the input data. Hence, this form of lateral inhibition fails to provide factorial coding except for the exceptional case in which all pairs of patterns co-occur together. Furthermore, this approach is also incompatible with using lateral inhibition to form a topological representation, since modifying the strength of competition by a function of the distance between the nodes would upset the delicate balance in the strength of the lateral weights.

The rules for learning lateral weights to produce local coding and those to produce distributed coding are incompatible. The rules for local coding cannot be used to form a factorial representation since they allow only one node to be active at a time. Alternatively, the rules for factorial coding cannot be used to represent mutually exclusive inputs since the lateral weights would weaken until each pair of nodes was occasionally coactive. This incompatibility is a fundamental limitation of the architecture used, not of any particular learning algorithm. Hence, modification to the architecture rather than the learning rules is required.

2.2. Pre-synaptic Lateral Inhibition

An architecture that uses lateral inhibition of pre-synaptic inputs (figure 1(b)), to provide competition for inputs rather than between outputs, overcomes the incompatibility between local and distributed coding. At equilibrium the output of each node will be:

$$y_i = \sum_{j=1}^{m} \left(q_{ij} x_j - \sum_{k=1 (k \neq i)}^{n} \left(w_{ikj} y_k \right) \right)^+ .$$

Where w_{ikj} is the strength of the inhibitory synapse from node k to synapse j of node i, and $(z)^+$ is the positive half-rectified value of z.

If the lateral weights have somehow become selective in only inhibiting inputs to which the inhibiting node is most responsive, then each node will attempt to ‘block’ its preferred inputs from activating other nodes. If two nodes try to represent the same input pattern there will be strong competition between them. If two nodes represent distinct patterns each node can respond to its preferred input without inhibiting the other node (assuming there is not much overlap between patterns). Thus the only constraint, for both local and distributed coding, is that lateral weights are sufficiently strong to allow nodes to claim their preferred input pattern. There is thus no incompatibility between the requirements for local and factorial coding, and an identical algorithm can find either as appropriate to the input data. In addition, since lateral weights can continue to increase, modifying them by a function of distance between nodes (to encourage the formation of a topologically ordered representation) does not prevent correct codes being found eventually. Any learning algorithm that refines the lateral weights to allow a node to strongly inhibit its preferred input pattern from reaching other nodes, while reducing its inhibition of other inputs, can generate distributed, local, and topological representations. This ability is primarily a property of the architecture rather than any particular learning rules. Furthermore, since this method only needs to assume that the firing rates of individual nodes are equal, rather than coactivations of pairs of nodes, it is stable if there are more, or less, nodes than input patterns and if the input data contains pairs of patterns which are mutually exclusive.

One disadvantage of this architecture would seem to be the larger number of lateral weights required, however, these weights do not need to be individually learnt, resulting in this method being computationally efficient. In order for a node, k, to block one of its preferred inputs, j, from activating another node, i, the lateral weight w_{ikj} needs to inhibit the input j from reaching node i. This needs to happen if the inhibiting node, k, is responsive to input j *i.e.* w_{ikj} needs to be strong if q_{kj} is strong. Thus, the lateral weights need simply to be a scaled version of the inhibiting node's feedforward weights, *i.e.*

$$w_{ikj} = \frac{\alpha}{\beta} q_{kj},$$

and the activation function becomes:

$$y_i = \sum_{j=1}^{m} \left(q_{ij} x_j - \sum_{k=1 (k \neq i)}^{n} \left(\frac{\alpha}{\beta} q_{kj} y_k \right) \right)^{+}.$$

In a biologically plausible model the lateral weights could be represented and learnt separately, using a learning rule, such as; $\Delta w_{ikj} = \alpha f^{lateral}(x_j, y_k)$. While the feedforward weights were learnt, independently, as; $\Delta q_{kj} = \beta f^{forward}(y_k, x_j)$. The assumption is simply that the learning rule for the forward weights ($f^{forward}$) is equivalent to the learning rule for the lateral weights ($f^{lateral}$), and that both weights have the same initial value.

Further reduction in computational complexity is achieved since the output of the network can be found without numerical integration. Instead, the process of competition between the nodes can be approximated by selecting a single winning node (win) to inhibit all the others at each iteration:

$$y_{win} = \sum_{j=1}^{m} \left(q_{win,j} x_j \right), \qquad y_i = \sum_{j=1}^{m} \left(q_{ij} x_j - \frac{\alpha}{\beta} q_{win,j} y_{win} \right)^{+} \ \forall i \neq win.$$

The inhibiting node is selected to be that which is most strongly activated by the current input (*i.e.* the node for which $\sum_j q_{ij} x_j$ is greatest), but modified by an habituation function to keep all nodes winning with approximately equal probability. This simplification succeeds since the inhibiting node competes with all others, forcing preferred inputs to become differentiated. However, since nodes which have learnt distinct input patterns do not inhibit each other it does not, necessarily, generate a local code and other nodes can continue to respond (and learn) even if they have not been selected as the winner.

The pre-synaptic lateral inhibition architecture was implemented within a competitive learning algorithm. For each stimulus presented to the network the node, i, with the highest value of $\left(\sum_j q_{ij} x_j\right) \times (1 + \mu (I - nP_i)) + \nu$ was selected to inhibit all others (where P_i is the number of iterations for which node i was the winning node, I is the total number of iterations, μ is the habituation scale factor, and ν is a random variable uniformly distributed in the range ± 0.005 times the maximum node activation). After inhibition by the winning node (using the equation given in the previous paragraph), learning was performed to update the synaptic weights, q_{ij}, as a function of y_i and x_j for all nodes. The exact form of the learning rule is not important as long as it has the properties, described above, of increasing the strength of connection between a node and its preferred stimulus while decreasing the strength of connection to other inputs. It was found that variations of normalised Hebbian learning, the outstar rule, and the covariance rule could all produce similar results to those given in section 3. The actual learning rule used was:

$$\Delta q_{ij} = \beta \ \frac{m \left(\frac{x_j}{\bar{x}} - 1 \right)}{\sum_{k=1}^{m} \left| \frac{x_k}{\bar{x}} - 1 \right|} \ \frac{n \left(\frac{y_i}{\bar{y}_i} - 1 \right)^{+}}{\sum_{k=1}^{n} \left(\frac{y_k}{\bar{y}_k} - 1 \right)^{+}}.$$

where $\tilde{x} = \tau\bar{x} + (1-\tau)\tilde{x}$ is the long-term average of the mean past activity for all the inputs, $\tilde{y}_i = \tau y_i + (1-\tau)\,\tilde{y}_i$ is the long-term average of the past output activity for node i, and m and n are the number of inputs and nodes in the network. This rule is qualitatively similar to the covariance rule, except the post-synaptic term is only allowed to be greater than or equal to zero. Hence, post-synaptic activation exceeding a threshold is required before synaptic modification is enabled. Also, the terms have been rearranged so that weight changes are proportional to the ratio of the current activity to previous activity, rather than the difference. In addition, the pre- and post-synaptic terms are normalised so that the total change in synaptic weight that can occur at each iteration is normalised across the network and for individual dendrites. This gives equal importance to each training pattern. Synaptic weights were initialised to have zero strength (tests indicate that random initial weights can also be used). Results were fairly insensitive to the values of the parameters used. All the results shown in this paper were generated using the parameters: $\alpha = 0.0009$, $\beta = 0.00000225$, $\tau = 0.001$, and $\mu = 0.001$.

2.3. Other Non-Standard Models of Lateral Inhibition

It is obvious that inhibition could potentially act between several other points in a neural network, some more of which have been explored in other non-standard architectures.

2.3.1. Negative Feedback Model In this model (figure 1(c)) inhibitory feedback from each node inhibits the inputs to the entire network (Fyfe & Baddeley 1995; Fyfe 1997; Charles & Fyfe 1998; Harpur & Prager 1996). As with pre-synaptic lateral inhibition these weights could be learnt, but as a simplification are set to the same values as the corresponding feedforward weights. However, unlike the model proposed here, inhibition is to the inputs themselves and a node cannot entirely inhibit the input to all other nodes (without entirely inhibiting its own input). Since the inhibitory strength between individual nodes cannot vary there is no possibility of finding topological representations (without resorting to direct synaptic modifications on neighbouring nodes (Fyfe 1996)).

2.3.2. SONNET-2 In this model (figure 1(d)) inhibition is between the feedforward connections (Nigrin 1993; Marshall & Gupta 1998). The change in weight of the inhibitory synapses, and the strength of inhibition, is a function of both the output activation of the inhibiting node and the input. In theory this architecture could develop local, distributed and topological representations, but it is far more complex than the pre-synaptic lateral inhibition model, and no such results have yet appeared.

2.3.3. Yuille's Winner-Take-All Model In this models all the inputs to a node are inhibited equally and in proportion to the total output from all other nodes in the network (Yuille & Greynacz 1989; Yuille & Geiger 1995). At equilibrium the activation of each node will be:

$$y_i = \left(\sum_{j=1}^{m} q_{ij} x_j\right) e^{-\lambda \sum_{k\neq i} y_k}.$$

This network was explicitly designed to form winner-take-all representations. In order to be modified to form a distributed code it would need to find a compromise value for the lateral inhibition strength to allow more than one node to be active simultaneously, and would thus suffer form the same problems as the standard model.

3. Results

A widely used test for factorial coding is the bars data set (Földiák 1990; Harpur & Prager 1996; Fyfe 1997; Charles & Fyfe 1998; Hinton *et al.* 1995; Hinton & Ghahramani 1997). Input consists of a 8x8 grid on which each for the 16 possible horizontal and vertical bars are active with probability $\frac{1}{8}$. Typical examples of input patterns are shown in figure 2(a). When a pre-synaptic lateral inhibition network of 16 nodes was trained on this data it found, after 1600 presentations, a correct representation, where each node represented exactly one bar and each bar was represented exactly once. The synaptic weights corresponding to the represented bar became much stronger than connections to inputs from other grid points (figure 2(b)) and the response of nodes became specific to the presence of that bar in the input data (figure 2(c)). When tested with a further 500 patterns, after training with 4000 pattern presentations, the response generated by this network (for input data consisting of n bars) was such that the n most active nodes were those which correctly represented all the active bars in 100% of unseen (as well as previously seen) test patterns. A more reasonable test is to assume that the number of bars in the input pattern is unknown and to analyse only those nodes that have an activation exceeding a predefined threshold. In this case the active nodes correctly represent all the active bars in the input pattern (and only those bars) for 99% of test patterns. It can be seen that the greater the number of active bars in a test pattern the less distinct active nodes become with respect to the activity level of other nodes. This is due to the overlap between bars of perpendicular orientation, so that an input pattern that contains several bars of one orientation will partially activate all the nodes representing bars at the perpendicular orientation.

Although the learning algorithm encourages nodes to represent a single unique input there is no guarantee that this will happen (the results above are for a typical experiment in which a suitable encoding was found). The results are sensitive to variations in the training process (variations caused by different randomly generated training sets) such that with 16-node networks one node often came to respond to two separate bars (which were thus indistinguishable by the network) while another node represented none. To test the robustness of the algorithm a network consisting of 16 nodes was trained using 54 sets of randomly generated training data, of these experiments only 35% learnt to use each node to represent a single bar. However, the robustness was improved by using more nodes than there were independent input patterns (reaching 100% for a network of 30 nodes, see figure 3(a)). In cases where excess nodes were used, although the allocation of specific bars to specific nodes was still sensitive to changes in the training process, the algorithm robustly found a subset of 16 nodes each of which represented a single unique bar. The robustness of the network to changes in the parameters of the learning algorithm was also tested. Networks consisting of 16 nodes trained using 54 sets of random parameters (with values uniformly chosen in the range $\pm 50\%$ from the hand-picked values used above) learnt a suitable representation in only 31% of cases, while networks consisting of more nodes, tested in the same manner, had increasing robustness (see figure 3(a)). These results are extremely similar to the effects of modifying the training process. Since randomising the network parameters also resulted in random modifications to the training process this suggests that the network is much less sensitive to changes in parameter values than to changes in the learning process.

For a network consisting of 20 nodes (trained using the standard parameter values) the solution was found within 1200 presentations (see figure 4). When tested with a further 500 patterns, after training with 4000 pattern presentations, the response generated by this network (for input data consisting of n bars) was such that the n most active nodes where those which correctly represented all the active bars in 82% of unseen test patterns. The n most active nodes correctly responded to 94% of all bars presented within those test patterns. Nodes which had an activity value exceeding a threshold correctly represented all the active bars in the input pattern (and only those bars) for 77% of unseen test patterns. All these figures rose to 100% after further training. Figure 3(b) shows how the accuracy improves during training for three sizes of network. It can be seen that the profile of accuracy verses training time is very similar for all network sizes and remains similar when experiments are repeated with random variations to the training process.

For any network containing more than 16 nodes there are nodes which do not represent any input pattern. As can be seen from figure 4 using pre-synaptic lateral inhibition is stable, with unused nodes remaining unresponsive to all inputs and having small, uniform, weights. (The representation formed when there are too few nodes, with some nodes representing multiple bars, is also stable.) It is thus not necessary to know beforehand how many patterns are present in the data. Nodes which do not represent an input pattern will still be selected as the inhibiting node

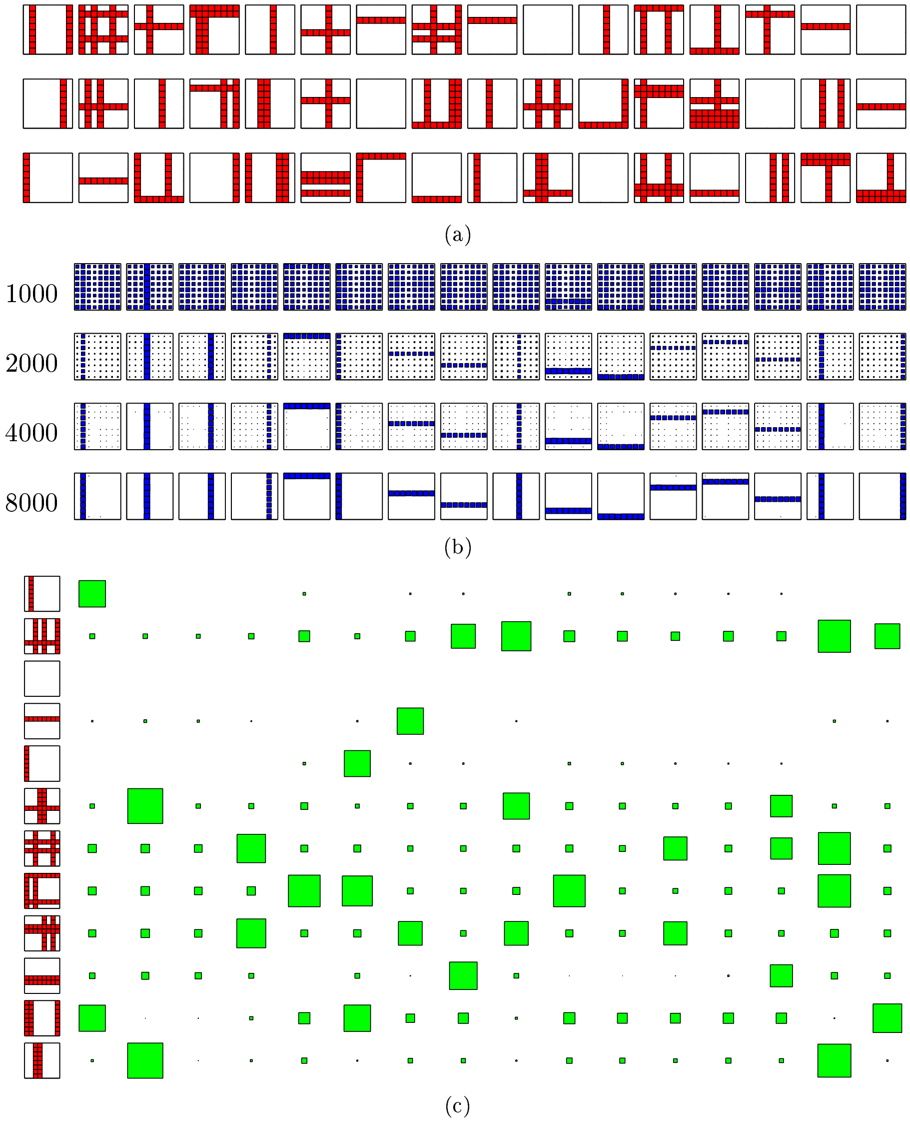
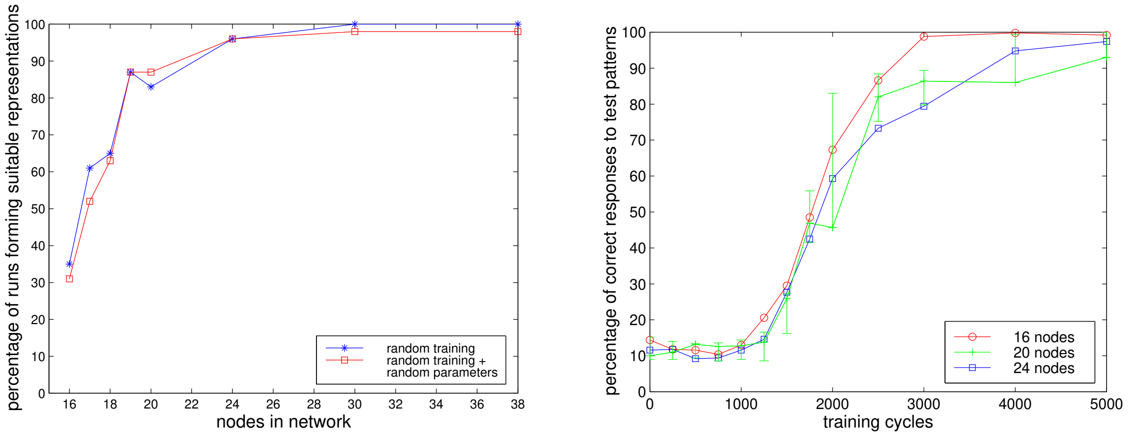


Figure 2. Learning a factorial code. (a) 48 typical input patterns for the bars problem. Dark squares represent active inputs in an 8x8 grid. (b) The synaptic weights for the 16 nodes in a network after training on 1000, 2000, 4000 and 8000 input patterns. The size of each square is proportional to the magnitude of the synaptic weight scaled by the largest weight for that row. (Weights are scaled by row so that the relatively small weights early in training will be visible.) (c) The response of the network after training on 4000 input patterns. The left-most column shows the input pattern, with the remainder of each row showing the response of the network to this pattern. The size of the response of each node, in the same order as the RFs are shown in (b), is represented by the size of each square.



(a) (b)

Figure 3. The changes in robustness and accuracy. (a) The change in robustness with number of nodes in the network. The percentage of experiments, of 4000 training cycles in length, for which the nodes form suitable weights to represent all the bars is shown. Results are given for experiments performed with variations in the training process and for randomly selected parameters of the learning algorithm as well as variations in the training process. (b) The change in the accuracy of the response of the network with training time. The percentage of complete input patterns that are correctly represented by the response of the network is shown. Values are calculated by assuming that nodes ought to respond to the preferred stimulus which is learnt by the end of training. The value at iteration I is calculated from the number of correct responses given for the next 500 training patterns after I. Results are shown for networks containing 16, 20 and 24 nodes. The accuracy is calculated from the response of all nodes with activation exceeding a predefined threshold. The error bars show the maximum and minimum accuracy values found for a 20-node network over the course of 20 successful experiments with random variations in the training process.

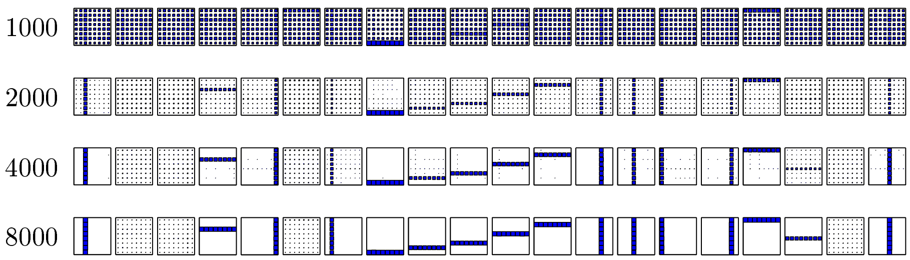


Figure 4. Learning a factorial code with excess nodes. The synaptic weights for the 20 nodes in a network after training on 1000, 2000, 4000 and 8000 input patterns. The size of each square is proportional to the magnitude of the synaptic weight scaled by the largest weight for that row.

(due to habituation being used to keep all nodes winning the competition as often as each other), but when they do inhibit other nodes their weak weights are insufficient to block the inputs to other nodes. Hence, those nodes which better represent the current inputs are still activated, and update their synaptic weights more strongly than the inhibiting node.

So far all the experimental results have been for perfect input patterns. The network can still learn the bars problem even when the input data is corrupted with a considerable level of independent noise (figure 5(a) shows examples of such patterns). In this case networks of both 16 and 20 nodes, using the same learning parameters as above, could both find a factorial coding. The noise slows down learning so that a longer training period is required. With a network of 20 nodes the solution was found after 3000 presentations (figure 5(b)). The response of individual

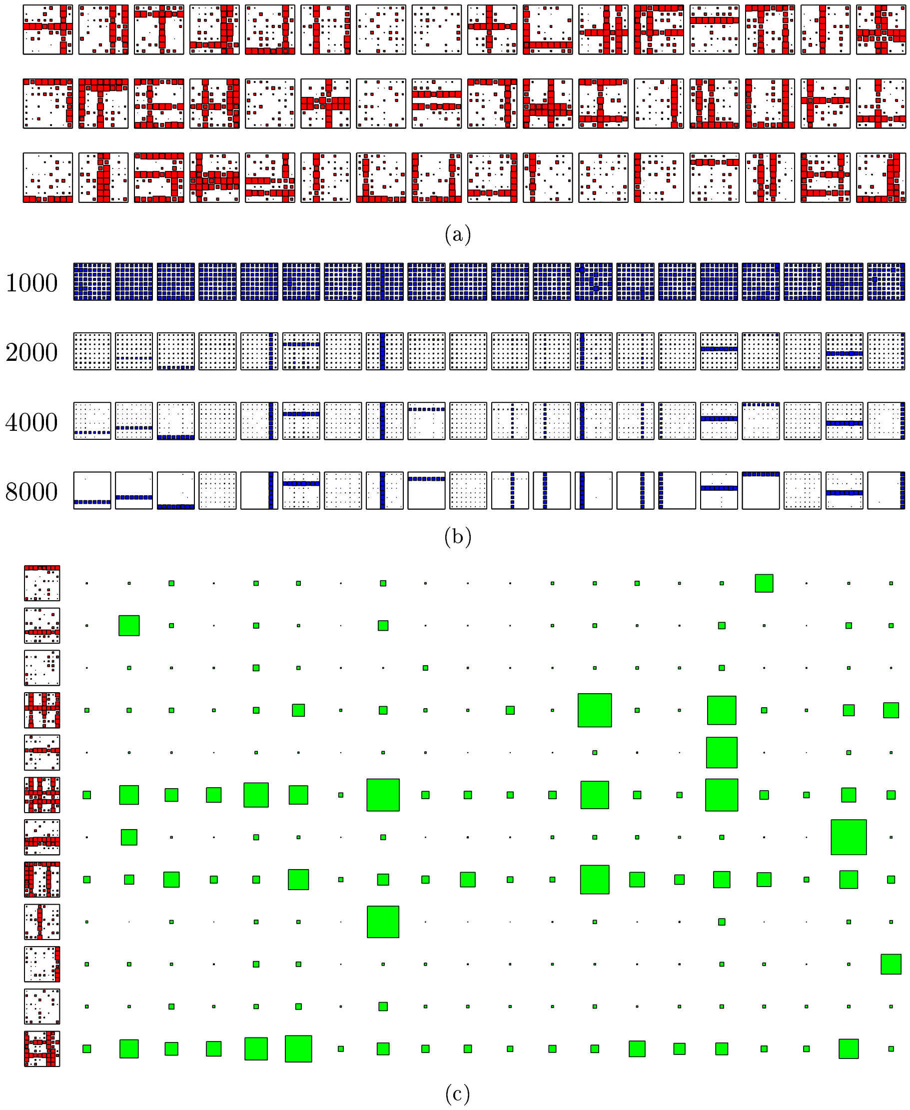


Figure 5. Learning a factorial code with noisy data. (a) 48 typical input patterns for the noisy bars problem. The size of each square is proportional to the activation of that input in an 8x8 grid. (b) The synaptic weights for the 20 nodes in a network after training on 1000, 2000, 4000 and 8000 input patterns. The size of each square is proportional to the magnitude of the synaptic weight scaled by the largest weight for that row. (c) The response of the network after training on 8000 input patterns. The left-most column shows the input pattern, with the remainder of each row showing the response of the network to this pattern. The size of the response of each node, in the same order as the RFs are shown in (b), is represented by the size of each square.

nodes to the presence of particular bars in the input pattern is still fairly clear given sufficient training (figure 5(c)). When tested with a further 500 patterns, after training with 4000 pattern presentations, the response generated by this network (for input data consisting of n bars) was such that the n most active nodes where those which correctly represented all the active bars in 97% of unseen test patterns. In general, noise slowed down learning so that the weights were less differentiated, than before, after a given number of training cycles. In addition, noise in the inputs generates noise in the outputs due to nodes which ought to be inactive being partially activated by inputs which are not parts of the bar patterns, and nodes which ought to be active being less activated by bars which contain reduced input values. Hence, the difference between an active and an inactive node is reduced by the noise and more training is needed to make weights sufficiently selective to overcome this reduction in discernibility. With perfect data it was noted that the overlap between patterns causes those nodes which are active to become less distinct as more bars are present in the input pattern. The same problem is seen with the response to noisy data but starts to occur with fewer active bars in the input data due to the additional, partial, activation of nodes by the background noise.

A local encoding is appropriate when the input data consists of mutually exclusive events. To demonstrate that the same algorithm can represent mutually exclusive input patterns the bars data was modified so that only one bar was active for any one presentation. When a network of 16 nodes was trained on this data it found a correct representation, where each node represents exactly one bar, within 2600 presentations. The weights corresponding to the represented bar become much stronger than connections to inputs from other grid points (figure 6(a)) and the response of nodes becomes specific to the presence of that bar in the input data (figure 6(b)). Moreover, when this network, having been trained using single bars only, was tested with patterns containing multiple bars it generated a correct factorial code with all nodes, corresponding to bars in the input, being active simultaneously (figure 6(c)). When tested with a further 500 patterns, after training with 4000 pattern presentations, the response generated by this network (for input data consisting of n bars) was such that the n most active nodes where those which correctly represented all the active bars in 95% of unseen test patterns. Furthermore, input distributions in which only some of the input features were mutually exclusive could also be learnt and accurately represented. The bars data was modified so that at most, a single horizontal and a single vertical bar occurred together: a single horizontal bar, at a random position, occurred with probability 0.75 and a single vertical bar, at a random position, occurred with probability 0.75. When a network of 16 nodes was trained on this data it found a suitable representation, where each node represented exactly one bar and each bar was represented exactly once within 2000 training cycles. After 4000 iterations, nodes which had an activity value exceeding a threshold correctly represented the active bars in the input pattern (and only those bars) for 100% of test patterns. No spurious activations were generated.

To demonstrate that the algorithm can generate topologically ordered representations, the bars data was again modified. In this application it was required that neighbouring bars be represented by neighbouring nodes. Since perpendicular bars have more overlap, and thus more similarity, than parallel bars one orientation of bars was used only. In addition, the activation of a bar was modified to cause some activation of neighbouring bars, the magnitude of this activation decreasing exponentially with distance (figure 7(a)). The lateral inhibitory weights were modulated by the distance between nodes, such that:

$$y_i = \sum_{j=1}^{m} \left(q_{ij} x_j - \frac{\alpha}{\beta} q_{win,j} y_{win} \Omega\left(win, i\right) \right)^+ \quad \forall i \neq win$$

where $\Omega\left(win, i\right) = \left(1 - \exp \frac{\|\vec{z}_{win} - \vec{z}_i\|^2}{-2\sigma_I^2} - \exp \frac{\|\vec{z}_{win} - \vec{z}_i\|^2}{-2\sigma_E^2}\right)$ is a function of the distance between the inhibited node i and the inhibiting node win, $\vec{z}_i$ is the physical position of a node in the network and $\|\vec{z}_{win} - \vec{z}_i\|$ is the Euclidean distance between nodes win and i (the parameters used were: $\sigma_I = \frac{1}{3}l$ and $\sigma_E = \frac{1}{3}\sigma_I$, where l is the maximum distance between any two nodes in the network). This function provides mutual excitation over a short-range and longer-range inhibition which increases in strength with distance. This encourages neighbouring nodes to represent similar input patterns

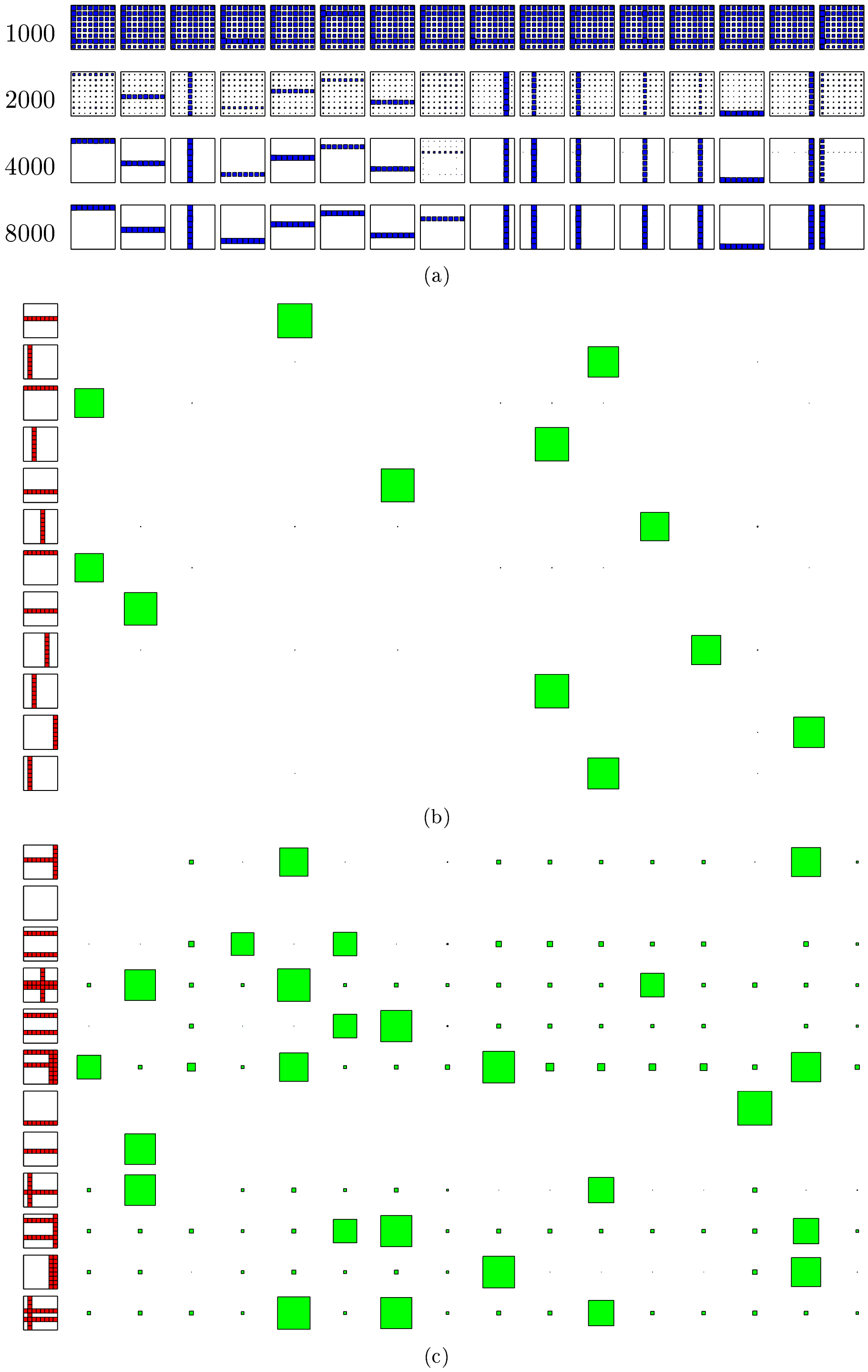


Figure 6. Learning to represent mutually exclusive input patterns. (a) The synaptic weights for the 16 nodes in a network after training on 1000, 2000, 4000 and 8000 input patterns. The size of each square is proportional to the magnitude of the synaptic weight scaled by the largest weight for that row. (b) The response of the network after training on 4000 input patterns. The left-most column shows the input pattern, with the remainder of each row showing the response of the network to this pattern. The size of the response of each node, in the same order as the RFs are shown in (a), is represented by the size of each square. (c) The response of the network to input patterns containing multiple bars after training on 4000 input patterns containing single bars.

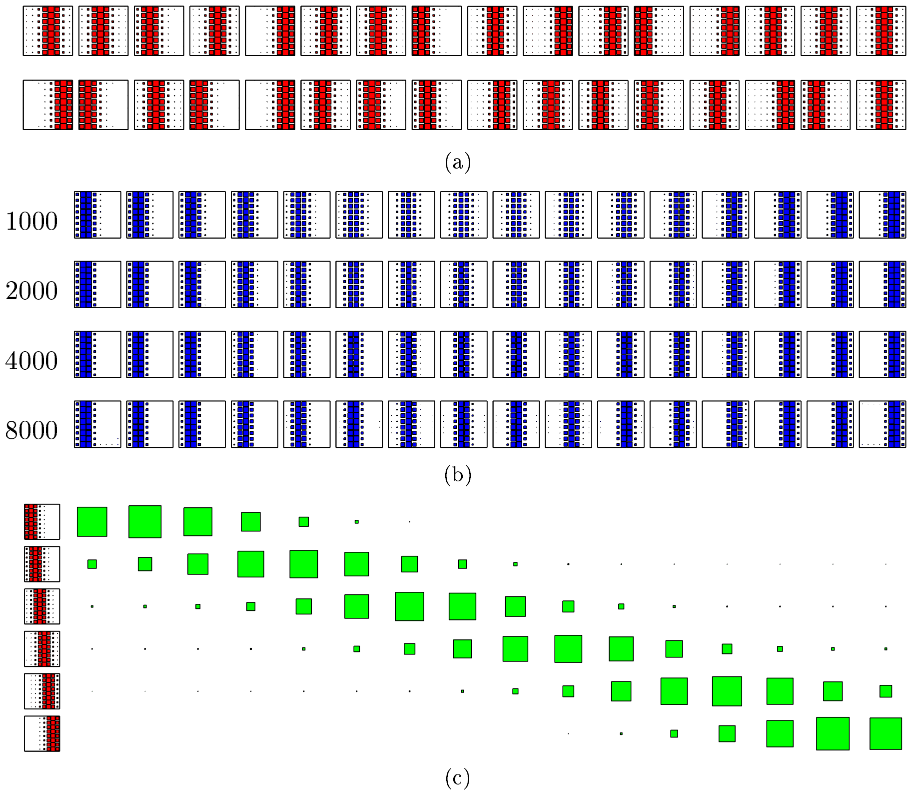


Figure 7. The development of a simple topologically ordered representation. (a) 32 typical input patterns for the topological bars problem. The size of each square is proportional to the activation of that input in an 8x8 grid. (b) The synaptic weights for the 16 nodes in a network after training on 1000, 2000, 4000 and 8000 input patterns. The size of each square is proportional to the magnitude of the synaptic weight scaled by the largest weight for that row. (c) The response of the network after training on 4000 input patterns. The left-most column shows the input pattern, with the remainder of each row showing the response of the network to this pattern. The size of the response of each node, in the same order as the RFs are shown in (b), is represented by the size of each square.

(figure 7(b)). The response of the network to inputs was then a distributed representation with several neighbouring nodes responding to each input (figure 7(c)). As a more convincing, yet still simple, example of topological map formation, the same algorithm was applied to the problem of using 100 nodes arranged on a 10 by 10 square lattice to represent points uniformly distributed over the unit square of the two-dimensional plane. Figure 8 shows the preferred input of each node, with those of neighbouring nodes linked by lines.

4. Conclusions

This paper has introduced a neural network architecture that uses pre-synaptic lateral inhibition. The functioning of such a network has been illustrated on some simple test cases. This paper suggests that a neural network architecture using pre-synaptic lateral inhibition has significant advantages over the standard model of lateral inhibition between node outputs. In this architecture nodes compete for the right to receive inputs rather than for the right to generate outputs. Unlike the standard model, this architecture allows a single learning algorithm to find distributed, local, and (with minor modification) topological representations of the input. It is thus not necessary to know *a priori* the structure of the data in order to generate a good representation for it. In addition, the representations are stable so that it is also not necessary to know *a priori* the number

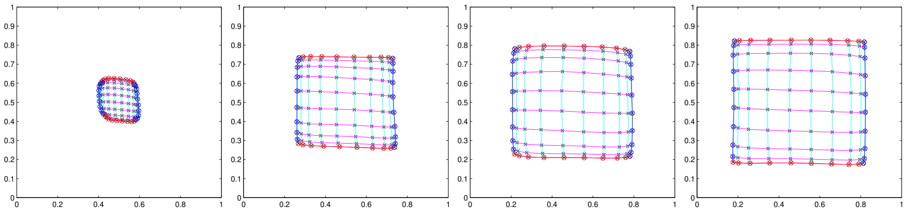


Figure 8. The development of a simple topological map. The map is trained with data uniformly distributed over the unit square of the two-dimensional plane and is shown (from left to right) after training with 1000, 2000, 4000, and 8000 input patterns. Nodes were arranged on a 10 by 10 square lattice. Each node is shown projected onto the input plane at the position of its preferred input, with neighbouring nodes joined by lines.

of nodes required to form the representation. Furthermore, these advantages are achieved with no increase in computational complexity over the standard architecture. These abilities are a function of the network architecture, rather than any clever learning rules.

The proposed architecture requires inhibition specific to particular synaptic inputs. Such synapse specific inhibition might be implemented biologically either through pre-synaptic inhibition via an excitatory synapse on the axon terminal of the excitatory input (Bousher 1970; Shepherd 1990; Kandel *et al.* 1995), or through an inhibitory synapse proximal to the excitatory input on the dendritic tree (Somogyi & Martin 1985). The former mechanism enables an excitatory neuron to directly inhibit the input to another node without the action of an inhibitory interneuron. However, although, axo-axonal, terminal-to-terminal, synapses are common in both vertebrate and invertebrate nervous systems (Brown 1991; Kandel *et al.* 1995), there is no evidence for this mechanism of inhibition between cells in the cortex (Mountcastle 1998). The latter mechanism relies, as does the standard architecture of lateral inhibition, on the action of inhibitory interneurons. However, there is also no support for this mechanism of inhibition in the cortex since the majority of inhibitory synapses to pyramidal cells terminate on the soma or the axon initial segment rather than the dendrites. Hence, although this architecture is computationally attractive and provides a good model of cortical representation its implementation is biologically implausible.

References

Ahalt S C and Fowler J E 1993 Vector quantization using artificial neural network models *Proceedings of the International Workshop on Adaptive Methods and Emergent Techniques for Signal Processing and Communications* ed Docampo D and Figueras A R pp 42–61

Barlow H B 1989 Unsupervised learning *Neural Computation* **1** 295–311

Becker S and Plumbley M 1996 Unsupervised neural network learning procedures for feature extraction and classification *International Journal of Applied Intelligence* **6** 3 185–203

Bousher D 1970 *Introduction to the Anatomy and Physiology of the Nervous System* (Blackwell)

Brown A G 1991 *Nerve Cells and Nervous Systems: An Introduction to Neuroscience* (Springer-Verlag)

Charles D and Fyfe C 1998 Modelling multiple cause structure using rectification constraints *Network: Computation in Neural Systems* **9** 2 167–82

Desieno D 1988 Adding a conscience to competitive learning *International Conference on Neural Networks* (IEEE Press) pp 117–24

Földiák P and Young M P 1995 Sparse coding in the primate cortex *The Handbook of Brain Theory and Neural Networks* ed Arbib M A (MIT Press) pp 895–8

Földiák P 1989 Adaptive network for optimal linear feature extraction *International Joint Conference on Neural Networks* (IEEE Press) pp 401–5

——1990 Forming sparse representations by local anti-Hebbian learning *Biological Cybernetics* **64** 165–70

Fyfe C and Baddeley R 1995 Finding compact and sparse-distributed representations of visual images *Network: Computation in Neural Systems* **6** 333–44

Fyfe C 1996 A scale-invariant feature map *Network: Computation in Neural Systems* **7** 269–75

——1997 A neural net for PCA and beyond *Neural Processing Letters* **6** 1/2 33–41

Harpur G and Prager R 1996 Development of low entropy coding in a recurrent network *Network: Computation in Neural Systems* **7** 2 277–84

Hinton G E and Ghahramani Z 1997 Generative models for discovering sparse distributed representations *Philosophical Transactions of the Royal Society of London. Series B* **352** 1177–90

Hinton G E, Dayan P, Frey B J and Neal R M 1995 The wake-sleep algorithm for unsupervised neural networks *Science* **268** 1158–61

Intrator N and Cooper L N 1992 Objective function formulation of the BCM theory of visual cortical plasticity: statistical connections, stability conditions *Neural Networks* **5** 3–17

Johnson M H 1997 *Developmental Cognitive Neuroscience: An Introduction* (Blackwell)

Kandel E R, Schwartz J H and Jessell T M 1995 *Essentials of Neural Science and Behavior* (Appleton & Lange)

Knudsen E I, du Lac S and Esterly S D 1990 Computational maps in the brain *Neurocomputing 2* ed Anderson J A *et al* (MIT Press) chapter 22

Kohonen T 1997 *Self-Organizing Maps* (Springer)

Marshall J A and Gupta V S 1998 Generalization and exclusive allocation of credit in unsupervised category learning *Network: Computation in Neural Systems* **9** 279–302

Marshall J A 1995 Adaptive perceptual pattern recognition by self-organizing neural networks: context, uncertainty, multiplicity, and scale *Neural Networks* **8** 3 335–62

Mountcastle V B 1998 *Perceptual Neuroscience: The Cerebral Cortex* (Harvard University Press)

Nigrin A 1993 *Neural Networks for Pattern Recognition* (MIT Press)

Oja E 1995 PCA, ICA, and nonlinear Hebbian learning *Proceedings of the International Conference on Artificial Neural Networks* (IEE Press) pp 89–94

O'Leary D D M 1993 Do cortical areas emerge from a protocortex? *Brain Development and Cognition: A Reader* ed Johnson M H (Blackwell) pp 323–37

Olshausen B A and Field D J 1996 Natural image statistics and efficient coding *Network: Computation in Neural Systems* **7** 333–9

Ritter H, Martinetz T and Schulten K 1992 *Neural Computation and Self-Organizing Maps: An Introduction* (Addison-Wesley)

Rumelhart D E and Zipser D 1985 Feature discovery by competitive learning *Cognitive Science* **9** 75–112

Shepherd G M (ed.) 1990 *The Synaptic Organisation of the Brain* (Oxford University Press)

Sirosh J and Miikkulainen R 1994 Cooperative self-organization of afferent and lateral connections in cortical maps *Biological Cybernetics* **71** 66–78

Somogyi P and Martin K A C 1985 Cortical circuitry underlying inhibitory processes in cat area 17 *Models of the Visual Cortex* ed Rose D and Dobson V G (Wiley) chapter 54

Sur M 1989 Visual plasticity in the auditory pathway: visual inputs induced into auditory thalamus and cortex illustrate principles of adaptive organisation in sensory systems *Dynamic Interactions in Neural Networks : Models and Data* ed Arbib M A and Amari S (Springer-Verlag) pp 35–51

Swindale N V 1996 The development of topography in the visual cortex: a review of models *Network: Computation in Neural Systems* **7** 2 161–247.

Thorpe S J 1995 Localized versus distributed representations *The Handbook of Brain Theory and Neural Networks* ed Arbib M A (MIT Press) pp 549–52

Yuille A L and Geiger D 1995 Winner-take-all mechanisms *The Handbook of Brain Theory and Neural Networks* ed Arbib M A (MIT Press) pp 1056–60

Yuille A L and Greynacz N M 1989 A winner-take-all mechanism based on presynaptic inhibition feedback *Neural Computation* **1** 334–47