Modification of Internal Representations as a Mechanism for Learning in Neural Systems

Ken Kangda Wren

University College London

A dissertation submitted for the degree of Doctor of Philosophy at Physiology Department, UCL, London University

1999

Dedicated to my parents
and Laura,
without whose support, this research would not
have been completed.

Declaration:

This project has been carried out under the supervision of Dr. A. R. Gardner-Medwin, whose original idea prompted this research. Except where explicit reference is made, the material contained in this dissertation is the result of my independent research, and is, to the best of my knowledge, original.

Acknowledgement

I would like to thank Dr. A. R. Gardner-Medwin, who has demonstrated to me how disciplined investigations should be carried out. His integrity also serves as an example for one's conduct in life. I am extremely grateful for his sensibility and patience, as well as the freedom that he has given me in this research.

Description of Thesis

Title: Modification of Internal Representations as a Mechanism for Learning in Neural Systems.

1. Incoming sensory signals are processed by hierarchically organised modules in the brain. In certain contexts, this may be modelled by a feedforward layered network of interconnected binary units. The activity patterns in the intermediate layers are *internal representations*.

2. A new learning algorithm uses projections from the desired output to modify internal representations. Biologically realistic 2-layer synaptic rules can then be applied to cause the associated input to evoke the modified representation(s) that are more readily trained to produce the target output.

3. Simulation is carried out on benchmark tasks for 3-layer feedforward networks. Comparisons with other popular algorithms are made. The results suggest that the new algorithm has better generalisation performance with faster or equal learning speed on the tasks simulated.

4. The learning algorithm is generalised to a multi-layer network setting, in which internal representations are dynamically constructed.

5. The above will be put into the context of efficient sensory coding that is based on Barlow's 'redundancy reduction' proposal.

Chapter 1 Introduction

The study seeks to gain insights into sensory representation and learning mechanisms in the brain with the aid of computer simulation of networks of artificial neurons. A new learning algorithm for a certain class of networks will be proposed and investigated.

This chapter introduces a novel approach to thinking about learning, which underlies most of the investigations in the thesis. Learning, in most models, including those considered here, is assumed to be brought about by changes in synaptic weights. But the effects of learning are more usually discussed in terms of the resultant internal representations (i.e. the patterns of cellular activity that arise from the stimuli), and how these representations relate to the learning objective. This perspective can be constructive simply because the activity patterns are readily observable variables, more so than weight changes. The starting point of this thesis is the suggestion that explicit changes of internal representations, with the objective of achieving representations that make learning easier, may in fact be built into a learning algorithm.

The Reverse Activation (RA) algorithm, the main subject of the thesis, derives from thinking about what is a desirable change of representation and how this may be achieved. Part of the learning process can then be described in terms of the modification of internal representations. Weight changes are still associated with the mechanics of learning. However, unlike in most conventional models, the step that drives the weight changes is the explicit decision on the desired internal representation.

After this general approach has been set out, a review of both relevant biological issues and related theoretical approaches will follow, before the RA algorithm is analysed with both theory and simulations.

Section 1.1 Biological Basis of Standard Network Models

The term neuron is used repeatedly in the thesis to refer to abstract neurons used in artificial network models. A biological neuron has more complex behaviours than the stereotypical abstract neuron. Radically different types of neurons exist in the brain. Further, each of the brain's regions is a vast network of distinctive sub-networks of neurons. In contrast to this complexity, most network models have a simple architecture consisting of identical units that are essentially summation devices coupled with a transfer function. Despite these differences, there are many reasons for accepting such networks as relevant to the study of the brain.

Despite their diversity, most types of biological neuron can be seen as computing devices that receive inputs and generate outputs that are characterised by the frequencies of the action potentials generated. The behaviour of such cells is largely based on a single parameter, the soma membrane potential, (cf. Amit, 1989). Hence, a network of identical abstract neurons is a useful idealisation of real networks in the brain. Further, there is no intrinsic argument to suggest that the artificial neural net will be any more 'real' if all the known properties of neurons are incorporated. Firstly, some neuronal properties are vital, and some are presumably of little consequence to the global properties of a neural net, but known properties are not necessarily more important than the unknown properties. Secondly, it is always necessary to prioritise and leave out properties that may not be important to one's modelling purpose. A network of simple neurons is only a first order approximation of the biological one, but there is seldom good reason to think that what it can achieve could not be achieved by real neurons, nor vice versa - that what it cannot achieve could be achieved by real neurons.

At an appropriate scale, the organisation of neurons in the brain is fairly uniform. The same simple architecture found in one locality (e.g. the retina) may appear elsewhere also (e.g. the olfactory bulb) (Shepherd, 1974). Further, perhaps more importantly, in many parts of the brain (e.g. association areas of neocortex), the architecture seems to be governed by simple rules, with large numbers of neurons or functional groups of neurons forming connections specified largely by global conditions such as layering, and of cell and synaptic densities. The study of simple networks seems likely to be important for the understanding of local functions, as well as large scale organisation in the brain.

Section 1.2 Standard Network Formalism: The Weight-Centric Approach

Most artificial networks, be it recurrent or feedforward, are pattern associators: they associate an output activity pattern to an input activity pattern by way of system dynamics as determined by the architecture and the weights of connections. They are particularly useful in understanding associative memory and feature detection in sensory pathways. To focus the argument, let us concentrate on multi-layered feedforward networks.

Almost all learning algorithms for this type of network can be derived from the mathematical idea of *hill climbing in weight space*: the network performance is measured by some explicit analytic function of the current weight matrices in the net;

learning involves iteratively changing each weight according to its effect on the performance function for the purpose of optimisation. Learning and generalisation are thus reduced to *interpolation* and *extrapolation*: the network represents a particular model (in the sense used for Statistical Inference), where the weights are the adjustable parameters. Any such learning algorithm is a way of computing the 'best fit' parameters. Weights are the fundamental variables in this formalism, while activity patterns (on intermediate layers) are somewhat incidental; that is, such learning models lose nothing if the significance of these patterns is completely disregarded. However, these activity patterns attract great interest because they correspond (or at least the individual elements of them do) to the most important observable in neurophysiological studies.

This approach gives a simple mathematical formalism to the learning problem, and leads to many different learning algorithms, the most popular of which is Back-Propagation (BP). One disadvantage of the approach, apart from being a rather rigid view of learning, is that most of the derived algorithms, such as BP, are not biologically plausible (see Section 3.2.2). Further, it is unsatisfactory that representational patterns, despite being a primary variable in neural science studies, are peripheral in these models of learning.

Section 1.3 An Alternative Proposal: The Pattern-Centric Approach

The algorithm proposed in this study represents a departure from the above framework; it will be referred to as the *Reverse Activation (RA) algorithm*. The algorithm takes activity patterns, rather than weights, as the fundamental variables in learning. Learning, to a large extent, becomes a matter of *actively* changing *internal* *representations* (defined as the activity patterns on the intermediate layers). Weight changes are still the intermediary, but they are driven by the goal of achieving a chosen modified representation via local Hebbian type synaptic rules. The key question is what constitutes an improved internal representation.

In the context of feedforward networks, the ideal representation for a novel input would be one that leads, with no weight changes, to the generation of appropriate outputs. Novel inputs are only likely to produce such ideal representations if there is a remarkable correspondence between the information processing in a network and the characteristics of the world that govern what are appropriate outputs for particular input patterns. More realistically, the existing representation of a novel input will not be ideal, but may be improved by altering the input processing so that fewer weight changes are required between representation and output in order to generate appropriate patterns.

The objective of directly manipulating internal representations may seem like a pipe-dream. The basic mechanism proposed however is very simple. It is suggested, on intuitive grounds, that the pattern created at the level of a representation by combining input activation (driven from the sensory input pattern) with reverse activation (driven from the paired output pattern) will generally be an improved representation for learning the input-output pairing, better than the representation produced by the input activation alone given existing weights. The weight matrix through which this reverse activation operates is obviously critical for this strategy to work, and consideration of this matrix will be an important subject of the thesis. But intuitively it seems plausible that weights based loosely on prior associations between activity patterns in internal representations and outputs may have the desired effect. If so, then the processes of learning a new input-output pairing can be split between two sites: (1) the input connections to the representation, which learns to generate an improved representation, and (2) the output connections from the representation, which more easily learn to generate the correct output. Several questions arise and will be addressed: can the suggestion be analysed theoretically; to what extent could it benefit learning; and how can the necessary conditions be arranged?

There have been earlier attempts at a pattern-centric approach, particularly the so-called CHIR ("choosing internal representations") (Grossman et. al. 1988; Grossman, 1989; Nabatovsky et. al. 1990; Abramson et. al. 1993); also see (Domany et. al. 1995). These proposals rely on active search in a vast table of potential internal representation patterns. Some other versions of CHIR (Rujan, Machand, 1989; Mezard, Nadal, 1989) take a more explicitly geometric approach, which still amounts to a 'home-in' mechanism in the high dimensional representation space in order to determine the 'appropriate' representations. Further, the final number of hidden units and hidden layers in the solution found is uncertain, and there is no guarantee that the trivial solution (i.e. one exclusive hidden unit for each input-output pair) would not emerge, see (Domany et. al. 1995) for instance.

Unlike the above CHIR's, the RA algorithm does not rely on a time-consuming explicit search in the representation space. Instead, it iteratively modifies existing representations via biological mechanisms. The weight-based algorithms such as BP also iteratively modify internal representations, but only as a by-product of weight modification. *The important difference in RA is that weight changes are driven by changes in internal representations, while in BP the exact opposite happens.*

The RA algorithm raises a problem because it is not obvious how to study it analytically since it is not based on any easily expressible optimisation procedure. As often happens in this area, computer simulation is necessary for validation. The presented simulations will concentrate mainly on comparing the RA algorithm with the back-propagation (BP) algorithm. Both algorithms are tested on two benchmark tasks, and are compared on the basis of learning speed, generalisation, and the ease of parameter tuning. The result shows that on tasks tested, the RA algorithm has better learning speed and generalisation performance. The latter is consistent with known theories on generalisation. It will be argued that the very mechanism for improving internal representations in the RA algorithm promotes better generalisation performance. RA is outlined and studied in Chapter 4, 5 and 6. A population search technique may be applied to the RA algorithm to improve its practicality, as discussed in Chapter 6.

Section 1.4 Relationship to Broader Theoretical Issues

The pattern-centric approach to learning is readily related to broader issues in the study of the brain. Crudely speaking, the subject of information processing in the brain can be studied at the system level or at the neural (network) level. The former concerns overall characteristics and complex functions of the brain, and offers explanations in terms of information and computational theories. The latter concerns the implementation or the manifestation of system level theories in terms of computational algorithms that can be justifiably described as being 'neural-network', based on known biological and physiological evidence. A complete understanding requires comprehension at both levels. Ideally, one formulates computational theories, which then can be seen at work in a neural network context; conversely, one can hope that a particular discovery at the neural network level has a certain higher level rationale.

The RA algorithm concerns the neural level. However, it naturally relates to two higher level issues: learning and efficient coding. In Section 6.5, these topics will be discussed, in particular, the concept of 'redundancy reduction' and how it leads to compact coding, factorial coding and sparse coding strategy. The RA algorithm provides an arena for studying the effect of sparseness on learning in feedforward networks of more than 2 layers.

The RA algorithm implicitly requires a short-term memory for paired patterns, independent of the representational changes that will eventually be brought about, contributing to long term memory. It therefore touches on the issue of memory consolidation in the brain. Temporal storage is required for at least the most recent activity patterns on each layer so that conditions can be set up for creating and learning the improved representations. Quite different mechanisms, possibly in different sites, may be involved as an intermediate step to the consolidation of the long-term memory, which could be modelled as the inter-layer weights. Both high quality transient memory and the ability to re-generate patterns of activity without related sensory stimuli (in imagination, rehearsal, dreams, etc.) are in fact prominent features of the nervous system, whose functional role is not clear. This adds to the plausibility and interest of the mechanisms of the algorithm.

Under this model, short-term memory assists the formation of efficient internal representations that are part of long-term memory. It is also possible to model the opposite interaction, in which long-term memory facilitates short-term recall, using the same network architecture. For this purpose, a network of hierarchically arranged layers, with the inter-layer connections functioning as long-term memory and the within-layer connections functioning as short-term memory, may be used. The RA algorithm can be generalised to such a network; see Section 4.4.

Chapter 2 Biological Background

It is helpful to review the biological reality behind the theoretical speculation ahead, for motivation, context and perspective. The chapter may be skipped by readers familiar with the subject. Section 2.1.1, 2.1.3, 2.1.4 and Section 2.2.1 contain standard facts/theories based mainly on Shepherd (1974) and Nicholls, Martin, Wallace (1992).

Section 2.1 Neurons

2.1.1 Membrane Potentials

Most neuronal behaviours stem from the selectivity properties of channels on the cell membrane. Some channels may be open only to cations, some to anions. While most anion channels are non-specific, cation channels may be specific to, for instance, potassium, sodium or calcium. Ionic channels are usually gated. The selectivity and gate mechanisms are responsible for the electrical signals generated within the nervous system. Various mechanisms can cause ion channels to change states thereby disturbing the established equilibrium and pushing the membrane away from its resting state. Some channels respond to chemical signals such as neurotransmitters, some to membrane deformations due to mechanical forces, and still others to the membrane potential itself. These mechanisms provide the means through which neurons respond to stimuli and each other.

The properties of K^+, Na^+ channels and their active transport can account for the resting membrane potential. This, usually –70 mV, is the potential that governs the concentration differential inside and outside the neuron for species of permeant ions for which there is no active transport (mainly Cl^-); the equilibrium achieved is dynamic. Sodium action potentials, stereotyped cycles of rapid membrane depolarisation and repolarisation lasting up to 2 milliseconds, result from the properties of voltage gated Na+ and K+ channels and occur in an all-or-nothing fashion.

If a depolarising potential raises the local membrane potential sufficiently, the sodium channels on that patch will open rapidly, but transiently. The increase will cause a sudden influx of Na^+ ions since sodium is much less concentrated inside the cell. The local membrane potential then will shoot up to typically +40mV within 0.5-1 millisecond. Potassium permeability also responds to increase in membrane potential, though its reaction is slower but more persistent, lasting several milliseconds. The resulting persistent outgoing potassium current will drive the membrane potential rapidly down, even to below the resting potential for a time, causing the so-called refractory period, before the resting potential is restored, thus completing the cycle, known as an action potential.

The local effects caused by an action potential induce depolarisation of the neighbouring membrane patches, which automatically undergo the same cycle; the induced action potential is exactly the same but for its location and timing. Further, because of the refractory period at the end of each depolarisation, the induced action potentials travel in a clean wave away from the initial patch and do not re-excite membrane areas that have recently undergone an action potential. The speed and range of this conduction are limited not only by the channel properties, but also by the diameter and insulation (myelination) of an axon. Dendritic action potentials are typically longer lasting and involve Ca2+ rather than Na+ entry.

2.1.2 The Frequency Code

Because the action potential is all-or-nothing and self-reproducing through the use of local energy stores, it provides the basic means of long distance communication in a biochemical environment, where reliable communications via passive flow of analogue electrical signals are possible only on a scale measured in tens of micrometers.

Since the action potential generated down an axon is exactly the same as the original action potential, there is no transmission loss. However the all-or-nothing dependence of action potential on stimuli also means that no information is conveyed in the time course ('shape') of the potential. It is the event itself, or more precisely, the number of action potentials in a given period, which carries information. This is called *frequency coding.*

Given the time scale of an action potential of the order of 1 millisecond, one may divide time into 1 millisecond intervals so that there is either 1 action potential generated or none. The frequency code can therefore be represented as a sequence of 1 and 0's. The upper limit of transmission rate is around 1000 bits/second. However, neurons on average fire less than half of the time, and there is correlation between firing intervals. These redundancies alone place the upper limit at about 500 bits/second. One may expect further redundancies implemented in order to counter noise.

If the input signals to a neuron have a measurable information content, one may devise experiments to measure the transmission rate of the neuron. This is usually only possible for sensory neurons or low level cortical neurons. In this capacity, sensory neurons of insects and frogs can transmit information at a rate ranging from 60 bits/second to 300 bits/second (Bialek *et. al.*, 1991). Current evidence from mammalian lateral geniculate neurons indicates a rate no more than 30 bits/second (Tovee *et. al.*, 1993).

The above approach is not adequate for studying information processing in the cortex. Each cortical neuron can receive signals directly from as many as 10^4 other neurons, only a small fraction of which are sensory afferent signals. It is seldom clear exactly what information is conveyed to and by a particular neuron, and information about most aspects of sensory stimuli are probably conveyed in a population code, spread across many neurons.

There is perhaps a deeper reason why cortical neurons must be analysed differently. Cortical neurons are not merely encoders that transmit information: there is no homunculus waiting to analyse the information. The population of cortical neurons as a whole is in some sense the 'end-user' of sensory information. The point of interest is not so much how a neuron encodes the incoming information and passes it on, but how it responds to the incoming information (relayed to it by lower level neurons). If one accepts that mental activity is a collective phenomenon made up by the individual responses of cortical neurons, then the activity pattern across cortical cell populations becomes a primary concern in this context. Thus, as one moves into the cortex, one stops focusing on the details of the frequency code adopted by an individual neuron, but on how sensory information is represented by the activity patterns of cell populations; the concept of 'population code' or 'internal representation' becomes the theme. We shall address these issues further in Section 2.2 and 2.3.

The abstract neuron used in modelling cortical functions is often assumed to be binary, i.e. it either fires at the maximum rate in a discrete time interval or does not fire at all. This no doubt is a caricature of the real situation. However, binarised activity is one way to combat noise, which should be useful to the biological brain. It is a natural extension of the frequency code.

2.1.3 Synapses

The states of ionic channels on the cell membrane, and therefore the membrane potential, can be altered via a variety of mechanisms. Sensory neurons respond to direct mechanical (pressure) and physical (light, odour) stimulation. Most neurons including sensory neurons also receive direct electrochemical stimuli from other neurons, so that signals can be passed on, enhanced, modulated, and transformed from neuron to neuron. A *synapse* is a physical point of contact through which such interactions take place. At a synaptic site, the gap between the membranes of two cells ranges from 20-300 Angstrom, or 2-30 nanometers across, depending on the nature of the synapse. By far the most common and more sophisticated synapses are chemical synapses. They are strongly directional. The postsynaptic cell can act on the presynaptic cell via the same synapse but generally not in the same manner as the forward action.

Chemical synapses rely on neurotransmitters to change the postsynaptic membrane potential. It takes time however for *vesicles*, *little parcels of neurotransmitters,* to be released, to diffuse across the synaptic cleft, and to take effect. The delay between the pre- and post-synaptic potential is typically 0.5 to 1 millisecond. Of the delay, only about one-tenth can be accounted for by diffusion. The rest of the 'long' interval is mainly due to the fact that to release the vesicles, Calcium must be present. It has been found that the direct effect of the presynaptic potential is mainly the opening of Ca^{2+} channels, through which extracellular Ca^{2+} ions flow inwardly.

Ample experimental evidence has demonstrated the quantal nature of transmitter emission. Neurotransmitters are released in multiples of a quantum. Each quantum is capable of eliciting a miniature postsynaptic potential (PSP) of certain amplitude. The total PSP depends on the number of quanta released. The probability of a quantum being released upon the arrival of a presynaptic potential is constant; each release is typically statistically independent. These assumptions explain the observed statistics of fluctuations in postsynaptic potentials very well.

One striking fact of the vertebrate nervous system is that the mean number of quanta released per presynaptic impulse by synapses in the central nervous system can be as much as 300 times lower than those in the periphery (such as neuromuscular junctions). However, the probability of release per presynaptic impulse can be as high as 0.9 in the central system. This dramatic difference in the mean quantal content is merely an indication that the central nervous system is concerned with the integration of information so that no one synapse has a dominant effect.

Once arrived at the postsynaptic membrane, some neurotransmitters act by directly activating appropriate ion channels. Many transmitters act by indirect mechanisms: they combine with receptors that are not ion channels themselves. The resulting substance then either is acted upon by other intracellular messengers, or acts directly, to modify the activity of other receptors, ion channels or ion pumps, thereby changing the membrane potential. Indirect synapses are usually slower.

One important empirical principle concerning synaptic arrangements is Dale's law, which states that a neuron can manufacture only one type of neural transmitter. Note that this *does not* imply that a presynaptic neuron can exert only excitatory or inhibitory influences upon all of its postsynaptic cells. The actual sign of the effect of a transmitter depends on the receptors, different types of which may exist in the postsynaptic cells. By combining transmitter and receptor mechanisms, chemical synapses exhibit extraordinary flexibility.

2.1.4 Synaptic Integration and Plasticity

A cortical neuron can receive as many as 10^4 convergent synapses. The effect of an individual synapse is rarely enough on its own to generate action potentials in the postsynaptic cell. The overall activity of the postsynaptic cell is the result of the interplay between inputs from many convergent synapses.

The efficacy or strength of a (chemical) synapse usually refers to the size of the resulting post-synaptic potential (PSP) and the length of synaptic delay, for a given 'amount' of presynaptic stimulation. Efficacy can vary both in the short term and in the long term. These variations can be due to either pre- or post-synaptic mechanisms. By altering the efficacy of synapses, a neuronal system may be able to learn.

It is tempting to assume that the PSPs of all synapses are integrated by a simple numerical summation, and that the efficacy of a synapse, which is modifiable, can be seen as the weighting factors in the sum; as the system learns, these 'weights' will be modified in some way as a result of repeated pre- and post-synaptic activities. This is, broadly speaking, what standard artificial neural network theory assumes, partly because other alternatives are difficult to handle. This simplified picture of neuronal computation is used extensively in network modelling (discussed in Chapter 3). Presently, let us compare this picture with the current knowledge of biological synaptic integration and plasticity.

Intracellular recordings show that the collective effect of simultaneous stimulation, i.e. spatial integration, crucially depends on the relative positions of synapses. Even when the two synapses virtually overlap, their simultaneous effect may differ from the numerical sum of the separate effects. As far as spatial integration is concerned, those synapses nearer the axon hillock seem to be more effective than those further from it. When timing is considered, the above already complicated picture gets worse. A welltimed inhibitory PSP (IPSP) further down the axon/dendrite can kill off an excitatory PSP (EPSP) very effectively. In addition, even at a single synaptic site, repetitive stimulation may enhance PSPs by virtue of temporal integration, with each PSPs adding to the falling phase of the one before; this happens when the frequency of stimulation is high enough (which is possible since PSPs have a much longer time course than action potentials).

Is the plasticity of synapses any simpler to capture in modelling? Experimental evidence does support the basic idea that synaptic strength can be modified. In invertebrates (Leech and Aplysia), short-term and long-term synaptic changes have been extensively studied, and can directly account for modifications in the animal's behaviour. However, the detailed modification prescriptions in various learning models such as those introduced in Section 3.2 are difficult if at all possible to verify.

Studies do provide quite a detailed qualitative picture on how synaptic efficacy is modified. Profound biochemical and morphological changes are involved. Most synapses, direct or indirect, are regulated by a second chemical messenger system. The system is activated by sufficient depolarisation, or by sufficient presence of substances accompanying specific pre- or post-synaptic events. Once activated, either the presynaptic terminal or the postsynaptic terminal, or both, will undergo changes. Chemical messengers may trigger the production of proteins that will increase or decrease the mean quantal content of transmitter release by the presynaptic terminal; they may also effect morphological changes, e.g. the presynaptic terminal may increase in size. There are also messengers that act on the postsynaptic dendrite so that it becomes more sensitive to transmitters because of newly available receptors, e.g. previously 'locked' ion channels can be now activated by the transmitter; postsynaptic morphological changes may also take place.

One can see clearly from the above that plasticity is a complex phenomenon that involves a long sequence of biochemical events. As such, it lends itself to regulation by many potential mechanisms. There is increasing evidence that indeed even the plasticity of a synapse is regulated. This is termed as metaplasticity, that is, a modification of the synapse that manifests itself not as a change in the synaptic efficacy, but as the change in the ability of the synapse to change its efficacy (Fischer et. al., 1997). The biological utility of metaplasticity is intuitively appealing (locking and unlocking of memory storage capacity for instance).

To conclude our brief review, while there is sufficient evidence to show that synaptic integration and plasticity do not conform to the simple form assumed in many network models, present evidence does suggest that biological synaptic integration and plasticity tend to be more sophisticated and thus potentially more powerful.

Section 2.2 Basic Characteristics of the Cortex

2.2.1 Cortical Layering and Columnar Organisations

The neocortex has six layers, compared to the three layers of archicortex and the four to five layers of paleocortex. The grey matter of the cortex, where most neuronal cell bodies lie, is about 2 mm thick, and covers the entire cortical surface. Wrapped inside is the white matter, which contains mostly fibres between cortical regions and glial cells. Sensory and subcortical efferent and afferent fibres are a small fraction of all the fibres in white matter. The input and output fibres enter and leave any cortical region through the depths.

Common to most cortical areas are the vertical arrangement of afferent/efferent fibres and the strong vertical orientation of axons/apical dendrites of cortical neurons in each layer. The contrast between the perpendicular and horizontal organisation is striking. In the perpendicular direction, the cortex is highly organised into layers; each layer is characterised by its cell and fibre content. Horizontally, i.e. within each layer, neurons and fibres are distributed more or less isotropically and homogeneously.

By columnar organisation one refers to the fact that neurons along a line perpendicular to the cortical surface have similar receptive field and response properties. In other words, they appear to be involved in the processing of the same bits of input signals. Physiological and anatomical evidence both put the diameter of such columns at 30-500 micrometers. Neighbouring columns are sharply demarcated from each other: they either have distinct receptive fields or have different response properties (e.g. responding to blue rather than red; responding to tactile signals rather than auditory). However, connections between neighbouring columns appear to be rather non-specific compared to vertical connections within a column. Available evidence supports the idea that the activation of one column has a non-specific inhibitory effect on the neurons in nearby columns but a small non-specific excitatory effect on those further on; this is particularly true for pyramidal cells. This fact has inspired the winner-takes-all coding strategy, which has many interesting applications; see Section 3.3.

2.2.2 Localization of Cortical Functions

Amongst the earliest investigations of the brain are those concerned with the association of functions to specific locations. In particular, it is found that visual, auditory, olfactory, and somotosensory sensations are all localised to distinct regions of the cortex; there are also the motor cortex for motor control, association cortex for integrated memory and the prefrontal cortex for planning and other complex executive functions. Each of these areas is divided into subregions. Such divisions correlate well with Brodmann areas, which are based on morphology (Shepherd, 1974) (Nicholls, Martin, Wallace, 1992).

The properties of neurons in each functionally uniform area are spatially ordered as well (Shepherd, 1974) (Nicholls, Martin, Wallace, 1992). For example, in the area 1 of somotosensory SI cortex, which receives tactile information, the receptive field of a neuron varies systematically with its position on the surface of the cortex so that the cortical surface contains a topographical map of the body. Such a map can be found also in areas of motor cortex. Similarly, topographical maps of visual scene are found in the visual cortex, and tonographical maps in auditory cortex.

On the one hand, all cortical areas have the same basic cell compositions, the same basic 'circuitry'}, and the same coding strategy (i.e. topographical representation) but on the other hand different areas of the cortex specialise to perform different functions. The inevitable questions are why and how.

It is relatively easy to explain how localisation is implemented. To obtain sensory information of different modalities, different physical/chemical processes must be utilised. For instance, sensory neurons that detect pressure are very different from those that detect odour. Hence at the detection level, the nervous system must have 'localisation'. Functional specialisation in the cortex thus might be seen as a simple consequence of physical/chemical necessity, and would be a direct result of well-designed carefully-specified sensory innervation. The sensory innervation argument however cannot account for the sharp demarcation observed between the receptive fields of neighbouring cortical columns, the basis of topographical maps. It has been demonstrated instead (Kohonen, 1990) that lateral inhibition can achieve topographical maps even when each neuron receives exactly the same inputs; cf. Section 3.3.

One important advantage of localisation, which partially answers the why question, is that it keeps the brain at a reasonable size. Most of the brain volume is filled with myelinated axons that link different parts of the brain. The number of cells is proportional to the surface area of the cortex, so the number of axons would be roughly proportional to the square of the surface area. Thus given a fixed average volume of an axon, the volume occupied by axons alone would increase as the square of the surface area. Further, as the volume increases, the average volume for axons would also increase due to longer lengths and consequently necessarily bigger cross-sections and thicker myelin sheets. Hence the volume would actually go up more than proportionally to the square of the surface area. Had the human brain, with its surface area and average connection probability, been a mass of cells which are uniformly randomly connected, its volume would have been enormous (Mitchison, 1992).

Chapter 3 Neural Network Models

Artificial neural network models attempt to incorporate some of the above qualitative biological characteristics of neurons and their interactions into quantitative terms so as to carry out more concrete investigations. Inevitable in this process some biological realism must be sacrificed in order to draw upon useful mathematical tools. The validity of this trade off is ultimately justified or refuted by the results.

In what follows, we shall deal with networks of simple formal neurons (see e.g. Amit, 1989). They are based on two basic assumptions: 1) sub-threshold excitations lead to no activity; and 2) at any instant, a neuron receives an input that is the linear sum of all inputs from individual input synapses; the weights in the summation correspond to the efficacy of each of the modifiable input-synapses.

Section 3.1 Rosenblatt's Simple Perceptron

3.1.1 The Basic Architecture

The most basic network of formal neurons consists of binary units arranged in two layers: the input (I) and the output (O) layer. Most concepts in neural networks are best illustrated in this simple context. The matrix W_{IO} of modifiable weights specifies the connection strengths from a cell in I to a cell in O layer so that given input pattern P_I , the activation A_O to output cell O_i is given by

$$A_O(i)=\sum_j W_{IO}(i,j)P_I(j)$$

The activity of the cell O_i is binarised according to

$$\Theta\,(A_O(i)-\theta\,(i)\,)$$

where Θ is a step-function, and θ is a modifiable threshold. Note that this threshold can be absorbed. Since

$$A_O(i)-\theta\,(i)=\sum_j W_{IO}\,(i\,,j)P_I(j)-\theta\,(i)=\sum_{j'} W'_{IO}\,(i\,,j')P'_I(j'),$$

where W'_{IO} is W_{IO} with $-\theta$ (i) listed as an additional column, and where P'_I has an additional unit that is 1 (i.e. always 'on'). Such individually adjustable thresholds will not be explicitly mentioned from now on.

The above network is able to associate an output pattern with a given input pattern. The detailed relation depends on the status of the forward weights.

3.1.2 The Learning Procedure

The perceptron training procedure is as follows (Rosenblatt, 1962). To learn a specific association task, a sample set of input-output pairs is prepared. These patterns are then presented to the network one by one. For each input, bit errors of the network-generated output pattern relative to the target pattern are then noted. If there is no error, then no weight modification takes place; one proceeds to the next pair of patterns. If there are errors, some modification for each weight is computed. Connection weights are changed iteratively. That is, when they are modified, the modifications are small. However, the training set is presented repeatedly and hence the modifications are done repeatedly. One complete presentation of the training set is called an *epoch*. The small weight changes in each epoch accumulate as the cycles go on, until no modification takes place, i.e. until all mappings are achieved. Learning time is measured by the number of epochs required. Most neural networks and learning algorithms follow the above training procedure.

The perceptron synaptic rule is a particular way of calculating the necessary weight modifications. It can be derived from performing gradient descent on the mean-square error-surface over all mappings in the training set. According to this rule, the change $\Delta W(i, j)$ to weight $W(i, j)$ is given by the following

$$\Delta W(i, j)= \lambda\,(O^T{}_i - O_i)I_j\,, \qquad \textbf{\textit{(Eq. 3.1)}}$$

where $O^T{}_i$ is the target output activity at cell i, O_i is the current output activity at cell i evoked by the input, and I_j is the activity of the input cell j in the input pattern, and λ is some positive constant called *step-size*, small compared to the size of the weights. Note that the step-size is the quantum of weight change in the binary setting and is also referred to as the *learning rate.*

There is a legitimate concern over whether the above synaptic rule is biologically plausible, particularly regarding the availability of error signals $(O^T{}_i - O_i)$ at the presynaptic sites. Gardner-Medwin suggested (in private correspondence) one way of interpreting the rule (Eq. 3.1). Note the prescribed modification, $\Delta W(i, j)= \lambda\,(O^T{}_i - O_i)I_j$, can be separated into two stages: one of anti-learning i.e. forgetting while the internally generated output is on, as suggested by the term $-\lambda O_i I_j$, and one of positive learning while the target output is on, suggested by the term $+\lambda\, O^T{}_i\, I_j$. Each individual stage is simple Hebbian or anti-Hebbian associative learning, which is arguably the most biologically plausible synaptic rule. Anti-Hebbian-learning or active forgetting has been suggested in many different contexts and seems to be important to understanding many phenomena in memory and learning; see (Crick, Mitchison,1983; Hopfield *et. al.* 1983), or (Dormany *et. al.* 1995; Hassoun 1996). For experimental evidence of the existence of anti-Hebbian synaptic modification see for example (Bell *et. al.*, 1990).

In the perceptron algorithm, there are 3 principle options for when a computed weight modification may be implemented. The two most commonly used methods are *on-line* updating and *batch* updating.

In the former, each weight is updated according to a prescribed formula (such as Eq. 3.1) immediately following the presentation of *each* pattern in training set. In the latter, the training set patterns are learned as a whole: each weight is updated only after all the input-output pairs are presented, and the modification is given by the sum of all the required modification calculated from individual patterns. The third less well-known updating procedure, proposed and referred to here as *'total-on-line'* for convenience, is a training procedure in which individual pairings in the training set are learned completely, one at a time: weights are modified iteratively till the latest input-output mapping is learned perfectly before the next input-output mapping is presented. This 'perfect learning' is of course at the expense of possibly damaging the mappings already 'perfectly' learned before the presentation of the latest mapping.

There are subtle differences between these three methods (Finoff 94; Hassoun, 1995; Ripley 96; Saad, Solla, 1996). Briefly, the on-line method is the most volatile and sensitive to step-size. The total-on-line method can be the most stable with respect to step-size. The batch method is somewhere in between in this respect, and it is also most mathematically sound but least biologically plausible. These will be explained in more details. For small 2-layer perceptrons, such distinctions are less important as far as performance is concerned.

Given the on-line or total on-line training procedures, the perceptron convergence theorem states that if a solution set of weights exists for the problem, then the net will converge to a solution with the Pereceptron synaptic rule. The convergence of the batch procedure is guaranteed by gradient descent (to be discussed shortly).

3.1.3 Limitations

A well-known point about the simple perceptron is that it cannot learn any set of mappings that are not linearly separable (Minsky, Papert, 1969). Notice that each output unit classifies the input patterns into two categories: those that turn it on, and those that do not. Input patterns can be represented as points in a vector space, in fact, as the corners of a hyper-cube. The above categorisation is geometrically represented by a plane separating the two sets of 'corners'. This plane is in fact parameterised by the weights onto the output unit concerned. If the mapping problem is such that the implied categorisation by an output unit is not achievable by any plane, then it is not linearly separable. Since no planes means no solution weights, the simple perceptron cannot learn such a problem.

Section 3.2 Multi-layer Perceptrons

To resolve the above problem, extra layers of units can be introduced between the input and output layer. Typically, 3-layer networks are studied, which with a sufficiently large number of intermediate cells can learn any well-defined binary mapping, if necessary by employing cells that individually detect specific input patterns. Additional internal layers usually do not enhance the computational capabilities, though they may permit an economy of cells.

Consider a network of 3 layers, the input layer (I), the hidden layer (H), and the output layer (O), with numbers of binary cells (in the 0-1 representation) equal to N_I, N_H, N_O respectively. Let us call any non-linear operation that turns an activation pattern into a binary pattern a *binarisation* operation for short. This may involve thresholds that are either fixed or, for example, adjusted to achieve a particular number of active cells. All input cells project to all hidden cells, which in turn project to all output cells. Call the I⇒H weight matrix W_{IH}, and the H⇒O weight matrix W_{HO}. The patterns on the H layer are referred to as *internal representations*.

3.2.1 Pattern-Centric vs. Weight-Centric Learning Strategies

There are two ways to extend the basic perceptron learning procedure from 2-layer networks to 3-layer networks. The pattern-centric way is to devise an algorithm that establishes a pattern P_H^* on the H layer that is desirable as a new internal representation and to apply the perceptron rule to W_{IH} directly so that input pattern P_I comes to evoke P_H^* instead of its initial representation P_H. The weight changes are divided so as to achieve two new mappings: $P_I \Rightarrow P_H^*$ and $P_H^* \Rightarrow P_O$, where P_O is the target output pattern. The perceptron rule can be applied at each stage. An alternative (weight-centric) way would be to invoke a global output error function that can be differentiated against each connection weight, thereby determining its appropriate modification to achieve gradient descent. Both of the above can be regarded as generalisations of the 2-layer perceptron learning procedure.

As discussed in Section 1.3, the pattern-centric strategy is intuitively appealing. It puts internal representation at the very heart of learning and processing, as it should be. Human learning experience lends support to any learning strategy that *actively* constructs internal representations of the external environment (in an effort to accomplish a task). This approach will be carried forward in Chapter 4 and the following chapters. The weight-centric strategies on other hand treat internal representation as a passive by-product. Unfortunately, past attempts in the pattern-centric direction have been largely unsuccessful. The so-called CHIR (Section 1.3) relies on cumbersome prescribed search mechanisms in the high-dimensional space of potential representations and remains unattractive in practice.

3.2.2 The Standard Gradient-Descent Algorithm

Gradient descent prescribes that the appropriate weight change for each connection should be proportional to the negative of the partial derivative of the chosen error function with respect to that connection. As such it is a very general strategy, and is adaptable to many learning environments (including networks with stochastic neurons).

The most popular Back-Propagation algorithm (BP) has a mean square error function. We shall discuss BP for illustration. There are other less popular but well-known algorithms proposed for such multi-layer feedforward networks of continuous neurons. They are all gradient descent methods of one form or another. The main difference was in the error functions used: cf. for example (Peterson *et. al.*, 1989), which contains the so-called Boltzmann Machine type algorithms, which do stochastic gradient descent on an entropy measure. For more examples, consult (Hassoun 1995; Ripley 1996).

The Back-Propagation Procedure

On a standard feedforward, deterministic network with the usual quadratic error function, the basic gradient descent prescription reads as follows:

$$\Delta W(i, j) \text{ (due to one training pattern)} = \lambda \sum_k (O^t_k - O_k)\, \partial O_k / \partial W(i, j).$$

This is the central element of the so-called back-propagation (BP) algorithm. However, in order to apply gradient descent to binary networks, it is necessary to turn the binary neurons into ones with continuous activity during learning. Let the transfer function of each neuron be f. It is customary to choose f to be

$$f(A) = \tanh(\beta A), \qquad \textbf{\textit{(Eq. 3.2)}}$$

where A denotes activation, and the parameter β evidently controls the 'sharpness' of the transfer at A=0: it is called 'steepness'. As it goes to infinity, the transfer function is essentially a thresholding function taking ±1 depending on the sign of the activation A. Originally, $(f(A)+1)/2$ is used so that the activity level is between 0 and 1. However it is a well-known rule of thumb that using *tanh* rather than the shifted version improves learning speed in simulation by 30-50%. This has been confirmed by many studies, see for example (Stornetta *et. al.*, 1987; Peterson *et. al.*, 1989).

Since for a 3-layer network, the output is given by

$$O_k = f(\Sigma_i W_{HO}(k, i) f(\Sigma_j W_{IH}(i, j) I_j)),$$

applying gradient descent gives the following learning rule:

$$\Delta W_{HO}(k, i) = \lambda\, f'(A_O(k))\, (O^t_k - O_k)\, H_i$$

$$\Delta W_{IH}(i, j) = \lambda\, I_j\, f'(A_H(j))\, \Sigma_k f'(A_O(k))\, W_{HO}(k, j)(O^t_k - O_k) \qquad \textbf{\textit{(Eq. 3.3a)}}$$

Note that the rule for the H ⇒ O weights reduces to the perceptron rule in the binary limit with f' regularised by the step-size λ (the difference in $f'(A_O(k))$ from cell to cell is eliminated in the limit). However it is not possible to turn the rules into binary form for both H⇒O weights and I⇒H weights at the same time. This is because in the second part of the rule, it is the square of f' that appears. As one takes the binary limit, it is thus impossible to keep both $\lambda f'$ and $\lambda(f')^2$ finite but non-zero by adjusting λ. Also, the ability to absorb f' into the step-size means that steepness β and step-size λ are not independent parameters. In fact, the behaviour of the network remains unchanged if one scales the steepness β to 1 and scales the learning rate by β^2 and all initial weights by β (Thimm et. al. 1996).

Following the above training, the internal representation on H layer(s) can sometimes be interpreted in a neural context. The biologically controversial part of this learning method is the second half of the rule (Eq.3.3). Note that the weight change for synapses between I- and H-layers requires information that is only available on the O-layer, namely, the information about the error and the H$\Rightarrow$O weights. For this reason, this algorithm is given the name *Back-Propagation* since it is evidently necessary to somehow propagate the information from the output layer to successive layers all the way back to the layer immediately above the input layer. Further, the nature of the error signals (containing derivatives and so on) is such that it is not easily coded by the activity of cells. Some independent memory must be associated with each cell in order to retain and transmit such information.

The algorithm has been applied extensively due its generality and mathematical simplicity. There has been extensive investigation into this algorithm. Its properties are by now well-known. Below is a brief summary. Detailed survey of the state of BP research can be found in (Hassoun, 1995), and (Ripley, 1996) also contains useful insights.

A Brief Review of Performance Properties

Learning is usually slow and unstable when the rule is applied in its basic form. Arguably the single most effective method of improving convergence is the use of a *momentum term,* cf. (Rumelhart, *et. al.* 1986; Hassoun, 1995). The idea is that each required weight modification has a lingering contribution in all subsequent modifications, but the contribution decays as α^n, where $0<\alpha<1$ and n is discrete time. That is

$$\Delta W_{n+1} = \lambda E_n + \alpha \Delta W_n, \quad 0<\alpha<1 \qquad \textbf{\textit{(Eq. 3.3b)}}$$

where ΔW_n is the weight modification for a connection at step n, λ the step-size, E_n the error correction to that weight calculated according to some learning algorithm, such as in (Eq. 3.3a).

Note that any learning algorithm can be supplemented by momentum smoothing, regardless of the details. The algorithm in use, what ever it is, calculates the weight modification required for the current step according to that algorithm. The momentum term simply allows the weight modification carried out in the previous step to make a weakened contribution also.

Momentum smoothing results in large modifications in flat regions of the error surface, and prevents over-shooting in a rugged terrain, thereby making convergence more reliable. Usually learning is not sensitive to the precise value of α as long as it is not too small or too close to 1 (Rumelhart, *et. al.* 1986; Müller *et. al.*, 1991; Hassoun, 1995); for detailed investigations in the context of gradient descent/BP algorithms, see (Tugay *et. al.*, 1989; Tollenaere, 1990). There have been proposals of self-adapting momentum terms (Fahlman, 1988). However, the learning rule becomes extremely cumbersome and seems even more remote from biological reality than ordinary BP.

Regardless of the modifications above, performance is sensitive to step-size. For fixed parameters, performance is slow when step-size is excessively small, but unstable when step-size is too big, and achieves optimum for an appropriate range of intermediate step-sizes. In general, to speed up learning further, larger steps are needed at the beginning but increasingly smaller steps are necessary for convergence.

Update schedules can affect convergence speed also. In classical BP, as in (Rumelhart *et. al.* 1986), to be consistent with the mathematics of gradient descent, the *batch* updating schedule must be adopted. As the gradient of the error function depends on all patterns in the training set, the required weight changes are only known after a complete presentation of the training set. This evidently is unrealistic in a biological context, as the 'training set' in the real world may be indefinite, changing, and may contain many redundant examples. Further, each connection weight is modified only once every epoch in this strict gradient descent scenario, which seems excessively cautious. Thus, the *on-line* updating schedule is often suggested for BP. It is found that on-line updating approximates stochastic gradient descent if the step-size used is vanishingly small; however, there is no essential difference between this infinitesimal on-line and finite batch updating procedures (Finnoff, 1994). Using *finely-tuned* finite step-size, the on-line method may but does not always improve learning speed when the training set contains a large number of redundant (same or similar) examples and when there are local minima in the error function; the quasi random character of on-line updating gives an 'annealing'-like effect (Finnoff, 1994). However, on-line updating, unlike the batch method, *cannot* converge *unless* step-size is gradually reduced eventually to zero as (learning) epochs go by (Ripley, 1996). It is thus more volatile and tends to oscillate if step-size is not tuned and scheduled correctly. How this is done is a matter of trial and error. Many heuristics have been proposed, though all are computationally expensive and none definitive. There are adherents to either the batch or the on-line method, but there is as yet no conclusive evidence to favour either method. The total-on-line method (Section 3.1.2) seems not to have been studied with BP, and some observations will be discussed in Section 6.4.

Finally, BP, like other gradient descent methods, is very sensitive to initial weights, i.e. where the system starts gradient descent matters greatly. Further, if initial weights are too big, units are likely to be saturated (i.e. close to either of the two extreme activity levels), which make learning impossible or slow. The usual practice is to normalise random initial weights so that they fall within $\pm 3L/N^{1/2}$, where N is the number of training patterns, and L is the typical length of the input pattern, cf. (Hassoun, 1995); this simple normalisation can improve learning speed. Note that weights are thus expected to grow with $N^{1/2}$, and that periodic normalisation is necessary if learning is on-line with no defined training set. Such normalisation would destroy past knowledge since the activity of a BP network depends continuously on the weights. It is hard to reconcile this with biological reality.

Section 3.3 Self-Organising Networks

3.3.1 Supervised and Unsupervised Learning

Multi-layer perceptrons belong to the class of networks that do *supervised learning* in the sense that they are trained with a specific set of input-output mappings. Another class of networks is designed to achieve, iteratively, as their output, a particular type of representation for a given input population. The training procedure usually involves presenting sample inputs randomly selected from the population; weights are modified following each presentation. The modification algorithm is such that weights will converge so that continued presentation will no longer lead to any change. The resulting activity patterns on each layer, associated with each input, are then regarded as the internal representations, which can be interpreted as achieving feature extraction.

Note however that the difference between supervised learning and unsupervised learning is not fundamental. The representation achieved through unsupervised learning is useless unless it can facilitate the implementation of some learning goal. This goal, in abstract language, is a set of defined mappings from the input population to a certain output population. The representations achieved through unsupervised learning may go some way towards achieving this overall mapping if the representations selected by the unsupervised algorithm render the mapping problem more readily solvable as a 2-layer problem. To be constructive in this way, the weight training algorithm of an unsupervised network must implement valid assumptions about the statistical structures of the input population and their relation to the likely learning goals.

From this point of view, supervised learning (on a three-layer net) merely makes the implicit goals explicit, while relying entirely on output errors to drive the creation of appropriate internal representations on the hidden layer.

3.3.2 Competitive Learning Strategy

One of the most widely used and versatile unsupervised learning algorithms is *winner-takes-all* or *competitive* learning (Amari, Arbib, 1977); see also (Hassoun 1995). The architecture of the network is the same as the simple perceptron except that the input cells are continuous so that the input patterns can be any real vector. However, given any input pattern P_I, the activity at the next level is given by

$$O_i = \Theta\,(\Sigma_j W_{IO}\,(i\,,j)P_I(j) - \text{Max}\,\{\Sigma_k W_{IO}\,(i\,,k)P_I(k)\,;\,i=1,2,...N_O\}).$$

That is, only the one with the largest activation is allowed to be on. This function can be implemented biologically through lateral inhibition.

In a training situation, *only* weights onto the winner cell w are modified according to

$$\Delta W_{IO}(k_w, i) = \lambda\,(P_I(i) - W_{IO}(k_w, i)). \qquad \textbf{\textit{(Eq. 3.4)}}$$

Had λ been 1, the new weight vector would simply be the input vector. On average, when λ is small, and the sampling of the input population extensive, the cell concerned would tend to become an encoder of a group of inputs clustering close to each other. Due to the exclusive nature of the winner-takes-all rule, each cell will become sharply tuned to a particular cluster, thus serving as a detector for that cluster. The input weight vector onto each cell is therefore a prototype (cluster centre). By creating prototypes, a substantial amount of correlation in the input population is eliminated. The network can be seen as a classifier, which *discovers* the categories (clusters) as it samples the input populations.

3.3.3 Kohonen Network

The competitive algorithm simply represents one strategy that appears to be important for the brain to adopt in order to eliminate the most common type of redundancy that exists in our natural environment, namely, local correlation resulting from the continuous nature of most properties. The cortical topographical representations of body surface or retinal positions can be reproduced by this coding strategy. This is explicitly demonstrated by the Kohonen network (Kohonen, 1990)

It is a type of *soft* competitive learning algorithm. The network consists of continuous neurons such as the ones above. The architecture is still the two-layer perceptron one: an input layer and output layer with full forward connections between the layers. Training is exactly like that for ordinary competitive networks, but instead of having only the weights onto the winner cell w modified as in (Eq. 3.4), the modification rule is modified by a neighbourhood function $N_w(k)$ to become $\Delta W_{IO}(k, i) = \lambda\ N_w(\ k-k_w)$ $(P_I\,(i)\ -W_{IO}(k, i)\ )$, where $N_w(\ k-k_w)$ is positive, with maximum value 1 and declines to zero with distance $k-k_w$, the distance from the winner cell *w*. When the network is large, one can approximate the output layer by a continuous line or continuous sheet. Then, the above becomes, $\Delta W_{IO}(p, q) = \lambda\ N_w(\ p-p_w)\ (P_I\ (q)\ -W_{IO}(p, q)\ )$, where p, q are coordinates on the output and input 'sheet' respectively (just like the j and i labels in the discrete case). The neighbourhood function may be chosen as the symmetric Gaussian centred at 0 (so that it is maximum at p_w).

Remarkably, the above algorithm is capable of producing topographical representations of the input space such as those observed in the cortex. In particular, each cell in the resulting network shows a well-defined receptive field that is sharply demarcated from that of neighbouring cells, despite the fact that all output cells receive the same input signals.

Section 3.4 Homogeneous and Hierarchically Organised Attractor Networks

3.4.1 Autoassociative Attractor Networks

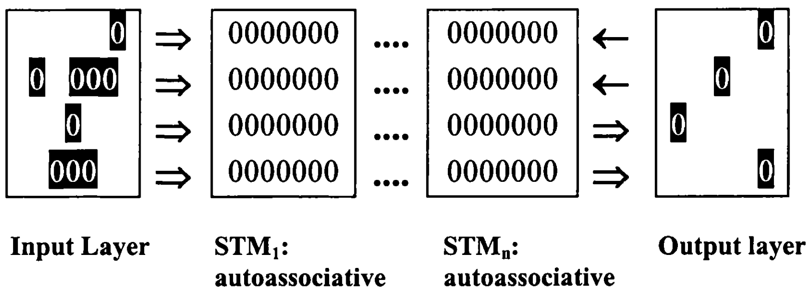
So far we have discussed networks that have feedforward connections only. These networks must be driven externally, with no dynamic interaction. This need not be the case if there are loop connections. In general, let us consider a uniformly connected network of N (binary) neurons, with connectivity R, i.e. on average each cell projects and receives projections from RN cells. For such a network it is no longer natural to see it as layered (unless there are functional differences in the connections. In the homogeneous case we simply need a N×N weight matrix to describe the network. Once an input pattern is fed into such a network, activities can be sustained without being driven externally and the activity pattern will evolve in time and can settle into a previously experienced state (Marr, 1971; Gardner-Medwin, 1976; see also Willshaw & Buckingham, 1990).

A convenient way to study the dynamics is to see the net as a point meandering its way in the state space of the network. (A network state is the collection of instantaneous states of all cells in the network.) A trajectory is completely determined by the initial state and the weight matrix. There are usually fixed points in the dynamics. The network will settle in such a state once it is reached. What is relevant are those fixed points that are robust, called *stable states*. That is, following perturbation, the network is capable of returning to and staying in those states. The set of states starting from which the network will reach a given stable state in finite time is called the *basin of attractions*. The parallel between such dynamics and the act of recall is self-evident. It is fair to say that any system with a reasonably rich dynamics containing numerous stable states (and cycles) can be used to model a memory. This is the basic idea that has been popularised by (Hopfield, 1982).

Uniformly connected networks, called *autoassociative* or *attractor networks* have been extensively studied by physicists because it is amenable to thermodynamics; a comprehensive exposition to this field can be found in (Amit, 1989). These techniques reveal the essential properties and limitations of such a system as a model for biological memory. The main results are that they have limited capacity, relatively fast convergence; they are poor at storing and recalling non-orthogonal patterns (i.e. patterns with lots of overlaps), though there are algorithms that can diminish this problem (Gardner-Medwin, 1989; Gardner-Medwin & Kaul, 1995). This kind of network, incorporated into a hierarchical network, may be appropriate for modelling short-term memory (STM).

3.4.2 Hierarchically Organised Attractor Networks

The following hybrid structure is commonly proposed, e.g. (Marr, 1971; Amit, 1989).



It is a hierarchically organised multi-layer network, with each layer an autoassociative network. In addition to the forward connections between layers, there are also backward connections from higher level layers to lower level ones. For the above structure to be distinct from a purely autoassociative structure, one must assume that the there are functional differences between the internal connections within each layer, the forward inter-layer connections, and the backward inter-layer connections. The three classes of weights may behave differently and play different roles. For instance, the autoassociative layers in the above structure may model short-term memory (STM) while the inter-layer connections, long-term memory (LTM). The activity patterns on each layer can be induced in part by extrinsic connections and in part by connections within.

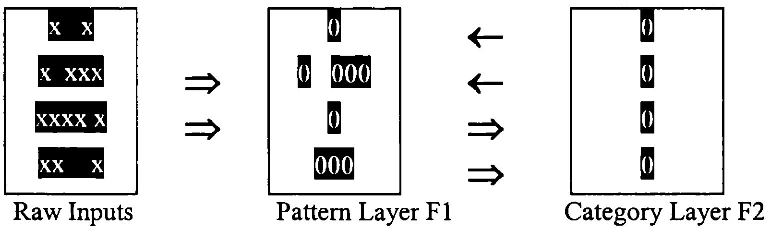
Examples of this type of networks include the following.

Bi-direction Associative Memory (BAM)

BAM (Kosko, 1988) is perhaps the simplest in this class. It is essentially a 2-layer perceptron with symmetric connections that run in both directions. Within each layer, the connections are trivial (i.e. none). As such, it is merely a Hopfield associative memory with incomplete connections.

The ART Network

ART, Adaptive Resonance Theory (Carpenter, Grossberg, 1987), is an unsupervised network which consists of two bi-directionally connected layers F1, the pattern layer, and F2, the category layer. Each cell in F2 is a category node and only one can be on at any time (due to lateral inhibition). The connection from F2 to F1 is such that the 'on'-node can turn on, in layer F1, the 'prototype' pattern of the category that the node represents, in the absence of other influences to F1. The connections from F1 to F2 have modifiable weights that can be changed in case the category assigned to a pattern by these weights needs to be changed. Graphically, the ART network is as follows, where an input layer to F1 is added for later discussions.



Given a pattern on F1, a category node on F2 will be chosen via the forward F1-to-F2 connections. This node will tend to evoke the prototype of that category on F1 via the backward F2-to-F1 connections. A tuneable "vigilance parameter" will decide whether the prototype is sufficiently close to the existing pattern on F1. If it is, then the forward and backward connections become a positive feedback loop: a resonance will be established and all connections will be reinforced. If it is not, an alternative category will be assigned (by suppressing the failed node) to see if resonance can be established. If all existing categories fail, a new category will be created with its prototype as the existing pattern on F1. Once resonance is established, all weights will be modified to promote the new category arrangement. In a steady state of the network, whatever the pattern on F1 is, a resonance can be established. That is, in a steady state, any pattern on F1 and the category node it evokes constitutes a fixed-point of the dynamics, under the given "vigilance" level.

The Wake-Sleep Network

The Wake-Sleep model (Hinton et. al. 1995) is a multi-layer, unsupervised network of stochastic model neurons. Every adjacent pair of layers in the network is connected by top-down and bottom-up connections. Bottom-up “recognition” connections convert inputs into representations in successive hidden layers, and top-down “generative” connections reconstruct the representation in one layer from the representation in the layer above. The top-down and bottom-up weights are trained separately in two distinct phases. In the wake phase, neurons are driven by bottom-up, recognition connections, and the top-down, generative connections are trained to increase the probability that they would reconstruct the correct activity patterns in the layer below. In the sleep phase, neurons are driven by top-down connections, and bottom-up connections are trained to increase the probability that they would produce the correct activity in the layer above.

Chapter 4 Theoretical Aspects of the Reverse Activation Algorithm

Before we launch into a detailed justification and analysis of the RA algorithm, we shall first outline the procedures and issues involved, as well as their relation to other network models with backward connections. The terminology established in Chapter 3 for 3-layer feedforward perceptron will be used throughout. In particular, assume that each output cell has a 'backward' connection to each H-cell.

Section 4.1 An Overview of the RA Algorithm

4.1.1 Fundamental Steps in the RA Algorithm

The algorithm is a pattern-centric algorithm. That is, to learn to map input pattern P_I to output pattern P_O on a 3-layer feedforward perceptron (as defined in Section 3.1 and 3.2), the algorithm first constructs an internal representation pattern P_H^*. Then the perceptron learning rule (or some other valid rule) is applied on the $I \Rightarrow H$ weights and on the $H \Rightarrow O$ weights to attempt to achieve the P_I to P_H^* mapping and the P_H^* to P_O mapping respectively. It thus breaks down the 3-layer problem into two 2-layer problems.

Note that the 2-layer learning need not be carried out to completion, i.e. P_H^* need not be achieved completely. Weights are only modified one step at a time and learning is stopped as soon as the P_I-to-P_O mapping is achieved. In other words, P_H^* may not be the final internal representation adopted by the network, it merely provides a target to motivate the $I \Rightarrow H$ and the $H \Rightarrow O$ weights to move in the right directions.

The key is therefore how P_H^* is constructed. The RA algorithm uses the following procedure, which requires a reverse connection matrix from the O to the H layer, to be discussed later.

1) Impose the input P_I and output P_O patterns simultaneously on the I and O layers respectively;

2) Compute the combined activation pattern on H layer

   $A_H^{\Psi}=W_{IH}\,P_I + \psi W_{OH}\,P_O$ ***(Eq. 4.1)***

   where W_{IH} and W_{OH} denote the weight matrices from I⇒H, H⇒O, and O⇒H; and ψ is a pre-set, non-negative number called the *reverse activation strength.*

3) Produce binary activity pattern P_H^* by way of the following:

   P_H^* (j)=1 only if A_H^{Ψ} (j) is one of the top W activation amongst all $j=1,2,\ldots N_H$,

   where W is a pre-set number, fixing the activity ratio of the internal representation.

4.1.2 The Key Elements and the Biological Plausibility of RA

The Reverse Activation Matrix

The Two fundamental questions arise about the reverse activation matrix W_{OH}. What determines the individual weights, i.e. the form of the matrix W_{OH}? And how is its overall effect, i.e. the reverse activation strength (ψ) modulated?

The basic requirement for the W_{OH} is that it should be an adequate inverse to the forward matrix. That is, on the 2-layer network of the H and the O layer, given any output pattern, the reverse matrix should be capable of producing a pattern on H which produces the output itself via the forward matrix. This is because the purpose of the reverse connections, when activated by a desired output pattern, is to shift the activity on the H layer towards a pattern that will reproduce the output pattern via the forward matrix. The best choice of W_{OH} in fact appears to be the transpose of W_{HO}, as discussed in Chapter 4.

Once learning of a set of I-O mappings has taken place, only the forward connections are taken into account in assessing learned performance. The reverse connections may be able in principle to contribute to improving the quality of an output pattern through dynamic interplay of the H and O layers during recall, but this would take time to settle and only the correctness of a learned output on the first step of such a dynamic process is actually considered here.

Reverse Activation Strength

The reverse activation strength ψ is an important tuneable parameter for the RA algorithm. It is needed partly to counteract arbitrary scaling of the $I \Rightarrow H$ weights relative to the $H \Rightarrow O$ weights. But its more fundamental importance is to control how much the new internal representation is to differ from the one generated by the input and the existing weights. Note that when ψ is 0, the modified representation coincides with the existing representation. And when it is infinite, the chosen representation is completely determined by the output and the $O \Rightarrow H$ weights. In between, it regulates the relative contributions of the paired input in determining the internal representation. Another way to see it is that ψ decides how learning is 'shared' between the $I \Rightarrow H$ and

H$\Rightarrow$O connections: when $\psi=0$, the input representation remains unchanged and learning is entirely carried out on W_{HO}; whereas with $\psi=\infty$, much of the learning involves changes in W_{IH}, which may result in a new representation that requires little if any change to W_{HO} to produce the desired output.

In the initial simulations of RA, ψ is fixed prior to training and remains fixed throughout the epochs. It is necessary to try out different values to determine the optimal range (rather like tuning for optimal step-size or momentum in BP). An alternative version selects ψ randomly from a pre-determined range prior to each superposition of inputs and outputs, so that ψ changes every time it is used. The advantage of the latter is that it obviates the need to tune ψ. It is interesting that this seems to work almost as well as employing a constant and optimal ψ.

How could ψ be modulated in a biological context? Two possibilities are through effects of diffuse neuromodulators and, perhaps more simply, by varying the strength with which the desired output pattern is activated. The latter mechanism strictly contravenes the simplifying assumption made in the model that neurons are binary, but it is of course quite feasible with more realistic neurons that have variable firing rates

Binarisation and Activity Ratio

The binarisation procedure is quite crucial in the construction of the internal representation. It is done by fixing the activity ratio of the H layer. Then any activation pattern is binarised by allowing only the few most-activated cells to be 'on'. Why is this necessary?

The problem of constructing a binary internal representation comes down to determining which cells should be 'on' or 'off'. The object of the construction procedure must be to include the 'good' cells (whatever that means) and exclude the 'bad' ones. RA amounts to saying that the way to measure 'goodness' is via the 'combined activation' defined in the (Eq. 4.1). As such the absolute value of the activation of each cell has little relevance in determining if a cell should be included in a representation or not because the activation is subject to arbitrary scaling. It is the relative order of activation that matters to the RA construction procedure.

As a result, fixing the activity ratio of the H and O layer is inevitable so that only the top few cells are allowed to be 'on'. This is referred to as ramped binarisation. This makes the activity ratio on the H layer a tuneable parameter, providing a perfect opportunity to study the effect (on performance) of different activity ratio constraints for internal representations. As such, the RA procedure is a way of solving a given mapping task by constructing internal representations of a given activity ratio.

This binarisation procedure is also applied during recall on both the H layer and the O layer, for consistency. The behaviour of the network is more robust as a result.

A Non-gradient Descent Method

One key difference between RA and BP or other gradient descent methods is that the internal representations constructed are not driven by output-errors. The input and output mappings alone determine directly what the appropriate internal representations should be. Not having a defined error-surface in which to descent, it is hard to study the method analytically. For instance, it is not clear why the process should converge let alone learn anything at all.

Generalisation to Multiple Layers

RA can be generalised straightforwardly to networks of the type described in Section 3.4.2. Internal representations on successive layers are determined by the fixed points of the dynamics resulting from the bi-directional linkage between the layers, keeping the input and output layers clamped. In Section 4.4, it is proved that such fixed points always exist and explained how this is consistent with the RA for 3-layer networks.

Biological Plausibility of Assumptions

The process of improving representations on the H layer is essentially a matter of recruiting 'better' cells for the purpose of generating the desired output and dropping 'bad' cells. One could look on this as analogous to learning to notice features of an input that lead you to the right conclusions about it, and learning to ignore features that lead to the wrong conclusions, based on previous learning. The criterion for 'good' cells is that they are strongly activated from (and by inference associated with) the correct output as well as the input, using a suitable reverse weight matrix. In fact the reverse matrix adopted for the simulations (the transpose of the forward weights) is likely to be one of the more simple to establish biologically, since the reverse connection between cells O_i and H_k is the same as the forward connection between the same cells, and this might be expected on the basis of simple associative (Hebbian) synaptic modification. Reciprocal connections from higher level centres are very common in the brain (e.g. Mumford, 1991,1992; Lee et. al. 1998), though their properties in relation to forward connections are not generally known.

The total number of active cells is kept fixed in the RA simulations ramped binarisation so that it is the ranking of the H cells that should govern which cells should be employed for a representation, not their absolute levels of activation. If there are too few active cells then the capacity of a network to represent and learn about different events is restricted, while too many active cells can lead to problems of overlap and interference. Ramped binarisation could in practice be implemented by negative feedback employing widespread recurrent inhibition set by the number of active cells.

RA makes use of the simple perceptron learning rule for weight modifications, which is a relatively plausible learning rule in a biological context, based on association (Section 3.1.2). The simple perceptron rule is not in fact essential to the algorithm itself. Any other 2-layer rule can be applied once an internal representation is constructed. It is substantially more plausible than a learning rule based on back-propagation.

The RA algorithm requires the existence of STM so that target-patterns on each layer can be repeatedly recalled to train the inter-layer weights and improve representations. Though the relationship between STM and LTM in producing consolidation is far from clear, it is evident that the nervous system contains the capability of recalling at least some aspects of the representations of recent stimuli and appropriate responses, both during waking and sleep, partly on the basis of human reports of subjective experience and partly from hippocampal animal studies (e.g. Skaggs & McNaughton, 1996).

4.1.3 RA in Relation to Other Bi-directional Models

Compared with BAM

BAM (Section 3.4.2) resembles the bi-directionally connected internal and output layer in the proposed RA network. However, the potential dynamics on these two layers, as a result of the bi-directionality, does not play any part in either training or recall in RA. During RA training, while the modified internal representation is being chosen, both the input and output layers are constrained to the input and the target output patterns. During recall, the output is defined as the result of the first forward sweep. Further, the reverse activation strength parameter, which plays a crucial role in the choice of internal representations, makes the effect of the reverse connections variable and asymmetrical. In BAM however the forward and reverse connections are treated completely symmetrically.

The key purpose of the reverse connection in RA is to allow the output play a part in shaping the internal representations of the input. In the future, one might consider giving the dynamics a role in the computation of outputs, especially for *novel* inputs, but this is not a fundamental feature of RA. Nor indeed is the symmetry of the forward and backward weights, which is a simple and approximate solution to the attainment of an ideal reverse matrix (Section 4.3.3).

Compared with the ART Network

One can incorporate ART (Section 3.4.2) into a 3-layer supervised network, while retaining the spirit of ART, to make it comparable to RA. ART's layer F1 naturally identifies with the hidden layer, receiving pre-processed inputs and activation from layer F2, which identifies with the output layer. In the forward sweep, the input is reduced to a prototype pattern on F1, which hopefully is associated with the desired category nodes on F2. ART demands that in a steady state of the network (i.e. with resonance achieved) any internal representation pattern must be similar to the prototype pattern of an output category. In other words, in any steady state the internal pattern evoked from the input alone must be always similar to the internal pattern evoked from the associated output alone. How 'similar' will depend on the level of "vigilance" chosen during learning (for very high levels, they should be the same). The RA network does not use this symmetry as a goal that drives the construction of internal representations.

However, in RA and in ART both the inputs and the required outputs play a direct role in determining the internal representations that would finally emerge through their respective learning procedure. Note that the "vigilance" parameter acts to either accept or reject the pattern evoked from the input as the internal representation depending on whether its similarity to the prototype of the corresponding output is high enough. In case of acceptance, the network learns by changing weights between F1 to F2 only. In the second case, the prototype of the target category will be the chosen internal representation and the network learns by changing the weights from the input to layer F1 only. Only in case of very high "vigilance" level, the internal representation would be solely determined by the output and the network learns by changing the weights from the input to layer F1 only. This is similar to the situation when the "reverse activation strength" parameter in RA is chosen to be very high. However, there is an important difference. The "reverse activation strength" parameter in RA provides a graded control as to how much the input (or output) will contribute to the construction of the internal representations, in other words, how learning will be shared between the group of connections linking the input to F1 and the group linking F1 and F2. In ART on the other hand, for any given input-output pair, the internal representation is either 100% input driven or 100% output driven.

Compared with the Wake-Sleep Network

The hidden and output layers in a RA network may be compared with the Wake-Sleep network (Section 3.4.2), ignoring the stochastic nature of the neurons in Wake-Sleep. Wake-Sleep, applied in a supervised fashion, would demand that the hidden-layer pattern should evoke the required pattern on the output layer and that the output pattern should be able to evoke the chosen internal representation pattern; connections in each direction are trained separately and alternately. This closely resembles the resonance requirement in ART except that the output pattern is not limited to being exclusive categories.

Without allowing reverse connections from the hidden layer to the input layer, i.e. applying Wake-Sleep only to the hidden and output layers, the internal representation that would finally emerge after learning is mostly determined by the required output. This differs from both RA and ART, in which the input also plays an ongoing role in moulding the internal representation. Further, even if one allows reverse connections from the hidden layer to the input layer and applies Wake-Sleep to the complete network, the input and output would always play an equal part in determining the internal representations. This is not the case in RA, nor in ART.

Section 4.2 Modification of Internal Representations via Reverse Connections in RA

In this section, we study what a modified internal representation should be, and how it can be generated in RA in details.

4.2.1 The Basis for Constructing Internal Representations

The 'Minimal Disturbance Rule'

The intuition behind the RA algorithm is very simple. Any modification of weights may cause interference i.e. may damage performance on mappings already learned by the network. It is desirable that *a novel pair of input and output should be learned with minimal disturbance to previous learning and hence to the existing weights.* In general, it seems likely that spreading small weight changes over a smaller number of relevant weights will help to diminish overall interference.

In order to apply the perceptron learning algorithm to both of these projections, there must be a target pattern on each of the H and O layers. The target for the O layer is clear - it is the desired output pattern. The target P_H^* for the H layer may differ from the initial P_H evoked by input pattern P_I so as to be better at eliciting the desired output with no or reduced changes to H$\Rightarrow$O weights; this reduces the necessary disturbance to the existing W_{HO} matrix. But it must also not differ too much from P_H, so that disturbance to W_{IH} is kept to a minimum. The upshot is that the target H- pattern P_H^* should be a compromise between the set of cells on H that are most easily activated from the input pattern P_I and those that are most effective at eliciting the correct output pattern P_O. The RA algorithm relies on the notion that the effectiveness at eliciting the correct output pattern correlates with the reverse activation of the H layer from the desired output, operating through a suitable connection matrix W_{OH} that can be set up in a practical manner.

Combining Forward and Reverse Activation

At the cellular level, choosing a good representational pattern comes down to deciding whether each H-cell should be 'on' or 'off'. The two factors to be considered in this decision are the cell's ease of activation by the input *and* its 'effectiveness' in evoking the desired output, given the existing weights.

Ease of activation from the input is directly available to the H cells in the form of forward activation vector A_I, where $A_I(j)=\sum_i W_{IH}(j, i)P_I(i)$, from the input layer to the H cells. A representational pattern that is easy to implement is one in which the 'on' cells already have high activation from the input.

The effectiveness of an H-cell for evoking the required output depends *only* on the output pattern and the forward H$\Rightarrow$O weights. In particular, it has nothing to do with the I$\Rightarrow$H weights and the input, much as the ease of activation has nothing to do with the H$\Rightarrow$O weights and the output. To the extent that the reverse activation correlates with this effectiveness, it influences the choice of a target representation by means of the reverse activation vector A_O from the output layer to the H cells, expressed as

$A_O(j)=\Sigma_i W_{OH}(j, i)P_O(i)$. The following diagram summarises our considerations so far.

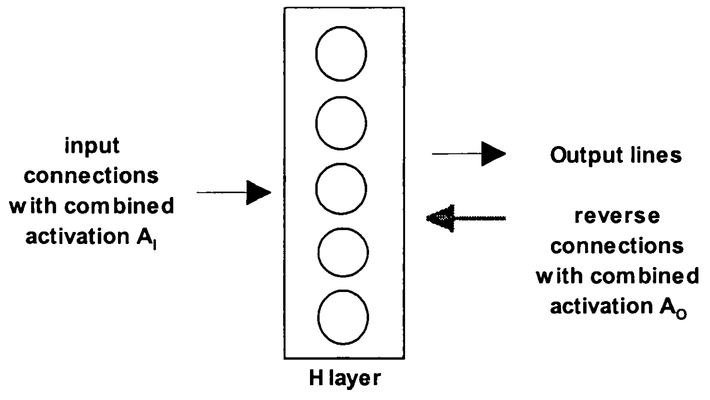

input connections with combined activation A_I
Output lines
reverse connections with combined activation A_O
H layer

Figure 4.2a. Three types of connections associated with each H cell. Every H cell projects to the output layer, and receives projections from the input as well as the output layer. If the input and the target output patterns are imposed on the respective layer, the instantaneous activation received by an H cell from the input layer is denoted as A_I , and similarly, the instantaneous activation from the output layer is denoted as A_O.

Since both the forward and reverse activations onto individual cells are relevant to their selection for a new representation, it is helpful to portray them on a 2-D scatter plot (Fig. 4.2b). Each H cell is plotted with its forward and reverse activations (A_I, A_O) as Y- and X-coordinates, respectively. We shall refer to such a plot as *the activation scatter of H cells*, usually for a particular set of input-output pairs, given the initial weights. Note that with N_H H-cells and N input-output pairs, the total number of points in the plot will be $N_H \times N$ (so each H cell appears N times).

It is important to understand this plot because it illustrates the learning process and the thinking behind RA. This will be explained here in an informal manner, begging for the moment the question of how it may be valid to treat reverse activation (on the X-axis) as equivalent to effectiveness for generating the corresponding output. A strong positive correlation in the scatter predicts 'easy' training, since the initial weights are already such that those H cells strongly activated from the inputs also tend to be effective for turning on the desired outputs. A negative correlation indicates that the new learning task is at odds with the current weight configuration and past experience of the network: the H cells most associated with the correct output pattern are those that are poorly activated from the input. An absence of correlation would indicate independence between the past experience of the network and the new learning task at hand. The learning process can be represented through shifts in the positions of points on this plot, as indicated by arrows.

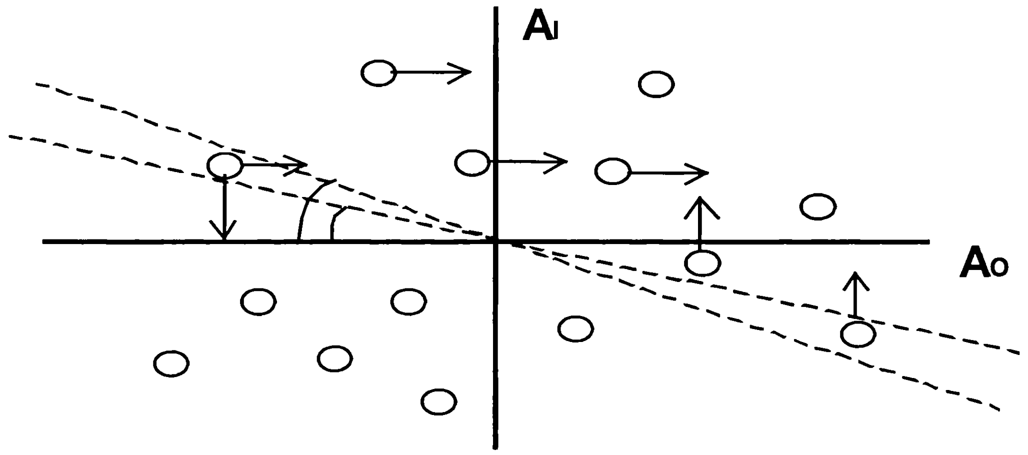


Figure 4.2b. How internal representation may be modified. Each H cell is plotted according to the activation it receives when the input and the target output pattern are imposed on the respective layer. The vertical coordinate A_I is the forward activation received from the input pattern; the horizontal coordinate A_O is the reverse activation from the output pattern. The horizontal line represents a simple threshold applied with input activation alone: those cells above this line will become the active representation. Non-zero strengths of reverse activation (Ψ) give slanted threshold lines on the diagram, with cells then activated only if they are above the slanted line. In general this leads to recruitment of H cells with high A_o and dropping of cells with low A_o. Subsequent learning on the W_{IH} matrix, with this as a target representation, leads to the vertical shifts indicated, while learning on the W_{HO} matrix (reflected in W_{OH}) results in horizontal shifts.

What is Required of the Reverse Matrix

The reverse matrix needs to be set up so that the reverse activation A_O received by an H cell from the target output pattern is a reasonable indicator of how effective the cell is in evoking the target output pattern. Putting it in another way, the reverse matrix must be such that a selection of H cells with high reverse activation will require less H$\Rightarrow$O weight change to evoke the target output pattern. Stated more formally:

> The required backward connection matrix tracks the state of the forward H$\Rightarrow$O connection matrix in such a way that *the effectiveness of any* H *cell for producing a particular output pattern is indicated by the activation received from the output pattern via the reverse connections.* If so, for an H-cell H_j, the reverse activation
>
> $$A_O(j)=\Sigma_i W_{OH}(j, i)P_O(i). \qquad \textbf{\textit{(Eq. 4.2)}}$$
>
> Can be used a proxy for its effectiveness for producing an output pattern $P_O=\{O_1,...,O_i,...\}$ given the current forward weights W_{HO}.

However, the above is not very meaningful in that it does not explicitly provide a way of testing whether any matrix fulfils the requirements: how does one know whether any particular backward matrix is adequate for this purpose?

> *Note that it is the forward matrix that ultimately determines how effective an H cell is in evoking a particular target output, so the backward matrix W_{OH} must be determined by the forward matrix W_{HO}.*

Therefore, the proper way to assess whether a matrix fulfils its role, i.e. whether it does compute 'effectiveness' when plugged into (Eq. 4.2) is to see how well the activity pattern on H-layer generated by the backward matrix actually produces the target output pattern P_O (via the forward matrix).

If the forward matrix W_{HO} has an inverse, then one may choose the backward matrix to be $W_{OH}= W_{HO}^{-1}$. For a linear network of continuous neurons, this choice of reverse matrix would give a perfect indication of effectiveness of any H cell in producing any output P_O when plugged into (Eq. 4.2) because

$$W_{HO}\, A_O = W_{HO}\, W_{HO}^{-1} P_O = P_O.$$

In other words, if H cells are activated according to their reverse activation A_O as computed by the reverse matrix $W_{OH} = W_{HO}^{-1}$, the output pattern can be produced perfectly without any change to the forward weights. In this sense, this particular choice of backward matrix W_{HO}^{-1} provides an adequate computation of effectiveness.

However, the matrix inverse may not exist; and the case for it breaks down in the presence of binarisation. A simpler and more general candidate might be

$$W_{OH} = W_{HO}^{T}, \text{ i.e., } W_{HO}(i, j) = W_{OH}(j, i) \text{ for all } i,j, \qquad \textbf{\textit{(Eq. 4.3)}}$$

where 'T' indicates 'transpose'. In this case, we simply have

$$A_O(j) = \Sigma_i\, W_{HO}(i, j)\, P_O(i). \qquad \textbf{\textit{(Eq. 4.4)}}$$

This choice of reverse matrix assumes that the bigger the total weights from cell H_j to the target 'on'-output-cells, the more 'effective' it should be. Intuitively, this seems a reasonable bet. In Section 4.3 the reverse matrix is considered further. Meanwhile, the reverse matrix can be considered as the transpose in the discussions that follow.

4.2.2 Constructing a Modified Representation

An initial representation (P_H) of the input pattern on the H layer, evoked by forward activation alone, corresponds to the cells above a horizontal threshold line in Fig. 4.2b.

The desired modification of this representation corresponds to recruitment of cells on the right of the diagram, strongly activated from the correct output and therefore associated with these output cells on the basis of past experience and likely to be effective at activating the these output cells. With a finite reverse activation strength (Ψ), the total activation onto H cells is influenced by the X co-ordinate on the diagram (A_O). A threshold uniformly applied to all the cells will then correspond to recruitment of those above the slanted line, with gradient -Ψ:

$$A_I + \psi A_O = \text{threshold} \qquad \textit{(Eq. 4.5)}$$

The desirable H cells on the right tend to be recruited and those poorly associated with the output pattern, on the left, are lost from the representation. This is the mechanism for the creation of a new representation P_H^*.

To establish the new representation P_H* from the input *alone*, the network must change the I⇒H weights. The I⇒H weight modification, using P_H* as a target, results in the vertical movement of cells on the scatter plot (see Fig 4.2b). Though it is desirable to recruit cells with large positive A_O values and to exclude cells with negative A_O values, the ease of activation (A_I) from the input is also important. Cells with initially high A_O but a very negative A_I require substantial changes to the I⇒H weights to be recruited, and may cause too much interference. This corresponds to the use of a very large Ψ, represented by a very steep slanted line on the scatter plot. Since learning can also take place on the H⇒O weights, corresponding to the horizontal movement of the cells (see Fig 4.2b) cells, the principle of minimal disturbance to the existing weights necessitates an appropriate trade-off between learning (hence changing weights) on the two sets of connections. The trade-off is regulated by Ψ. This is discussed in Section 4.2.4.

4.2.3 Setting the Number of Cells in a Representation

The new representation pattern P_H^* depends on the binarisation process used to select active cells on the basis of combined activation $A_I+\psi A_O$. A process called *ramped binarisation* is employed.

Ramped Binarisation

The absolute value of the combined activation $A_I+\psi A_O$ is employed as an indicator of how advantageous it is for a particular H cell to be 'on' in the new internal representation. However, it may happen that all H cells carry a small or negative combined input and reverse activation; this may happen if the new mapping is very different from those already learned by the network. In this case, one still has to choose the best available internal representation, given the circumstances. One is not in fact interested in the absolute value of the combined activation of each H cell, but only the relative order of H cells according to these values.

The natural way to obtain the improved representation P_H^* from the combined activation pattern $A_I+\psi A_O$ is therefore by ranking H cells according to their activation, turning on the best activated of the H cells for the new representation. Operationally, this is achieved by ramping, i.e. lowering the threshold on the H layer systematically from a high level until the desired number of cells are on: *ramped binarisation.* Ramping is easy to achieve biologically via a feedback mechanism involving controlled mutual inhibition.

From a biological point of view, the absolute value of synaptic efficacy and membrane potential are prone to many fluctuating factors. A binarised activity pattern is one possible mechanism to achieve robust behaviour against such noise. However, for this purpose, the binarisation procedure itself must be robust so that cell activity patterns are preserved as far as possible against fluctuations or scaling of weights and activation levels. Ramped binarisation meets the requirement better than ordinary binarisation procedures that are based on a fixed threshold (such as 0), essentially through feedback control.

Ramped binarisation is adopted as the normal procedure for RA. The effect on a scatter plot is illustrated in Fig. 4.2c. The slanted threshold line is moved up or down until the required number of cells is above it.

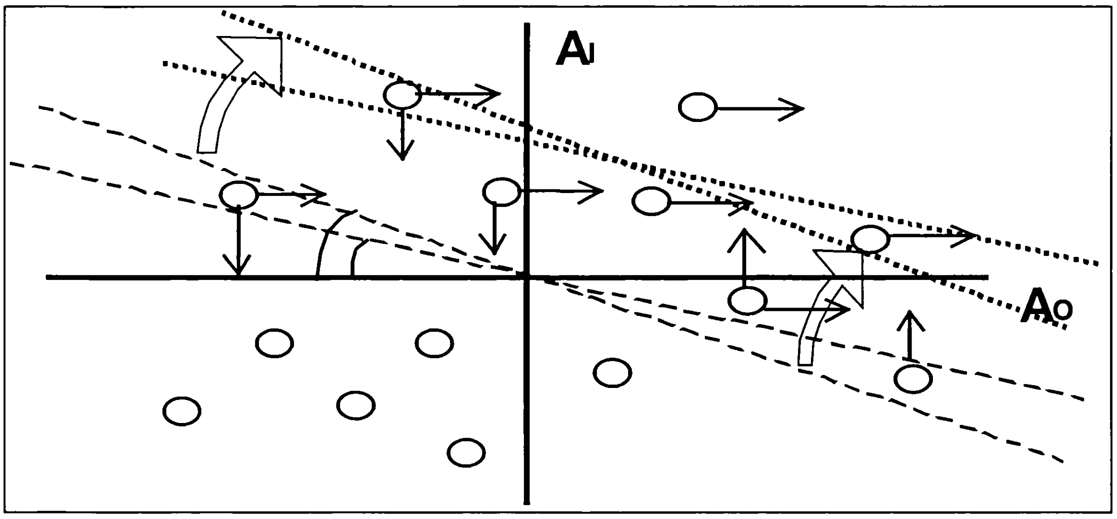


Figure 4.2c. How ramped binarisation fixes internal representation. An activation scatter for H cells, as in Fig 4.2b. The modified internal representation pattern is constructed by turning on only cells that fall above the tilted line, which is moved up or down by controlled inhibition until the correct number of cells remain above the line. Hollow arrows show examples of thresholds adjusted to give just 2 active cells in the illustration.

The use of ramped binarisation makes the absolute value of activation meaningless; only the relative value counts. Systematic vertical and horizontal shifts in the scatter plots are irrelevant, and the relative vertical and horizontal scale changes are significant through affecting the appropriate value of ψ for a given activity ratio.

The details of the scatter and the choice of activation strength i.e. angle of the tilt, determine the precise cut that gives the right number of active cells. For example, for a small activity ratio, one would need to move the tilted threshold line to the 'north-east' region of the scatter for most choices of tilting angles. The smaller the activity ratio the more 'north-east' the region has to be.

These considerations may seem academic but we shall come back to them in Chapter 5, where they have practical implications to the tuning of reverse activation strength.

The Significance of Activity Ratio

One of the direct consequences of ramped binarisation procedure is that *activity ratio* of patterns on each layer, on the intermediate layer in particular, naturally becomes an integral part of learning on binary networks. This is interesting in view of Section 6.5 where the significance of (low) activity ratio in efficient cortical representations will be discussed. By setting the strength of ramping, the activity ratios on a given layer may be fixed at any desired level (without weight changes). It is particularly interesting to consider the impact of activity ratio of internal representations on learning performance.

More detailed analysis in Section 4.3.4 reveals that activity ratio affects the potential performance of RA algorithm also in a direct way, independent from efficient representation considerations.

Generalised Competitive Learning

Ramped binarisation, pushed to the extreme with only one 'on' cell, produces the competitive learning situation, for which the ability to extract features contained in the inputs *without supervision* is described in Section 3.3. Thus the RA learning algorithm with ramped binarisation is a form of 'n-bit' competitive learning with supervision; however, the supervision is less specific than in other supervised algorithms, since the teaching signals are not in the form of specific output errors, but are the required outputs themselves. To distinguish our situation from the usual competitive learning, or from the usual supervised learning, one may call it *n-bit competitive learning with constraints (as opposed to 'with supervision').*

4.2.4 Random Tuning of Reverse Activation Strength

The RA algorithm involves the superposition of input and reverse activation with a weighting factor, the reverse activation strength ψ. The problem of how to determine the appropriate ψ is important here. See Chapter 5 and 6 (Fig. 5.8, 5.14 and 6.1 in particular) for simulations showing the proposed solutions perform in practice. Some information from these simulations is introduced into the discussion here, because it has led to development of the strategy of random tuning.

To monitor the progress of training, it is common to plot the number of correct mappings achieved at the end of each epoch against the number of training epochs that have been carried out. Typically, the performance level rises relatively quickly before flattening out to approach an asymptotic level.

For the RA algorithm with a fixed ψ value ('fixed-ψ RA') throughout learning, the observed learning curve approaches its asymptotic level very quickly. Further, it is found that for bad ψ values, learning performance settles into its asymptotic level much quicker than the performance of more successful trials (see for example Fig. 5.4).

For good ψ values, the resulting small net weight changes in the latter stage of training simply reflect the fact that most of the mappings are correctly achieved so that modifications only take place rarely. However, in the case of bad ψ values, there is certainly no shortage of opportunities for weights to change. Yet, when net weight changes at the end of each epoch are recorded, they are found to be declining quickly to very nearly 0, corresponding to performance settling into its asymptotic state. The appropriate conclusion to be drawn here is *that for bad ψ values, the modifications tend to cancel each other out, much more so than for good ψ values (see for example Fig. 6.1).*

This suggests that if ψ is allowed to fluctuate randomly, then those modifications that result from inappropriate ψ values will tend to cancel each other, while those that result from good ψ values will add up and generally move in beneficial directions. This bias will enable the network to learn positively over time, on average. This somewhat speculative conjecture is vindicated in simulations. The quality of learning is surprisingly good, comparable or better than the best of fixed-ψ RA training in some cases. This version of RA shall be referred to as random-ψ RA, which is detailed in Chapter 5.

Section 4.3 The Reverse Weight Matrix

This section considers in more detail a crucial aspect of the RA algorithm, the backward connection matrix from the output to the intermediate layer. This issue is self-contained and relates to the H and O layers only. That is, it is intrinsically a 2-layer problem.

4.3.1 Identification of the Ideal Reverse Weight Matrix

Let P_O be any pattern on the output layer. Since the output layer has N_O cells, P_O belongs to the N_O-dimensional real vector space. In general, one assumes that P_O is generated according to some process described by a probability density distribution P over the vector space. This assumption covers the situations where P_O is a Gaussian vector, or where P_O is always a binary vector, etc. Throughout the rest of the Chapter, let us assume that the distribution *P* is uniform.

Recall (Eq. 4.2) and the associated requirements for the backward matrix W_{OH}. One may place optimisation criteria on this to make the requirements more concrete.

*With respect to a given binarization mechanism **B** in the network, the backward matrix is required to be such that the expectation calculated with respect to P*

$$E\{d(P_O, P_O^*)\} \qquad \textbf{(Eq. 4.6)}$$

is adequately small, where, in matrix-vector notation,

$$P_O^* = \boldsymbol{B}_O(\ W_{HO}\boldsymbol{B}_H(W_{OH}P_O)\ ), \qquad \textbf{(Eq. 4.7)}$$

and d(,) is some distance (i.e. error) measure for binary patterns.

Let us clarify the above with the following diagram.

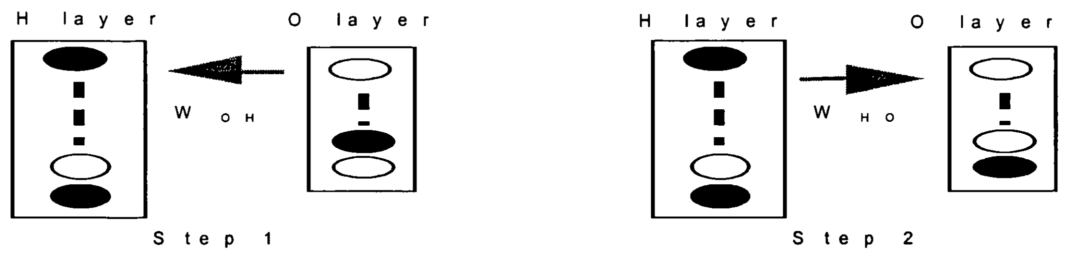


Figure 4.3 The defining property of the reverse connection matrix W_{OH}. Given any binary pattern on the output layer, drawn from some probability distribution P, the matrix W_{OH} feeds it backwards to produce a binary pattern $P_H=\mathbf{B}_H(W_{OH}P_O)$ on the H layer; this is then fed forwards by the matrix W_{HO} to produce a binary pattern $P_O^*= \mathbf{B}_O(W_{HO}P_H)$ on the O layer. The matrix W_{OH} is such that the average distance between P_O^* and P_O is small. With the appropriate choice of distance, this property should imply that the average $H\Rightarrow O$ weight modification required to evoke pattern P_O from pattern P_H, which is computed by the backward connection matrix W_{OH}, is small.

Evidently, the distance measure $d(\ ,\ )$ is the entity through which one ultimately expresses precisely what one means by the 'effectiveness' of an H cell (in producing certain pattern P_O), which determines what the reverse weight matrix W_{OH} should be.

The distance natural to the present situation is *bit error*, that is,

$$d(P_O, P_O^*)=\Sigma_i\, |P_O(i)-P_O^*(i)|,$$

because given that the synaptic modification rule is perceptron, it reflects the amount of $H\Rightarrow O$ weight modifications required in order to produce pattern P_O from pattern $P_H=\mathbf{B}_H(W_{OH}P_O)$.

How does one find a reverse matrix W_{OH} that satisfies the above? It may be instructive to note that the required reverse matrix by its definition above performs an inversion operation; the problem corresponds formally to the so called *inverse problem*, which is involved in modelling brain functions such as vision and sensorimotor control. Let us examine its relevance to the present situation.

4.3.2 Standard Inverse Problems: Inverse Optics, Inverse Models

In the context of artificial vision (Kawato *et. al.* 1993), the desired outputs (patterns on the O layer of Figure 4.3) may be the equivalent of the input images. The H layer patterns correspond to the internal representations of the visual scene, in terms of, for instance, lines, edges, colour and so on; the H layer plays the role of the visual cortical areas. It is stated that visual recall (in this artificial model) is a *forward optical problem*: it constructs something similar to a low-level 'retinal' pattern P_O, from its internal representation P_H bearing more relation to the outside world. The forward matrix W_{HO} then is the manifestation of a model of the forward optics. The earlier visual pathway $I \Rightarrow H$ in this model, performs an inverse transform (*inverse optics*), which turns a retinal visual image P_O into an internal representation P_H.

In the context of motor control, cf. (Jordan, 1990), the O layer patterns represent the actual movement required (expressed in task coordinates such as speed, joint angles, reach and so on), while the H layer patterns represent firing patterns of motor neurons. Each firing pattern P_H is transformed *forwardly* into movements by a known map such as the forward matrix W_{HO}. However, the central motor control must do the opposite: it turns a desired movement into firing patterns. That is, it performs the inverse transform that produces the appropriate firing pattern P_H from a desired movement represented by pattern P_O; this is called an *inverse model.*

In either of the contexts above, at the simplest level, one usually assumes that the cells in each layer have continuous outputs and that the problem is continuously differentiable and approximately linear. The upshot is that one ends up with the demand that the backward weight matrix W_{OH} must be such that

$$W_{HO}W_{OH}=1, \qquad \textbf{\textit{(Eq. 4.8)}}$$

where 1 is the unit $N_O \times N_O$ matrix. This is so that patterns originating on the O layer can be reproduced: in the context of vision, the reconstruction of the image from its internal representation; in the context of motor control, the *intended* movements (on the O layer in terms of task coordinates) resulting from 'motor neuron' firing patterns (H layer).

This equation amounts to N_O^2 linear equations with $N_H \times N_O$ unknowns and $N_H \times N_O$ coefficients from the known matrix W_{HO}.

In the motor control case, one usually has $N_O < N_H$ because the O patterns, which represent movements, come from a space of a much lower dimension than the space of firing patterns; in other words, many different firing patterns may achieve the same desired motor task. In this case, infinitely many matrices W_{OH} may exactly solve (Eq. 4.8); one may write

$$W_{OH} = W_{HO}{}^{-1}{}_{R} ,$$

where subscript 'R' indicates that the 'inverse' is only valid if it multiplies on the right of W_{HO}.

In the inverse optics case, it is usually assumed that $N_O > N_H$ because 'retinal' images are supposed to be reduced dimensionally (at least in the artificial setting), i.e., to be more efficiently represented by lower dimensional patterns on H. It is obvious that (Eq. 4.8) has no exact solution in this case. For technical reasons, the column vectors of the forward matrix W_{HO} are made linearly independent, which is always possible without changing any essential aspect of the underlying problem.

The usual practice is then to choose a distance (error) measure $d(\, , \,)$ on the space of image patterns P_O, and seeks an 'optimal' solution that minimises the expectation

$E\{d(P_O, P_O^*)\}$, where $P_O^* = W_{HO}W_{OH}P_O$. The most common choice of distance $d(,)$ is the Euclidean distance, that is, one seeks to minimise the average mean square error. Let us assume that the probability density distribution of the images P_O is uniform, for simplicity. The optimal solution is then given by

$$W_{OH}=(W_{HO}^T W_{HO})^{-1} W_{HO}^T; \qquad \textbf{\textit{(Eq. 4.9)}}$$

This is the *pseudoinverse* solution: the backward matrix W_{OH} is the pseudoinverse $W_{HO}^{\#}$ of the forward matrix $W_{HO.}$

Our reverse matrix problem differs from the two standard cases above in some important respects. The expectation (Eq. 4.6) contains the non-linearizable binarisation operation **B**, and is calculated over binary patterns only.

Another important problem associated with the linear continuous solutions is that they cannot be computed by local operations (not to mention that the solutions must be computed differently according to whether N_H is greater or smaller than N_O). The backward matrix must track any changes in the forward matrix in order to continue to compute adequately the effectiveness of each H cell, i.e. to maintain the (pseudo) inverse relation. If one accepts the linear continuous solutions, the backward matrix modification ΔW_{OH} required is determined by solving

$$(\Delta W_{HO})W_{OH} + W_{HO}(\Delta W_{OH})=0.$$

This is merely a set of linear simultaneous equations with unknowns ΔW_{OH} (i , j), which in general depends on every element of $(\Delta W_{HO})W_{OH}$ and W_{OH}. It cannot be computed via local synaptic rules.

4.3.3 The Transpose as a Possible Inverse Operator

Due to the above difficulties regarding the inversion problem, one is forced to seek alternative reverse matrices. Recall that symmetric connection weights are used in autoassociative memories. In particular, the transpose has been used to perform inversion type tasks for binary patterns in the so-called *bi-directional associative memory*, BAM, a bi-directionally connected 2-layer perceptron-type network (Kosko, 1988, Baum *et. al.* 1988). The transpose in this context is used to retrieve a given set of binary patterns that have been transformed by a forward matrix. It is proved that the dynamics on this 2-layer network, where the feed-backward connection matrix is the transpose of the feed-forward one, is always stable in that it always settles into a stationary state, in which the transpose is evidently an inversion operator.

One advantage for choosing the transpose as the reverse matrix for RA is that the algorithm can be generalised readily to a multi-layer setting (as in Section 4.4.) Another advantage is that local learning rule can be used to compute the transpose during learning, provided that the reverse matrix is the transpose initially. Apply Gardner-Medwin's interpretation of the perceptron learning rule as a 2-stage Hebb learning: one of anti-learning (forgetting) when the internally generated output is on, and one of positive learning when the target output is on. Then since the same Hebb rule applies to both the forward and backward connections one has,

H cell in the selected representation on H	O cell in the output pattern P_O (or P_O^c)	Forward weight change	Backward weight change
on	on	+ (-)	+ (-)
on	off	0 (0)	0 (0)
off	on	0 (0)	0 (0)
off	off	0 (0)	0 (0)

Table 4.1. Two-stage Hebb learning results in symmetric weight changes. P_O denotes the target output, which is imposed on O layer in the positive Hebb learning stage. P_O^c denotes the current, internally generated output, which is imposed during the negative Hebb learning stage.

Thus if the two weight matrices start off as the transpose of each other, perceptron training will preserve this relation. This assumes that the weight of influence of H-cells on O-cells is the same as, or proportional to, that of O-cells on H-cells.

Although, the transpose performs 'dynamic inversion' in BAM, it is not known how good it is in performing 'one-shot' inverse operations.

4.3.4 Weight Statistics, Activity Ratios and Inversion by Transpose

In the following, an investigation on inversion by the transpose matrix is presented. The conclusions are the result of certain characteristics on the weight statistics. We shall argue on intuitive grounds only that these characteristics tend to hold.

As a matter of consistency, the type of weight statistics suitable for a learning algorithm should be exactly the same statistics that are produced by such an algorithm if it is applied for a long time in past learning. Recall that the RA algorithm uses the ordinary perceptron rule. We examine what statistics the perceptron rule will produce.

In what follows, for simplicity, any probability distribution involved will be assumed to have zero mean. This makes no consequential difference because on a binary network with ramped binarisation, the absolute value of weights has no effect on the network's behaviour. It is only the differences that count. One can always shift the origin to make the mean of distributions zero for any particular and therefore all connections (since no one connection should be special).

It is shown in the following that the perceptron learning rule and the activity ratio constraint imply that 1) any pair of out-going connections from a common H-cell to the O layer tends to have negatively correlated weights, and that 2) the incoming connections onto any particular O cell tend to be statistically independent. The basis of argument is the *central limit theorem,* see e.g. (Feller, 1966) and properties of Gaussian (i.e. normal) distributions.

Weights Are Gaussian Random Variables

Let us firstly examine the statistics of the forward matrix W_{HO}. Let the present time be n, and the present forward matrix be $W_{HO}(n)$. One has

$$W_{HO}(n)= W_{HO}(0) +\Sigma^n{}_1\, \Delta W_{HO}(s),$$

where $W_{HO}(0)$ is the initial matrix, $\Delta W_{HO}(s)$ is the modification at time s=1,2,...n. At any entry $W_{HO}(i,j)(n)$ of the matrix, one evidently has

$$W_{HO}(i,j)(n)= W_{HO}(i,j)(0) +\Sigma^n{}_1\, \Delta W_{HO}(i,j)(s). \qquad \textbf{\textit{(Eq. 4.10)}}$$

That is, any entry $W_{HO}(i,j)(n)$ is a sum of random numbers (given a long period of unspecified learning expericence). One can always define time s, which merely registers the number of opportunities (or 'turns') for the weight $W_{HO}(i,j)$ to be modified, such that modifications $\Delta W_{HO}(i,j)(s)$ and $\Delta W_{HO}(i,j)(s')$, $s\neq s'$, are statistically independent. In other words, one can always lump successive modifications together and count them as one modification so that the 'lumps' are statistically independent. 'Lumping' is the most common technique to achieve statistical independence, cf. (Feller, 1966). It is also consistent to assume that the initial value $W_{HO}(i,j)(0)$ is statistically independent to any subsequent modifications $\Delta W_{HO}(i,j)(s)$.

Under the above assumptions, provided that past learning tasks can be modelled by some stochastic process obeying very general technical conditions (such as the existence of a second moment), one can apply the well known *central limit theorem*, which says that such a sum $\sum^n_1 \Delta W_{HO}(i,j)(s)$ is a Gaussian random variable if n is large enough. The approach to Gaussian distribution is usually very fast as n increases (Feller, 1966). For instance, for modifications $\Delta W_{HO}(i,j)(s)$ where $s=1,\ldots,n$ that are drawn from a uniform distribution, n=10 is sufficient for the sum to de described accurately by a normal distribution.

One can then conclude that

> *the present value* $W_{HO}(i,j)(n)$ *of any entry of the forward matrix can be modelled as a Gaussian random variable for reasonably large n.*

In the absence of any other assumption about the nature of past learning tasks, it is inevitable as well as convenient to assume that all of the entries $W_{HO}(i,j)(n)$ of matrix $W_{HO}(n)$ are generated from a common Gaussian distribution; let us also shift the distribution so as to have zero mean.

Non-positive Correlation of Outgoing Weights From the Same H Cell

Note that we have not yet made any assumptions about the correlation amongst the matrix entries. Information about correlation may be obtained by looking at the learning process more closely. We first look at the pair of out-going weights $W_{HO}(i,j)(n)$ and $W_{HO}(k,j)(n)$ from a common H cell j.

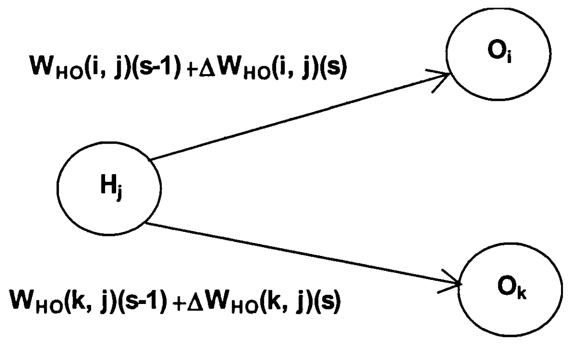

$W_{HO}(i, j)(s-1) + \Delta W_{HO}(i, j)(s)$
O_i
H_j
$W_{HO}(k, j)(s-1) + \Delta W_{HO}(k, j)(s)$
O_k

Figure 4.3a

Past increment $\Delta W_{HO}(i,j)(s)$ and $\Delta W_{HO}(k,j)(s)$ may be regarded as having identical statistics (not necessarily Gaussian) and 0 means. Consider the pair as a random vector $X \equiv (\Delta W_{HO}(i,j), \Delta W_{HO}(k,j))$. Then the above amounts to saying that X, at any time, must be generated from a distribution $P(X_1, X_2)$ that has 0 mean, and that is symmetric in X_1 and X_2.

In this case, its covariance matrix can be written in the form of

$$\begin{pmatrix} \sigma^2 & \rho\sigma^2 \\ \rho\sigma^2 & \sigma^2 \end{pmatrix} \qquad \textit{(Eq. 4.11a)}$$

where σ^2 is the variance, and $|\rho| < 1$ because of the equation

$\int dX_1 dX_2 \, (X_1 - X_2)^2 \, P(X_1, X_2) > 0$ for any probability density function P.

Applying the central limit theorem again, one concludes that

> *any pair* ($W_{HO}(i,j)(n)$, $W_{HO}(k,j)(n)$) *of out-going weights from a common* H-*cell* j, *as the sum of a large number of random vectors X (not necessarily Gaussian) with covariant matrix (Eq. 4.11a), is a bivariate Gaussian variable with covariance matrix (Eq. 4.11a), where*

σ^2 is redefined to absorb a constant factor n (directly corresponding to the number of independent weight modifications made in the past).

Next, let us argue that the correlation coefficient ρ is non-positive if weights have been modified by a perceptron-type rule, assuming 1) fixed activity ratio that is less than 0.5; 2) the independence of output cell activity otherwise, excluding the non-independence originated from the constant activity ratio assumption and; 3) independence of errors.

There are only 2 possible occasions out of 16 (including the 4 in which no errors occur on the two selected output cells) in which both modifications $\Delta W_{HO}(i,j)$ and $\Delta W_{HO}(k,j)$ take the same sign. The following table lists the signs of corrections in contingencies where either $\Delta W_{HO}(i,j)$ or $\Delta W_{HO}(k,j)$ is non-zero.

Sign of correction if	*require i, k on*	*require i, k off*	*require i on, k off*	*require i off, k on*
1.only cell i is wrong	+ 0	- 0	+ 0	- 0
2.only cell k is wrong	0 +	0 -	0 -	0+
3. both are wrong	+ +	- -	+ -	- +

Table 4.2

Note that only the 4 contingencies (++), (--), (+-) and (-+) are relevant to the correlation of between $\Delta W_{HO}(i,j)$ and $\Delta W_{HO}(k,j)$.

Assuming independence of network outputs and target outputs during past learning, then the two components of the Gaussian vector ($\Delta W_{HO}(i,j)$, $\Delta W_{HO}(k,j)$ are independent of each other as the correlation between them are calculated to be zero. Further, the correlation is negative for small networks. Due to the special condition the total number of 'on' cells in outputs is fixed, the probability of a cell being 'on' is not strictly independent of other cells. In fact, contingencies (++ or --) are forced to be less frequent than contingencies (+- or -+), given activity ratios less than 0.5. This effect is only significant for small networks. For large networks, as N_O goes to infinity, the effect diminishes: the correlation approaches zero from below zero. (Note if there is always only one 'on' cell in the outputs, then contingencies (++) or (--) do not occur, implying a negative correlation).

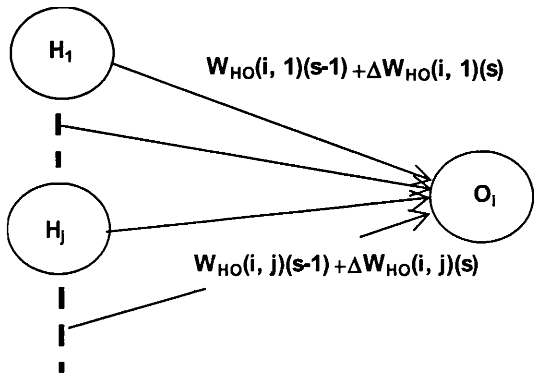
In summary,

> *The correlation between* $\Delta W_{HO}(i,j)$ *and* $\Delta W_{HO}(k,j)$ *is non-positive under the assumed conditions. The parameter* ρ *in (Eq. 4.11a) is non-positive, thus the weights* $W_{HO}(i,j)(n)$ *and* $W_{HO}(k,j)(n)$ *are also non-positively correlated. The covariance matrix (Eq. 4.11a) is such that*

$$-1<\rho\leq 0. \qquad \textbf{\textit{(Eq. 4.11b)}}$$

Independence of Incoming Weights To the Same O Cell

Next, let us look at any row vector X(n) of the forward matrix $W_{HO}(n)$ at time n, i.e. the incoming weights of a particular O cell. Let $X_j(n) = W_{HO}(i,j)(n)$, $j=1,2,\ldots N_H$, with i fixed:



H_1
$W_{HO}(i, 1)(s-1)+\Delta W_{HO}(i, 1)(s)$
O_i
H_j
$W_{HO}(i, j)(s-1)+\Delta W_{HO}(i, j)(s)$

Figure 4.3b

If the weights have been modified by perceptron-type rules, one notices that any modification $\Delta X(s)$ to the row vector has a constant length (as a vector) proportional to the activity ratio α_H of the H layer. Further, assuming that all these connection weights are equally likely to be wrong and modified, the modification vector ΔX must have an isotropic distribution in the N_H dimensional space. Under these two conditions, the central limit theorem enables one to conclude that

> *The row vector* $X(n)$ *of the forward matrix* $W_{HO}(n)$, *i.e. the incoming weights onto any particular* O *cell, is a* N_H*-dimensional Gaussian vector with a covariance matrix* $C=\sigma^2 \mathbf{1}$, *where* $\mathbf{1}$ *is the unit* $N_H \times N_H$ *matrix; in particular, this implies that any two incoming weights of a common* O *cell are statistically independent.*

Formalising the Inversion Problem in the Context of Gaussian Weights and Fixed Activity Ratios

Let the binarisation procedure **B** in (Eq. 4.7) be the ramped binarisation procedure (Section 4.2.3). Let us describe what this amounts to in case of a transpose matrix. Assume that the activity ratio on H layer is α_H with total cell number N_H and the activity ration on O layer, α_O. Then using the transpose, for any output P_O, (Eq. 4.7) reads

$$P_O^*(l) = \mathbf{B}_O(\Sigma_j W_{HO}(l, j)\mathbf{B}_H (\Sigma_k W_{HO}(k, j) P_O(k))) , l=1,2,\ldots N_O$$

Note that all $\mathbf{B}_H$ does is that it picks out the top-$N_H\alpha_H$ numbers out of N_H numbers, each of which is the sum of $N_O\alpha_O$ number of weights (since there are only $N_O\alpha_O$ 'on' output cells in the output patterns. The operator $\mathbf{B}_O$ does likewise.

It is possible to calculate the probability of the 'recovered' pattern P_O^* being the same as the original P_O. Recall the statistical structure of the weights in W_{HO}: each weight $W_{HO}(k, j)$ is a Gaussian random variable independently drawn from the same Gaussian distribution of mean zero and variance σ^2. Let us ignore the possible negative correlation between weights in the same column (i.e. outgoing weights from a common H cell). It will be clear that any negative correlation only *increases* the probability. Thus, in the assumption of large N_H and N_O, any cell that survives the binarisation operator must have its activation in the top-α_H (or top-α_O) portion of the relevant Gaussian distribution.

The problem of calculating the 'recovery' probability under the large number assumption thus translates into the following integration exercise on Gaussian distributions:

> *Given* $N_H\alpha_H$ sets of $N_O\alpha_O$ *numbers*
>
> $\{w(k,j)|\ k=1,2,\ldots,N_O\alpha_O\}_j,\ j=1,2,\ldots,N_H\alpha_H$
>
> *independently drawn from the Gaussian distribution* $G(0, \sigma^2)$ <u>*such that*</u> *the sum* $\Sigma_k w(k,j)$ *for each set belongs to the top-*α_H *portion of the Gaussian distribution* $G(0, N_O\alpha_O\sigma^2)$*, what is the probability* $\boldsymbol{p}(\alpha_O, \alpha_H, \sigma^2)$ *for the sum* $\Sigma_j w(k,j)$ *to be in the top-*α_O *portion of the Gaussian distribution* $G(0, N_H\alpha_H\sigma^2)$ *for* $k=1,2,\ldots,N_O\alpha_O$*?*

The solution $\boldsymbol{p}(\alpha_O, \alpha_H, \sigma^2)$ to this problem gives the probability of all the 'on'-O-cells in P_O being 'on' in P_O^*. The bigger this probability the better the inversion. Note that although the sets are independent from each other, the numbers within each set are not independent by virtue of the condition imposed on their sum even though the numbers are otherwise drawn independently.

While the problem is well-defined, the calculation for $\boldsymbol{p}(\alpha_O , \alpha_H , \sigma^2)$ is highly complex and the result cannot be expressed analytically. Let us simply list some qualitative but precise properties in simple situations.

With a single active Cell on both the H and O Layers

The problem simplifies to one of calculating the probability of any number, randomly drawn from the top-α_H portion of $G(0,\sigma^2)$, being also in the top-α_O portion of $G(0,\sigma^2)$. It is clear that as long as $\alpha_H \leq \alpha_O$, the probability is 1. That is, with a single 'on'-cell, and for large H and O layers, the transpose performs accurate inversion with probability 1. Note that for finite (small) cell numbers, these conclusions cease to be strict, since instead of considering the top α fraction of each probability distribution, what is relevant is the top α fraction of a set of samples from this distribution.

1) the probability will be less than 1 (since being top amongst a sample of say, 10, leaves finite chance for being outside the 10%- or even the 20%-percentile of the population);

2) the smaller the ratio α_H/α_O, the higher the probability of perfect inversion since the chances of being top amongst a sample of say, N_H=20, implies a good chance of being top amongst a sample of say, N_O=3.

3) other factors being equal, any pair-wise negative correlation that exists amongst outgoing weights from common H cells increases the probability of perfect inversion as it implies that the peers, against which comparisons are made, are drawn from $G(0,\sigma^2)$ randomly but with a negative bias.

With a single active Cell on just the O Layer

This setting corresponds to the tasks simulated in Chapter 5. The problem simplifies to calculating the probability of the sum of $N_H\alpha_H$ numbers, each randomly drawn from the top-α_H portion of $G(0,\sigma^2)$, being in the top-α_O portion of the distribution $G(0, N_H\alpha_H\sigma^2)$.

Note that $G(0, N_H\alpha_H\sigma^2)$ can be a much wider distribution than $G(0,\sigma^2)$. As a result, it is necessary to have $\alpha_H << \alpha_O$, in order for the probability to approach 1. For finite cell numbers, given the properties of Gaussian distributions, one has

1) the smaller the ratio α_H/α_O, the higher the probability of perfect inversion.

2) other factors being equal, any pair-wise negative correlation that exists amongst outgoing weights from common H cells enhances the probability of perfect inversion as it implies that peers, against which the comparisons are made, are the sums of numbers drawn from $G(0,\sigma^2)$ with a negative bias.

Note that the above implies that, in the 1-'on'-output-cell setting, the smaller the H-layer activity ratio, the better the transpose performs inversion.

It can be conjectured that in general, the smaller the H-layer activity ratio relative to the O-layer activity ratio, the better the quality of inversion by the transpose.

4.3.5 Comments on Initial Weight Statistics and Activity Ratio Setting for RA learning

For the RA algorithm to function, the network setting must be such that it allows 'adequate' inversion by the reverse matrix. Otherwise the reverse activation received by each H cell will contain little information about its effectiveness in evoking the target output pattern; the basis of reverse activation becomes invalid. Having chosen the transpose as the reverse matrix, it is important that the network parameters allow 'adequate' inversion by the transpose. It is impossible to define what level of accuracy is 'adequate' since there is as yet a logical gap between the ability to invert and the ability for the RA algorithm to construct 'good' representation to learn. But it is clearly relevant.

From the last section, it can be seen that the adequacy, in case of the transpose, is partly determined by the weight statistics. Because of this and our choice of the transpose, certain restrictions on the initial weight statistics must be imposed. For instance, Section 4.3.4 implies that positive correlation between outgoing weights from common H cells is highly undesirable. In one of the tasks described in Chapter 5, initial weights are generated from mock-learning random mapping tasks using perceptron rules, thereby producing the desired statistics described in the last section. It is not entirely clear whether the above mock-learning preparation is essential for RA-with transpose. However, Section 4.3.4 does suggest that independence amongst weights may be good enough. This is indeed used also in simulation, for which RA-with-transpose seems to function 'normally'.

Another perhaps more important factor in determining the adequacy of the transpose is the activity ratio of the internal representations in relation to that of the output patterns. In general, internal representations with small activity ratio (i.e. sparse representations) seem to be desirable, as far as inversion is concerned.

However, it is not clear how the ability of the transpose to invert accurately affects overall learning performance (using the RA algorithm). Further, activity ratio must affect learning in other ways. For instance, if the activity ratio is too low, there may not be enough representational capacity on the H layer to solve a given problem (even though the transpose can invert perfectly). We shall come back to this when discussing simulation results.

Section 4.4 Reverse Activation Algorithm in Multi-layer Networks

Consider the multi-layer hierarchically arranged autoassociative network introduced in Section 3.4.2. Label the input layer as the 1st layer and the output layer as the Nth, with intermediate layer labelled accordingly. Denote this network by $1 \Rightarrow 2 \Leftrightarrow \ldots \Leftrightarrow N$. Recall that each layer can function independently as an autoassociative memory, the properties of which correspond well with short-term memory. In addition, there are forward and backward connections linking every unit in one layer with every unit in neighbouring layers (only forward connections from the input layer), usually modelling long-term memory. As it has been assumed that the these three classes of connections can function independently, let us ignore the internal autoassociative weights, and concentrate on the difficulties presented by multiple hidden layers.

Let us continue to assume that the inter-layer connections are symmetric: forward weights equal backward weights. One can generalise the RA algorithm to such a multi-layer network functioning as a feedforward memory. The key part of the algorithm is how to construct improved internal representational patterns on each intermediate layer.

This involves superimposing the activation from the input and output pattern via forward and backward connections (with a certain chosen reverse activation strength).

For the simple 3-layer net this operation is straightforward. With multiple intermediate layers, it is more complicated as no layer is directly connected to both the input and the output layer. When the input and output pattern are imposed on their respective layer, what pattern is selected (after ramped binarisation) on layer K depends on what patterns are selected on layer (K-1) and layer (K+1), which in turn depend on what patterns are selected on layer K and, respectively, layer K-2 and layer K+2. That is, one has a dynamic situation.

4.4.1 Using Stationary States to Construct Internal Representations

It may that the dynamics of a multi-layer network can settle into a stationary state so that the patterns are mutually reinforcing and thus stable. *The internal patterns so produced when the network is in a stationary state (while the input and the output layer are clamped) are the improved internal representations layer by layer.* They can then be implemented via perceptron rules. The above is then the key of the generalised RA algorithm in a multi-internal-layer setting.

It is crucial that the inter-layer dynamics, established when the input and output layers are clamped, is such that there are always stationary states (not just cycles) in which to settle. Otherwise one has no *natural* basis to favour one set of internal representations over any other. This is a very stringent requirement. Fortunately, this requirement can be met, thanks to the assumption that the backward and forward connections are symmetric.

One can prove this assertion by employing standard techniques. In (Hopfield,1982), it is proven that the dynamics of a network of symmetrically connected 0-threshold binary neurons always admits stationary states; in (Kosko, 1988), it is proven that any real connection weight matrix admits stationary states when it is used as a bi-directional associative memory, so called BAM theory. In fact Kosko's result follows from Hopfield's result since BAM is a special case of Hopfield net. The present assertion regarding dynamics on the partially clamped network $1 \Rightarrow 2 \Leftrightarrow \ldots \Leftrightarrow N$ follows similarly ("partially clamped" since the input and output patterns are fixed on the input-output layers). However, the present context is sufficiently different to justify a more detailed explanation. The following gives the important steps in the proof.

4.4.2 Proof that Stationary States Always Exist on the Given Network

Given the network $1 \Rightarrow 2 \Leftrightarrow \ldots \Leftrightarrow N$, let the weight matrices connecting layer K to K+1 be denoted as W^{K_K+1}; the corresponding backward connection matrix is thus $W^{K+1\_K} = (W^{K\_K+1})^T$. Note that the dynamics established when the input and the output patterns are imposed is governed by the following energy function, in vector notations,

$$E\,(P_O, P_I; W, \psi) = (-1/2)\,(P_2^T\, W^{1\_2} P_1 + \psi^{N-2} P_N^T\, W^{N-1\_N} P_{N-1}) +$$

$$(-1/2) \sum_{K=2}^{K=N-1} (\psi)^{(K-2)} \{(P_K^T\, W^{K-1\_K}\, P_{K-1}) + (\psi)(P_K^T\,(W^{K\_K+1})^T\, P_{K+1})\} \qquad \textbf{\textit{(Eq. 4.12)}}$$

where P_K is the pattern on layer K, treated as vectors, with $P_O \equiv P_N$ and $P_I \equiv P_1$ *fixed,* being the input and output patterns, and ψ is a positive constant, the reverse activation strength.

With respect to any cell in any hidden layer (K=2,...,N-1), the derivative of this energy function against the activity of that cell is *proportional* to the combined activation that it receives from the two neighbouring layers with a reverse activation strength ψ; the proportionality being $-(\psi)^{(K-2)}$. In discrete time, the pattern on each layer is updated synchronously or asynchronously in turn according to the ramped binarisation updating rule. Note that in such an update, the state of a cell is changed (i.e. turned 'on' from 'off' or 'off' from 'on') *iff. the resulting value for the energy function above* *is strictly lowered.* That is, the energy function is strictly decreasing along the dynamic flow.

Each term in the energy function is bounded below, so the energy function is also bounded below. Further, this energy function is well-defined, as it is symmetric with respect to P_K and P_{K+1} for all K=2,...,N-1 (evident by taking the transpose of each term, which should leave it unchanged since it is merely a real number). These conditions ensure that local minima exist for the energy function (Eq. 4.12). Since the dynamics strictly reduces the energy function, the system will settle into at least a local minimum eventually, which implies that no further changes in the firing patterns will result from future updates. The convergence of such dynamics, essentially a Hopfield net, is usually swift; see examples in (Amit, 1989).

To summarise, one concludes that

> *any multilayer, bi-directionally connected network* $1 \Rightarrow 2 \Leftrightarrow \ldots \Leftrightarrow N$ *with real connection matrices admits stationary states when the input and the output patterns are imposed on the respective layers. The generalised RA algorithm then selects the patterns in the stationary states so achieved as the representational patterns on each internal layer. These are then implemented in the feedforward map via the simple perceptron rule layer by layer.*

In the energy function (Eq. 4.12), a universal reverse activation strength has been chosen. This is not strictly necessary. It is possible to have different strengths for different pairs of layers. In which case, one can replace, in (Eq. 4.12), ψ by ψ_{K-1}, ψ^{K-2} by $\psi_0\psi_1,\ldots\ \psi_{K-2}$, and ψ^{N-2} by $\psi_0\psi_1,\ldots\ \psi_{N-2}$, where K=2,..., N-1, and ψ_{K-1} is the reverse activation strength chosen for weights between layer K and K+1.

These parameters, as in the original RA, control 'how the task of learning will be shared amongst the forward matrices'. A large ψ_{K-1} implies more 'burden' on connections coming into layer K and less on connections from layer K to K+1.

4.4.3 Interpretation of Generalised RA

When a pair of input and output patterns are imposed, the ensued dynamics on the multi-layer net can be seen as an automatic search for a pattern configuration in which the representational pattern on every layer is consistent with the activation that it receives from its neighbouring layers, which is ultimately determined by the input and the output pattern. If such patterns are chosen as the target internal representations, overall weight modifications required to implement them in the forward mapping (by changing the forward weights layer by layer) are expected to be small since they are already mutually reinforcing. Note that in the 3-layer setting, the dynamic selection process is trivial as there is no dynamics in the 3-layer net when the I and O layers are clamped.

It should be interesting to find out how the generalised RA would work in multi-layer simulations, though this work has not been carried out for the thesis. Immediately, one can see that the technique of randomly tuning the reverse activation strengths (see Section 4.2.4) is particularly relevant and perhaps essential to the generalised RA algorithm due to the possibility of having numerous reverse activation strength parameters.

Chapter 5 Simulation of Reverse Activation Algorithm

It is nearly impossible to work out analytically how the reverse activation (RA) algorithm would perform in practice. In this Chapter, simulations are carried out for 3-layer networks using a matrix of reverse weights that is the transpose of the forward weights (Section 4.3.3). In other words, reverse weights (W_{OH}) are equal to the forward weights (W_{HO}) connecting the same pairs of cells.

Section 5.1 Methodology

The purpose of these simulations is firstly to obtain information about the properties of the RA algorithm itself, and secondly to make comparisons with standard algorithms.

Standard three-layer binary networks are employed. The number of cells in the I and O layers is fixed by the chosen learning task. Training is carried out for a selection of initial conditions (independently generated initial weights), under each combination of tuneable parameters. Hence if there are X_{ini} initial conditions and X_{com} sample combinations of tuneable parameters, one has $X_{ini} \times X_{com}$ trials in all. Each trial consists of a fixed number of training epochs; it is continued according to the criterion that it should be prolonged enough for the network performance to reach its asymptotic level (so that further training will not yield any new information). Data associated with learning is recorded at the end of each epoch in each trial.

As the data will reveal, performance can crucially depend on the choice of various learning parameters. This is undesirable in a working system, so an alternative procedure is explored in Section 4.2.4, allowing the various parameters to take random values throughout learning. These simulations are repeated even for the same initial conditions because each trial involves random processes.

5.1.1 Sampling Tuneable Parameters

There are 3 tuneable parameters in the RA simulation: the step-size (i.e. unit weight change) λ, the reverse activation strength ψ, and the H layer activity ratio α_H.

Step-size and updating procedure

The RA algorithm aims to find a representation that would allow learning with minimal disturbance to previous learning and the existing weights (4.2.1). It does not introduce new synaptic learning rules. Instead, it is merely a procedure for creating improved internal representations; once a representation is determined, connection weights are modified using perceptron rules till the correct output is produced.

Given the above background, the most natural updating procedure for learning with RA is *total-on-line (3.1.2)*, i.e. the most recent item is always learned with perfect accuracy and thus can be recalled perfectly before the next presentation. This also seems more biologically plausible than either batch or on-line updating, which do not learn any single input-output mapping until after the whole training set is repeated, often many times over.

There is an added advantage to total-on-line updating. The step-size λ is involved in RA only because RA uses the standard perceptron rule. Since the general effect of λ on simple and multi-layer perceptron learning is well-known (cf. Section 3.1 and 3.2), in order to concentrate on the new aspects of RA learning, it is necessary to isolate and minimise the effect of λ. To achieve this, total-on-line learning procedure was ideal.

In this procedure, the network is 'forced' to learn the most recent item perfectly. Provided that the network is not too small (so that learning even one mapping is difficult) and using the perceptron rule, the weight changes that achieve the new mapping are restricted to those involving the active I and O cells and are a function of the local landscape of the error surface, relatively independent of the step-size involved. In other words, perfect learning of the new input-output mapping tends to lead to the same weight configurations whatever the step-size: if the step-size is very small, then more iterations may be required; if it is bigger, then those configurations can be achieved with less iterations. The limit is that excessively large step changes may make even learning one mapping unstable. Thus the effect of step-size on performance is less with the total-on-line updating procedure, compared to the on-line or batch alternative. This leaves one free to explore the effects of activation strength ψ and the H layer activity ratio α_H, the two new elements introduced by RA.

The main RA simulations are thus done with a fixed step-size, 0.005, small enough given the initial weights. It is of course useful to know how RA might cope in other updating procedures. This will be discussed in Chapter 6.

The Size and Activity ratio of the Hidden (H) Layer

The size N_H of the hidden layer, in this kind of studies, is usually chosen so that it is not too big (or too small) as to make learning too easy (or too difficult respectively) to the extent that different learning algorithms become indistinguishable in performance. For any given learning task, there is probably no unique choice that achieves the above balance perfectly and there is not any general method to determine what size the hidden layer should be. It is mostly a matter of trial and error, combined with hind sight. In other words, one carries out some preliminary simulation for a chosen size of the hidden layer and see if learning is too easy or too difficult for the learning algorithms concerned; when the full simulation data has been collected, one checks again that the hidden layer used is 'reasonable'. As such there is always a degree of subjectivity involved and the final data set may still be open to debate as to whether the chosen size for the hidden layer is too easy or too difficult to the learning algorithms being tested. The simulation carried out in this work is no exception in this respect. (However for the second task, the mirror symmetry task, the hidden-layer size is taken from what is in the literature directly.)

The activity ratio α_H is fixed on the H layer for the RA algorithm, so an exhaustive sampling of α_H is possible, i.e. $W=N_H\alpha_H=\{1, 2, \ldots N_H\}$ where W is the number of active cells. However, one would expect the learning performance to be more sensitive to activity ratio α_H when it is very low; as α_H becomes higher (towards 50%), the change in performance will get progressively less. The real interest is in finding out at what ratio peak performance can be achieved. In data collection, it is sufficient to sample more at the lower end of activity ratio and progressively less as the ratio gets bigger.

Reverse Activation Strength

The reverse activation strength ψ (≥ 0) is the most important parameter in the RA algorithm in determining performance. What is a 'fair' sampling method for ψ? If plotting a function by sampling its variables can be a guide, a reasonable definition of *fair sampling* of a continuous parameter x on which a variable y depends continuously is the following:

A fair sampling set $\{x_j, j=1,2,\ldots,\}$ *is such that the behaviour of the dependent variable y changes in a 'steady enough' manner between sampled parameter values (such that no major turning points are missed while no unnecessary time is spent on 'flat' regions). In other words, ideally one has* $\Delta y_j = y(x_{j+1}) - y(x_j)$ *roughly constant for all* $j=1,2,\ldots$.

It is largely a matter of judgement how fair sampling might be done as it relies on answering the question of how the dependent variable changes with the parameter to be sampled, which is the object of the sampling exercise in the first place. In the present situation, performance of the RA algorithm will only differ for two different values of ψ if they lead to different cells recruited in the representation. This means that there is no point in sampling in such small steps that no change in the representation results. Hence a reasonable procedure seems to be that the samples should be spaced so as roughly to alter the representations by equal numbers of cells.

Note that for given prior experience (as reflected by initial weights) and a chosen activity ratio α_H, ψ alone affects the new internal representation pattern for an input-output pair. Changing the value of ψ effects the changes in internal representation and hence the changes in weights. As the activity ratio is fixed, the set of all possible internal representations are all on the surface of a N_H-dimensional ball of radius W ($=N_H\alpha_H$). Any change in internal representation amounts to a rotation. Thus one can visualise the chosen internal representations rotating as ψ varies.

It is reasonable to expect that the smaller the difference between the chosen internal representations the smaller the difference will be between the resulting weight changes. Therefore a fair sampling set of ψ must have the property that as one goes from one sample value to the next, the internal representations constructed rotates in a steady manner. Note that this does not mean that the sampling set for ψ has to be uniform. For instance, the function *arctan*(ψ) does not vary steadily when ψ is sampled uniformly.

The use of the function *arctan* to explain the above point is not accidental. From its definition, ψ is the *negative of the gradient* of the tilted threshold line in the activation scatter of H-cells; see Figure 4.2b and Figure 4.2c of Section 4.2 for example. Note the tilting angle has a range of $[0, -\pi/2)$ as ψ has a range of $[0, \infty)$. Given an input-output mapping to be learned, two different strengths ψ_1 and ψ_2 (two different tilting angles) will result in identical weight changes <u>unless</u> there are H cells that fall into the 'gap' between the two tilted threshold lines on the activation scatter. This is because only then will the resulting internal representation patterns differ for ψ_1 and ψ_2. For new learning, a scatter graph such as Fig. 4.2b shows little correlation (4.2.1), so with appropriate scaling the number of cells lying between lines of different tilt is approximately proportional to the angle between them ($\arctan(-\psi_1)$ - $\arctan(-\psi_2)$). For this reason, ψ values are generally sampled uniformly in $\arctan(-\psi)$, corresponding to uniformly spaced tilting angles. For each step increase in the sample values of ψ the modified representation is likely to differ by a roughly constant number of cells, which is likely to lead to steady changes in the resulting weight modifications and hence in the behaviour of the network in training.

A rationale for random selection of ψ values from within their sampling set was put forward earlier (4.2.2, 4.2.4), and this is employed in some of the simulations, with an independent random choice each time the RA algorithm is used to create a modified representation. A uniform probability distribution is employed over a set of values ranged uniformly in $\arctan(\psi)$, as described above.

5.1.2 Preparations of Initial Conditions

In testing a learning algorithm, the usual practice is to use independently generated random numbers from a uniform distribution (over the interval [0, 1] for instance) as initial weights. While this is also adequate for the RA algorithm, some comment is due here because the theory behind the RA algorithm makes certain assumptions about the statistics of weights in the network as detailed in Section 4.3.4

As explained, the reverse activation through symmetric backward connections conveys useful information on whether an H cell should be on or off when any pair of forward weights from the same H cell is negatively correlated or at least statistically independent (Section 4.3.4). Two types of initial weights are set up for simulations. The first type contains sets of randomly independently generated weights. There is thus no correlation between the weights; call them 0-correlation initial conditions. The second type consists of sets of weights, each of which is the result of mock learning of a randomly generated I $\rightarrow$ H $\rightarrow$ O mapping task (consisting of 20 triples of input, intermediate and output patterns). The perceptron rule is used to improve weights between successive layers. The activity ratios of all the mock I, H and O patterns are kept the same, on average, as for the actual learning task. This indeed produces the weight statistics analysed in Section 4.3.4. Note that this rather elaborate setup for generating initial weights is not unique. Various initialisation heuristics have also been proposed for BP in the past; it is arguable that the type of mock-learning procedure used here might be beneficial for BP learning (Denoeux, T.; Lengelle, R.; 1993).

The typical size of the initial weights used in simulation was such that they are suitable for BP algorithm. Recall the end of Section 3.2.2, for BP the initial weights should fall roughly within three times the ratio between the typical length of the input pattern (as a vector) and the square-root of the number of training patterns. For the two tasks used in the simulations here, this bound is around 1.35 and 2.4 respectively. For task 1, at least 97.5% of the initial weights fall within this range, and 100% for the second task. The RA algorithm itself, with ramped binarisation, is not sensitive to initial weight size.

Section 5.2 Data for Two Benchmark Learning Tasks

The algorithm is simulated on two well-known benchmark classification tasks. For the first problem, we shall obtain systematic information on how performance depends on ψ and α_H. The issue of generalisation ability will be emphasised in the second task.

5.2.1 CT Discrimination Task

The task is to discriminate between binary patterns C and T in all translations and orientations. A 6×5 grid was chosen as the input layer, and with four orientations 0, $\pi/2$, π, $3\pi/2$, which is the usual practice (Rumelhart *et. al.* 1986). Each input pattern is either a C- or a T-pattern with certain translation and orientation. The output pattern consists of 2 units, one for C, and one for T. There are in all 124 input patterns, 62 C-patterns and 62 T-patterns.

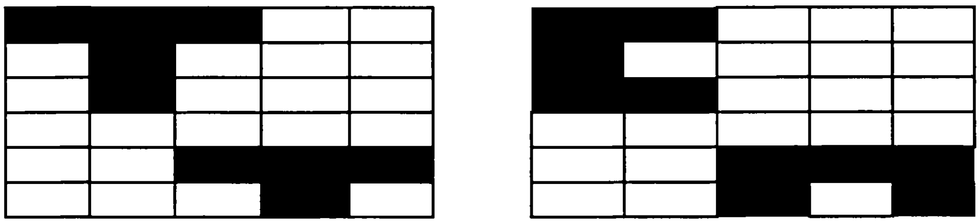


Figure 5.1. Input patterns of the CT problem. Represent both C and T with 5 on-units on a 6×5 grid, except for 14 of the T-patterns where T is on the edge of the grid, and is represented by 4 on-units. Note that there is never more than one letter pattern in the actual input patterns; the figure is for illustration only. Combining orientation with translation, these are 62 C-patterns, and 62 T-patterns.

Since the algorithm involves activity ratio clamping, to give a decent range for the activity ratio parameter α, the H-layer is allowed to have 20 units. Simulation results show that this is a good number to work with (that is, learning is not made too easy or too difficult) for both BP and RA.

To mimic past learning experience, and hence to reproduce the required statistics of weights, 12 sets of initial weights were prepared by mock-learning I $\rightarrow$ H $\rightarrow$ O mappings. For each set, there were 20 randomly generated triplet of input, hidden and output patterns, using the perceptron rule to successive layers. All weights were set randomly and independently at the outset with small values (comparable to the step-size used in mock learnings) . The activity ratios of all the mock-I and O patterns were set at 5/30 and 1/2 respectively, similar to the actual CT task, while the activity ratio of the mock-H patterns ranged from 8/20 to 14/20. Note that the mock-learning input patterns amount to about 0.17% of all possible input patterns of activity ratio 5/30. The resulting statistics of the initial weights are illustrated in the following charts.

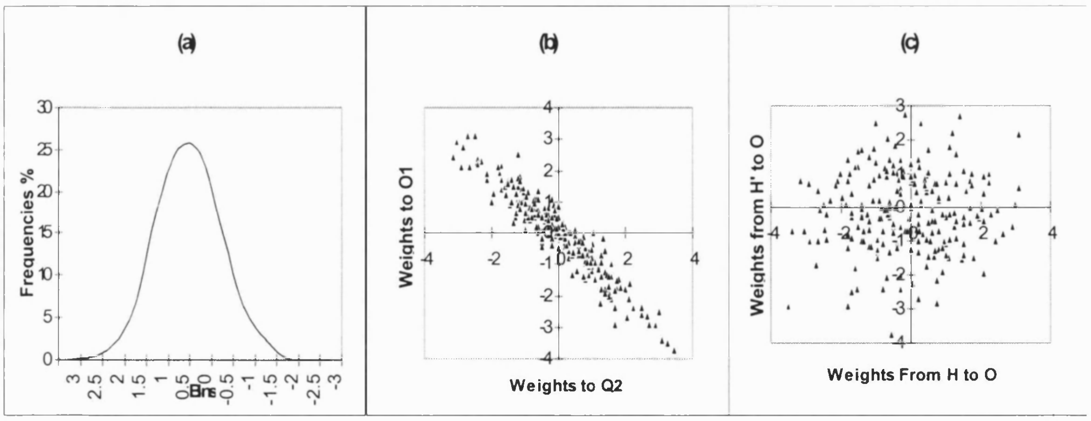


Figure 5.2. Statistics of initial weights. All three charts are plotted using actual weights in all the 12 sets of initial weights, which are the results of mock learning using simple perceptron rule. (a) is the distribution of values of the observed weights from H to O. (b) is the scatter where the coordinates of each point are the weights from an H cell onto the two output cells. (c) is the scatter where the coordinates of each point are weights onto one *common* output cell from arbitrary H-cells labelled H and H'.

It is also useful to plot the activation scatter as in Figure 4.2b&c. For the initial weights used in the simulation, given the training set patterns, it takes the following form.

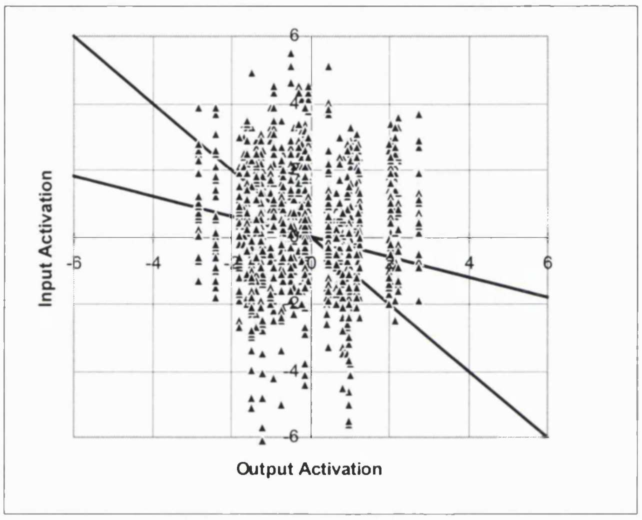


Figure 5.3. Activation scatter of H-cells, superimposed for all training pairs, before learning. Two different reverse activation strengths, 0.3 (16.7 degrees) and 1 (45 degrees), are shown as the tilting threshold lines. The strong tendency of vertical alignment above is an artefact of the fact that there are only two output cells which correspond to exclusive categories; the possible values of reverse activation of each H cell are therefore limited to two.

For each initial condition, the network was trained with a particular combination of reverse activation strength ψ and activity ratio α on the H-layer (more conveniently identified by the number of on-cells W on H-layer). The following set of combinations is chosen:

$$\{\psi=0,\ 0.0875,\ 0.1763,\ 0.2679,\ 0.3640,\ 0.4663,\ 0.5774,\ 0.7002,\ 0.8391,\ 1.0000,\ 1.1918,\ 2.7478,\ 10^6\}$$

$$\times$$

$$\{W=N_H\alpha=1, 2, 3, 4, 6, 8, 10, 12, 14, 16, 19\}. \qquad \textit{(Eq. 5.1)}$$

The ψ values correspond to the tangent of tilting angles ranging from 0 to 50 at 5-degree intervals, with the additional 70 and 90 degrees to provide evidence of completeness of this sampling set. Thus there are $13\times11=143$ combinations for which data is obtained. Performance, measured by the percentage of correct mappings was generally asymptotic after 90 passes. Step-size was 0.005, compared with weight size of the order of 1.

Typical time-course

One characteristic of the RA algorithm seems to be its fast convergence to an asymptotic performance, *good or bad*, obtained typically within 25 epochs, with a substantial part of this performance achieved within the first 10 epochs. Convergence was faster if the learning parameters were non-optimal. An advantage of fast convergence is that in practical applications unsuccessful training sessions (due to inappropriate parameter combinations) can be discovered and abandoned very early on, saving time and resources. Figure 5.4 shows a typical range of time courses.

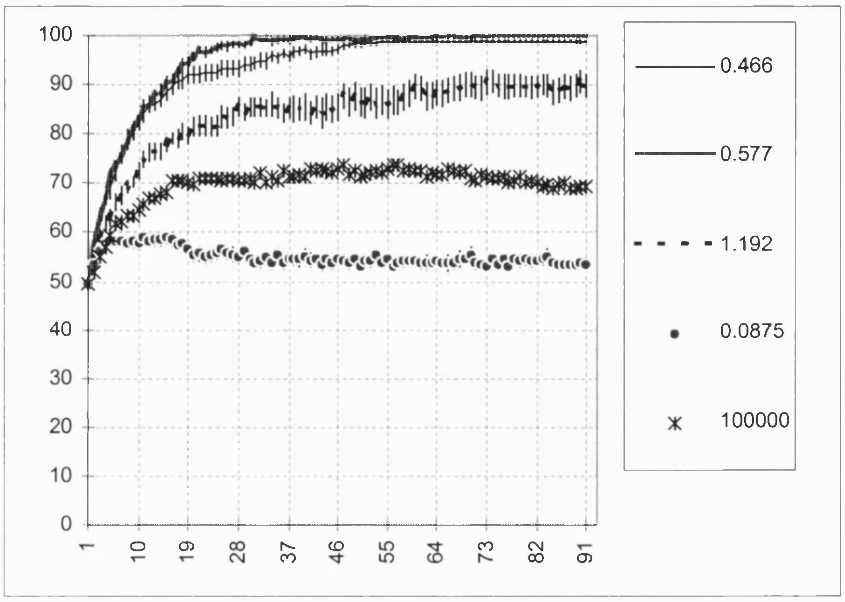


Figure 5.4. Typical time-courses of learning with the RA algorithm. Performance, measured in percentage-correct at the end of each epoch (a pass over the 124 task-mappings) is plotted. All five curves were obtained with activity ratio of 0.5, or $W=N_H\alpha=10$, i.e. all internal representations contain exactly 10 'on' H-cells, with different reverse activation strengths (listed to the right), during learning. Each curve is the average of 12 repeats of the same learning parameter combination, with independently generated initial weights. Means are plotted plus-minus 1 S.E.M. (standard deviation over square root of the number of independent repeats). Where they seem absent, either there is no variation (as in late epochs of $\psi=0.58$) or it is smaller than data symbols. The first points plotted were before training. Time-courses are similar for other activity ratios.

For each parameter combination, the only source of variation in performance comes from the 12 independent sets of initial weights. For the trials illustrated above, which share the same W, the difference between average performance of different ψ values are statistically highly significant. Standard errors are typically small (0-1) for optimal or extremely non-optimal learning parameters. They are the largest (4.5) in the latter epochs of trials with intermediate learning parameter combinations, indicative of a transition in the properties of the network. Paired t-test of performance at the end of each epoch confirms that confidence in the eventual outcome of the training trial converges very quickly. The statistical separation of the sets of trials with different ψ

values and eventual degrees of success is clear after a few epochs, long before the asymptotic performance is reached. The best combination (W=10 and ψ=0.58) can be separated from for instance (W=10 and ψ=1.19) with confidence greater than 99.9% from the very first epoch. Thus, early performance is a good predictor of later performance.

A noteworthy feature is that although both extremely large ($\psi=10^5$) and small reverse activation strengths (ψ=0,0.09) tend to be non-optimal, the very large ones achieve better learning results. Recall that larger ψ means more changes to I$\Rightarrow$H weights and less to H$\Rightarrow$O weights; smaller ψ means the opposite (Section 4.2.2). Given that there are far more I$\Rightarrow$H weights than H$\Rightarrow$O weights in the CT task, this observation should not be surprising.

Performance dependence on ψ and α

Typically, for a fixed activity ratio α, the final performance level gradually reaches a plateau and then falls off again as the reverse activation strength ψ increases. The performance also depends on the activity ratio (Fig. 5.5).

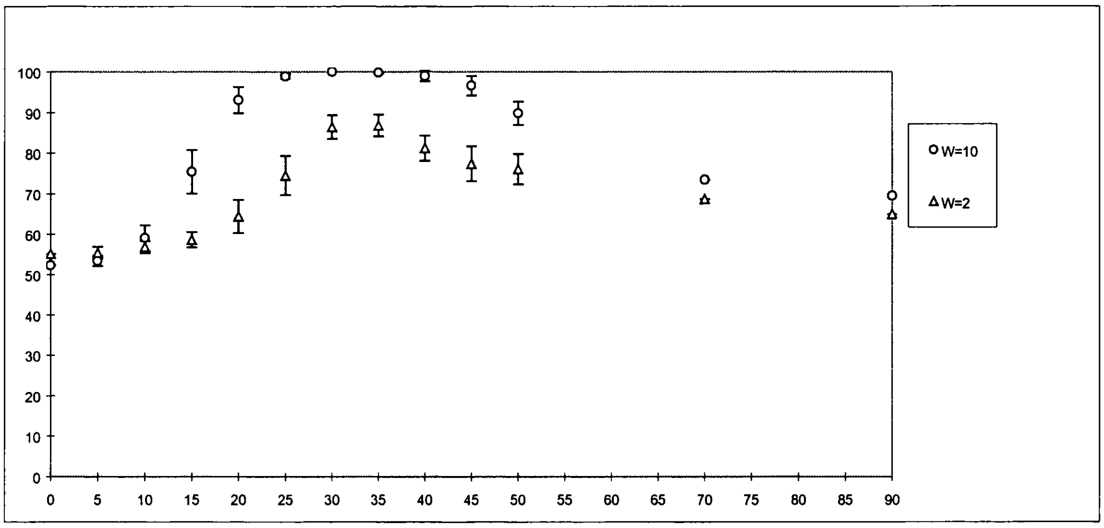


Figure 5.5. Typical dependence of performance on fixed reverse activation strength ψ for the CT task. Data for two different activity ratios are illustrated, one for α=0.5, one for α=0.1, i.e. W=10, W=2 respectively. The ψ values are marked along the horizontal axis, measured in degrees of tilting angle, i.e. in *arctan*(ψ). The vertical axis is the percentage of correct mappings achieved after 90 epochs. Each data point is the average of 12 repeats with independent initial weights, ± 1 S.E.M. (where larger than the symbol size).

What is important here is the existence of an optimal range for the reverse activation strength ψ. The optimal value of ψ (which is the multiplier of the reverse weights in forming a new representation) depends naturally on the relative scaling of the H $\Rightarrow$ O and O $\Rightarrow$ H weights, which are taken as equal here; and it may depend on the nature of the task and the number of cells in each layer. But in this example it does not depend much on the activity ratio chosen for the H layer. The full dependence of asymptotic performance on a variety of combinations of reverse activation strength and activity ratios is plotted in the Fig. 5.6. Optimal performance is for approximately W=5-16 (i.e. α=0.25-0.8) and ψ=0.5-0.8 (angles of 25-40^{o}). For W=8,10,12 and ψ=0.57,0.7 (angles of 30, 35^{o}) 100% performance was achieved for *every* observed initial condition.

The dependence of asymptotic performance on a variety of combinations of reverse activation strength and activity ratios is plotted in the following.

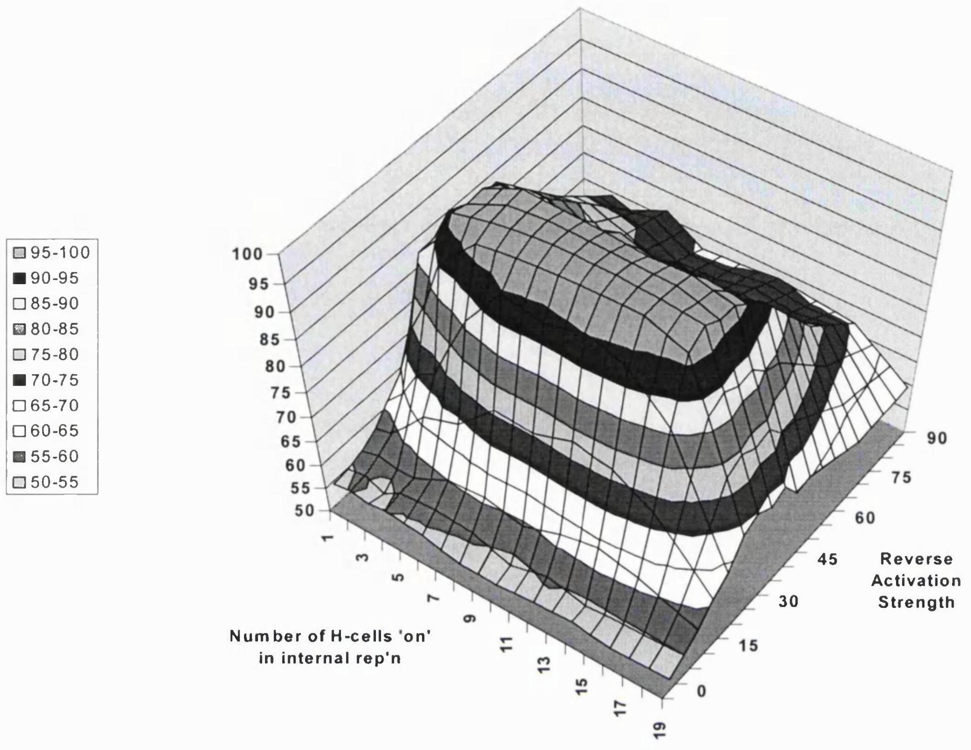


Figure 5.6. Performance dependence on (W, ψ)**.** Percentage of correct mappings achieved after 90 epochs, the asymptotic level, is plotted against 19×19 W-ψ combinations. Data points for combinations {W=1, 2, 3, 4, 6, 8, 10, 12, 14, 16, 19}×{*arctan* (ψ)=0, 5, 10,15, 20, 25, 30, 35, 40, 45, 50, 70, 90°} are means from the simulations. The rest of the plotted grid points are linearly interpolated values for clarity in presentation. Such points are not used in discussions in the text. Each data point is averaged over 12 repeats with independently generated initial weights. Standard errors are not shown, for clarity, but examples are shown in Fig. 5.5.

Recall that the smaller the W, the more accurate the inversion performed by the transpose (Section 4.3.4). The above observations suggest either that better inversion does not necessarily imply better RA learning, or that for outputs consisting of only one 'on' cells out of two, the inversion performed by the transpose is similarly accurate for all W's that are not too close to 20. Both appear to be true. The fact that the output layer has only two output cells with strongly negatively correlated incoming weights makes the transpose an accurate inverse operator for W up to 14. However learning with W=8,10,12 clearly was better than with W=1,2,3, even though inversion is slightly more accurate for small W's. The conclusion is that activity ratio is affecting learning in ways other than through the quality of inversion. In other words, the benefit of having completely accurate inversion is offset by some disadvantage associated with having too low an activity ratio on the H layer, quite likely simply the paucity of representational capacity on the H layer with sparse coding on a limited number of elements; more of this in Chapter 6.

The Underlying Weight Changes

It is instructive to compare the activation scatter of H cells before and after learning with RA.

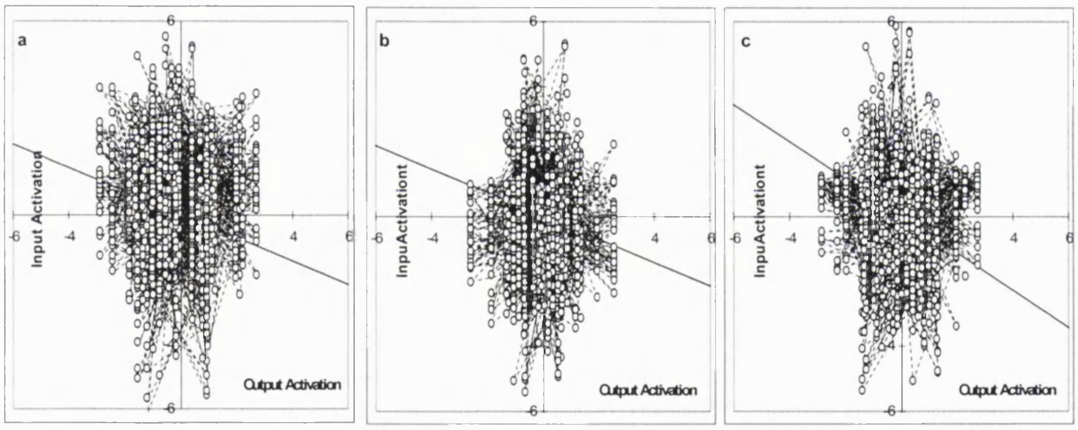


Figure 5.7. Activation scatter of H-cells with respect to the desired input-output pairs. This is shown (a) before learning (b) following training with ψ=0.36 (W=4) (c) following training with ψ=0.57 (W=4), all on exactly the same scale for comparison. Each circle corresponds to an H cell with coordinates given by the input and reverse activation it receives for a pair of input and output vectors. Thus, for each pair of input and output patterns, 20 points are plotted. There are 62 input-output pairs (randomly chosen from the training set) used to plot these charts. The reverse activation strength used during training is represented as the tilting threshold line in (b) and (c).

One can discern the effects of RA learning in the above. Cells that are initially above the threshold line but receive negative reverse activations (implying that they contribute to output errors) must have been either moved rightwards, i.e. their projections to the correct O-cell are increased, or pushed downwards, i.e. they are turned off in internal representations. This results in the empty wedge shaped area in the cloud of cells in the first and second quadrants. Examine in particular the 2 columns of cells to the extreme left of the 'crowd' in each of the charts. Each column in fact turns out to concern a single H cell, which usually has large negative projections to the correct O cell. RA learning has reduced the weights of projections from the input layer onto these cells so that they are turned off in internal representations.

Performance when reverse activation strength ψ is random

As explained in Section 4.2.4, by allowing ψ take random values (whenever superposition of input and reverse activation occurs in training), there is then no need to find the optimal value of ψ by guess-work, a great simplification, if performance does not suffer significantly.

In a random-tuning scenario, whenever the imposition of input and reverse activation (via reverse connections) is needed, the reverse activation strength ψ is randomly generated with equal chances from the fair sampling set (Eq. 5.1). Because of these random selection processes, it is necessary to repeat trials with random reverse activation strengths even for the same set of initial weights. In simulation, trials starting with a single set of initial weights were repeated 4 times using different random sequences of ψ values. The same 4 sequences were used for all initial conditions and W to allow for paired comparisons.

For the CT problem, random tuning has proven to be quite effective as demonstrated in Fig. 5.8.

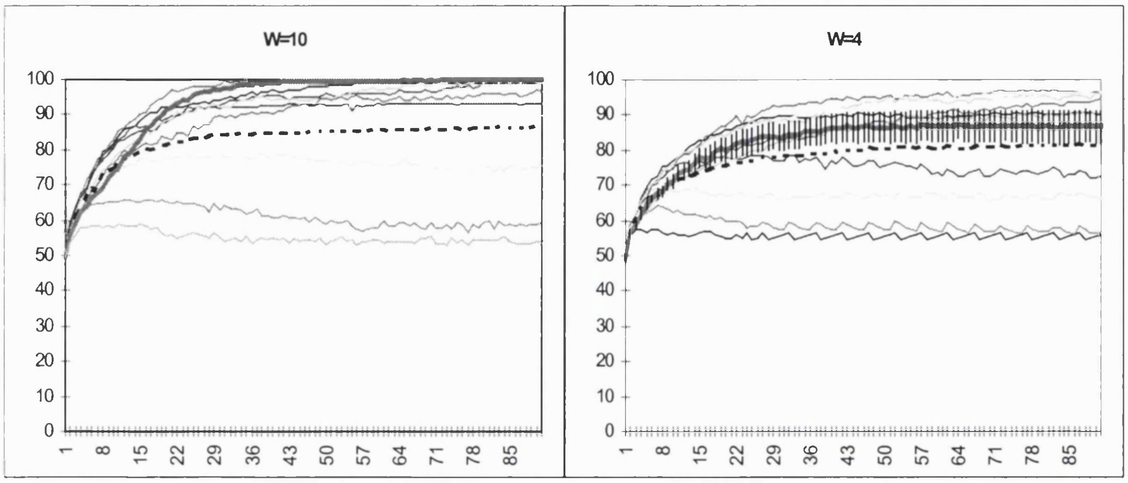


Figure 5.8. Time courses of learning with fixed and randomly varying ψ, for W=10 and W=4. Thin lines are the time-courses of learning with fixed ψ ranging from 0.0875 (5 degrees), to 1.192 (50 degrees), at a 5-degree interval; each is an average over the 12 independent sets of initial weights. The dotted lines are the means of all of these. The thick lines are the means for learning with ψ varying in 4 different random sequences, and for the same 12 different initial conditions. For the random sequences, standard errors for W=10 are too small to be plotted; For W=4, error bars are shown as the average ±1 S.E.M. (for the 4 sequences) for variance due to the different initial conditions, which accounted for 98% of the total variance after 90 epochs.

For any trial with a fixed W, there is now a new source of variation for performance, coming from the random selection process for ψ, in addition to the variation due to the changes in initial weights. For W=4, analysis of variance at the end of each epoch revealed that initial conditions contribute around 98% of the total sum of squares (i.e. total variance times the total degrees of freedom 4×12-1=47) throughout the learning process. For W=10, variation vanished through perfect performance achieved between the 35th and 40th epoch; prior to that, initial conditions were overwhelmingly (98%) the dominant source of variations.

In all cases, performance with random variations of ψ suffers in the early stages compared with fixed-ψ learning with the optimal values. This is only to be expected as fewer 'correct' weight changes are made per cycle than when the parameter ψ is fixed at an optimal value.

Comparison with Back-Propagation

A systematic BP algorithm simulation was also carried out on the same problem with the same sets of initial weights. The BP code used is for a standard BP algorithm, i.e. batch updates with a momentum term, taken from the textbook by Müller *et. al.* (1991). To apply the algorithm, all the units are turned into graded response units, taking value from [-1, +1]. (As discussed in Section 3.2, this choice achieves faster learning than [0,1].) Testing of performance done was by the mid-point criterion, i.e. only the sign of the activation of output cells needed to be correct. The above design for such comparisons is standard, cf. for instance (Peterson *et. al.*, 1989). Apart from batch updates, on-line and total-on-line updates have also been attempted, which will be discussed in Section 6.4.

The dependence of BP performance on its free parameters is well known. Extensive sampling of the combinations of step-size (0.0004 to 0.1), steepness (0.45 to 1) i.e. the sharpness of the sigmoid transfer function, and momentum (0.2 to 0.9) was carried out for each of the 12 sets of initial weights used for RA-learning; see Section 3.2 for definitions of these terms. The optimal combination (i.e. having the best average asymptotic performance or the fastest convergence amongst those with equal asymptotic performance) observed in this set turns out to be: step-size 0.02, steepness 0.45, and momentum 0.9. The comparisons are shown in Fig. 5.9.

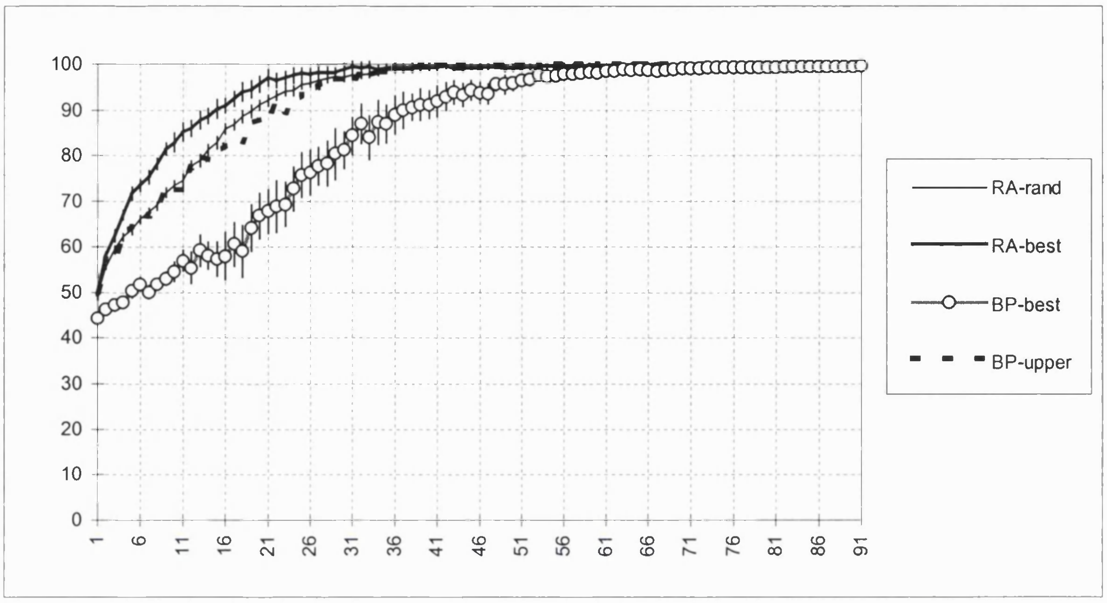


Figure 5.9. Comparing the performance of RA and BP. The average time courses of learning with optimal parameter combinations for BP (step-size 0.02, steepness 0.45, momentum 0.9) and RA (W=10, ψ=0.577) are plotted, (±1 S.E.M., n=12). The average learning curve for W=10 with random ψ is also shown (±1 average S.E.M. for the same 12 initial conditions, using 4 random sequences). For reference, the highest of any of the observed BP learning curves at each epoch is shown dotted.

It may be that better performance can be achieved through combinations outside the tested set since the observed optimal combination was at the extreme of the investigated set. Note however, the range of definition for momentum is (0, 1); see (Eq. 3.3b). When it is <0.5, the effect of momentum is too weak and when it is too close to 1, it destroys learning. The value 0.9 emerged in our simulation is indeed the most commonly used value for it (Rumelhart, *et. al.* 1986; Tugay *et. al.*, 1989; Tollenaere, 1990; Müller *et. al.,* 1991; Hassoun, 1995). The basic BP algorithm is most sensitive to step-size. Steepness merely has the effect of scaling the effective step-size in learning; see comments following (Eq. 3.3a).

At its best, RA seems superior to the basic BP simulated for learning the CT task, particularly at the early stages. Note that the upper bound (at each epoch) of *all* observed BP learning curves is also plotted (dotted). At each stage of the learning, none of the BP trials with any tested parameter combination was above this line; this ensures that one does not bias the comparison of early stage performance unfavourably to BP by having selected the so-called 'optimal' parameter combination solely according to the asymptotic performance rather than some early performance.

As is apparent from Fig. 5.9, BP was more variable than RA. This is not only true for the optimal parameter combinations. Variation was more pronounced for poor combinations, as for RA. Greater variation means greater difficulty in determining whether a trial is worth continuing with the ongoing parameter combination. In other words, a potentially important characteristic of RA that distinguishes it from BP is the extent to which one can predict the 'goodness' of a parameter combination by looking at performance during early stages of learning. Consider, if one ranks all the parameter combinations according to their performance at the end of the *n*th epoch, how sure can one be that this rank order will persist as learning continues? This can be measured directly by *the correlation* between the *intermediate ranking* and *the ultimate ranking* of parameter combinations. The *intermediate ranking* is the one determined by performances at the end of an *intermediate* epoch. The *ultimate ranking* is determined by the asymptotic performance at the end of the 90th epoch. (There may be joint No. 1's and so on in the ranking.) The intermediate ranking may differ from the ultimate ranking but should converge to it as learning goes on, by definition. One expects the correlation between the two to start from around 0 and converge to 1.

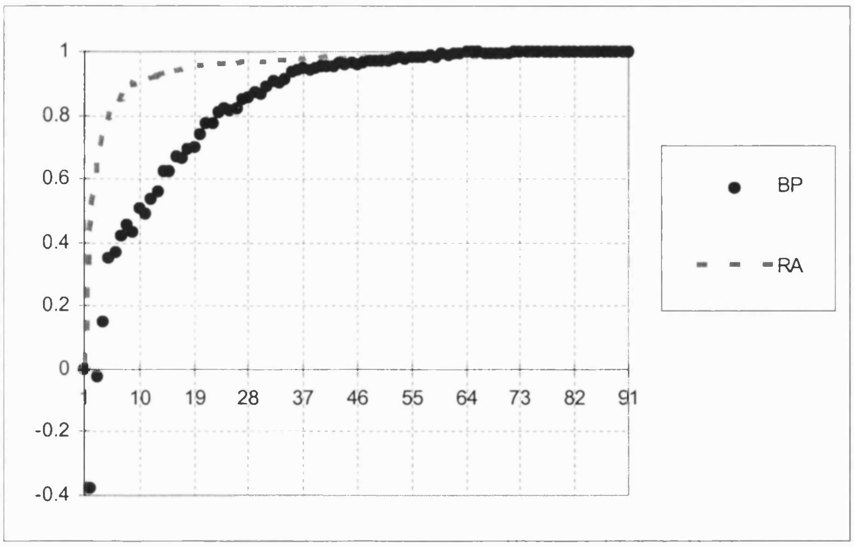


Figure 5.10. Convergence of the ranking of parameter combinations. At the end of each epoch, parameter combinations (143 combinations tested for RA; 87 for BP) were sorted according to their average performance. The linear correlation between the resulting *intermediate* ranking and the *ultimate* ranking, obtained at the end of the 90th epoch, is plotted on the vertical axis. Large correlation indicates greater predictability of the ultimate 'goodness' of any particular parameter combination from its early performance.

It is hard to make such a comparison truly fair. This is largely because of the lack of comparability of the learning parameters of RA and BP. One might, for example, improve the apparent correlation for one condition in such a comparison by including more inappropriate step-sizes, which will give trials that are easily distinguishable from good step-sizes early during learning. A fair comparison should include in the sampled parameter space a 'natural mix', in some sense, of good and bad combinations, which is not a criterion that is easily formalised and met. However, note that at the 10th epoch the correlation is roughly 0.5 and 0.9 for BP and RA respectively. This would have required a very large bias of the sort described, but the issue is not pursued further.

A higher predictability of the outcome of RA, based on earlier performance, may perhaps be seen as due to an apparent defect of RA, compared with BP. For BP, even non-optimal parameter combinations can usually achieve reasonable performance at the end of the 90th epoch, or with a larger number of epochs. They would ultimately achieve perfect learning with sufficient epochs; after all, the theory of gradient descent guarantees this. This is *not* true for RA: an asymptotic level, good or bad, is achieved quickly, and no amount of further iterations can improve it. To improve performance, it is necessary to change to more appropriate parameter combinations altogether.

However, by way of compensation, the predictability of ultimate performance and the fast convergence to it can be used to circumvent the problem. As suggested in Section 4.2.4, these properties may explain the surprising success (see Figure 5.8) of the technique of random-tuning of reverse activation strength.

5.2.2 Mirror Symmetry Discrimination Task

The second bench-mark task studied, the mirror symmetry task, involves discriminating 3 types of symmetry possessed (exclusively) by binary patterns on a 4×4 grid; these are left-right, top-down, and one of the possible diagonal symmetries. For this task, the standard 3-layer network has a configuration of 16-12-3 (cf. Peterson *et. al.*, 1989).

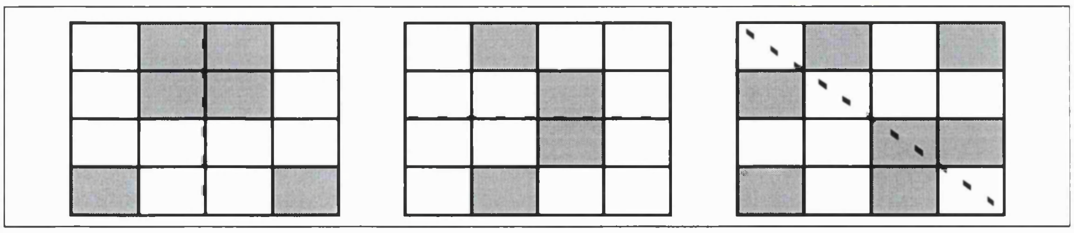


Figure 5.11. Examples of the three types of symmetries to be discriminated. The dotted lines indicate the axis of symmetry.

The standard training arrangement for this task (Peterson *et. al.*, 1989) is adopted as follows. Training is carried out on a set of 100 randomly generated sample patterns with activity ratio falling into a chosen range (patterns generated with activity ratio outside the range are rejected), each having *one and only one* of the 3 symmetries with equal probabilities, subject to the activity ratio range requirement. Training stops when performance is 100% on these patterns. Generalisation is then tested on another non-overlapping set of 100 random patterns subject to the same constraints. The training and testing procedure is repeated for 10 sets of *randomly generated* initial weights.

Other aspects of the simulation are similar to the CT problem. The parameter combinations tested (for each initial condition) were as follows

$$\{\psi = 1.0818, 1.1709, 1.2685, 1.3764, 1.4966, 1.6319, 1.7856, 1.9626, 2.1692, 2.4142, 2.7106, 3.0777, 3.545733, 4.1653, 5.027339, 6.313752, 8.448957, 12.7062, 25.4517\}$$
$$\times$$
$$\{W = N_H\alpha = 1, 2, 3, 4, 6, 8\}. \qquad \textbf{\textit{(Eq. 5.2)}}$$

There were thus 19 × 6=114 (W, ψ) combinations for each of the 10 sets of initial weights. The reverse activation strengths, expressed in terms of tilting angles, range from 47.25 to 87.75 degrees. The performance for angles outside this range was far from optimal and therefore not systematically tested. Fig 5.16 provides the clue for why this is so: the initial H-cell activation scatter is very elongated, i.e. the sensitive region corresponds to larger values of reverse activation strength; smaller values or equivalently, smaller angles, simply do not effect enough changes to existing internal representations. Likewise, the activity ratios outside the tested range all have far from optimal performance as the data will soon show.

The momentum term: smoothing in addition to RA

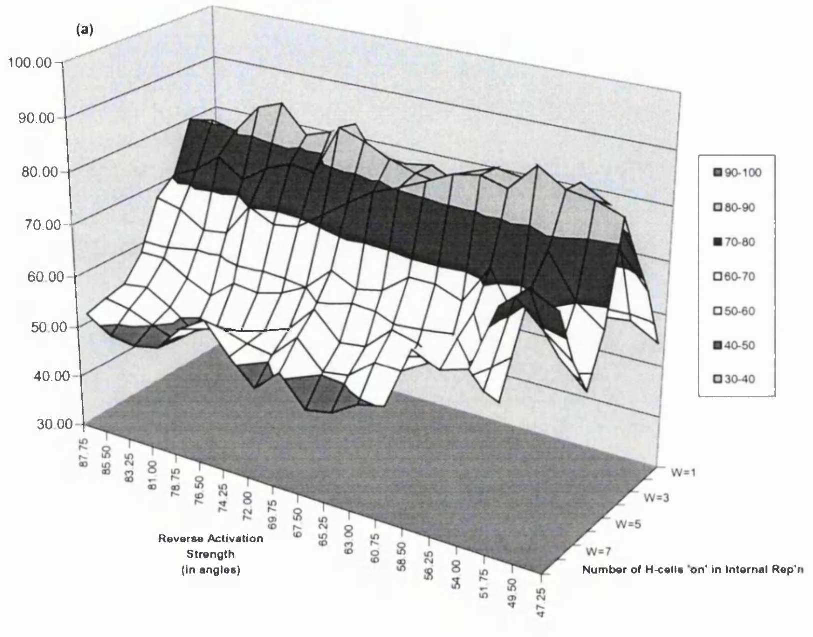
Recall that any learning algorithm can be supplemented by momentum smoothing, regardless of the details. The algorithm in use, what ever it is, calculates the weight modification required for the current step according to that algorithm. The momentum term simply allows the weight modification carried out in the previous step to make a weakened contribution in the current step. See (Eq. 3.3b) in Section 3.2.2. This smoothes out the learning dynamics over the error-surface in weight space and makes convergence more reliable. Smoothing is particularly helpful when the learning task is difficult.

For RA, momentum smoothing is applied in the same way as prescribed by (Eq. 3.3b). The algorithm calculates the required weight modification as before. The actual weight modification in this step however has an additional, weakened contribution from the actual weight modification that took place in the previous step; so it goes on.

Usually learning is not sensitive to the precise value of momentum as long as it is not exceedingly close to 0 or 1 (Rumelhart, *et. al.* 1986; Müller *et. al.*, 1991; Hassoun, 1995); for detailed investigations in the context of gradient descent/BP algorithms, see (Tugay *et. al.*, 1989; Tollenaere, 1990). Momentum terms ranging from 0.2-0.9 were tried for RA in preliminary simulations for the symmetry task, with little evident difference in performance. Although 0.9, which is the rule-of-thumb optimal number for momentum terms (Müller *et. al.*, 1990; Wasserman, 1989), was finally chosen

Performance dependence on ψ and α

The broad characteristic of dependence of asymptotic performance on reverse activation strength and activity ratio is similar to that observed in the CT discrimination problem. However, the 'area' of optimal combinations was considerably smaller; performance was much more sensitive to these parameters. The overall dependence is illustrated in Fig. 5.12, for both zero-momentum and momentum=0.9. It is clear that the latter gives superior learning results and it appears to give a smoother dependence on parameters. However momentum does not improve, and in some cases worsens, the performance of extremely non-optimal parameter combinations.



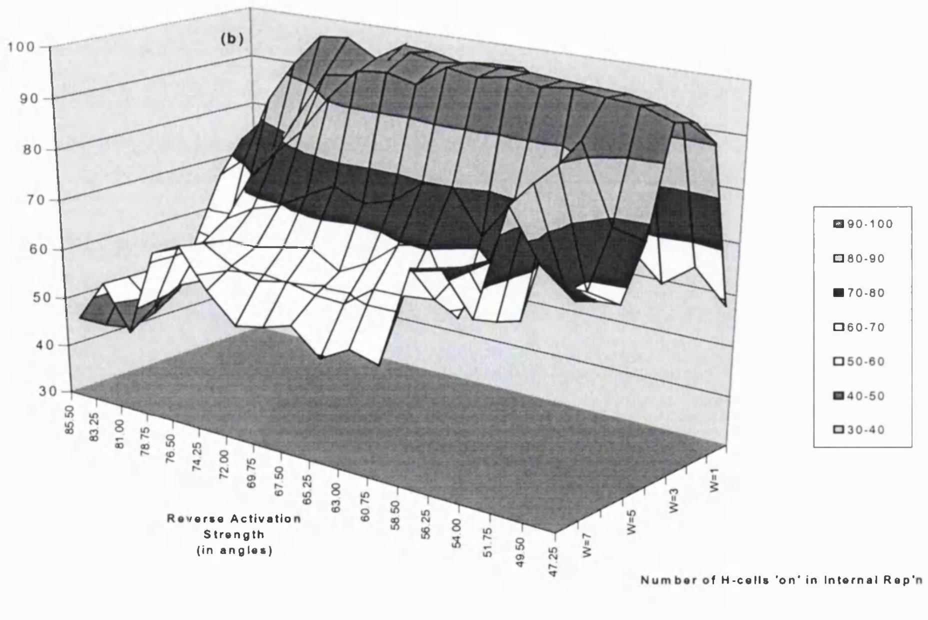


Figure 5.12. Performance vs. (W, ψ) for symmetry discrimination. Plots are for (a) momentum=0 and (b) momentum=0.9. Percentage performance after 90 epochs is plotted against combinations of W and ψ. Points shown for W=5 and 7 are interpolated from adjacent points. Otherwise, points are means for 10 sets of initial conditions. Standard errors are not shown, but for (a) were up to 7 on the 'slopes' and were mostly 2-3 on flatter regions. For (b) Standard errors were smaller.

Performance fell sharply for $\psi<1.08$ (or 47^{o}) in preliminary simulations, though the fall is not clear over the range of values studied for this chart). The optimal reverse activation strength 'scatters' in a wider range than it does for CT, thus is more difficult to tune. For W=1, it ranges from 1-5 (45-80 degrees). For W=2,3,4 it is more critical and shifts toward the lower end of the range. The fact that the optimal values of ψ are larger than for CT is largely a consequence of the relative variations of I and O activation, with an elongated scatter (Fig. 5.16).

The activity ratio proved the more critical parameter in these simulations, with the optimum ranging from 0.16 (W=2) to 0.33 (W=4), instead of the value 0.5 observed for the CT simulation. There are different ways of considering an optimal activity ratio: 1) the value at which the greatest average asymptotic performance can be achieved with the best choice of ψ, or 2) the activity ratio that most often turns out to be optimal for a fixed value of ψ, or 3) how often, regardless of ψ, performance exceeds a reasonable threshold level. The data of Fig. 5.12b are re-analysed in Fig. 5.13 to show the optimal activity ratios, using all three indicators. Note that maxima may be shared between activity ratios with equal performance; points within ± 1 S.E.M. of one another were treated as equivalent. All three criteria identify W=2-4 as optimal.

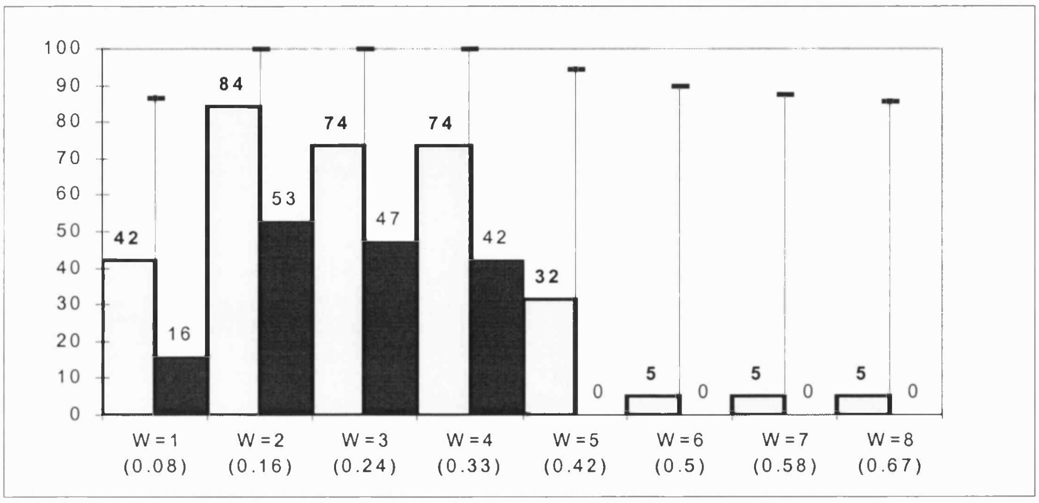


Figure 5.13. Optimal Activity Ratios for the Mirror Symmetry Task. Three different ways of evaluating different activity ratios on the H layer are shown. The vertical lines show the greatest average performance (for any ψ), equivalent to the peak values in Figure 5.12b when W is kept constant. The dark histogram shows the fraction of the points in Fig. 5.12b, for a particular activity ratio, that are optimal for the corresponding value of ψ, while the light histogram shows the fraction of these points that exceed a 77% performance criterion (chance = 33%)..

In Section 4.3.4 the theory suggested that for a given output activity ratio (here 33%), good inversion by the transpose matrix required that the activity ratio on H should be smaller (<33%), as shown here for good learning performance with RA. Clearly also, W must not be too small. Activity ratio affects learning also through other factors.

Performance when reverse activation strength ψ is random

The optimum value of ψ (fixed during learning) depended, for this task, on the activity ratio and was in some circumstances fairly critical (Fig. 5.12b). As with the CT task, it might be possible to resolve this difficulty if ψ is allowed to fluctuate randomly. In the CT task, learning performance so achieved matched the best achieved when ψ was fixed (5.1.2). For the symmetry problem, this only proved to be the case for the lowest activity ratio (W=1), as shown in Fig. 5.14 where the average time courses of random-ψ learning and for the best fixed value of ψ are plotted for W=1, 2, 3, and 4.

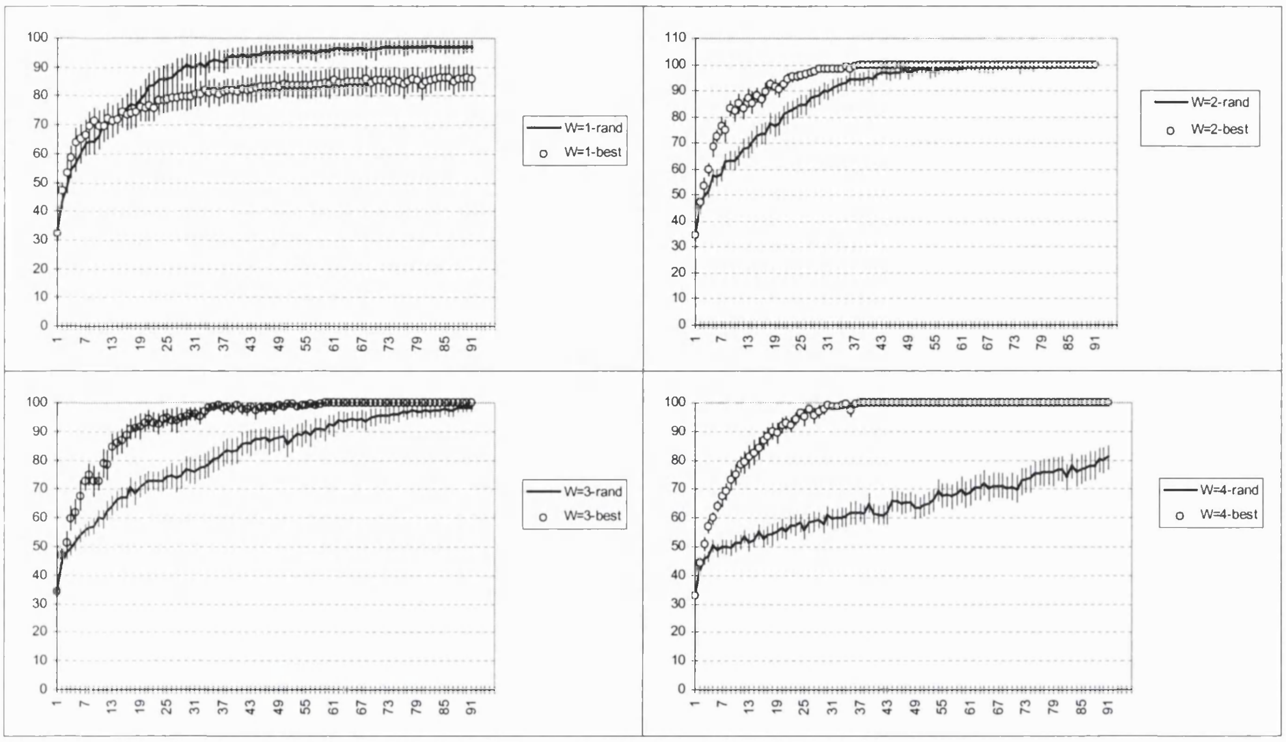


Figure 5.14. Fixed and random ψ values with the mirror symmetry task. For random ψ variations, the average is taken over 40 trials: 10 initial weight sets, each repeated with 4 set sequences of independently selected ψ values. Average standard errors (n=10, averaged over the 4 repeats) are shown. The curve for fixed ψ (mean ± 1 S.E.M.) is for ψ giving the best asymptotic level for each W.

Analysis of variance again revealed that variation in asymptotic performance came mainly from the initial weights. The randomness of reverse activation strength contributed less than 0.5% of the total variation in the case of W=1, and less than 5% for W=3 and 4. When W=2, there was no variation in the observed asymptotic performance.

Note that the greater the activity ratio, the more learning suffered by having to allow the reverse activation strength to fluctuate randomly (and hence to take bad values). Why this is so is not clear. There is room to improve the random tuning technique, as discussed in Section 6.2.

Comparison with Back-Propagation

As for the CT task, the BP code from (Müller *et. al.*, 1991) was used in a comparison with the RA algorithm. For the mirror symmetry task, there is the added advantage of having BP data on precisely the same task from the literature (Peterson *et. al.*, 1989) as an independent yardstick. Extensive sampling of the learning parameters are carried out in the same way as for the CT task. The best parameter combination emerged in this set is: step-size 0.04, steepness 0.45, and momentum 0.9. The same comment following Fig. 5.9 applies here also.

To make the comparison fairer to BP, all performance levels are normalised in order to account for the fact that BP may start learning with less than chance level (33%) performance (unlike RA, it does not have a built-in mechanism for ensuring that only one output cell is on). *Normalised performance* levels are defined as the ratio between the difference in the absolute performance level of the current epoch and the initial level, and the difference between the target absolute performance level (100%) and the initial level. In other words, it shows what proportion of what is left to learn (i.e. the difference between 100% and the initial level) has been learned at any point in time.

It was found that BP typically requires over 100 epochs (150-200) to learn the task or to reach near-asymptotic level. Although most of the learning is done within 100 epochs, the convergence from this point on is usually painfully slow. This confirms the observations of (Peterson *et. al.*, 1989), where BP is compared with another algorithm on the same task and the number of epochs for BP on the same task are quoted as typically 150 epochs.

Below, the observed time courses of performance during training are plotted for up to 100 epochs. *Normalised performance* level is used; it is defined as

(Performance_j – Performance_0) /(100% – Performance_0)

where j=0,1,...100... is the epoch counter with 0 indicating the initial condition. This removes some of the bias caused by the fact that BP does not have ramped binarisation and hence tends to start with a worse performance level than RA initially.

In selecting the 'best' parameter combination for BP for this comparison, one looks for not only the highest average performance at the 100th epoch but also the fastest progression in prior epochs. Further, the upper bound of *all* simulated BP trials at each epoch is also plotted: no observed BP trials could rise faster than this line.

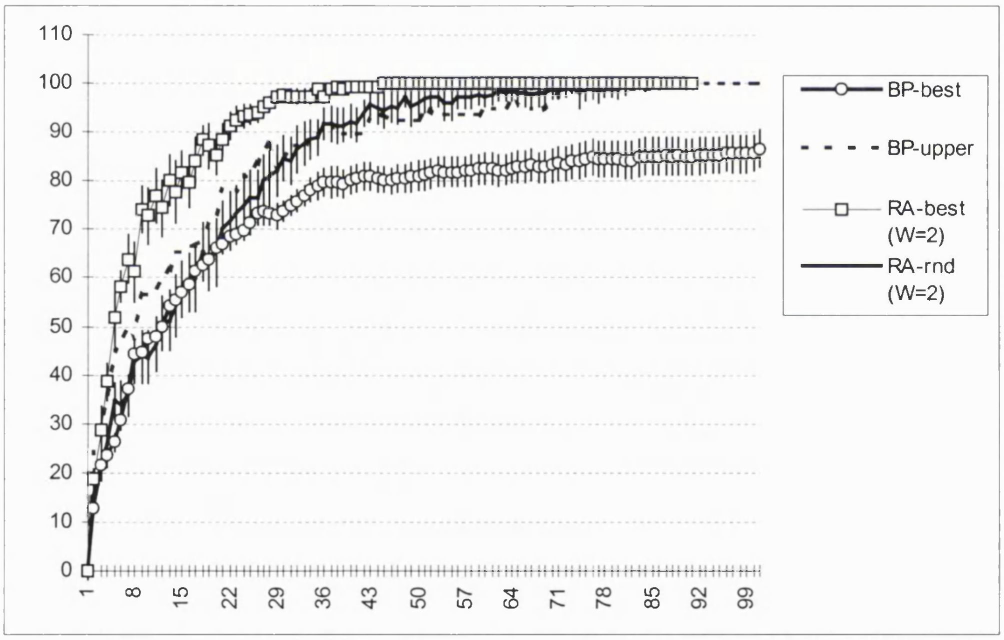


Figure 5.15. Comparing the performance of RA and BP. The vertical axis measures the *normalised performance level* (defined the text). This shows how fast each algorithm learns what remains to learn given its particular starting level. For BP, the average performance of the best parameter combination (step-size 0.04, steepness 0.45, and momentum 0.9) is plotted ± 1 S.E.M., calculated from the 10 independent repeats. In addition, the normalised absolute upper bound for all observed BP trials is shown as the strong dotted line. For RA, the best average time courses, corresponding to W=2 and ψ=2.71 (69.75 degrees), and also random-ψ learning with W=2, are plotted ± 1 S.E.M. (see Figure 5.14 for details).

It is also instructive to compare the different effects of BP and RA learning on connection weights. Plotted below is the activation scatter of H cells before and after learning, given the initial and resulting weight matrices respectively and the 100 pairs of patterns in the training set.

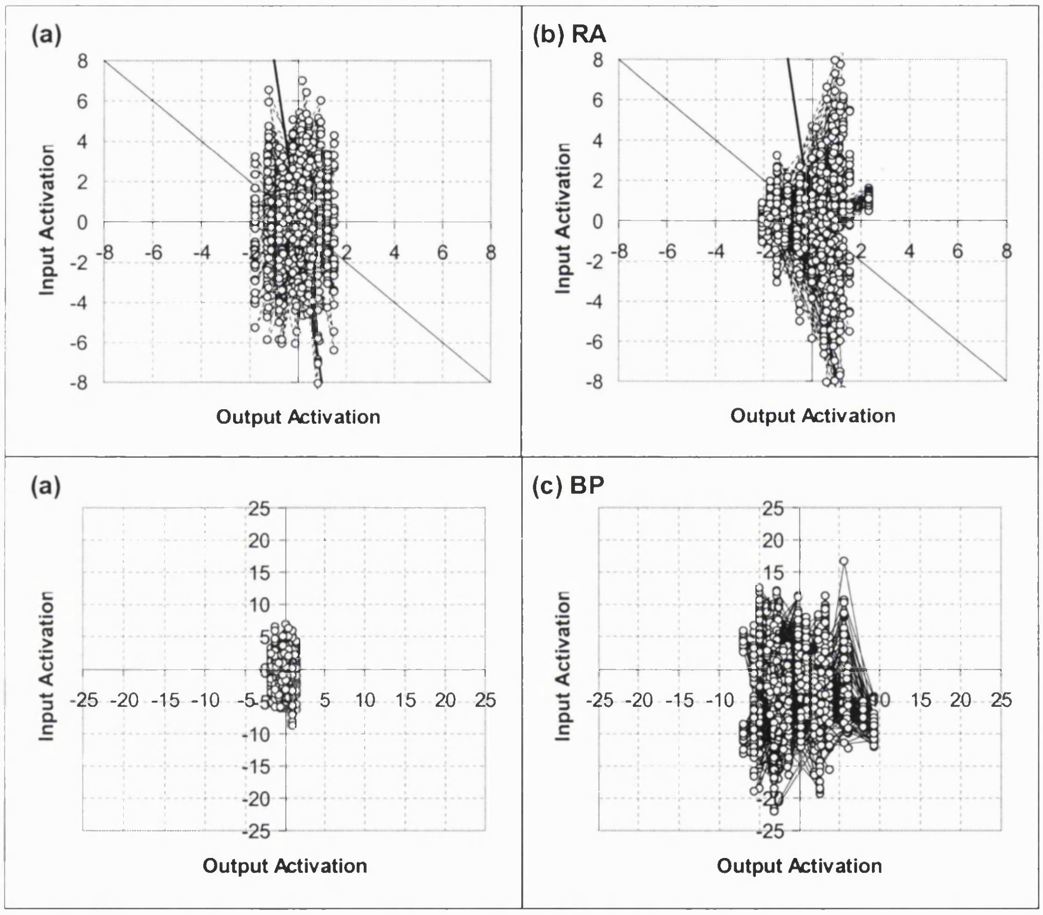


Figure 5.16. Different effects of BP and RA learning on weights. The activation scatter of H cells given (a) the initial weights, (b) the corresponding weights resulted from RA learning with random-ψ and W=2, (c) the corresponding weights resulting from BP. (a) is plotted twice on different scales. Activation strengths 1 (45 degrees) and 8 (82 degrees) are represented on chart (a) and (b).

Firstly notice the extremely elongated initial scatter, partly due to the fact that the input patterns contain more on-cells than do the output patterns (roughly 8-to-1 compared with the 5- to-1 for CT); comparison can be made with Figure 5.3 for CT, where the scatter is similar on the two axes. This is one of the factors behind the fact that all optimal activation strengths observed for this task are bigger than for CT.

Although chart (b) corresponds to random reverse activation strength during RA learning, the effect of RA is clear: H cells that once received large input activation but were detrimental to producing the correct outputs (indicated by their negative reverse activation) are either shifted to the right horizontally or down vertically. This results in the fan-shaped scatter of (b). The fan-shape was evident for the separate graphs for each of the 10 initial conditions and activity ratios W/N_H, though less pronounced for large W. Apart from this characteristic change, the distributions of activation and weights were little changed..

In contrast, for BP, the scatter of activation was much altered (Fig 5.16d). Firstly, note the dramatic (approximately 3-fold) increase in activation. Direct observation confirmed that weights increased from the initial 0-1 range to 0-10, consistent with the $N^{1/2}$ growth formula (cf. Section 3.2) for weights subject to BP learning. This partly explains the slow convergence. Secondly, the shape of the activation scatter for weights trained by BP was less easy to characterise.

Generalisation Performance

After the network has learned perfectly the 100 training input-output maps, by whichever algorithm, one can test for generalisation. Poor generalisation indicates that the learning algorithm has learned on the basis of features of the training set other than the symmetry differences, on the basis of which they were chosen.

The procedure was taken from Peterson *et. al.* (1989). Training and testing sessions were carried out separately for 6 randomly generated training sets. Each of these sessions was repeated 10 times with independently generated initial weights. The mappings in the 6 training sets were randomly generated according to the specific criteria set out at the beginning of Section 5.2.2. The 6 sets were divided into 3 groups of 2 each, labelled A, B, C. Inputs in Group A had the lowest range of activity ratio, as indicated by the group average of 0.4; group B contains patterns with intermediate range of activity ratios with a group average of 0.5; group C has the highest range of activity ratios here with a group average of 0.6. Associated with each group is a third set of 100 mappings (of the same specification as the group) for testing generalisation performance. Thus, a naive network is trained on training set A1 and tested on A3 for generalisation (repeated for 10 independent sets of initial conditions); the same exercise is carried out on A2 (with the same initial conditions), tested on A3. This way one has 2 separate estimates for generalisation performance (measured by percentage-correct on the corresponding, non-overlapping, testing set) for group A and likewise for B and C, 6 estimates in total.

Thus there are, in all, 9 sets of sample patterns of 100 each into 3 groups, randomly generated according to specifications; *none of the 3 within each group has common patterns*. It may be useful to note that on the 4×4 grid there are about 1500 patterns having *one and only one* of the three symmetries to be discriminated in the task (Peterson *et. al.*, 1989). Thus the above set up is possible.

The table below summarises the performance in these tests. Also listed is data from (Peterson *et. al.*, 1989) for performance of the Mean Field algorithm (MF), which is a form of gradient descent learning algorithm. For BP learning, generalisation performance is better if one uses the 0-1 binary representation during learning (Peterson *et. al.*, 1989) (On the other hand, it is known that learning is faster if the (-1)-(+1) representation is used (Peterson *et. al.*, 1989)). Apart from the RA data and the BP data for (-1)-(+1) representation, results are taken from (Peterson *et. al.*, 1989).

	Group A	**Group B**	**Group C**
***RA-rand*-ψ (best)**	***76%, 78%; 77%***	***80%, 78%; 79%***	***68%, 74%; 71%***
MF (best)	68%, 57%; 63%	67%, 57%; 62%	70%, 69%; 70%
BP (best for (-1)-(+1) rep'n)	49%, 46%; 48%	52%, 48%; 50%	59%, 62%; 61%
BP (0-1)	69%, 59%; 64%	67%, 59%; 63%	69%, 72%; 71%

Table 5.1 Generalisation Performance. For each algorithm and each test group, which contains two training sets, the median generalisation performance for set 1, 2, and the average of the two medians are listed in that order. For RA and BP, the medians are calculated for trials with the best parameter combinations for learning the training sets (see Captions for Figure 5.14&5.15). Generalisation was tested after achieving perfect performance on the training set. Data for Mean Field learning and BP learning with the (0-1) representation are the best median results reported by Peterson *et. al.* (1989).

The above table suggests that RA learning tends to give the best generalisation performance. For the RA and BP data obtained in the present simulation, the average performance is usually within 1% of the median with standard error less than 3%, all calculated on the 10 repeats with independent initial weights. The BP data above confirms those in (Peterson *et. al.*, 1989).

Chapter 6 Discussion and Conclusions

We set out to develop a pattern-centric learning algorithm for multi-layer perceptron-like networks. In this approach, internal representations are constructed first; they then drive weight changes (via simple perceptron rule). This is the exact opposite of the standard weight-centric approach to learning where internal representations are byproducts of weight changes, which are calculated from some error/energy function. The pattern-centric approach is more consistent with research on sensory representation in the brain and provides a more direct link between ideas in that area and network modelling. Unlike the disappointing pattern-centric attempts in the past, the RA algorithm does not involve any cumbersome search mechanism in the vast pattern space. Instead, the algorithm tries to involve processes that resemble known biological mechanisms in the brain. In RA, internal representation is constructed directly by a (non-linear) superposition of activation of input and output on the representation layer.

Theoretical analysis of RA, carried out in Section 4.2-4.4, demonstrates the rationale behind the key elements of RA, namely, (symmetric) reciprocal connections, adjustable reverse connection strengths with possible random fluctuations during learning, and ramped binarisation. Given plausible assumptions (Section 4.3), one can show pair-wise independence and non-positive correlation, respectively, between connection weights from the same hidden layer cell and between connection weights converging to the same output layer cell in a network that uses perceptron-like learning. In this context, symmetric reverse connections are capable of inversion operation (from output patterns to hidden layer representations). The quality of the inversion depends on activity ratios (of the output and internal representation patterns). This is one of several ways through which activity ratio can play a part in learning in such networks. RA is also extended to networks with more than one representation layer (Section 4.4).

Subsequent simulation on 3-layer networks demonstrates the feasibility of the ideas discussed in theory. On the toy problems studied, RA performed learning, consistent with our theory. The performance is comparable to the basic BP algorithm and the data set suggests that it can be made better than basic BP in terms of convergence speed and generalisation. RA algorithm also performed adequately when reverse connection strengths are not explicitly tuned but are allowed to fluctuate randomly during learning. The simulation raised some interesting questions on the role of activity ratio, the random tuning technique and its improvement, and generalisation after training. These are discussed below along with some technical issues.

Section 6.1 Optimal Activity Ratios

Simulation data for the CT and the symmetry task, particularly the latter, demonstrates the crucial role played by activity ratio of internal representations. This is possibly mainly due to its effect on the accuracy of the transpose as an inverse operator (as analysed in Section 4.3.4), i.e. the ability of the transpose to deliver accurate information on the effectiveness of H cells in evoking right outputs.

What the simulation also demonstrates is the highest inversion quality (by having H layer activity ratio much lower than that on the O layer) does not correspond to the fastest learning however. One reason for this may be the fact that given the small size of the network, low activity ratio too severely limits the representational capacity of the H layer hence reducing the degrees of freedom, making problem solving more difficult. A rule of thumb for choosing optimal activity ratios in practice therefore seems to be the following:

the activity ratio of the internal representation should be small enough (at least less than the activity ratios of the output layer) to ensure reasonable accuracy for the inversion operation performed by the transpose matrix. It however should not be too small to the extent of limiting the representational capacity of the hidden layer too severely. In general, the activity ratio that enables fastest learning is probably the biggest ratio that still allows 'reasonable' inversion by the transpose. In case of simple output patterns (1-'on'-cell only), this ratio is 1/NO according to Section 4.3.4.

It will be interesting to carry out more simulations to explore how optimal activity ratio varies with output activity ratios as well as the size of the H layer in RA learning.

It is important to stress here that the above concerns only the speed of learning (the training set). Faster learning does not mean better learning. Generalisation quality is arguably the ultimate measure for the quality of learning achieved.

Section 6.2 Random Tuning of Reverse Activation Strengths

6.2.1 Purposeful Random Fluctuation

The one key features of the RA algorithm is fast convergence (but not necessarily to the right state). This offers clues as to how the ψ-tuning problem can be solved. As demonstrated repeatedly in the last section, convergence is particularly fast for very non-optimal values of ψ.

This has been the basis for allowing ψ fluctuating randomly during learning. The key inference drawn from the data on performances was that training settles into a state in which weight changes tend to cancel each other very quickly when the value of ψ is far from optimal. Examples of direct recordings indeed support this idea.

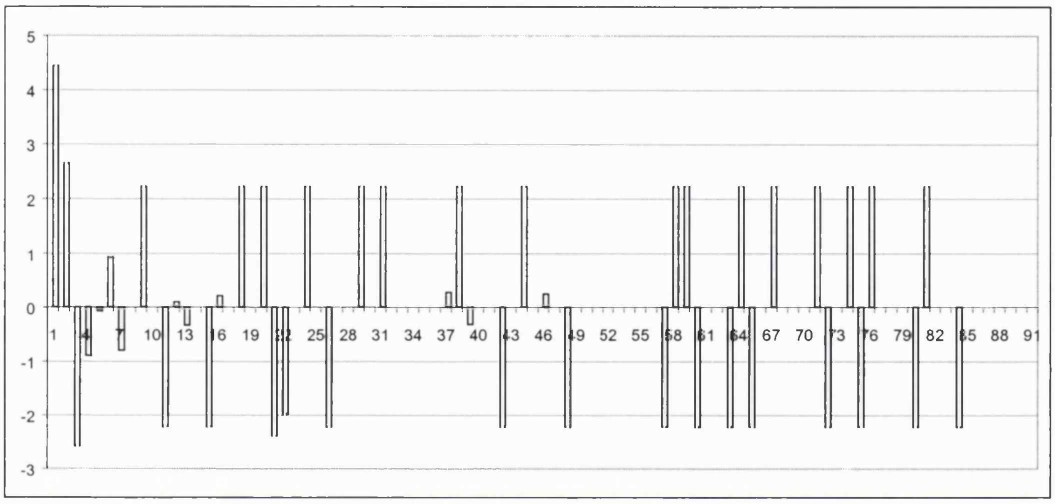


Figure 6.1. Average net weight changes during learning against time (epochs). The net change to each I to H connection weight, measured in units of step-size, during an epoch are recorded and averaged (connections with net zero change are excluded from the averaging process). Thus during the first epoch, net weight change made were on average 4.3 step-sizes. Note the strong tendency for any weight change to be precisely reversed during the epochs that followed so that from the 10th epoch onwards, the net effect of the learning on weights are almost zero (precisely zero for the last twenty epochs or so).

For the better ψ values however, by definition, weight changes have less tendency to cancel each other out (otherwise performance will not improve). Thus when ψ is allowed to fluctuate randomly during learning, one expects the network to benefit from weight changes when the ψ value happens to be good *on average* since the net effect of bad ψ values on weights converge to zero *on average.*

As seen in the last section, randomly fluctuating reverse activation strength does result in very effective learning. However as Figure 5.14 clearly demonstrated, learning slows down because of the noise introduced by the bad ψ values. The case of W=4 in Figure 5.14 provides the clearest evidence for the explanation offered in Section 4.2.4 as to how the random-tuning technique works. It can be seen there that learning was taking place on average despite the evidently detrimental effects of the bad ψ values.

The learning curve strongly suggests that perfect performance would eventually be achieved given more epochs. These results encourage one to develop the idea further to better take advantage of the above property of the RA algorithm.

One obvious extension is to allow the probability distribution from which the values of activation strength ψ are randomly drawn to evolve so that it can become more and more localised at the optimal values. This may be called *'purposeful' random tuning* strategy whereas the original technique shall be called *blind random tuning.*

At the centre of the purposeful random tuning strategy is a 'fitness' measure for any particular values of key parameters, ψ in this case. For each parameter value, the measure is evaluated based on history of the network behaviour during periods when that value happens to be 'in charge' by chance; these periods are sampling periods for that particular value. The fitness measure for each parameter value is updated whenever that value is used. The probability distribution is then changed incrementally according to the latest fitness numbers so that it becomes more and more concentrated on the most suitable set of values.

Several simple-minded but quickly available 'fitness' measures have failed to deliver any clear improvements. One may have to allow longer sampling periods, counted in epochs rather than patterns.

This opens a new possibility. One obvious basis for measuring fitness if sampling periods are extended to epochs is the performance at the end of an epoch. In this case, ψ is allowed to fluctuate from epoch to epoch, rather than from pattern to pattern. The performance improvement (or the lack of it) at the end of the epoch is then recorded and used to see how it ranks amongst all the latest records for other values of ψ. The probability distribution can then be modified incrementally according to the new ranking order. Recall Figure 5.10, which shows that there is a strong predictability of the ultimate fitness of ψ from its performances in early epochs.

6.2.2 Applying a Population Search Algorithm

Figure 5.10 inspired another tuning method, which may be seen as a Population Search Algorithm. One can start with a population of otherwise identical networks learning with the RA algorithm, each with different reverse activation strength ψ. Every 2-5 epochs, those networks whose performance is not in the top 50% of the population can be terminated. After several round of elimination only a small number of networks, for which the ψ value most probably will be optimal.

Figure 5.10 suggests that this method will be quite effective. The ranking according to the performance of the 10th epoch is more than 95% correlated with the ultimate ranking. It is quite remarkable considering that this means that the ordering of performance is almost fixed as early as the 10th epoch. Note that for the above elimination method to work, one does not even require the strong correlation of detailed rankings. All one needs is that if a ψ value belongs to the top half of the entire population at the beginning it should be highly likely for it to remain in the top half in the end.

Further, this method may be combined with the blind random tuning technique. Given that usually there is a range of activation strengths which are optimal, one can cut short the above elimination process and start applying blind random-ψ learning on the remaining range of ψ values. For instance for the CT and mirror symmetry problem, one elimination process should be enough to enhance the speed of random-ψ learning dramatically, cf. Figure 5.6 and 5.12.

Section 6.3 Generalisation Performance

The data on the mirror symmetry task suggests better generalisation performance for RA learning. More simulation is needed to confirm the assertion.

One reason for this may lie in the nature of how multi-layer perceptrons work. A general feature of such neural networks is that 'similar' inputs tend to evoke 'similar' outputs. Generalisation fails when two 'similar' or 'dissimilar' inputs are supposed to evoke, respectively, two 'dissimilar' or 'similar' outputs. Successful generalisation relies on the creation of internal representations that increase or decrease the similarity between the input patterns as the case may be. RA learning implements this idea very directly. Recall that in the RA algorithm, internal representation is selected by imposing the input and the output pattern simultaneously. By effectively 'appending' the output to the input pattern when selecting internal representations, one increases or decreases the similarity between any pair of input patterns and hence the similarity between their respective internal representation patterns according to whether the associated outputs are the same or not. Thus, at the heart of RA learning, there is a built-in mechanism that directly benefits generalisation capability.

There is perhaps a more profound factor at work in case of RA logarithm. It concerns activity ratio fixing, which will be discussed in Section 6.5.

Section 6.4 Technical Questions That Require Further Investigations

6.4.1 Comparisons between Variants of BP and RA

Although simulations demonstrated that the default version of RA compared favourably with basic BP, further comparisons with the numerous improved variants of BP are necessary in order to gauge RA's practical potentials.

For instance, it is known that BP can learn faster with on-line updating (than with batch updating) provided that step-size is appropriately scheduled to decline to zero during learning. Detailed work and review on these variants can be found in (Fahlman 1988; Wasserman, 1989; Hassoun 1995; Ripley 1996). Owing to the difficulties in tuning such variant BP's and the fact that these techniques could be applied to RA as well, comparisons of variants of both types of algorithm are omitted in this developmental stage. Classical BP, one with batch updating and momentum smoothing, is closest to the underlying gradient descent idea behind the algorithm. Likewise, default RA, one with total-on-line updating, fixed or random activation strengths, and momentum smoothing, is closest to its original derivation based on biologically plausible learning mechanisms.

The on-line version of both BP and RA were tried. However, results in trial runs were too erratic and the attempt on systematic simulation was abandoned, as it may be unduly complex. It is more appropriate to carry it out as a separate project. The preliminary simulation suggests the following. The total-on-line version of BP (which seemed to be absent in the literature) can produce great performance but with higher sensitivity to initial weights. It may be less sensitive to step-size compared to other BP as long as step-size is not too big (Section 5.1.1). However, the number of iterations required for a single mapping was counted in the hundreds during the early epochs; it was also prone to be trapped in local minima even when learning a single input-output pair. To put these in context, for the standard version of RA, which is total-on-line, the number of iterations required to learn single mappings in early epochs rarely exceeds 10. Further, it *never* fails to learn any single input-output pair.

The batch version of RA in test trials seemed to be more sensitive to step-size than the default RA, as expected from Section 5.1.1. More fine-tuning is needed. Together with additional parameters such as reverse activation strengths to tune, this version is exceedingly cumbersome.

6.4.2 Simple and Complex Outputs

In the simulation so far, the output patterns, which correspond to exclusive categories, have consisted of simple outputs: patterns having all but one cell turned off.

In theory, complex outputs do not represent additional computational complexity. This is because the internal representations that can solve a problem with complex outputs can also solve the equivalent problem in which the outputs are transformed to simple patterns (by assigning each distinct complex output pattern with an exclusive unit), and importantly, vice versa.

This is easily proved in the following way. Given a complex problem, first let RA solve its equivalent version with simple outputs. Then, keep the I-to-H weights (and thus the internal representation). The H-to-O weight matrix that will solve the complex problem is simply the existing H-to-O weight matrix (found by RA learning in the simple-output equivalent) multiplied by the *fixed* matrix consisting of only 0's and 1's which maps each simple output pattern back to its original complex form. The process can also be reversed.

However, equal complexity does not mean equal 'ease'. It is not immediately clear if complex outputs will be easier or harder for RA than simple outputs. RA should continue to function as long as the transpose still perform inversion 'adequately', which in principle is possible given the appropriate relation between the H layer and O layer activity ratio; see the final three subsections of Section 4.3.4.

Section 6.5 Efficient Sensory Processing and Representation

This section attempts to put the investigation so far into the broader context of efficient sensory processing in the brain. It will present firstly an overview on what it means and then relate these to what role RA can play in exploring these issues.

6.5.1 The Goal of Sensory Information Processing

Sensory information, coded in terms of impulse frequencies, enter the brain via millions of parallel fibres originated from sensory neurons. The process that transforms a raw sensory signal into patterns of activity in high-level cortical neurons is of great interest. Is there are a information-theoretic principle that applies to the transformations carried out in the brain? The overriding principle may be termed as 'redundancy reduction' through successive transformations (Atteneave, 1954; Barlow, 1961). The subtler part of this principle, which is not usually appreciated, is that depending on the type of 'redundancy' referred to, the principle leads to radically different conclusions on the type of coding required to achieve 'redundancy reduction'. These lead to the concepts of compact coding, factorial coding, and sparse coding/combinatorial coding. All can be said to reduce redundancies. A good and perhaps the only review of all these strategies in a coherent context can be found in (Field, 1994).

6.5.2 Different Concepts of Redundancy Reduction

Compact Coding

In the most naive interpretation, based on standard information theory as in (Shannon, 1949), the principle requires the reduction of overall redundancy. This is the same as minimising the set of (binary) symbols coding the signals. Such a strategy is referred to as compact coding and seems to be used in the early stages of sensory processing: 100 million receptors in the retina converging into only 1 million optic fibres, 50 million olfactory receptors (in the rabbits) into only 50,000 mitral cells, and less obviously trichromacy in colour vision, etc.

Factorial Coding

However, in higher cortical processing there is then a great expansion of the number of cells. Why? The answer also lies in 'redundancy reduction' but with a different type of redundancy. Barlow suggested that the detection of association of events is easier in a neural network context if the activity of representational elements is as statistically independent as possible, conditioned on the set of possible input signals (Barlow, 1961; Hentsche, Barlow, 1991; Gardner-Medwin, Barlow, 1992). This essentially traces back to the inability of Hebb-type synaptic learning rules (Hebb, 1949) to code higher order statistics amongst input cells: to a post-synaptic cell, a pair of input cells are indistinguishable from another pair with identical first order statistics but different second order ones.

A factorial code (cf. Schmidhuber, 1992) tries to achieve the above 'independence' as far as possible. Factorialisation is equivalent to reducing redundancy of order greater than 1 in the resulting representation, i.e. reducing the non-independence amongst representational elements.

Sparse Coding/Combinatorial Coding

Sparse coding, where signals are represented by a large number of symbols with a very low activity ratio, is a form of factorialisation (Hentsche, Barlow, 1991). The sparser the representation, the closer the code satisfies the factorial condition. Combinatorial coding, where each signal is given an exclusive representational element, is an extreme form of sparse coding if the resulting activity ratio is very small. There is a trade off of course between sparseness and representational capacity. It is easy to show that in order to achieve better sparseness while preserving representational capacity, the number of binary element roughly grows to the order of $1/(-\alpha\log(\alpha))$, where α is the activity ratio. The great expansion of the number of representational cells in higher cortical areas but with very low activity ratios supports the idea of sparse coding.

6.5.3 Investigating Sparseness in the RA Context

Sparse codes became a subject of interest in the study of simple auto-associative networks of binary units. The critical loading of such networks increases as activity ratio falls (Gardner, 1988). There is no comparable result for feedforward networks with more than 2 layers. Some authors start from the position that sparse codes are useful and consider ways to achieve sparse codes on a 2-layer perceptron-like network. One may for example minimise, via gradient descent, an explicit cost function that relates to higher order redundancies (Hentsche, Barlow, 1991).

The RA algorithm essentially constructs representations with specified activity ratio under the constraint of an explicit input-output task. It on the one hand explicitly constrains the activity ratio of the representational patterns while on the other hand forces the network to solve a given task. If the task is solved using the required activity ratio then the representation achieved must have extracted important features for achieving the task. As such, RA provides an arena for studying the effect of representational activity ratio on solving classification tasks.

Recall Section 4.3.4, where it is found that sparser activity ratio on the hidden layer leads to more accurate inversion by symmetric connection matrix, given ramped binarisation. This is one way sparser representation is preferred in the RA context. However simulation also suggests that the 'optimal' activity ratio, as far as learning speed is concerned, is in fact the biggest activity ratio that is just low enough to ensure reasonable quality of the information transmitted by the transpose about outputs.

Learning speed is far from everything of course. It may be that sparser internal representations, while taking longer to achieve, lead to better generalisation. It is well known in theory and in practice that given any two feedforward multilayer networks of identical performance on the training set, the one with less degrees of freedom tends to generalises better, cf. (Müller et. al., 1991; Hassoun 1995). Usually, 'degrees of freedom' is understood as the size of the hidden layer. However, for RA, 'degrees of freedom' is determined also by activity ratio. Given the same hidden layer, compared with BP or others, the RA algorithm, through activity ratio fixing, has a much lower degrees of freedom. The 'degree of freedom' principle implies that given two networks of identical performance, the one with sparser internal representations (constructed via the RA algorithm) is likely to have a better generalisation performance.

While our simulation data does not contradict this statement, the difference between generalisation performance is not statistically significant. More study into this is therefore needed.

BIBLIOGRAPHY

Abramson, S.; Saad, D.; Marom,E. (1993), "Training a Network with Ternary Weights Using the CHIR Algorithm", *IEEE Trans. on Neural Networks*, 4(6), pp. 997-1000.

Amari, S., Arbib, M. (1977), "Competition and cooperation in neural nets", *Systems Neuroscience* (J. Metzler, Ed.), Academic Press, San Diego, pp. 119-165.

Amit, D. J. (1989), *Modelling Brain Function: The world of attractor neural networks,* Cambridge University Press, Cambridge.

Atteneave, J. J. (1954), "Some informational aspects of visual perception", *Psychol. Rev.,* 61, pp. 183-193.

Barlow, H. B. (1961), "The coding of sensory messages", *Current Problems in Animal Behaviour* (W. H. Thorpe and O. L. Zangwill, Eds.) Cambridge University Press, Cambridge, pp. 330-360.

Baum, E.; Moody, J.; Wilczek, F. (1988), "Internal representations for associative memory", *Biol. Cybern.* 59, pp. 217-228.

Bell, C.; Caputi, A.; Grant, K.; Serrier J. (1990), "Storage of a sensory pattern by anti-Hebbian synaptic plasticity in an electric fish", *Proc. Natl. Acad. Sci. USA*, 90, pp. 4650-4654.

Bialek, W.; Rieke, F.; de Ruyter van Steveninck, R.; Warland, D. (1991), "Reading a neural code", *Science*, 252, pp. 1854-1857.

Box, G.E.; Tiao, G.C.(1973) *Bayesian Inference in Statistical Analysis*, Addison-Wesley, Reading, MA.

Carpenter, G.; Grossberg, S. (1987), "A massively parallel architecture for a self-organizing neural pattern recognition machine", *Comput. Vis., Graph., Image Proc.*, 37, pp. 54-115.

Crick, F.C.; Mitchison, G. (1983), "The function of dream sleep", *Nature*, 304, pp.111-114.
Denoeux, T.; Lengelle, R. (1993), "Initialisation of backpropagation network with prototypes", *Neural Networks*, 6, pp. 351-363.

Desimone, R. (1992), "Neural circuits for visual attention in primate brain", *Neural Networks for Vision and Image Processing,* (G. Carpenter, S. Grossberg, Eds.) MIT Press, Cambridge, MA. pp. 343-364.

Dormany, E.; Van Hemmen, J.L.; Schulten, K. (1995), *Models of Neural Networks: Physics of Neural Networks,* Springer-Verlag, Berlin.

Fahlman, S.E. (1988), "Fast learning variations on back-propagation: an empirical study", *Proc. 1988 Connectionist Models Summer School* (D. Touretzky, G. Hinton, T. Sejnowski, Eds.), Pittsburgh.

Feller, W. (1966), *Chapter* VIII.4, *An Introduction to Probability Theory and Its Applications II,* John Wiley & Sons.

Field, D.J. (1987), "Relations between the statistics of natural images and the response properties of cortical cells ", *J. Opt. Soc. Am.,* 4, pp. 2379-2794.

Field, D.J. (1993), "Scale-invariance and self-similar 'wavelet' transforms: An analysis of natural scenes and mammalian visual systems", *Wavelets, Fractals and Fourier Transforms: New Developments and New Applications.* (M. Farge, J. Hunt, J. Vassilicos, Eds.), Oxford: Clarendon, pp. 151-193.

Field, D.J. (1994), "What is the goal of sensory processing", *Neural Computat.* 6, pp. 559-601.

Finnoff, W. (1994), "Diffusion approximation for constant learning rate backpropagation algorithm and resistence to local minima", *Neural Computat.* 6 (2), pp. 285-295.

Fischer, T.M; Blazis, D.E.; Priver, N.A.; Carew, T.J. (1997), "Metaplasticity at identified inhibitory synapses in Aplysia", *Nature*, 389, pp. 860-865.

Foldiak, D. J. (1990), "Forming sparse representations by local anti-Hebian learning", *Biol. Cybern.*, 64, pp. 165-170.

Fuster, J. M. (1989), *The Prefrontal Cortex, Second Edition,* Raven, New York.

Gardner, E. (1988) "The space of interactions in neural network models", *J. Phys. A*, 21, pp. 257-270.

Gardner-Medwin, A. R. (1976), "The recall of events through the learning of associations between their parts", *Proc. R. Soc. Lond.* B. 194, pp. 375-402.

Gardner-Medwin, A. R. (1989) "Doubly modifiable synapses: a model of short and long-term auto-associative memory", *Proc. R. Soc. Lond.* B, 238, pp.137-154.

Gardner-Medwin, A. R.; Barlow, H.B. (1992), "The effect of sparseness in distributed representations on the detectability of associations between sensory events", *J. Physiol.* 452, pp. 282-300.

Gardner-Medwin AR & Kaul S (1995) "Possible mechanisms for reducing memory confusion during sleep", *Behavioural Brain Research.* 69, pp. 167-175

Gardner-Medwin, A. R., Wren, K. K. (1998), "A well-defined measure for ignorance of a topic", working paper.

Goldman-Rakic, P. S. (1987), "Circuitry of primate prefrontal cortex and regulation of behaviour by representational memory", *Handbook of Physiology Vol. 5.* (F. Plum, Ed.), *American Physiological Society,* pp. 373-417.

Grossman, T.; Meir, R.; Domany, D. (1988), "Learning by choosing internal representations", *Complex Systems Vol. 2,* pp. 555-575.

Grossman, T. (1989), *Complex Systems Vol. 3,* pp. 407.

Hassoun, M.H. (1995), *Fundamentals of Artificial Neural Networks,* M.I.T. Press.

Hebb, D.O. (1949) *The Organisation of Behaviour,* Wiley, NY.

Hentsche, H.G.; Barlow, H.B. (1991), "Minimum-entropy coding with Hopfield networks", *Network* 2, pp. 135-148.

Hinton, G.; Dayan, P.; Frey, B.; Neal, M. (1995), "The 'wake-sleep' algorithm for unsupervised neural networks", *Science,* 268, pp. 1158-1161.

Hopfield, J.J, (1982), "Neural networks and physical systems with emergent collective computational abilities", *Proc. Nat. Acd. Sci. USA,*81, pp. 3088-3092.

Hopfield, J.J.; Feinstein, D.I.; Palmer, R.G. (1983), "'Unlearning' has a stabilizing effect in collective memories", *Nature* 304, pp. 158-159.

Jordan, M.I., (1990), "Motor learning and the degrees of freedom problem", *Attention and Performance XIII* (M. Jeannerod, Ed.), Erlbaum, Hillsdale, NJ, pp. 796-836.

Kawato, M.; Hayakama, H.; Inui, T. (1993), "A forward-inverse network model for visual perception", *Network: Comput. Neural Syst.* 4, 415.

Kohonen, T. (1990), "The self-organizing map", *Proc. IEEE,* 78, pp. 1464-1480.

Kosko, B. (1988), "Bi-directional associative memories", *IEEE Trans. Sys., Man, Cybern.,* 18 NO. 1, pp. 149-160.

Lee, T.S.; Mumford, D.; Romero, R.; Lamme, V.A. (1998), "The role of primary visual cortex in higher level vision", *Vision Research* 38, pp. 2429-2454.

Maloney, L. T. (1986), "Evolution of linear models of surface spectral reflectance with small number of parameters", *J. Opt. Soc. Am. A,* 3, pp. 1673-1683.

Marr, D. (1971) "Simple memory: a theory for archicortex", *Phil. Trans. R. Soc. Lond.* B 262, pp. 23-81.

Mezard, M.; Nadal, J. (1989), *J. Phys. A, 22,* pp. 2191.

Minsky, M.L.; Papert, S. (1969), *Perceptrons: An Essay in Computational Geometry,* MIT Press, Cambridge, MA.

Mitchison, G. (1992), "Axonal trees and cortical architecture", *Trends Neurosci.,* 15, pp. 122-126.

Müller, B.; Reinhard, J. (1991), *Neural Networks: An Introduction,* Berlin: Springer Verlag.

Mumford, D. (1991), "On the computational architecture of the neocortex I", *Biological Cybernetics* 65, pp. 135-145.

Mumford, D. (1992), "On the computational architecture of the neocortex II", *Biological Cybernetics* 66, pp. 241-251.

Nabatovsky, D.; Grossman, T.; Domany, E. (1990), "Learning by CHIR without storing internal representations", *Complex Systems* 4, pp 519.

Nicholls, J. G.; Martin, A. R.; Wallace, B. G. (1992), *From Neuron To Brain, Third Editition,* Sinauer Associates, Inc.

Oja, E. (1992), "Principal components, minor components, and linear neural networks", *Neural Networks,* 5, pp. 927-935.

Peterson, C.; Hartman, E. (1989), "Explorations the mean field theory learning algorithm", *Neural Netw.,* 2, pp. 475-494.

Ripley, B.D. (1996), *Pattern Recognition and Neural Networks,* Cambridge University Press.

Rosenblatt, F. (1962), *Principles of Neurodynamics,* Spartan Books.

Rujan, P.; Machand, M. (1989), *Complex Systems,* 3, pp. 229.

Rumelhart, D.; Hinton, G.; Williams, R. (1986), "Learning internal representations by error propagation", *Parallel Distributed Processing I.,* MIT Press, Cambridge, MA.

Saad, D.; Solla, S.A. (1996), "Dynamics of on-line gradient descent learning for multilayer neural networks", *Advances in Neural Information Processing Systems,* 8, MIT. Press.

Schmidhuber, J. (1992), "Learning factorial codes by predictability minimisation", Neural Computat., 4, pp. 863-879.

Shannon, C.E.; Weaver, W. (1949), *The Mathematical Theory of Communication,* University of Illinois Press.

Shepherd, G.M. (1974), *The Synaptic Organisation of the Brain,* Oxford University Press.

Skaggs W.E., McNaughton B.L. (1996), "Replay of neuronal firing sequences in rat hippocampus during sleep following spatial experience", *Science* 271, pp.1870-3.

Stewart, G.W. (1973), *Introduction to Matrix Computations,* Academic Press.

Stornetta, W.S.; Huberman,B.A. (1987), "An improved three-layer back-propagation algorithm", *Proc. IEEE First International Conference on Neural Networks* (M. Caudill, C. Butler, Eds.).

Tanaka, K.; Saito, H.; Fukada, Y.; Moriya, M. (1991), "Coding visual images of objects in the inferotemporal cortex of the macaque monkey", *J. Neurophysiol.,* 66, pp. 170-189.

Thimm, G.; Moerland, P.; Fiesler, E. (1996), "Interchangeability of learning rate and gain in backpropagation neural networks", *Neural Computation* 8 (2).

Tollenaere, C.B. (1990), "SuperSAB: Fast adaptive back propagation with good scaling properties", *Neural Networks,* 3 (5), pp. 561-573.

Tovee, M. J.; Rolls, E. T.; Treves, A.; Bellis, R. P. (1993), "Information encoding and the response of single neurons in the primate temporal visual cortex", *J. Neurophysiol.,* 70, pp. 650-654.

Tugay, M.; Tanik, Y. (1989), "Properties of the momentum LMS algorithm", *Signal Processing,* 18, pp. 117-127.

Willshaw, J.; Buneman, O. P.; Longuet-Higgins, H. C. (1969), "Non-holographic associative memory", *Nature*, 222, pp. 960.

Willshaw J.; Buckingham J. (1990) "An assessment of Marr's theory of the hippocampus as a temporary memory store", *Phil. Trans. R. Soc. Lond.* B 329, pp. 205-215.

Wasserman, P. (1989), *Neural Computing: Theory and* Practice, Van Nostrand Reinhold, New York.

Zeki, S. (1993), *A Vision of the Brain*, Blackwell Scientific Publications.